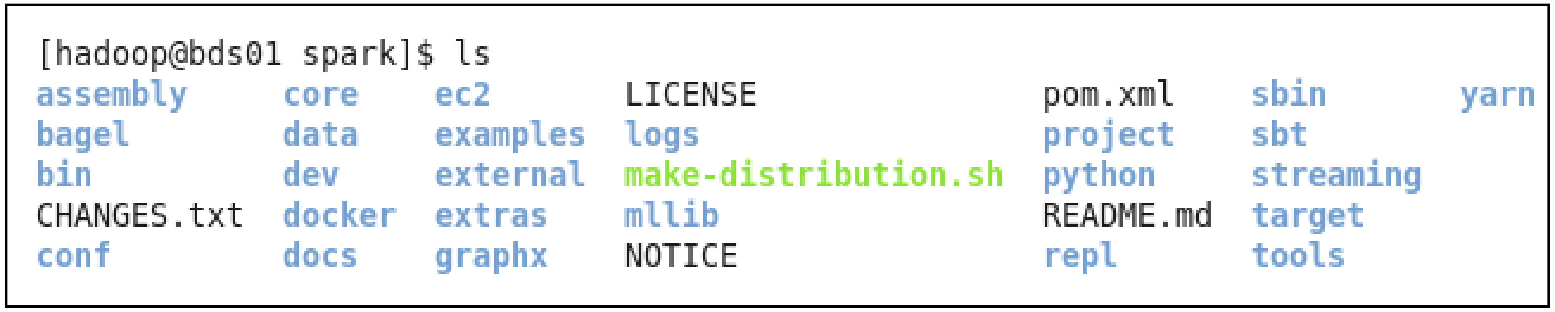
Spark Application 개발

- 챕터 시작 페이지

3장

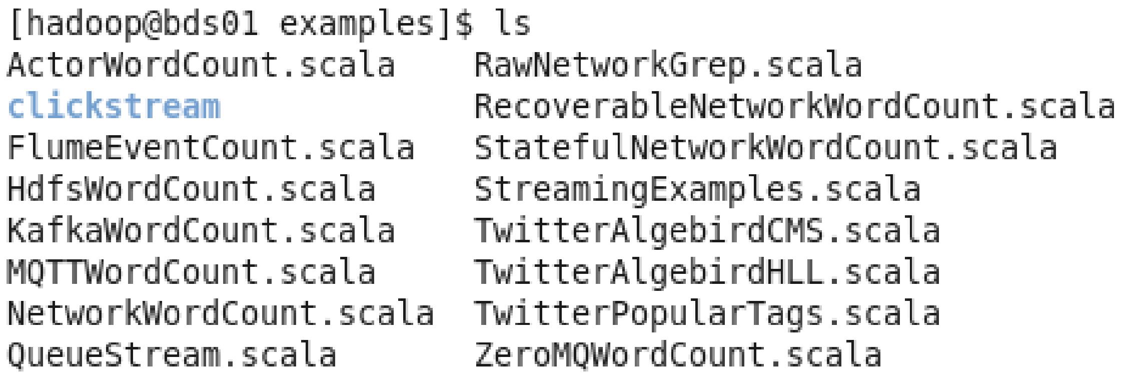
Spark Streaming

- spark/examples



```
[hadoop@bds01 spark]$ ls
assembly      core    ec2       LICENSE                pom.xml    sbin         yarn
bagel         data    examples  logs                   project    sbt
bin           dev     external  make-distribution.sh   python     streaming
CHANGES.txt   docker  extras    mllib                  README.md  target
conf          docs    graphx    NOTICE                 repl       tools
```

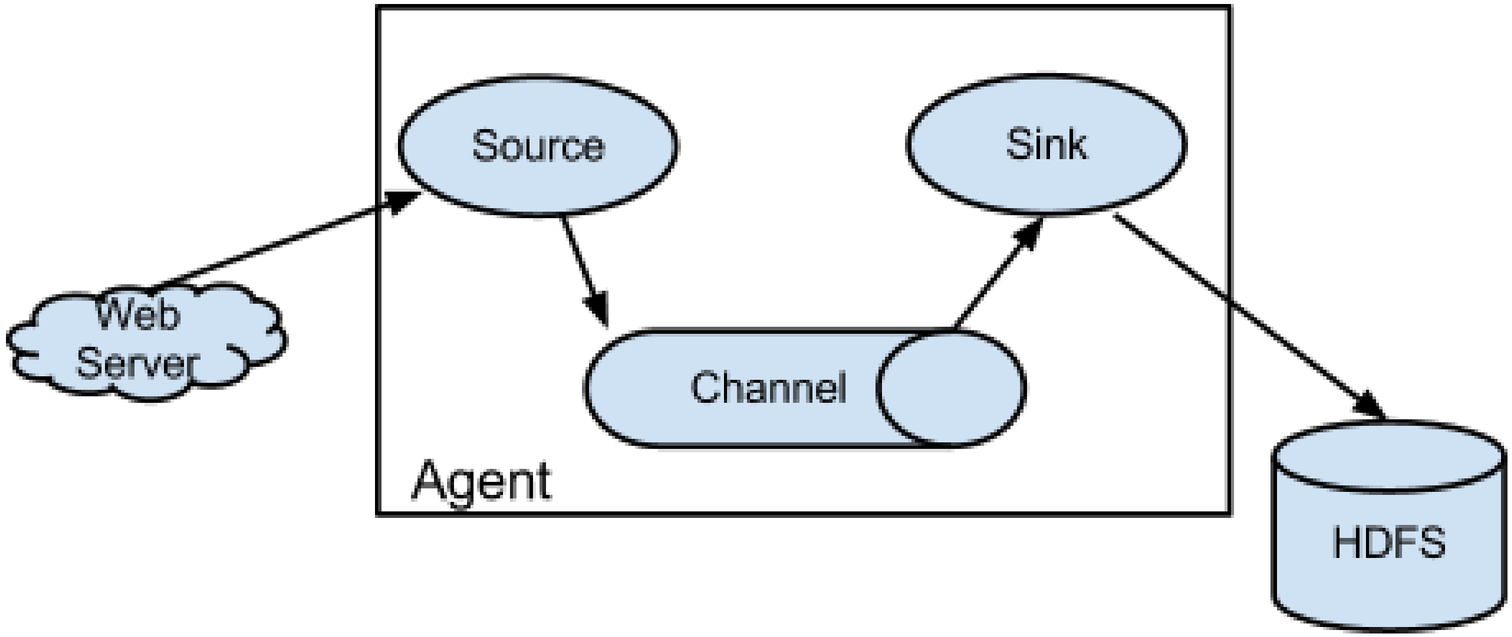
- 예제 코드



```
[hadoop@bds01 examples]$ ls
ActorWordCount.scala     RawNetworkGrep.scala
clickstream              RecoverableNetworkWordCount.scala
FlumeEventCount.scala    StatefulNetworkWordCount.scala
HdfsWordCount.scala      StreamingExamples.scala
KafkaWordCount.scala     TwitterAlgebirdCMS.scala
MQTTWordCount.scala      TwitterAlgebirdHLL.scala
NetworkWordCount.scala   TwitterPopularTags.scala
QueueStream.scala        ZeroMQWordCount.scala
```

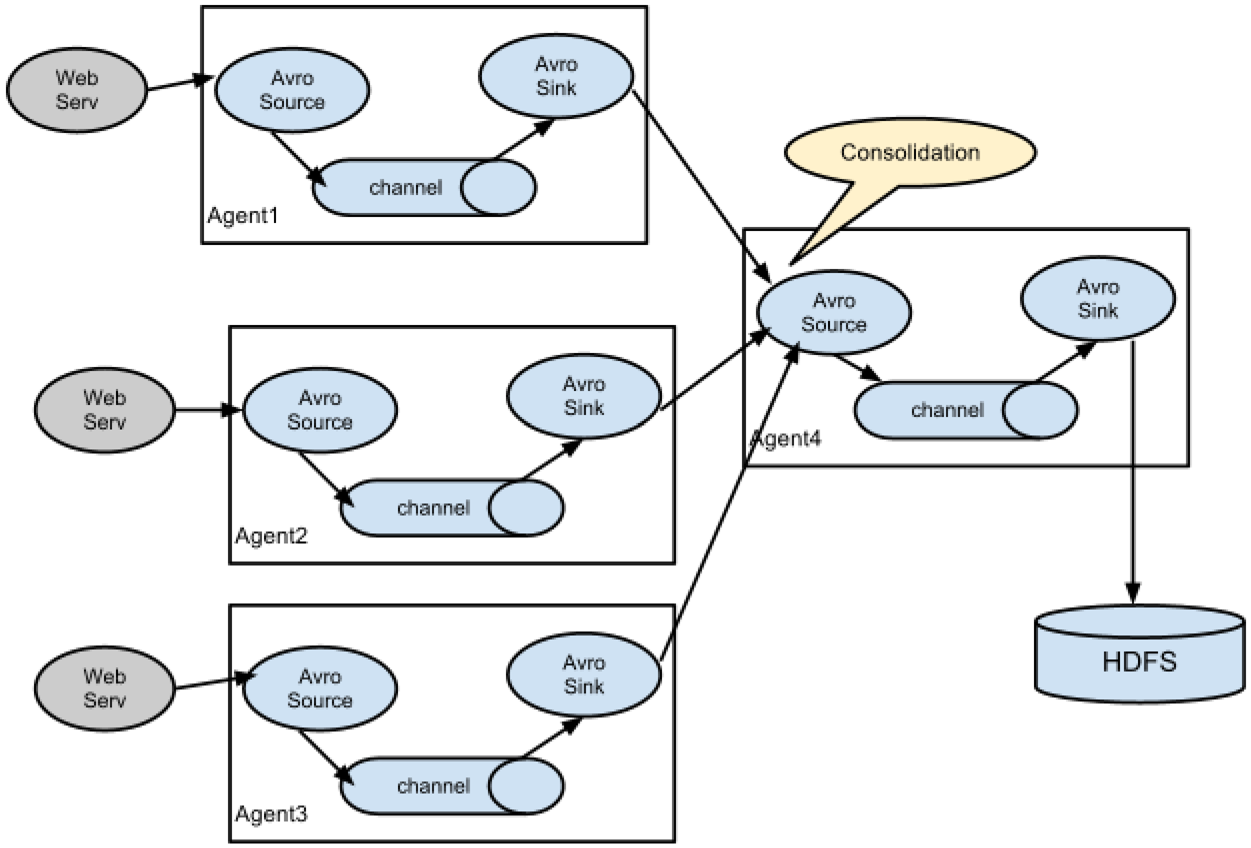
Flume-NG

Apache Flume™



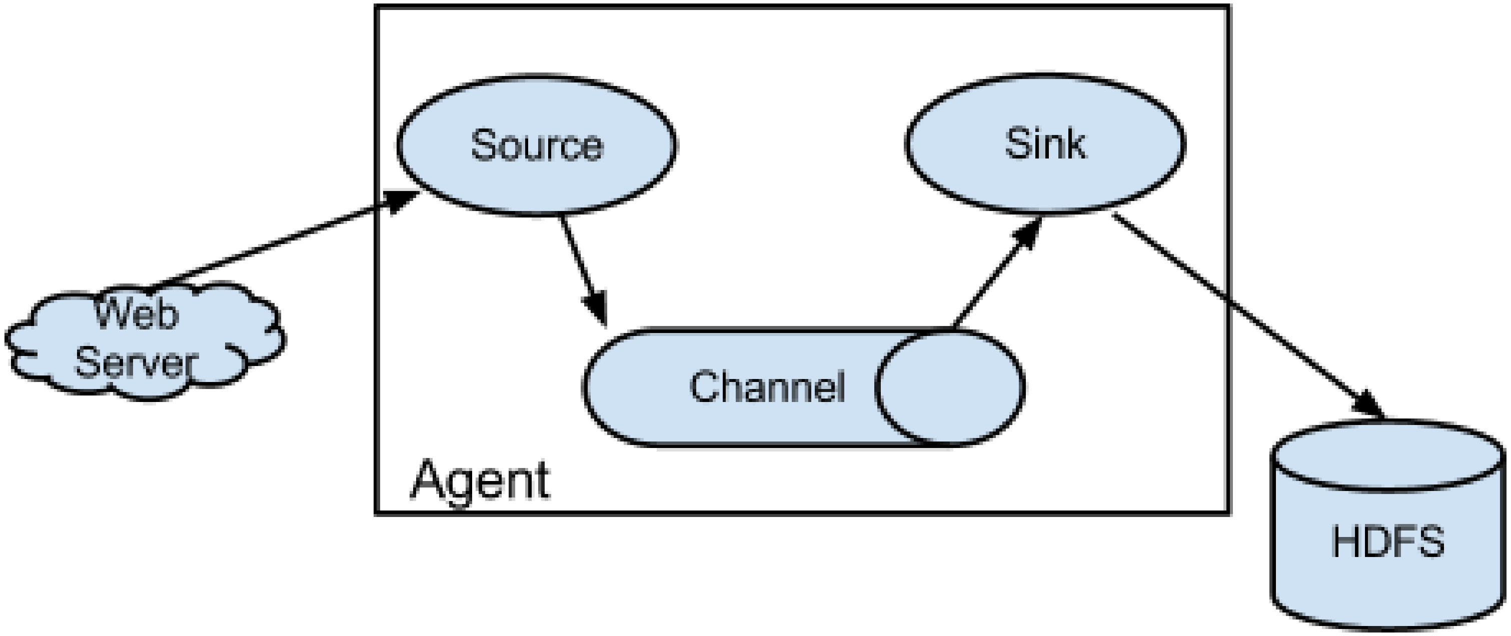
Flume-NG

- 분산 데이터 수집/전송시스템
- 최초 설계 목적은 이벤트나 로그 구조의 데이터를 지속적으로 하둡 HDFS에 저장
- 에이전트
- 확장하여 다양한 분야에 활용 가능



Flume-NG

- 노드 : Flume이 구동되는 머신
- 모든 노드에는 "source"와 "sink"가 있음
  - source 예 : tail –F /var/log/httpd/access_log
  - sink 예 : dfs("hdfs://namenode/log/%{host}%/%y%m%d")
- 데이터플로우 : 노드들의 체인



SAMSUNG SAMSUNG SDS

Flume-NG 설치

Flume 설치

```
$ tar –zxvf apache-flume-1.4.0-bin.tar.gz
$ ln –s apache-flume-1.4.0-bin  apache-flume
cp flume-conf.properties.template flume-conf
```

conf/flume-env.sh 에 다음의 내용 추가

```
JAVA_HOME=/usr/java/java
FLUME_CLASSPATH="/home/hadoop/apache-flume/lib/"
export PATH=$PATH:/home/hadoop/hadoop/bin/
```

Flume-NG 설치

- Starting an agent

```
$ bin/flume-ng agent -n $agent_name -c conf -f conf/flume-conf.properties.template
```

Flume-NG 설치

- conf/flume.conf 에 설정

```
### agent의 각 요소에 이름을 부여
a1.sources = r1
a1.sinks = k1
a1.channels = c1
### source 설정
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444
### sink 설정
a1.sinks.k1.type = logger
### 채널 설정
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100
### source 와 sink 를 채널에 연결
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

Flume-NG 설치

- conf/flume.conf 에 설정

```
# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1
### source 설정
a1.sources.r1.type = exec
a1.sources.r1.command = tail -F /home/hadoop/syslog/a.txt
a1.sources.r1.channels = c1
### sink 를 hdfs로 설정
a1.sinks.k1.type = hdfs
a1.sinks.k1.channel = c1
a1.sinks.k1.hdfs.path = hdfs://hadoop01:9000/user/hadoop/logdata/a.txt
a1.sinks.k1.hdfs.filePrefix = events-
a1.sinks.k1.hdfs.round = true
a1.sinks.k1.hdfs.roundValue = 10
a1.sinks.k1.hdfs.roundUnit = minute
### 채널 설정
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100
### Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

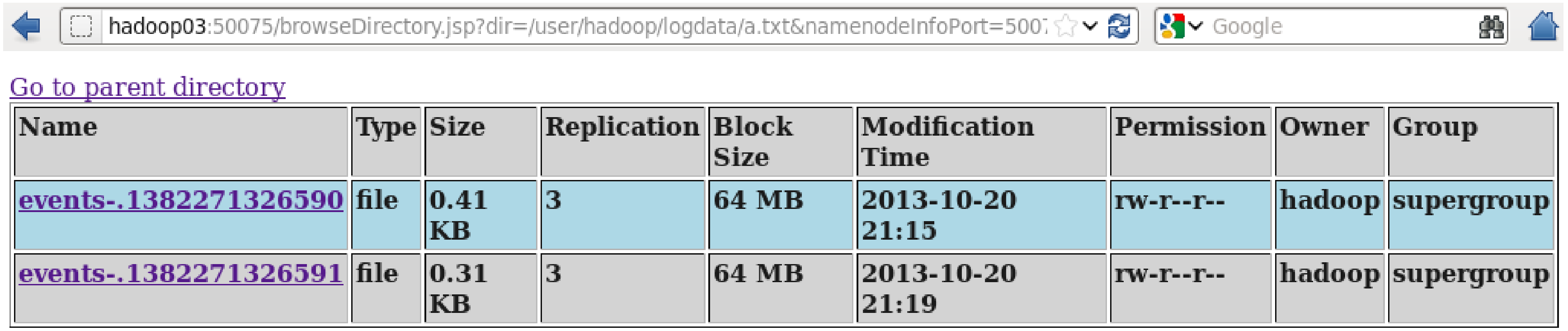
Flume-NG 설치

예제 실행

```
$ bin/flume-ng agent --conf ./conf/ -f conf/flume.conf ₩
    -Dflume.root.logger=DEBUG,console -n agent
```

다음과 같이 실행 로그 출력

```
2013-06-18 14:00:49,784 (hdfs-hdfs-sink-call-runner-0) [INFO -
org.apache.flume.sink.hdfs.BucketWriter.doOpen(BucketWriter.java:189)] Creating
hdfs://localhost:54310/tmp/system.log//FlumeData.1371589249458.tmp
```


hadoop03:50075/browseDirectory.jsp?dir=/user/hadoop/logdata/a.txt&namenodeInfoPort=500
Google

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
events-.1382271326590	file	0.41 KB	3	64 MB	2013-10-20 21:15	rw-r--r--	hadoop	supergroup
events-.1382271326591	file	0.31 KB	3	64 MB	2013-10-20 21:19	rw-r--r--	hadoop	supergroup

Flume-NG 설치

- Flume Sources

Flume Sources	Avro Source
	Thrift Source
	Exec Source
	JMS Source
	Spooling Directory Source
	NetCat Source
	Syslog Sources
	Syslog UDP Source
	HTTP Source
	Legacy Sources
	Custom Source

Flume-NG 설치

HDFS Sink

Name	Default	Description
channel	–	
type	–	The component type name, needs to be `hdfs`
hdfs.path	–	HDFS directory path (eg hdfs://namenode/flume/webdata/)
hdfs.filePrefix	FlumeData	Name prefixed to files created by Flume in hdfs directory
hdfs.fileSuffix	–	Suffix to append to file (eg `.avro` - *NOTE: period is not automatically added*)
hdfs.inUsePrefix	–	Prefix that is used for temporal files that flume actively writes into
hdfs.inUseSuffix	`.tmp`	Suffix that is used for temporal files that flume actively writes into
hdfs.rollInterval	30	Number of seconds to wait before rolling current file (0 = never roll based on time interval)
hdfs.rollSize	1024	File size to trigger roll, in bytes (0: never roll based on file size)
hdfs.rollCount	10	Number of events written to file before it rolled (0 = never roll based on number of events)
hdfs.idleTimeout	0	Timeout after which inactive files get closed (0 = disable automatic closing of idle files)
hdfs.batchSize	100	number of events written to file before it is flushed to HDFS
hdfs.codeC	–	Compression codec. one of following : gzip, bzip2, lzo, lzop, snappy
hdfs.fileType	SequenceFile	File format: currently `SequenceFile`, `DataStream` or `CompressedStream` (1) DataStream will not compress output file and please don't set codeC (2) CompressedStream requires set hdfs.codeC with an available codeC
hdfs.maxOpenFiles	5000	Allow only this number of open files. If this number is exceeded, the oldest file is closed.
hdfs.minBlockReplicas	–	Specify minimum number of replicas per HDFS block. If not specified, it comes from the default Hadoop config in the classpath.
hdfs.writeFormat	–	Format for sequence file records. One of "Text" or "Writable" (the default).
hdfs.callTimeout	10000	Number of milliseconds allowed for HDFS operations, such as open, write, flush, close. This number should be increased if many HDFS timeout operations are occurring.
hdfs.threadsPoolSize	10	Number of threads per HDFS sink for HDFS IO ops (open, write, etc.)
hdfs.rollTimerPoolSize	1	Number of threads per HDFS sink for scheduling timed file rolling

Spark Streaming

- 두 개의 터미널에서 실행
- 첫 번째 터미널

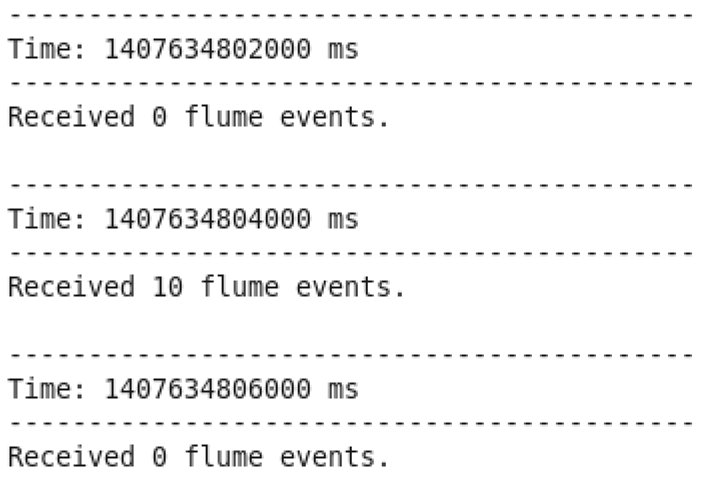
```
bin/run-example
org.apache.spark.streaming.examples.FlumeEventCount   bds02
4545
```

- 두 번째 터미널

```
bin/flume-ng agent -n a1 -c conf -f conf/flume_avro.txt
```

Spark Streaming

결과 확인



```
-------------------------------------------
Time: 1407634802000 ms
-------------------------------------------
Received 0 flume events.

-------------------------------------------
Time: 1407634804000 ms
-------------------------------------------
Received 10 flume events.

-------------------------------------------
Time: 1407634806000 ms
-------------------------------------------
Received 0 flume events.
```

Spark Streaming

- FlumeEventCount

```
package org.apache.spark.streaming.examples

import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming._
import org.apache.spark.streaming.flume._
import org.apache.spark.util.IntParam

object FlumeEventCount {
  def main(args: Array[String]) {
    if (args.length != 3) {
      System.err.println(
        "Usage: FlumeEventCount <master> <host> <port>")
      System.exit(1)
    }

StreamingExamples.setStreamingLogLevels()
```

Spark Streaming

- FlumeEventCount(계속)

```
val Array(master, host, IntParam(port)) = args

val batchInterval = Milliseconds(2000)
// Create the context and set the batch size
val ssc = new StreamingContext(master, "FlumeEventCount",
        batchInterval,  System.getenv("SPARK_HOME"),
         StreamingContext.jarOfClass(this.getClass))

// Create a flume stream
val stream = FlumeUtils.createStream(ssc,
        host,port,StorageLevel.MEMORY_ONLY_SER_2)
```

Spark Streaming

- FlumeEventCount(계속)

```
    // Print out the count of events received events in each batch
    stream.count().map(cnt => "Received " +  cnt +  " flume
            events." ).print()

    ssc.start()
    ssc.awaitTermination()
  }
}
```