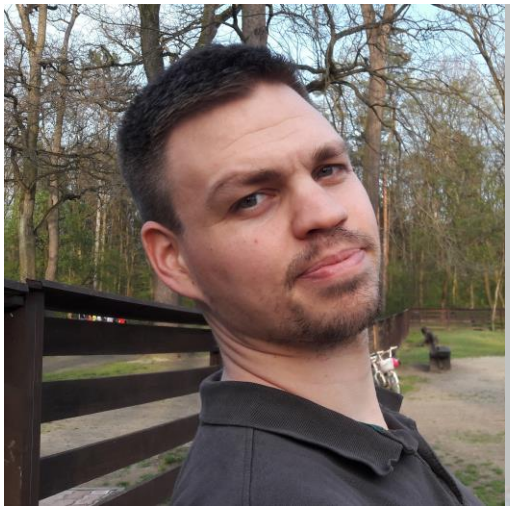



Dr. Hegedűs Péter, Dr. Ferenc Rudolf

Nagyméretű adatbázisok

Jelen tananyag a Szegedi Tudományegyetemen készült az Európai Unió támogatásával.

Projekt azonosító: EFOP-3.4.3-16-2016-00014

Szegedi Tudományegyetem
Cím: 6720 Szeged, Dugonics tér 13.
www.u-szeged.hu
www.szechenyi2020.hu

Apache Flume

Összefoglalás

Az olvasó ebből a kombinált videó és olvasóleckéből gyakorlati tudást szerezhet az Apache Flume rendszer felépítéséről, valamint alapvető felhasználásának módjáról. Áttekintjük az eszköz architekturális felépítését, valamint az alapvetően támogatott komponenseit. Több konkrét példa konfiguráció segítségével bemutatjuk, hogyan használható a Flume különböző adat forrásokkal (hálózat, könyvtrá) és adatnyelőkkel (konzol, HDFS).
A lecke fejezetei:

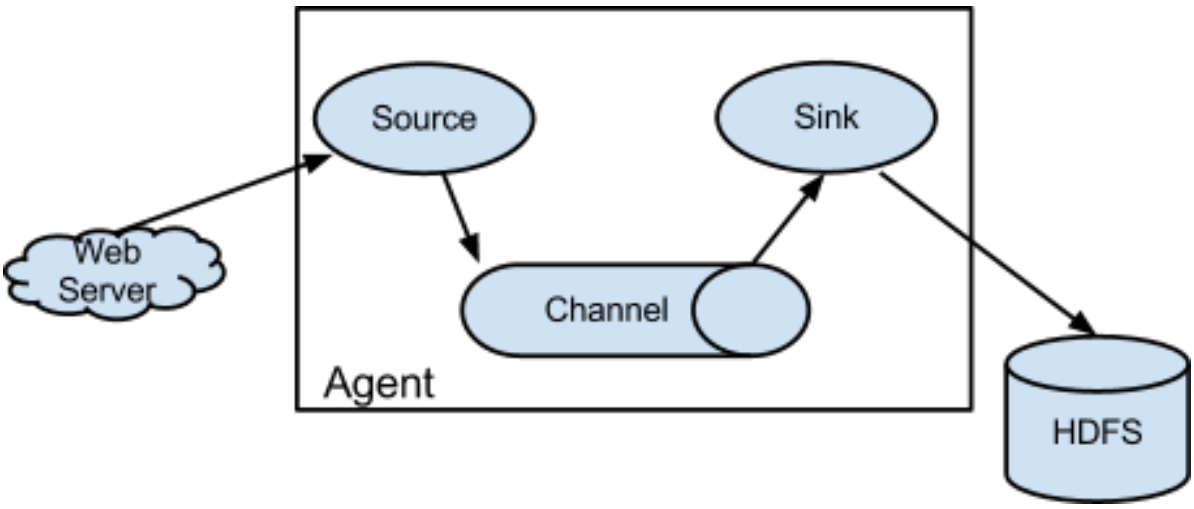
- 1. fejezet: **Az Apache Flume eszköz felépítésének áttekintése, alapvető adatforrások és nyelők típusai (olvasó)**
- 2. fejezet: **Az Apache Flume eszköz telepítése, futtatása és Hello World példa beüzemelése (olvasó)**
- 3. fejezet: **Komoplexebb Apache Flume konfiguráció, HDFS sink bemutatása (videó)**

Téma típusa: **gyakorlati**
Olvasási idő: **35 perc**

📖 1. fejezet
Az Apache Flume eszköz

Az Apache Flume egy elosztott, megbízható és nagy rendelkezésre állású szolgáltatás nagy mennyiségű napló adat gyűjtéséhez, egyesítéséhez és mozgatásához. Az architektúrája egyszerű és rugalmas, amely adatok folyamszerű átvitelére (stream) lett tervezve (lásd lenti ábra). A Flume robosztus, hibatűrő és megbízható számos beépített mechanizmus által. Egyszerű és könnyen bővíthető adatmodellt alkalmaz.

A Flume segítségével adat forrásokat (source) és adatnyelőket (sink) definiálhatunk, az adat forrásokból stream szerűen érkeznek az események, amelyeket egy csatorna az adatnyelőhöz vezet, ami eltárolja azt. A Flume számos adatforrást [1] és adatnyelőt támogat [2], ami persze bővíthető is. Számunkra a legérdekesebb a HDFS adatnyelő, amely tetszőleges stream forrás adatait valós időben tárolni tudja HDFS-re (lásd ábra). A Flume agent működéséről és a konfigurációjának módjáról a következő fejezetek tartalmaz további hasznos információt.

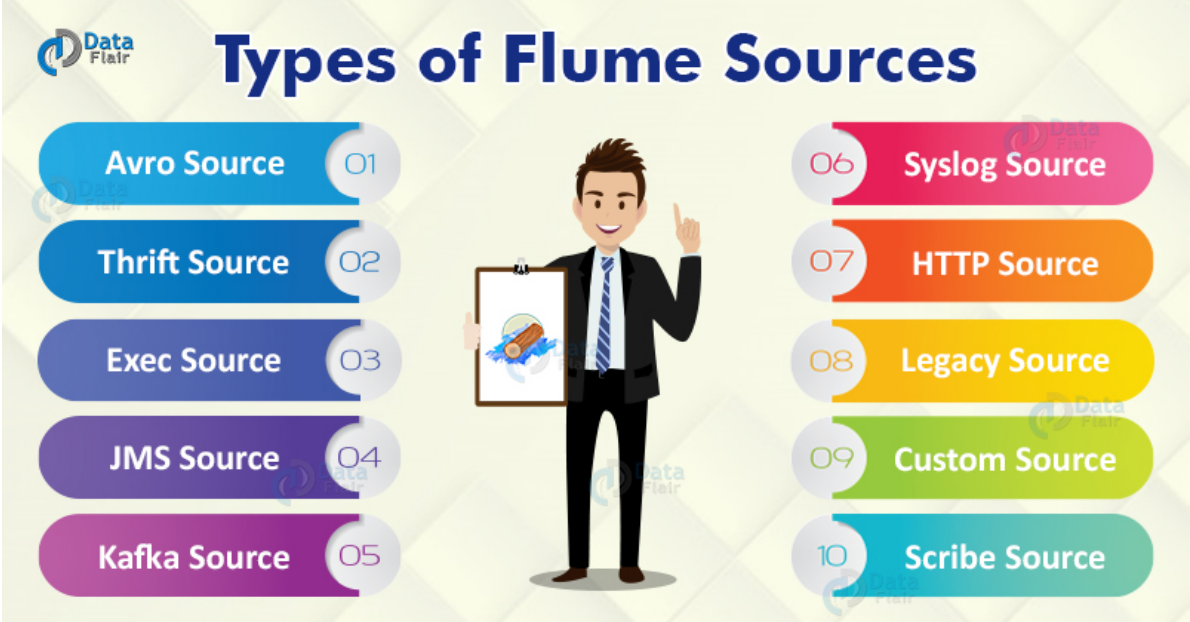


Flume alapvető működése

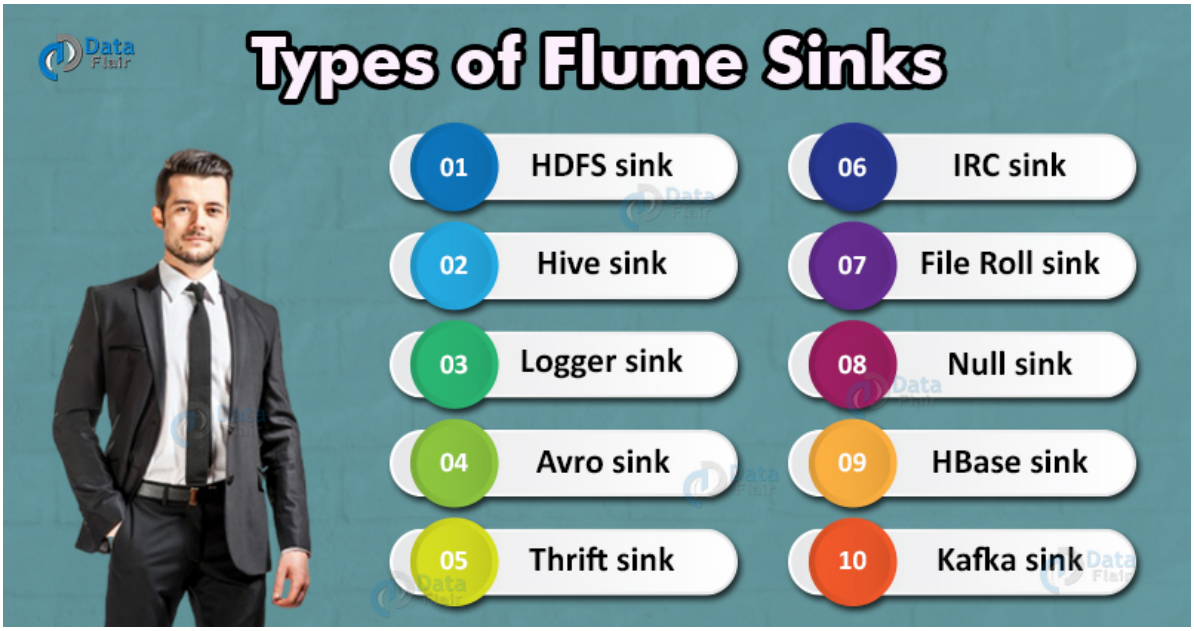
A Flume adatfolyam alapvető eleme az esemény (event), amely nem más, mint egy hasznos adatot tartalmazó bájtsorozat opcionális string attribútumokkal. A Flume ügynök (agent) egy JVM folyamat, amely futtatja a Flume komponenseket, amelyek segítségével a külső forrásból érkező események eljutnak a következő adat nyelőig (egy ugrás, vagy hop).

A Flume adat forrás elkapja és továbbítja azokat az eseményeket, amelyek valamilyen külső forrásból érkeznek hozzá, pl. egy web szervertől. A külső forrás olyan formátumban küldi az eseményeket a Flume adat forráshoz, amit az felismer. Azaz több speciális, előre implementált adat forrás típus és adatnyelő létezik, például egy Apache Avro [3] source az Apache Avro rendszer kliense által küldött, vagy másik vezeték Avro target-jéből származó eseményeket tudja kezelni. Amint egy Flume source fogad egy eseményt, azt azonnal egy vagy több csatornán (channel) továbbítja. A csatorna egy passzív tároló, amely addig tárolja az eseményeket, amíg azt egy Flume sink fel nem dolgozza. A fájl csatorna egy példa a csatornára, ami egy lokális fájlban tárolja az eseményeket. A sink eltávolítja az eseményt a csatornából és egy külső tárolóba helyezi, mint például a HDFS (a Flume HDFS sink segítségével), vagy továbbítja egy másik Flume source felé a következő Flume agent (következő ugrás, next hop) számára az folyamban. Egy ügynökön belül a forrás és nyelő aszinkron módon működnek, ahol a csatorna biztosítja az események pufferelését. A Flume tetszőleges ugrás számú komplex adatfolyamok definiálását is lehetővé teszi. További technikai részletekért tekintsük át a hivatalos dokumentációt [4].

A Flume által támogatott főbb adat források és nyelők típusai az alábbi két ábrán láthatók. Természetesen lehetőség van teljesen egyedi, saját forrás és nyelő implementálására is a megfelelő interfészek megvalósítása által.



> https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/02/types-of-flume-sources.jpg



> https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/02/types-of-flume-sinks.jpg

📖 2. fejezet
Apache Flume Hello World példa

Lássuk hogyan működik gyakorlatban az Apache Flume. Mielőtt telepítenénk a Flume ügynököt és létrehoznánk a megfelelő konfigurációt, indítsuk el a Docker alapú Hadoop klasztert, amit a `3g_BigData-hadoop-SPOC` olvasólecke 1. fejezetében ismertettünk. Ezután lépjünk be a `namenode` container-re egy bash-t futtatva:

```
$ docker exec -it namenode bash
```

Erre a csomópontra fogjuk telepíteni az Apache Flume ügynököt kényelmi okok miatt (a HDFS fájlrendszert localhost-on keresztül elérjük), de más, tetszőleges hálózaton belüli gépre is telepíthető lenne. A telepítés menete:

1. Töltsük le az Apache Flume-ot a következő URL-ről: http://xenia.sote.hu/ftp/mirrors/www.apache.org/flume/1.9.0/apache-flume-1.9.0-bin.tar.gz

```
root@b055549cdaca:/# curl
http://xenia.sote.hu/ftp/mirrors/www.apache.org/flume/1.9.0/apache-flume-1.9.0-
bin.tar.gz -o apache-flume-1.9.0-bin.tar.gz
```

2. Csomagoljuk ki a letöltött Flume disztribúciót

```
root@b055549cdaca:/# tar xzf apache-flume-1.9.0-bin.tar.gz
```

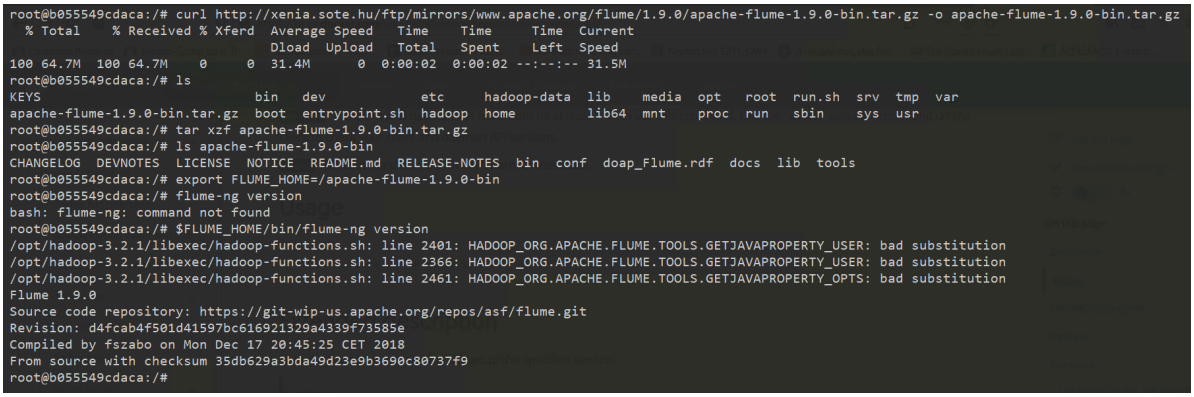
3. Állítsuk be a `FLUME_HOME` környezeti változót

```
root@b055549cdaca:/# export FLUME_HOME=/apache-flume-1.9.0-bin
```

4. Teszteljük, hogy a Flume agent megfelelően működik a következő parancs segítségével

```
root@b055549cdaca:/# $FLUME_HOME/bin/flume-ng version
```

Amennyiben látjuk a képernyőre kiírva, hogy `Flume 1.9.0`, a telepítés sikeres (a folyamatot az alábbi ábra mutatja).



Sikeres telepítés után hozzuk létre az első Flume adat folyamunkat. Ehhez egy egyszerű ügynököt konfigurálunk be, amelyik a hálózat egy bizonyos portján figyeli az adatokat (netcat source), és minden ott beérkező üzenetet (event) a képernyőre ír ki egy konzol logger segítségével (logger sink). Az átviteli csatorna memóriában tárolja az eseményeket. Az `example.conf` tartalma legyen a következő (saját gépen tetszőleges szövegszerkesztővel készítsük el, majd másoljuk át a docker container-be).

```
# example.conf: A single-node Flume configuration

# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# Describe the sink
a1.sinks.k1.type = logger

# Use a channel which buffers events in memory
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
```

Miután elkészítettük a specifikációnak megfelelő Flume adatfolyam leírót, másoljuk át a fájlt a `namenode` docker container-be a megfelelő Flume konfigurációs könyvtárba:

```
$ docker cp example.conf namenode:/apache-flume-1.9.0-bin/conf/example.conf
```

Ezután készen állunk arra, hogy elindítsuk az ügynök szolgáltatást (Flume agent service) a következő utasítással:

```
root@b055549cdaca:/apache-flume-1.9.0-bin# bin/flume-ng agent --conf conf --
conf-file conf/example.conf --name a1 -Dflume.root.logger=INFO,console
```

A paraméterekkel megadjuk az agent nevét és a konfiguráció helyét, valamint mivel nincs log4j konfigurációnk, egy JVM paraméterrel beállítjuk, hogy konzolra történjen a log üzenetek kiírása (azaz az eventek kiírása). Az agent az elindulás után a `namnode` 44444-es portján figyel, és minden odaérkező üzenetet (event) elkap, majd acsatornán keresztül a sink felé továbbít, ami log üzenetben kiírja a kapott event-et. Tesztelés gyanánt nyissunk egy újabb terminál ablakot, és kapcsolódjunk a `namenode` container-hez a fenti módon, majd küldjünk egy `Hello Wolrd!` üzenetet a `localhost:44444`-es címre `netcat` program segítségével:

```
$ docker exec -it namenode bash
root@b055549cdaca:/# netcat localhost 44444
Hello World!
OK
```

Ennek hatására az ügynököt futtató terminálban azt kell látnunk, hogy a konzolra log üzenetben kiíródott a `Hello World!` üzenet!

```
2020-08-16 16:06:12,261 (conf-file-poller-0) [INFO - org.apache.flume.conf.FlumeConfiguration$AgentConfiguration.addComponentConfig(FlumeConfiguration.java:12
03)] Processing:k1
2020-08-16 16:06:12,261 (conf-file-poller-0) [INFO - org.apache.flume.conf.FlumeConfiguration$AgentConfiguration.addComponentConfig(FlumeConfiguration.java:12
03)] Processing:r1
2020-08-16 16:06:12,261 (conf-file-poller-0) [INFO - org.apache.flume.conf.FlumeConfiguration$AgentConfiguration.addComponentConfig(FlumeConfiguration.java:12
03)] Processing:k1
2020-08-16 16:06:12,262 (conf-file-poller-0) [WARN - org.apache.flume.conf.FlumeConfiguration$AgentConfiguration.validateConfigFilterSet(FlumeConfiguration.ja
va:623)] Agent configuration for 'a1' has no configfilters.
2020-08-16 16:06:12,279 (conf-file-poller-0) [INFO - org.apache.flume.conf.FlumeConfiguration.validateConfiguration(FlumeConfiguration.java:163)] Post-validat
ion flume configuration contains configuration for agents: [a1]
2020-08-16 16:06:12,280 (conf-file-poller-0) [INFO - org.apache.flume.node.AbstractConfigurationProvider.loadChannels(AbstractConfigurationProvider.java:151)]
 Creating channels
2020-08-16 16:06:12,285 (conf-file-poller-0) [INFO - org.apache.flume.channel.DefaultChannelFactory.create(DefaultChannelFactory.java:42)] Creating instance o
f channel c1 type memory
2020-08-16 16:06:12,290 (conf-file-poller-0) [INFO - org.apache.flume.node.AbstractConfigurationProvider.loadChannels(AbstractConfigurationProvider.java:205)]
 Created channel c1
2020-08-16 16:06:12,291 (conf-file-poller-0) [INFO - org.apache.flume.source.DefaultSourceFactory.create(DefaultSourceFactory.java:41)] Creating instance of s
ource r1, type netcat
2020-08-16 16:06:12,296 (conf-file-poller-0) [INFO - org.apache.flume.sink.DefaultSinkFactory.create(DefaultSinkFactory.java:42)] Creating instance of sink: k
1, type: logger
2020-08-16 16:06:12,298 (conf-file-poller-0) [INFO - org.apache.flume.node.AbstractConfigurationProvider.getConfiguration(AbstractConfigurationProvider.java:1
20)] Channel c1 connected to [r1, k1]
2020-08-16 16:06:12,303 (conf-file-poller-0) [INFO - org.apache.flume.node.Application.startAllComponents(Application.java:162)] Starting new configuration:{
sourceRunners:{r1=EventDrivenSourceRunner: { source:org.apache.flume.source.NetcatSource{name:r1,state:IDLE} }} sinkRunners:{k1=SinkRunner: { policy:org.apach
e.flume.sink.DefaultSinkProcessor@331d3c25 counterGroup:{ name:null counters:{} } }} channels:{c1=org.apache.flume.channel.MemoryChannel{name: c1}} }
2020-08-16 16:06:12,306 (conf-file-poller-0) [INFO - org.apache.flume.node.Application.startAllComponents(Application.java:169)] Starting Channel c1
2020-08-16 16:06:12,349 (lifecycleSupervisor-1-0) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.register(MonitoredCounterGroup.java:119)] Mon
itored counter group for type: CHANNEL, name: c1: Successfully registered new MBean.
2020-08-16 16:06:12,349 (lifecycleSupervisor-1-0) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.start(MonitoredCounterGroup.java:95)] Compone
nt type: CHANNEL, name: c1 started
2020-08-16 16:06:12,350 (conf-file-poller-0) [INFO - org.apache.flume.node.Application.startAllComponents(Application.java:196)] Starting Sink k1
2020-08-16 16:06:12,350 (conf-file-poller-0) [INFO - org.apache.flume.node.Application.startAllComponents(Application.java:207)] Starting Source r1
2020-08-16 16:06:12,352 (lifecycleSupervisor-1-2) [INFO - org.apache.flume.source.NetcatSource.start(NetcatSource.java:155)] Source starting
2020-08-16 16:06:12,362 (lifecycleSupervisor-1-2) [INFO - org.apache.flume.source.NetcatSource.start(NetcatSource.java:166)] Created serverSocket:sun.nio.ch.S
erverSocketChannelImpl[/127.0.0.1:44444]
2020-08-16 16:10:58,372 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:95)] Event: { headers
:{} body: 48 65 6C 6C 6F 20 57 6F 72 6C 64 21            Hello World! }
```

3. fejezet
Adat stream mentése HDFS-re

Ebben a fejezetben a HDFS sink-re koncentrálunk. Az alábbi videó lecke bemutatja, hogyan kell módosítani a 2. fejezetben összeállított Flume konfigurációt ahhoz, hogy a forráson érkező eseményeket ne egyszerűen kiírassuk logger-rel, hanem egyenesen HDFS-re tároljuk. A videóban használt konfigurációs állomány (`example-hdfs.conf`) az alábbi:

```
# example-hdfs.conf: A single-node Flume configuration

# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
```

```
a1.sources.r1.port = 44444

# Describe the sink HDFS
a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path = hdfs://localhost:9000/flume/testdata
a1.sinks.k1.hdfs.rollInterval = 0
a1.sinks.k1.hdfs.fileType = DataStream

# Use a channel which buffers events in memory
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
```

video/4g_BigData-hdfs-sqoop-flume-SPOC/flume-hdfs.mp4

További feladatok

1. Készítsünk egy olyan Flume konfigurációt, ahol az adatok forrása egy könyvtár (`spool dir`), ahonnan a fájlokat a Flume folyam HDFS-re tárolja! ⭐
2. Módosítsuk úgy a fenti konfigurációt, hogy az adatok a syslog-ból érkezzenek! ⭐
3. Készítsünk egy összetett Flume folyamot, ami két ügynököt kapcsol össze: **A1**: `syslog -> http`, **A2**: `http -> hdfs`! ⭐⭐⭐

Referenciák

[1] https://data-flair.training/blogs/flume-source/

[2] https://data-flair.training/blogs/flume-sink/

[3] https://avro.apache.org/

[4] https://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html#the-plugins-d-directory