# 新手教程

Flume NG 旨在比Flume OG更简单、更小，容易部署。为了达到这个目标，我们不承诺保持Flume NG对Flume OG的向后兼容性。 当前，我们恳求有关Flume NG正确性、易用性和与其它系统集成可能性测试的反馈。

### 有什么变化？

虽然许多原来的概念相同，但Flume NG(下一代Flume)与Flume OG(原Flume)在实现上存很大差异。如果你已经熟悉Flume，下面有一些需要你了解的：

* 仍有source和sink并且它们仍做相同的事。它们现在通过channel相连。
* Channel可插拨并可配置持久性。Flume NG发布了一个内存channel，用于快速，非持久事件交付和一个基于文件的channel用于可持久交付。
* 不再有逻辑结点与物理结点。我们称物理结点为agent，agent可以运行零个或多个source和sink。
* 不再有master和ZooKeeper依赖。Flume运行基于一个简单的文件配置系统。
* 所有的一切都是插件，有些插件面对最终用户，有些插件面对工具与系统开发者。可插拨的组件包括channel, source, sink, interceptor, sink processor, 和event serializer。

请通过[JIRAs](https://issues.apache.org/jira/browse/FLUME)新建或投票你认为重要的特性。

## 开始 Flume NG

Flume网站提供了源代码tar和二进制文件的下载，如果你不打算修改flume，

二进制文件是更好的选择。

构建：

从源码构建flume NG，你需要git，sun JDK1.6，Apache Maven 3.x，

大约90MB的磁盘空间和网络连接。

1. 检出源代码

$ git clone https://git-wip-us.apache.org/repos/asf/flume.git flume

$ cd flume

$ git checkout trunk

2. 编译项目

maven编译需要比默认配置更多的内存。我们建议你设置以下Maven选项：

export MAVEN\_OPTS="-Xms512m -Xmx1024M -XX:PermSize=256M -XX:MaxPermSize=512M"

# 构建代码并运行测试(注意：使用MVN install，而不是MVN package，因为我们每天部署Jenkins SNAPSHOT Jar包，并且flume是一个多工程项目)

$ MVN install

# ...或跳过测试

$ MVN install DskipTests

(请注意，flume编译需要Google Protocol Buffers。您下载并安装它。)

编译后会产生flume-ng-dist/target ，它们分别是：

apache-flume-ng-dist-1.4.0-SNAPSHOT-bin.tar.gz -二进制分发包，可直接运行。

apache-flume-ng-dist-1.4.0-SNAPSHOT-src.tar.gz -源代码分发包。

如果你只是想运行flume，bin版本更适合你。把它复制出来，解压，你可以准备去运行了。

$ CP flume-ng-dist/target/apache-flume-1.4.0-SNAPSHOT-bin.tar.gz

$ tar -zxvf apache-flume-1.4.0-SNAPSHOT-bin.tar.gz

$ cd apache-flume-1.4.0-SNAPSHOT-bin

3. 基于提供的模板文件去创建你自己的属性文件(或者自己从头开始)

$ cp conf/flume-conf.properties.template conf/flume.conf

4. 这一步是可选的，创建基于模板文件(或从头开始创建)flume-env.sh文件。Flume-NG的可执行文件和源文件名为"flume-env.sh"

在conf目录中通过命令行--conf/-c指定 。使用flume-env.sh可以指定调试或分析选项，通过JAVA\_OPTS开发自己的定制Flume NG组件，比如说sources和sinks。

$ cp conf/flume-env.sh.template conf/flume-env.sh

5. 配置和运行Flume NG

当你配置了Flume NG，具体如何配置见下文，你能够运行它通过命令bin/flume-ng，这个脚本有大量的参数和模式。

## Configuration

Flume使用Java属性文件进行配置。代理运行时使用-f<file>选项指定配置文件。这个文件可以放在任何地方，但建议放在conf目录下。

让我们开始学习一个简单的实例，复制这段内容到conf/flume.conf

|  |
| --- |
| # Define a memory channel called ch1 on agent1  agent1.channels.ch1.type = memory    # Define an Avro source called avro-source1 on agent1 and tell it  # to bind to0.0.0.0:41414. Connect it to channel ch1.  agent1.sources.avro-source1.channels = ch1  agent1.sources.avro-source1.type = avro  agent1.sources.avro-source1.bind =0.0.0.0  agent1.sources.avro-source1.port =41414    # Define a logger sink that simply logs all events it receives  # and connect it to the other end of the same channel.  agent1.sinks.log-sink1.channel = ch1  agent1.sinks.log-sink1.type = logger    # Finally, now that we've defined all of our components, tell  # agent1 which ones we want to activate.  agent1.channels = ch1  agent1.sources = avro-source1  agent1.sinks = log-sink1 |

上面的实例创建了一个内存通道（一个不可靠或进最大努力的通道）、一个 Avro RPC source、一个日志存储器，并把他们关联到一起。 Avro source接收的所有事件路由到通道ch1，并且传输到日志接收器。重要的一点，定义组件是配置Flume的首要事情；他们必须使用<agent>.channels、<agent>.sources,和sections来激活。多个源、收集器、通道配置时使用空格分隔。

更多的细节，请参考javadoc

org.apache.flume.conf.properties.PropertiesFileConfigurationProvider class.

如下是这时继承的源、收集器、渠道列表。每个组件有它必须、可选的属性，请看javadoc。

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Type** | **Description** | **Implementation Class** |
| Channel | memory | In-memory, fast, non-durable event transport | MemoryChannel |
| Channel | file | A channel for reading, writing, mapping, and manipulating a file | FileChannel |
| Channel | jdbc | JDBC-based, durable event transport (Derby-based) | JDBCChannel |
| Channel | recoverablememory | A durable channel implementation that uses the local file system for its storage | RecoverableMemoryChannel |
| Channel | org.apache.flume.channel.PseudoTxnMemoryChannel | Mainly for testing purposes. Not meant for production use. | PseudoTxnMemoryChannel |
| Channel | *(custom type as FQCN)* | Your own Channel impl. | *(custom FQCN)* |
| Source | avro | Avro Netty RPC event source | AvroSource |
| Source | exec | Execute a long-lived Unix process and read from stdout | ExecSource |
| Source | netcat | Netcat style TCP event source | NetcatSource |
| Source | seq | Monotonically incrementing sequence generator event source | SequenceGeneratorSource |
| Source | org.apache.flume.source.StressSource | Mainly for testing purposes. Not meant for production use. Serves as a continuous source of events where each event has the same payload. The payload consists of some number of bytes (specified by *size* property, defaults to 500) where each byte has the signed value Byte.MAX\_VALUE (0x7F, or 127). | org.apache.flume.source.StressSource |
| Source | syslogtcp |  | SyslogTcpSource |
| Source | syslogudp |  | SyslogUDPSource |
| Source | org.apache.flume.source.avroLegacy.AvroLegacySource |  | AvroLegacySource |
| Source | org.apache.flume.source.thriftLegacy.ThriftLegacySource |  | ThriftLegacySource |
| Source | org.apache.flume.source.scribe.ScribeSource |  | ScribeSource |
| Source | *(custom type as FQCN)* | Your own Source impl. | *(custom FQCN)* |
| Sink | hdfs | Writes all events received to HDFS (with support for rolling, bucketing, HDFS-200 append, and more) | HDFSEventSink |
| Sink | org.apache.flume.sink.hbase.HBaseSink | A simple sink that reads events from a channel and writes them to HBase. | org.apache.flume.sink.hbase.HBaseSink |
| Sink | org.apache.flume.sink.hbase.AsyncHBaseSink |  | org.apache.flume.sink.hbase.AsyncHBaseSink |
| Sink | logger | Log events at INFO level via configured logging subsystem (log4j by default) | LoggerSink |
| Sink | avro | Sink that invokes a pre-defined Avro protocol method for all events it receives (when paired with an avro source, forms tiered collection) | AvroSink |
| Sink | file\_roll |  | RollingFileSink |
| Sink | irc |  | IRCSink |
| Sink | null | /dev/null for Flume - blackhole all events received | NullSink |
| Sink | *(custom type as FQCN)* | Your own Sink impl. | *(custom FQCN)* |
| ChannelSelector | replicating |  | ReplicatingChannelSelector |
| ChannelSelector | multiplexing |  | MultiplexingChannelSelector |
| ChannelSelector | (custom type) | Your own ChannelSelector impl. | *(custom FQCN)* |
| SinkProcessor | default |  | DefaultSinkProcessor |
| SinkProcessor | failover |  | FailoverSinkProcessor |
| SinkProcessor | load\_balance | Provides the ability to load-balance flow over multiple sinks. | LoadBalancingSinkProcessor |
| SinkProcessor | *(custom type as FQCN)* | Your own SinkProcessor impl. | *(custom FQCN)* |
| Interceptor$Builder | host |  | HostInterceptor$Builder |
| Interceptor$Builder | timestamp | TimestampInterceptor | TimestampInterceptor$Builder |
| Interceptor$Builder | static |  | StaticInterceptor$Builder |
| Interceptor$Builder | regex\_filter |  | RegexFilteringInterceptor$Builder |
| Interceptor$Builder | *(custom type as FQCN)* | Your own Interceptor$Builder impl. | *(custom FQCN)* |
| EventSerializer$Builder | text |  | BodyTextEventSerializer$Builder |
| EventSerializer$Builder | avro\_event |  | FlumeEventAvroEventSerializer$Builder |
| EventSerializer | org.apache.flume.sink.hbase.SimpleHbaseEventSerializer |  | SimpleHbaseEventSerializer |
| EventSerializer | org.apache.flume.sink.hbase.SimpleAsyncHbaseEventSerializer |  | SimpleAsyncHbaseEventSerializer |
| EventSerializer | org.apache.flume.sink.hbase.RegexHbaseEventSerializer |  | RegexHbaseEventSerializer |
| HbaseEventSerializer | Custom implementation of serializer for HBaseSink.  *(custom type as FQCN)* | Your own HbaseEventSerializer impl. | *(custom FQCN)* |
| AsyncHbaseEventSerializer | Custom implementation of serializer for AsyncHbase sink.  *(custom type as FQCN)* | Your own AsyncHbaseEventSerializer impl. | *(custom FQCN)* |
| EventSerializer$Builder | Custom implementation of serializer for all sinks except for HBaseSink and AsyncHBaseSink.  *(custom type as FQCN)* | Your own EventSerializer$Builder impl. | *(custom FQCN)* |

可以通过 flume NG可执行文件执行Flume NG代理，或 Avro客户端来进行测试。不论什么时候，你需要一个指定的命令（如代理或Avro的客户端）和conf目录(--conf <conf dir>)，其他的选项都是特定的命令。

使用 flume.conf 启动flume服务端

bin/flume-ng agent --conf ./conf/ -f conf/flume.conf -Dflume.root.logger=DEBUG,console -n agent1

注意，代理名称通过 -n agent1指定，并且必须与 -f conf/flume.conf中提供的匹配。

输出信息如下：

$ bin/flume-ng agent --conf conf/ -f conf/flume.conf -n agent1

2012-03-1616:36:11,918(main) [INFO - org.apache.flume.lifecycle.LifecycleSupervisor.start(LifecycleSupervisor.java:58)] Starting lifecycle supervisor1

2012-03-1616:36:11,921(main) [INFO - org.apache.flume.node.FlumeNode.start(FlumeNode.java:54)] Flume node starting - agent1

2012-03-1616:36:11,926(lifecycleSupervisor-1-0) [INFO - org.apache.flume.node.nodemanager.DefaultLogicalNodeManager.start(DefaultLogicalNodeManager.java:110)] Node manager starting

2012-03-1616:36:11,928(lifecycleSupervisor-1-0) [INFO - org.apache.flume.lifecycle.LifecycleSupervisor.start(LifecycleSupervisor.java:58)] Starting lifecycle supervisor10

2012-03-1616:36:11,929(lifecycleSupervisor-1-0) [DEBUG - org.apache.flume.node.nodemanager.DefaultLogicalNodeManager.start(DefaultLogicalNodeManager.java:114)] Node manager started

2012-03-1616:36:11,926(lifecycleSupervisor-1-1) [INFO - org.apache.flume.conf.file.AbstractFileConfigurationProvider.start(AbstractFileConfigurationProvider.java:67)] Configuration provider starting

2012-03-1616:36:11,930(lifecycleSupervisor-1-1) [DEBUG - org.apache.flume.conf.file.AbstractFileConfigurationProvider.start(AbstractFileConfigurationProvider.java:87)] Configuration provider started

2012-03-1616:36:11,930(conf-file-poller-0) [DEBUG - org.apache.flume.conf.file.AbstractFileConfigurationProvider$FileWatcherRunnable.run(AbstractFileConfigurationProvider.java:189)] Checking file:conf/flume.confforchanges

2012-03-1616:36:11,931(conf-file-poller-0) [INFO - org.apache.flume.conf.file.AbstractFileConfigurationProvider$FileWatcherRunnable.run(AbstractFileConfigurationProvider.java:196)] Reloading configuration file:conf/flume.conf

2012-03-1616:36:11,936(conf-file-poller-0) [DEBUG - org.apache.flume.conf.properties.FlumeConfiguration$AgentConfiguration.isValid(FlumeConfiguration.java:225)] Starting validation of configurationforagent: agent1, initial-configuration: AgentConfiguration[agent1]

SOURCES: {avro-source1=ComponentConfiguration[avro-source1]

CONFIG: {port=41414, channels=ch1, type=avro, bind=0.0.0.0}

RUNNER: ComponentConfiguration[runner]

CONFIG: {}

}

CHANNELS: {ch1=ComponentConfiguration[ch1]

CONFIG: {type=memory}

}

SINKS: {log-sink1=ComponentConfiguration[log-sink1]

CONFIG: {type=logger, channel=ch1}

RUNNER: ComponentConfiguration[runner]

CONFIG: {}

}

2012-03-1616:36:11,936(conf-file-poller-0) [INFO - org.apache.flume.conf.properties.FlumeConfiguration.validateConfiguration(FlumeConfiguration.java:119)] Post-validation flume configuration contains configuation foragents: [agent1]

2012-03-1616:36:11,937(conf-file-poller-0) [DEBUG - org.apache.flume.channel.DefaultChannelFactory.create(DefaultChannelFactory.java:67)] Creating instance of channel ch1 type memory

2012-03-1616:36:11,944(conf-file-poller-0) [DEBUG - org.apache.flume.source.DefaultSourceFactory.create(DefaultSourceFactory.java:73)] Creating instance of source avro-source1, type avro

2012-03-1616:36:11,957(conf-file-poller-0) [INFO - org.apache.flume.sink.DefaultSinkFactory.create(DefaultSinkFactory.java:69)] Creating instance of sink log-sink1 typelogger

2012-03-1616:36:11,963(conf-file-poller-0) [INFO - org.apache.flume.node.nodemanager.DefaultLogicalNodeManager.onNodeConfigurationChanged(DefaultLogicalNodeManager.java:52)] Node configuration change:{ sourceRunners:{avro-source1=EventDrivenSourceRunner: { source:AvroSource: { bindAddress:0.0.0.0port:41414} }} sinkRunners:{log-sink1=SinkRunner: { policy:org.apache.flume.sink.DefaultSinkProcessor@79f6f296counterGroup:{ name:nullcounters:{} } }} channels:{ch1=org.apache.flume.channel.MemoryChannel@43b09468} }

2012-03-1616:36:11,974(lifecycleSupervisor-1-1) [INFO - org.apache.flume.source.AvroSource.start(AvroSource.java:122)] Avro source starting:AvroSource: { bindAddress:0.0.0.0port:41414}

2012-03-1616:36:11,975(Thread-1) [DEBUG - org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:123)] Polling sink runner starting

2012-03-1616:36:12,352(lifecycleSupervisor-1-1) [DEBUG - org.apache.flume.source.AvroSource.start(AvroSource.java:132)] Avro source started

**flume-ng 全局选项**

|  |  |
| --- | --- |
| **Option** | **Description** |
| --conf,-c <conf> | Use configs in <conf> directory |
| --classpath,-C <cp> | Append to the classpath |
| --dryrun,-d | Do not actually start Flume, just print the command |
| -Dproperty=value | Sets a JDK system property value |

**flume-ng 代理选项**

当运行一个代理命令时，Flume NG代理会使用给定的配置文件启动(必须给定一个文件)

|  |  |
| --- | --- |
| **Option** | **Description** |
| --conf-file,-f <file> | Indicates which configuration file you want to run with (required) |
| --name,-n <agentname> | Indicates the name of agent on which we're running (required) |

**flume-ng avro-client 选项**

一个Avro client运行后会发送从stdin输入的文件或者数据到一个被Flume NG Avro Source监听的主机端口。

|  |  |
| --- | --- |
| **Option** | **Description** |
| --host,-H <hostname> | Specifies the hostname of the Flume agent (may be localhost) |
| --port,-p <port> | Specifies the port on which the Avro source is listening |
| --filename,-F <filename> | Sends each line of <filename> to Flume (optional) |
| --headerFile,-F <file> | Header file containing headers as key/value pairs on each new line |

The Avro client treats each line (terminated by \n, \r, or \r\n) as an event. Think of the avro-client command as cat for Flume. For instance, the following creates one event per Linux user and sends it to Flume's avro source on localhost:41414.

In a new window type the following:

|  |
| --- |
| $ bin/flume-ng avro-client --conf conf -H localhost -p41414-F /etc/passwd -Dflume.root.logger=DEBUG,console |

You should see something like this:

|  |
| --- |
| 2012-03-1616:39:17,124(main) [DEBUG - org.apache.flume.client.avro.AvroCLIClient.run(AvroCLIClient.java:175)] Finished  2012-03-1616:39:17,127(main) [DEBUG - org.apache.flume.client.avro.AvroCLIClient.run(AvroCLIClient.java:178)] Closing reader  2012-03-1616:39:17,127(main) [DEBUG - org.apache.flume.client.avro.AvroCLIClient.run(AvroCLIClient.java:183)] Closing transceiver  2012-03-1616:39:17,129(main) [DEBUG - org.apache.flume.client.avro.AvroCLIClient.main(AvroCLIClient.java:73)] Exiting |

And in your first window, where the server is running:

|  |
| --- |
| 2012-03-1616:39:16,738(New I/O server boss #1([id:0x49e808ca, /0:0:0:0:0:0:0:0:41414])) [INFO - org.apache.avro.ipc.NettyServer$NettyServerAvroHandler.handleUpstream(NettyServer.java:123)] [id:0x0b92a848, /1  27.0.0.1:39577=> /127.0.0.1:41414] OPEN  2012-03-1616:39:16,742(New I/O server worker #1-1) [INFO - org.apache.avro.ipc.NettyServer$NettyServerAvroHandler.handleUpstream(NettyServer.java:123)] [id:0x0b92a848, /127.0.0.1:39577=> /127.0.0.1:41414] BOU  ND: /127.0.0.1:41414  2012-03-1616:39:16,742(New I/O server worker #1-1) [INFO - org.apache.avro.ipc.NettyServer$NettyServerAvroHandler.handleUpstream(NettyServer.java:123)] [id:0x0b92a848, /127.0.0.1:39577=> /127.0.0.1:41414] CON  NECTED: /127.0.0.1:39577  2012-03-1616:39:17,129(New I/O server worker #1-1) [INFO - org.apache.avro.ipc.NettyServer$NettyServerAvroHandler.handleUpstream(NettyServer.java:123)] [id:0x0b92a848, /127.0.0.1:39577:> /127.0.0.1:41414] DISCONNECTED  2012-03-1616:39:17,129(New I/O server worker #1-1) [INFO - org.apache.avro.ipc.NettyServer$NettyServerAvroHandler.handleUpstream(NettyServer.java:123)] [id:0x0b92a848, /127.0.0.1:39577:> /127.0.0.1:41414] UNBOUND  2012-03-1616:39:17,129(New I/O server worker #1-1) [INFO - org.apache.avro.ipc.NettyServer$NettyServerAvroHandler.handleUpstream(NettyServer.java:123)] [id:0x0b92a848, /127.0.0.1:39577:> /127.0.0.1:41414] CLOSED  2012-03-1616:39:17,302(Thread-1) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:68)] Event: { headers:{} body:[B@5c1ae90c}  2012-03-1616:39:17,302(Thread-1) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:68)] Event: { headers:{} body:[B@6aba4211}  2012-03-1616:39:17,302(Thread-1) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:68)] Event: { headers:{} body:[B@6a47a0d4}  2012-03-1616:39:17,302(Thread-1) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:68)] Event: { headers:{} body:[B@48ff4cf}  ... |

恭喜！你已经拥有一个运行中的Aapche Flume！

## Providing Feedback

宋琼韬