Getting Started

What is Flume NG?

Flume NG aims to be significantly simpler, smaller, and easier to deploy than Flume OG. In doing so, we do not commit to maintaining backward compatibility of Flume NG with Flume OG. We're currently soliciting feedback from those who are interested in testing Flume NG for correctness, ease of use, and potential integration with other systems.

What's Changed?

Flume NG (Next Generation) is a huge departure from Flume OG (Original Generation) in its implementation although many of the original concepts are the same. If you're already familiar with Flume, here's what you need to know.

- You still have *sources* and *sinks* and they still do the same thing. They are now connected by *channels*.
- Channels are pluggable and dictate durability. Flume NG ships with an in-memory channel for fast, but non-durable event delivery and a file-based channel for durable event delivery.
- There's no more logical or physical nodes. We call all physical nodes *agents* and agents can run zero or more sources and sinks.
- There's no master and no ZooKeeper dependency anymore. At this time, Flume runs with a simple file-based configuration system.
- Just about everything is a plugin, some end user facing, some for tool and system developers. Pluggable components include channels, sources, sinks, interceptors, sink processors, and event serializers.

Please file JIRAs and/or vote for features you feel are important.

Getting Flume NG

Flume is available as a source tarball and binary on the Downloads section of the Flume Website. If you are not planning on creating patches for Flume, the binary is likely the easiest way to get started.

Building From Source

To build Flume NG from source, you'll need git, the Sun JDK 1.6, Apache Maven 3.x, about 90MB of local disk space and an Internet connection.

1. Check out the source

```
$ git clone https://git-wip-us.apache.org/repos/asf/flume.git flume
$ cd flume
$ git checkout trunk
```

2. Compile the project

The Apache Flume build requires more memory than the default configuration. We recommend you set the following Maven options:

```
export MAVEN_OPTS="-Xms512m -Xmx1024m -XX:PermSize=256m -XX:
MaxPermSize=512m"
```

```
# Build the code and run the tests (note: use mvn install, not mvn
package, since we deploy Jenkins SNAPSHOT jars daily, and Flume is a multi-
module project)
$ mvn install
# ...or build the code without running the tests
$ mvn install -DskipTests
```

(Please note that Flume requires that Google Protocol Buffers compiler be in the path for the build to be successful. You download and install it by following the instructions here.)

This produces two types of packages in flume-ng-dist/target. They are:

- apache-flume-ng-dist-1.4.0-SNAPSHOT-bin.tar.gz - A binary distribution of Flume, ready to run.
- apache-flume-ng-dist-1.4.0-SNAPSHOT-src.tar.gz - A source-only distribution of Flume.

If you're a user and you just want to run Flume, you probably want the -bin version. Copy one out, decompress it, and you're ready to go.

```
$ 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. Create your own properties file based on the working template (or create one from scratch)

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

4. (Optional) Create your flume-env.sh file based on the template (or create one from scratch). The flume-ng executable looks for and sources a file named "flume-env.sh" in the conf directory specified by the --conf/-c commandline option. One use case for using flume-env.sh would be to specify debugging or profiling options via JAVA_OPTS when developing your own custom Flume NG components such as sources and sinks.

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

5. Configure and Run Flume NG

After you've configured Flume NG (see below), you can run it with the `bin/flume-ng` executable. This script has a number of arguments and modes.

Configuration

Flume uses a Java property file based configuration format. It is required that you tell Flume which file to use by way of the `-f <file>` option (see above) when running an agent. The file can live anywhere, but historically - and in the future - the `conf` directory will be the correct place for config files.

Let's start with a basic example. Copy and paste this into `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 to 0.0.0.0:41414. 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
```

This example creates a memory channel (i.e. an unreliable or "best effort" transport), an Avro RPC source, and a logger sink and connects them together. Any events received by the Avro source are routed to the channel `ch1` and delivered to the logger sink. It's important to note that defining components is the first half of configuring Flume; they must be activated by listing them in the `<agent>.channels`, `<agent>.sources`, and sections. Multiple sources, sinks, and channels may be listed, separated by a space.

For full details, please see the javadoc for the `org.apache.flume.conf.properties.PropertiesFileConfigurationProvider` class.

This is a listing of the implemented sources, sinks, and channels at this time. Each plugin has its own optional and required configuration properties so **please** see the javadocs (for now).

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	org.apache.flume.source.StressSource

		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).	
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)*

The flume-ng executable lets you run a Flume NG agent or an Avro client which is useful for testing and experiments. No matter what, you'll need to specify a command (e.g. `agent` or `avro-client`) and a conf directory (`--conf <conf dir>`). All other options are command-specific.

To start the flume server using the flume.conf above:

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

Notice that the agent name is specified by `-n agent1` and must match a agent name given in `-f conf/flume.conf`

Your output should look something like this:

```
$ bin/flume-ng agent --conf conf/ -f conf/flume.conf -n agent1
2012-03-16 16:36:11,918 (main) [INFO - org.apache.flume.lifecycle.
LifecycleSupervisor.start(LifecycleSupervisor.java:58)] Starting lifecycle
supervisor 1
2012-03-16 16:36:11,921 (main) [INFO - org.apache.flume.node.FlumeNode.
start(FlumeNode.java:54)] Flume node starting - agent1
2012-03-16 16:36:11,926 (lifecycleSupervisor-1-0) [INFO - org.apache.flume.
node.nodemanager.DefaultLogicalNodeManager.start(DefaultLogicalNodeManager.
java:110)] Node manager starting
2012-03-16 16:36:11,928 (lifecycleSupervisor-1-0) [INFO - org.apache.flume.
lifecycle.LifecycleSupervisor.start(LifecycleSupervisor.java:58)] Starting
lifecycle supervisor 10
2012-03-16 16:36:11,929 (lifecycleSupervisor-1-0) [DEBUG - org.apache.
flume.node.nodemanager.DefaultLogicalNodeManager.start
(DefaultLogicalNodeManager.java:114)] Node manager started
2012-03-16 16:36:11,926 (lifecycleSupervisor-1-1) [INFO - org.apache.flume.
conf.file.AbstractFileConfigurationProvider.start
(AbstractFileConfigurationProvider.java:67)] Configuration provider
starting
2012-03-16 16:36:11,930 (lifecycleSupervisor-1-1) [DEBUG - org.apache.
flume.conf.file.AbstractFileConfigurationProvider.start
(AbstractFileConfigurationProvider.java:87)] Configuration provider started
2012-03-16 16:36:11,930 (conf-file-poller-0) [DEBUG - org.apache.flume.
conf.file.AbstractFileConfigurationProvider$FileWatcherRunnable.run
(AbstractFileConfigurationProvider.java:189)] Checking file:conf/flume.
conf for changes
2012-03-16 16:36:11,931 (conf-file-poller-0) [INFO - org.apache.flume.conf.
file.AbstractFileConfigurationProvider$FileWatcherRunnable.run
(AbstractFileConfigurationProvider.java:196)] Reloading configuration file:
conf/flume.conf
2012-03-16 16:36:11,936 (conf-file-poller-0) [DEBUG - org.apache.flume.
conf.properties.FlumeConfiguration$AgentConfiguration.isValid
(FlumeConfiguration.java:225)] Starting validation of configuration for
agent: 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: {}


}
2012-03-16 16:36:11,936 (conf-file-poller-0) [INFO - org.apache.flume.conf.
properties.FlumeConfiguration.validateConfiguration(FlumeConfiguration.
java:119)] Post-validation flume configuration contains configuation  for
agents: [agent1]
2012-03-16 16:36:11,937 (conf-file-poller-0) [DEBUG - org.apache.flume.
channel.DefaultChannelFactory.create(DefaultChannelFactory.java:67)]
Creating instance of channel ch1 type memory
2012-03-16 16:36:11,944 (conf-file-poller-0) [DEBUG - org.apache.flume.
source.DefaultSourceFactory.create(DefaultSourceFactory.java:73)] Creating
instance of source avro-source1, type avro
2012-03-16 16:36:11,957 (conf-file-poller-0) [INFO - org.apache.flume.sink.
DefaultSinkFactory.create(DefaultSinkFactory.java:69)] Creating instance
of sink log-sink1 typelogger
2012-03-16 16:36:11,963 (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:0.0.0.0 port:41414 } }} sinkRunners:{log-sink1=SinkRunner: {
policy:org.apache.flume.sink.DefaultSinkProcessor@79f6f296 counterGroup:{
name:null counters:{} } }} channels:{ch1=org.apache.flume.channel.
MemoryChannel@43b09468} }
2012-03-16 16:36:11,974 (lifecycleSupervisor-1-1) [INFO - org.apache.flume.
source.AvroSource.start(AvroSource.java:122)] Avro source starting:
AvroSource: { bindAddress:0.0.0.0 port:41414 }
2012-03-16 16:36:11,975 (Thread-1) [DEBUG - org.apache.flume.
SinkRunner$PollingRunner.run(SinkRunner.java:123)] Polling sink runner
starting
2012-03-16 16:36:12,352 (lifecycleSupervisor-1-1) [DEBUG - org.apache.
flume.source.AvroSource.start(AvroSource.java:132)] Avro source started
```

flume-ng global options

Option	Description
--conf,-c <conf>	Use configs in <conf> directory
	Append to the classpath

--classpath,-C <cp>	
--dryrun,-d	Do not actually start Flume, just print the command
-Dproperty=value	Sets a JDK system property value

flume-ng agent options

When given the agent command, a Flume NG agent will be started with a given configuration file (required).

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 options

Run an Avro client that sends either a file or data from stdin to a specified host and port where a Flume NG Avro Source is listening.

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 -p 41414 -F /etc
/passwd -Dflume.root.logger=DEBUG,console
```

You should see something like this:

```
2012-03-16 16:39:17,124 (main) [DEBUG - org.apache.flume.client.avro.
AvroCLIClient.run(AvroCLIClient.java:175)] Finished
2012-03-16 16:39:17,127 (main) [DEBUG - org.apache.flume.client.avro.
AvroCLIClient.run(AvroCLIClient.java:178)] Closing reader
2012-03-16 16:39:17,127 (main) [DEBUG - org.apache.flume.client.avro.
AvroCLIClient.run(AvroCLIClient.java:183)] Closing transceiver
2012-03-16 16:39:17,129 (main) [DEBUG - org.apache.flume.client.avro.
AvroCLIClient.main(AvroCLIClient.java:73)] Exiting
```

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

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

Congratulations! You have Apache Flume running!

Providing Feedback

For help building, configuring, and running Flume (NG or otherwise), the best place is always the user mailing list. Send an email to user-subscribe@flume.apache.org to subscribe and user@flume.apache.org to post once you've subscribed. The archives are available at http://mail-archives.apache.org/mod_mbox/incubator-flume-user/ (up through part of July 2012) and http://mail-archives.apache.org/mod_mbox/incubator-flume-user/http://mail-archives.apache.org/mod_mbox/flume-user/ (starting through part of July 2012 onwards).

If you believe you've found a bug or wish to file a feature request or improvement, don't be shy. Go to https://issues.apache.org/jira/browse/FLUME and file a JIRA for the version of Flume. For NG, please set the "Affects Version" to the appropriate milestone / release. Just leave any field you're not sure about blank. We'll bug you for details if we need them. Note that you must create an Apache JIRA account and log in before you can file issues.