exec

노트북: temp
만든 날짜: 2019-04-30 오후 3:30 **수정한 날짜:** 2019-04-30 오후 3:31
작성자: includesorrow@gmail.com
URL: https://flume.apache.org/FlumeUserGuide.html#flume-sources

Exec Source

Exec source runs a given Unix command on start-up and expects that process to continuously produce data on standard out (stderr is simply discarded, unless property logStdErr is set to true). If the process exits for any reason, the source also exits and will produce no further data. This means configurations such as `cat [named pipe]` or `tail -F [file]` are going to produce the desired results where as `date`will probably not - the former two commands produce streams of data where as the latter produces a single event and exits.

Required properties are in **bold**.

Property Name	Default	Description
channels	–	
type	–	The component type name, needs to be `exec`
command	–	The command to execute
shell	–	A shell invocation used to run the command. e.g. /bin/sh -c. Required only for commands relying on shell features like wildcards, back ticks, pipes etc.
restartThrottle	10000	Amount of time (in millis) to wait before attempting a restart
restart	false	Whether the executed cmd should be restarted if it dies
logStdErr	false	Whether the command’s stderr should be logged

batchSize	20	The max number of lines to read and send to the channel at a time
batchTimeout	3000	Amount of time (in milliseconds) to wait, if the buffer size was not reached, before data is pushed downstream
selector.type	replicating	replicating or multiplexing
selector.*		Depends on the selector.type value
interceptors	–	Space-separated list of interceptors
interceptors.*		

Warning The problem with ExecSource and other asynchronous sources is that the source can not guarantee that if there is a failure to put the event into the Channel the client knows about it. In such cases, the data will be lost. As a for instance, one of the most commonly requested features is the `tail -F [file]`-like use case where an application writes to a log file on disk and Flume tails the file, sending each line as an event. While this is possible, there's an obvious problem; what happens if the channel fills up and Flume can't send an event? Flume has no way of indicating to the application writing the log file that it needs to retain the log or that the event hasn't been sent, for some reason. If this doesn't make sense, you need only know this: Your application can never guarantee data has been received when using a unidirectional asynchronous interface such as ExecSource! As an extension of this warning - and to be completely clear - there is absolutely zero guarantee of event delivery when using this source. For stronger reliability guarantees, consider the Spooling Directory Source, Taildir Source or direct integration with Flume via the SDK.

Example for agent named a1:

```
a1.sources = r1a1.channels = c1a1.sources.r1.type =
execa1.sources.r1.command = tail -F
/var/log/securea1.sources.r1.channels = c1
```

The 'shell' config is used to invoke the 'command' through a command shell (such as Bash or Powershell). The 'command' is passed as an argument to 'shell' for execution. This allows the 'command' to use features from the shell such as wildcards, back ticks, pipes, loops, conditionals etc. In the absence of the 'shell' config, the 'command' will be invoked directly. Common values for 'shell' : '/bin/sh -c', '/bin/ksh -c', 'cmd /c', 'powershell -Command', etc.

```
a1.sources.tailsource-1.type = execa1.sources.tailsource-1.shell = /bin/bash -ca1.sources.tailsource-1.command = for i in /path/*.txt; do cat $i; done
```