Flume

OutLine

- Flume简介
- Flume Core
- 【实践】搭建基于Flume日志收集系统

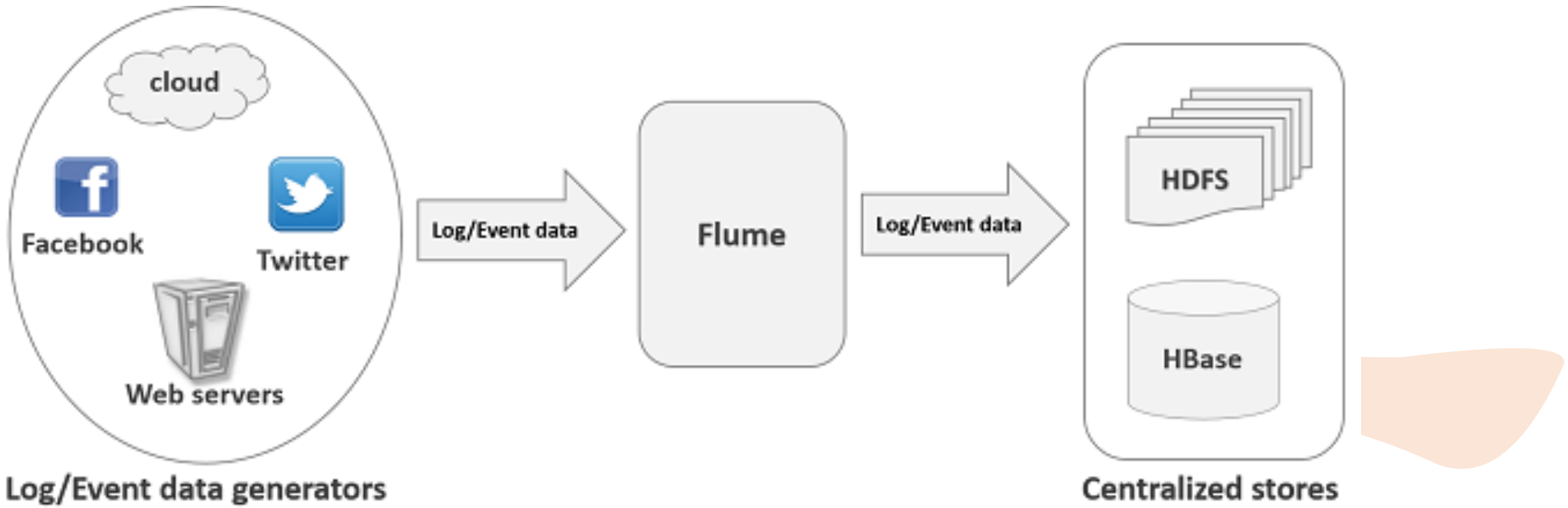
——八斗大数据内部资料，盗版必究——

简 介

- Apache软件基金顶级项目
- Apache Flume是一个分布式、可信任的弹性系统，用于高效收集、汇聚和移动大规模日志信息从多种不同的数据源到一个集中的数据存储中心（HDFS、HBase）
- 功能：
  - 支持在日志系统中定制各类数据发送方，用于收集数据
  - Flume提供对数据进行简单处理，并写到各种数据接收方（可定制）的能力
- 多种数据源：
  - Console、RPC、Text、Tail、Syslog、Exec等

——八斗大数据内部资料，盗版必究——

简 介



——八斗大数据内部资料，盗版必究——

特 点

- Flume可以高效率的将多个网站服务器中收集的日志信息存入HDFS/HBase中
- 使用Flume，我们可以将从多个服务器中获取的数据迅速的移交给Hadoop中
- 支持各种接入资源数据的类型以及接出数据类型
- 支持多路径流量，多管道接入流量，多管道接出流量，上下文路由等
- 可以被水平扩展

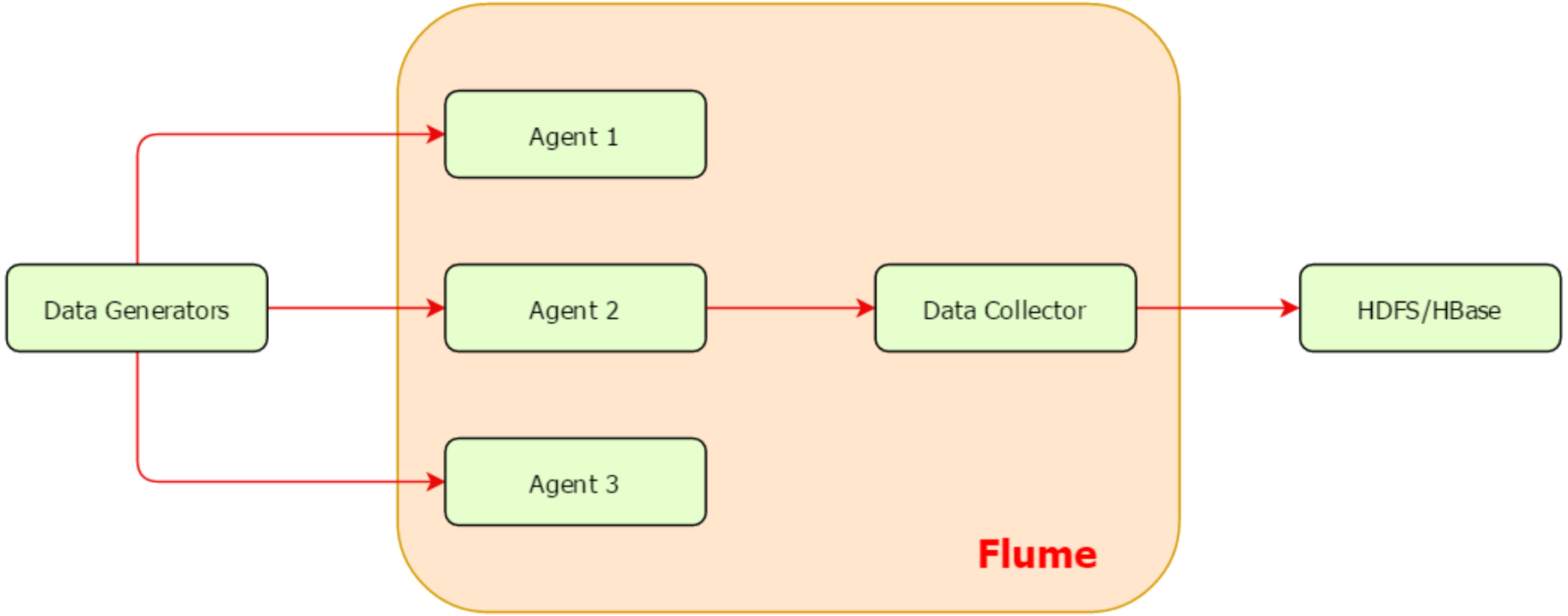
——八斗大数据内部资料，盗版必究——

OutLine

- Flume简介
- Flume Core
- 【实践】搭建基于Flume日志收集系统

——八斗大数据内部资料，盗版必究——

外 部 架 构



- 数据发生器（如：facebook,twitter）产生的数据被被单个的运行在数据发生器所在服务器上的agent所收集，之后数据收容器从各个agent上汇集数据并将采集到的数据存入到HDFS或者HBase中

——八斗大数据内部资料，盗版必究——

事件(Flume Event)

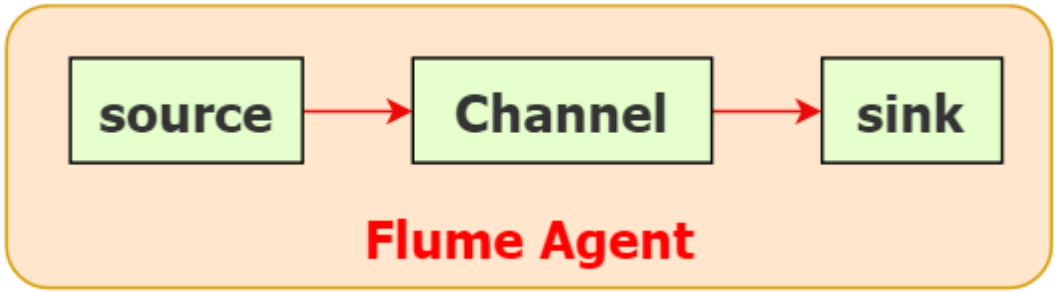
- Flume使用Event对象来作为传递数据的格式，是内部数据传输的最基本单元
- 由两部分组成：转载数据的字节数组+可选头部

- Header 是 key/value 形式的，可以用来制造路由决策或携带其他结构化信息(如事件的时间戳或事件来源的服务器主机名)。你可以把它想象成和 HTTP 头一样提供相同的功能——通过该方法来传输正文之外的额外信息。Flume提供的不同source会给其生成的event添加不同的header
- Body是一个字节数组，包含了实际的内容，

——八斗大数据内部资料，盗版必究——

代理(Flume Agent)

- Flume内部有一个或者多个Agent
- 每一个Agent是一个独立的守护进程(JVM)
- 从客户端哪儿接收收集，或者从其他的Agent哪儿接收，然后迅速的将获取的数据传给下一个目的节点Agent



- Agent主要由source、channel、sink三个组件组成。

——八斗大数据内部资料，盗版必究——

Agent Source

- 一个Flume源
- 负责一个外部源（数据发生器），如一个web服务器传递给他的事件
- 该外部源将它的事件以Flume可以识别的格式发送到Flume中
- 当一个Flume源接收到一个事件时，其将通过一个或者多个通道存储该事件

——八斗大数据内部资料，盗版必究——

Agent Channel

- 通道：采用被动存储的形式，即通道会缓存该事件直到该事件被sink组件处理
- 所以Channel是一种短暂的存储容器，它将从source处接收到的event格式的数据缓存起来,直到它们被sinks消费掉,它在source和sink间起着一共桥梁的作用,channel是一个完整的事务,这一点保证了数据在收发的时候的一致性. 并且它可以和任意数量的source和sink链接

- 可以通过参数设置event的最大个数
- Flume通常选择FileChannel，而不使用Memory Channel
  - Memory Channel：内存存储事务，吞吐率极高，但存在丢数据风险
  - File Channel：本地磁盘的事务实现模式，保证数据不会丢失（WAL实现）

——八斗大数据内部资料，盗版必究——

Agent Sink

- Sink会将事件从Channel中移除，并将事件放置到外部数据介质上
  - 例如：通过Flume HDFS Sink将数据放置到HDFS中，或者放置到下一个Flume的Source，等到下一个Flume处理。
  - 对于缓存在通道中的事件，Source和Sink采用**异步处理**的方式
- Sink成功取出Event后，将Event从Channel中移除
- Sink必须作用于一个确切的Channel
- 不同类型的Sink：
  - 存储Event到最终目的的终端：HDFS、Hbase
  - 自动消耗：Null Sink
  - 用于Agent之间通信：Avro

——八斗大数据内部资料，盗版必究——

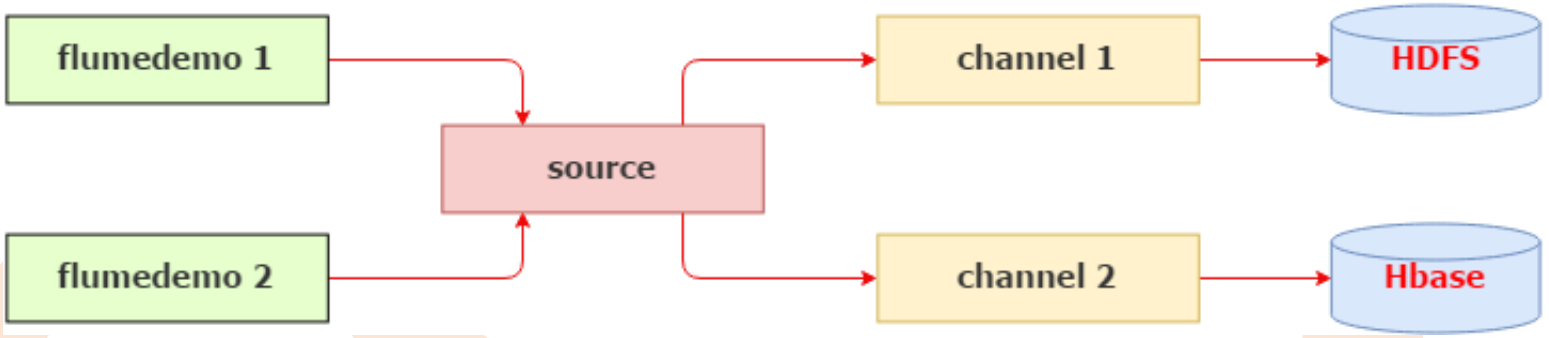
Agent Interceptor

- Interceptor用于Source的一组拦截器，按照预设的顺序必要地方对events进行过滤和自定义的处理逻辑实现
- 在app(应用程序日志)和 source 之间的，对app日志进行拦截处理的。也即在日志进入到source之前，对日志进行一些包装、清新过滤等等动作
- 官方上提供的已有的拦截器有：
  - Timestamp Interceptor：在event的header中添加一个key叫：timestamp,value为当前的时间戳
  - Host Interceptor：在event的header中添加一个key叫：host,value为当前机器的hostname或者ip
  - Static Interceptor：可以在event的header中添加自定义的key和value
  - Regex Filtering Interceptor：通过正则来清洗或包含匹配的events
  - Regex Extractor Interceptor：通过正则表达式来在header中添加指定的key,value则为正则匹配的部分
- flume的拦截器也是chain形式的，可以对一个source指定多个拦截器，按先后顺序依次处理

——八斗大数据内部资料，盗版必究——

Agent Selector

- channel selectors 有两种类型:
  - Replicating Channel Selector (default)：将source过来的events发往所有channel
  - Multiplexing Channel Selector：而Multiplexing 可以选择该发往哪些channel
- 对于有选择性选择数据源，明显需要使用Multiplexing 这种分发方式



- 问题：Multiplexing 需要判断header里指定key的值来决定分发到某个具体的channel，如果demo和demo2同时运行在同一个服务器上，如果在不同的服务器上运行，我们可以在 source1上加上一个 host 拦截器，这样可以通过header中的host来判断event该分发给哪个channel，而这里是在同一个服务器上，由host是区分不出来日志的来源的，我们必须想办法在header中添加一个key来区分日志的来源
  - 通过设置上游不同的Source就可以解决

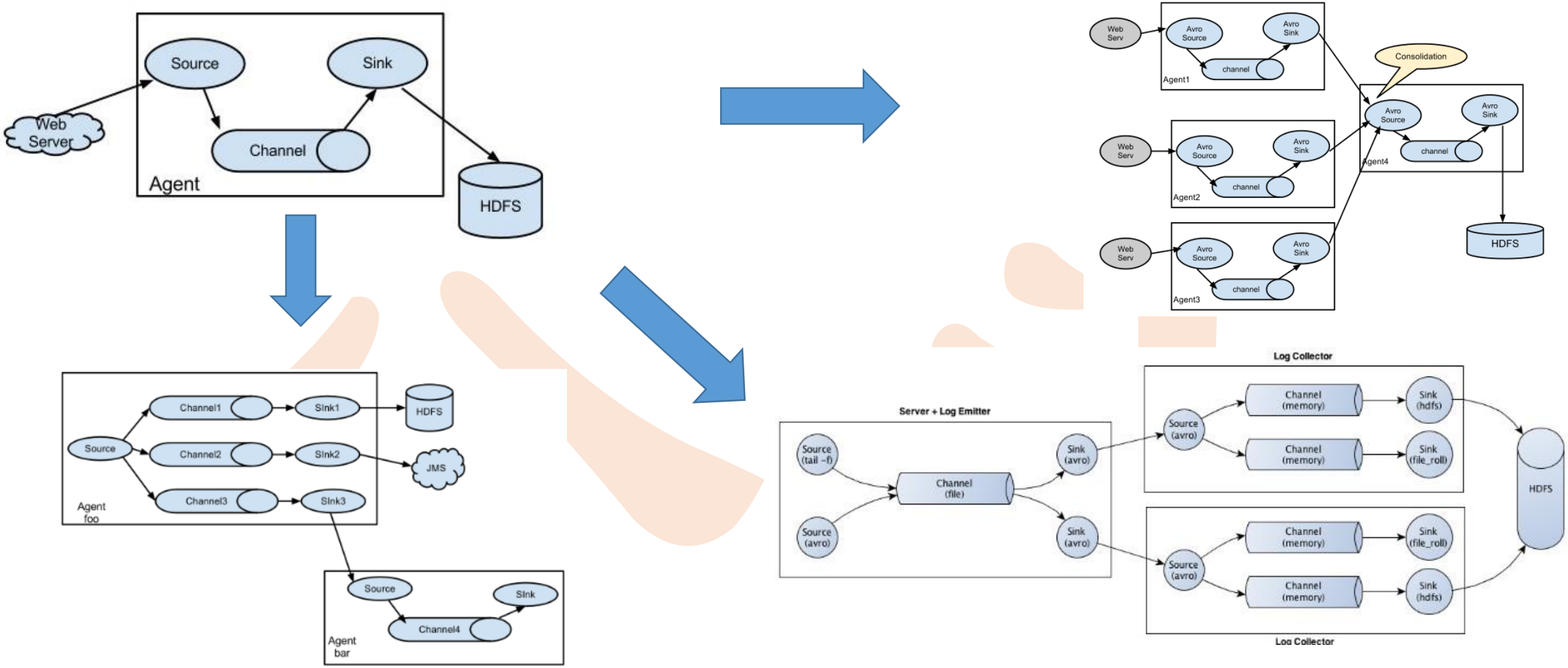
——八斗大数据内部资料，盗版必究——

可靠性

- Flume保证单次跳转可靠性的方式：传送完成后，该事件才会从通道中移除
- Flume使用事务性的方法来保证事件交互的可靠性。
- 整个处理过程中，如果因为网络中断或者其他原因，在某一步被迫结束了，这个数据会在下一次重新传输。
- Flume可靠性还体现在数据可暂存上面，当目标不可访问后，数据会暂存在Channel中，等目标可访问之后，再进行传输
- Source和Sink封装在一个事务的存储和检索中，即事件的放置或者提供由一个事务通过通道来分别提供。这保证了事件集在流中可靠地进行端到端的传递。
  - Sink开启事务
  - Sink从Channel中获取数据
  - Sink把数据传给另一个Flume Agent的Source中
  - Source开启事务
  - Source把数据传给Channel
  - Source关闭事务
  - Sink关闭事务

——八斗大数据内部资料，盗版必究——

复杂的流动



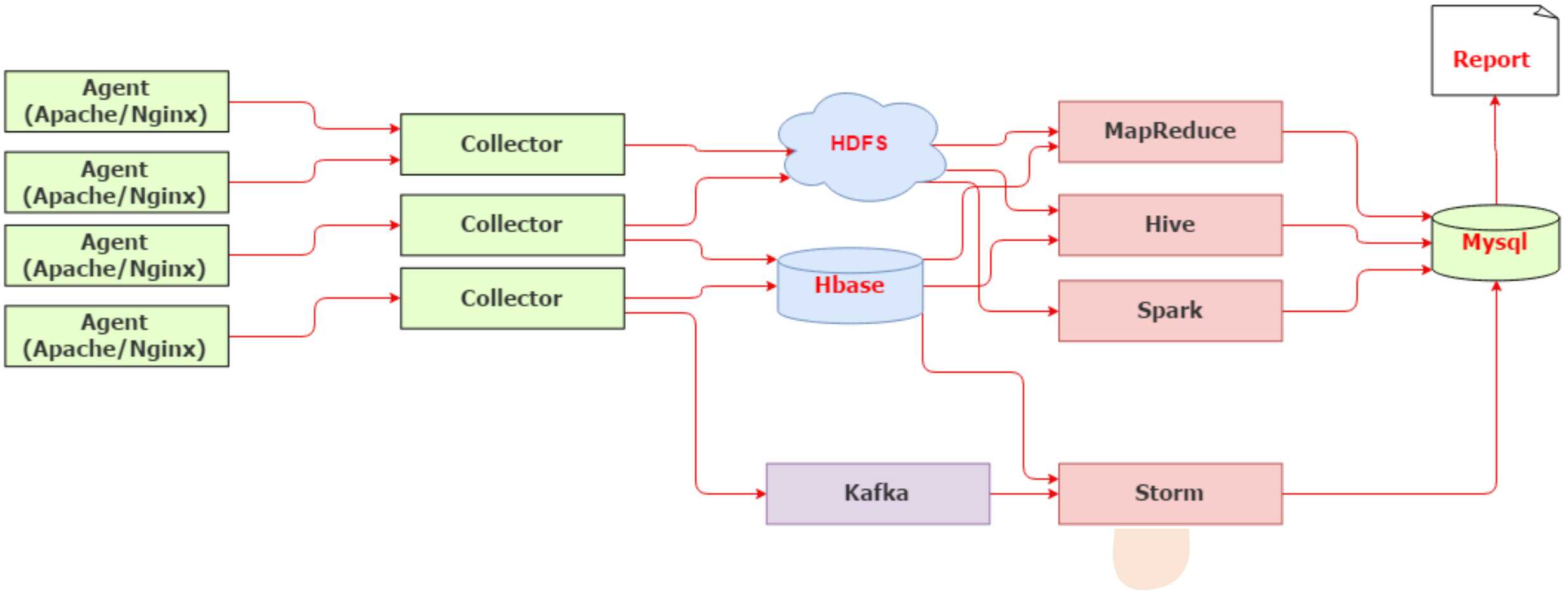
——八斗大数据内部资料，盗版必究——

OutLine

- Flume简介
- Flume Core
- 【实践】搭建基于Flume日志收集系统

——八斗大数据内部资料，盗版必究——

日志采集系统



——八斗大数据内部资料，盗版必究——

安 装

- 下载源码包：
-]# wget "http://mirrors.cnnic.cn/apache/flume/1.6.0/apache-flume-1.6.0-bin.tar.gz"
- 解压：
-]# tar xvzf apache-flume-1.6.0-bin.tar.gz
- 修改配置文件：
- 在/usr/local/src/apache-flume-1.6.0-bin/conf目录下

——八斗大数据内部资料，盗版必究——

实践一：NetCat方式

-]# vim flume_netcat.conf

- 运行flume-ng
-]# ./bin/flume-ng agent --conf conf --conf-file ./conf/flume_netcat.conf --name a1 -Dflume.root.logger=INFO,console

- 验证:
-]# telnet 127.0.0.1 44444

```
[root@master conf]# vim flume.conf
# 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
```

——八斗大数据内部资料，盗版必究——

实践二：Exec方式

-]# vim flume.conf

```
[root@master conf]# vim flume_exec.conf
# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = exec
a1.sources.r1.command = tail -f /home/badou/flume_test/monitor_source/1.txt

# 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
```

- 运行flume-ng
-]# ./bin/flume-ng agent --conf conf --conf-file ./conf/flume_exec.conf --name a1 -Dflume.root.logger=INFO,console

- 验证：
-]# echo 'ccc' >> 1.txt

```
2017-06-07 03:14:37,127 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSin
ava:94)] Event: { headers:{} body: 63 63 63                                  ccc }
2017-06-07 03:14:41,129 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSin
ava:94)] Event: { headers:{} body: 63 63 63                                  ccc }
2017-06-07 03:14:41,129 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSin
ava:94)] Event: { headers:{} body: 63 63 63                                  ccc }
2017-06-07 03:14:41,130 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSin
ava:94)] Event: { headers:{} body: 63 63 63                                  ccc }
2017-06-07 03:14:46,519 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSin
ava:94)] Event: { headers:{} body: 63 63 63 31                               ccc1 }
2017-06-07 03:14:50,670 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSin
ava:94)] Event: { headers:{} body: 63                                        c }
```

——八斗大数据内部资料，盗版必究——

实践三：HDFS

-]# vim flume.conf

```
[root@master conf]# vim flume.conf
# Define a memory channel called c1 on a1
a1.channels.c1.type = memory

# Define an Avro source called r1 on a1 and tell it
# to bind to 0.0.0.0:41414. Connect it to channel c1.
a1.sources.r1.channels = c1
a1.sources.r1.type = avro
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = 41414

a1.sinks.k1.type = hdfs
a1.sinks.k1.channel = c1
a1.sinks.k1.hdfs.path = hdfs://master:9000/flume_data_pool
a1.sinks.k1.hdfs.filePrefix = events-
a1.sinks.k1.hdfs.fileType = DataStream
a1.sinks.k1.hdfs.writeFormat = Text
a1.sinks.k1.hdfs.rollSize = 0
a1.sinks.k1.hdfs.rollCount = 600000
a1.sinks.k1.hdfs.rollInterval = 600

#
# Finally, now that we've defined all of our components, tell
# a1 which ones we want to activate.
a1.channels = c1
a1.sources = r1
a1.sinks = k1
```

- 运行flume-ng
- ./bin/flume-ng agent --conf conf --conf-file ./conf/flume.conf -name a1 -Dflume.root.logger=DEBUG,console

- 验证：
- ./bin/flume-ng avro-client --conf conf -H master -p 41414 -F /home/badou/flume_test/monitor_source/3.txt -Dflume.root.logger=DEBUG,console

——八斗大数据内部资料，盗版必究——

实践四：模拟使用Flume监听日志变化，并且把增量的日志文件写入到hdfs中

-]# vim flume.conf

- 运行flume-ng
- ./bin/flume-ng agent --conf conf --conf-file ./conf/flume.conf -name a1 -Dflume.root.logger=DEBUG,console

- 验证：
- echo "11113" >> 1.log

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

# Describe/configure the source
## exec表示flume回去调用给的命令，然后从给的命令的结果中去拿数据
a1.sources.r1.type = exec
a1.sources.r1.command = tail -F /home/badou/flume_test/monitor_source/1.log
a1.sources.r1.channels = c1

# Describe the sink
## 表示下沉到hdfs，类型决定了下面的参数
a1.sinks.k1.type = hdfs
a1.sinks.k1.channel = c1
## 下面的配置告诉用hdfs去写文件的时候写到什么位置，下面的表示不是写死的，而是可以动态的变化的。表示输出的目录名称是可变的
a1.sinks.k1.hdfs.path = /flume/tailout/%y-%m-%d/%H%M/
##表示最后的文件的前缀
a1.sinks.k1.hdfs.filePrefix = events-
## 表示到了需要触发的时间时，是否要更新文件夹，true:表示要
a1.sinks.k1.hdfs.round = true
## 表示每隔1分钟改变一次
a1.sinks.k1.hdfs.roundValue = 1
## 切换文件的时候的时间单位是分钟
a1.sinks.k1.hdfs.roundUnit = minute
## 表示只要过了3秒钟，就切换生成一个新的文件
a1.sinks.k1.hdfs.rollInterval = 3
## 如果记录的文件大于20字节时切换一次
a1.sinks.k1.hdfs.rollSize = 20
## 当写了5个事件时触发
a1.sinks.k1.hdfs.rollCount = 5
## 收到了多少条消息往dfs中追加内容
a1.sinks.k1.hdfs.batchSize = 10
## 使用本地时间戳
a1.sinks.k1.hdfs.useLocalTimeStamp = true
#生成的文件类型，默认是Sequencefile，可用DataStream：为普通文本
a1.sinks.k1.hdfs.fileType = DataStream

# Use a channel which buffers events in memory
##使用内存的方式
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
```

——八斗大数据内部资料，盗版必究——

Q & A

@八斗学院