Flume1.5.0 的说明以及安装文档

版本号：V0.1

修订日期	版本号	修订类别	描述	修改人
2015.07	V0.1	初稿		张鑫

目录

一、什么是 Flume?

flume 作为 cloudera 开发的实时日志收集系统，受到了业界的认可与广泛应用。Flume 初始的发行版本目前被统称为 Flume OG（original generation），属于 cloudera。但随着 FLume 功能的扩展，Flume OG 代码工程臃肿、核心组件设计不合理、核心配置不标准等缺点暴露出来，尤其是在 Flume OG 的最后一个发行版本 0.94.0 中，日志传输不稳定的现象尤为严重，为了解决这些问题，2011 年 10 月 22 号，cloudera 完成了 Flume-728，对 Flume 进行了里程碑式的改动：重构核心组件、核心配置以及代码架构，重构后的版本统称为 Flume NG（next generation）；改动的另一原因是将 Flume 纳入 apache 旗下，cloudera Flume 改名为 Apache Flume。

二、Flume OG 和 Flume NG 的区别

Flume-og 有 agent.collector.master 三种角色

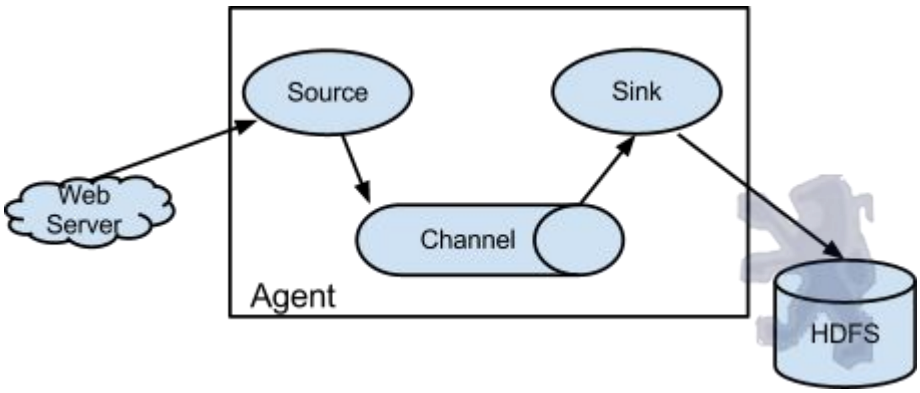
Flume-ng 只有 agent 一种角色

注：目前市场上没有明确指定的几乎都是指 Flume-NG，如若看到由三种角色的则是 Flume-og

三、Flume-NG 图解

Flume 以 agent 为最小的独立运行单位。一个 agent 就是一个 JVM。

单个 agent 由 Source、Sink 和 Channel 三大组件构成，如下图：



四、解析 Flume-NG 三大组件

说明 Flume NG 的功能，实际还是围绕着 Agent 的三个组件 Source、Channel、Sink 来。我们通过列表的方式分别对三个组件进行概要说明：

Flume Source

Source 类型	说明
Avro Source	支持 Avro 协议（实际上是 Avro RPC），内置支持
Thrift Source	支持 Thrift 协议，内置支持
Exec Source	基于 Unix 的 command 在标准输出上生产数据
JMS Source	从 JMS 系统（消息、主题）中读取数据，ActiveMQ 已经测试过
Spooling Directory Source	监控指定目录内数据变更
Netcat Source	监控某个端口，将流经端口的每一个文本行数据作为 Event 输入
Sequence Generator Source	序列生成器数据源，生产序列数据
Syslog Sources	读取 syslog 数据，产生 Event，支持 UDP 和 TCP 两种协议
HTTP Source	基于 HTTP POST 或 GET 方式的数据源，支持 JSON、BLOB 表示形式

Flume Channel

Channel 类型	说明
Memory Channel	Event 数据存储在内存中
JDBC Channel	Event 数据存储在持久化存储中，当前 Flume Channel 内置支持 Derby
File Channel	Event 数据存储在磁盘文件中
Spillable Memory Channel	Event 数据存储在内存中和磁盘上，当内存队列满了，会持久化到磁盘文件（当前试验性的，不建议生产环境使用）
Custom Channel	自定义 Channel 实现

Flume Sink

Sink 类型	说明
HDFS Sink	数据写入 HDFS

Logger Sink	数据写入日志文件
Avro Sink	数据被转换成 Avro Event，然后发送到配置的 RPC 端口上
Thrift Sink	数据被转换成 Thrift Event，然后发送到配置的 RPC 端口上
IRC Sink	数据在 IRC 上进行回放
File Roll Sink	存储数据到本地文件系统
Null Sink	丢弃到所有数据
HBase Sink	数据写入 HBase 数据库
Morphline Solr Sink	数据发送到 Solr 搜索服务器（集群）
ElasticSearch Sink	数据发送到 Elastic Search 搜索服务器（集群）
Custom Sink	自定义 Sink 实现

另外还有 Channel Selector、Sink Processor、Event Serializer、Interceptor 等组件，可以参考官网提供的用户手册。

五、安装启动 Flume

1、环境前提：hadoop2.2.0+hbase0.96.2(并以 flume 安装在 hs3 为案例)

2、下载解压 Flume1.5.0

#cd /opt/software/hadoop/flume/

#wget http://archive.apache.org/dist/flume/1.5.0/apache-flume-1.5.0-bin.tar.gz

#tar -xvf apache-flume-1.5.0-bin.tar.gz

#cd ./apache-flume-1.5.0-bin/conf

#cp flume-env.sh.template flume-env.sh

#vi flume-env.sh

增加 JAVA_HOME

```
# If this file is placed at FLUME_CONF_DIR/flume-env.sh, it will be so
# during Flume startup.

# Enviroment variables can be set here.

#JAVA_HOME=/usr/lib/jvm/java-6-sun
JAVA_HOME=/opt/software/java/jdk1.6.0_32
# Give Flume more memory and pre-allocate, enable remote monitoring vi
#JAVA_OPTS="-Xms100m -Xmx200m -Dcom.sun.management.jmxremote"

# Note that the Flume conf directory is always included in the classpa
#FLUME_CLASSPATH=""
```

3、将 flume 启动路径加入 profile

#vi /etc/profile

```
export FLUME_HOME=/opt/software/hadoop/flume/apache-flume-1.5.0-bin
export PATH=$PATH:$FLUME_HOME/bin
```

4、验证是否安装成功

#cd ../bin/

#sh flume-ng version

```
[root@hs3 conf]# vi flume-env.sh
[root@hs3 conf]# cd ../bin/
[root@hs3 bin]# sh flume-ng version
Flume 1.5.0
Source code repository: https://git-wip-us.apache.org/repos/asf/flume.git
Revision: 8633220df808c4cd0c13d1cf0320454a94f1ea97
Compiled by hshreedharan on Wed May  7 14:49:18 PDT 2014
From source with checksum a01fe726e4380ba0c9f7a7d222db961f
[root@hs3 bin]#
```

5、Agent 案例

Source 模拟为 log.txt(文本)持续不断的写入，用 shell 来模拟

注：详见 8.shell 语句模拟 webserver 持续写入 log.txt

Channel 为 memory，即使用内存存储中间结果

Sink 为 HBase（自定义 HBase 的 sink）

注：HBase 配置

hbase 的表 table 指定为 test_flume

hbase 的列簇 columnfamily 指定为 info

hbase 的 info 列簇下的列 age,address,number（三列）

hbase 的 rowkey 先设置为自定义模式 a1.sinks.k1.serializer.suffix 为 custom

hbase 的 rowkey 为 a1.sinks.k1.serializer.rowPrefixCol 所指定的列做 MD5 计算,如下图指定为 number（做 md5.32 位计算）即第三列。

#cd ../conf/

#vi exec_tail_to_hbase.conf

新建一个 agent 用来配置

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = exec
a1.sources.r1.channels = c1
a1.sources.r1.command = tail -F /home/aisino/test/log.txt

# Describe the sink
a1.sinks.k1.type = org.apache.flume.sink.hbase.AsyncHBaseSink
a1.sinks.k1.table = test_flume
a1.sinks.k1.columnFamily = info
a1.sinks.k1.serializer.payloadColumn = age,address,number
a1.sinks.k1.serializer.suffix = custom
#custom mean set by yourself
a1.sinks.k1.serializer.rowPrefixCol = number
a1.sinks.k1.serializer =  org.apache.flume.sink.hbase.AisinoAsyncHbaseEventSerializer
a1.sinks.k1.channel = memoryChannel

# 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
~
```

6、自定义 AisinoAsyncHbaseEventSerializer

注：详细 java 代码参考查看附

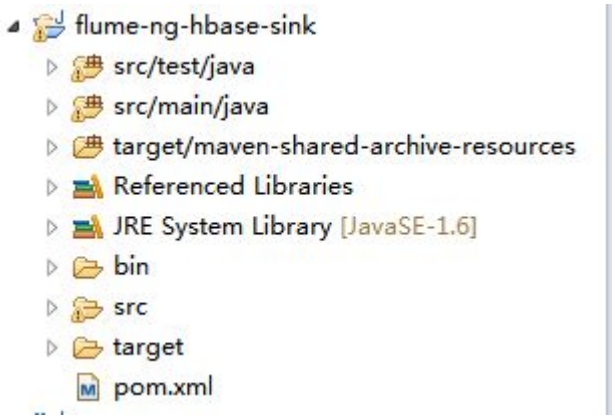
注：涉及 maven 命令参考查看附

下载官网 flume 源码 apache-flume-1.5.0-src.tar.gz

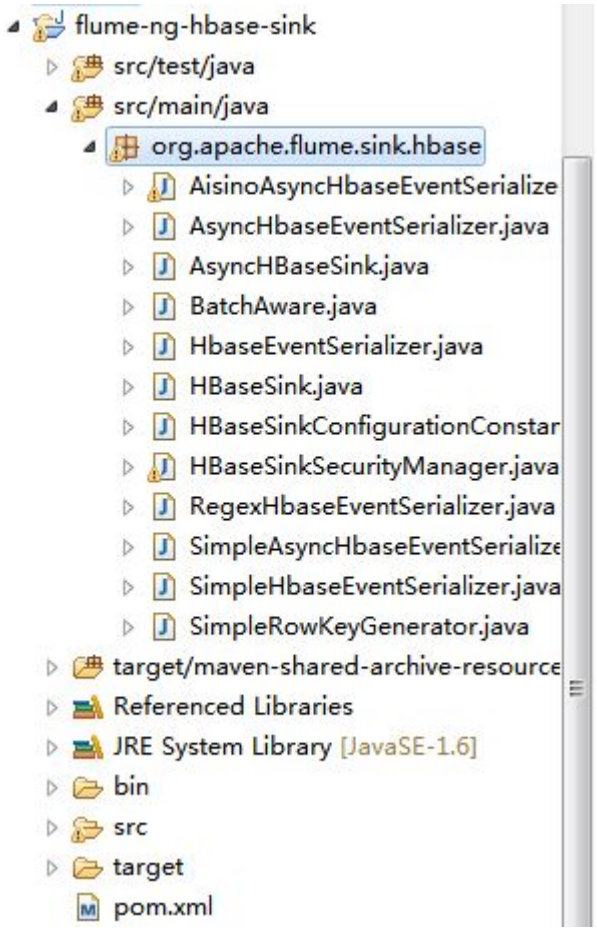
解压并且进入

apache-flume-1.5.0-src\apache-flume-1.5.0-src\flume-ng-sinks\flume-ng-hbase-sink

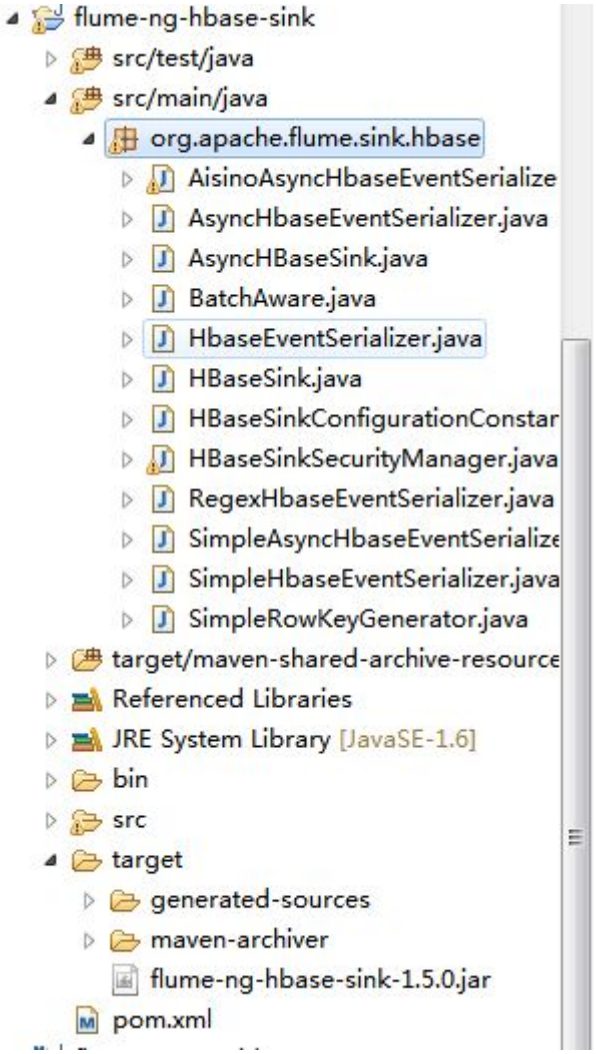
根据 pom.xml 来用 maven 转成 eclipse 项目后并导入 eclipse



导入 flume-ng-hbase-sink 之后新增 AisinoAsyncHbaseEventSerializer



Java 文件编写之后，用 maven 来编译 java 文件，并且打包（采用跳过 test 方式打包）生成如下 jar



将 flume-ng-hbase-sink-1.5.0.jar 替换掉 hs3（安装 flume 的节点）的路径下的 jar

/opt/software/hadoop/flume/apache-flume-1.5.0-bin/lib/flume-ng-hbase-sink-1.5.0.jar

7、启动 agent

前台执行命令为：

hs3#../bin/flume-ng agent -c . -f ./exec_tail_to_hbase.conf -n a1 -Dflume.root.logger=INFO,console

实际情况可将命令放置后台运行，目前放前台运行以配合调试及观察变化

后台执行命令为：

hs3#nohup ../bin/flume-ng agent -c . -f ./exec_tail_to_hbase.conf -n a1 -Dflume.root.logger=INFO,console &

8、另连 hs3 并且进 hbase shell 来查看数据的录入情况

#cd /opt/software/hadoop/hbase-0.96.2-hadoop2/bin

#sh hbase shell

hbase(main):006:0> create 'test_flume',{NAME=>'info'}

建表 test_flume,并建列簇 info，请与测试前先行将表建好

hbase(main):007:0> scan 'test_flume'

9、另连 hs3 模拟持续不断写入 log.txt

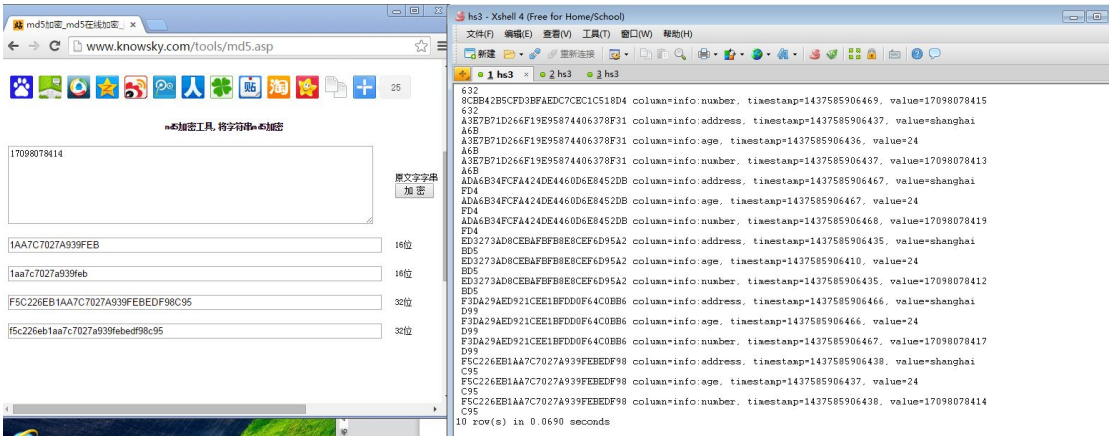
#cd /home/aisino/test

for i in {0..9};do echo "24,shanghai,1709807841$i" >> ./log.txt;echo $i;sleep 0.1;done

```
[root@hs3 test]# for i in {0..9};do echo "24,shanghai,1709807841$i" >> ./log.txt;echo $i;sleep 0.1;done
0
1
2
3
4
5
6
7
8
9
[root@hs3 test]# ll
total 56
-rw-r--r--. 1 root root 52838 Jul 15 16:56 flume-ng-hbase-sink-1.5.0.jar.bk
-rw-r--r--. 1 root root   240 Jul 22 10:28 log.txt
[root@hs3 test]# less log.txt
[root@hs3 test]# cat log.txt
24,shanghai,17098078410
24,shanghai,17098078411
24,shanghai,17098078412
24,shanghai,17098078413
24,shanghai,17098078414
24,shanghai,17098078415
24,shanghai,17098078416
24,shanghai,17098078417
24,shanghai,17098078418
24,shanghai,17098078419
[root@hs3 test]#
```

10、查看 hbase 数据录入的情况，截图如下

```
hbase(main):007:0> scan 'test_flume'
ROW                                   COLUMN+CELL
 1A3023BC30C51D6F4B71D6672A559        column=info:address, timestamp=1437585906465, value=shanghai
 29A
 1A3023BC30C51D6F4B71D6672A559        column=info:age, timestamp=1437585906457, value=24
 29A
 1A3023BC30C51D6F4B71D6672A559        column=info:number, timestamp=1437585906466, value=17098078416
 29A
 1D55138E1F7B69161228A82EF4715        column=info:address, timestamp=1437585906410, value=shanghai
 F26
 1D55138E1F7B69161228A82EF4715        column=info:age, timestamp=1437585906448, value=24
 F26
 1D55138E1F7B69161228A82EF4715        column=info:number, timestamp=1437585906435, value=17098078411
 F26
 216B2AA25F4ED3FEDC4852800B7B8        column=info:address, timestamp=1437585906467, value=shanghai
 362
 216B2AA25F4ED3FEDC4852800B7B8        column=info:age, timestamp=1437585906467, value=24
 362
 216B2AA25F4ED3FEDC4852800B7B8        column=info:number, timestamp=1437585906467, value=17098078418
 362
 4D5F0775B8F98B6670833609CBA1F        column=info:address, timestamp=1437585906437, value=shanghai
 298
 4D5F0775B8F98B6670833609CBA1F        column=info:age, timestamp=1437585906383, value=24
 298
 4D5F0775B8F98B6670833609CBA1F        column=info:number, timestamp=1437585906436, value=17098078410
 298
 8CBB42B5CFD3BFAEDC7CEC1C518D4        column=info:address, timestamp=1437585906438, value=shanghai
 632
 8CBB42B5CFD3BFAEDC7CEC1C518D4        column=info:age, timestamp=1437585906438, value=24
 632
 8CBB42B5CFD3BFAEDC7CEC1C518D4        column=info:number, timestamp=1437585906469, value=17098078415
 632
 A3E7B71D266F19E95874406378F31        column=info:address, timestamp=1437585906437, value=shanghai
 A6B
 A3E7B71D266F19E95874406378F31        column=info:age, timestamp=1437585906436, value=24
 A6B
 A3E7B71D266F19E95874406378F31        column=info:number, timestamp=1437585906437, value=17098078413
 A6B
 ADA6B34FCFA424DE4460D6E8452DB        column=info:address, timestamp=1437585906467, value=shanghai
 FD4
 ADA6B34FCFA424DE4460D6E8452DB        column=info:age, timestamp=1437585906467, value=24
 FD4
 ADA6B34FCFA424DE4460D6E8452DB        column=info:number, timestamp=1437585906468, value=17098078419
 FD4
 ED3273AD8CEBAFBFB8E8CEF6D95A2        column=info:address, timestamp=1437585906435, value=shanghai
```



注以 number 作 MD5 加密之后的结果作为 hbase 的 rowkey 如图所示

11、至此一个 agent 已经配置完成，大功告成！

六、文档附录

附：Hbase 相关简单命令

```
scan 'test_flume'
disable 'test_flume'
drop 'test_flume'
truncate ‘test_flume'
create 'test_flume',{NAME=>'info'}
get 'test_flume','F5C226EB1AA7C7027A939FEBEDF98C95'
```

附：maven 相关简单命令

Maven 转成 eclipse 项目
mvn eclipse:eclipse
去除 maven 项目的生成
如 target 包下的 jar 之类的
mvn clean
编译 java 文件
mvn compile
生成 jar 包，项目打包跳过 test
mvn package -Dmaven.test.skip=true -Pdev

附：exec_tail_to_hbase.conf 配置

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = exec
a1.sources.r1.channels = c1
a1.sources.r1.command = tail -F /home/aisino/test/log.txt

# Describe the sink
a1.sinks.k1.type = org.apache.flume.sink.hbase.AsyncHBaseSink
a1.sinks.k1.table = test_flume
a1.sinks.k1.columnFamily = info
a1.sinks.k1.serializer.payloadColumn = age,address,number
```

```
a1.sinks.k1.serializer.suffix = custom
#custom mean set by yourself
a1.sinks.k1.serializer.rowPrefixCol = number
a1.sinks.k1.serializer=    org.apache.flume.sink.hbase.AisinoAsyncHbaseEventSerializer
a1.sinks.k1.channel = memoryChannel

# 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
```

附：AisinoAsyncHbaseEventSerializer 代码

```
package org.apache.flume.sink.hbase;

/*
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.    See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.    The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.    You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.    See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */

import java.util.ArrayList;
import java.util.List;

import org.apache.flume.Context;
import org.apache.flume.Event;
```

```
import org.apache.flume.FlumeException;
import org.hbase.async.AtomicIncrementRequest;
import org.hbase.async.PutRequest;
import org.apache.flume.conf.ComponentConfiguration;
import org.apache.flume.sink.hbase.SimpleHbaseEventSerializer.KeyType;

import com.google.common.base.Charsets;

public class CAisinoAsyncHbaseEventSerializer implements AsyncHbaseEventSerializer {
    private byte[] table;
    private byte[] cf;
    private byte[][] payload;
    private byte[][] payloadColumn;
    private final String payloadColumnSplit = ",";
    private byte[] incrementColumn;
    private String rowSuffix;
    private String rowSuffixCol;
    private byte[] incrementRow;
    private KeyType keyType;

    @Override
    public void initialize(byte[] table, byte[] cf) {
      this.table = table;
      this.cf = cf;
    }

    @Override
    public List<PutRequest> getActions() {
      List<PutRequest> actions = new ArrayList<PutRequest>();
      if(payloadColumn != null){
        byte[] rowKey;
        try {
          switch (keyType) {
            case TS:
              rowKey = SimpleRowKeyGenerator.getTimestampKey(rowSuffix);
              break;
            case TSNANO:
              rowKey = SimpleRowKeyGenerator.getNanoTimestampKey(rowSuffix);
              break;
            case RANDOM:
              rowKey = SimpleRowKeyGenerator.getRandomKey(rowSuffix);
              break;
            case CUSTOM:
```

```
                    rowKey = SimpleRowKeyGenerator.getCustomKey(rowSuffix);
                    break;
                default:
                    rowKey = SimpleRowKeyGenerator.getUUIDKey(rowSuffix);
                    break;
            }
        for (int i = 0; i < this.payload.length; i++)
        {
                    PutRequest putRequest = new PutRequest(table, rowKey,
cf,payloadColumn[i], payload[i]);
                    actions.add(putRequest);
        }

          } catch (Exception e){
            throw new FlumeException("Could not get row key!", e);
          }
        }
        return actions;
    }

    public List<AtomicIncrementRequest> getIncrements(){
        List<AtomicIncrementRequest> actions = new
            ArrayList<AtomicIncrementRequest>();
        if(incrementColumn != null) {
          AtomicIncrementRequest inc = new AtomicIncrementRequest(table,
              incrementRow, cf, incrementColumn);
          //actions.add(inc);
        }
        return actions;
    }

    @Override
    public void cleanUp() {
        // TODO Auto-generated method stub

    }

    @Override
    public void configure(Context context) {
        String pCol = context.getString("payloadColumn", "pCol");
        String iCol = context.getString("incrementColumn", "iCol");
        rowSuffixCol = context.getString("rowPrefixCol", "rowkey");
        String suffix = context.getString("suffix", "uuid");
        if(pCol != null && !pCol.isEmpty()) {
```

```
        if(suffix.equals("timestamp")){
            keyType = KeyType.TS;
        } else if (suffix.equals("random")) {
            keyType = KeyType.RANDOM;
        } else if(suffix.equals("nano")){
            keyType = KeyType.TSNANO;
        }else if(suffix.equals("custom")){
            keyType = KeyType.CUSTOM;
        }else {
            keyType = KeyType.UUID;
        }
            String[] pCols = pCol.replace(" ", "").split(",");
            payloadColumn = new byte[pCols.length][];
            for (int i = 0; i < pCols.length; i++)
    {
            payloadColumn[i] = pCols[i].toLowerCase().getBytes(Charsets.UTF_8);
    }
    }

    if(iCol != null && !iCol.isEmpty()) {
        incrementColumn = iCol.getBytes(Charsets.UTF_8);
    }
    incrementRow =
            context.getString("incrementRow", "incRow").getBytes(Charsets.UTF_8);
}

@Override
public void setEvent(Event event) {
    String strBody = new String(event.getBody());
    String[] subBody = strBody.split(this.payloadColumnSplit);
    if (subBody.length == this.payloadColumn.length)
    {
            this.payload = new byte[subBody.length][];
            for (int i = 0; i < subBody.length; i++)
            {
                    this.payload[i] = subBody[i].getBytes(Charsets.UTF_8);
                    if ((new String(this.payloadColumn[i]).equals(this.rowSuffixCol)))
                    {
                            this.rowSuffix = subBody[i];
                    }
            }
    }
}
```

```
    @Override
    public void configure(ComponentConfiguration conf) {
        // TODO Auto-generated method stub
    }


}
```