**Flume自定义开发案例**

**案例1：Avro Client**

MyApp.java

**import** org.apache.flume.Event;

**import** org.apache.flume.EventDeliveryException;

**import** org.apache.flume.api.RpcClient;

**import** org.apache.flume.api.RpcClientFactory;

**import** org.apache.flume.event.EventBuilder;

**import** java.nio.charset.Charset;

**public** **class** MyApp {

**public** **static** **void** main(String[] args) {

MyRpcClientFacade client = **new** MyRpcClientFacade();

// Initialize client with the remote Flume agent's host and port

client.init("host.example.org", 41414);

// Send 10 events to the remote Flume agent. That agent should be

// configured to listen with an AvroSource.

String sampleData = "Hello Flume!";

**for** (**int** i = 0; i < 10; i++) {

client.sendDataToFlume(sampleData);

}

client.cleanUp();

}

}

**class** MyRpcClientFacade {

**private** RpcClient client;

**private** String hostname;

**private** **int** port;

**public** **void** init(String hostname, **int** port) {

// Setup the RPC connection

**this**.hostname = hostname;

**this**.port = port;

**this**.client = RpcClientFactory.*getDefaultInstance*(hostname, port);

// Use the following method to create a thrift client (instead of the above line):

// this.client = RpcClientFactory.getThriftInstance(hostname, port);

}

**public** **void** sendDataToFlume(String data) {

// Create a Flume Event object that encapsulates the sample data

Event event = EventBuilder.*withBody*(data, Charset.*forName*("UTF-8"));

// Send the event

**try** {

client.append(event);

} **catch** (EventDeliveryException e) {

// clean up and recreate the client

client.close();

client = **null**;

client = RpcClientFactory.*getDefaultInstance*(hostname, port);

// Use the following method to create a thrift client (instead of the above line):

// this.client = RpcClientFactory.getThriftInstance(hostname, port);

}

}

**public** **void** cleanUp() {

// Close the RPC connection

client.close();

}

}

**#配置文件：**case\_client.conf

a1.channels = c1

a1.sources = r1

a1.sinks = k1

a1.channels.c1.type = memory

a1.sources.r1.channels = c1

a1.sources.r1.type = avro

# For using a thrift source set the following instead of the above line.

# a1.source.r1.type = thrift

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 41414

a1.sinks.k1.channel = c1

a1.sinks.k1.type = logger

**#开始命令**

flume-ng agent -c conf -f conf/case\_client.conf -n a1 -Dflume.root.logger=INFO,console

**案例2：Custom Sink (目的:将获取的数据存入mysql中，还没有实现。)**

MySink.java

**import** org.apache.flume.Channel;

**import** org.apache.flume.Context;

**import** org.apache.flume.Event;

**import** org.apache.flume.EventDeliveryException;

**import** org.apache.flume.Transaction;

**import** org.apache.flume.conf.Configurable;

**import** org.apache.flume.sink.AbstractSink;

**public** **class** MySink **extends** AbstractSink **implements** Configurable {

**private** String myProp;

@Override

**public** **void** configure(Context context) {

String myProp = context.getString("myProp", "defaultValue");

// Process the myProp value (e.g. validation)

// Store myProp for later retrieval by process() method

**this**.myProp = myProp;

}

@Override

**public** **void** start() {

// Initialize the connection to the external repository (e.g. HDFS) that

// this Sink will forward Events to ..

}

@Override

**public** **void** stop () {

// Disconnect from the external respository and do any

// additional cleanup (e.g. releasing resources or nulling-out

// field values) ..

}

@Override

**public** Status process() **throws** EventDeliveryException {

Status status = **null**;

// Start transaction

Channel ch = getChannel();

Transaction txn = ch.getTransaction();

txn.begin();

**try** {

// This try clause includes whatever Channel operations you want to do

Event event = ch.take();

// Send the Event to the external repository.

// storeSomeData(e);

txn.commit();

status = Status.*READY*;

} **catch** (Throwable t) {

txn.rollback();

// Log exception, handle individual exceptions as needed

status = Status.*BACKOFF*;

// re-throw all Errors

**if** (t **instanceof** Error) {

**throw** (Error)t;

}

} **finally** {

txn.close();

}

**return** status;

}

}

**#配置文件：**case\_custom\_sink.conf

a1.channels = c1

a1.sources = r1

a1.sinks = k1

a1.channels.c1.type = memory

a1.sources.r1.channels = c1

a1.sources.r1.type = avro

# For using a thrift source set the following instead of the above line.

# a1.source.r1.type = thrift

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 41414

a1.sinks.k1.channel = c1

a1.sinks.k1.type = com.flume.test.MySink

**#开始命令**

flume-ng agent -c conf -f case\_custom\_sink.conf -n a1 -Dflume.root.logger=INFO,console

**案例3：custom source**

MySource.java

**import** java.util.List;

**import** org.apache.flume.Channel;

**import** org.apache.flume.Context;

**import** org.apache.flume.Event;

**import** org.apache.flume.EventDeliveryException;

**import** org.apache.flume.PollableSource;

**import** org.apache.flume.Transaction;

**import** org.apache.flume.conf.Configurable;

**import** org.apache.flume.source.AbstractSource;

**public** **class** MySource **extends** AbstractSource **implements** Configurable, PollableSource {

**private** String myProp;

@Override

**public** **void** configure(Context context) {

String myProp = context.getString("myProp", "defaultValue");

// Process the myProp value (e.g. validation, convert to another type, ...)

// Store myProp for later retrieval by process() method

**this**.myProp = myProp;

}

@Override

**public** **void** start() {

// Initialize the connection to the external client

}

@Override

**public** **void** stop () {

// Disconnect from external client and do any additional cleanup

// (e.g. releasing resources or nulling-out field values) ..

}

@Override

**public** Status process() **throws** EventDeliveryException {

// **TODO** Auto-generated method stub

List<Channel> list = getChannelProcessor().getSelector().getAllChannels();

Status status = **null**;

// Start transaction

Channel ch = list.get(0);

Transaction txn = ch.getTransaction();

txn.begin();

**try** {

// This try clause includes whatever Channel operations you want to do

// Receive new data

Event e = **null**;//getSomeData();EventBuilder.withBody(data, Charset.forName("UTF-8"), headerMap);EventBuilder.withBody(data, Charset.forName("UTF-8"));

// Store the Event into this Source's associated Channel(s)

getChannelProcessor().processEvent(e);

txn.commit();

status = Status.*READY*;

} **catch** (Throwable t) {

txn.rollback();

// Log exception, handle individual exceptions as needed

status = Status.*BACKOFF*;

// re-throw all Errors

**if** (t **instanceof** Error) {

**throw** (Error)t;

}

} **finally** {

txn.close();

}

**return** status;

}

}

**#配置文件：**case\_custom\_source.conf

a1.channels = c1

a1.sources = r1

a1.sinks = k1

a1.channels.c1.type = memory

a1.sources.r1.channels = c1

a1.sources.r1.type = com.wy.test.MySource

a1.sinks.k1.channel = c1

a1.sinks.k1.type = logger

**#开始命令**

flume-ng agent -c conf -f case\_custom\_source.conf -n a1 -Dflume.root.logger=INFO,console