Flume16 自定义Sink

介绍

Sink不断地轮询 Channel中的事件且批量地移除它们，并将这些事件批量写入到存储或者索引系统，或者被发送到另一个Flume Agent

Sink是完全事务性地，在从Channel批量删除数据之前，每个Sink用Channel启动一个事务，批量事件一旦成功写出道存储系统或下一个Flume Agent，Sink就利用Channel提交事务。事务一旦提交，该Channel从自己的内部缓冲区删除事件。

Sink组件的目的地包括 hdfs、logger、avro、thrift、ipc、file、null、HBase、solr、自定义。官方提供的Sink类型已经很多，但是有时候并不能满足实际开发当中的需求，此时我们就需要根据实际需求自定义某些Sink。

官方也提供了自定义sink接口：

http://flume.apache.org/FlumeDeveloperGuide.html#sink

根据官方说明自定义MySink需要继承AbstractSink类并实现Configurable接口

实现相应方法：

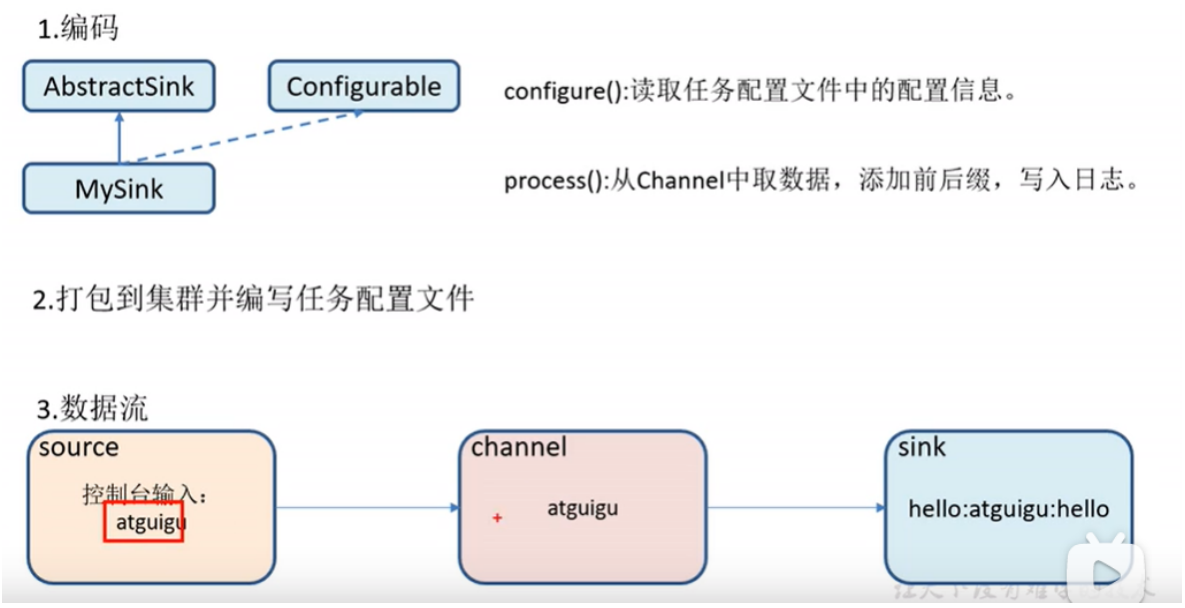
- configure(Context context) //初始化 context (读取配置文件内容)
- process() // 从Channel读取获取数据(event) 这个方法将被循环调用

使用场景：读取Channel数据写入MySQL或者其他文件系统。

```java
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;
      }
    }
    return status;
  }
}
```

需求

使用Flume 接收数据，并在Sink端给每条数据添加前缀和后缀，前后缀可在flume任务配置文件中设置


1.编码

configure():读取任务配置文件中的配置信息。

process():从Channel中取数据，添加前后缀，写入日志。

2.打包到集群并编写任务配置文件

3.数据流

步骤设置

- 写代码:

```
public class MySink extends AbstractSink implements Configurable {

    private Logger logger = Logger.Factory.getLogger(MySink.class);

    //定义两个属性，前后缀
    private String prefix;
    private String subfix;

    private String myProp;

    @Override
    public void configure(Context context) {

        //读取配置文件，并且赋值
        prefix = context.getString("prefix");
        subfix = context.getString("subfix", "atguigu");
    }

  /**
  *1. 获取channel
  *2. 从channel获取事务和数据
  *3. 发送数据
  */

    @Override
    public Status process() throws EventDeliveryException {
        //定义返回值
        Status status = null;

        // 得到channel
        Channel ch = getChannel();
        Transaction txn = ch.getTransaction(); //获取事务
        txn.begin(); //开启事务
        try {
            //获取数据
            Event event = ch.take();

            //业务逻辑修改
            if(event != null){

                //处理事务
                String body = new String(event.getBody);
                logger.info(prefix+body+subfix);
            }
            //提交事务
            txn.commit();

            //成功提交修改状态信息
            status = Status.READY;
        } catch (Throwable t) {
            //提交失败就回滚
```

```
                txn.rollback();
                //修改状态
                status = Status.BACKOFF;

                if (t instanceof Error) {
                    throw (Error)t;
                }
            }finally{
                transaction.close();
            }
            //返回状态
            return status;
        }
    }
```

打包，上传到 opt/module/flume/lib/下

- 添加配置文件

```
# 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 = com.atguigu.sink.MySink
a1.sinks.k1.prefix = sleep--
a1.sinks.k1.subfix = --banzhang

# 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
```

- 打开服务

```
bin/flume-ng agent -c conf/ -f job/mysink.conf -n a1 -
Dflume.root.loger=INFO.console
```

- 测试

```
nc localhost 44444
shazhu
```