**Flume监控Log4j日志**

方案一：Flume远程监控Windows端口

方案二：使用Linux中的Eclipse，这样生成的日志文件就在Linux上。（不推荐）

方案三：Windows和Linux共享文件夹

**1、Eclipse配置Log4j2**

**（1）pom.xml添加依赖**

|  |
| --- |
| <!-- log4j2 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter</artifactId>  <!--exclusions部分是为了将spring boot自带的log4j配置替换掉 是为了能够得到更为清晰的日志信息 -->  <exclusions>  <exclusion>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-logging</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-log4j2</artifactId>  </dependency>    <!-- flume -->  <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-flume-ng</artifactId>  <version>2.8.2</version>  </dependency>  <dependency>  <groupId>org.apache.flume.flume-ng-clients</groupId>  <artifactId>flume-ng-log4jappender</artifactId>  <version>1.6.0</version>  </dependency> |

**（2）清除SpringBoot启动时在控制台显示的图标**

因为测试过SpringBoot的图标的日志级别好像非常高，所以就不用接下来的设定日志显示级别来过滤掉了，直接在这里清除它。

在application.java中：

|  |
| --- |
| @SpringBootApplication  public class Application {    public static void main(String[] args) {  //SpringApplication.run(Application.class, args);  //清除spring boot的图标  SpringApplication springApplication = new SpringApplication(Application.class);  springApplication.setBannerMode(Banner.Mode.OFF);  springApplication.run(args);  }  } |

**（3）加入log4j2的配置**

在src/main/resources下面添加log4j2-spring.xml文件（加载log4j2的时候，会自动到当前目录下搜索这个文件名，如果更改名字，则需去application.properties中配置）

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!--日志级别以及优先级排序: OFF > FATAL > ERROR > WARN > INFO > DEBUG > TRACE > ALL -->  <!--Configuration后面的status，这个用于设置log4j2自身内部的信息输出，可以不设置。当设置成trace时，会看到log4j2内部各种详细输出 -->  <!--monitorInterval：Log4j能够自动检测修改配置 文件和重新配置本身，设置间隔秒数 -->  <configuration status="WARN" monitorInterval="30">    <!--先定义所有的appender -->  <appenders>  <!--输出控制台的配置 -->  <console name="Console" target="SYSTEM\_OUT">  <!--<JSONLayout compact="true" eventEol="true"/> -->  <!--输出日志的格式：只用了取具体日志的格式，去掉了多余信息的显示，例如时间和类 -->  <!-- %n用于换行-->  <PatternLayout pattern="%m%n" />  </console>    <!--输出到flume -->  <!--flume中XX.conf配置文件的type也要改为avro -->  <Flume name="flume" compress="false" type="Avro">  <Agent host="virhost01" port="44444" />  <PatternLayout pattern="%m%n" />  </Flume>  </appenders>    <!--然后定义logger，只有定义了logger并引入的appender，appender才会生效 -->  <loggers>  <!--过滤掉spring和mybatis的一些无用的warn级别以下的信息 -->  <!--因为接下来输出的日志信息设置为了warn级别，不想被以下的信息干扰 -->  <logger name="org.springframework" level="warn"></logger>  <logger name="org.mybatis" level="warn"></logger>  <!--这里设置的是全局的日志显示级别 -->  <root level="warn">  <appender-ref ref="Console" />  <appender-ref ref="flume" />  </root>  </loggers>    </configuration> |

**（4）写日志信息**

在你想要输出日志的类中，加载日志信息配置，getLogger(类名.class)：

**private static final** org.slf4j.Logger ***log*** = org.slf4j.LoggerFactory.*getLogger*(UserController.**class**);

然后就可以在该类中用***log***.warn()来写日志了。

可以先测试一下，程序运行到这里有没有输出日志里面的值，如果报错，把之前有关的flume配置信息都注释掉。

**2、配置Flume -> HDFS（用于离线日志）**

（1）修改Flume配置文件

|  |
| --- |
| a4.sources = r4  a4.channels = c4  a4.sinks = k4    # Configuration the sources  a4.sources.r4.type = avro  a4.sources.r4.bind = virhost01  a4.sources.r4.port = 44444    # Configuration the channels：使用内存缓冲区域做数据的临时缓存  a4.channels.c4.type = memory  a4.channels.c4.capacity = 10000  a4.channels.c4.transactionCapacity = 100    # Describe the sink  a4.sinks.k4.type = hdfs  # 设置文件存放目录，而不是文件名  a4.sinks.k4.hdfs.path = hdfs://virhost01:9000/project/shop/%Y-%m-%d/  # 使文件名中%y-%m-%d生效  a4.sinks.k4.hdfs.useLocalTimeStamp = true  # 设置日志文件前缀为log  a4.sinks.k4.hdfs.filePrefix=log  ## 设置文件隔多久发送，默认30s  ## rollInterval设置为60s，表明60s内的日志会写在一个hdfs的文件中，过60s就会写在一个新的文件中。  a4.sinks.k4.hdfs.rollInterval=60    a4.sinks.k4.hdfs.fileType = DataStream  a4.sinks.k4.hdfs.writeFormat = Text  a4.sinks.k4.hdfs.batchSize = 10    a4.sinks.k4.maxBytesToLog = 4096  ## 使文件的日志信息不会出现多余空行  a4.sinks.k4.serializer.appendNewline = false    #bind channels  a4.sources.r4.channels = c4  a4.sinks.k4.channel = c4 |

（2）运行flume

cd /opt/cdh/apache-flume-1.6.0-cdh5.9.3-bin

bin/flume-ng agent --conf conf --n a4 --f conf/flume-log4j.conf -Dflume.root.logger=DEBUG,console

（3）注意：

配置后若只开Eclipse会报如下的错：

org.apache.flume.FlumeException: No active client.

所以要先开flume再开eclipse；

所有在Console控制台中输出的WARN级别及以上的错误，都会被写入到HDFS的文件中。

**3、配置Flume -> Kafka（用于实时日志）**

（1）启动zookeeper

zkServer.sh start

（2）启动kafka服务

cd /opt/cdh/kafka\_2.11-2.3.1

bin/kafka-server-start.sh config/server.properties

（3）创建Kafka主题（topic）

查看现有topic

bin/kafka-topics.sh --zookeeper virhost01:2181 --list

若没有topic，则创建一个topic

bin/kafka-topics.sh --zookeeper virhost01:2181 --create --replication-factor 1 --partitions 3 --topic test01

（4）修改Flume配置文件

|  |
| --- |
| a1.sources = r1  a1.channels = c1  a1.sinks = k1    # Configuration the sources  a1.sources.r1.type = avro  a1.sources.r1.bind = virhost01  a1.sources.r1.port = 44444    # Configuration the channels：：使用内存缓冲区域做数据的临时缓存  a2.channels.c2.type = memory  a2.channels.c2.capacity = 10000  a2.channels.c2.transactionCapacity = 100    # Describe the sink  a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink  # 设置topic，test01是在kafka中创建的topic的名字  a1.sinks.k1.kafka.topic = test01  # 设置kafka的地址和端口号  a1.sinks.k1.kafka.bootstrap.servers = virhost01:9092  # 使文件的日志信息不会出现多余空行  a1.sinks.k1.sink.serializer.appendNewline=false    #bind channels  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1 |

（5）启动Flume（充当生产者）

cd /opt/cdh/apache-flume-1.6.0-cdh5.9.3-bin

bin/flume-ng agent --conf conf --n a1 --f conf/flume-kafka.conf -Dflume.root.logger=DEBUG,console

（6）启动消费者客户端

cd /opt/cdh/kafka\_2.11-2.3.1

bin/kafka-console-consumer.sh --bootstrap-server virhost01:9092 --topic test01

（7）注意：输出在kafka中的日志信息会出现一行日志，一行空行的情况，但在使用Spark Streaming 统计时不会有影响。

a1.sinks.k1.serializer.appendNewline = false

这个配置可以在hdfs的配置中去掉多余空行，但在kafka的配置中没有效果，但在使用Spark Streaming 统计时不会有影响。

**4、问题**

启动项目，第一次输出日志信息后，eclipse的控制台上会打印出Flume多余信息：

|  |
| --- |
| zsd（正常日志信息，以下为多余内容） Using default maxIOWorkers  2019-12-18 11:21:41,484 http-nio-8888-exec-8 ERROR Recursive call to appender flume |

百度查询如何消除无果，但这两行警告并不会输出到HDFS和Kafka。