## Spring-boot的logback集成logstash

本教程在以下前提条件下进行演示：

1).Elk环境已

2).springboot-maven工程

3).kafka环境

### 1.直接与logtash集成

主要思路：logback转成JSON格式数据，通过TCP传输到logstash中

#### 1.1 需要将logback日志解析成JSON日志，pom中需要引入:

<!-- logstash -->

<dependency>

<groupId>net.logstash.logback</groupId>

<artifactId>logstash-logback-encoder</artifactId>

<version>4.6</version>

</dependency>

更多使用方法参考 ：<https://github.com/logstash/logstash-logback-encoder>

#### 1.2 pom添加环境变量，即依据环境(test/dev/prod)连接logstash

<profiles>

<profile>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<id>dev</id>

<properties>

<env>dev</env>

<!-- dev logtash server -->

<logstashServer>39.108.174.117:4560</logstashServer>

</properties>

</profile>

<profile>

<id>test</id>

<properties>

<env>test</env>

<!-- test logtash server -->

<logstashServer>39.108.174.117:4560</logstashServer>

</properties>

</profile>

</profiles>

#### 1.3 logback配置

在bootstrap.yml中添加如下配置：

spring:

application:

name: omni-member-service

logstash:

server: @logstashServer@

说明：

1).Logback的加载要在application.yml之前，在bootstrap.yml之后，所以我们需要将spring.application.name的配置放置到bootstrap.yml文件中，如果不存在该文件，请创建一个

2).spring.logstash.server，此处我们取的是pom中对应的环境配置，用于logback中配置的tcp服务

1.3：logback配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <configuration scan=*"true"*>  <include  resource=*"org/springframework/boot/logging/logback/defaults.xml"* />  <!-- logback加载优先于application.yml文件，因此需要从bootstrap.yml文件中读取spring.application.name信息 -->  <springProperty scope=*"context"* name=*"springAppName"*  source=*"spring.application.name"* />  <!-- spring bootstrap配置文件中读取 logtash服务信息 -->  <springProperty scope=*"context"* name=*"logtashServer"*  source=*"spring.logstash.server"* />  <!-- 日志在工程中的输出位置 -->  <property name=*"LOG\_FILE"*  value=*"${BUILD\_FOLDER:-build}/${springAppName}"* />  <!-- 控制台的日志输出样式 -->  <property name=*"CONSOLE\_LOG\_PATTERN"*  value=*"%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr(${LOG\_LEVEL\_PATTERN:-%5p}) %clr([${springAppName:-},%X{X-B3-TraceId:-},%X{X-B3-SpanId:-},%X{X-Span-Export:-}]){yellow} %clr(${PID:- }){magenta} %clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint} %m%n${LOG\_EXCEPTION\_CONVERSION\_WORD:-%wEx}"* />  <!-- 控制台Appender -->  <appender name=*"console"*  class=*"ch.qos.logback.core.ConsoleAppender"*>  <filter class=*"ch.qos.logback.classic.filter.ThresholdFilter"*>  <level>INFO</level>  </filter>  <encoder>  <pattern>${CONSOLE\_LOG\_PATTERN}</pattern>  <charset>utf8</charset>  </encoder>  </appender>  <!-- 通过Tcp Socket输出到logstash服务端 -->  <appender name=*"logstash-tcp"*  class=*"net.logstash.logback.appender.LogstashTcpSocketAppender"*>  <destination>${logtashServer}</destination>  <!-- 调整适当大小，解决丢包问题。单位：B -->  <queueSize>1048576</queueSize>  <encoder  class=*"net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder"*>  <providers>  <timestamp/>  <pattern>  <pattern>  {  "level":"%level",  "service": "${springAppName:-}",  "trace": "%X{X-B3-TraceId:-}",  "span": "%X{X-B3-SpanId:-}",  "stack\_trace": "%exception{10}",  "req\_id": "%X{reqId}",  "elapsed\_time": "#asLong{%X{elapsedTime}}",  "pid": "${PID:-}",  "thread": "%thread",  "class": "%logger{40}",  "method\_name": "%method",  "message": "%message"  }  </pattern>  </pattern>  </providers>  </encoder>  </appender>  <logger name=*"org.springframework.web"* level=*"ERROR"* />  <!-- 仅针对member包的日志输出到logstash中 -->  <logger name=*"com.purcotton.omni.member"* level=*"info"*  additivity=*"false"*>  <appender-ref ref=*"logstash-tcp"* />  </logger>  <root level=*"INFO"*>  <appender-ref ref=*"console"* />  </root>  </configuration> |

#### 1.4 logstash配置

|  |
| --- |
| input {      tcp {          port => 4560  //4560端口监听          codec => json\_lines          type => "tcp-logback"          mode => "server"      }  }  //输出到es  output {      elasticsearch {           hosts => ["172.18.31.202:9200"]          index => "%{[serviceName]}-%{+YYYY-MM-dd}"      }  } |

*注意：*

*1). Logstash没有安装logstash-codec-json\_lines插件, bin/logtash-plugin install logstash-codec-json\_lines*

*2). %{[serviceName]}-%{+YYYY-MM-dd} 中的serviceName是从logback传过来的变量，此处使用服务名称+日期来建立索引*

### 2.与kafka、logstash集成

主要思路：logback将日志送到kafka，logstash再从kafka收集日志，解耦logback与logstash。

#### 2.1 添加pom依赖

|  |
| --- |
| <!-- logstash -->  <dependency>  <groupId>net.logstash.logback</groupId>  <artifactId>logstash-logback-encoder</artifactId>  <version>4.11</version>  </dependency>  <dependency>  <groupId>com.github.danielwegener</groupId>  <artifactId>logback-kafka-appender</artifactId>  <version>0.1.0</version>  <scope>runtime</scope>  </dependency> |

#### 2.2 添加pom环境变量

|  |
| --- |
| <profiles>  <profile>  <id>dev</id>  <activation>  <activeByDefault>true</activeByDefault>  </activation>  <properties>  <env>dev</env>  <!-- dev logtash server -->  <logstashServer>39.108.174.117:4560</logstashServer>  <!-- dev kafka server -->  <kafkaServer>120.78.246.199:9092</kafkaServer>  <kafkaLogTopic>omniLogstash</kafkaLogTopic>  </properties>  </profile>  <profile>  <id>test</id>  <properties>  <env>test</env>  <!-- test logtash server -->  <logstashServer>39.108.174.117:4560</logstashServer>  <!-- test kafka server -->  <kafkaServer>120.78.246.199:9092</kafkaServer>  <kafkaLogTopic>omniLogstash</kafkaLogTopic>  </properties>  </profile>  </profiles> |

#### 2.3 bootstrap.yml配置文件添加 kafka配置信息

|  |
| --- |
| spring:  logstash:  server: @logstashServer@  kafka-servers: @kafkaServer@  logs-topic: @kafkaLogTopic@ |

#### 2.4 logback.xml配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <configuration scan=*"true"*>  <include  resource=*"org/springframework/boot/logging/logback/defaults.xml"* />  <!-- logback加载优先于application.yml文件，因此需要从bootstrap.yml文件中读取spring.application.name信息 -->  <springProperty scope=*"context"* name=*"springAppName"*  source=*"spring.application.name"* />  <!-- spring bootstrap配置文件中读取 logtash服务信息 -->  <springProperty scope=*"context"* name=*"logtashServer"*  source=*"spring.logstash.server"* />  <!-- kafka配置 -->  <springProperty scope=*"context"* name=*"kafkaServer"*  source=*"spring.logstash.kafka-servers"* />  <springProperty scope=*"context"* name=*"kafkaLogTopic"*  source=*"spring.logstash.logs-topic"* />  <!-- 日志在工程中的输出位置 -->  <property name=*"LOG\_FILE"*  value=*"${BUILD\_FOLDER:-build}/${springAppName}"* />  <!-- 控制台的日志输出样式 -->  <property name=*"CONSOLE\_LOG\_PATTERN"*  value=*"%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr(${LOG\_LEVEL\_PATTERN:-%5p}) %clr([${springAppName:-},%X{X-B3-TraceId:-},%X{X-B3-SpanId:-},%X{X-Span-Export:-}]){yellow} %clr(${PID:- }){magenta} %clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint} %m%n${LOG\_EXCEPTION\_CONVERSION\_WORD:-%wEx}"* />  <!-- 控制台Appender -->  <appender name=*"console"*  class=*"ch.qos.logback.core.ConsoleAppender"*>  <encoder>  <pattern>${CONSOLE\_LOG\_PATTERN}</pattern>  <charset>utf8</charset>  </encoder>  </appender>  <!-- 通过Tcp Socket输出到logstash服务端 -->  <appender name=*"logstash-tcp"*  class=*"net.logstash.logback.appender.LogstashTcpSocketAppender"*>  <destination>${logtashServer}</destination>  <!-- 调整适当大小，解决丢包问题。单位：B -->  <queueSize>1048576</queueSize>  <encoder  class=*"net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder"*>  <providers>  <timestamp />  <pattern>  <pattern>  {  "level":"%level",  "service": "${springAppName:-}",  "trace": "%X{X-B3-TraceId:-}",  "span": "%X{X-B3-SpanId:-}",  "stack\_trace": "%exception{10}",  "req\_id": "%X{reqId}",  "elapsed\_time": "#asLong{%X{elapsedTime}}",  "pid": "${PID:-}",  "thread": "%thread",  "class": "%logger{40}",  "method\_name":  "%method",  "message": "%message"  }  </pattern>  </pattern>  </providers>  </encoder>  </appender>  <!-- 发送kafka -->  <appender name=*"kafkaAppender"*  class=*"com.github.danielwegener.logback.kafka.KafkaAppender"*>  <encoder  class=*"com.github.danielwegener.logback.kafka.encoding.LayoutKafkaMessageEncoder"*>  <layout class=*"net.logstash.logback.layout.LogstashLayout"*>  <includeContext>true</includeContext>  <includeCallerData>true</includeCallerData>  <includeMdc>true</includeMdc>  <!-- 自定义属性 -->  <!-- <customFields>{"serverName":"${springAppName}"}</customFields> -->  <fieldNames  class=*"net.logstash.logback.fieldnames.ShortenedFieldNames"* />  </layout>  <charset>UTF-8</charset>  </encoder>  <!--kafka topic 需要与配置文件里面的topic一致 否则kafka会沉默并鄙视你 -->  <topic>${kafkaLogTopic}</topic>  <!-- 路由分区策略，可依据HostName、ContextName、ThreadName、LoggerName，默认NoKey -->  <!-- <keyingStrategy -->  <!-- class="com.github.danielwegener.logback.kafka.keying.NoKeyKeyingStrategy"  /> -->  <!-- 交付策略 一种熔断策略(连接不是上默认发送到控制台/AsynchronousDeliveryStrategy)，一种阻塞策略(直到发送成功/BlockingDeliveryStrategy) -->  <deliveryStrategy  class=*"com.github.danielwegener.logback.kafka.delivery.AsynchronousDeliveryStrategy"* />  <!-- kafka连接地址 -->  <producerConfig>bootstrap.servers=${kafkaServer}</producerConfig>  <!-- 当消息不能发送时，发送到默认控制台，不阻塞主线程 -->  <appender-ref ref=*"console"* />  </appender>  <!-- <logger name="org.springframework.web" level="ERROR" /> -->  <!-- 仅针对member包的日志输出-->  <logger name=*"com.purcotton.omni.member"* level=*"info"*  additivity=*"false"*>  <!-- 到logstash中 -->  <!-- <appender-ref ref="logstash-tcp" /> -->  <!-- 输出到kafka中 -->  <appender-ref ref=*"kafkaAppender"* />  </logger>  <root level=*"debug"*>  <appender-ref ref=*"console"* />  </root>  </configuration> |

#### 2.5 logback配置

|  |
| --- |
| input {      kafka {          bootstrap\_servers => ["172.18.31.201:9092,172.18.31.202:9092,172.18.31.203:9092"]          consumer\_threads => 5          topics => ["omniLogstash"]          group\_id => "omniLogstash"          client\_id => "omniLogstash"          auto\_offset\_reset => "latest" //从最新的偏移量开始消费          decorate\_events => true          type => "bhy"  //所有插件通用属性,尤其在input里面配置多个数据源时很有用      }  }  filter {      // 对json格式数据进行转换      json {         source => "message"         #remove\_field => ["message"]      }  }  output {      // es配置，此处暂不能获取到appname      elasticsearch {          hosts => ["172.18.31.202:9200"]          index => "logstash-kafka-%{+YYYY-MM-dd}"      }  } |