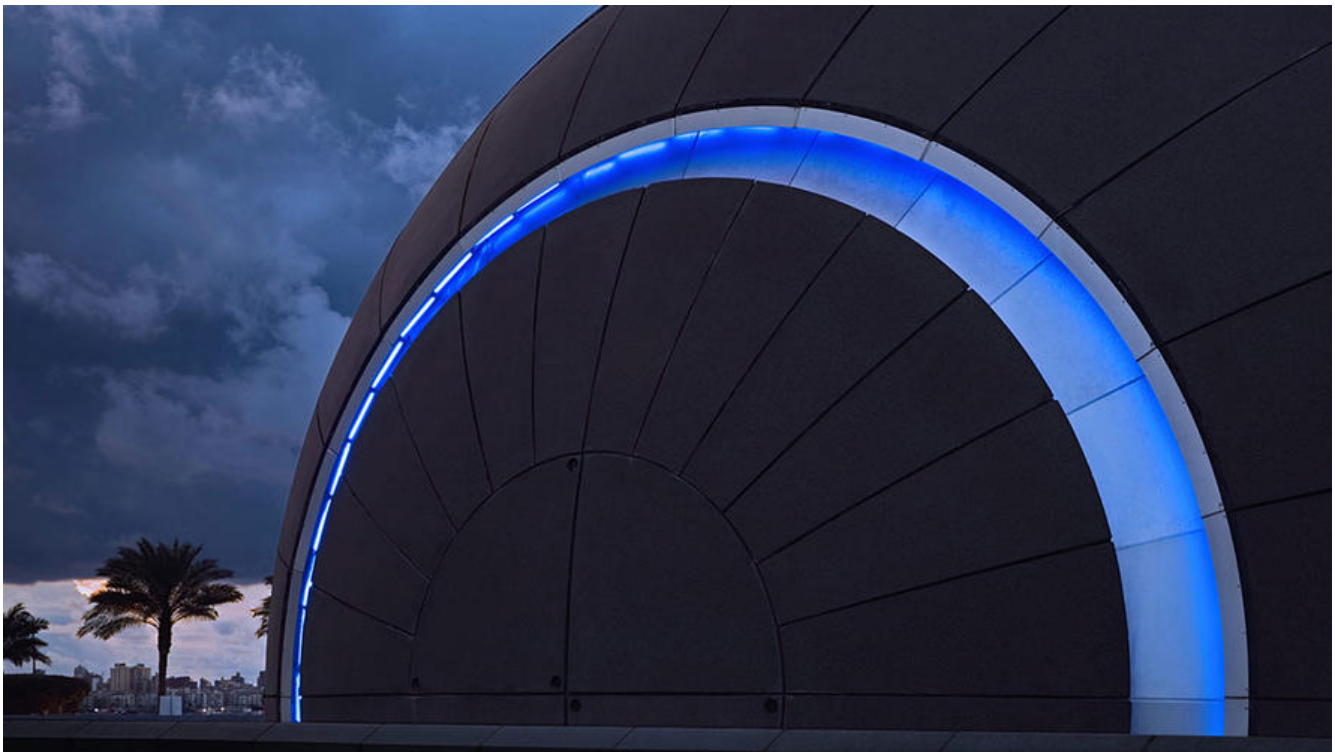
链滴

SkyWalking Agent 端日志插件的编写历程与使用说明

作者：vcjmhg

原文链接：https://ld246.com/article/1607866513663

来源网站：链滴



概述

前一段时间顺利完成了SkyWalking Agent端logger-plugin插件的开发，在此做个总结。一方面给插的使用方法写一中文说明，另一方面分享一下该插件开发过程中的一些考量以及收获。

logger-plugin插件，主要作用实现将将程序在调用过程中产生的日志比如错误日志信息，存入到span log中。然后可以通过web端直接查询，便于开发者排错与分析。同时为了提高使用的灵活性，我们提供了配置文件，通过配置文件可以对需要存入到span log的日志来源（log4j2、logbak、log4j）包名、日志级别、内容进行控制。但同时需要提醒的一点是，由于该插件直接作用于agent端，可能造成一定程度的**性能损失**，请谨慎使用。

相关PR地址如下：Support collecting logs of log4j, log4j2, and logback in the tracing context 以供参考。

使用方式

基于性能的考量，该插件在设计之初定位为可选插件，因而默认情况下该插件的功能并不会启动。如使用，需要在8.4.0发布之后，将插件从apache-skywalking-apm-bin/agent/optional-plugins/目下复制到apache-skywalking-apm-bin/agent/plugins/下，方能生效。

配置文件

首先需要说明的一点是，发布时默认是没有配置文件的，插件会按照如下默认配置内容运行:

<pre spellcheck="false" class="md-fences md-end-block md-fences-with-lineno ty-contain-m modeLoaded" lang="properties" cid="n20" mdtype="fences"> <span role="presentation" log4j.packages=*</span
 log4j.level=error
 lg4j2.packages=*
 <span role="pres

ntation">log4j2.level=error</spa
>
 logback.packages</spa
>=*
 <span clas
="cm-def">logback.level=error</pre>

上述配置文件含义如下：

1. 该插件默认会对所有支持的日志框架生效，包括log4j、log4j2、logbback。
2. 插件之后将高于 error级别的日志信息存入到span log中，而对trace, debug, info, warn级别的志信息不生效。
3. 默认情况下，会收集所有包级别的日志信息。

.如果需要自定义插件的配置信息，需要在apache-skywalking-apm-bin/agent/config/logger-plugn/目录下创建logconfig.properties文件进行配置。

配置文件可配置内容及其含义如下：

packages

含义：指定需要转存的日志信息的包名，默认匹配所有包。

取值：

- 包名：例如org.apache。若有多个包名中间需用逗号隔开
- *:匹配所有包级别的日志信息。

Level

含义：所需转存的日志信息的级别，默认情况是error级别的日志信息

日志级别从小到大排列如下：

trace < debug < info <warn< error < fatal

同时我们在自定义配置信息的时候需要注意，由于logback不支持fatal级别的日志信息，因而如果错的进行如下配置：

<pre spellcheck="false" class="md-fences md-end-block md-fences-with-lineno ty-contain-m modeLoaded" lang="properties" cid="n72" mdtype="fences"> <span role="presentation" logback.level=error</spa
></pre>

则会造成针对logback的配置失效，也即不会收集任何关于logback日志插件产生的日志信息，并会生错误日志，以供用户排查。

设计思路

其实这个插件刚开始插件之初，由于个人工程经验不足，当时考虑想要实现功能很多，比如需要支持则表达式用以过滤，支持日志的格式化配置等等。但在和其他社区成员的讨论过程中发现，需要功能实和该插件的定位不同，比如过滤日志信息的功能，日志系统中已经支持了。如果该插件再增加该功一方面会增加插件使用的复杂程度，另一方面，也会造成更大的性能损耗。而针对另一功能日志的格化，这与该插件的定位也是不符合的，apm系统本身就是为了实现追踪和快速定位，对收集到的日志

息希望尽量简练，有效，许多格式化的信息，根本无需收集。因而该功能最终也被废止。

同时针对日志转存到span的格式，和社区成员也进行仔细的沟通，为了适应未来SkyWalking的日志询功能，因此在转存日志信息时，存储到OAP端的span log日志是结构化的类似于下面的形式：

<pre mdtype="fences" cid="n81" lang="json" class="md-fences md-end-block md-fences-wih-lineno ty-contain-cm modeLoaded" spellcheck="false"> "logs": [
   {
   "time": <spa class="cm-number">1605225365711,
  "data": [
   {
   "key": "event",
   "value": "warn"
   },
   {
   "key": "log.kind",
 <pan role="presentation"> "value": "org.apache.skywalking.oap.server.analyzer.provider.trace.parser.listener.ultiScopesAnalysisListener"
   }, br/>   {
   "key": "message", /span>
   "value : "span {} has illegal status code {}"
   },
   {
   "key": <san class="cm-string">"param.1",
   <san class="cm-string cm-property">"value": "*span*"</spa >
   },
   {
   " ey": "param.2",
   "value": <span class="cm-string >"*tag.getValue()*"
   }
 <pan role="presentation">]
   }
 <san role="presentation">]</pre>

便于后续Web端实现灵活的查询。关于日志格式详细的讨论过程，可以参考《What format shoulde have in mind for logging to spans?》

同时在设计的过程中，刚开始找到插入点也是不合理的，比如刚开始选择的直接增强sl4j.Logger接口这回造成一个问题，一旦用户没有使用Log4j创建Logger对象就会造成日志插件失效，最终在BFergeron的帮助下，换成了一个更加合理的插入点。

总结

这篇文章更多的可能是个人在开发logger-plugin插件过程中踩过坑的总结吧，在整个开发过程中，实收获蛮多的，比如并非一个软件功能越多越好。这个需要从该软件本身的定位出发，兼顾性能和易性，做到一种平衡。同时简单写了写logger-plugin插件的使用方法，希望能够给感兴趣的同学以帮。