

arthas 使用教程

作者: [Gakkiyomi2019](#)

原文链接: <https://ld246.com/article/1617011164563>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

 <https://b3logfile.com/bing/20171222.jpg?imageView2/1/w/960/h/540/interlace/1/q/100>

arthas-使用教程

Arthas 是 Alibaba 开源的 Java 诊断工具，通过这个工具我们可以快速并准确的定位问题。

这里会简单介绍使用 arthas 进行 Troube Shooting，详细安装流程与深入的操作命令参数请自查阅官方文档学习。:pray:

官方文档: [https://arthas.aliyun.com/doc/index.html](https://ld246.com/forward?goto=https%3A%2F%2Ffarthas.aliyun.com%2Fdoc%2Findex.html)

资源监控

arthas 提供了 CPU，内存，运行环境等资源信息面板,并且可以设置刷新时间

命令

dashboard

 <https://b3logfile.com/file/2021/03/image-87b8067c.png?imageView2/2/interlace/1/format/jpg>

heapdump

dump java heap, 类似 jmap 命令的 heap dump 功能。

查看 java 堆内存的对象实例情况

`heapdump /arthas-output/nap.hprof` 将堆的使用情况 dump 到 /arthas-output/nap.hprof 文件

可以使用 jdk 自带的 `VisualVM` 来查看

 <https://b3logfile.com/file/2021/03/image-fa22cc7e.png?imageView2/2/interlace/1/format/jpg>

也可以使用 IntelliJ IDEA 自带的 HPROF 内存查看器 查看 输入 Action `Open profiler napshot`

 <https://b3logfile.com/file/2021/03/image-b5ed4494.png?imageView2/2/interlace/1/format/jpg>

Profiler

arthas 支持监控一段时间的火焰图来统计性能热点。

`profiler start` 启动搜集 CPU 的使用情况

`profiler stop` 从 start 到 stop 这个时间段的 CPU 火焰图

 <https://b3logfile.com/file/2021/03/image-3ae478b4.png?imageView2/2/interlace/1/format/jpg>

火焰图里，X 轴越长,代表使用的越多,Y 轴是调用堆栈信息。当前收集的是么类型的数据，比如 cpu 那么 x 轴长度越大,占用的 cpu 资源就越多。

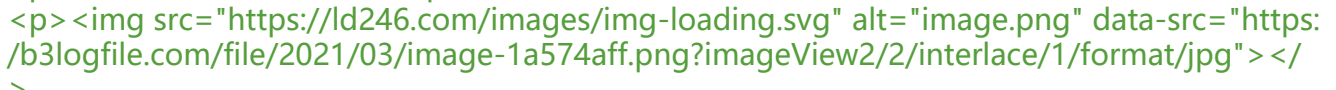
thread

查看当前线程信息，查看线程的堆栈

参数名称	参数说明
------	------

<code>id</code>	线程 id
<code>[n:]</code>	指定最忙的前 N 个线程并打印堆栈
<code>[b]</code>	找出当前阻塞其他线程的线程
<code>[i <code>&lt;value&gt;</code>]</code>	指定 cpu 使用率统计的采样间隔，单位为毫秒，默认值为 200
<code>[--all]</code>	显示所有匹配的线程

查看当前最吃资源的线程

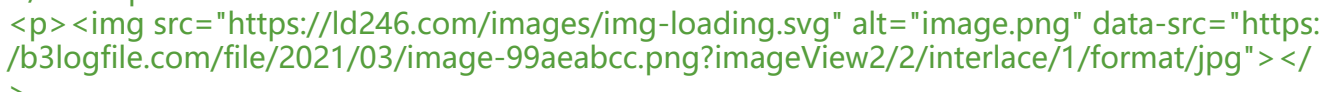
 <https://b3logfile.com/file/2021/03/image-1a574aff.png?imageView2/2/interlace/1/format/jpg>

方法调用时间追踪

命令-

trace

> 使用 trace 命令可以针对某一类的某一方法进行方法调用时间的追踪,精确定位到问题所在。
>
> 使用建议：
>
> - 首先使用 thread 命令查看当前高占用线程堆栈定位到方法
> - 然后使用 trace 监控方法进行用时追踪

 <https://b3logfile.com/file/2021/03/image-99aeabcc.png?imageView2/2/interlace/1/format/jpg>

在线热更新

arthas 允许我们修改一个正在运行的字节码文件，也允许我们从外部加载一个新的 class 文件来盖旧的。这样我们可以实现在线热部署，这样可以临时性的解决客户生产环境产生的问题,例如空指针常 joy:。

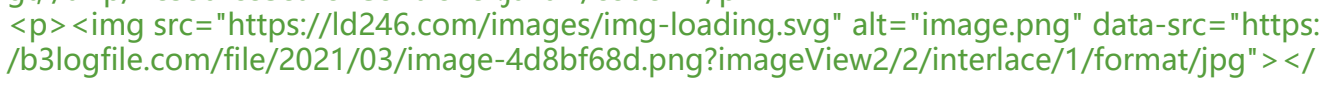
命令--

jad/mc/retransform 组合可实现热更新

jad

反编译代码，将已加载的字节码文件反编译到容器内文件夹。现场操作人员可以直接修复问题(果问题够简单)

`jad --source-only net.skycloud.cmdb.resource.api.rest.ResourceSearchController; /tmp/ResourceSearchController.java`

 <https://b3logfile.com/file/2021/03/image-4d8bf68d.png?imageView2/2/interlace/1/format/jpg>

```
>
<p>修改反编译出来的 class 文件 (除法分母不能为 0)</p>
<p></p>
>
<h4 id="mc">mc</h4>
<p>Memory Compiler/内存编译器，编译 <code>.java</code> 文件生成 <code>.class</code>。这也是我们可以从 <strong>外部加载 java 文件(打补丁)</strong> 的基础。</p>
<p>若前方技术支持无法解决，可以让后端伙伴进行 bug 修复，然后传输修复后的 java 文件来以解眉之急。不需要重新生成镜像。</p>
<p><code>mc /tmp/ResourceSearchController.java -d /tmp</code></p>
<p></p>
<p>
<h4 id="retransform">retransform</h4>
<p>加载外部的 <code>.class</code> 文件，retransform(重新转换) jvm 已加载的类。</p>
<p><code>retransform /tmp/net/skycloud/cmdb/resource/api/rest/ResourceSearchController.class</code></p>
<p></p>
>
<h5 id="查看已更新的类">查看已更新的类</h5>
<p><code>retransform -l</code></p>
<p></p>
>
<h5 id="恢复原来的类">恢复原来的类</h5>
<p>这两个命令一起执行</p>
<p><code>retransform --deleteAll</code></p>
<p><code>retransform --classPattern net.skycloud.cmdb.resource.api.rest.ResourceSearchController</code></p>
<h2 id="Troube-Shooting">Troube Shooting</h2>
<p>作为在现场的工作人员需要至少需要掌握以上命令来进行诊断。troube shooting 流程如下。</p>
>
<p></p>
>
```