## Arthas打印火焰图

### 启动profiler

开启arthas

java -jar arthas-boot.jar

### 开启火焰图

$ profiler start

Started [cpu] profiling

默认启动的是对CPU的收集。可以分析一个系统中性能耗时，占用时间比较多的方法。

也可以使用--event参数指定要采样的事件。例如：

# 对内存进行分析

$ profiler start --event alloc

可以使用下面命令看对event的支持。

$ profiler list

生成的svg文件，可以使用sz命令下载下来，直接使用浏览器打开。

# 获取已采集的sample的数量

$ profiler getSamples

# 查看profiler状态

$ profiler status

$ profiler stop --format html

输出文件格式

profiler stop --format html --file /home/profiler/arthas/20211230-1540.html

推荐输出图像格式

profiler stop --format svg --file /home/profiler/arthas/20211231-1111-atms.svg

### 关闭profiler

默认情况下，生成的结果保存到应用的工作目录下的arthas-output目录。可以通过 --file参数来指定输出结果路径。比如：

$ profiler stop --file /tmp/output.svg

profiler output file: /tmp/output.svg

OK

## 集成springboot

<dependency>

<groupId>com.taobao.arthas</groupId>

<artifactId>arthas-spring-boot-starter</artifactId>

<version>3.6.2</version>

</dependency>

application-dev.yml 中配置如下:

*# arthas tunnel server配置***arthas**:  
 **agent-id**: maomao  
 **tunnel-server**: ws://127.0.0.1:7777/ws

启动项目后，浏览器中输入:

<http://localhost:3658/>

或者

<http://localhost:8563/>

或者

cmd窗口中

telnet localhost 3658

{

**arthas.telnet-port**: -1:则不监听telnet端口。0:随机端口。

**arthas.http-port**: -1: 不监听http端口。0:随机端口。

为了防止一台机器上启动多个arthas端口冲突，可以配置为随机端口0或者-1，然后通过tunnel server的方式来访问arthas。

}

### 使用arthas-tunnel-server

<https://blog.csdn.net/xixingzhe2/article/details/124273198>

1. 下载:

<https://github.com/alibaba/arthas/releases>

2. 运行

windows

java -jar arthas-tunnel-server-3.6.2-fatjar.jar

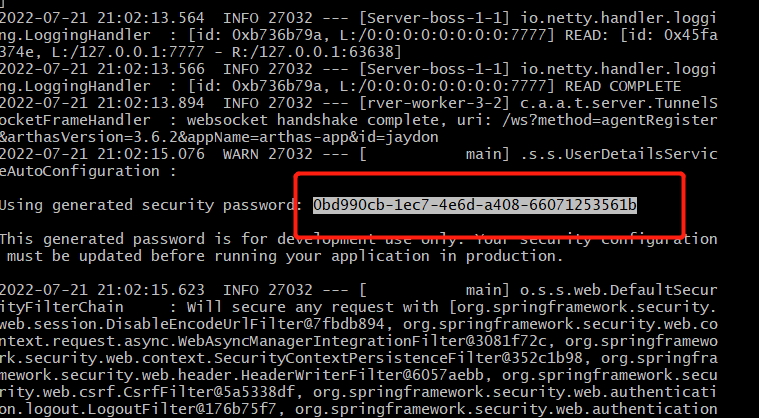
linux

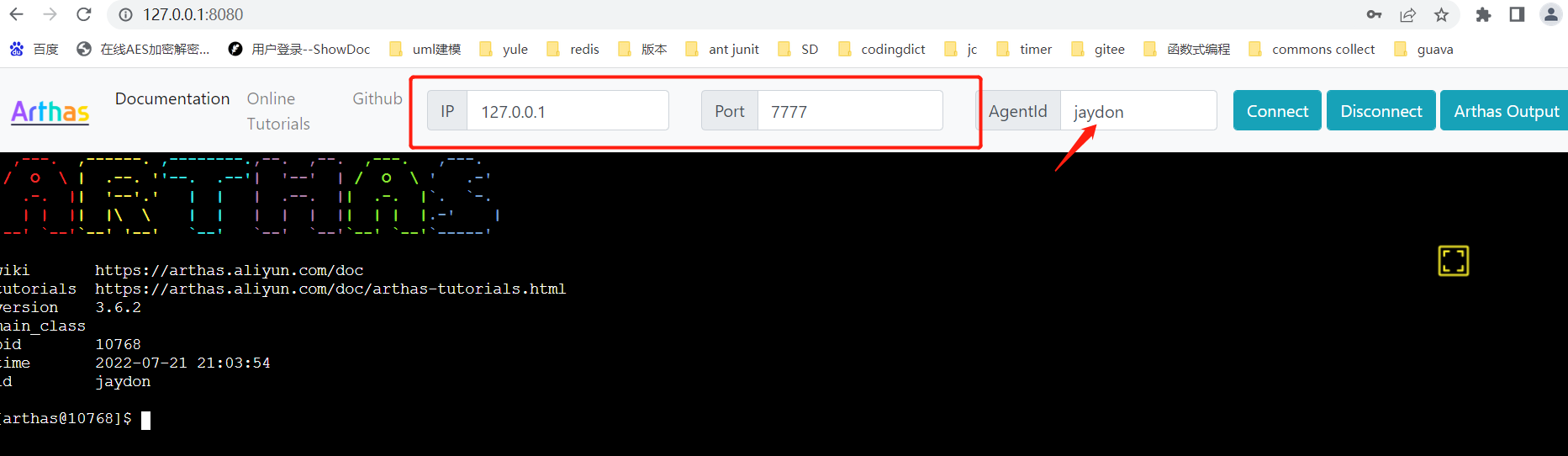
nohup java -jar arthas-tunnel-server-3.6.2-fatjar.jar > /dev/null 2>&1 &

3.登录查看注册上来的应用

http://127.0.0.1:8080/actuator/arthas 登陆用户名是arthas

密码在arthas tunnel server的日志里可以找到，比如：





默认情况下，arthas tunnel server的web端口是8080，arthas agent连接的端口是7777。

启动之后，可以访问 http://127.0.0.1:8080/actuator/arthas ，再通过agentId连接到已注册的arthas agent上

### 自定义tunnel-server端口

1、集成到springboot的项目中配置如下

**arthas**:  
 *# -1:则不监听telnet端口。0:随机端口* **telnetPort**: 3658  
 *# -1: 不监听http端口。0:随机端口* **httpPort**: 8563  
 **agent-id**: jaydon  
 *# 127.0.0.1只能本地访问，0.0.0.0则可网络访问，但是存在安全问题* **ip**: 0.0.0.0  
 *# appName* **appName**: arthas-app  
 *# tunnel-server地址***tunnel-server: ws://172.18.30.108:7878/ws** *# seconds* **sessionTimeout**: 1800  
 *# 禁掉stop 和 dump 命令* **disabledCommands**: stop,dump

2、修改 arthas-tunnel-server-3.6.2-fatjar.jar 中的配置



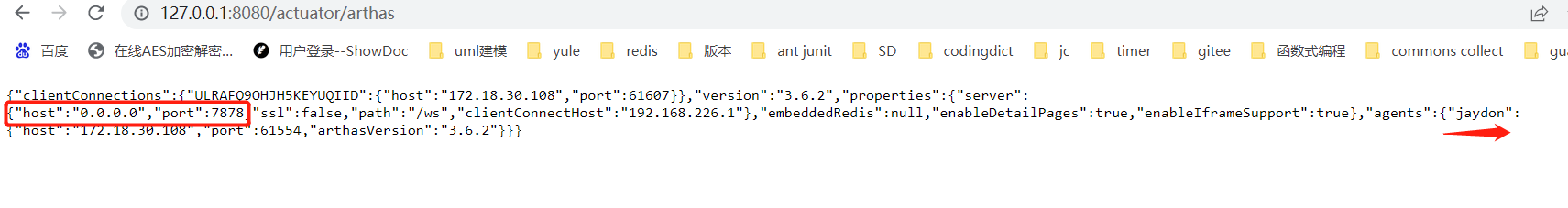
(arthas-tunnel-server必须部署在和集成了arthas的springboot服务在同一个主机中，允许 arthas-tunnel-server后才会在主机上监听 websocket的7878端口)



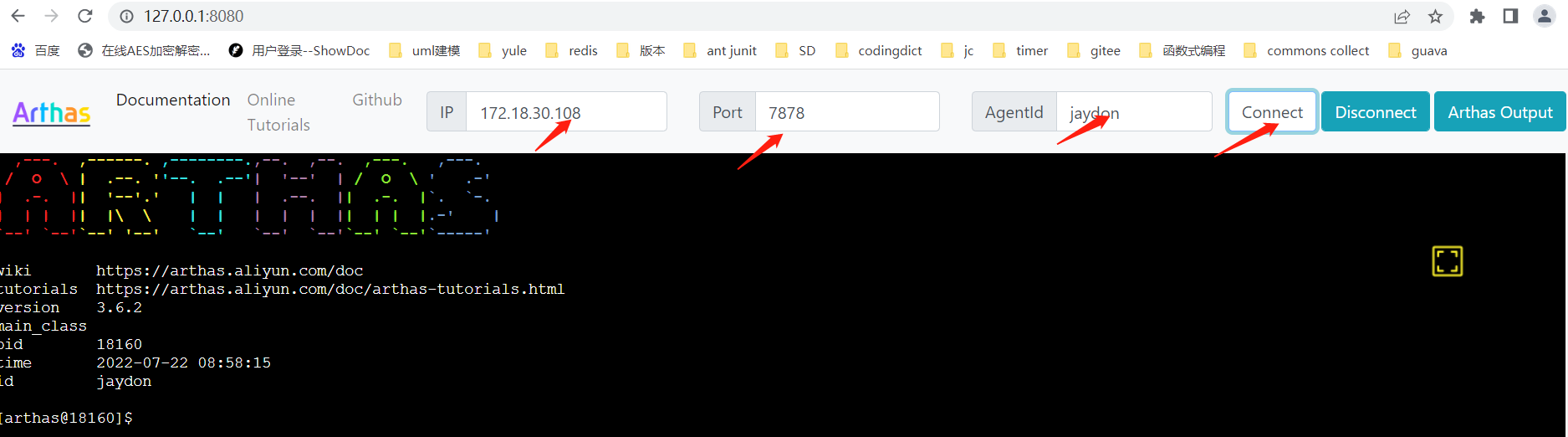
3、启动已经集成arthas的springboot项目

4、启动 arthas-tunnel-server-3.6.2-fatjar.jar

启动之后，可以访问 <http://127.0.0.1:8080/actuator/arthas>



访问  http://127.0.0.1:8080



## arthas命令

### trace命令

[arthas@4387]$ trace -h

USAGE:

trace [--exclude-class-pattern <value>] [-h] [-n <value>] [--listenerId <value>] [-p <value>] [-E] [--skipJDKMethod <value>] [-v] class-pattern metho

d-pattern [condition-express]

SUMMARY:

Trace the execution time of specified method invocation.

The express may be one of the following expression (evaluated dynamically):

target : the object

clazz : the object's class

method : the constructor or method

params : the parameters array of method

params[0..n] : the element of parameters array

returnObj : the returned object of method

throwExp : the throw exception of method

isReturn : the method ended by return

isThrow : the method ended by throwing exception

#cost : the execution time in ms of method invocation

EXAMPLES:

trace org.apache.commons.lang.StringUtils isBlank

trace \*StringUtils isBlank

trace \*StringUtils isBlank params[0].length==1

trace \*StringUtils isBlank '#cost>100'

trace -E org\\.apache\\.commons\\.lang\\.StringUtils isBlank

trace -E com.test.ClassA|org.test.ClassB method1|method2|method3

trace demo.MathGame run -n 5

trace demo.MathGame run --skipJDKMethod false

trace javax.servlet.Filter \* --exclude-class-pattern com.demo.TestFilter

trace OuterClass$InnerClass \*

WIKI:

https://arthas.aliyun.com/doc/trace

OPTIONS:

--exclude-class-pattern <value> exclude class name pattern, use either '.' or '/' as separator

-h, --help this help

-n, --limits <value> Threshold of execution times

--listenerId <value> The special listenerId

-p, --path <value> path tracing pattern

-E, --regex Enable regular expression to match (wildcard matching by default)

--skipJDKMethod <value> skip jdk method trace, default value true.

-v, --verbose Enables print verbose information, default value false.

<class-pattern> Class name pattern, use either '.' or '/' as separator

<method-pattern> Method name pattern

<condition-express> Conditional expression in ognl style, for example:

TRUE : 1==1

TRUE : true

FALSE : false

TRUE : 'params.length>=0'

FALSE : 1==2

'#cost>100'

获取 HsmCryptor类下面的所有超过500ms的方法，总共1万条：

trace com.tmc.cryptor.hsm.HsmCryptor \* '#cost>500' -n 10000

trace -E com.tmc.cryptor.hsm.HsmCryptor exportRSAKey|exportECKey '#cost>500' -n 10000

### profile命令

[arthas@4387]$ profiler -h

USAGE:

profiler [--allkernel] [--alluser] [-d <value>] [-e <value>] [--exclude <value>] [-f <value>] [--format <value>] [-b <value>] [-h] [--include <value>

] [-i <value>] [--threads] action [actionArg]

SUMMARY:

Async Profiler. https://github.com/jvm-profiling-tools/async-profiler

EXAMPLES:

profiler start

profiler stop

profiler list # list all supported events

profiler actions # list all supported actions

profiler start --event alloc

profiler stop --format html # output file format, support html,jfr

profiler stop --file /tmp/result.html

profiler stop --threads

profiler start --include 'java/\*' --include 'demo/\*' --exclude '\*Unsafe.park\*'

profiler status

profiler resume # Start or resume profiling without resetting collected data.

profiler getSamples # Get the number of samples collected during the profiling session

profiler dumpFlat # Dump flat profile, i.e. the histogram of the hottest methods

profiler dumpCollapsed # Dump profile in 'collapsed stacktraces' format

profiler dumpTraces # Dump collected stack traces

profiler execute 'start,framebuf=5000000' # Execute an agent-compatible profiling command

profiler execute 'stop,file=/tmp/result.html' # Execute an agent-compatible profiling command

WIKI:

https://arthas.aliyun.com/doc/profiler

OPTIONS:

--allkernel include only kernel-mode events

--alluser include only user-mode events

-d, --duration <value> run profiling for <duration> seconds

-e, --event <value> which event to trace (cpu, alloc, lock, cache-misses etc.), default value is cpu

--exclude <value> exclude stack traces containing PATTERN, for example: '\*Unsafe.park\*'

-f, --file <value> dump output to <filename>

--format <value> dump output file format(html, jfr), default valut is html

-b, --framebuf <value> size of the buffer for stack frames (default: 1'000'000)

-h, --help this help

--include <value> include stack traces containing PATTERN, for example: 'java/\*'

-i, --interval <value> sampling interval in ns (default: 10'000'000, i.e. 10 ms)

--threads profile different threads separately

<action> Action to execute

<actionArg> Attribute name pattern.