Jps ：通常用于查询进程id，有了进程id之后才可以使用下面的命令。

Jinfo

Jstat

Jmap:

Jconsole

VisualVM

记一次问题定位：

高毅晓峰私募基金抢购服务的系统调用链路中， 历史系统TTS前置机常出现OOM宕机。

根因🡪Tomcat http线程占用内存过大。

在历史系统行内TTS服务调用中， 并发一上来的时候就会出现OOM，由于该服务本身存在将文字转为语音文件并上传的原因，因此一开始认为是底层服务每次响应一个请求消耗内存过大的原因， 使用线程池进行并发请求控制。 在后来的定位中 使用VisualVM 抽样器观察线程的内存占用情况时，发现tomcat http-nio线程每个可以达到200M左右， 经排查为遗留排至中server.maxHttpHeaderSize设置有误，每个设置为100M，设置原因是由于将请求文本放在param中传递，而请求文本为银行售卖基金的话术，通常在千字左右。将其减小后内存占用显著下降，不再出现OOM。 在后续版本中将其放在request body中传输。

2. 在UAT测试环境，不指定JVM启动参数，可以看到JVM被分配了更大的内存(后续指定参数为1800m), 但却会被OOMKilled？？？？ Docker file中limit 2g.

延申问题：不指定JVM堆大小参数时，启动时默认的堆大小参数是如何计算的。

答：Unless the initial and maximum heap sizes are specified on the command line, they are calculated based on the amount of memory on the machine.

容器配置: CPU2核, 内存2G

未设置堆参数时,JVM的配置：

原因:

The JVM sees the available memory of the machine instead of the memory available only to the Docker container. This can lead to cases where applications running inside the container is killed when tries to use more memory beyond the limits of the Docker container.

结论： 由于未指定JVM堆参数，JVM启动时通过检测机器上剩余可用内存并设置堆大小参数， 而该参数超过了deploy文件中对container的内存限制，因此当内存使用超出限制时，将容器kill并报OOMkilled

参考：

https://dzone.com/articles/why-my-java-application-is-oomkilled