Title: Limit JVM memory consumption in a Docker container

Post Body:

I've got a Spring Boot application implementing a service which I want to run in a Docker container. I've followed the guideline of the official Spring docs which suggest to create a DockerFile similar to this:

```
FROM frolvlad/alpine-oraclejdk8:slim VOLUME /tmp ADD gs-spring-boot-docker-0.1.0.jar app.jar RUN sh -c 'touch /app.jar' ENTRYP
```

Then once the image is pushed to Docker I use Docker Compose to launch it this way:

```
  spring-boot-docker:     ports:     - '80:80'     expose:     - '80'     image: my-repo/spring-boot-docker:0.1.0-SNAPSHOT
```

Here I've got the JAVA_OPTS variable which limits the memory allocation, however, when I execute docker stats spring-boot-docker, the memory taken by the container is excessive (I understand the total memory taken by the JVM might be much more than 64M, but in this case is totally boundless).

I've also tried with the mem_limit param, but this slows down the application noticeably.

Accepted Answer:

After struggling for a while, it seems the JAVA_OPTS variable can be passed to the container when it's based in a Tomcat image, but Spring Boot uses Java itself as the base image.

I've found out this tutorial which solved the problem for me, just modifying the way the process is launched in the DockerFile and adding a JAVA_OPTS variable directly in the ENTRYPOINT:

```
ENTRYPOINT exec java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar
```

This way, the JVM will pick the value from the command itself.

Highest Rated Answer: None