Title: Monitor dockerised Spring Boot application internals via JMX behind a Kubernetes cluster with VisualVM

Post Body:

I want to monitor internals of JVM for my Spring Boot application inside a Docker which is running as a pod in a Kubernetes cluster. But I couldn't find satisfactory answer anywhere even after spending considerable time. I tried referring the accepted answer on this but it was connecting only when my docker was running locally, and ceased to connect while behind a Kubernetes cluster.

Accepted Answer:

Assume I wanted to monitor on port 8001 while my app was serving on 8000. Adding these to my VM options worked fine(VisualVM was showing this process for monitoring) while running Docker locally and mapping port 8001 from my local to Docker(-p 8001:8001)

```
-Dcom.sun.management.jmxremote \ -Djava.rmi.server.hostname=localhost \ -Dcom.sun.management.jmxremote.port=8001 \ -Dcom.sun.m
```

But it didn't work on a pod in remote Kubernetes Cluster. I found this but my requirement was to monitor without going via Service and then by reading couple of other articles I got it working, so collating those steps below for saving someone's time:-

1. Removed VM options mentioned above in the startup script
2. Updated my application.yaml as below where I enabled jmx and added url for running JMXMP Connector server

```
spring:   application:     name: stack-application   jmx:     enabled: true     url: service:jmx:jmxmp://localhost:8001/ serve
```

3. Updated my Kubernetes deployment YAML under Deployment block as:

```
apiVersion: apps/v1 kind: Deployment ----your content----           ports:             - name: stack-app-port               co
```

4. Added following dependency to my pom.xml for downloading JMXMP as after my research I concluded that JMX monitoring over RMI is a tough job and hence JMXMP is everyone's recommendation.

```
        <dependency>             <groupId>org.glassfish.main.external</groupId>             <artifactId>jmxremote_optional-rep
```

5. Created a new class `ConnectorServiceFactoryBeanProvider` which fetches URL from our application.yaml

```
@Configuration public class ConnectorServiceFactoryBeanProvider {     @Value('${spring.jmx.url}')     private String url;
```

6. Build and deploy your docker on Kubernetes and find out the IP address of the pod. You may use `kubectl describe pod` for that on CLI
7. Now to start VisualVM, we also need to add the JMXMP jar downloaded above in classpath. I created an alias to do the same, and since the JAR was downloaded in my local .m2 directory the command looked like this:-

```
alias viz='jvisualvm -cp '$JAVA_HOME:~/.m2/repository/org/glassfish/main/external/jmxremote_optional-repackaged/5.0/jmxremote_
```

8. Now, execute 'viz' or your alias, it'll start the Visual VM application shipped with your Java.
9. Click on the +JMX icon in the toolbar of VisualVM or go to (File -> Add JMX Connection...) add the link

```
service:jmx:jmxmp://<IP address obtained in step 6 above>:8001
```

and Check 'Do not require SSL connection'. Once you hit OK, you should see your remote application internals on VisualVM in a while. Screenshot attached below.

VisualVM screenshot monitoring remote app on 8001

Highest Rated Answer: None