Linux DevOps Lab

# JAVA PLATFORM

05_Extras. Spring Boot

Home task

**Legal Notice:** This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

CONFIDENTIAL | Effective Date: 25-Jan-19

## PREREQUISITES

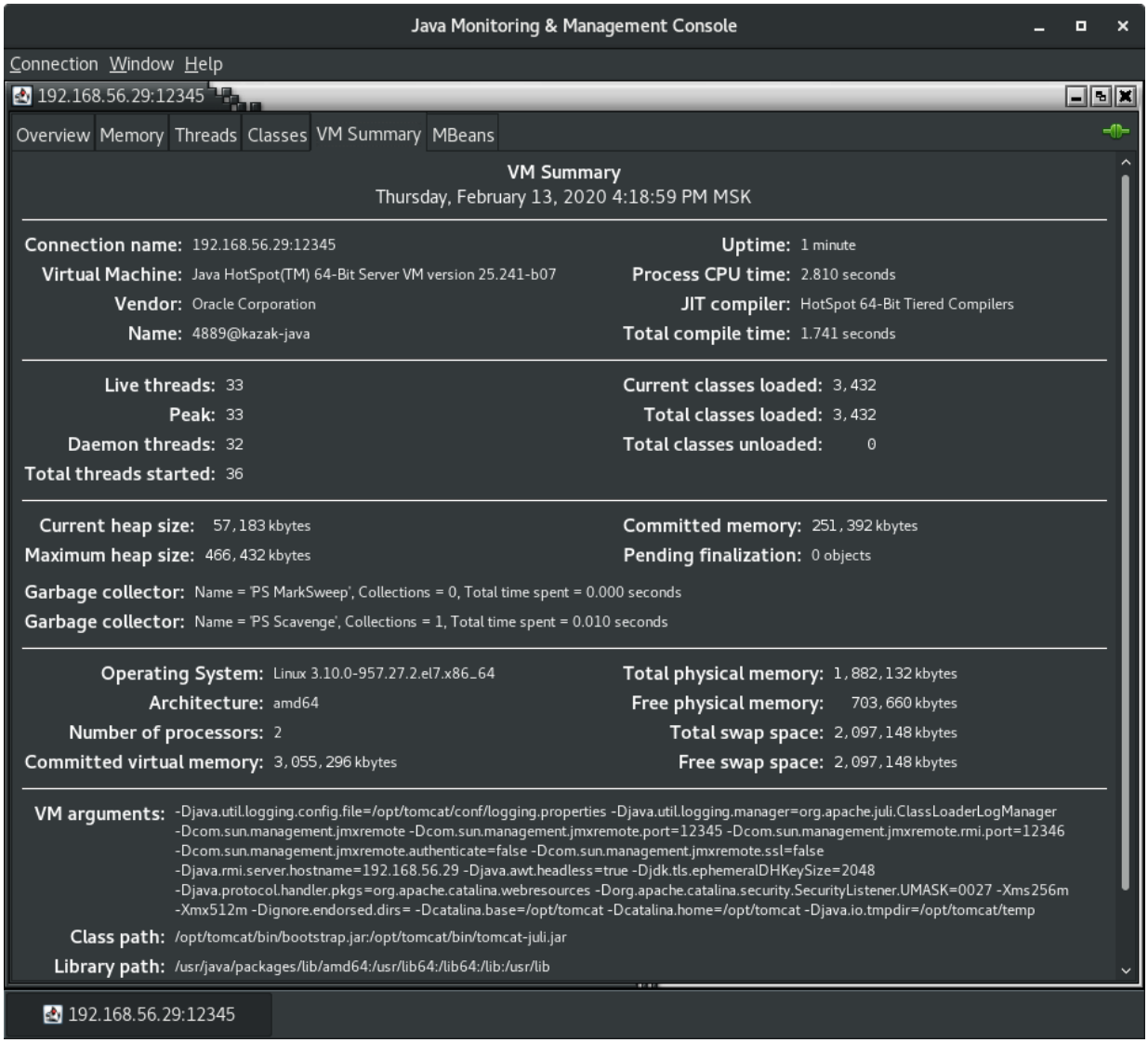
Oracle JDK 1.8 installed

## GOAL

Understand principals of jmx and rmi

## TASK

1) Deploy TestApp.war with next parameters: - Done
   - jmx ports 12345 12346 - Done
   - connect via jconsole and retrieve heap and thread count information - Done




   - write simple java application to gather Tomcat heap information and thread count - Done

2) Press **Parse employees with memory leak** button and upload JSON-file with employees info - Done
3) Run application before and after parsing, describe what you see. - Done

```
student@localhost ~/devopstraining/java/java301/appl ‹master*›
$ java GatherInfo
Number of Threads:      31
Init Heap Memory:       256M
Used Heap Memory:       41M
Commited Heap Memory:   245M
Max Heap Memory:        455M
```

Information is gathered correctly corresponding to the output of jconsole. 41M of heap memory used is ok.

```
student@localhost ~/devopstraining/java/java301/appl ‹master*›
$ java GatherInfo
Number of Threads:      30
Init Heap Memory:       256M
Used Heap Memory:       464M
Commited Heap Memory:   509M
Max Heap Memory:        509M
```

Once MemoryLeak Submit button is pressed, we could see Heap Memory usage is coming to it’s highest value and the GC isn’t managing to clean everything just in time. 464M of heap memory used is definitely NOT ok. Heap Memory usage will grow up to the maximum value and then application will throw an OutOfMemoryError.

Application do works!



CONFIDENTIAL | Effective Date:25-Jan-19