Java and Containers – Make it Awesome !
About Me

- Ex Linux Kernel and glibc hacker
- Eclipse OpenJ9 Cloud Optimization Architect
- Maintain the AdoptOpenJDK community Docker Images for Eclipse OpenJ9 and HotSpot
- Interested in every aspect of running Java Apps in K8s including Cloud Native as well as Legacy migration to Cloud

Dinakar Guniguntala (@dinogun)
IBM Runtime Technologies
Demo Repo

https://github.com/dinogun/watson-springboot
What is AdoptOpenJDK

- AdoptOpenJDK provides prebuilt OpenJDK binaries from a fully open source set of build scripts and infrastructure.
  - https://adoptopenjdk.net
- Extensive testing as part of the CI pipeline.
- REST API for scriptable downloads.
- Docker Images published at https://hub.docker.com/_/adoptopenjdk
Step 1: Begin at the Beginning

Build!
Build Goals

- Size Matters – Small is Better!
  - Affects provisioning / deployment time
  - Consider Images based on Alpine Linux (5 MB vs 80 MB for Ubuntu)
  - Slim images (Cloud workloads use only a small subset)
  - Use jlink to use only the required modules (Java 9 onwards)

- Should be easy to maintain
  - Consider using standard build tool docker images (Eg. maven / gradle)
  - Use build tool plugins for Docker where possible

<table>
<thead>
<tr>
<th>Java Version</th>
<th>Latest</th>
<th>Slim</th>
<th>Alpine</th>
<th>Alpine-Slim</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 (JDK)</td>
<td>338 MB</td>
<td>213 MB</td>
<td>239 MB</td>
<td>98 MB</td>
</tr>
<tr>
<td>11 (JDK)</td>
<td>423 MB</td>
<td>354 MB</td>
<td>324 MB</td>
<td>239 MB</td>
</tr>
</tbody>
</table>

https://hub.docker.com/u/adoptopenjdlk
Step 2: Expand and Flourish

Startup and Scale!
Startup Goals

- Use of a Shared Class Cache helps reduce startup times by up to 40%
  - `-Xshareclasses`

```java
java -Xshareclasses:cacheDir=/opt/shareclasses -jar /opt/app/japp.jar
```
```
docker run -it -v /path/on/host/shareclasses/dir:/opt/shareclasses japp
```

- `-Xquickstart` mode
  - Good for very short running applications
Instant startup with CRIU!!

Instance 1

Startup time

checkpoint

Instance 2

restore

Time to restore

Time to first response

Instance 3

Timeline

Time to first response

https://criu.org
Step 3: Keep it Steady

Container Aware!
Optimize at Runtime

- Container Aware JVM
  - `-XX:+UseContainerSupport` to turn on this feature. (Default now with the latest JVM)
  - Use `-XX:MaxRAMPercentage` and `-XX:InitialRAMPercentage` instead of `-Xmx` and `-Xms`.
  - `Runtime.availableProcessors()` based on cgroup limits.

Kruize: Right size and Optimize your Containers!

https://github.com/kruize/kruize
Step 4: Don’t Lose When You Snooze

Idle Aware!
Optimize at Idle Time

- OpenJ9 reduces footprint on Idle
  - `-XX:+IdleTuningGcOnIdle` and
  - `-XX:+IdleTuningCompactOnIdle`

Step 5: Crashing in the Cloud ? NP !

Javacores and Dumps
Get Debugging Info on the Fly!

- Do you have to restart your containerized Java app to dump debugging info? 
  **NO!**
  - Use OpenJ9DiagnosticMXBean instead

**Method Summary**

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>resetDumpOptions()</td>
</tr>
<tr>
<td></td>
<td>Reset the JVM dump options to the settings specified when the JVM was started removing any additional configuration done since then.</td>
</tr>
<tr>
<td>void</td>
<td>setDumpOptions(java.lang.String dumpOptions)</td>
</tr>
<tr>
<td></td>
<td>This function sets options for the dump subsystem.</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>triggerClassicHeapDump()</td>
</tr>
<tr>
<td></td>
<td>This function triggers the heap dump agent and requests for a heap dump in CLASSIC format.</td>
</tr>
<tr>
<td>void</td>
<td>triggerDump(java.lang.String dumpAgent)</td>
</tr>
<tr>
<td></td>
<td>This function triggers the specified dump agent.</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>triggerDumpToFile(java.lang.String dumpAgent, java.lang.String fileNamePattern)</td>
</tr>
<tr>
<td></td>
<td>This function triggers the specified dump agent.</td>
</tr>
</tbody>
</table>
$ docker run -m2g --cpu-quota="100000" --cpu-period="200000" -it openj9 java
-Xdump:java:events=vmstop App
Useful Links

- **Demo**
  - [https://github.com/dinogun/watson-springboot](https://github.com/dinogun/watson-springboot)

- **Eclipse OpenJ9 Sources**
  - [https://github.com/eclipse/openj9](https://github.com/eclipse/openj9)

- **Kruize** – Right size and optimize your containers
  - [https://github.com/kruize/kruize](https://github.com/kruize/kruize)

- **AdoptOpenJDK Docker Images**
  - [https://hub.docker.com/r/adoptopenjdk/openjdk8-openj9/](https://hub.docker.com/r/adoptopenjdk/openjdk8-openj9/)

- **CRIU** - [https://criu.org](https://criu.org)

@dinogun
Qs
@dinogun