Building Java Applications with Dockerfiles



Esteban Herrera Author | Developer | Consultant

@eh3rrera eherrera.net

Overview



Using Docker

- Dockerfile
- Maven and Gradle images
- Multi-stage builds

Overview

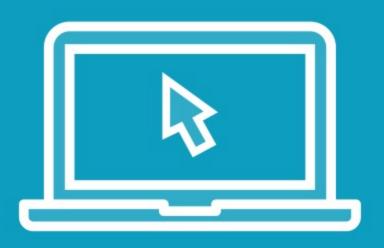


Important Concepts

- Memory and CPU options
- Alternative base images

Using a Dockerfile

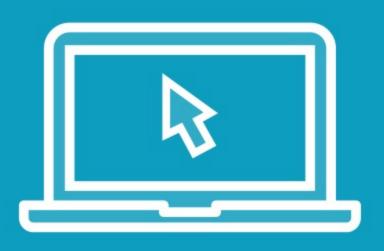
Demo



Run JAR and WAR applications with a Dockerfile

Using Maven and Gradle Docker Images

Demo



Maven and Gradle images

- Dockerfiles
- docker run command

Using Multi-stage Builds

Multi-stage Builds

```
FROM gradle:jdk11
WORKDIR /my-app
COPY app app
RUN gradle build
```

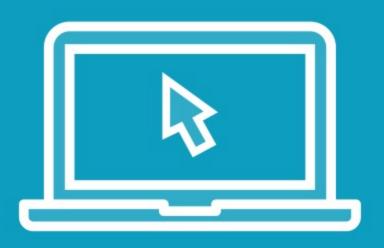
```
FROM openjdk:11
WORKDIR /my-app
COPY build/libs/app.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Multi-stage Builds

```
FROM gradle:jdk11 AS builder
WORKDIR /my-app
COPY app app
RUN gradle build
```

```
FROM openjdk:11
WORKDIR /my-app
COPY --from=builder build/libs/app.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Demo



Multi-stage build for a WAR application

- Maven
- Gradle

Memory and CPU Options in Containers

Control groups (cgroups)

Limit how much resources like CPU time, system memory, or network bandwidth containers can use.

Important Limits for Containers



The amount of memory available



The number of available CPUs



CPU constraints, like shares and quotas

docker run -m 200m my-image

Memory Option for the Docker Run Command

-m, --memory="<number>[<unit>]" Memory limit. Number is a positive integer.

Unit can be one of b, k, m, or g. Minimum is 4M.

```
docker run --cpu-shares=1024 my-image
docker run --cpu-shares=512 my-image
docker run --cpus=1 my-image
docker run --cpu-period=50000 --cpu-quota=25000 my-image
```

CPU Options for the Docker Run Command

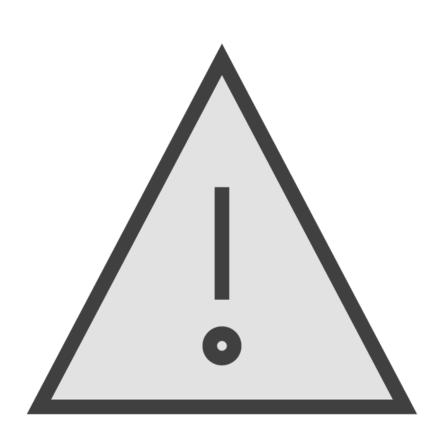
--cpu-shares, -c CPU shares (relative weight)

--cpus Number of CPUs

--cpu-period Limit CPU CFS (Completely Fair Scheduler) period

--cpu-quota Limit CPU CFS (Completely Fair Scheduler) quota

Before Java 8u131



No option to recognize CPU limitations

For memory

Use -Xmx to set the max heap

For Java 9 and Java 8u131+



For CPU these options are automatically set

- -XX:ParallelGCThreads
- -XX:ClCompilerCount

For Memory

- -XX:+UnlockExperimentalVMOptions
- XX:+UseCGroupMemoryLimitForHeap
- XX:InitialRAMFraction
- XX:MaxRAMFraction (defaults to 4)

XX:MaxRAMFraction

Value	Percentage of RAM for the heap
1	100%
2	50%
3	33%
4	25%

For Java 10 and Java 8u191+



Deprecated

- XX:InitialRAMFraction
- XX:MaxRAMFraction
- XX:MinRAMFraction

Added

- -XX:InitialRAMPercentage
- -XX:MaxRAMPercentage
- -XX:MinRAMPercentage

Warning

- UseCGroupMemoryLimitForHeap

For Java 10 and Java 8u191+



-XX:+UseContainerSupport flag is activated by default

The total number of CPUs available to Java is calculated from --cpus, --cpu-shares, --cpu-quota

-XX:ActiveProcessorCount for the number of processors

Java 11+



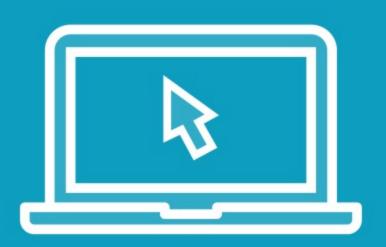
Removed

- -XX:+UseCGroupMemoryLimitForHeap

Added

- -XshowSettings:system (on Linux)
- -XX:+PreferContainerQuotaForCPUCount

Demo



Flags

- UseCGroupMemoryLimitForHeap
- UseContainerSupport
- MaxRAMPercentage

Execute beforehand

- docker pull openjdk:8u131-slim
- docker pull openjdk:8u191-alpine
- docker pull openjdk:11.0.10-slim

Stats.java

```
public class Stats {
    public static void main(String[] args) {
        Runtime rt = Runtime.getRuntime();
        System.out.printf("Heap size: %dMB%n",
                          rt.totalMemory()/1024/1024);
        System.out.printf("Maximum size of heap: %dMB%n",
                          rt.maxMemory()/1024/1024);
        System.out.printf("Available processors: %d%n",
                          rt.availableProcessors());
```

Alternatives for Choosing a Base Image

Oracle JDK Image



License prohibits public distribution

You can only get it using

- Oracle Container Registry
- Docker Store

Alternative Base Images



Azul Zulu

OpenJ9

Create your own image

- From a Linux distribution or another image
- JLink

Summary



Dockerfiles

Maven and Gradle images

Multi-stage builds

Summary



Memory and CPU

- Try to use Java 11 or at least, Java 8u191

Alternative base images

- Oracle JDK
- Azul Zulu
- OpenJ9
- Create your own image

Up Next:

Building Java Applications with Build Tools and Plugins