27. Oktober 2020
JCON

Java & Spring Boot im Container

INNOQ

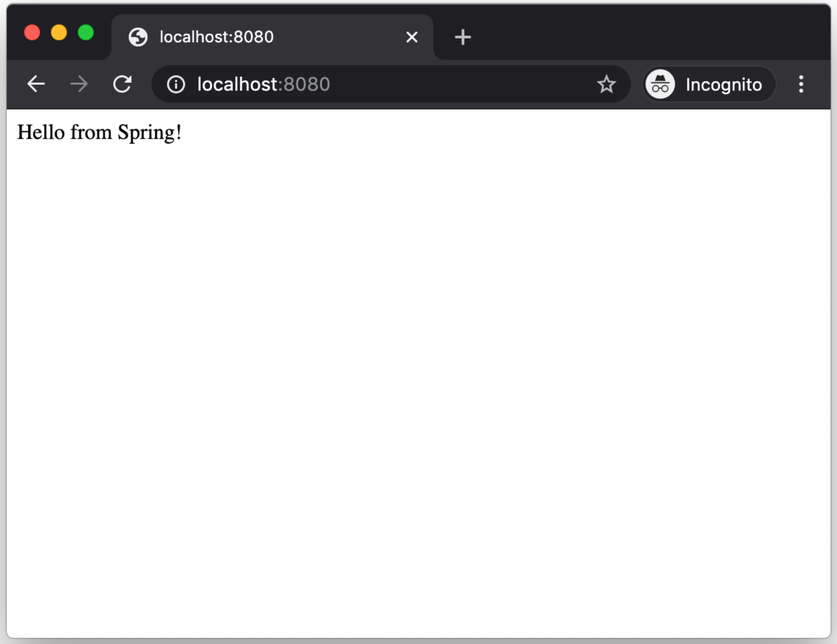
MICHAEL VITZ

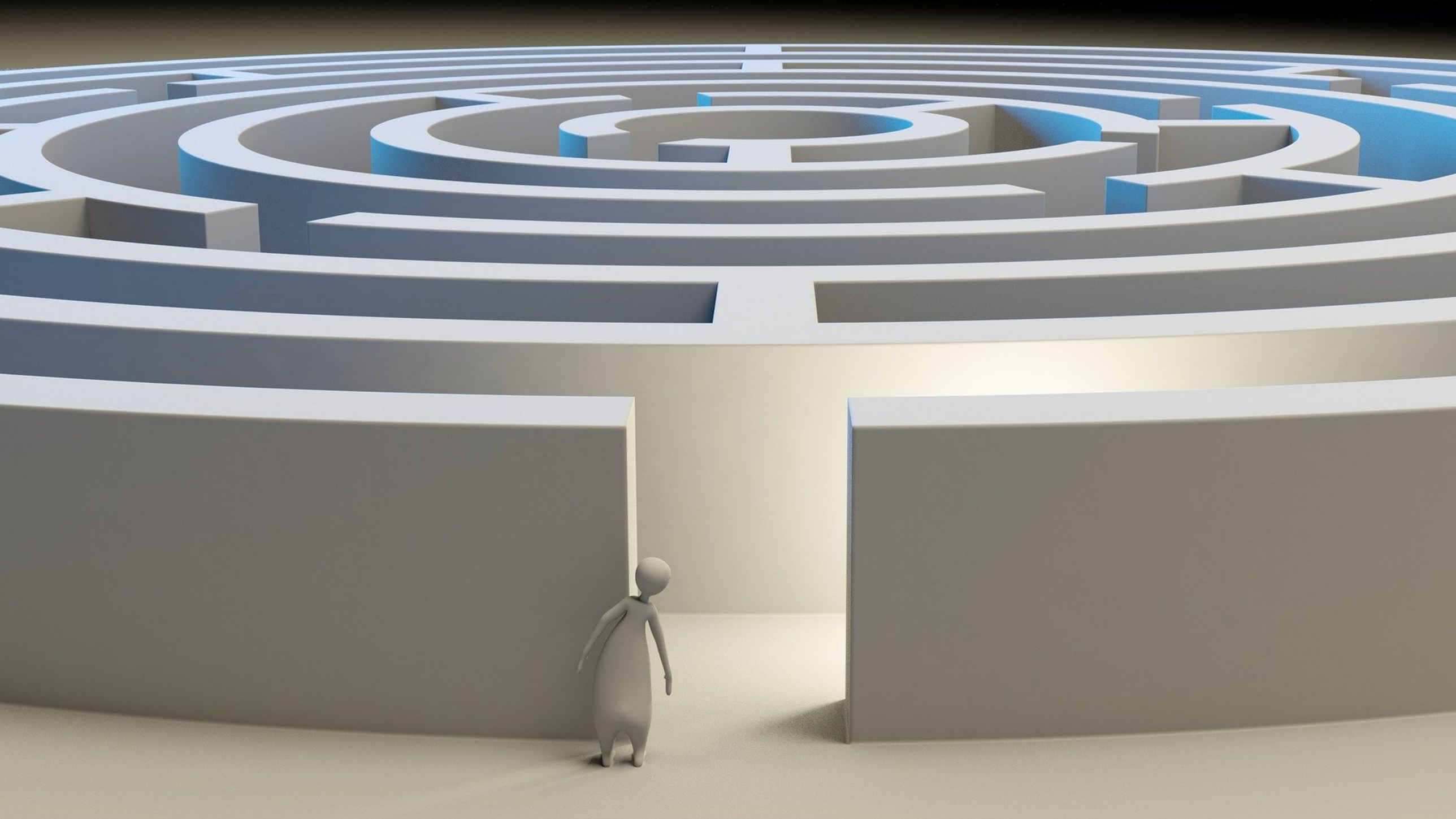
Senior Consultant
INNOQ Deutschland GmbH
@michaelvitz

Example Application

```xml
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.4</version>
    <relativePath/>
  </parent>

  <groupId>com.innoq</groupId>
  <artifactId>spring-container</artifactId>
  <version>1.0.0-SNAPSHOT</version>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
```

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplicatio
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @GetMapping
    public String index() {
        return "Hello from Spring!”;
    }
}
```





"Fat"-JAR Container

```
FROM adoptopenjdk/openjdk11:jdk-11.0.9_11-alpine-slim

COPY ./target/spring-container-*.jar /spring-container.jar

CMD ["java", "-jar", "/spring-container.jar"]
EXPOSE 8080
```

```
FROM adoptopenjdk/openjdk11:jdk-11.0.9_11-alpine-slim

RUN mkdir -p /app



WORKDIR /app

COPY ./target/spring-container-*.jar /app/spring-container.jar

CMD ["java", "-jar", “/app/spring-container.jar"]
EXPOSE 8080
```

```
FROM adoptopenjdk/openjdk11:jdk-11.0.9_11-alpine-slim

RUN mkdir -p /app && \
    chown -R daemon /app

USER daemon
WORKDIR /app

COPY ./target/spring-container-*.jar /app/spring-container.jar

CMD ["java", "-jar", “/app/spring-container.jar"]
EXPOSE 8080
```

Docker

```
docker build -t spring-container .
```

Docker

- + No changes in POM required
- + Straightforward Dockerfile
- + No additional abstraction

- - Separate step in build process
- - “Fat”-JAR

Fabric8 Maven-Docker-Plugin

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <version>0.34.1</version>
</plugin>
```

```
./mvnw verify docker:build
```

https://github.com/fabric8io/docker-maven-plugin

Fabric8 Maven-Docker-Plugin

- + Straightforward Dockerfile
- + No separate step in build process
- + Additional capabilities (start/stop/watch/...)

- +- Only small abstraction
- +- No plugin configuration in POM required

- - “Fat”-JAR

Fabric8 Maven-Docker-Plugin

```
<configuration>
  <images>
    <image>
      <name>spring-container-fabric8</name>
      <build>
        <from>adoptopenjdk/openjdk11:jdk-11.0.9_11-alpine-slim</from>
        <runCmds>
          <run>mkdir -p /app &amp;&amp; chown -R daemon /app</run>
        </runCmds>
        <user>daemon</user>
        <workdir>/app</workdir>
        <assembly>
          <targetDir>/app</targetDir>
          <descriptorRef>artifact</descriptorRef>
        </assembly>
        <cmd>
          <exec>
            <arg>java</arg>
            <arg>-jar</arg>
            <arg>/app/${project.artifactId}-${project.version}.jar</arg>
          </exec>
        </cmd>
        <ports>
          <port>8080</port>
        </ports>
      </build>
```

Fabric8 Maven-Docker-Plugin

- +- Some more abstraction

- - Dockerfile in XML
- - “Fat”-JAR

"Fat"-JAR?

```
$ du -h target/spring-container-1.0.0-SNAPSHOT.jar
16M target/spring-container-1.0.0-SNAPSHOT.jar
```

```
Sending build context to Docker daemon  19.96MB
Step 1/7 : FROM adoptopenjdk/openjdk11:jdk-11.0.9_11-alpine-slim
 ---> 6e24b2c53f87
Step 2/7 : RUN mkdir -p /app &&     chown -R daemon /app
 ---> Using cache
 ---> 04631ac529dd
Step 3/7 : USER daemon
 ---> Using cache
 ---> fe0fe11bb555
Step 4/7 : WORKDIR /app
 ---> Using cache
 ---> 4a95f3163d2d
Step 5/7 : COPY ./target/spring-container-*.jar /app/spring-container.jar
 ---> 7763afdd5b50
Step 6/7 : CMD ["java", "-jar", "/app/spring-container.jar"]
 ---> Running in a327c50e7a72
Removing intermediate container a327c50e7a72
 ---> 311762838046
Step 7/7 : EXPOSE 8080
 ---> Running in 834b132542c6
Removing intermediate container 834b132542c6
 ---> 1622208fcb32
Successfully built 1622208fcb32
```

Docker Layers

- Only contain diff to previous layer
- Read only (except Read/Write layer at runtime)
- Rule of thumb: Every instruction -> Layer
- Can be cached and reused by builds
- Size does matter during transfer

maven-dependency-plugin

maven-dependency-plugin

```
<plugin>
  <artifactId>maven-dependency-plugin</artifactId>
  <executions>
    <execution>
      <goals>
        <goal>copy-dependencies</goal>
      </goals>
      <configuration>
        <includeScope>runtime</includeScope>
      </configuration>
    </execution>
  </executions>
</plugin>
```

```
FROM adoptopenjdk/openjdk11:jdk-11.0.9_11-alpine-slim

RUN mkdir -p /app/lib && \
    chown -R daemon /app

USER daemon
WORKDIR /app

COPY ./target/dependency/ /app/lib
COPY ./target/spring-container-*.jar /app/spring-container.ja

CMD [ "java", \
      "-classpath", \
      "/app/spring-container.jar:/app/lib/*", \
      “com.innoq.spring.container.Application” ]
EXPOSE 8080
```

https://maven.apache.org/plugins/maven-dependency-plugin/

maven-dependency-plugin

- + Works with every Java application
- + Only downloads dependencies
- + Dockerfile stays clean

- - Not obvious that plugin is required for Image building

jib

jib

```
<plugin>
  <groupId>com.google.cloud.tools</groupId>
  <artifactId>jib-maven-plugin</artifactId>
  <version>2.6.0</version>
  <configuration>
    <to>
      <image>spring-container-jib</image>
    </to>
  </configuration>
</plugin>
```

```
./mvnw verify jib:dockerBuild
```

https://github.com/GoogleContainerTools/jib/tree/master/jib-maven-plugin

jib

- + Works with every Java application
- + Distroless Image

- +- No own Dockerfile
- +- Can be used without Docker daemon

- - Level of abstraction

Spring Boot

Spring Boot Extract

```
jar xf target/spring-container-1.0.0-SNAPSHOT.jar
```

Spring Boot Extract

- + Works with older Spring Boot versions
- + Straightforward script

- +- Separate build step

- - Spring Boot dependent
- - Spring Loader is included by default

Spring Boot Layered JAR

```
java \
  -Djarmode=layertools \
  -jar target/spring-container.jar \
  extract
```

Spring Boot Layered JAR

- + Layers are customisable (e.g. layer for company wide dependencies)
- + Straightforward script

- +- Separate build step

- - Spring Boot dependent

Spring Boot Build Packs

```
./mvnw spring-boot:build-image
```

Spring Boot Build Packs

- + No need for configuration

- - Multiple abstraction layers
- - Loss of control

Zombies

```
#!/usr/bin/env sh
set -euo pipefail
IFS=$'\n\t'

java \
     -XX:+UnlockExperimentalVMOptions \
     -XX:+UseJVMCICompiler \
     -jar /app/spring-container.jar
```

```
…

CMD ["/app/run.sh"]
EXPOSE 8080
```

https://blog.phusion.nl/2015/01/20/docker-and-the-pid-1-zombie-reaping-problem/

```
#!/usr/bin/env sh
set -euo pipefail
IFS=$'\n\t'

exec java \
     -XX:+UnlockExperimentalVMOptions \
     -XX:+UseJVMCICompiler \
     -jar /app/spring-container.jar
```

```
…

CMD ["/app/run.sh"]
EXPOSE 8080
```

https://blog.phusion.nl/2015/01/20/docker-and-the-pid-1-zombie-reaping-problem/

Additional thoughts

Additional thoughts

- Consider Docker Multi-Stage Builds
- Check and configure JVM Memory Management
- Look at other Solutions
- Containers are not a silver bullet
- https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Thanks! Questions?

INNOQ
www.innoq.com

Michael Vitz
michael.vitz@innoq.com
+49 151 19116015
@michaelvitz

https://www.innoq.com/en/articles/2020/08/java-spring-docker-images/

https://github.com/mvitz/javaspektrum-spring-container

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany
+49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
+49 2173 3366-0

Kreuzstr. 16
80331 München
Germany
+49 2173 3366-0

Hermannstrasse 13
20095 Hamburg
Germany
+49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 0116