Four Approaches to Reducing Java Startup Time

bellsoft

Catherine Edelveis

🥑 Developer Advocate at BellSoft

😍Love Java, Spring, JavaFX

👩‍💻Tech writer

𝕏 cat_edelveis

🦋 cat-edelveis.bsky.social

Pasha Finkelshteyn

🥑 Developer Advocate at BellSoft

≈15 years in JVM, mostly

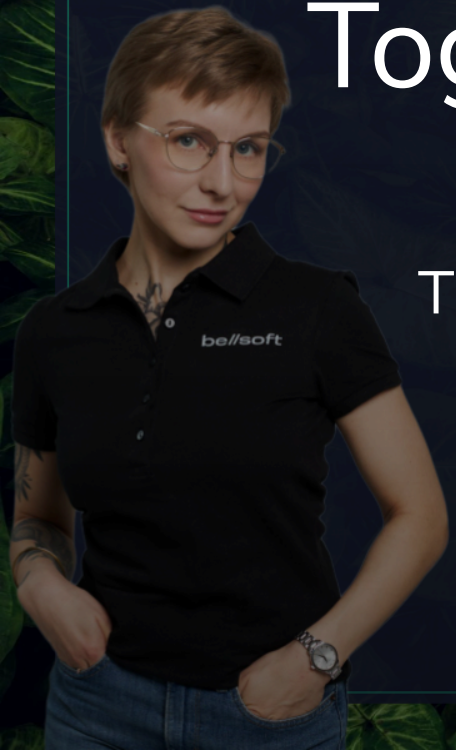
@asm0dey.site

asm0di0

Together, we are...

Together, we are...

The BellSoft's DevRel Gang!



About BellSoft

Founded in 2017 by Java and Linux experts with 15+ years of experience working at Sun/Oracle.

- Member of:
  - JCP Executive Committee
  - OpenJDK Vulnerability Group
  - GraalVM Advisory Board
  - Linux Foundation
  - Cloud Native Computing Foundation

About BellSoft

- Products:
  - Liberica JDK
  - Liberica Native Image Kit
  - Alpaquita Linux

Liberica is the JDK officially recommended by

Enough about us :)

We are here today to tell you a story about...

Enough about us :)

We are here today to tell you a story about...

YOU!

But first, four quick questions ;)

Who knows

one way

to reduce the startup of their Java application?

Who knows

two ways

to reduce the startup of their Java application?

Who knows

three ways

to reduce the startup of their Java application?

Who knows

four ways

to reduce the startup of their Java application?

Once upon a time, all was well in one startup...

Our startup is doing well!

- Java 21 LTS
- Spring Boot 3.4
- Spring AI
- MongoDB, Redis
- JTE
- Deploying to the cloud

```
@GetMapping(value = "/assistant", produces = TEXT_HTML_VALUE)
public JteModel assistant() {
  return templates.assistantView("Bot Assistant - Chat");
}
```

Starting point: `Dockerfile`

```
FROM bellsoft/liberica-runtime-container:jdk-21-musl as builder
ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-21-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar /app/app.jar
```

Starting point: `Dockerfile`

```
FROM bellsoft/liberica-runtime-container:jdk-21-musl as builder
ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-21-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar /app/app.jar
```

Starting point: `Dockerfile`

```
ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-21-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar /app/app.jar
ENTRYPOINT ["java", "-jar", "/app/app.jar"]
```

Starting point: `Dockerfile`

```
ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-21-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar /app/app.jar
ENTRYPOINT ["java", "-jar", "/app/app.jar"]
```

Starting point: Startup

```
docker compose logs chat-api bot-assistant | grep Started
Started BotAssistantApplication in 2.108 seconds (process running for 3.181)
Started ChatApiApplication in 2.83 seconds (process running for 3.469)
```

Total startup time of both services: 6.5 s

We are growing!

- So many new users!
- Scale the services to meet the traffic

We are growing!

- So many new users!
- Scale the services to meet the traffic

But wait...

We are growing!

- So many new users!
- Scale the services to meet the traffic

But wait...

Why the rollout is so slow?

We are an agile team, we need a fast feedback loop!

We are an agile team, we need a fast feedback loop!

Need faster rollout for faster feedback

We are an agile team, we need a fast feedback loop!

Need faster rollout for faster feedback

The longer the start, the slower the rollout

We are an agile team, we need a fast feedback loop!

Need faster rollout for faster feedback

The longer the start, the slower the rollout

Maybe Java is not cut out for the cloud?

We are an agile team, we need a fast feedback loop!

Need faster rollout for faster feedback

The longer the start, the slower the rollout

Maybe Java is not cut out for the cloud?

Maybe it’s not too late to migrate to Go?

Luckily, four brave developers work at this startup:

The Defender

The Sage

The Explorer

The Rebel

They are very different, but have something in common:

They care about their software product and can’t sit idle in the troubled time.

And that’s where

YOU

take the stage!

I want painless integration with no incompatibilities!

The Defender

AppCDS

- Save class metadata to the disc and read it from the archive
- Even better with Spring AOT (load even more classes!)

Considerations:

- Use the same JVM
- Use the same classpath
- Bigger Docker image due to the archive

AppCDS

- Save class metadata to the disc and read it from the archive
- Even better with Spring AOT (load even more classes!)

Considerations:

- Use the same JVM
- Use the same classpath
- Bigger Docker image due to the archive

Let's take it for a spin!

```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <executions>
        <execution>
            <id>process-aot</id>
            <goals>
                <goal>process-aot</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

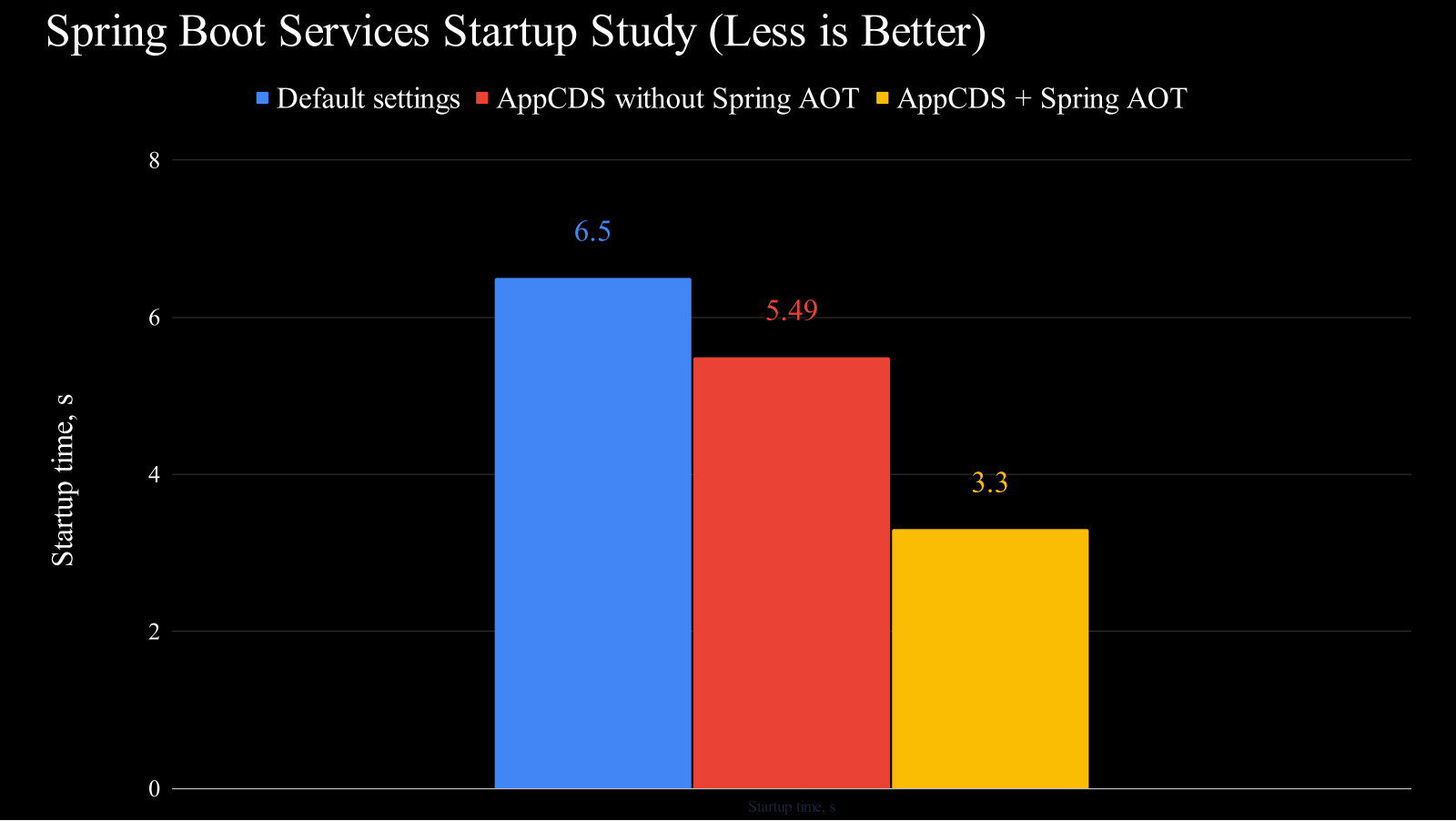
```
FROM bellsoft/liberica-runtime-container:jdk-21.0.7_9-musl as builder

ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-21.0.7_9-cds-musl as optimizer
ARG project
ENV project=${project}

WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar app.jar
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted
```

```
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-21.0.7_9-cds-musl as optimizer
ARG project
ENV project=${project}

WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar app.jar
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted


FROM bellsoft/liberica-runtime-container:jre-21.0.7_9-cds-musl

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
```

```
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted


FROM bellsoft/liberica-runtime-container:jre-21.0.7_9-cds-musl

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./
RUN java -Dspring.aot.enabled=true \
-XX:ArchiveClassesAtExit=./application.jsa \
-Dspring.context.exit=onRefresh -jar /app/app.jar
ENTRYPOINT ["java", "-Dspring.aot.enabled=true", \
"-XX:SharedArchiveFile=application.jsa", \
"-jar", "/app/app.jar"]
```

```
FROM bellsoft/liberica-runtime-container:jre-21.0.7_9-cds-musl

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./
RUN java -Dspring.aot.enabled=true \
-XX:ArchiveClassesAtExit=./application.jsa \
-Dspring.context.exit=onRefresh -jar /app/app.jar
ENTRYPOINT ["java", "-Dspring.aot.enabled=true", \
"-XX:SharedArchiveFile=application.jsa", \
"-jar", "/app/app.jar"]
```

```
FROM bellsoft/liberica-runtime-container:jre-21.0.7_9-cds-musl

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./
RUN java -Dspring.aot.enabled=true \
-XX:ArchiveClassesAtExit=./application.jsa \
-Dspring.context.exit=onRefresh -jar /app/app.jar
ENTRYPOINT ["java", "-Dspring.aot.enabled=true", \
"-XX:SharedArchiveFile=application.jsa", \
"-jar", "/app/app.jar"]
```


Spring Boot Services Startup Study (Less is Better)
Default settings
AppCDS without Spring AOT
AppCDS + Spring AOT
Startup time, s
8
6
4
2
0
6.5
5.49
3.3
Startup time, s

We need to prepare ourselves and gather knowledge!

The Sage

Project Leyden

- Beyond AppCDS: AOT Cache with pre-compiled code
- Condensers -> optimizations

Considerations:

- Still in the makings
- The more condensers applied, the bigger the cache

Project Leyden

- Beyond AppCDS: AOT Cache with pre-compiled code
- Condensers -> optimizations

Considerations:

- Still in the makings
- The more condensers applied, the bigger the cache

Meanwhile, we can experiment!

Let’s start with what is already implemented

JEP 483: Ahead-of-Time Class Loading & Linking (JDK 24)

> Improve startup time by making the classes of an application instantly available, in a loaded and linked state, when the HotSpot Java Virtual Machine starts. Achieve this by monitoring the application during one run and storing the loaded and linked forms of all classes in a cache for use in subsequent runs. Lay a foundation for future improvements to both startup and warmup time.

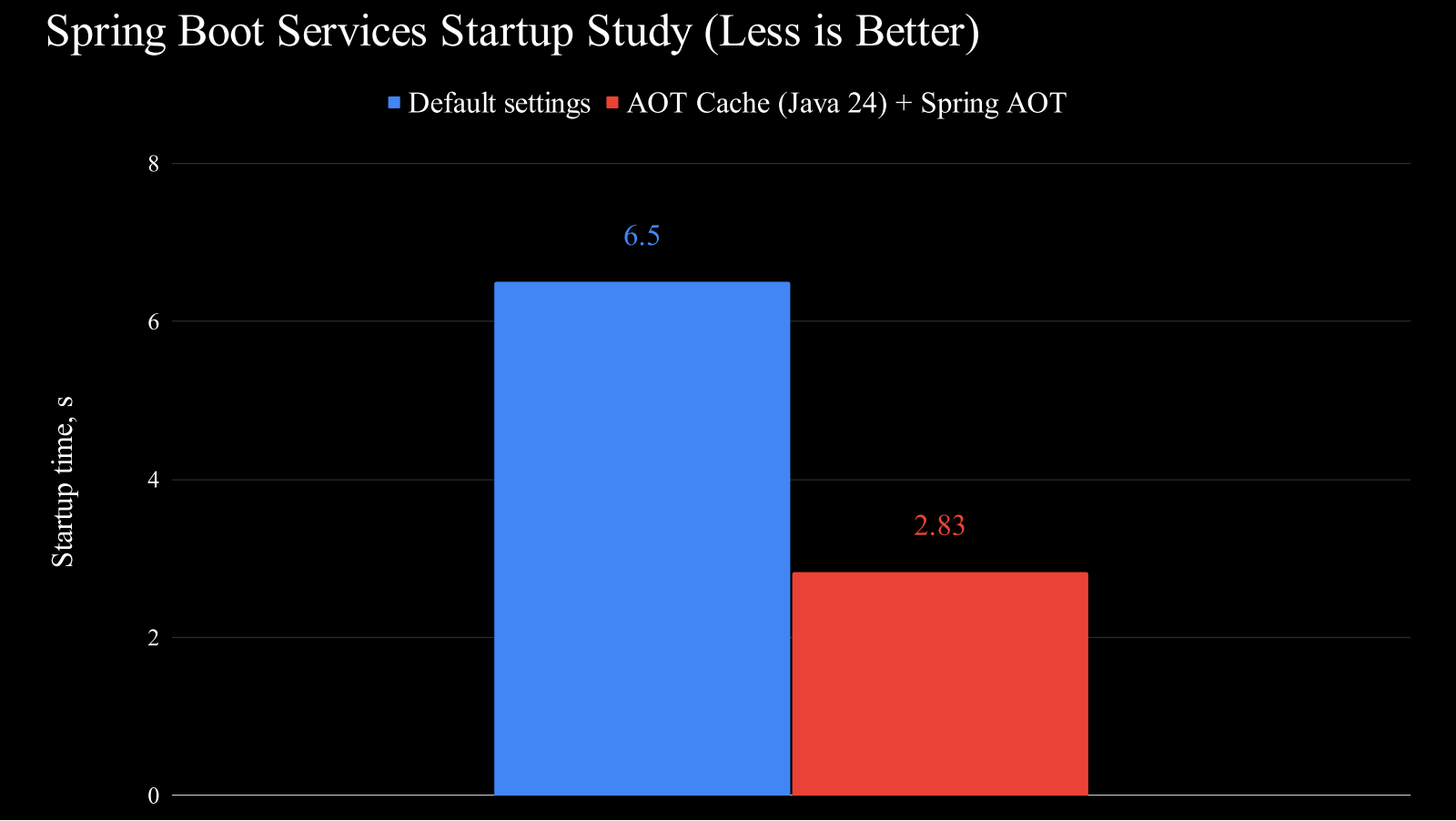
```
FROM bellsoft/liberica-runtime-container:jdk-24-musl AS builder

ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-24-cds-musl AS optimizer
ARG project
ENV project=${project}

WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar app.jar
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/liberica-runtime-container:jre-24-cds-musl AS runner
```

```
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw package

FROM bellsoft/liberica-runtime-container:jre-24-cds-musl AS optimizer
ARG project
ENV project=${project}

WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar app.jar
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/liberica-runtime-container:jre-24-cds-musl AS runner

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
```

```
WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar app.jar
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/liberica-runtime-container:jre-24-cds-musl AS runner

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./

RUN java -Dspring.aot.enabled=true -XX:AOTMode=record \
    -XX:AOTConfiguration=app.aotconf -Dspring.context.exit=onRefresh -jar /app/app.jar
RUN java -Dspring.aot.enabled=true -XX:AOTMode=create \
    -XX:AOTConfiguration=app.aotconf -XX:AOTCache=app.aot -jar /app/app.jar

ENTRYPOINT ["java", "-Dspring.aot.enabled=true", "-XX:AOTCache=app.aot",
```

```
COPY --from=builder /app/${project}/target/*.jar app.jar
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/liberica-runtime-container:jre-24-cds-musl AS runner

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./

RUN java -Dspring.aot.enabled=true -XX:AOTMode=record \
    -XX:AOTConfiguration=app.aotconf -Dspring.context.exit=onRefresh -jar /app/app.jar
RUN java -Dspring.aot.enabled=true -XX:AOTMode=create \
    -XX:AOTConfiguration=app.aotconf -XX:AOTCache=app.aot -jar /app/app.jar

ENTRYPOINT ["java", "-Dspring.aot.enabled=true", "-XX:AOTCache=app.aot",
 "-jar", "/app/app.jar"]
```

```
COPY --from=builder /app/${project}/target/*.jar app.jar
RUN java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/liberica-runtime-container:jre-24-cds-musl AS runner

RUN apk add curl
WORKDIR /app
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./

RUN java -Dspring.aot.enabled=true -XX:AOTMode=record \
    -XX:AOTConfiguration=app.aotconf -Dspring.context.exit=onRefresh -jar /app/app.jar
RUN java -Dspring.aot.enabled=true -XX:AOTMode=create \
    -XX:AOTConfiguration=app.aotconf -XX:AOTCache=app.aot -jar /app/app.jar

ENTRYPOINT ["java", "-Dspring.aot.enabled=true", "-XX:AOTCache=app.aot",
 "-jar", "/app/app.jar"]
```


Spring Boot Services Startup Study (Less is Better)
Default settings
AOT Cache (Java 24) + Spring AOT
Startup time, s
8
6
4
2
0
6.5
2.83

Not enough?

Let’s push it even further and use the builds of premain in Leyden!

We will be happy to get your feedback on what’s broken!

```
FROM bellsoft/alpaquita-linux-base:glibc AS downloader
ADD https://is.gd/tZhyPF /java.tar.gz # just a placeholder
RUN apk add tar
RUN tar -zxvf java.tar.gz && mv /jdk-24 /java

FROM bellsoft/alpaquita-linux-base:glibc AS builder
ARG project

WORKDIR /app
COPY --from=downloader /java /java
ADD ../pom.xml ./
ADD ${project} /app/${project}
ENV JAVA_HOME=/java \
    project=${project}

RUN cd ${project} && ./mvnw package

FROM bellsoft/alpaquita-linux-base:glibc AS optimizer
ARG project
```

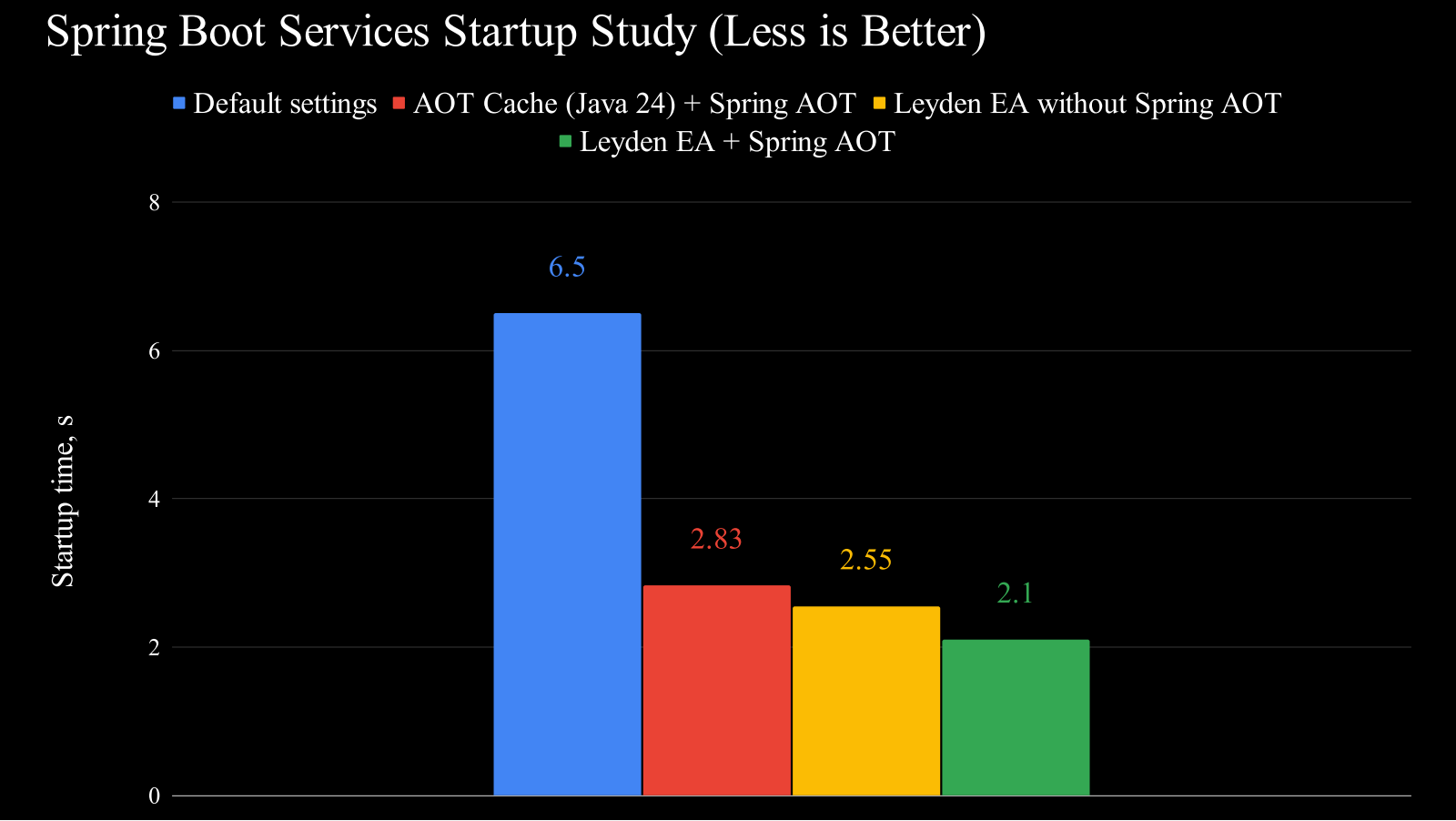
```
RUN apk add tar
RUN tar -zxvf java.tar.gz && mv /jdk-24 /java

FROM bellsoft/alpaquita-linux-base:glibc AS builder
ARG project

WORKDIR /app
COPY --from=downloader /java /java
ADD ../pom.xml ./
ADD ${project} /app/${project}
ENV JAVA_HOME=/java \
    project=${project}

RUN cd ${project} && ./mvnw package

FROM bellsoft/alpaquita-linux-base:glibc AS optimizer
ARG project

WORKDIR /app
```

```
    project=${project}

RUN cd ${project} && ./mvnw package

FROM bellsoft/alpaquita-linux-base:glibc AS optimizer
ARG project

WORKDIR /app
COPY --from=builder /app/${project}/target/*.jar app.jar
COPY --from=downloader /java /java
ENV project=${project}

RUN /java/bin/java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/alpaquita-linux-base:glibc AS runner

RUN apk add curl
WORKDIR /app
```

```
ENV project=${project}

RUN /java/bin/java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/alpaquita-linux-base:glibc AS runner

RUN apk add curl
WORKDIR /app

COPY --from=downloader /java /java
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./

RUN /java/bin/java -XX:CacheDataStore=./application.cds \
    -Dspring.context.exit=onRefresh -jar /app/app.jar

ENTRYPOINT ["/java/bin/java", "-XX:CacheDataStore=./application.cds", "-jar", "/app/app.jar"]
```

```
ENV project=${project}

RUN /java/bin/java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/alpaquita-linux-base:glibc AS runner

RUN apk add curl
WORKDIR /app

COPY --from=downloader /java /java
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./

RUN /java/bin/java -XX:CacheDataStore=./application.cds \
    -Dspring.context.exit=onRefresh -jar /app/app.jar

ENTRYPOINT ["/java/bin/java", "-XX:CacheDataStore=./application.cds", "-jar", "/app/app.jar"]
```

```
ENV project=${project}

RUN /java/bin/java -Djarmode=tools -jar app.jar extract --layers --destination extracted

FROM bellsoft/alpaquita-linux-base:glibc AS runner

RUN apk add curl
WORKDIR /app

COPY --from=downloader /java /java
COPY --from=optimizer /app/extracted/dependencies/ ./
COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
COPY --from=optimizer /app/extracted/application/ ./

RUN /java/bin/java -XX:CacheDataStore=./application.cds \
    -Dspring.context.exit=onRefresh -jar /app/app.jar

ENTRYPOINT ["/java/bin/java", "-XX:CacheDataStore=./application.cds", "-jar", "/app/app.jar"]
```


Spring Boot Services Startup Study (Less is Better)
Default settings
AOT Cache (Java 24) + Spring AOT
Leyden EA without Spring AOT
Leyden EA + Spring AOT
Startup time, s
8
6
4
2
0
6.5
2.83
2.55
2.1

I found something exciting... Can’t wait to explore!

The Explorer

GraalVM Native Image

- Ahead-of-time compilation
- Standalone executable (.exe)
- No OpenJDK required to run

Native Image: Any Considerations?

- Resource-demanding build process
- Several minutes to build the image
- Static compilation instead of dynamic one

Native Image: Any Considerations?

- Resource-demanding build process
- Several minutes to build the image
- Static compilation instead of dynamic one

Let the journey begin!

First, let’s try it locally

Build a fat JAR

Build the image:

```
$JAVA_HOME/bin/native-image \
-jar bot-assistant/target/bot-assistant-1.0-SNAPSHOT-spring-boot.jar
```

The image was built, let’s run it!

I never knew the journey can be so pleasant!

I never knew the journey can be so pleasant!

Uh-oh...

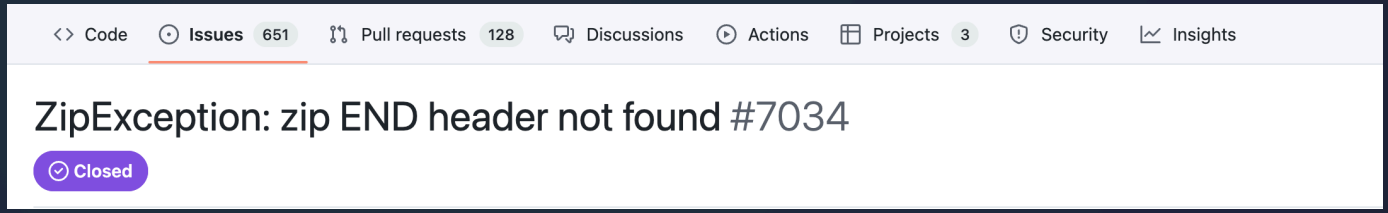
A dragon!

```
Exception in thread "main" java.lang.IllegalStateException:
java.util.zip.ZipException: zip END header not found
        at org.springframework.boot.loader.ExecutableArchiveLauncher.<init>(ExecutableArchiveLaun
        at org.springframework.boot.loader.JarLauncher.<init>(JarLauncher.java:42)
        at org.springframework.boot.loader.JarLauncher.main(JarLauncher.java:65)
Caused by: java.util.zip.ZipException: zip END header not found
```

A dragon!

```
Exception in thread "main" java.lang.IllegalStateException:
java.util.zip.ZipException: zip END header not found
        at org.springframework.boot.loader.ExecutableArchiveLauncher.<init>(ExecutableArchiveLaun
        at org.springframework.boot.loader.JarLauncher.<init>(JarLauncher.java:42)
        at org.springframework.boot.loader.JarLauncher.main(JarLauncher.java:65)
Caused by: java.util.zip.ZipException: zip END header not found
```

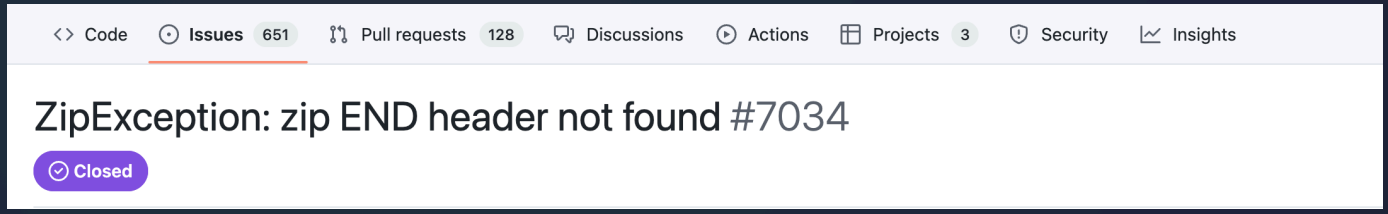
Apparently, a known issue



➡️ Use `mvn -Pnative native:compile` to build the image.

Maven plugin calls `native-image java -jar` under the hood but uses lots of flags that solve problems like the one above.

Apparently, a known issue



➡️ Use `mvn -Pnative native:compile` to build the image.

Maven plugin calls `native-image java -jar` under the hood but uses lots of flags that solve problems like the one above.

Alright, let's try with a plugin!

If you use Spring Boot Parent POM, the Native Image profile is already configured!

➡️ Enable it by simply running

```
mvn -Pnative native:compile
```

Another dragon!

```
Error: Classes that should be initialized at run time got initialized during image building:
com.ctc.wstx.api.CommonConfig was unintentionally initialized at build time.
com.ctc.wstx.stax.WstxInputFactory was unintentionally initialized at build time.
com.ctc.wstx.api.ReaderConfig was unintentionally initialized at build time.
com.ctc.wstx.util.DefaultXmlSymbolTable was unintentionally initialized at build time.

To see how the classes got initialized, use
 --trace-class-initialization=com.ctc.wstx.api.CommonConfig,com.ctc.wstx.stax.WstxInputFactory,co
```

Another dragon!

```
Error: Classes that should be initialized at run time got initialized during image building:
com.ctc.wstx.api.CommonConfig was unintentionally initialized at build time.
com.ctc.wstx.stax.WstxInputFactory was unintentionally initialized at build time.
com.ctc.wstx.api.ReaderConfig was unintentionally initialized at build time.
com.ctc.wstx.util.DefaultXmlSymbolTable was unintentionally initialized at build time.

To see how the classes got initialized, use
 --trace-class-initialization=com.ctc.wstx.api.CommonConfig,com.ctc.wstx.stax.WstxInputFactory,co
```

Add `native` profile:

```
<profiles>
    <profile>
        <id>native</id>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.graalvm.buildtools</groupId>
                    <artifactId>native-maven-plugin</artifactId>
                    <executions>
                        <execution>
                            <id>build-native</id>
```

Add `native` profile:

```
    <profile>
        <id>native</id>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.graalvm.buildtools</groupId>
                    <artifactId>native-maven-plugin</artifactId>
                    <executions>
                        <execution>
                            <id>build-native</id>
                            <goals>
```

Add `native` profile:

```
                    </execution>
                </executions>
                <configuration>
                    <imageName>bot-assistant</imageName>
                    <outputDirectory>target/native</outputDirectory>
                    <mainClass>com.github.asm0dey.botassistant.BotAssistantApplication</mainCl
                </configuration>
            </plugin>
        </plugins>
    </build>
</profile>
```

Add the `--trace-class-initialization` argument:

```
<profile>
    <id>native</id>
    <build>
        <plugins>
            <plugin>
                <groupId>org.graalvm.buildtools</groupId>
                <artifactId>native-maven-plugin</artifactId>
                <executions>
                    <execution>
                        <id>build-native</id>
                        <goals>
```

Run

```
mvn -Pnative native:compile
```

Add the `--trace-class-initialization` argument:

```
            <configuration>
                <imageName>bot-assistant</imageName>
                <outputDirectory>target/native</outputDirectory>
                <mainClass>com.github.asm0dey.botassistant.BotAssistantApplication</mainClass>
                <buildArgs>
                    <buildArgs>--trace-class-initialization=com.ctc.wstx.util.DefaultXmlSymbol
                </buildArgs>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Run

```
mvn -Pnative native:compile
```

More detailed output:

```
Error: Classes that should be initialized at run time got initialized during image building:

 com.ctc.wstx.api.CommonConfig was unintentionally initialized at build time.
 org.springframework.http.codec.xml.XmlEventDecoder caused initialization of this class
 with the following trace:
        at com.ctc.wstx.api.CommonConfig.<clinit>(CommonConfig.java:59)
        at com.ctc.wstx.stax.WstxInputFactory.<init>(WstxInputFactory.java:149)
        at java.lang.invoke.DirectMethodHandle$Holder.newInvokeSpecial(DirectMethodHandle$Holder)
        at java.lang.invoke.Invokers$Holder.invokeExact_MT(Invokers$Holder)
        at jdk.internal.reflect.DirectConstructorHandleAccessor.invokeImpl(DirectConstructorHandl
        at jdk.internal.reflect.DirectConstructorHandleAccessor.newInstance(DirectConstructorHand
```

More detailed output:

```
Error: Classes that should be initialized at run time got initialized during image building:

 com.ctc.wstx.api.CommonConfig was unintentionally initialized at build time.
 org.springframework.http.codec.xml.XmlEventDecoder caused initialization of this class
 with the following trace:
       at com.ctc.wstx.api.CommonConfig.<clinit>(CommonConfig.java:59)
       at com.ctc.wstx.stax.WstxInputFactory.<init>(WstxInputFactory.java:149)
       at java.lang.invoke.DirectMethodHandle$Holder.newInvokeSpecial(DirectMethodHandle$Holder)
       at java.lang.invoke.Invokers$Holder.invokeExact_MT(Invokers$Holder)
       at jdk.internal.reflect.DirectConstructorHandleAccessor.invokeImpl(DirectConstructorHandl
       at jdk.internal.reflect.DirectConstructorHandleAccessor.newInstance(DirectConstructorHand
```

We found the culprit!

Alright, let's add the option to initialize `org.springframework.http.codec.xml.XmlEventDecoder` at runtime:

```
<profile>
    <id>native</id>
    <build>
        <plugins>
            <plugin>
                <groupId>org.graalvm.buildtools</groupId>
                <artifactId>native-maven-plugin</artifactId>
                <executions>
                    <execution>
                        <id>build-native</id>
                        <goals>
```

Run

```
mvn -Pnative native:compile
```

Alright, let's add the option to initialize `org.springframework.http.codec.xml.XmlEventDecoder` at runtime:

```
                <configuration>
                    <imageName>bot-assistant</imageName>
                    <outputDirectory>target/native</outputDirectory>
                    <mainClass>com.github.asm0dey.botassistant.BotAssistantApplication</mainClass>
                    <buildArgs>
                        <buildArgs>--initialize-at-run-time=org.springframework.http.codec.xml.Xml
                    </buildArgs>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

Run

```
mvn -Pnative native:compile
```

The same dragon!

```
Error: Classes that should be initialized at run time got initialized during image building:
 com.ctc.wstx.api.CommonConfig was unintentionally initialized at build time. To see why com.ctc.
com.ctc.wstx.stax.WstxInputFactory was unintentionally initialized at build time. To see why com.
com.ctc.wstx.api.ReaderConfig was unintentionally initialized at build time. To see why com.ctc.w
com.ctc.wstx.util.DefaultXmlSymbolTable was unintentionally initialized at build time. To see why
To see how the classes got initialized, use --trace-class-initialization=com.ctc.wstx.api.CommonC
```

This is one evasive dragon...

Apparently, another known issue

org.springframework.http.codec.xml.XmlEventDecoder Native Image Error - got initialized during image building #31806

Closed as not planned

➡️ Need to add the `--strict-image-heap` option.

This mode requires only the classes that are stored in the image heap to be marked with –initialize-at-build-time. This effectively reduces the number of configuration entries necessary to achieve build-time initialization.
Note that --strict-image-heap is enabled by default in Native Image starting from GraalVM for JDK 22.

Let's add the `--strict-image-heap` option:

```
                <configuration>
                    <imageName>bot-assistant</imageName>
                    <outputDirectory>target/native</outputDirectory>
                    <mainClass>com.github.asm0dey.botassistant.BotAssistantApplication</mainClass
                    <buildArgs>
                        <buildArgs>--strict-image-heap</buildArgs>
                    </buildArgs>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

Run

```
mvn -Pnative native:compile
```

Success!

Success!

...Or is it?

Success!

...Or is it?

The image was build, let's run it.

Another dragon!

```
2025-04-11T12:25:33.511+03:00 ERROR 64619 --- [bot-assistant] [nio-8081-exec-2] o.a.c.c.C.[.[.[/]

java.lang.UnsatisfiedLinkError:
jdk.jfr.internal.JVM.isExcluded(Ljava/lang/Class;)Z
[symbol: Java_jdk_jfr_internal_JVM_isExcluded
or Java_jdk_jfr_internal_JVM_isExcluded__Ljava_lang_Class_2]

        at org.graalvm.nativeimage.builder/com.oracle.svm.core.jni.access.JNINativeLinkage.getOrF
        at org.graalvm.nativeimage.builder/com.oracle.svm.core.jni.JNIGeneratedMethodSupport.nati
        at jdk.jfr@21.0.6/jdk.jfr.internal.JVM.isExcluded(Native Method) ~[na:na]
                at jdk.jfr@21.0.6/jdk.jfr.internal.MetadataRepository.register(MetadataRepository
```

Another dragon!

```
2025-04-11T12:25:33.511+03:00 ERROR 64619 --- [bot-assistant] [nio-8081-exec-2] o.a.c.c.C.[.[.[/]

java.lang.UnsatisfiedLinkError:
jdk.jfr.internal.JVM.isExcluded(Ljava/lang/Class;)Z
[symbol: Java_jdk_jfr_internal_JVM_isExcluded
or Java_jdk_jfr_internal_JVM_isExcluded__Ljava_lang_Class_2]

        at org.graalvm.nativeimage.builder/com.oracle.svm.core.jni.access.JNINativeLinkage.getOrF
        at org.graalvm.nativeimage.builder/com.oracle.svm.core.jni.JNIGeneratedMethodSupport.nati
        at jdk.jfr@21.0.6/jdk.jfr.internal.JVM.isExcluded(Native Method) ~[na:na]
                at jdk.jfr@21.0.6/jdk.jfr.internal.MetadataRepository.register(MetadataRepository
```

Apparently, when we create a connection to Redis, Spring creates a custom JFR event

Even the Tracing Agent doesn't detect that!

So, let's enable JFR explicitly:

```
                <imageName>bot-assistant</imageName>
                <outputDirectory>target/native</outputDirectory>
                <mainClass>com.github.asm0dey.botassistant.BotAssistantApplication</mainClass
                <buildArgs>
                    <buildArgs>--strict-image-heap</buildArgs>
                    <buildArgs>--enable-monitoring=jfr</buildArgs>
                </buildArgs>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Run

```
mvn -Pnative native:compile
```

Finally, everything works!

Finally, everything works!

...on my machine

Let’s move on to building a native image in a Docker container.

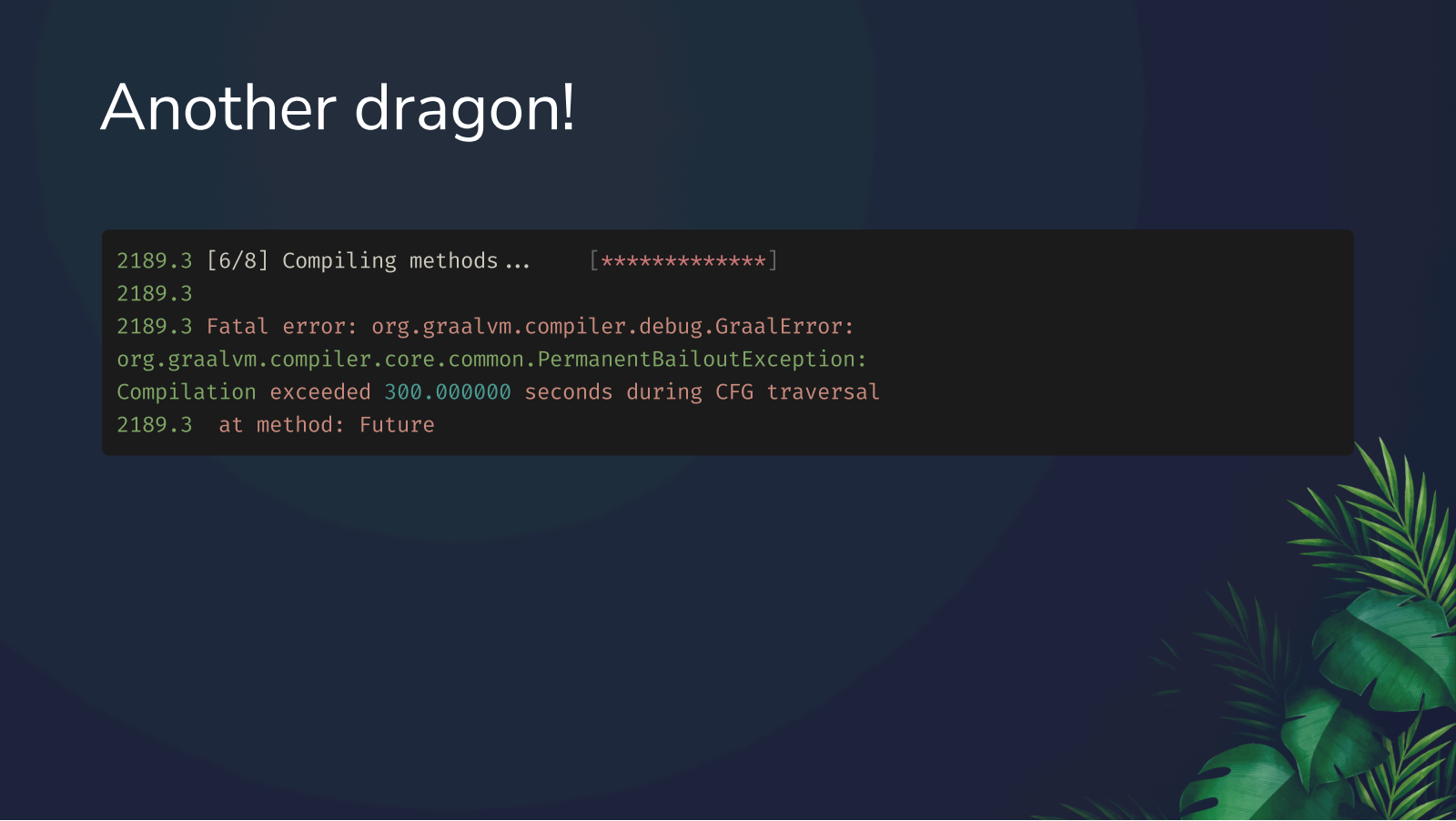
```
FROM bellsoft/liberica-native-image-kit-container:jdk-21-nik-23.1.6-musl as builder
ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw -Pnative native:compile

FROM bellsoft/alpaquita-linux-base:stream-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
ENTRYPOINT ["/app/app"]
COPY --from=builder /app/${project}/target/native/${project} /app/app
```

```
FROM bellsoft/liberica-native-image-kit-container:jdk-21-nik-23.1.6-musl as builder
ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw -Pnative native:compile

FROM bellsoft/alpaquita-linux-base:stream-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
ENTRYPOINT ["/app/app"]
COPY --from=builder /app/${project}/target/native/${project} /app/app
```

```
FROM bellsoft/liberica-native-image-kit-container:jdk-21-nik-23.1.6-musl as builder
ARG project
ENV project=${project}

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw -Pnative native:compile

FROM bellsoft/alpaquita-linux-base:stream-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
ENTRYPOINT ["/app/app"]
COPY --from=builder /app/${project}/target/native/${project} /app/app
```

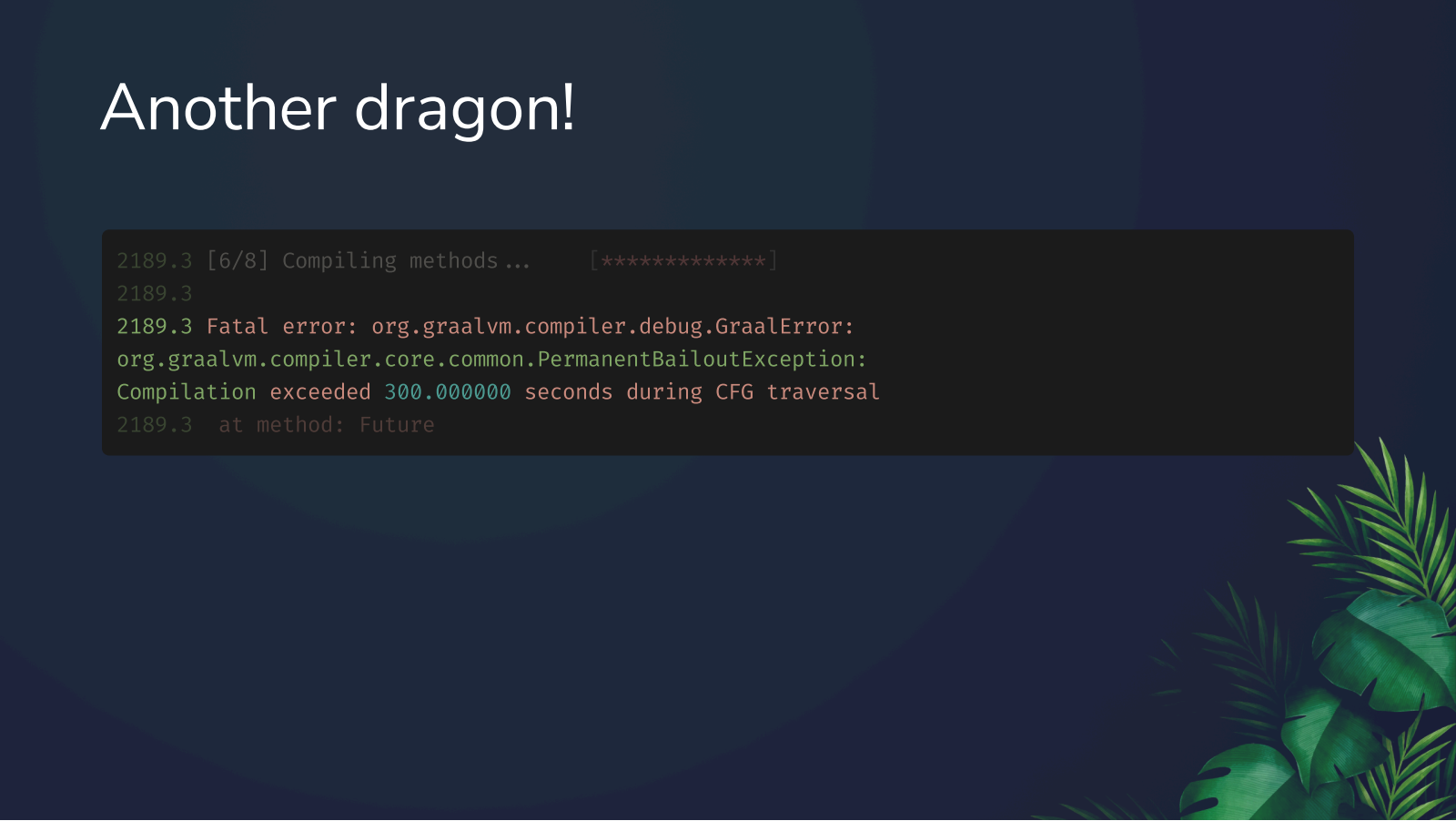
Another dragon!

```
2189.3 [6/8] Compiling methods ...     [*************]
2189.3
2189.3 Fatal error: org.graalvm.compiler.debug.GraalError:
org.graalvm.compiler.core.common.PermanentBailoutException:
Compilation exceeded 300.000000 seconds during CFG traversal
2189.3  at method: Future
```



Another dragon!

```
2189.3 [6/8] Compiling methods ...     [************]
2189.3
2189.3 Fatal error: org.graalvm.compiler.debug.GraalError:
org.graalvm.compiler.core.common.PermanentBailoutException:
Compilation exceeded 300.000000 seconds during CFG traversal
2189.3  at method: Future
```



The build was taking too long and never finished

Solution:

- Increase timeout time
- Build the image on a powerful PC or in the CI
- Plug a laptop into a power socket

The build was taking too long and never finished

Solution:

- Increase timeout time
- Build the image on a powerful PC or in the CI
- Plug a laptop into a power socket

The build was taking too long and never finished

Solution:

- Increase timeout time
- Build the image on a powerful PC or in the CI
- Plug a laptop into a power socket

Let's give it another try!

Another dragon!

```
520.7 [8/8] Creating image ...       [*****]
520.7 ------------------------------------------------------------------------------------
520.7                     53.2s (13.5% of total time) in 2385 GCs | Peak RSS: 6.77GB | CPU load:
520.7 ------------------------------------------------------------------------------------
520.7 Produced artifacts:
520.7  /app/bot-assistant/target/native/svm_err_b_20250415T153317.420_pid287.md (build_info)
520.7 ====================================================================================
520.7 Failed generating 'bot' after 6m 32s.
520.7
520.7 The build process encountered an unexpected error:
520.7
```

Another dragon!

```
520.7 ═══════════════════════════════════════════════════════════════════════════
520.7 Failed generating 'bot' after 6m 32s.
520.7
520.7 The build process encountered an unexpected error:
520.7
520.7 > java.lang.RuntimeException: There was an error linking the native image:
Linker command exited with 1
520.7
520.7 Linker command executed:
520.7 /usr/bin/gcc -z noexecstack -Wl,--gc-sections -Wl,--version-script,/tmp/SVM-109235547950005
520.7
```

Linking issue: musl libc lacks required tools

- Option 1: add required packages (libstc++, etc.) with `apk add`
- Option 2: switch from musl-based Alpaquita to glibc-based

```
FROM bellsoft/liberica-native-image-kit-container:jdk-21-nik-23.1.6-stream-musl as builder

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw -Pnative native:compile

FROM bellsoft/alpaquita-linux-base:stream-musl
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
```

Linking issue: musl libc lacks required tools

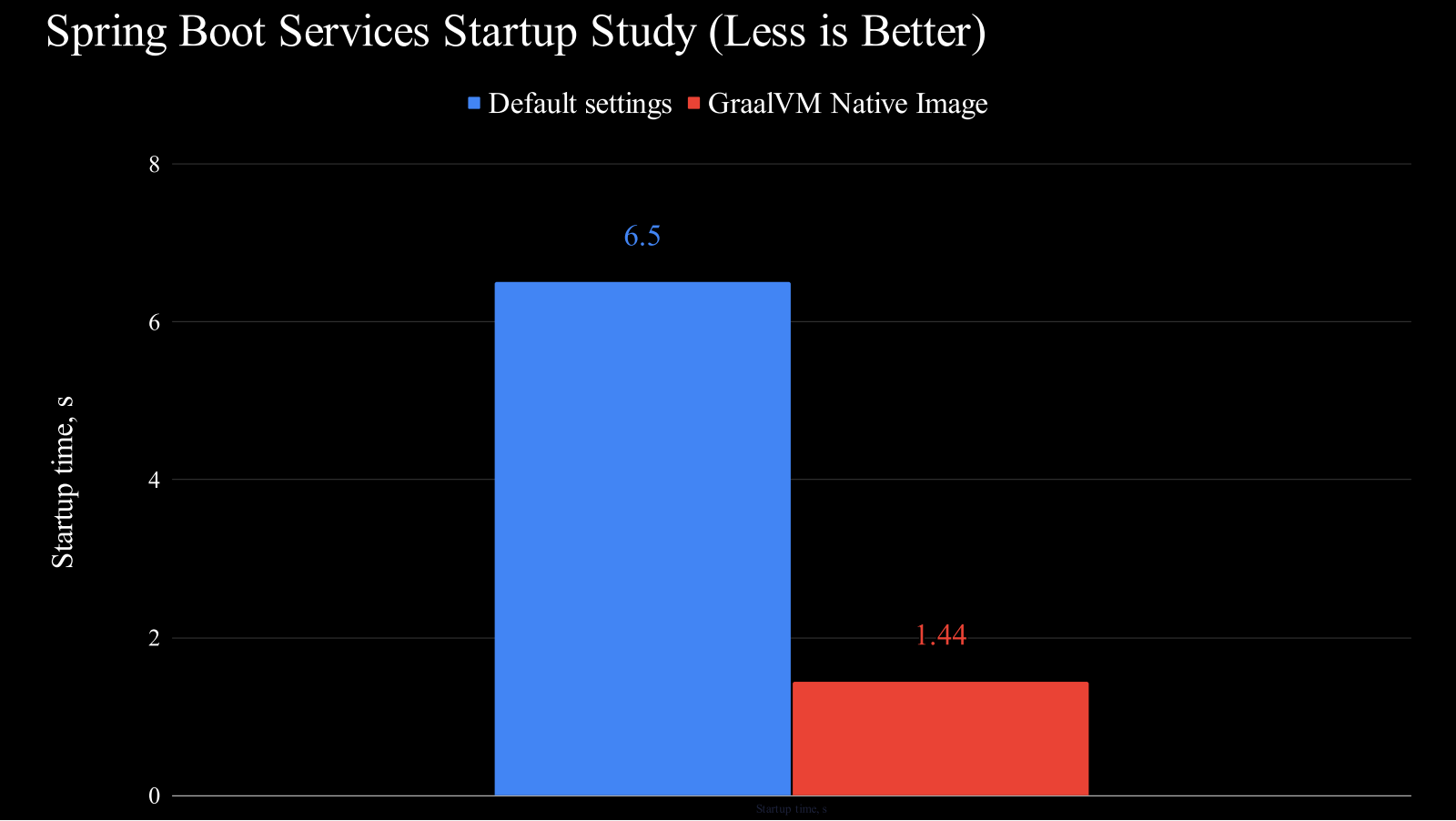
- Option 1: add required packages (libstc++, etc.) with `apk add`
- Option 2: switch from musl-based Alpaquita to glibc-based

```
FROM bellsoft/liberica-native-image-kit-container:jdk-21-nik-23.1.6-stream-glibc as builder

WORKDIR /app
ADD ${project} /app/${project}
ADD ../pom.xml ./
RUN cd ${project} && ./mvnw -Pnative native:compile

FROM bellsoft/alpaquita-linux-base:stream-glibc
ARG project
ENV project=${project}

RUN apk add curl
WORKDIR /app
```

We did it!

We did it!

Both services compile and run successfully


Spring Boot Services Startup Study (Less is Better)
Default settings
GraalVM Native Image
Startup time, s
8
6
4
2
0
6.5
1.44
Startup time, s


ITS OVER
WE DID IT

We need drastic changes, I won’t settle for half-measures!

The Rebel

Coordinated Restore at Checkpoint (CRaC)

> The CRaC (Coordinated Restore at Checkpoint) Project researches coordination of Java programs with mechanisms to checkpoint (make an image of, snapshot) a Java instance while it is executing. Restoring from the image could be a solution to some of the problems with the start-up and warm-up times. The primary aim of the Project is to develop a new standard mechanism-agnostic API to notify Java programs about the checkpoint and restore events. Other research activities will include, but will not be limited to, integration with existing checkpoint/restore mechanisms and development of new ones, changes to JVM and JDK to make images smaller and ensure they are correct.

https://openjdk.org/projects/crac/

TL;DR

- Pause and restart a Java application
- A snapshot of the current JVM state
- JVM is still there: dynamic performance optimization is possible after restore

CRaC is not a canonical part of JDK

Only some vendors provide it: BellSoft, Azul

Azul was the first to implement, BellSoft supports a fork

Project CRaC: Any Considerations?

- A snapshot may contain sensitive data
- May need to augment the code for reliable checkpoint and restore

Trivial example

```
public static void main(String args[]) throws InterruptedException {
  // This is a part of the saved state
  long startTime = System.currentTimeMillis();
  for(int counter: IntStream.range(1, 10000).toArray()) {
    Thread.sleep(1000);
    long currentTime = System.currentTimeMillis();
    System.out.println("Counter: " + counter + "(passed " + (currentTime-startTime) + " ms)");
    startTime = currentTime;
  }
}
```

Trivial example

```
public static void main(String args[]) throws InterruptedException {
  // This is a part of the saved state
  long startTime = System.currentTimeMillis();
  for(int counter: IntStream.range(1, 10000).toArray()) {
    Thread.sleep(1000);
    long currentTime = System.currentTimeMillis();
    System.out.println("Counter: " + counter + "(passed " + (currentTime-startTime) + " ms)");
    startTime = currentTime;
  }
}
```

Trivial example

```
public static void main(String args[]) throws InterruptedException {
  // This is a part of the saved state
  long startTime = System.currentTimeMillis();
  for(int counter: IntStream.range(1, 10000).toArray()) {
    Thread.sleep(1000);
    long currentTime = System.currentTimeMillis();
    System.out.println("Counter: " + counter + "(passed " + (currentTime-startTime) + " ms)");
    startTime = currentTime;
  }
}
```

Trivial example

```
public static void main(String args[]) throws InterruptedException {
  // This is a part of the saved state
  long startTime = System.currentTimeMillis();
  for(int counter: IntStream.range(1, 10000).toArray()) {
    Thread.sleep(1000);
    long currentTime = System.currentTimeMillis();
    System.out.println("Counter: " + counter + "(passed " + (currentTime-startTime) + " ms)");
    startTime = currentTime;
  }
}
```

Trivial example

```
public static void main(String args[]) throws InterruptedException {
  // This is a part of the saved state
  long startTime = System.currentTimeMillis();
  for(int counter: IntStream.range(1, 10000).toArray()) {
    Thread.sleep(1000);
    long currentTime = System.currentTimeMillis();
    System.out.println("Counter: " + counter + "(passed " + (currentTime-startTime) + " ms)");
    startTime = currentTime;
  }
}
```

Trivial example

```
public static void main(String args[]) throws InterruptedException {
  // This is a part of the saved state
  long startTime = System.currentTimeMillis();
  for(int counter: IntStream.range(1, 10000).toArray()) {
    Thread.sleep(1000);
    long currentTime = System.currentTimeMillis();
    System.out.println("Counter: " + counter + "(passed " + (currentTime-startTime) + " ms)");
    startTime = currentTime;
  }
}
```

Trivial example

```
public static void main(String args[]) throws InterruptedException {
  // This is a part of the saved state
  long startTime = System.currentTimeMillis();
  for(int counter: IntStream.range(1, 10000).toArray()) {
    Thread.sleep(1000);
    long currentTime = System.currentTimeMillis();
    System.out.println("Counter: " + counter + "(passed " + (currentTime-startTime) + " ms)");
    startTime = currentTime;
  }
}
```

Docker...

```
FROM bellsoft/liberica-runtime-container:jdk-crac-slim

ADD Example.java /app/Example.java
WORKDIR /app
RUN javac Example.java
ENTRYPOINT java -XX:CRaCCheckpointTo=/app/checkpoint Example
```

Docker...

...is not that simple

```
FROM bellsoft/liberica-runtime-container:jdk-crac-slim

ADD Example.java /app/Example.java
WORKDIR /app
RUN javac Example.java
ENTRYPOINT java -XX:CRaCCheckpointTo=/app/checkpoint Example
```

Docker...

...is not that simple

```
FROM bellsoft/liberica-runtime-container:jdk-crac-slim

ADD Example.java /app/Example.java
WORKDIR /app
RUN javac Example.java
ENTRYPOINT java -XX:CRaCCheckpointTo=/app/checkpoint Example
```

Docker...

...is not that simple

```
FROM bellsoft/liberica-runtime-container:jdk-crac-slim

ADD Example.java /app/Example.java
WORKDIR /app
RUN javac Example.java
ENTRYPOINT java -XX:CRaCCheckpointTo=/app/checkpoint Example
```

Docker...

...is not that simple

```
FROM bellsoft/liberica-runtime-container:jdk-crac-slim

ADD Example.java /app/Example.java
WORKDIR /app
RUN javac Example.java
ENTRYPOINT java -XX:CRaCCheckpointTo=/app/checkpoint Example
```

Docker...

...is not that simple

```
FROM bellsoft/liberica-runtime-container:jdk-crac-slim

ADD Example.java /app/Example.java
WORKDIR /app
RUN javac Example.java
ENTRYPOINT java -XX:CRaCCheckpointTo=/app/checkpoint Example
```

Docker...

...is not that simple

```
FROM bellsoft/liberica-runtime-container:jdk-crac-slim

ADD Example.java /app/Example.java
WORKDIR /app
RUN javac Example.java
ENTRYPOINT java -XX:CRaCCheckpointTo=/app/checkpoint Example
```

This does not create the checkpoint yet!

And then

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run -d pre_crack
```

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run -d pre_crack
# will fail
```

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run --privileged -d pre_crack
```

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run --privileged -d pre_crack
# will work, but security folks will hate us
```

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -d pre_crack
```

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -d pre_crack
# Not so frightening if you think about it
```

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -d pre_crack
# Not so frightening if you think about it
```

- `CAP_SYS_PTRACE` : we need to access the whole process tree

transfer data to or from the memory of arbitrary processes using `process_vm_readv(2)` and `process_vm_writev(2)`

And then

Build it

```
docker build -t pre_crack -f crac2/Dockerfile crac2
```

Run it

```
docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -d pre_crack
# Not so frightening if you think about it
```

- `CAP_SYS_PTRACE` : we need to access the whole process tree

transfer data to or from the memory of arbitrary processes using `process_vm_readv(2)` and `process_vm_writev(2)`

- `CAP_CHECKPOINT_RESTORE` : somehow there is a special cap for this

Update `/proc/sys/kernel/ns_last_pid` ; Read the contents of the symbolic links in `/proc/pid/map_files` for other processes

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

Now we're ready!

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

Now we're ready!

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

Now we’re ready!

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

Now we're ready!

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And then

Checkpoint it

```
ID=$(docker run --cap-add CAP_SYS_PTRACE --cap-add CAP_CHECKPOINT_RESTORE -p8080:8080 -d pre_crac

# wait some time

docker exec -it $ID jcmd 129 JDK.checkpoint
docker commit $ID cracked
```

Now we're ready!

```
docker run --rm -d \
    --entrypoint java \
    --network host cracked:latest \
    -XX:CRaCRestoreFrom=/app/checkpoint
```

And it just works!

And it just works!

Until it doesn't

In our startup

In our startup

- `bot-assistant` module just works
- `chat-api` doesn't work!

In our startup

- `bot-assistant` module just works
- `chat-api` doesn’t work!

Only some projects are guaranteed to work.

In our startup

- `bot-assistant` module just works
- `chat-api` doesn't work!

Only some projects are guaranteed to work.

Others... Request support from the  team:  is not supported for now :(

What are the options?

What are the options?

- Fall back to another solution

What are the options?

- Fall back to another solution
- Request support

What are the options?

- Fall back to another solution
- Request support
- Implement support on our own!

Implementing support on our own.

Custom `MongoClient`

```
public class MongoClientProxy implements MongoClient {
    volatile MongoClient delegate;
    public MongoClientProxy(MongoClient initialClient) {
        this.delegate = initialClient;
    }
    public void close() {
        delegate.close();
    }
    // Delegate everything else the same way
```

Implementing support on our own.

Custom `MongoClient`

```
public class MongoClientProxy implements MongoClient {
    volatile MongoClient delegate;
    public MongoClientProxy(MongoClient initialClient) {
        this.delegate = initialClient;
    }
    public void close() {
        delegate.close();
    }
    // Delegate everything else the same way
```

Implementing support on our own.

Custom `MongoClient`

```
public class MongoClientProxy implements MongoClient {
    volatile MongoClient delegate;
    public MongoClientProxy(MongoClient initialClient) {
        this.delegate = initialClient;
    }
    public void close() {
        delegate.close();
    }
    // Delegate everything else the same way
```

Implementing support on our own.

Custom `MongoClient`

```
public class MongoClientProxy implements MongoClient {
    volatile MongoClient delegate;
    public MongoClientProxy(MongoClient initialClient) {
        this.delegate = initialClient;
    }
    public void close() {
        delegate.close();
    }
    // Delegate everything else the same way
```

Implementing support on our own

Custom CRaC Resource

```
@Component
static public class MongoClientResource implements Resource {

    private final MongoClientProxy mongoClientProxy;
    private final MongoConnectionDetails details;

    public MongoClientResource(MongoClient mongoClientProxy, MongoConnectionDetails details) {
        this.mongoClientProxy = (MongoClientProxy) mongoClientProxy;
        this.details = details;
        Core.getGlobalContext().register(this);
    }

    @Override
    public void beforeCheckpoint(Context<? extends Resource> context) {
```

Implementing support on our own

Custom CRaC Resource

```
static public class MongoClientResource implements Resource {

    private final MongoClientProxy mongoClientProxy;
    private final MongoConnectionDetails details;

    public MongoClientResource(MongoClient mongoClientProxy, MongoConnectionDetails details) {
        this.mongoClientProxy = (MongoClientProxy) mongoClientProxy;
        this.details = details;
        Core.getGlobalContext().register(this);
    }

    @Override
    public void beforeCheckpoint(Context<? extends Resource> context) {
        mongoClientProxy.delegate.close();
    }
```

Implementing support on our own

Custom CRaC Resource

```
@Component
static public class MongoClientResource implements Resource {

    private final MongoClientProxy mongoClientProxy;
    private final MongoConnectionDetails details;

    public MongoClientResource(MongoClient mongoClientProxy, MongoConnectionDetails details) {
        this.mongoClientProxy = (MongoClientProxy) mongoClientProxy;
        this.details = details;
        Core.getGlobalContext().register(this);
    }

    @Override
    public void beforeCheckpoint(Context<? extends Resource> context) {
        mongoClientProxy.delegate.close();
```

Implementing support on our own

Custom CRaC Resource

```
static public class MongoClientResource implements Resource {

    private final MongoClientProxy mongoClientProxy;
    private final MongoConnectionDetails details;

    public MongoClientResource(MongoClient mongoClientProxy, MongoConnectionDetails details) {
        this.mongoClientProxy = (MongoClientProxy) mongoClientProxy;
        this.details = details;
        Core.getGlobalContext().register(this);
    }

    @Override
    public void beforeCheckpoint(Context<? extends Resource> context) {
        mongoClientProxy.delegate.close();
    }
```

Implementing support on our own

Custom CRaC Resource

```
private final MongoClientProxy mongoClientProxy;
private final MongoConnectionDetails details;

public MongoClientResource(MongoClient mongoClientProxy, MongoConnectionDetails details) {
    this.mongoClientProxy = (MongoClientProxy) mongoClientProxy;
    this.details = details;
    Core.getGlobalContext().register(this);
}

@Override
public void beforeCheckpoint(Context<? extends Resource> context) {
    mongoClientProxy.delegate.close();
}
```

Implementing support on our own

Custom CRaC Resource

```
        this.mongoClientProxy = (MongoClientProxy) mongoClientProxy;
        this.details = details;
        Core.getGlobalContext().register(this);
    }

    @Override
    public void beforeCheckpoint(Context<? extends Resource> context) {
        mongoClientProxy.delegate.close();
    }

    @Override
    public void afterRestore(Context<? extends Resource> context) {
        mongoClientProxy.delegate = MongoClients.create(details.getConnectionString());
    }
}
```

Implementing support on our own

Custom CRaC Resource

```
        this.details = details;
        Core.getGlobalContext().register(this);
    }

    @Override
    public void beforeCheckpoint(Context<? extends Resource> context) {
        mongoClientProxy.delegate.close();
    }

    @Override
    public void afterRestore(Context<? extends Resource> context) {
        mongoClientProxy.delegate = MongoClients.create(details.getConnectionString());
    }
}
```

Implementing support on our own

Custom CRaC Resource

```
        this.details = details;
        Core.getGlobalContext().register(this);
    }

    @Override
    public void beforeCheckpoint(Context<? extends Resource> context) {
        mongoClientProxy.delegate.close();
    }

    @Override
    public void afterRestore(Context<? extends Resource> context) {
        mongoClientProxy.delegate = MongoClients.create(details.getConnectionString());
    }
}
```

Implementing support on our own

Replacing `MongoClient` in the context

```
@Bean
@Primary
public MongoClient mongoClient(MongoConnectionDetails details) {
    MongoClient initialClient = MongoClients.create(details.getConnectionString());
    return new MongoClientProxy(initialClient);
}
```

Implementing support on our own

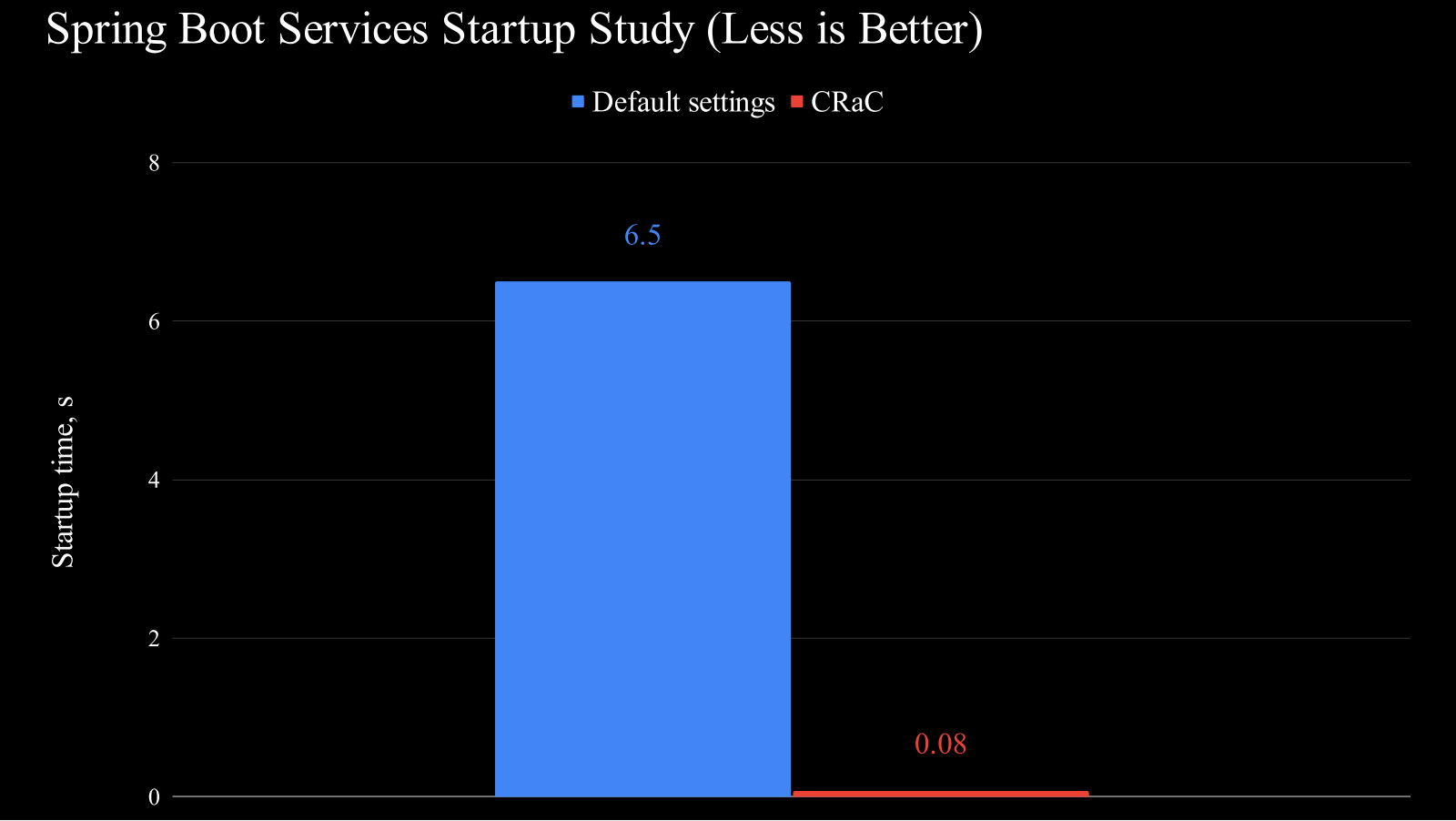
Replacing `MongoClient` in the context

```
@Bean
@Primary
public MongoClient mongoClient(MongoConnectionDetails details) {
    MongoClient initialClient = MongoClients.create(details.getConnectionString());
    return new MongoClientProxy(initialClient);
}
```

Implementing support on our own

Replacing `MongoClient` in the context

```
@Bean
@Primary
public MongoClient mongoClient(MongoConnectionDetails details) {
    MongoClient initialClient = MongoClients.create(details.getConnectionString());
    return new MongoClientProxy(initialClient);
}
```


Spring Boot Services Startup Study (Less is Better)
Default settings
CRaC
Startup time, s
8
6
4
2
0
6.5
0.08

And they all lived happily ever after!

Quick Recap

Which hero are you?

🛡️ → AppCDS for the smoothest integration
🧙 → Project Leyden EA builds to prepare to use it in the future
🎒 → Native Image for fast start
✊🏼 → CRaC for almost instant start

Testing locally? Use a power cord!

This story came to an end, yours has just begun

Go for it!

Thank you for your attention!

Find us at

- @asm0dey.site
- @cat-edelveis.bsky.social
- asm0di0
- cat_edelveis