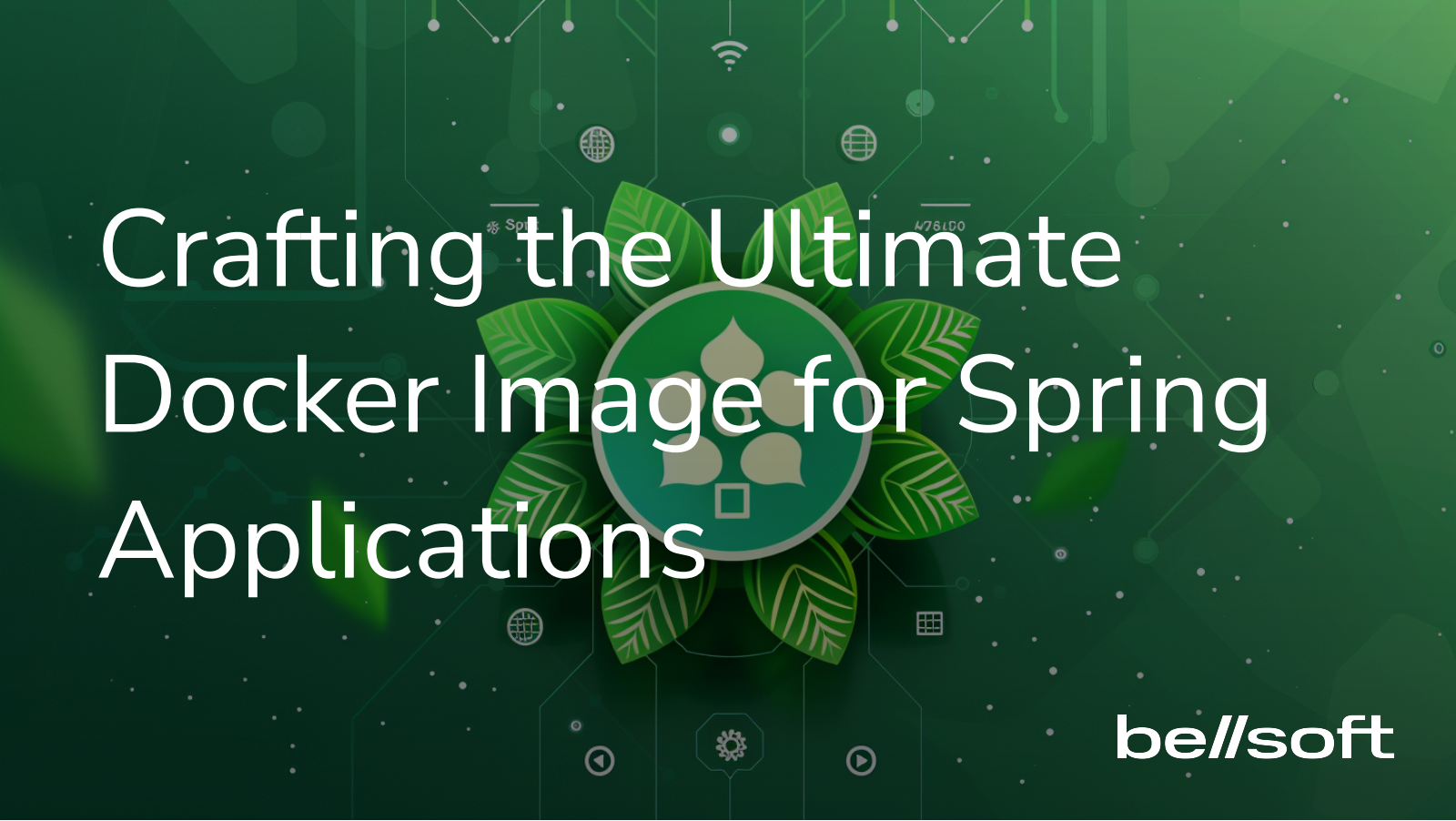

Crafting the Ultimate Docker Image for Spring Applications
bellsoft

whoami

Pasha Finkelshteyn @asm0dey.site @asm0di0

whoami

- Pasha Finkelshteyn

whoami

- Pasha Finkelshteyn
- Dev 🥑 at BellSoft

whoami

- Pasha Finkelshteyn
- Dev at BellSoft
- ≈10 years in JVM. Mostly and

whoami

- Pasha Finkelshteyn
- Dev 🥑 at BellSoft
- ≈10 years in JVM. Mostly ☕ and
- And 🍃

whoami

- Pasha Finkelshteyn
- Dev 🥑 at BellSoft
- ≈10 years in JVM. Mostly ☕ and
- And 🌱
- 🐦 asm0di0

whoami

- Pasha Finkelshteyn
- Dev at BellSoft
- ≈10 years in JVM. Mostly and
- And
- asm0di0
- @asm0dey@fosstodon.org

BellSoft

- Vendor of Liberica JDK
- Contributor to the OpenJDK
- Author of ARM32 support in JDK
- Own base images
- Own Linux: Alpaquita

Liberica is the JDK officially recommended by

BellSoft

- Vendor of Liberica JDK
- Contributor to the OpenJDK
- Author of ARM32 support in JDK
- Own base images
- Own Linux: Alpaquita

Liberica is the JDK officially recommended by

We know our stuff!

So, what is ultimate?

- Smallest
- Fatsest startup
- Something inbetween

So, let’s look at all of them!

How do we create an image?

Let’s start from trivial.

```
1   FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3   COPY . /app
4   RUN cd /app && ./gradlew build
5   ENTRYPOINT     java -jar /app/build/libs/spring-petclinic*.jar
```

How do we create an image?

Let’s start from trivial.

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY . /app
4  RUN cd /app && ./gradlew build
5  ENTRYPOINT     java -jar /app/build/libs/spring-petclinic*.jar
```

How do we create an image?

Let’s start from trivial.

```
1   FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3   COPY . /app
4   RUN cd /app && ./gradlew build
5   ENTRYPOINT     java -jar /app/build/libs/spring-petclinic*.jar
```

How do we create an image?

Let’s start from trivial.

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY . /app
4  RUN cd /app && ./gradlew build
5  ENTRYPOINT     java -jar /app/build/libs/spring-petclinic*.jar
```

How do we create an image?

Let’s start from trivial.

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY . /app
4  RUN cd /app && ./gradlew build
5  ENTRYPOINT     java -jar /app/build/libs/spring-petclinic*.jar
```

`Dockerfile` directives

1. Each directive creates a layer of the image.
2. Layers are *immutable*
3. Some layers are zero-sized

`Dockerfile` directives

1. Each directive creates a layer of the image.
2. Layers are *immutable*
3. Some layers are zero-sized

```
1  FROM bash
2
3  COPY . /app
4  RUN rm -rf /app
```

`Dockerfile` directives

1. Each directive creates a layer of the image.
2. Layers are *immutable*
3. Some layers are zero-sized

```
1  FROM bash
2
3  COPY . /app
4  RUN rm -rf /app
```

`Dockerfile` directives

1. Each directive creates a layer of the image.
2. Layers are *immutable*
3. Some layers are zero-sized (for example `RUN rm -rf /app`)

```
1  FROM bash
2
3  COPY . /app
4  RUN rm -rf /app
```

`Dockerfile` directives

1. Each directive creates a layer of the image.
2. Layers are *immutable*
3. Some layers are zero-sized (for example `RUN rm -rf /app`)

```
1  FROM bash
2
3  COPY . /app
4  RUN rm -rf /app
```

4. We would like image to be light, but it's not :(

Our `Dockerfile`

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY . /app
4  RUN cd /app && ./gradlew build
5  ENTRYPOINT java -jar /app/build/libs/spring-petclinic*.jar
```

Our `Dockerfile`

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY . /app
4  RUN cd /app && ./gradlew build
5  ENTRYPOINT java -jar /app/build/libs/spring-petclinic*.jar
```

Our `Dockerfile`

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY . /app
4  RUN cd /app && ./gradlew build
5  ENTRYPOINT java -jar /app/build/libs/spring-petclinic*.jar
```

Our `Dockerfile`

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY . /app
4  RUN cd /app && ./gradlew build
5  ENTRYPOINT java -jar /app/build/libs/spring-petclinic*.jar
```

Result

```
Cmp   Size  Command
     10 MB  FROM blobs
    109 MB  cd /app && ./gradlew build -xtest
     85 MB  (missing)
    591 MB  (missing)
```

Result

```
1   Cmp   Size  Command
2         10 MB  FROM blobs
3        109 MB  cd /app && ./gradlew build -xtest
4         85 MB  (missing)
5        591 MB  (missing)
```

109 MB of Java

Result

```
1   Cmp   Size  Command
2         10 MB  FROM blobs
3        109 MB  cd /app && ./gradlew build -xtest
4         85 MB  (missing)
5        591 MB  (missing)
```

109 MB of Java

85 MB of the app

Result

```
Cmp   Size  Command
      10 MB  FROM blobs
     109 MB  cd /app && ./gradlew build -xtest
      85 MB  (missing)
     591 MB  (missing)
```

109 MB of Java

85 MB of the app

591 MB of Java build caches

600+ MB are changed on every build!

Why do we care? We care because:

1. `push` takes longer time to start (update is longer)
2. `pull` takes longer (update takes longer & scaling takes longer)

Also, more disk psace is inefficiently used

Optimizing. Round 1.

Let’s build it outside of container

```
FROM bellsoft/liberica-runtime-container:jdk-25-musl

COPY . /app
RUN cd /app && ./gradlew build
ENTRYPOINT java -jar /app/build/libs/spring-petclinic*.jar
```

Optimizing. Round 1.

Let’s build it outside of container

```
FROM bellsoft/liberica-runtime-container:jdk-25-musl

COPY build/libs/spring-petclinic-3.3.0.jar /app/app.jar
CMD java -jar /app/app.jar
```

Optimizing. Round 1.

Let’s build it outside of container

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl
2
3  COPY build/libs/spring-petclinic-3.3.0.jar /app/app.jar
4  CMD java -jar /app/app.jar
```

Just copying the prebuilt `jar` file

Results

```
Cmp   Size  Command
      10 MB  FROM blobs
     109 MB  cd /app && ./gradlew build -xtest
      85 MB  (missing)
     591 MB  (missing)
```

No Gradle caches and build dir!

Results

```
Cmp   Size  Command
      10 MB  FROM blobs
     109 MB  (missing)
      69 MB  (missing)
```

No Gradle caches and build dir!

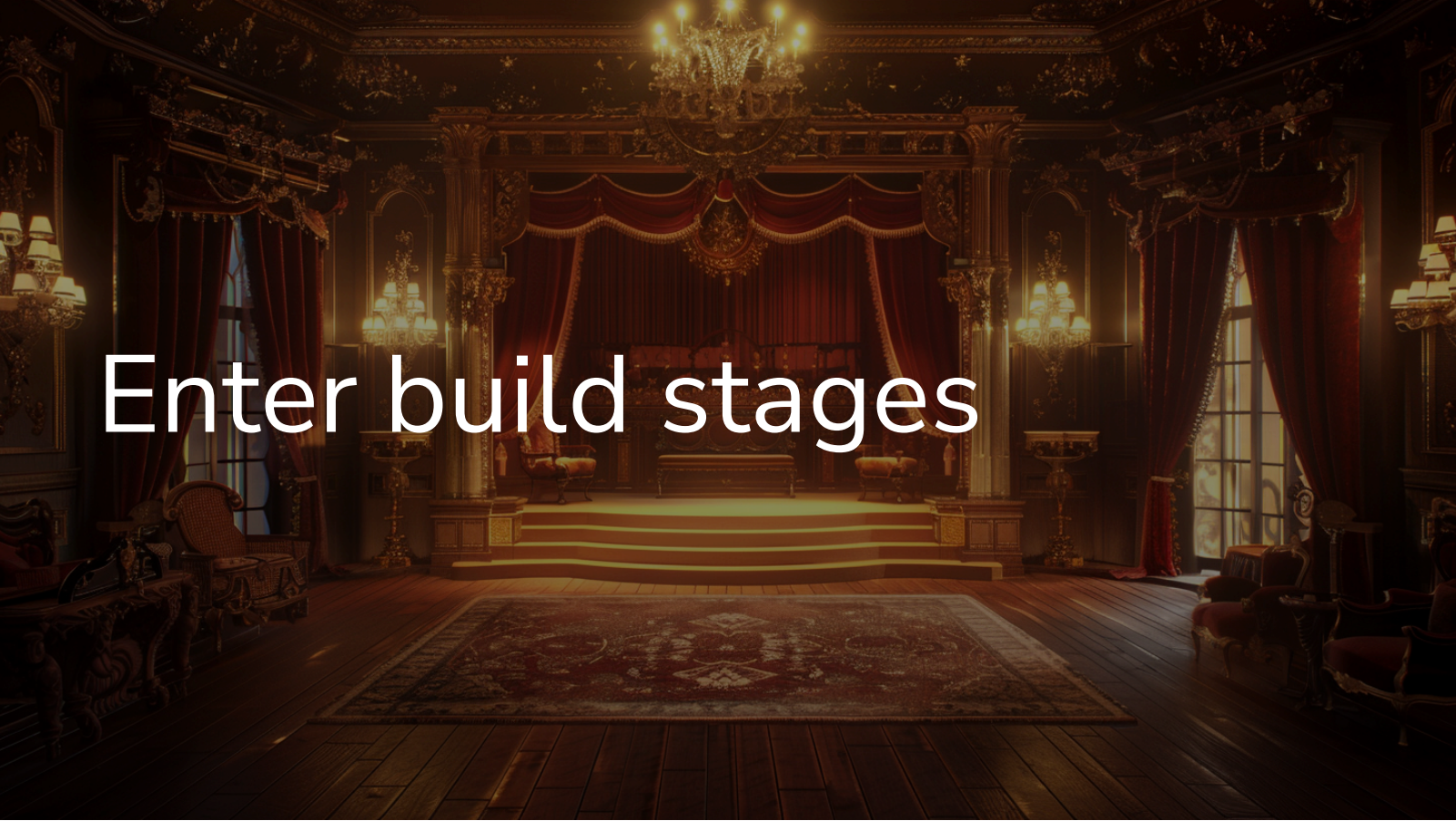
Saved 600+ MB

But the build is not clean now :(

Saved 600+ MB

But the build is not clean now :(

We have to build everything outside, what if environment affects the build?


Enter build stages

Multi-stage builds

Holy grail of pure builds

- Allow clean builds
- Allow optimal packaging
- Allow different base images

Staged example

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-25-musl as runner
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
```

Staged example

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-25-musl as runner
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
```

Staged example

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-25-musl as runner
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
```

Staged example

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-25-musl as runner
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
```

Staged example

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-25-musl as runner
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
```

Staged example

```
1  FROM bellsoft/liberica-runtime-container:jdk-25-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-25-musl as runner
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
```

Result

- jar - same size
- JRE - 48.28

That’s all folks!

Pasha Finkelshteyn @asm0dey.site @asm0di0

That’s all folks!

Or is it?

Pasha Finkelshteyn @asm0dey.site @asm0di0

What’s these 60 MiB?

We have to pull 60 MiB of petclinic on every small commit!

Is there a way to optimize it?

Layers!

```
1  FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9  WORKDIR /app
10 RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
```

- Build image

Layers!

```
1  FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9  WORKDIR /app
10 RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11
```

- Build image
- Introduce new "optimizer" stage

Layers!

```
4   RUN cd /app && ./gradlew build -xtest
5
6   FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8   COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9   WORKDIR /app
10  RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11
12  FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner
13
14  ENTRYPOINT ["java", "org.springframework.boot.loader.launch.JarLauncher"]
```

- Build image
- Introduce new "optimizer" stage

Layers!

```
6   FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8   COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9   WORKDIR /app
10  RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11
12  FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner
13
14  ENTRYPOINT ["java", "org.springframework.boot.loader.launch.JarLauncher"]
15  COPY --from=optimizer /app/extracted/dependencies/ ./
```

- Build image
- Introduce new "optimizer" stage
- Extract the jar to layered structure

Layers!

```
8   COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9   WORKDIR /app
10  RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11
12  FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner
13
14  ENTRYPOINT ["java", "org.springframework.boot.loader.launch.JarLauncher"]
15  COPY --from=optimizer /app/extracted/dependencies/ ./
16  COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
17  COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
```

- Build image
- Introduce new "optimizer" stage
- Extract the jar to layered structure

Layers!

```
9   WORKDIR /app
10  RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11
12  FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner
13
14  ENTRYPOINT ["java", "org.springframework.boot.loader.launch.JarLauncher"]
15  COPY --from=optimizer /app/extracted/dependencies/ ./
16  COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
17  COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
18  COPY --from=optimizer /app/extracted/application/ ./
```

- Build image
- Introduce new "optimizer" stage
- Extract the jar to layered structure
- Copy layers

Layers!

```
9   WORKDIR /app
10  RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11
12  FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner
13
14  ENTRYPOINT ["java", "org.springframework.boot.loader.launch.JarLauncher"]
15  COPY --from=optimizer /app/extracted/dependencies/ ./
16  COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
17  COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
18  COPY --from=optimizer /app/extracted/application/ ./
```

- Build image
- Introduce new "optimizer" stage
- Extract the jar to layered structure
- Copy layers

Layers

Because Spring Boot jar is complex!

```
Usage:
  java -Djarmode=tools -jar my-app.jar

Available commands:
  extract      Extract the contents from the jar
  list-layers  List layers from the jar that can be extracted
```

Layers

```
extracted
├── application
│   └── spring-petclinic-4.0.0-SNAPSHOT.jar
├── dependencies
│   └── lib
│       ├── angus-activation-2.0.2.jar
│       ├── antlr4-runtime-4.13.2.jar
│       ├── aspectjweaver-1.9.24.jar
│       ├── attoparser-2.0.7.RELEASE.jar
│       ├── bootstrap-5.3.8.jar
│       ├── byte-buddy-1.17.7.jar
│       ├── cache-api-1.1.1.jar
│       ├── caffeine-3.2.2.jar
│       ├── checker-qual-3.49.3.jar
│       ├── classmate-1.7.0.jar
│       ├── commons-logging-1.3.5.jar
```

Layers

```
91  │   │   │   ├── spring-orm-7.0.0-M9.jar
92  │   │   │   ├── spring-tx-7.0.0-M9.jar
93  │   │   │   ├── spring-web-7.0.0-M9.jar
94  │   │   │   ├── spring-webmvc-7.0.0-M9.jar
95  │   │   │   ├── thymeleaf-3.1.3.RELEASE.jar
96  │   │   │   ├── thymeleaf-spring6-3.1.3.RELEASE.jar
97  │   │   │   ├── tomcat-embed-core-11.0.11.jar
98  │   │   │   ├── tomcat-embed-el-11.0.11.jar
99  │   │   │   ├── tomcat-embed-websocket-11.0.11.jar
100 │   │   │   ├── txw2-4.0.5.jar
101 │   │   │   ├── unbescape-1.1.6.RELEASE.jar
102 │   │   │   └── webjars-locator-lite-1.1.1.jar
103 │   ├── snapshot-dependencies
104 │   └── spring-boot-loader
105 │
106 6 directories, 98 files
```

Layers

```
91      │       ├── spring-orm-7.0.0-M9.jar
92      │       ├── spring-tx-7.0.0-M9.jar
93      │       ├── spring-web-7.0.0-M9.jar
94      │       ├── spring-webmvc-7.0.0-M9.jar
95      │       ├── thymeleaf-3.1.3.RELEASE.jar
96      │       ├── thymeleaf-spring6-3.1.3.RELEASE.jar
97      │       ├── tomcat-embed-core-11.0.11.jar
98      │       ├── tomcat-embed-el-11.0.11.jar
99      │       ├── tomcat-embed-websocket-11.0.11.jar
100     │       ├── txw2-4.0.5.jar
101     │       ├── unbescape-1.1.6.RELEASE.jar
102     │       └── webjars-locator-lite-1.1.1.jar
103     ├── snapshot-dependencies
104     └── spring-boot-loader
105
106 6 directories, 98 files
```

Layers

```
4  ├── dependencies
5  │   └── lib
6  │       ├── angus-activation-2.0.2.jar
7  │       ├── antlr4-runtime-4.13.2.jar
8  │       ├── aspectjweaver-1.9.24.jar
9  │       ├── attoparser-2.0.7.RELEASE.jar
10 │       ├── bootstrap-5.3.8.jar
11 │       ├── byte-buddy-1.17.7.jar
12 │       ├── cache-api-1.1.1.jar
13 │       ├── caffeine-3.2.2.jar
14 │       ├── checker-qual-3.49.3.jar
15 │       ├── classmate-1.7.0.jar
16 │       ├── commons-logging-1.3.5.jar
17 │       ├── context-propagation-1.2.0-M1.jar
18 │       ├── error_prone_annotations-2.40.0.jar
19 │       ├── font-awesome-4.7.0.jar
```

Layers

```
extracted
├── application
│   └── spring-petclinic-4.0.0-SNAPSHOT.jar
├── dependencies
│   └── lib
│       ├── angus-activation-2.0.2.jar
│       ├── antlr4-runtime-4.13.2.jar
│       ├── aspectjweaver-1.9.24.jar
│       ├── attoparser-2.0.7.RELEASE.jar
│       ├── bootstrap-5.3.8.jar
│       ├── byte-buddy-1.17.7.jar
│       ├── cache-api-1.1.1.jar
│       ├── caffeine-3.2.2.jar
│       ├── checker-qual-3.49.3.jar
│       ├── classmate-1.7.0.jar
│       ├── commons-logging-1.3.5.jar
```

Layered image structure

```
> du -h --max-depth=1 extracted
66M     extracted/dependencies
0       extracted/spring-boot-loader
0       extracted/snapshot-dependencies
892K    extracted/application
66M     extracted
```

Layered image structure

```
> du -h --max-depth=1 extracted
66M     extracted/dependencies
0       extracted/spring-boot-loader
0       extracted/snapshot-dependencies
892K    extracted/application
66M     extracted
```

- Dependencies: ~66MiB

Layered image structure

```
> du -h --max-depth=1 extracted
66M     extracted/dependencies
0       extracted/spring-boot-loader
0       extracted/snapshot-dependencies
892K    extracted/application
66M     extracted
```

- Dependencies: ~66MiB
- Application: ~892K

66.0MiB > 60Mib!

Pasha Finkelshteyn @asm0dey.site @asm0di0

What are we optimizing?

Pull size!

Pull size here is usually only around 1MiB!

We just reinvented how the BellSoft's buildpack works!

And it is amazing

Pasha Finkelshteyn @asm0dey.site @asm0di0

Diversion: buildpacks

https://paketo.io/

Trivial usage:

```
/usr/sbin/pack build petclinic \
  --builder bellsoft/buildpacks.builder:musl \
  --path . \
  -e BP_JVM_VERSION=21
```

Diversion: buildpacks

https://paketo.io/

Trivial usage:

```
/usr/sbin/pack build petclinic \
  --builder bellsoft/buildpacks.builder:musl \
  --path . \
  -e BP_JVM_VERSION=21
```

Diversion: buildpacks

https://paketo.io/

Trivial usage:

```
/usr/sbin/pack build petclinic \
  --builder bellsoft/buildpacks.builder:musl \
  --path . \
  -e BP_JVM_VERSION=21
```

Diversion: buildpacks

https://paketo.io/

Trivial usage:

```
/usr/sbin/pack build petclinic \
  --builder bellsoft/buildpacks.builder:musl \
  --path . \
  -e BP_JVM_VERSION=21
```

Result

```
1   ID         TAG              SIZE      COMMAND
2   2774e3f214 petclinic:latest 0B        Buildpacks Process Types
3   <missing>                   1.44kiB   Buildpacks Launcher Config
4   <missing>                   2.44MiB   Buildpacks Application Launcher
5   <missing>                   59.17MiB  Application Layer
6   <missing>                   729.21kiB Software Bill-of-Materials
7   <missing>                   97.87MiB  Layer: 'jre', Created by buildpack: bellsoft/buildpacl
8   <missing>                   200.00B   Layer: 'java-security-properties', Created by buildpac
9   <missing>                   4.27MiB   Layer: 'helper', Created by buildpack: bellsoft/buildp
10  <missing>                   2.97kiB
11  <missing>                   7.48MiB
```

Result

```
ID           TAG               SIZE      COMMAND
2774e3f214 petclinic:latest 0B        Buildpacks Process Types
<missing>                  1.44kiB   Buildpacks Launcher Config
<missing>                  2.44MiB   Buildpacks Application Launcher
<missing>                  59.17MiB  Application Layer
<missing>                  729.21kiB Software Bill-of-Materials
<missing>                  97.87MiB  Layer: 'jre', Created by buildpack: bellsoft/buildpacl
<missing>                  200.00B   Layer: 'java-security-properties', Created by buildpac
<missing>                  4.27MiB   Layer: 'helper', Created by buildpack: bellsoft/buildp
<missing>                  2.97kiB
<missing>                  7.48MiB
```

Result

```
ID           TAG               SIZE      COMMAND
2774e3f214 petclinic:latest 0B        Buildpacks Process Types
<missing>                  1.44kiB   Buildpacks Launcher Config
<missing>                  2.44MiB   Buildpacks Application Launcher
<missing>                  59.17MiB  Application Layer
<missing>                  729.21kiB Software Bill-of-Materials
<missing>                  97.87MiB  Layer: 'jre', Created by buildpack: bellsoft/buildpacl
<missing>                  200.00B   Layer: 'java-security-properties', Created by buildpac
<missing>                  4.27MiB   Layer: 'helper', Created by buildpack: bellsoft/buildp
<missing>                  2.97kiB
<missing>                  7.48MiB
```

Result

```
ID           TAG               SIZE      COMMAND
2774e3f214 petclinic:latest 0B        Buildpacks Process Types
<missing>                  1.44kiB   Buildpacks Launcher Config
<missing>                  2.44MiB   Buildpacks Application Launcher
<missing>                  59.17MiB  Application Layer
<missing>                  729.21kiB Software Bill-of-Materials
<missing>                  97.87MiB  Layer: 'jre', Created by buildpack: bellsoft/buildpacl
<missing>                  200.00B   Layer: 'java-security-properties', Created by buildpac
<missing>                  4.27MiB   Layer: 'helper', Created by buildpack: bellsoft/buildp
<missing>                  2.97kiB
<missing>                  7.48MiB
```

Buildpacks

Why do we need them?

- Buildpacks transform your application source code into container images
- The Paketo open source project provides production-ready buildpacks for the most popular languages and frameworks
- Use Paketo Buildpacks to easily build your apps and keep them updated

- Reminds of s2i images
- Build better than default
- Spring-aware

Is this all?

We've optimized pull/push size, right?

Optimization is multidimensional

When startup time is more important...

There are several solutions for the Spring application startup time

1. Class Data Sharing (CDS)
2. Project Leyden
3. Coordinated Restore at Checkpoint
4. Native Image

Class Data Sharing (CDS)

- **When**: Java 5 (!)
- **Why**: reduce the startup and memory footprint of multiple JVM instances running on the same host
- **How**: stores data in an archive

How does it help us?

Class Data Sharing (CDS)

- **When**: Java 5 (!)
- **Why**: reduce the startup and memory footprint of multiple JVM instances running on the same host
- **How**: stores data in an archive

How does it help us?

It does not! 🧌

Class Data Sharing (CDS)

- **When**: Java 5 (!)
- **Why**: reduce the startup and memory footprint of multiple JVM instances running on the same host
- **How**: stores data in an archive

How does it help us?

It does not!  But

Application Class Data Sharing

- **When**: Java 10
- **Why**: add application classes to the archive

And then:

- **When**: Java 13, JEP 350
- **Why**: allow addition of classes into the archive upon app exit!

And this helps! How?

How to use AppCDS with Spring?

- `-XX:ArchiveClassesAtExit=application.jsa` to create an archive
- `-Dspring.context.exit=onRefresh` to start and immediately exit the application

NB:

1. Use the same JDK
2. Use the same arguments

And even better!

Spring AOT

Add `-Dspring.aot.enabled=true`

Even more classes!!!

Practice

```
1  FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9  WORKDIR /app
10 RUN java -Djarmode=tools -jar /app/app.jar extract --layers --launcher
11
12 FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner
13
14 ENTRYPOINT ["java", "org.springframework.boot.loader.launch.JarLauncher"]
15 COPY --from=optimizer /app/extracted/dependencies/ ./
16 COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
17 COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
18 COPY --from=optimizer /app/extracted/application/ ./
```

Practice

```
1  FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9  WORKDIR /app
10 RUN java -Djarmode=tools -jar /app/app.jar extract --layers --launcher
11
12 FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner
13
14 ENTRYPOINT ["java", "org.springframework.boot.loader.launch.JarLauncher"]
15 COPY --from=optimizer /app/extracted/dependencies/ ./
16 COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
17 COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
18 COPY --from=optimizer /app/extracted/application/ ./
```

Practice

```
1  FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9  WORKDIR /app
10 RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11 FROM bellsoft/liberica-runtime-container:jre-cds-slim-musl as runner
12
13 ENTRYPOINT ["java", "-jar", "app.jar"]
14 COPY --from=optimizer /app/extracted/dependencies/ ./
15 COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
16 COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
17 COPY --from=optimizer /app/extracted/application/ ./
```

Practice

```
1  FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-slim-musl as optimizer
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9  WORKDIR /app
10 RUN java -Djarmode=tools -jar /app/app.jar extract --layers --destination extracted
11 FROM bellsoft/liberica-runtime-container:jre-cds-slim-musl as runner
12
13 ENTRYPOINT ["java", "-jar", "app.jar"]
14 COPY --from=optimizer /app/extracted/dependencies/ ./
15 COPY --from=optimizer /app/extracted/spring-boot-loader/ ./
16 COPY --from=optimizer /app/extracted/snapshot-dependencies/ ./
17 COPY --from=optimizer /app/extracted/application/ ./
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-cds-slim-musl as runner

ENTRYPOINT ["java", "-jar", "app.jar"]
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.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-cds-slim-musl as runner

ENTRYPOINT ["java", "-jar", "app.jar"]
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.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-cds-slim-musl as runner

ENTRYPOINT ["java", \
            "-Dspring.aot.enabled=true", \
            "-XX:SharedArchiveFile=application.jsa", \
            "-jar", "app.jar"]
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.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-cds-slim-musl as runner

ENTRYPOINT ["java", \
            "-Dspring.aot.enabled=true", \
            "-XX:SharedArchiveFile=application.jsa", \
            "-jar", "app.jar"]
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.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-cds-slim-musl as runner

ENTRYPOINT ["java", \
            "-Dspring.aot.enabled=true", \
            "-XX:SharedArchiveFile=application.jsa", \
            "-jar", "app.jar"]
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.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner

ENTRYPOINT ["java", \
            "-Dspring.aot.enabled=true", \
            "-XX:AOTCache=app.aot", \
            "-jar", "app.jar"]
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:AOTCacheOutput=app.aot \
  -Dspring.context.exit=onRefresh \
  -jar app.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner

ENTRYPOINT ["java", \
            "-Dspring.aot.enabled=true", \
            "-XX:AOTCache=app.aot", \
            "-jar", "app.jar"]
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:AOTCacheOutput=app.aot \
  -Dspring.context.exit=onRefresh \
  -jar app.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner

ENTRYPOINT ["java", \
            "-Dspring.aot.enabled=true", \
            "-XX:AOTCache=app.aot", \
            "-jar", "app.jar"]
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:AOTCacheOutput=app.aot \
  -Dspring.context.exit=onRefresh \
  -jar app.jar
```

Practice

```
#omitted
FROM bellsoft/liberica-runtime-container:jre-slim-musl as runner

ENTRYPOINT ["java", \
            "-Dspring.aot.enabled=true", \
            "-XX:AOTCache=app.aot", \
            "-jar", "app.jar"]
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:AOTCacheOutput=app.aot \
  -Dspring.context.exit=onRefresh \
  -jar app.jar
```

What does it cost ?

```
111M Nov  4 23:47 app.aot
```

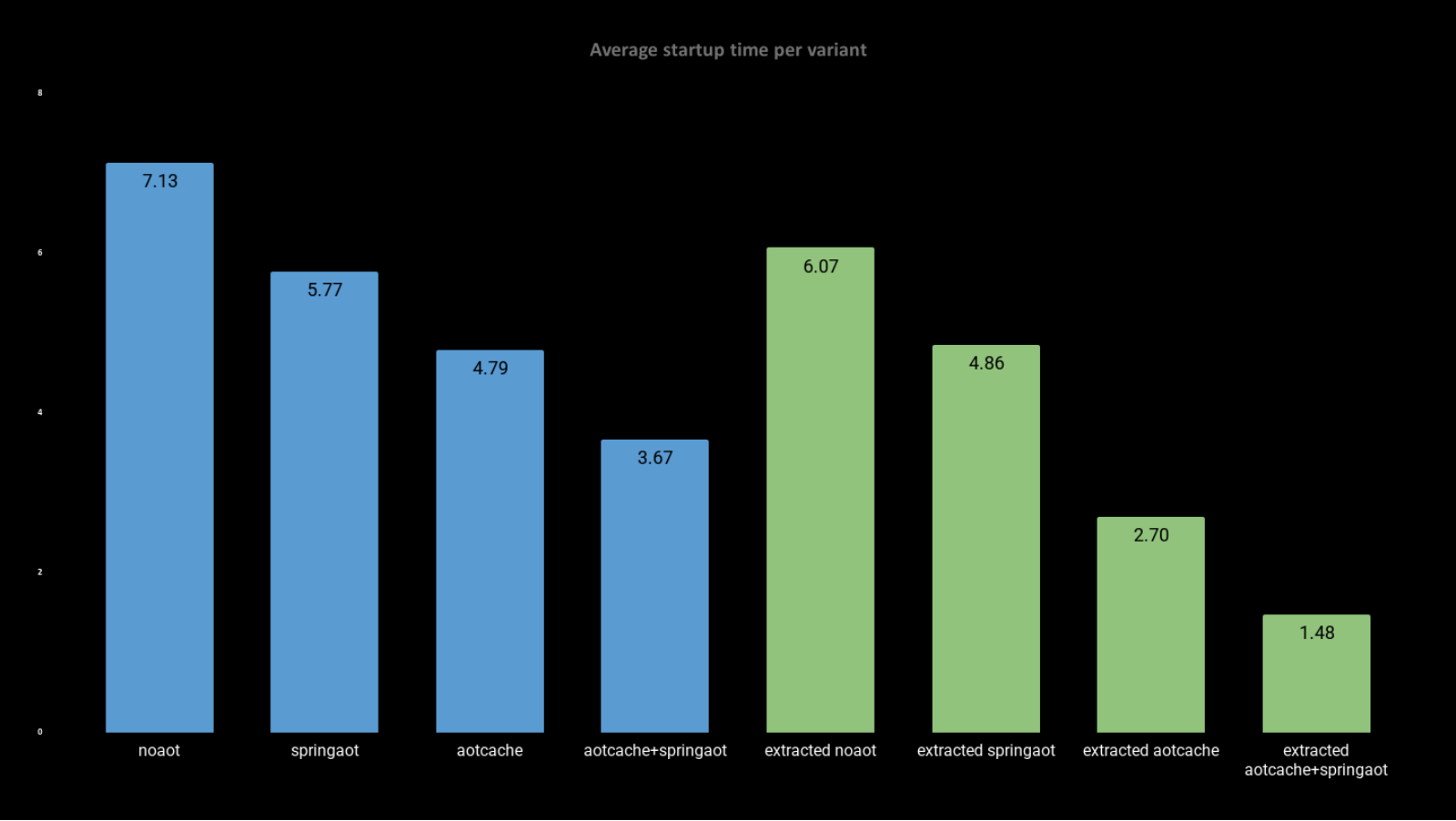
Which is not small at all!

What does it cost ?

```
1   111M Nov  4 23:47 app.aot
```

Which is not small at all!

But you're trading pull speed for startup speed


Average startup time per variant
8
6
4
2
0
7.13
5.77
4.79
3.67
6.07
4.86
2.70
1.48
noaot
springaot
aotcache
aotcache+springaot
extracted noaot
extracted springaot
extracted aotcache
extracted aotcache+springaot

Pushing further with CRaC

CRaC: Coordinated Restore at Checkpoint

> 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/

In a perfect world it should be:

```
1   FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3   COPY . /app
4   RUN cd /app && ./gradlew build
5
6   FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer
7
8   COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9   WORKDIR /app
10  RUN java -Dspring.context.checkpoint=onRefresh -XX:CRaCCheckpointTo=/checkpoint -jar /app/ap
11
12  FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner
13
14  ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
15  COPY --from=optimizer /app/app.jar /app/app.jar
16  COPY --from=optimizer /checkpoint /checkpoint
```

In a perfect world it should be:

```
1   FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3   COPY . /app
4   RUN cd /app && ./gradlew build
5
6   FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer
7
8   COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9   WORKDIR /app
10  RUN java -Dspring.context.checkpoint=onRefresh -XX:CRaCCheckpointTo=/checkpoint -jar /app/ap
11
12  FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner
13
14  ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
15  COPY --from=optimizer /app/app.jar /app/app.jar
16  COPY --from=optimizer /checkpoint /checkpoint
```

In a perfect world it should be:

```
1   FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3   COPY . /app
4   RUN cd /app && ./gradlew build
5
6   FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer
7
8   COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9   WORKDIR /app
10  RUN java -Dspring.context.checkpoint=onRefresh -XX:CRaCCheckpointTo=/checkpoint -jar /app/ap
11
12  FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner
13
14  ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
15  COPY --from=optimizer /app/app.jar /app/app.jar
16  COPY --from=optimizer /checkpoint /checkpoint
```

But in reality

```
#12 6.051       Suppressed: java.lang.RuntimeException: Native checkpoint failed.
#12 6.051               at java.base/jdk.crac.Core.translateJVMExceptions(Core.java:114) ~[r
#12 6.051               at java.base/jdk.crac.Core.checkpointRestore1(Core.java:192) ~[na:na
#12 6.051               at java.base/jdk.crac.Core.checkpointRestore(Core.java:299) ~[na:na]
#12 6.051               at java.base/jdk.crac.Core.checkpointRestore(Core.java:278) ~[na:na]
#12 6.051               at java.base/javax.crac.Core.checkpointRestore(Core.java:73) ~[na:na
#12 6.051               at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(
#12 6.051               at java.base/java.lang.reflect.Method.invoke(Method.java:580) ~[na:n
#12 6.051               at org.crac.Core$Compat.checkpointRestore(Core.java:141) ~[crac-1.4.
#12 6.051                ... 17 common frames omitted
```

CRaC is hard!

Let’s try to fix it with arcane magic

```
1  FROM bellsoft/liberica-runtime-container:jdk-musl as builder
2
3  COPY . /app
4  RUN cd /app && ./gradlew build -xtest
5
6  FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer
7
8  COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
9  WORKDIR /app
10 RUN java -Dspring.context.checkpoint=onRefresh -XX:CRaCCheckpointTo=/checkpoint -jar /app/app.jar
11
12 FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner
13
14 ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
15 COPY --from=optimizer /app/app.jar /app/app.jar
16 COPY --from=optimizer /checkpoint /checkpoint
```

CRaC is hard!

Let’s try to fix it with arcane magic

```
# syntax=docker/dockerfile:1-labs
FROM bellsoft/liberica-runtime-container:jdk-musl as builder

COPY . /app
RUN cd /app && ./gradlew build -xtest

FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer

COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
WORKDIR /app
RUN java -Dspring.context.checkpoint=onRefresh -XX:CRaCCheckpointTo=/checkpoint -jar /app/app.jar

FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner

ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
COPY --from=optimizer /app/app.jar /app/app.jar
COPY --from=optimizer /checkpoint /checkpoint
```

CRaC is hard!

Let’s try to fix it with arcane magic

```
# syntax=docker/dockerfile:1-labs
FROM bellsoft/liberica-runtime-container:jdk-musl as builder

COPY . /app
RUN cd /app && ./gradlew build -xtest

FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer

COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
WORKDIR /app
RUN java -Dspring.context.checkpoint=onRefresh -XX:CRaCCheckpointTo=/checkpoint -jar /app/app.jar

FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner

ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
COPY --from=optimizer /app/app.jar /app/app.jar
COPY --from=optimizer /checkpoint /checkpoint
```

CRaC is hard!

Let’s try to fix it with arcane magic

```
# syntax=docker/dockerfile:1-labs
FROM bellsoft/liberica-runtime-container:jdk-musl as builder

COPY . /app
RUN cd /app && ./gradlew build -xtest

FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer

COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
WORKDIR /app
RUN java -Dspring.context.checkpoint=onRefresh -XX:CRaCCheckpointTo=/checkpoint -jar /app/app.jar

FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner

ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
COPY --from=optimizer /app/app.jar /app/app.jar
COPY --from=optimizer /checkpoint /checkpoint
```

CRaC is hard!

Let’s try to fix it with arcane magic

```
# syntax=docker/dockerfile:1-labs
FROM bellsoft/liberica-runtime-container:jdk-musl as builder

COPY . /app
RUN cd /app && ./gradlew build -xtest

FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer

COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
WORKDIR /app
RUN --security=insecure java -Dspring.context.checkpoint=onRefresh \
  -XX:CRaCCheckpointTo=/checkpoint -jar /app/app.jar

FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner

ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
```

CRaC is hard!

Let’s try to fix it with arcane magic

```
# syntax=docker/dockerfile:1-labs
FROM bellsoft/liberica-runtime-container:jdk-musl as builder

COPY . /app
RUN cd /app && ./gradlew build -xtest

FROM bellsoft/liberica-runtime-container:jre-crac-slim as optimizer

COPY --from=builder /app/build/libs/spring-petclinic-3.3.0.jar /app/app.jar
WORKDIR /app
RUN --security=insecure java -Dspring.context.checkpoint=onRefresh \
  -XX:CRaCCheckpointTo=/checkpoint -jar /app/app.jar || true

FROM bellsoft/liberica-runtime-container:jre-crac-slim as runner

ENTRYPOINT java -XX:CRaCRestoreFrom=/checkpoint
```

And this is not all!

Did you hear about `buildx` ?

```
docker buildx create --buildkitd-flags \
    '--allow-insecure-entitlement security.insecure' \
    --name insecure-builder
```

And this is not all!

Did you hear about `buildx` ?

```
docker buildx create --buildkitd-flags \
    '--allow-insecure-entitlement security.insecure' \
    --name insecure-builder
```

And this is not all!

Did you hear about `buildx` ?

```
1  docker buildx use insecure-builder
```

And this is not all!

Did you hear about `buildx` ?

```
docker buildx build \
    --allow security.insecure \
    -f Dockerfile.crac \
    -t pet-crac --output type=docker .
```

And this is not all!

Did you hear about `buildx` ?

```
docker buildx build \
    --allow security.insecure \
    -f Dockerfile.crac \
    -t pet-crac --output type=docker .
```

And this is not all!

Did you hear about `buildx` ?

```
docker buildx build \
    --allow security.insecure \
    -f Dockerfile.crac \
    -t pet-crac --output type=docker .
```

And this is not all!

Did you hear about `buildx` ?

```
1  docker run --rm -it --privileged pet-crac
```

And this is not all!

Did you hear about `buildx` ?

```
Restarting Spring-managed lifecycle beans after JVM restore
HikariPool-1 - Thread starvation or clock leap detected (housekeeper delta=1d19h37m42s102ms798µs806ns)
Tomcat started on port 8080 (http) with context path '/'
Restored PetClinicApplication in 0.186 seconds (process running for 0.19)
```

And this is not all!

Did you hear about `buildx` ?

```
Restarting Spring-managed lifecycle beans after JVM restore
HikariPool-1 - Thread starvation or clock leap detected (housekeeper delta=1d19h37m42s102ms798µs806ns)
Tomcat started on port 8080 (http) with context path '/'
Restored PetClinicApplication in 0.186 seconds (process running for 0.19)
```

Is it a good solution?

It depends

Is it a good solution?

It depends

- If "It Works" is enough for you

Is it a good solution?

It depends

- If "It Works" is enough for you -> YES
- If you need more predictable and maintanable thing

Is it a good solution?

It depends

- If "It Works" is enough for you -> YES
- If you need more predictable and maintanable thing -> NO

How to make it better?

How to make it better?

1. Build JAR (in docker or not)

How to make it better?

1. Build JAR (in docker or not)
2. Create new image that will run the JAR with CRaC arguments in `ENTRYPOINT`

How to make it better?

1. Build JAR (in docker or not)
2. Create new image that will run the JAR with CRaC arguments in `ENTRYPOINT`
3. Run the container with capabilities:

How to make it better?

1. Build JAR (in docker or not)
2. Create new image that will run the JAR with CRaC arguments in `ENTRYPOINT`
3. Run the container with capabilities:
    1. CAP_SYS_PTRACE

How to make it better?

1. Build JAR (in docker or not)
2. Create new image that will run the JAR with CRaC arguments in `ENTRYPOINT`
3. Run the container with capabilities:
    1. CAP_SYS_PTRACE
    2. CAP_CHECKPOINT_RESTORE

How to make it better?

1. Build JAR (in docker or not)
2. Create new image that will run the JAR with CRaC arguments in `ENTRYPOINT`
3. Run the container with capabilities:
    1. CAP_SYS_PTRACE
    2. CAP_CHECKPOINT_RESTORE
4. Container will run and stop

How to make it better?

1. Build JAR (in docker or not)
2. Create new image that will run the JAR with CRaC arguments in `ENTRYPOINT`
3. Run the container with capabilities:
    1. CAP_SYS_PTRACE
    2. CAP_CHECKPOINT_RESTORE
4. Container will run and stop
5. Commit the container like

```
docker commit container-id new-tag
```

Pros and Cons

Pros:

1. Does not require arcane magic
2. Works more predictably
3. Does not depend on unstable features
4. Does not require privileged containers in

Cons:

1. Organization-specific
2. Requires more steps

And now we have all flavours of ultimate docker images!

Quick summary?

1. Use layers for faster deployment
2. Use CDS for faster startup without many compromises
3. Use CRaC for a *lightning-fast* startup (with compromises)
4. Use native image if you're prepared to sacrifice some build time

Thank you!

Find me @

- https://asm0dey.site
- @asm0dey.site
- @asm0dey@fosstodon.org
- asm0di0
- me@asm0dey.site
- asm0dey

Pasha Finkelshteyn @asm0dey.site @asm0di0

END