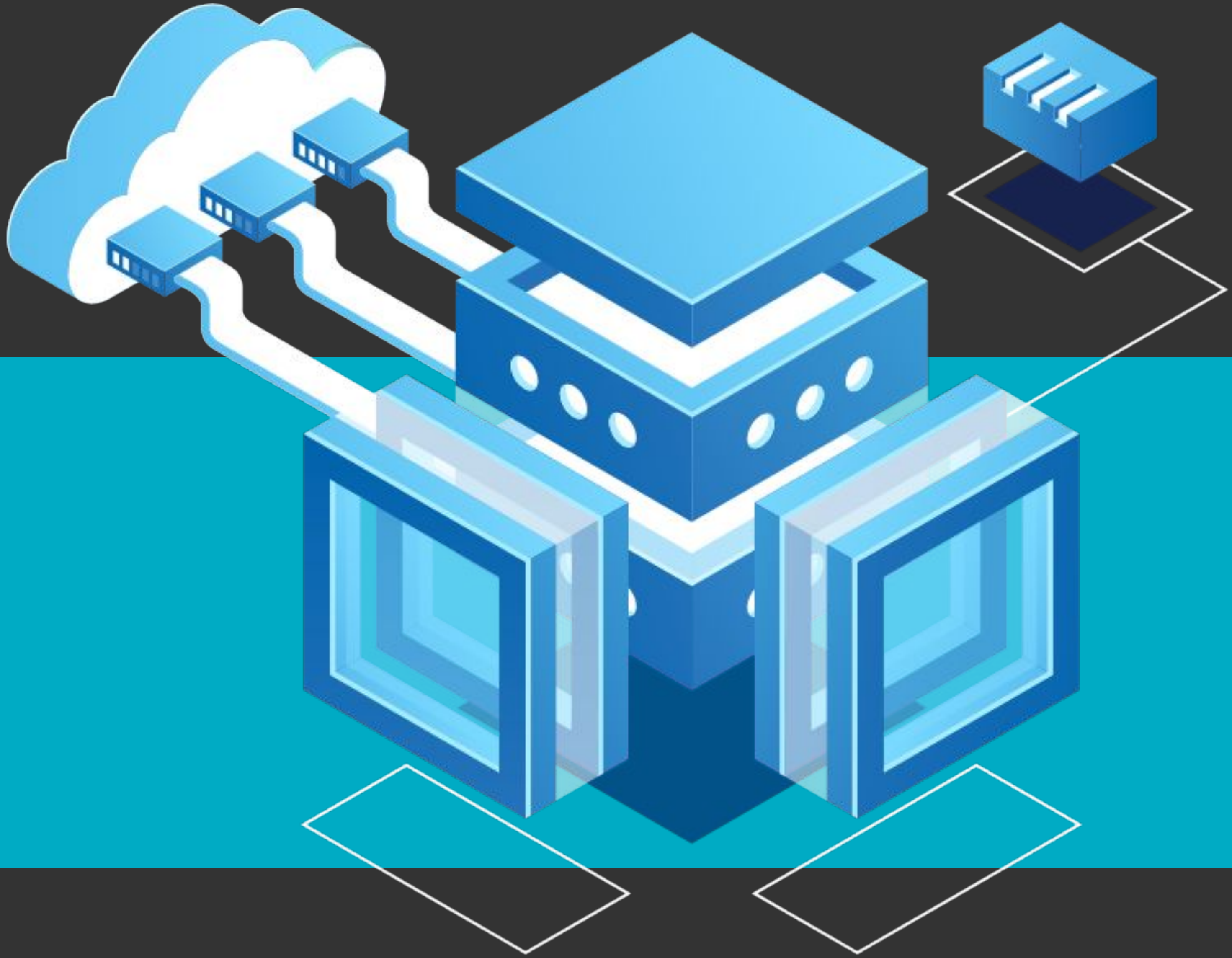
Chasing the Grail

Dmitry Chuyko

2021



Who we are

@dchuyko

Dmitry Chuyko

BELLSOFT

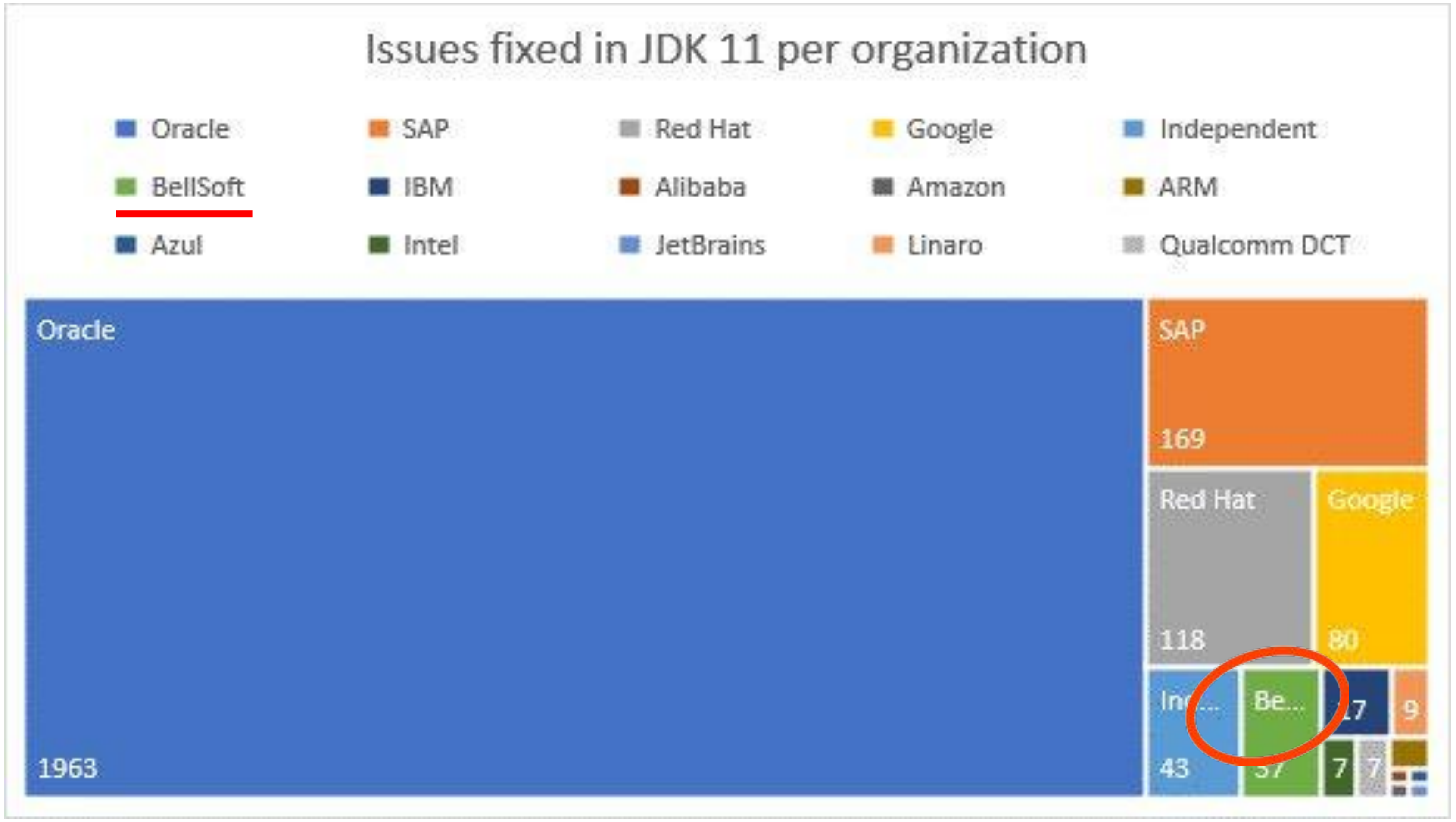
Liberica www.bell-sw.com
supported OpenJDK binaries

ex-employers:

ORACLE

OpenJDK Contributions

JDK 11



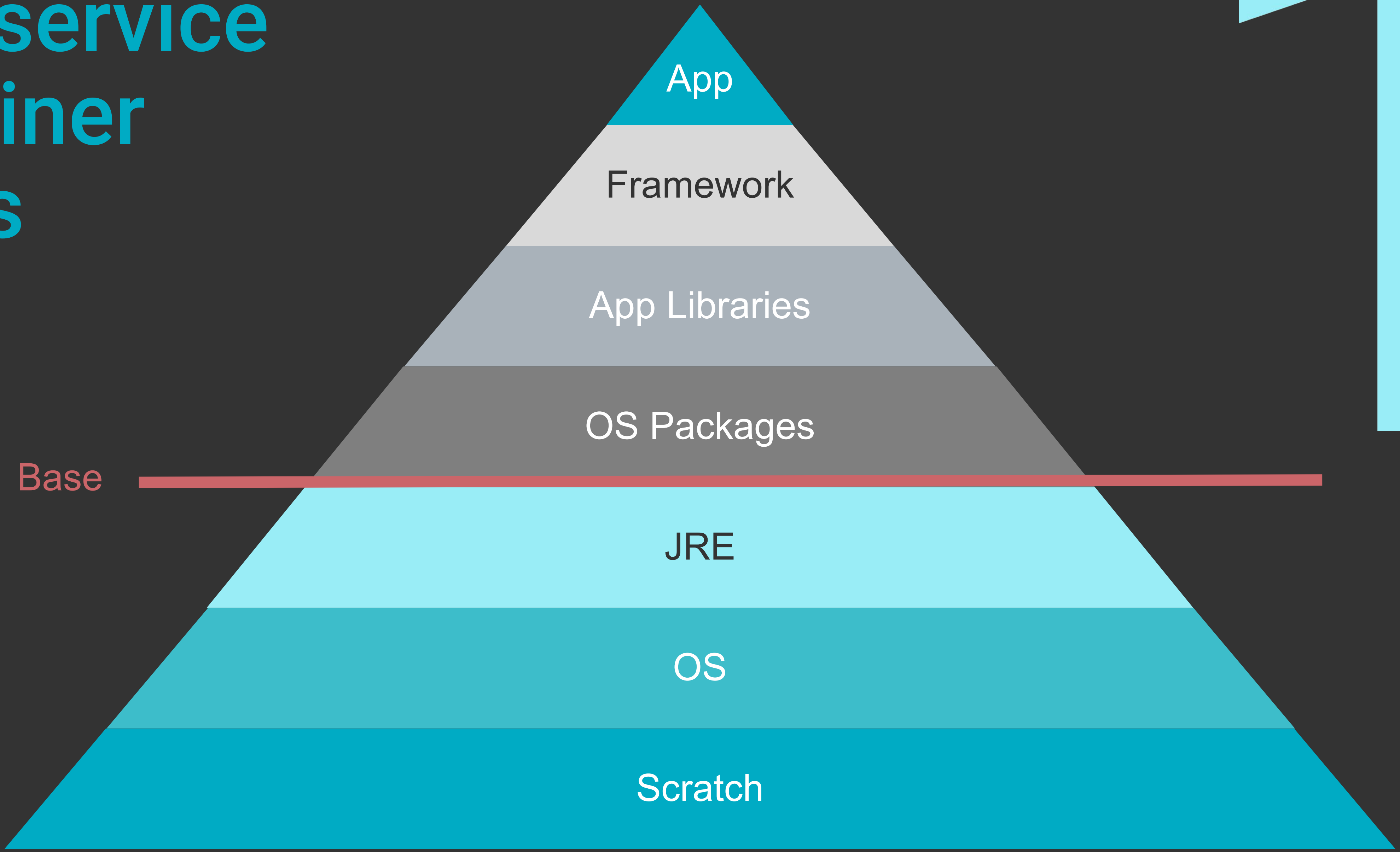
BELLSOFT

Deployment

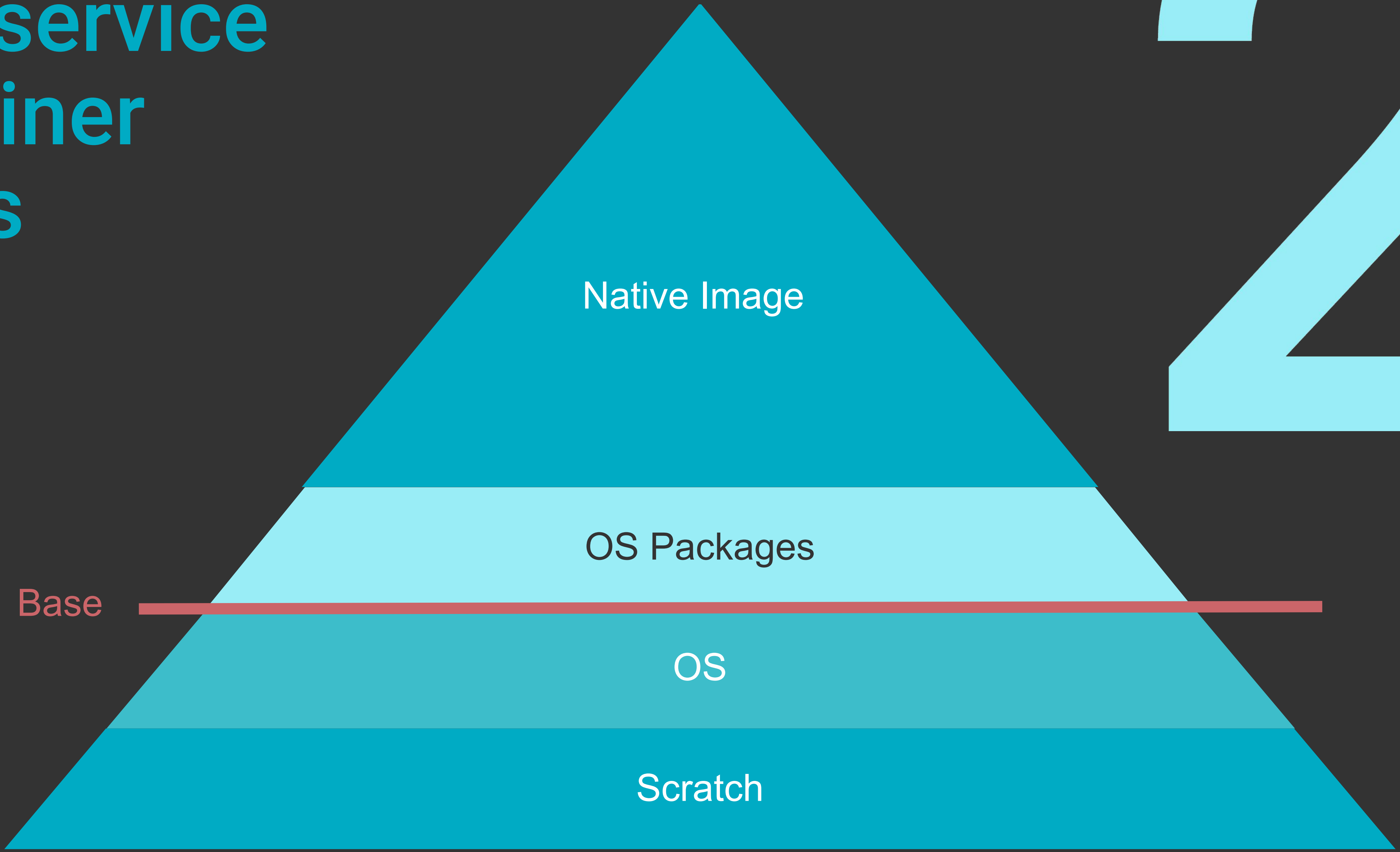
...package an application with all of
its dependencies into
a standardized unit for
software development.

— Docker

Microservice Container Layers



Microservice Container Layers



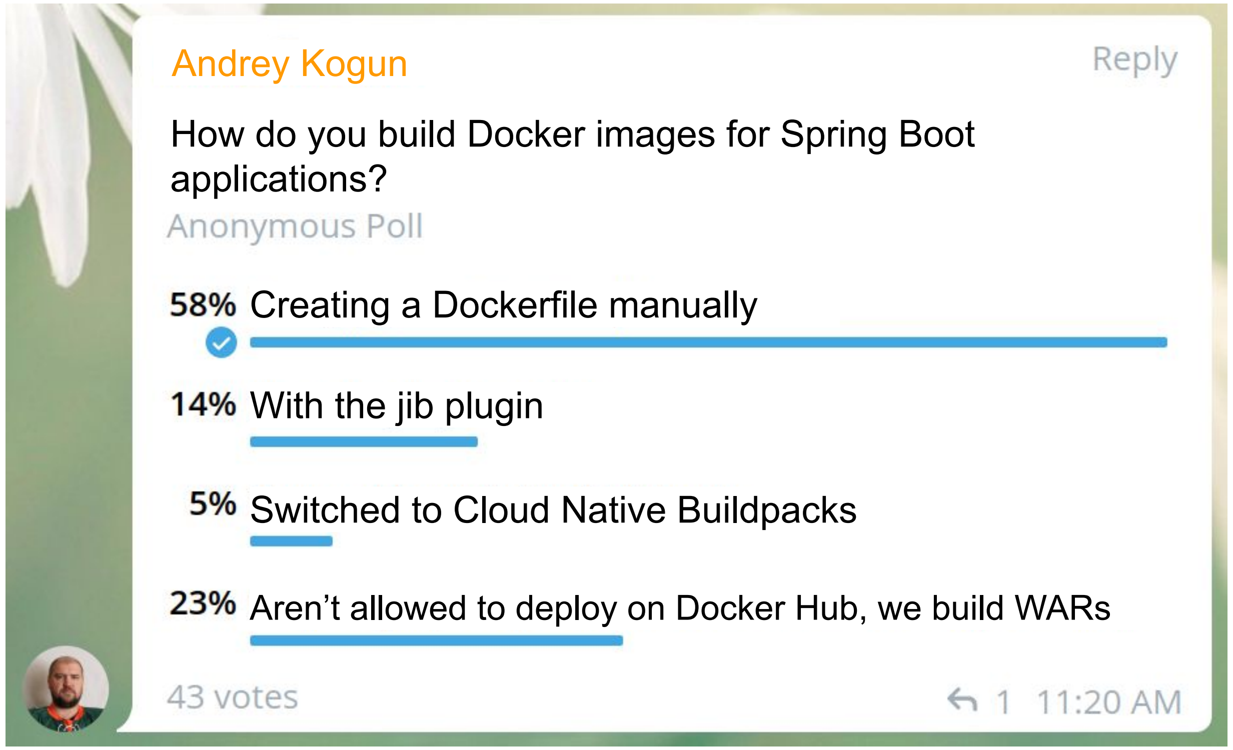
Sample Microservice

- Spring Initializr
- Spring Boot 2.4.4 (release)
- Spring Data JPA
- Lombok
- Spring Web
- H2
- Java 11

- Thin jar 2.7 kB
- Fat jar 37 MB

Developer Voice

- Aleksey Nesterov. Spring: Your next Java microframework
- Vladimir Plizga. Spring Boot "fat" JAR: Thin parts of a thick artifact



Andrey Kogun Reply

How do you build Docker images for Spring Boot applications?

Anonymous Poll

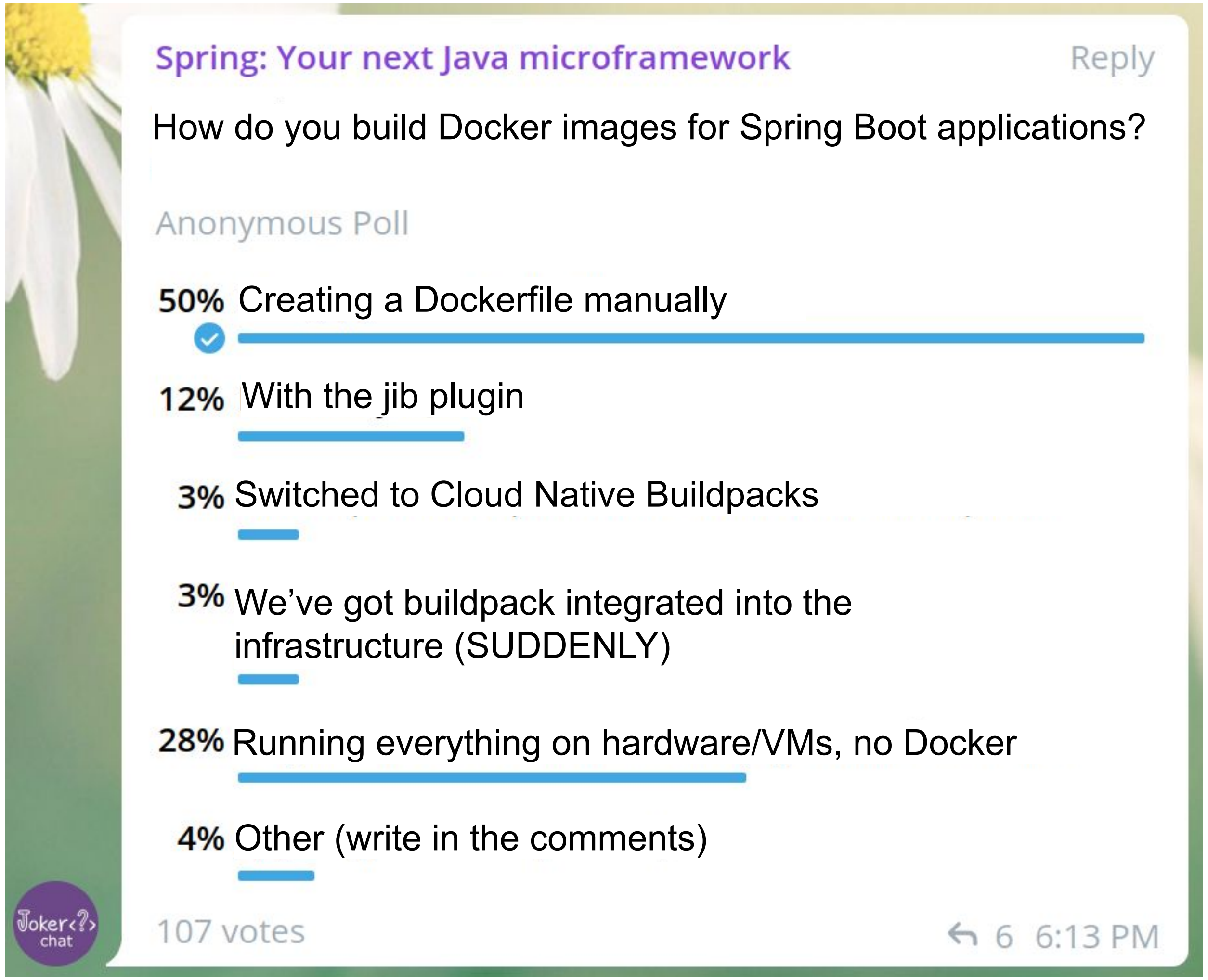
58% Creating a Dockerfile manually ✓

14% With the jib plugin

5% Switched to Cloud Native Buildpacks

23% Aren't allowed to deploy on Docker Hub, we build WARs

43 votes 1 11:20 AM



Spring: Your next Java microframework Reply

How do you build Docker images for Spring Boot applications?

Anonymous Poll

50% Creating a Dockerfile manually ✓

12% With the jib plugin

3% Switched to Cloud Native Buildpacks

3% We've got buildpack integrated into the infrastructure (SUDDENLY)

28% Running everything on hardware/VMs, no Docker

4% Other (write in the comments)

107 votes 6 6:13 PM

Base/Parent Images

A **base image** has **FROM scratch** in its Dockerfile.

A **parent image** is the one that your image is based on. It refers to the contents of the FROM directive in the Dockerfile. Each subsequent declaration in the Dockerfile modifies this parent image. Most Dockerfiles start from a parent image rather than a base image. **However, the terms are sometimes used interchangeably**.

OS + JDK images

- **Based on OS images**
- **JDK package installation**
  - Package manager
  - Package
  - Same vendor
- **JDK binary installation**
  - Requirements
  - Compatibility
- **Ask your provider about testing**

Pull Time (100 Mbps)

```
$ time docker pull openjdk
...
real    0m27.990s
user    0m0.095s
sys 0m0.096s
```

28 s

Uncompressed Size (disk)

```
$ docker history openjdk
IMAGE               CREATED             CREATED BY                                      SIZE
95b80f783bd2        12 days ago         /bin/sh -c #(nop)  CMD ["jshell"]               0B
<missing>           12 days ago         /bin/sh -c set -eux;   objdump="$(command -v…   336MB
<missing>           12 days ago         /bin/sh -c #(nop)  ENV JAVA_VERSION=15.0.1      0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV PATH=/usr/java/openjd…   0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV JAVA_HOME=/usr/java/o…   0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV LANG=C.UTF-8             0B
<missing>           12 days ago         /bin/sh -c set -eux;  microdnf install   gzi…   40.1MB
<missing>           12 days ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           12 days ago         /bin/sh -c #(nop) ADD file:ca74b6a4572ba9ecd…   148MB
<missing>           8 weeks ago         /bin/sh -c #(nop)  LABEL org.opencontainers.…   0B

$ docker images | head -n 1; docker images | grep openjdk
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
openjdk             latest              95b80f783bd2        12 days ago         524MB
```

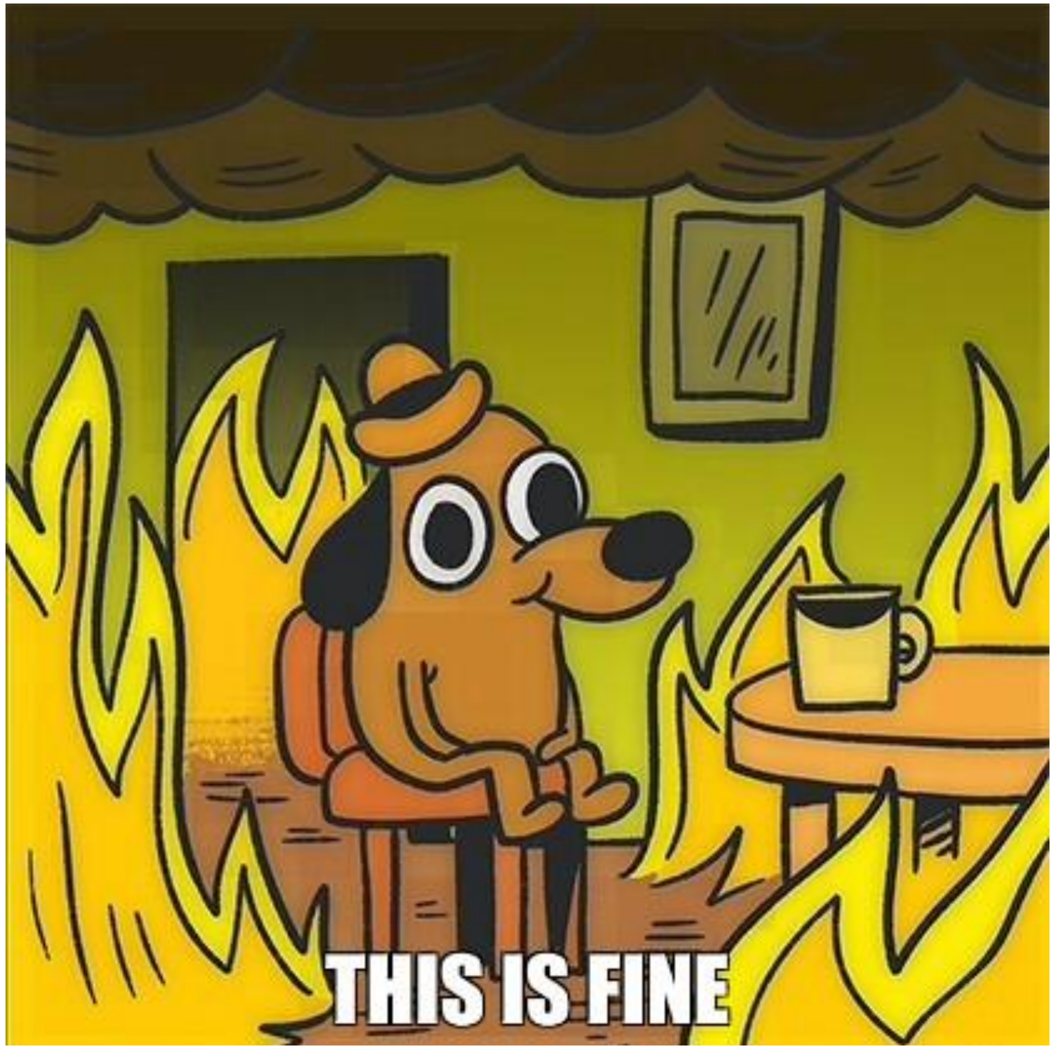
524 MB

Deployment Costs. Cloud

x 0.251 GB
x 1k deploys

= 0.25 TB

- <u>Tens of seconds</u> for a <u>single</u> pull
- Shared HW
- Shared I/O limits
- Keep old versions
- On-premise / private cloud?
- Elastic fleet
- 10 Mbps



Smaller Containers Can Help

Images are transferred over the network across domains, so less traffic is cheaper. At the same time, every deployment will go faster.

The paid registry needs to contain less volume of data, and less data is transferred out.

Everyone wants out of the box service

“...out of the box services that assist you when building Microservices, monoliths or any application in a linux container (Docker/Rocket) environment and is built on top of Kubernetes.”

— *Fabric8*

Everyone wants out of the box service

“transform your application source code into images that can run on any cloud.”

— *Cloud Native Buildpacks*

Fabric8? Centos?

```
$ docker run -it fabric8/java-centos-openjdk8-jre java -version

openjdk version "1.8.0_262"
OpenJDK Runtime Environment (build 1.8.0_262-b10)
OpenJDK 64-Bit Server VM (build 25.262-b10, mixed mode)

$ docker run -it bellsoft/liberica-openjre-centos:8-x86_64 java -version
openjdk version "1.8.0_282"
OpenJDK Runtime Environment (build 1.8.0_282-b08)
OpenJDK 64-Bit Server VM (build 25.282-b08, mixed mode)

REPOSITORY                          TAG          SIZE
bellsoft/liberica-openjre-centos    8-x86_64     329MB
fabric8/java-centos-openjdk8-jre    latest       424MB
```

```
$ docker run -it fabric8/java-alpine-openjdk8-jre java -XX:StartFlightRecording -version

Unrecognized VM option 'StartFlightRecording'
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

- JDK-8223147: JFR Backport
  - Fix Version/s: openjdk8u262

Cloud Native Buildpacks

```
BP_NATIVE_IMAGE=false

$ mvn spring-boot:build-image
...
[INFO]     [creator]     ===> EXPORTING
[INFO]     [creator]     Adding layer 'paketo-buildpacks/ca-certificates:helper'
[INFO]     [creator]     Adding layer 'paketo-buildpacks/bellsoft-liberica:helper'
[INFO]     [creator]     Adding layer
'paketo-buildpacks/bellsoft-liberica:java-security-properties'
[INFO]     [creator]     Adding layer 'paketo-buildpacks/bellsoft-liberica:jre'
[INFO]     [creator]     Adding layer 'paketo-buildpacks/bellsoft-liberica:jvmkill'
[INFO]     [creator]     Adding layer 'paketo-buildpacks/executable-jar:classpath'
[INFO]     [creator]     Adding layer 'paketo-buildpacks/spring-boot:helper'
[INFO]     [creator]     Adding layer 'paketo-buildpacks/spring-boot:spring-cloud-bindings'
[INFO]     [creator]     Adding layer 'paketo-buildpacks/spring-boot:web-application-type'
[INFO]     [creator]     Adding 5/5 app layer(s)
[INFO]     [creator]     Adding layer 'launcher'
[INFO]     [creator]     Adding layer 'config'
[INFO]     [creator]     Adding layer 'process-types'
...
```

Uncompressed Size (disk)

```
...
[INFO] Successfully built image 'docker.io/library/demo244:0.0.1-SNAPSHOT'

$ docker images --format "table {{.Tag}}\t{{.Size}}" demo244

TAG                    SIZE
0.0.1-SNAPSHOT         281MB
```

281 MB

OS Layer	Wire	Disk	libc	pkg man	shell
Ubuntu	27 MB	73 MB	glibc	apt	bash
Debian	48 MB	114 MB	glibc	apt	bash
Debian Slim	26 MB	69 MB	glibc	apt	bash
CenOS	71 MB	215 MB	glibc	yum	bash
RHEL Atomic Base	31 MB	78 MB	glibc	microdnf	bash
GCR Distroless base	7.6 MB	17 MB	glibc	—	—
Alpine	**2.7 MB**	**5.6 MB**	musl	apk	ash
GCR Distroless static	0.6 MB	1.8 MB	—	—	—

Alpine Linux Port

“

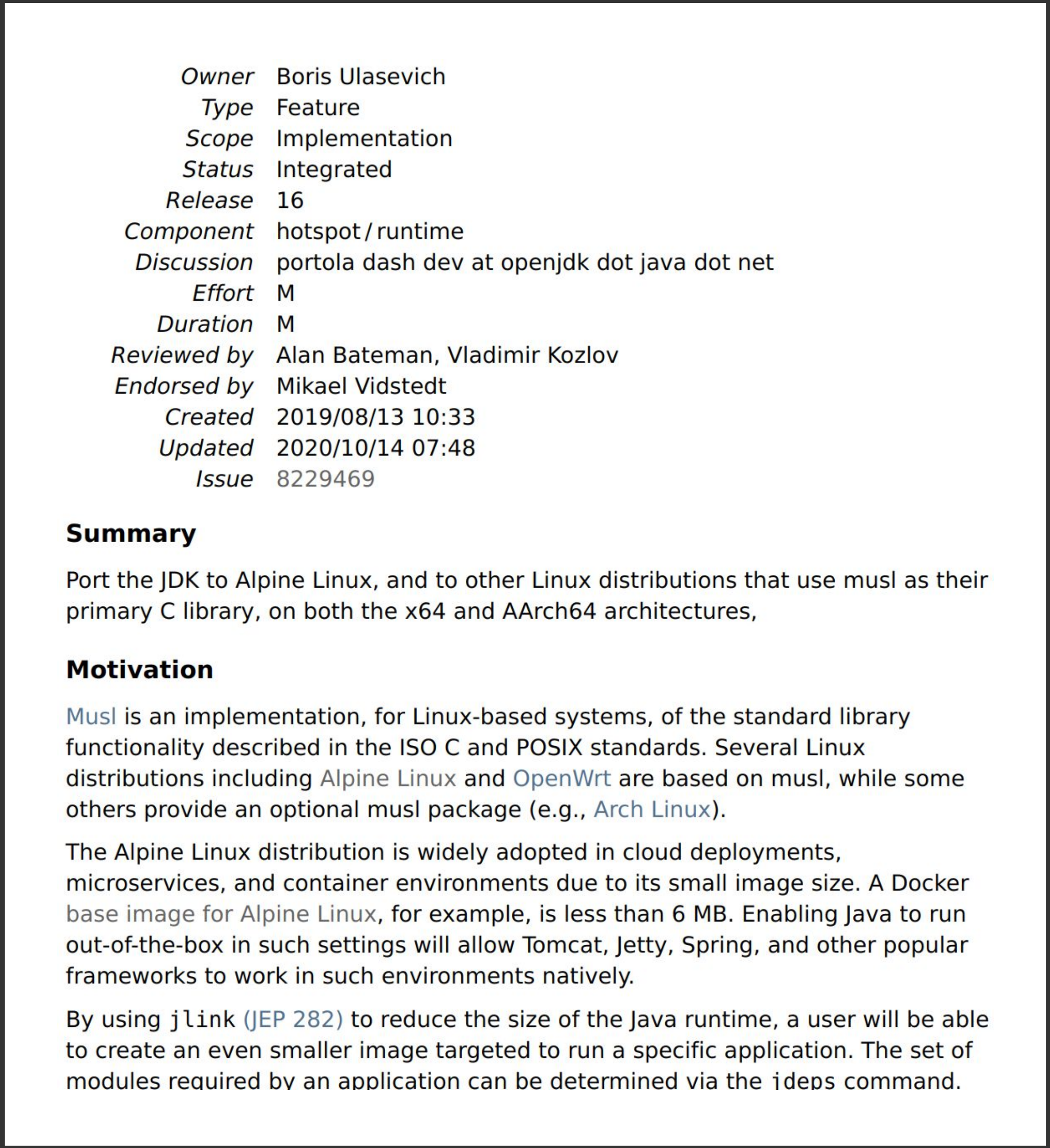
... is a security-oriented, lightweight Linux distribution based on musl libc and busybox.

— Alpine

”

JDK 16

- JEP 386: Alpine Linux Port
- openjdk.java.net/jeps/386
- openjdk.java.net/projects/portola
  - Port of the JDK to the Alpine Linux distribution,
    and in particular the musl C library



Owner	Boris Ulasevich
Type	Feature
Scope	Implementation
Status	Integrated
Release	16
Component	hotspot / runtime
Discussion	portola dash dev at openjdk dot java dot net
Effort	M
Duration	M
Reviewed by	Alan Bateman, Vladimir Kozlov
Endorsed by	Mikael Vidstedt
Created	2019/08/13 10:33
Updated	2020/10/14 07:48
Issue	8229469

Summary

Port the JDK to Alpine Linux, and to other Linux distributions that use musl as their primary C library, on both the x64 and AArch64 architectures,

Motivation

Musl is an implementation, for Linux-based systems, of the standard library functionality described in the ISO C and POSIX standards. Several Linux distributions including Alpine Linux and OpenWrt are based on musl, while some others provide an optional musl package (e.g., Arch Linux).

The Alpine Linux distribution is widely adopted in cloud deployments, microservices, and container environments due to its small image size. A Docker base image for Alpine Linux, for example, is less than 6 MB. Enabling Java to run out-of-the-box in such settings will allow Tomcat, Jetty, Spring, and other popular frameworks to work in such environments natively.

By using `jlink` (JEP 282) to reduce the size of the Java runtime, a user will be able to create an even smaller image targeted to run a specific application. The set of modules required by an application can be determined via the `jdeps` command.

Liberica JDK Images

OS + JDK 15 Image	Wire	Disk
bellsoft/liberica-openjdk-debian	126 MB	231 MB
bellsoft/liberica-openjdk-centos	183 MB	322 MB
bellsoft/liberica-openjdk-alpine	78 MB	132 MB
bellsoft/liberica-openjdk-alpine-musl	**76 MB**	**107 MB**

Pull Time

```
$ time docker pull bellsoft/liberica-openjdk-alpine-musl:latest
...
real    0m3.957s
user    0m0.026s
sys 0m0.061s
```

4 s

Fabric8? Alpine? JDK 8?

```
REPOSITORY                                 TAG   SIZE
bellsoft/liberica-openjdk-alpine-musl      8     152MB
fabric8/java-alpine-openjdk8-jdk           8     108MB


# zipinfo /usr/lib/jvm/jdk-8u282-bellsoft-x86_64/jre/lib/rt.jar

19839 files, 61949762 bytes uncompressed, 61949762 bytes compressed:  0.0%

# zipinfo /usr/lib/jvm/java-1.8-openjdk/jre/lib/rt.jar

19783 files, 70086222 bytes uncompressed, 31066529 bytes compressed:  55.7%
```

```
$ run -it -v $(pwd)/demo:/demo bellsoft/liberica-openjdk-alpine-musl:8 \
  java -jar /demo/spring-petclinic-2.4.2.jar
```

Avg. startup
6.03 s

```
$ run -it -v $(pwd)/demo:/demo fabric8/java-alpine-openjdk8-jdk \
  java -jar /demo/spring-petclinic-2.4.2.jar
```

Avg. startup
6.81 s

**Difference is
12.6%**

Fabric8? Profiling?

```
Download async-profiler, setup host

$ docker run --cap-add SYS_ADMIN -it -v $(pwd)/demo:/demo <...> ash

# apk add libstdc++

# java -jar ...

# ./profiler.sh -d 4 $(pidof java)
```

Fabric8? Profiling?

```
fabric8/java-alpine-openjdk8-jdk

         ns  percent  samples  top
 ----------  -------  -------  ---
12291873038   85.40%     1229  /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/serve    jvm.so
  150020578    1.04%       15  /lib/libz.so.1.2.11
   90100718    0.63%        9  /lib/ld-musl-x86_64.so.1
   90021783    0.63%        9  sun.net.www.ParseUtil.encodePath
   80099952    0.56%        8  tss_get

bellsoft/liberica-openjdk-alpine-musl:8

         ns  percent  samples  top
 ----------  -------  -------  ---
  979986354    6.03%       98  PhaseIdealLoop::is_dominator(Node*, Node*) [clone .part.0]
  740006769    4.55%       74  IndexSetIterator::advance_and_next()
  430016656    2.65%       43  PhaseChaitin::Split(unsigned int, ResourceArea*)
  360010587    2.22%       36  PhaseChaitin::build_ifg_physical(ResourceArea*)
  339999295    2.09%       34  SpinPause
```

Fabric8? Profiling?

```
fabric8/java-alpine-openjdk8-jdk

# objdump --syms /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/server/libjvm.so

SYMBOL TABLE:
no symbols


bellsoft/liberica-openjdk-alpine-musl:8

# objdump --syms /usr/lib/jvm/jdk-8u282-bellsoft-x86_64/jre/lib/amd64/server/libjvm.so \
  | wc -l

41695
```

Portola Expansion

- **JDK 11 LTS**
  - Not in mainline (yet)
  - Historical downports in Liberica 9+
- **JDK 8 LTS**
  - Liberica 8u on Dockerhub
- **AArch64**

Alpine?

```
$ docker run -it -v $(pwd)/lists:/lists alpine ash

# apk add openjdk8

# java -jar ....

# jcmd $(pidof java)

ash: jcmd: not found

# /usr/lib/jvm/java-1.8-openjdk/bin/jcmd $(pidof java) VM.uptime
```

Alpine?

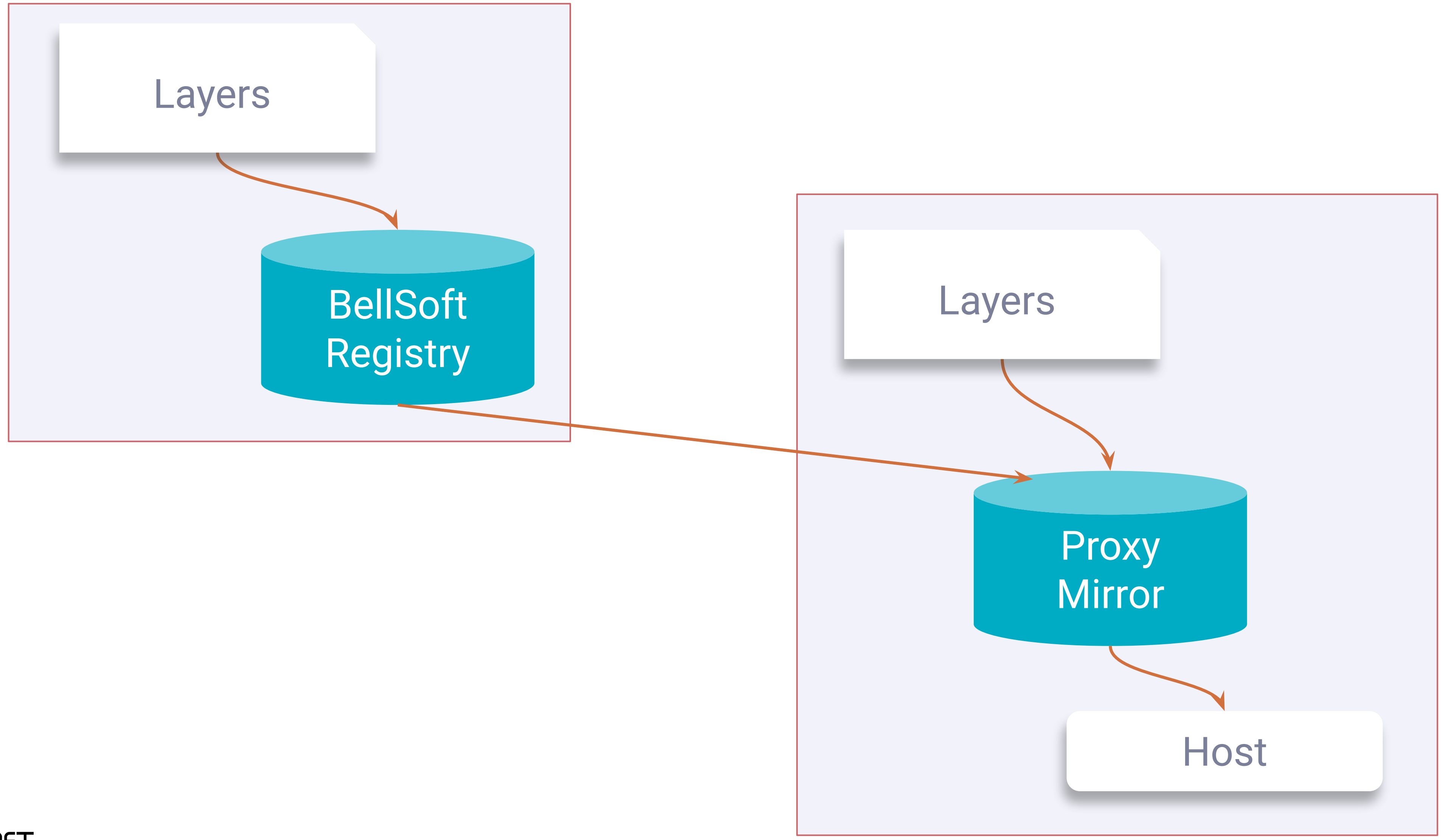
```
48:
2021-04-20 20:20:06
Full thread dump OpenJDK 64-Bit Server VM (25.275-b01 mixed mode):

"Service Thread" #9 daemon prio=9 os_prio=0 tid=0x00007fcb4675c800 nid=0x105 runn
[0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
.......

$ docker run -it -v $(pwd)/lists:/lists bellsoft/liberica-openjdk-alpine-musl:8 ash

# java -jar ...

# jcmd $(pidof java) VM.uptime

6:
26.695 s
```

Trusted Registry

“*On November 20, 2020, rate limits anonymous and free authenticated use of Docker Hub went into effect.*

— Dockerhub”

Deploy an Image. Networks



Early Access

Connect to Trusted Registry

```
Don't forget to configure credential helper

https://docs.docker.com/engine/reference/commandline/login/#credentials-store

$ docker login registry.bell-sw.com

Username: demo-user
Password:

Login Succeeded
```

Download and Run a Container

```
$ docker pull registry.bell-sw.com/demo/liberica-openjdk-alpine-musl:11.0.10-9
...

$ docker pull registry.bell-sw.com/demo/liberica-openjdk-alpine-musl:11.0.10-10-ea
...

$ docker images --format "table {{.Tag}}\t{{.Size}}" \
> registry.bell-sw.com/demo/liberica-openjdk-alpine-musl

TAG                 SIZE
11.0.10-10-ea       99.8MB
11.0.10-9           106MB

$ docker run --rm -it registry.bell-sw.com/demo/liberica-openjdk-alpine-musl:11.0.10-10-ea

openjdk version "11.0.10" 2021-01-19
OpenJDK Runtime Environment (build 11.0.10+10-ea)
OpenJDK 64-Bit Server VM (build 11.0.10+10-ea, mixed mode)
```

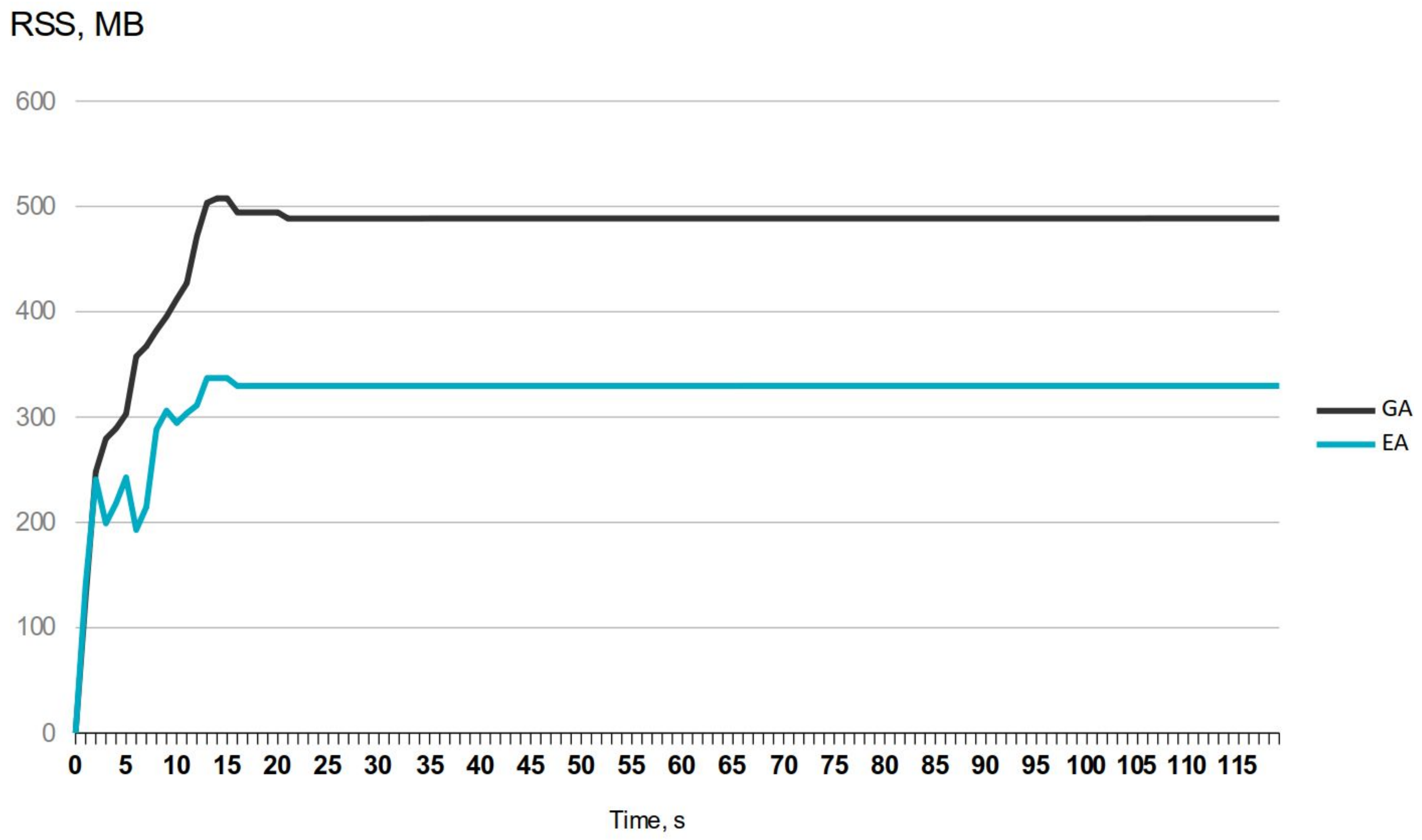
Experimental optimizations

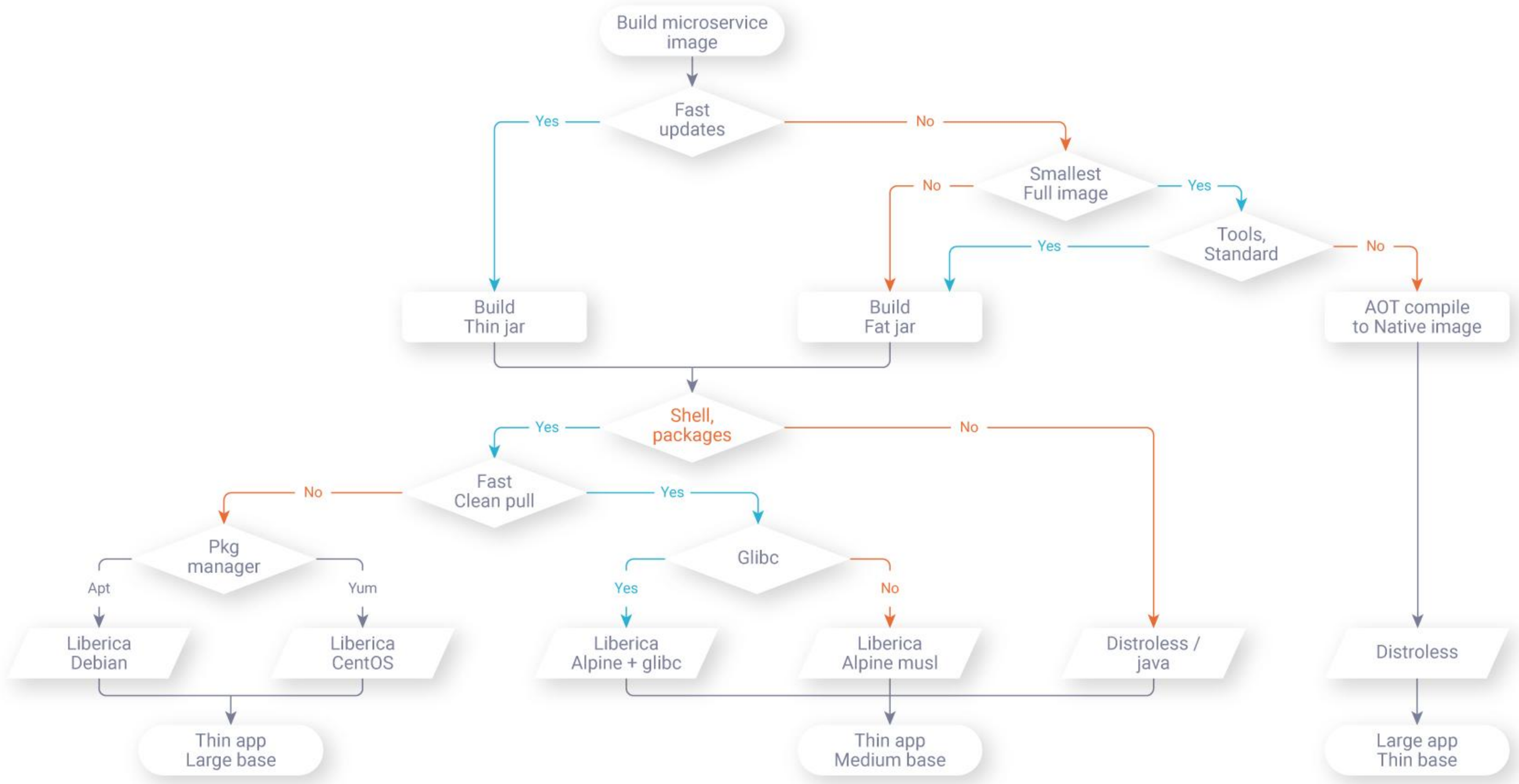
Experimental Optimizations

- Smaller size - 6%
- Faster GC (G1, safepoints,...) - Up to 11% better latency
- Decreased memory footprint

Footprint

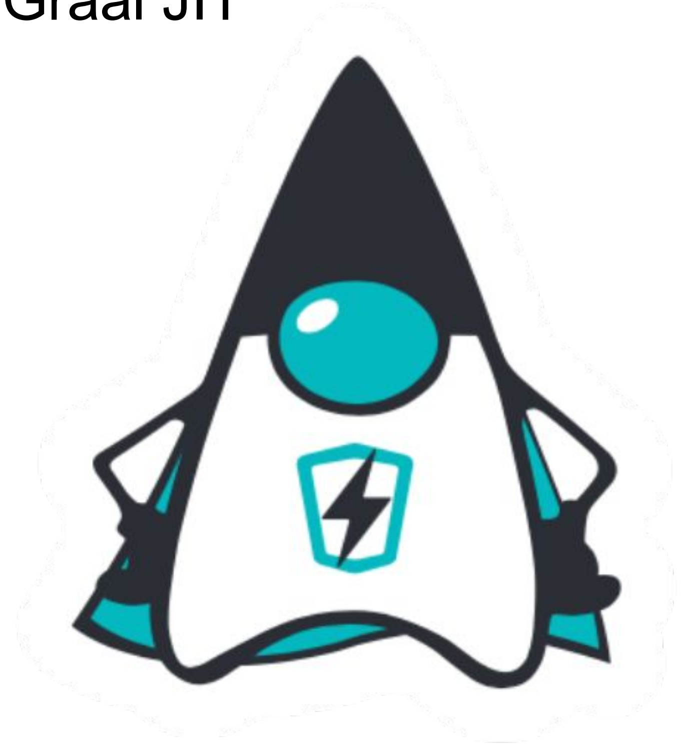
Petclinic. No load




Build microservice image
Fast updates
Yes
No
Smallest Full image
No
Yes
Tools, Standard
Yes
No
Build Thin jar
Build Fat jar
AOT compile to Native image
Shell, packages
Yes
No
Fast Clean pull
No
Yes
Pkg manager
Apt
Yum
Glibc
Yes
No
Liberica Debian
Liberica CentOS
Liberica Alpine + glibc
Liberica Alpine musl
Distroless / java
Distroless
Thin app Large base
Thin app Medium base
Large app Thin base

JDK 16

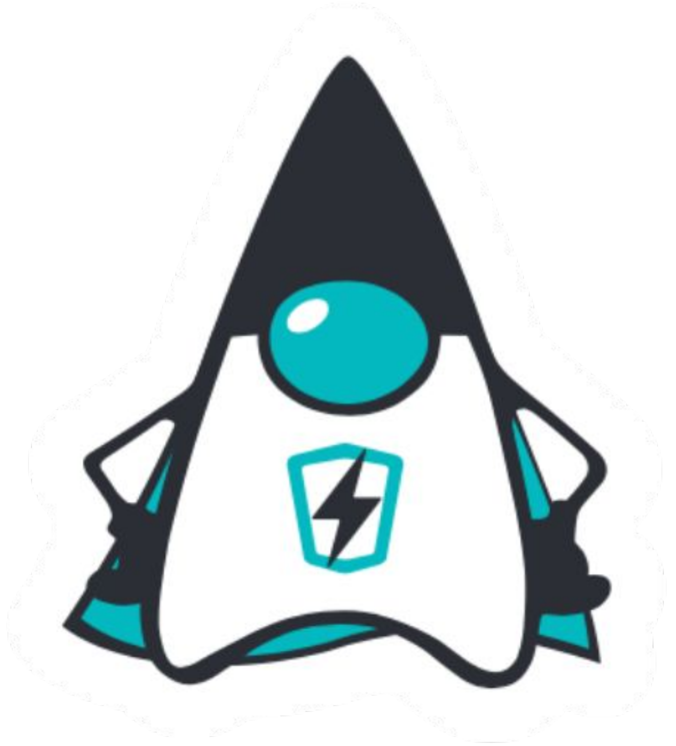
- JDK-8255616: Removal of experimental features AOT and Graal JIT
- Sources are still in the repo
- No new drops
- Features are still available in Liberica JDK 16 GA
- JVMCI is built and shipped, still experimental



BELLSOFT

GraalVM Licensing & Support

- Oracle GraalVM Enterprise Edition
  - For Non-production.
    Oracle Technology Network License Agreement
    for GraalVM Enterprise Edition.
  - For Production.
    Oracle Java SE Subscription
- GraalVM Community
  - GPLv2 + ”Classpath” Exception



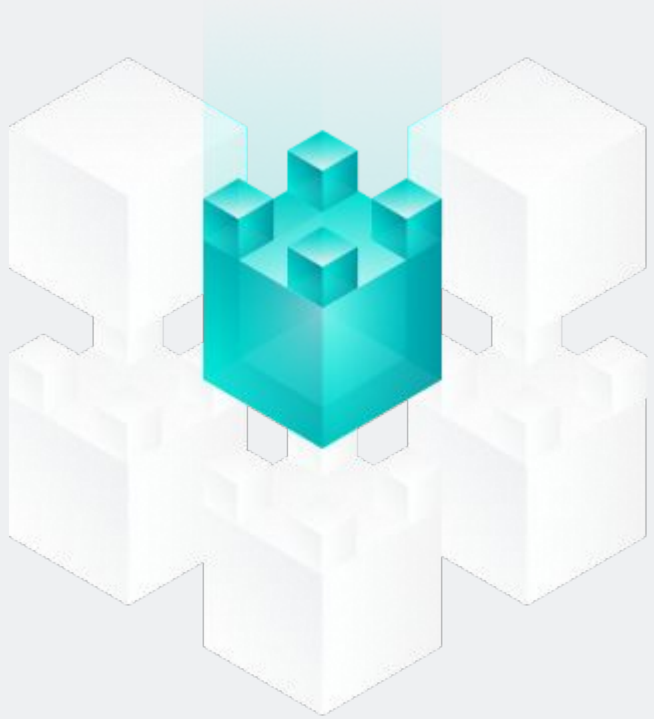
Liberica NIK

“*A utility that converts your JVM-based application into a fully AOT compiled native executable [...] based on the open source GraalVM Community Edition.*”

Liberica Native Image Kit

A wide and unique variety of supported platforms

Linux x86_64 (glibc), Linux Alpine x86_64 (musl), Linux AArch64 (glibc), Linux Alpine AArch64 (musl), Mac OS x86_64 are supported



BELLSOFT

Liberica Native Image Kit

Multiple languages interaction

Most binaries already work with Java, JavaScript, LLVM, Python, Ruby, R, and WebAssembly, as do GraalVM Native Image™ binaries

Liberica Native Image Kit

Running with most JDK versions

Update levels:

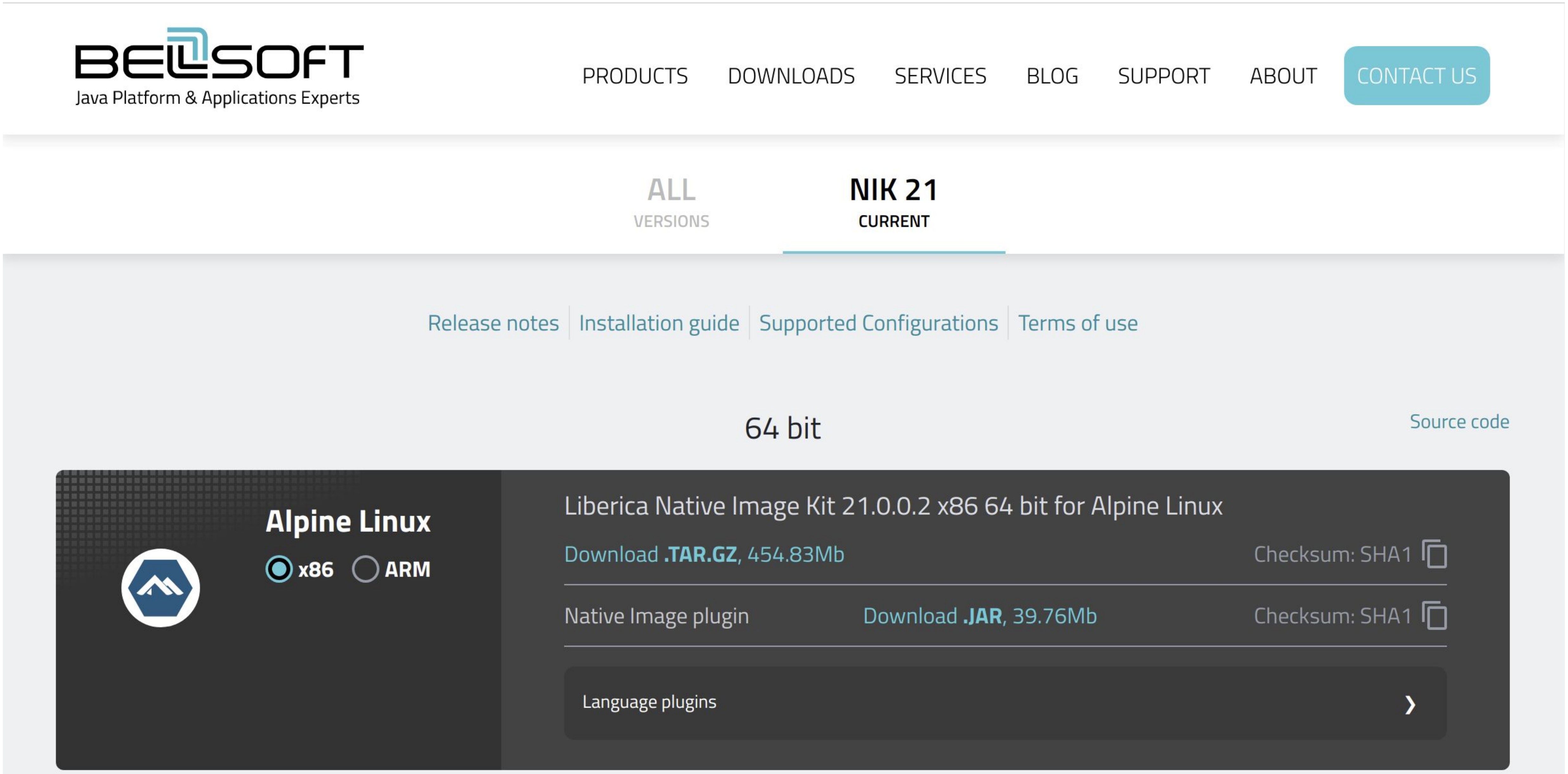
- JDK 11.0.10 and newer
- GraalVM CE 21.0 and newer

BELLSOFT

Graal Support for Alpine and musl

- Runnable tools
- Static musl linking
- Dynamic linking
- graal/pull/3141: linux-musl-amd64 support
- mx/pull/230: Linux-musl support
- fastr/pull/175: Added linux-musl support
- truffleruby/pull/2223: Added linux-musl support

https://bell-sw.com/pages/downloads/native-image-kit/



64 bit

Alpine Linux
[✓ x86] ARM

Liberica Native Image Kit 21.0.0.2 x86 64 bit for Alpine Linux

Download .TAR.GZ, 454.83Mb — Checksum: SHA1

Native Image plugin — Download .JAR, 39.76Mb — Checksum: SHA1

Sample Microservice

- spring-guides/gs-rest-service
- gs-rest-service/complete

- Thin jar 32 kB
- Fat jar 17 MB

Enable Native Image Support

```
pom.xml
...
  <pluginRepository>
     <id>spring-release</id>
     <name>Spring release</name>
     <url>https://repo.spring.io/release</url>
  </pluginRepository>
...
     <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <configuration>
           <classifier>exec</classifier>
        </configuration>
     </plugin>
...
```

Enable Native Image Support

```xml
<plugin>
   <groupId>org.springframework.experimental</groupId>
   <artifactId>spring-aot-maven-plugin</artifactId>
   <version>0.10.3</version>
   <executions>
      <execution>
         <id>test-generate</id>
         <goals>
            <goal>test-generate</goal>
         </goals>
      </execution>
      <execution>
         <id>generate</id>
         <goals>
            <goal>generate</goal>
         </goals>
      </execution>
   </executions>
</plugin>
```

Enable Native Image Support

```
...
  <repository>
     <id>spring-release</id>
     <name>Spring release</name>
     <url>https://repo.spring.io/release</url>
  </repository>
...
  <dependency>
    <groupId>org.springframework.experimental</groupId>
    <artifactId>spring-native</artifactId>
    <version>0.10.3</version>
  </dependency>
...
```

Enable Native Image Support

```
  <profile>
     <id>native-image</id>
     <build>
        <plugins>
           <plugin>
              <groupId>org.graalvm.nativeimage</groupId>
              <artifactId>native-image-maven-plugin</artifactId>
              <version>21.0.0</version>
              <configuration>
                 <!-- The native image build needs to know the entry point to your
application -->
                 <mainClass>com.example.restservice.RestServiceApplication</mainClass>
                 <buildArgs>
                    -Dspring.native.remove-yaml-support=true
                    -Dspring.spel.ignore=true
                 </buildArgs>
              </configuration>
...
```

Enable Native Image Support

```
...
              <executions>
                 <execution>
                    <goals>
                       <goal>native-image</goal>
                    </goals>
                    <phase>package</phase>
                 </execution>
              </executions>
           </plugin>
        </plugins>
     </build>
  </profile>
```

Flexible build

```
$ JAVA_HOME=$(pwd)/bellsoft-liberica-vm-core-openjdk11-21.2.0 mvn -Pnative install
```

Cloud Native Buildpacks

```
BP_NATIVE_IMAGE=true

$ mvn spring-boot:build-image

pom.xml:

<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
        <image>
            <builder>paketobuildpacks/builder:tiny</builder>
            <env>
                <BP_NATIVE_IMAGE>true</BP_NATIVE_IMAGE>
            </env>
            <pullPolicy>IF_NOT_PRESENT</pullPolicy>
        </image>
    </configuration>
</plugin>
```

Use Alpine for the Build

```
$ docker run -it -v $(pwd):/demo alpine

# apk add libstdc++
# apk add build-base
# apk add zlib-dev
# export JAVA_HOME=/demo/bellsoft-liberica-vm-openjdk11-21.0.0.2
# $JAVA_HOME/bin/gu install native-image

/demo/gs-rest-service/complete# /demo/apache-maven-3.6.3/bin/mvn -Pnative-image clean package

...

-rwxr-xr-x 1 root root  57M Mar 23 16:30 com.example.restservice.restserviceapplication
```

57 MB

Run the Image in Alpine

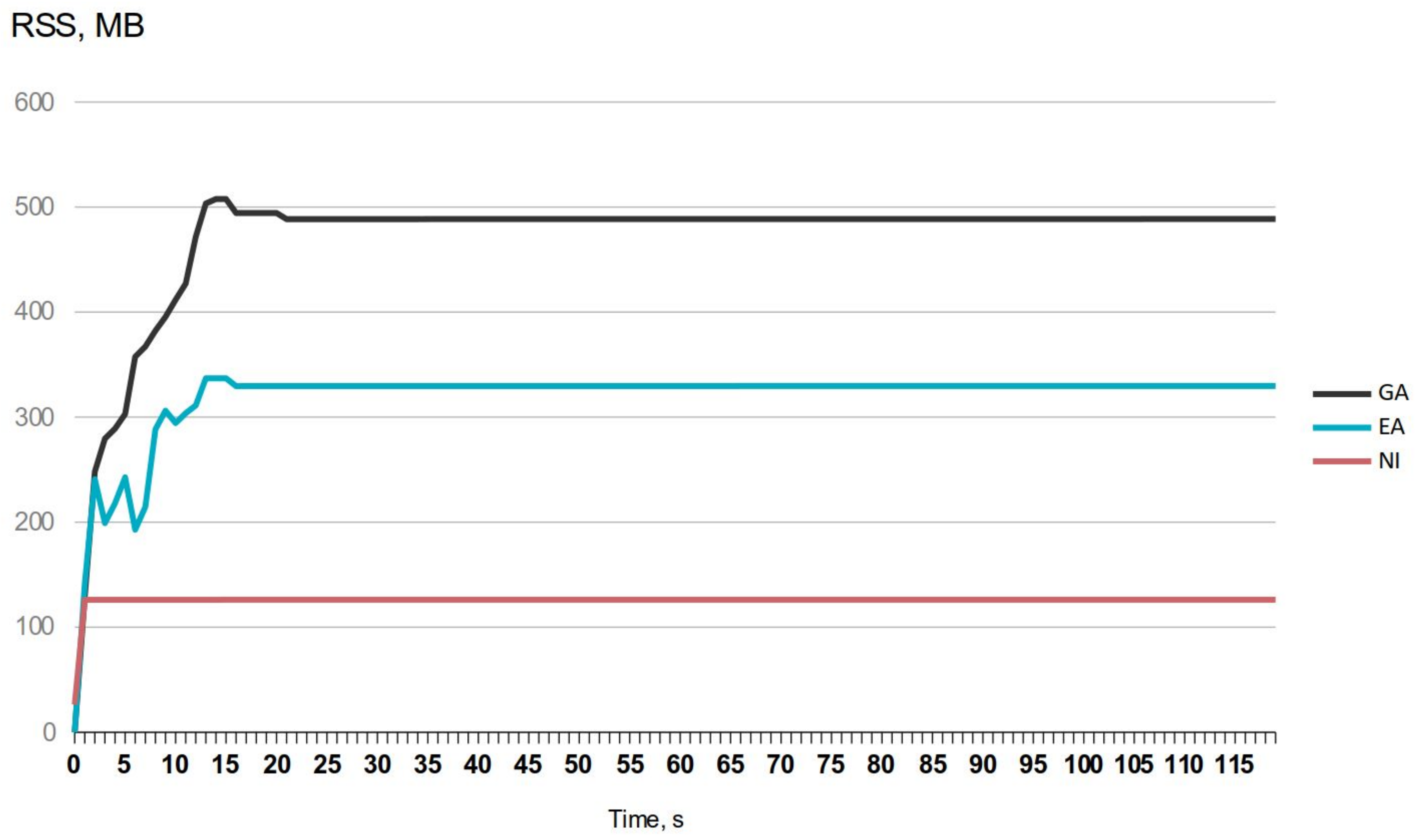
```
$ docker run -it -v $(pwd):/demo alpine

# apk add libstdc++

fetch https://dl-cdn.alpinelinux.org/alpine/v3.13/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.13/community/x86_64/APKINDEX.tar.gz
(1/2) Installing libgcc (10.2.1_pre1-r3)
(2/2) Installing libstdc++ (10.2.1_pre1-r3)
OK: 7 MiB in 16 packages


# /demo/gs-rest-service/complete/target/com.example.restservice.restserviceapplication

...
INFO 7 --- [           main] c.e.restservice.RestServiceApplication   : Started
RestServiceApplication in 0.055 seconds (JVM running for 0.057)
```

7+5.6 MB

1/20th s

Footprint

Petclinic. No load



Liberica Native Image Kit

See more at

bell-sw.com/pages/liberica-native-image-kit/

BELLSOFT

BELLSOFT

Thank you for your attention!

Web: www.bell-sw.com
Email: dmitry.chuyko@bell-sw.com
Twitter: @dchuyko