Alpine Musl Containers Now Upstream

Dmitry Chuyko

2021

Who we are

Dmitry Chuyko

BELLSOFT

Liberica www.bell-sw.com
supported OpenJDK binaries

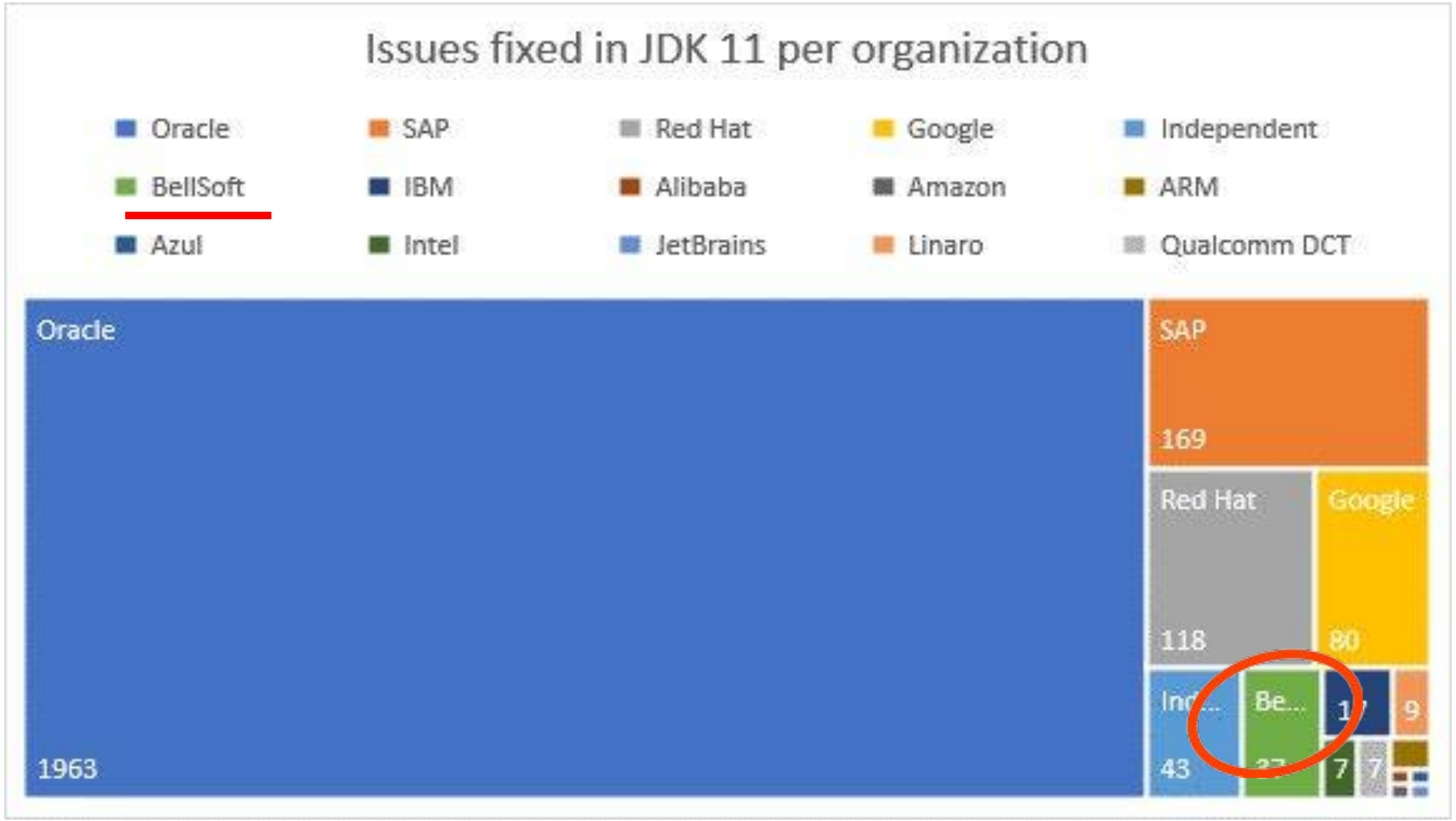
ex-employers:

ORACLE

@dchuyko

OpenJDK Contributions

JDK 11

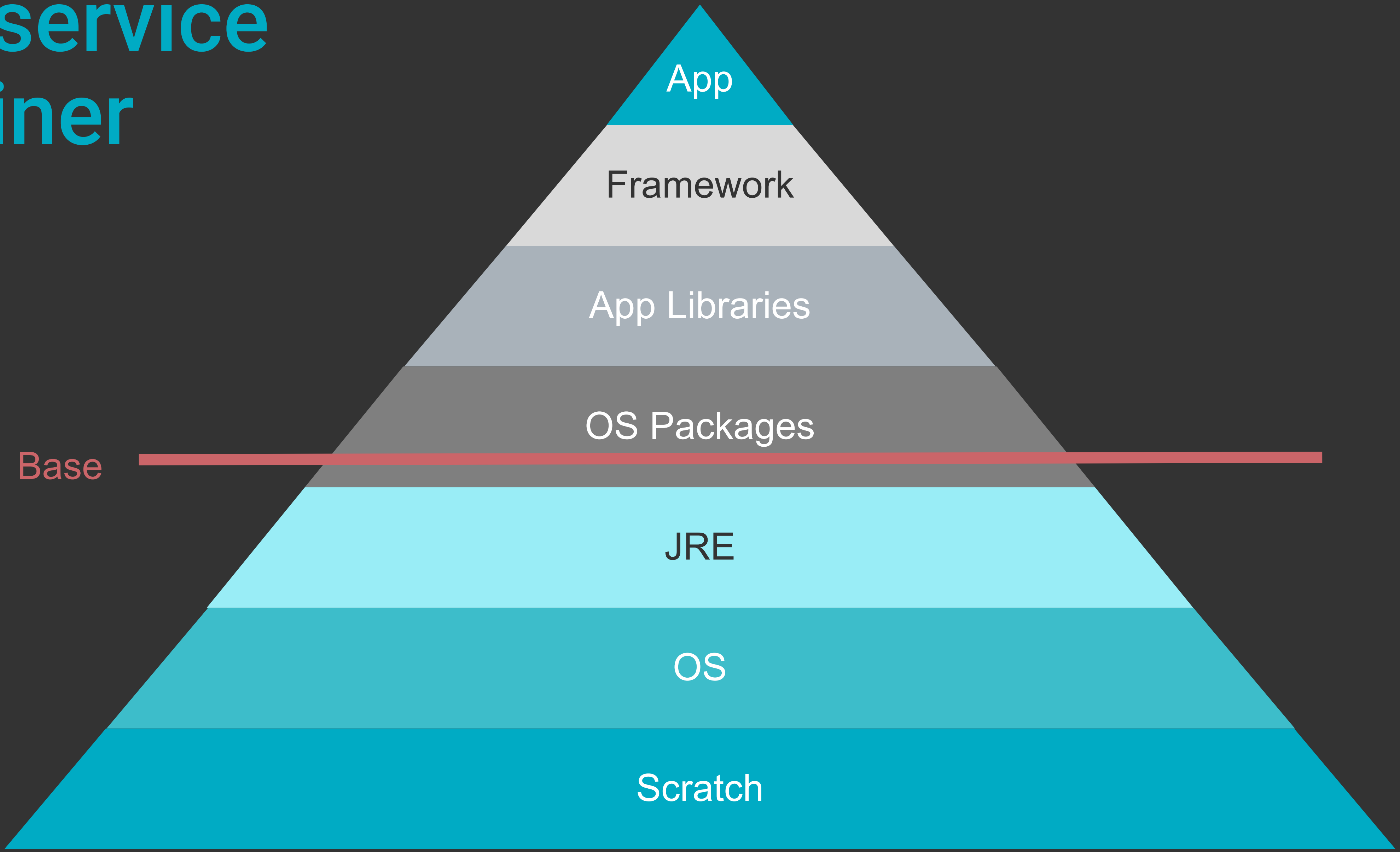


BELLSOFT

Deployment

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

Microservice container layers



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

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.

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

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

— Alpine

Alpine Linux. At a glance

- alpinelinux.org
- **Small**
  - Built around musl libc and busybox
  - Small packages
- **Simple**
  - OpenRC init system
  - apk package manager
- **Secure**
  - Position Independent Executables (PIE) binaries with stack smashing protection

Musl libc. At a glance

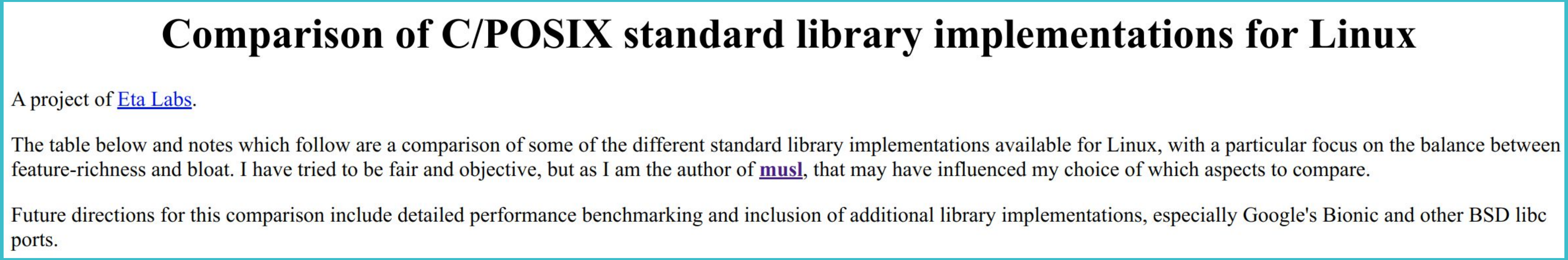
- musl.libc.org
- **Built on top of Linux syscall API (C bindings for the OS interfaces)**
- **Base language standard (ISO C)**
- **POSIX + widely-agreed extensions**
- **Lightweight (size), fast, simple, free (MIT)**
- **Strives to be correct in the sense of standards-conformance and safety**

Musl libc. Key Principles

- musl.libc.org/about.html
- **Simplicity**
  - Decoupling, minimize abstractions
  - Favors simple algorithms over more complex ones
  - Readable code
- **Resource efficiency**
  - Minimal size, low overhead, efficient static linking (Nx10kb)
  - Scalable (small stacks)
- **Attention to correctness**
  - Defensive coding, no race conditions
- **Ease of deployment (single binary)**
- **First-class support for UTF-8/multilingual text**

Libc implementations

- etalabs.net/compare_libcs.html
- **Note: outdated**



Comparison of C/POSIX standard library implementations for Linux

A project of Eta Labs.

The table below and notes which follow are a comparison of some of the different standard library implementations available for Linux, with a particular focus on the balance between feature-richness and bloat. I have tried to be fair and objective, but as I am the author of **musl**, that may have influenced my choice of which aspects to compare.

Future directions for this comparison include detailed performance benchmarking and inclusion of additional library implementations, especially Google's Bionic and other BSD libc ports.

Busybox. At a glance

- busybox.net
- **Many Unix utilities in a single executable file**
  - i.e. shell commands and the shell itself
- **Glibc, musl (Alpine), uLibc**
- **GPLv2**
- hub.docker.com/_/busybox

Busybox. Key Principles

- Swiss army knife, small
- Implementation of the standard Linux command line tools
- Smallest executable size
- Simplest and cleanest implementation
- Standards compliant
- Minimal run-time memory usage (heap and stack)
- Fast

Alpine Linux is perfect for containers

It is small. All necessary tools are available out of the box
or in packages.

What about JDK layer?

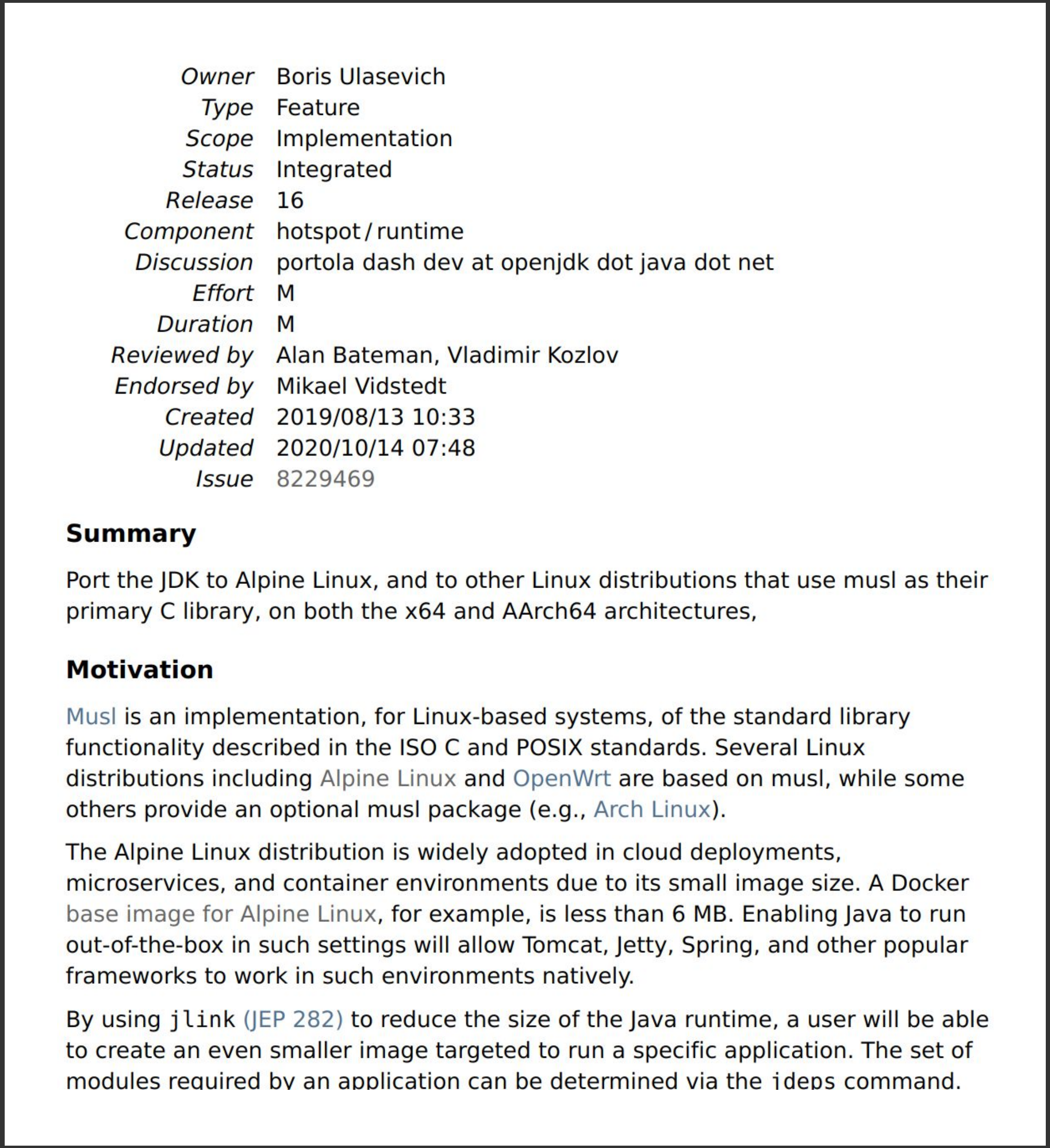
Alpine Linux Port

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.

— JEP 386

JDK 16

- **JEP 386: Alpine Linux Port**
- openjdk.java.net/jeps/386



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.

Project Portola

- openjdk.java.net/projects/portola
- **Port of the JDK to the Alpine Linux distribution, and in particular the musl C library**
- **Started by Mikael Vidstedt from Oracle in 2017**
- **Used for Alpine musl containers with JDK 9+**
- **Integrated into mainline in 2020 with JEP 386**
  - Delivered by BellSoft
  - JDK 16

Project Portola. Build

- **A new port**
  - Determine and distinguish C libraries
  - Conditional compilation
- **Native build**
- **Cross-toolchain for glibc environment**
- **Implement missing functions or make them compatible**
- **Testing environment**
- **Documentation**
  - https://github.com/openjdk/jdk/blob/master/doc/building.md#building-for-musl

JNI. Cross Build

```
$ x86_64-linux-musl-cross/bin/x86_64-linux-musl-gcc -std=c99 -I"$JAVA_HOME/include"
-I"$JAVA_HOME/include/linux" -shared -o libhelloworld.so -fPIC JNIHelloWorld.c

7.7K libhelloworld.so
```

```
$ docker run -it -v ~/jni:/jni bellsoft/liberica-openjdk-alpine-musl:15 java
-Djava.library.path=/jni -cp /jni JNIHelloWorld

Hello world!
```

```
$ docker run -it -v ~/jni:/jni bellsoft/liberica-openjdk-alpine:15 java
-Djava.library.path=/jni -cp /jni JNIHelloWorld

Hello world!
```

```
$ java -Djava.library.path=. JNIHelloWorld

Exception in thread "main" java.lang.UnsatisfiedLinkError: /home/tp/jni/libhelloworld.so:
/usr/lib/x86_64-linux-gnu/libc.so: invalid ELF header
```

Project Portola. Issues

- **LD_PRELOAD is not the same on different platforms**
  - Glibc resolves libs not like musl (or AIX libc)
  - jpackage and other launchers were fixed to still use proper JDK libs
- **Alpine used to have PaX/grsecurity in kernel by default**
  - Attempt to execute JIT code shut down the JVM
  - Added Memory protection check on startup
- **JDWP (Debug) sometimes had troubles with IPv4/IPv6 config**
  - Initialization was made more careful
- **Debugging (gdb)**
  - There's SIGSYNCCALL during JVM init
  - Debug with -XX:-MaxFDLimit

Project Portola. Issues

- **Running AWT in headless mode**
  - You may want to render images
  - Install freetype and fonts
- **Fontmanager**
  - For all real cases load awt lib before fontmanager
- **NMT**
  - Use latest Alpine (3.11+)
- **NUMA detection requires recent libnuma**
  - apk add numactl

Project Portola. Issues

- **lsof does not support ‘-p’ option on busybox**
  - Expect reduced output
- **Musl does not execute scripts that does not have a proper shebang**
  - Write proper # headers in *.sh
  - https://www.openwall.com/lists/musl/2020/02/13/4
- **Serviceability agent (private API) doesn’t work**

Shebang

```
$ docker run -it bellsoft/liberica-openjdk-alpine-musl:15 ash

-rwxr-xr-x    run.sh

echo "hello"

jshell> Runtime.getRuntime().exec("./run.sh")
|  Exception java.io.IOException: Cannot run program "./run.sh": error=8, Exec format error


-rwxr-xr-x    run.sh

#!/bin/sh
echo "hello"

jshell> Runtime.getRuntime().exec("./run.sh")
$1 ==> Process[pid=262, exitValue=0]
```

Variables

```
$ docker run -it -e "hibernate.format_sql=true" bellsoft/liberica-openjdk-alpine:15 ash

# set | grep hibernate

hibernate


$ docker run -it -e "hibernate.format_sql=true" bellsoft/liberica-openjdk-debian:15 bash

# set | grep hibernate

<empty>

$ docker run -it -e "hibernate_format_sql=true" bellsoft/liberica-openjdk-alpine-musl:15 ash

# set | grep hibernate

hibernate_format_sql='true'
```

SA

```
$ docker run -it bellsoft/liberica-openjdk-alpine:8 jstack -h
...
Options:
    -F  to force a thread dump. Use when jstack <pid> does not respond (process is hung)
    -m  to print both java and native frames (mixed mode)
    -l  long listing. Prints additional information about locks
    -h or -help to print this help message

$ docker run -it bellsoft/liberica-openjdk-alpine-musl:8 jstack -h
...
Options:
    -l  long listing. Prints additional information about locks
    -h or -help to print this help message

$ docker run -it bellsoft/liberica-openjdk-debian:11 jstack -h
...
Options:
    -l  long listing. Prints additional information about locks
    -h or -help to print this help message
```

Alpine Linux port in upstream

Unifies platform support across community and distributions. Helps maintenance and port development for perfect small containers. Liberica JDK Alpine musl containers are tested and TCK-verified.

Different uses are possible.

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

Alpine containers do help

The amount of transferred data for OS+JDK image can be decreased to 76 MB, overall pull time drops many times (like 28 s → 4 s or 6 s → 0.8 s).

Make More Users Happy

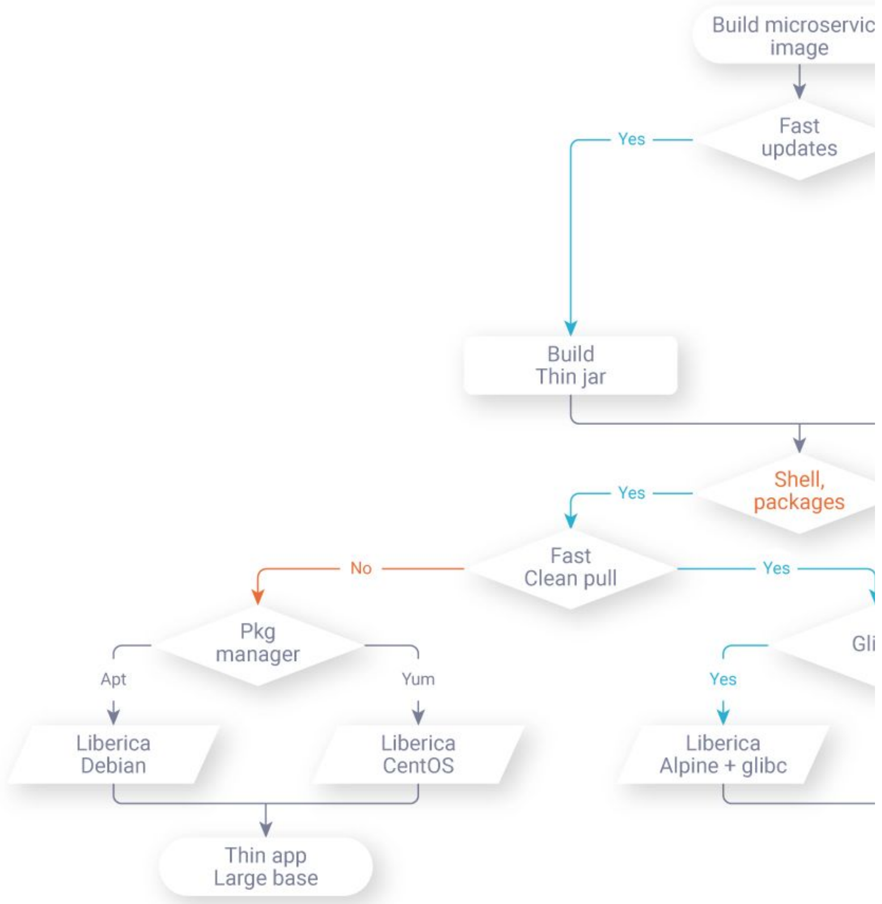
“*We plan to stay on Java 8.*

— NN% of users”

Portola Expansion

- **JDK 11 LTS**
  - Not in mainline (yet)
  - Historical downports in Liberica 9+
- **JDK 8 LTS**
  - Liberica 8u on Dockerhub
- **AArch64**
- **OpenWRT**
  - One more flavor of Raspberry Pi

Cheat Sheet



Build microservic
image
Fast
updates
Yes
Build
Thin jar
Shell,
packages
Yes
Fast
Clean pull
No
Yes
Pkg
manager
Apt
Yum
Gli
Yes
Liberica
Debian
Liberica
CentOS
Liberica
Alpine + glibc
Thin app
Large base