Contributing to OpenJDK is a competitive advantage

Alex Belokrylov
CEO

@gigabel

bell-sw.com | 2023
About BellSoft

BellSoft was founded in 2017 by Java and Linux experts with 15+ years of experience working in Sun/Oracle. Headquarters in San Jose, California.

Members of:

- JCP Executive Committee
- OpenJDK Vulnerability Group
- GraalVM Advisory Board
- Linux Foundation
- Cloud Native Computing Foundation
BellSoft’s contributions to OpenJDK

- Among top contributors to JDK 11 & 17
- Developed and integrated JEP 315 (aarch64 optimization) and JEP 386 (Alpine Linux port)
- Maintain the upstream Arm port
- Another important project is musl support in GraalVM

Since 2018, BellSoft focuses on Java in containers.
Why BellSoft contributes to OpenJDK?

- Contribute to a better Java future
- Improve platform well-being, security, and performance
- Help maintain company products

Liberica JDK

Liberica JDK is a 100% open source Java 8, 11, and 17 implementation.

Liberica Native Image Kit

Liberica Native Image Kit is a GraalVM-based tool for creating performant native images.

Alpaquita

Alpaquita Linux is 100% Alpine compatible, secure, and optimized for Java.

Release schedule for all products conforms to the LTS roadmap. All products are available for a large number of platforms.
OpenJDK development ecosystem
Moving Java forward — together

Companies
- Develop new language and platform features
- Maintain dependent products and HW/OS
- Fix issues, improve OpenJDK security

Java project and framework maintainers
- Propose platform changes
- Test Early Access builds, adopt new features
- Ensure Java ecosystem integrity

Independent contributors
- Develop OpenJDK in their area of expertise
- Fix issues, debugging them in their projects
- Study code

Regular Java developers
- Try new language features
- Ask questions (tough one!)
- Provide feedback on new features and ideas

JCP, JCP EC, JSR process, Java in Education
Step 1: Decide what to contribute
- Does it really solve a problem?
- Does it really benefit a sufficient number of users in the ecosystem, not just me?

Step 2: Decide how to contribute

Step 3: Implement
- Is your code well placed, written, and tested?
- Is it easy to review it?
- Is it easy to maintain it?

Step 4: Integrate

Step 5: Maintain the code
- Not just OpenJDK/JDK, LTS releases!

Step 6: Deprecate and retire code
Alpine Linux

Step 1: decide what to contribute

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

- Bullet 1
- Bullet 1
- Bullet 1
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
Step 2: decide how to contribute

Project Portola

- [openjdk.java.net/projects/portola](openjdk.java.net/projects/portola)
- JDK port to the Alpine Linux distribution, 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
Step 3: Implement

Musl port

- 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
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 do not have a proper shebang**
- Write proper # headers in *.sh

**Serviceability agent (private API) doesn’t work**
Alpine Linux Port

Step 4: Integrate

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

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.
PR & Review on Github

https://github.com/openjdk/jdk

- JDK-8247589: Implementation of Alpine Linux/x64 Port
  - Ensure all tests pass
    - Not just your new tests, and not just on the new platform
    - On all platforms!
  - 48 review comments during integration
  - Work with the reviewers to address their feedback
Step 5: Maintenance

Ensure it continues to work

https://www.openwall.com/lists/musl/2022/09/26/1
Subject: Revisiting LFS64 removal

⚠️ OpenJDK build will need to be fixed
“We stay on Java 8.”
~45% of users

“We stay on Java 11.”
~48% of users

Make More Users Happy

JDK 11 LTS
- 11.0.16 (July 2022)
- Historical downports in Liberica 9+
- Liberica 11u on Dockerhub

JDK 8 LTS
- Liberica 8u on Dockerhub
...is the operating system optimized for Java deployment, emphasizing high performance, security, small size, and flexibility.
<table>
<thead>
<tr>
<th>Top 4 features of Alpaquita Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enhanced security</strong></td>
</tr>
<tr>
<td>The lack of extra components means it is harder to break, and timely, frequent updates reliably remove the vulnerabilities. Additional security hardening is provided by userspace compilation options.</td>
</tr>
<tr>
<td><strong>Optimized performance</strong></td>
</tr>
<tr>
<td>Alpaquita’s features include tuned kernel, optimized libc, and optimized malloc options to boost the performance of your applications without sacrificing stability.</td>
</tr>
<tr>
<td><strong>Miniature size</strong></td>
</tr>
<tr>
<td>With its 2.9 Mb base image size, Alpaquita offers the smallest performant docker images, JDK docker images, and native images, making the deployment faster and memory footprint smaller.</td>
</tr>
<tr>
<td><strong>Liberica Lite and Liberica NIK</strong></td>
</tr>
<tr>
<td>Liberica Lite, the optimized version of Liberica JDK, enhances the performance and minimizes memory footprint. Liberica NIK allows creating the native images that benefit the project even more with Alpaquita Linux as the foundation.</td>
</tr>
</tbody>
</table>
Develop OpenJDK

$ git clone https://github.com/openjdk/jdk.git
$ cd jdk
$ ./configure
$ make images

JTREG
TCK
JCSTRESS
JMH
...

Create PRs
Use Skara automaton
Use bugs.openjdk.org

- Defects
- RFEs
- JEPS https://openjdk.org/jeps/0

Use mail.openjdk.org
Work on projects
Update projects differ
All contributions matter

Big, Small, Discussion, Feedback

Start with a small contribution

- Read the code in the area of interest of your daily job
- Read the mailing lists
- Ask questions
- Maybe you’ll find something that is not optimal, or a typo
- Suggest changes in a PR or discussion

Try new features

- Does it improve developer productivity?
- Provide feedback
Conclusion

JDK releases are the most active JSRs

It is possible to contribute to OpenJDK

- Allocate resources
- Follow the process
- Collaborate
- Influence the most important platform

Make contributions a daily job

- A part of business model
- Stay in touch
- Public visibility

Contributions bring value and help to build products and services
Thank you for your attention!