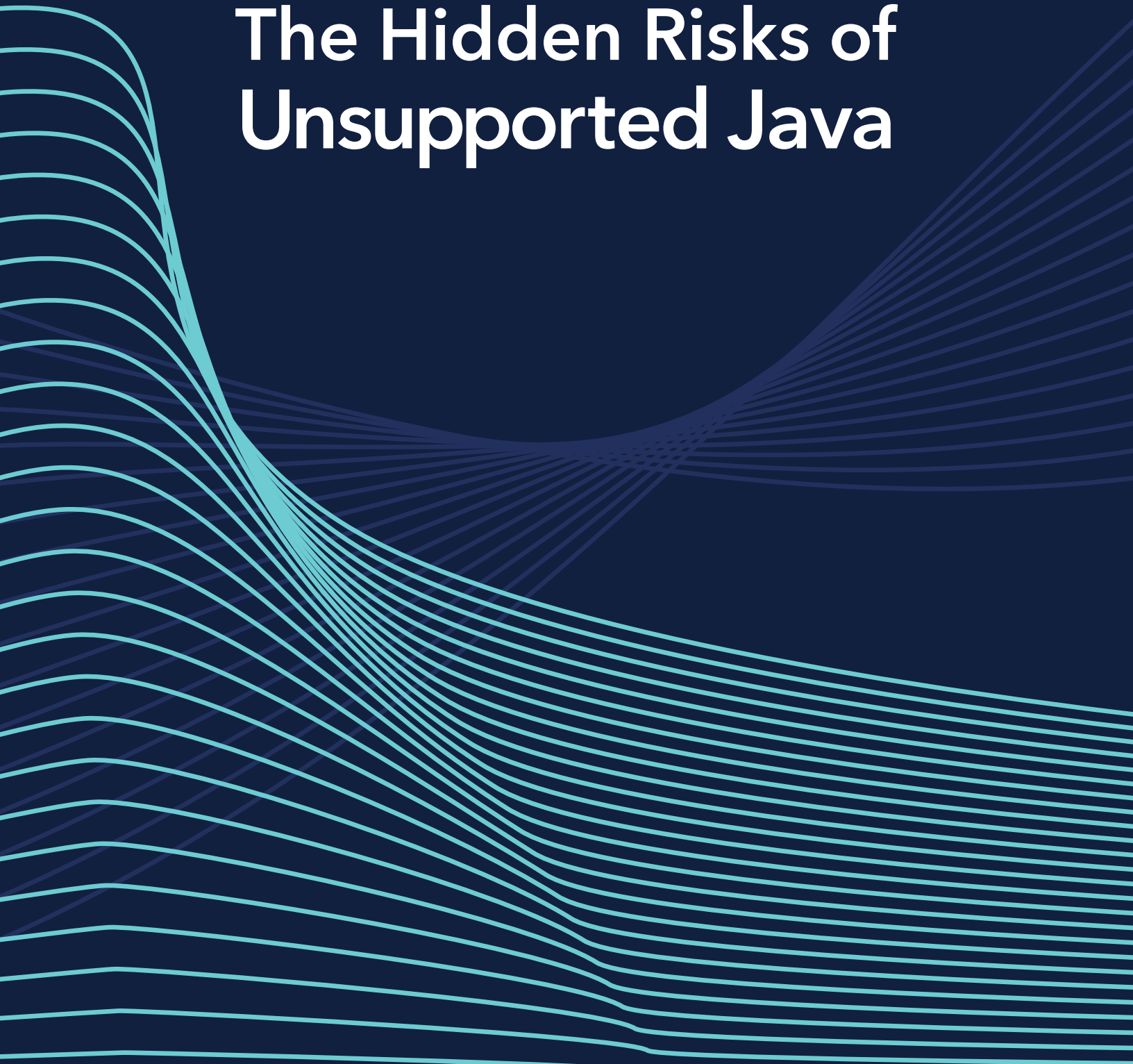




# The Hidden Risks of Unsupported Java



# The Hidden Risks of Unsupported Java

It's been more than 2 years since Oracle required subscription-based licensing for Java 8 and 11 support, the versions in production by the majority of Java users. Throughout this realignment, some enterprises embraced support, either from Oracle or from challengers, while others chose free versions. Now that enterprises have had some experience with unsupported OpenJDK versions or expensive support from incumbent Oracle or challenger OpenJDK distributors, enterprises should understand risks of unsupported OpenJDKs.

## In this eBook, we discuss:

- The understandable allure of free
- Hidden risks of unsupported OpenJDK that can severely impact your enterprise
- How to choose the best Java support
- How Azul is uniquely suited to meet enterprise needs

The Azul logo is positioned in the bottom right corner of the page. It features the word "azul" in a bold, lowercase, sans-serif font. The letter "z" is stylized with horizontal lines through it, and the letter "l" has a blue-to-white gradient. The background of the page features a decorative graphic of multiple thin, light blue lines that curve and flow from the bottom left towards the right, creating a sense of movement and depth.

azul

# The Easy Allure of \$Free Java

## Who doesn't like free?

**Nobody, of course.**

We all love to get something for nothing, particularly when we've been used to free candy.

Since the advent of Oracle subscription-based licensing that explicitly ruled out Java's commercial production use without paying, enterprises have had to choose between:

1. Paying high Oracle licensing fees for the pleasure of keeping the status quo
2. Using OpenJDK with an annual support subscription (not from Oracle)
3. Using a free OpenJDK and getting community updates and patches only, but no support.

Before you take the "easy route" and stay on or move to a free version, carefully weigh the benefits and costs. You need to consider the predictable benefits of free OpenJDK distributions against the high and very unpredictable costs of them.

# Risks of Unsupported OpenJDK

The risks of a free, unsupported OpenJDK are related to:

- security,
- compliance, and
- intellectual property issues.

All these, of course, impact operations and ultimately your whole organization.

## So how much can you afford these risks?

Risk	Likelihood	Avoidable	Potential Impact
New security vulnerabilities in your Java version	High	No	Security incident or breach
Free OpenJDK quarterly update internal testing shortfalls	Medium	Yes	Security incident or breach, and unstable performance of key applications
GPL v2 Section 3 source code violations leading to “cease and desist” letters	Low	Yes	Significant legal time and costs, leading to revenue loss and/or operational disruption, which may be significant
Unsupported software-related compliance violations	High	Yes	Fines and/ or legal fees
Slow security patch availability for publicly known vulnerabilities	High	Yes	Security incident or breach
Failure to inherit comprehensive Java IP rights to all OpenJDK code	Low	Yes	Intellectual property rights loss and/or operational disruptions

It is imperative to know the risks. This way, you can best assess how your free OpenJDK use cases relate to your organizational risk. Let's detail 4 key risks of unsupported OpenJDK you need to mitigate.

# Risk #1:

## Like all software, free OpenJDK has security vulnerabilities.

To date, the OpenJDK platform contains more than 8 million lines of complex code. As the platform continues to rapidly evolve, it brings more new features built using new code. And since new code is written by imperfect humans, more new bugs will always come along for the ride.

### **New security issues surface in every Java release.**

Particularly important bugs that impact security are also known as Common Vulnerabilities and Exposures (CVE). CVEs are essentially a free, collaborative list of publicly disclosed computer system security flaws.

Each identified vulnerability on the CVE comes with a Common Vulnerability Scoring System (CVSS) score to rate severity along a scale of 0-10. A rating of 0 means there is no risk.

As the score moves toward 10, the greater the risk, but the scale is logarithmic – think Richter Scale. A score of 7-8.9 means a high-risk vulnerability and a score 9 or 10 means it's critical.

This crucial list with its risk ratings helps IT, CISOs (Chief Information Security Officers), security teams and DevOps prioritize security efforts and action to fix exposures and vulnerabilities.

While the severity ratings highlight significance, like the magnitude of an earthquake, the deeper dive – the driving vectors of attack - describes where and how infrastructures are impacted.

This, in turn, directs organizational impact and best response organizations should take.

With this context, let's consider some recent history in Java, specifically Java 8, which still makes up the majority of Java deployments.

Take a look at the following table and what do you notice?

<b>Release</b>	<b># of High-Risk CVEs a.k.a Bugs</b>	<b># of Critical Risk CVEs a.k.a Bugs</b>
7/2020	2 at 8.3, 1 at 7.4	
4/2020	2 at 8.3, 1 at 8.1, 1 at 7.5	
1/2020	1 at 8.1, 3 at 7.5	
4/2019	2 at 8.1, 1 at 7.5	1 at 9.0
10/2018	3 at 8.3	1 at 9.0
4/2018	3 at 8.3, 1 at 7.7 and 1 7.4	1 at 9.0

**Nearly every release has bugs. High risk and critical risks are identified all the time.**

Once high and critically severe vulnerabilities become public knowledge, they can be quickly weaponized by hackers. When this happens, it exposes your operation to serious problems. Interrupted services, corrupted transactions, and data theft are just some of them.

**But what If you don't get timely Java updates to rapidly apply security patching?**

Without timely updates, your business is then exposed to an active, flourishing, global community of both state and non-state hackers. They have both the skill and the will to start weaponizing high and critical risk severities before you wake up the next day.

**Can your operations team keep up to head off damage?**



**Contact Azul**

1-650-230-6500

[azul.com/products/core](http://azul.com/products/core)

©2021 Azul Systems, Inc. 385 Moffett Park Drive Suite 115, Sunnyvale, CA 94089-1306. All rights reserved

## OpenJDK quarterly updates need extensive testing to ensure security.

Every free build of OpenJDK is **only** available as a Patch Set Update (PSU) binary.

PSU builds contain the latest security updates, which is great. But they also contain everything added by the OpenJDK community in the past 90 days.

This includes both:

- New features
- Non-critical bug fixes (that don't impact security)

**With this proliferation of functionality, you need to make sure each and every new PSU binary doesn't break and de-stabilize your applications.** And to ensure applications stay fully operational, PSU builds need thorough testing.

Then there are Security Only quarterly updates, also known as a Critical Patch Update or CPU. Like PSUs, they are released on the Tuesday closest to the 17th day of January, April, July and October.

These important stabilized updates contain new security or CVE updates and incorporate the prior quarter's PSU features and fixes, now deemed stable.

**Fortunately, stabilized security-only updates are designed for rapid deployment.**

This means operations teams can quickly address security issues and deploy with absolute confidence to ensure stability and minimize any operational risk.

Trouble is, no free distributions and too few OpenJDK distributors ship Security Only Updates — thus, truly stable builds are only available as part of subscriptions.

Also, there's a perilous timing gap in security patch availability.

With unsupported OpenJDK builds, enterprises need to mind the gap: that's the interim between the time vulnerabilities are publicly announced (thereby becoming widely known to hackers) and the time free OpenJDK patches are supplied.

This can take days, and in some cases, weeks.

Remember that the gap that could imperil your apps is a quarterly occurrence.

**Using a free, unsupported or poorly supported OpenJDK build, your systems may be at risk for days - maybe even weeks - every quarter.**

**It all adds up to continuously reoccurring, unpredictable risk.**

In the perilous gap that repeats quarterly, anything can happen. Enterprises are left exposed. With bad actors working tirelessly to exploit them, operational teams take a deep breath and hope nothing happens... this quarter.

**Is this really a risk worth taking?**

## **How important is a Security-only release?**

The unstable July 2020 PSU quarterly update included a significant regression, supposedly a "fix".

Untested, it broke many common applications like Apache Hadoop clusters, Apache Lucene and Apache Solr, among others.

The stabilized security-only update did not include the supposed fix. For more than 2 weeks, free OpenJDK users had to choose between security or stability. Over this period, many mission-critical systems had to degrade performance to reduce risk.



# Risk #2:

## Unsupported builds of OpenJDK don't protect your intellectual property.

To understand how unsupported OpenJDK doesn't protect you and your intellectual property rights, it helps to know how the line is drawn between open and closed source licensing.

Here is the overview:

1. OpenJDK is licensed under a combination of open-source licenses:
  - [GPL v2](#) for much of the Java Virtual Machine (JVM)
  - [GPL v2 + the Classpath Exception \(CPE\)](#) for parts of the Java Development Kit (JDK) and parts of the VM. The Classpath Exception (CPE) allows that any application code on either the class path or module path is not affected by the copyleft nature of the GPLv2.
  - Several other open-source licenses for parts of the JDK (Apache 2.0, LGPL, MIT, BSD, etc)
2. Oracle Java SE (Hotspot) aka the Oracle JDK is the closed source Oracle-equivalent of OpenJDK
3. OpenJDK Community Technology Compatibility Kit (TCK) is not open source. Instead, it is licensed to OpenJDK-based Java SE implementations under an OpenJDK Community TCK License Agreement (OCTLA). To date, only three companies signed this license agreement for Java versions 7, 8, and 9+ (including 11 and 17).

With that in mind, when a first version of a software is initially released under the General Public License (GPL), it is considered distributed. When someone else modifies that version and then makes it available, it is considered *redistributed*.

From a licensing perspective, there's no difference between "redistribute" and distribute since "redistribute" is a repetition of the original that was distributed.

So redistributing OpenJDK carries the burden of source code servicing.

Specifically, GPLv2 Section 3 defines very clear source code guidelines and responsibilities of the "redistributor" of any GPLv2 binary. **The redistributor must support source code requests for up to 3 years.**

For software vendors (e.g., ISVs), original equipment manufacturers (OEMs), or other organizations that redistribute \$free OpenJDK builds together with their Java application, you might be violating GPLv2 requirements.

You could end up with a "cease and desist" letter at a minimum.

Or you could be required to make your applications public which means you likely will stop selling them. That means significant legal risks and cost, operational disruption and potentially significant revenue losses.

## Pay attention to software licensing

In March 2021, the maintainer of one software library gave a second Ruby library maintainer that incorporates code from that first library some bad news. It was shipping under an incompatible software license.

The first library is licensed under the GPLv2 license and the second was incorrectly listed as an MIT licensed project. This licensing mistake forced the second library to discontinue distribution of some versions.

Impacting hundreds of thousands of projects, it was a completely preventable licensing mistake.

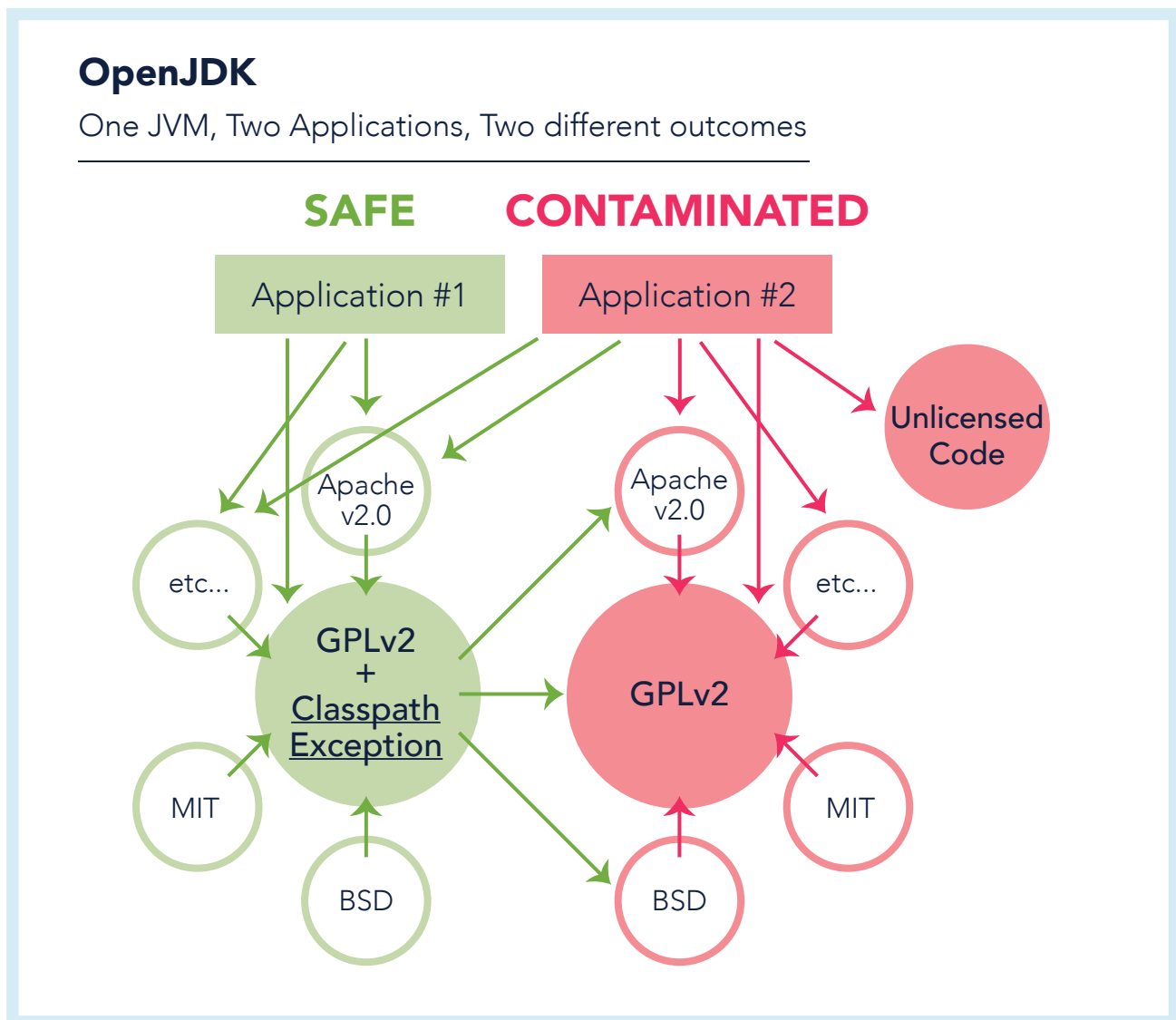
## What is Copyleft?

Copyleft is the practice of granting the right to freely distribute and modify intellectual property. It requires that the same rights be preserved in derivative works created from that property.

## The potential for open-source license contamination is a real risk.

License contamination can force you to open source your application's IP or obtain additional and potentially expensive licenses from third parties.

The following illustration shows the reality of OpenJDK when 2 applications using APIs run on a single Java virtual machine.



Without scrupulously verifying every piece of code in every module in every release of OpenJDK, there may be **risks that jeopardize your intellectual property (IP)** ownership. Without indemnification guarantees, it's impossible to manage that risk.

## What is an indemnification guarantee?

An indemnification guarantee protects any intellectual property in your applications.

It means your organization uses specific builds of OpenJDK that went through formal certification and verification. It ensures that your applications or products that:

1. include,
2. embed and/or,
3. distribute

a given OpenJDK in your applications or products does not contaminate application's IP or code with license requirements (including but not limited to the source code disclosure requirements of GPLv2).

In addition, without inheriting comprehensive Java IP rights to all OpenJDK code by using a JDK binary that passes the Oracle-licensed OpenJDK Technology Compatibility Kit (TCK) test suite, there are legal challenges.

Unfortunately, not every freely available OpenJDK binary contains these rights.

Some OpenJDK distributions simply lack access to TCKs.

And if they do, TCKs are inconsistently applied to each binary build. As such, any random Java IP patent holder and not just Oracle, can sue users who rely on non-TCK tested builds. And any Java IP patent holder could be a patent troll, or anyone else with a Java IP patent.

Therefore, as you assess the benefits of unsupported Java against their risks, ask yourself:

- Are embedded or redistributed non-verified binaries in the apps you build at risk for IP litigation?
- Can your organization afford a legal battle?
- Can you afford customer backlash over redistribution of non-IP protected binaries?

# Risk #3:

## Compliance issues could arise leading to expensive legal fees, fines, brand damage, and revenue loss.

Running mission-critical software on OpenJDK versions without well-defined support services is not a smart business move. It's even more dangerous when your business depends on OpenJDK, like a SaaS company delivering products using Java-based services and/or when your customers and regulators have visibility into production applications.

**There are big consequences to finding “unsupported and unsecured software running in production.” And when you run a free OpenJDK version, it's a realistic possibility.** It can inevitably lead to compliance issues. You can lose compliance certifications from standards bodies important to proving you follow industry security standards.

In addition, you jeopardize compliance with legal requirements. This is particularly true if you're a publicly-traded company where you face regular audits, and possibly the wrath of regulatory and government agencies.

And regardless of whether or not your organization is public, you must face your own customers. Not being able to prove compliance with industry standards costs you both current and future customers.

## **To remain in compliance, apps need stability and timely security updates.**

So, to comply, your customer-facing, revenue-generating or business-critical Java-based applications must be carefully managed by operations teams. This means ensuring that all components of the runtime stack have appropriate support service levels.

Those support levels typically focus on system stability and security.

This requires, in practical terms, timely access to and near-immediate deployment of vulnerability fixes and security updates. It also requires out-of-cycle-bugs patches, covering your systems at all times, even weekends and holidays.

Support for system stability and security is the only way to ensure regulatory compliance.

## **Certain industries have higher compliance burdens.**


This is particularly true in certain industries and if your organization must comply to rigorous standards like Payment Card Industry (PCI) Data Security Standards (DSS).

For example, PCI requires software updates within 30 days of the release of security patches as part of a support contract. Without it, your compliance with the PCI DSS is in jeopardy.

As many recent news reports testify, power and utility companies are massive, highly public, and frequent targets for state and non-state hackers.

So, in the United States, North American Electric Reliability Corporation (NERC) compliance standards require that Bulk Electric Systems (BES) implement strict security controls. This important standards body requires clear Service Level Agreements (SLAs) to ensure resilience against potential cyberattacks.

In healthcare, companies must comply with the Health Insurance Portability and Accountability Act (HIPAA) Security Rule. While it doesn't specifically address patching, it does subject identified vulnerabilities to HIPAA administrative safeguards and security management process standards.



## As you assess your own risks of unsupported OpenJDK distributions, ask yourself:

- ❑ What regulatory mandates drive operations teams?
- ❑ Do you have proven valid support contracts for all essential software?
- ❑ Can you afford the hours or days of downtime (and damage to your reputation) while you address an entirely preventable security or product issue from not applying the latest Java updates when released??
- ❑ Can you really afford potential impacts that free and unsupported applications may have to your security posture, compliance, reputation and finances?
- ❑ Is your organization in an industry where attacks are frequent?

# Risk #4:

## Assuming a “\$Free” and Unsupported OpenJDK is enough.

In the past, Sun Microsystems and then Oracle led all OpenJDK projects and maintained “Long-term Support (LTS) versions like 6, 7, 8 and 11, directly contributing fixes to them. Furthermore, they did this without requiring subscription.

Unfortunately, those days are long gone. The altruistic spirit and open-source commitment of Sun Microsystems has given way to Oracle’s commercial aims.

The focus for Oracle now, publicly at least, is on supporting and maintaining the current “cutting edge” OpenJDK stream. Oracle does backport fixes to Oracle Java versions such as 7, 8 and 11, but requires you to have an expensive subscription to use them.

The OpenJDK community owns the backporting process for other OpenJDK versions, but this process takes time and resources.

**As a result, risks arise around community backport shortfalls for commonly-used Long-Term Support (LTS) versions, such as 8 and 11. Risks include both completeness and quality of backported code.**



## Unsupported OpenJDK distributions can't guarantee timely access to updated, secure binaries.

Using an unsupported OpenJDK build does not guarantee timely access to updated and secured binaries. Nor does it protect IT infrastructure from software regressions that arise from the OpenJDK community, either due to inactivity or incomplete activities.

Whether it's Oracle's regression contribution or a community volunteer's coding mistake, operations teams with no rapid out-of-cycle patching support risk unscheduled downtime of mission-critical applications.

In a 2018 Java update, for example, Cassandra servers failed to start.

In this case, many operations teams couldn't fall back to the previous updates because the most recent one contained critical security risk fixes.

In the past, development teams could work across all supported versions, because Sun and Oracle were responsive and fixed them – in Java 8, 7, 6, any long-term support version.

Now, it doesn't work that way.

What was once free from Sun and then Oracle, now requires payment.

While Oracle continues to provide thorough security guarantees across supported versions, it's expensive. Meanwhile unsupported OpenJDK builds can expose more issues across Java apps than ever before.

It all makes speedy access to out-of-cycle patches crucial for mission-critical deployments.

## Going without a support subscription risks your business.

While some open-source projects have found success by only leveraging volunteer contributors, Java's success was built on the sponsorship of Sun Microsystems and later Oracle. So OpenJDK works within both its legacy and the community that grew up around it. Although it's a vibrant community, it is no longer backstopped by a large, dedicated sponsor.

Thus, as a result:

1. **Only Oracle, the one-time steward of Java, contributes bug fixes and security updates to the latest Java cutting-edge versions.**
2. **The OpenJDK community then backports Oracle changes** to the Java releases that most organizations use (e.g., Java 11 and 8).
3. **This adds backport uncertainty.** The free OpenJDK community made up of mostly volunteers must complete this cascading-style backporting (e.g., 11→8→7→6, etc.).

Meeting timely requirements for patching public bugs is a large task even for Java's large and dedicated community of volunteers.

Let's take an example.

In the October 2019 update, Oracle fixed 19 bugs and backported them to Oracle Java SE 8. The OpenJDK project lacked the resources to complete backporting before the security release date. This meant they had to defer this work to a later quarterly release.

When this happens, we're back to the perilous gap.

Enterprises need to care about security and, by extension, Oracle JDK compatibility.

**The conclusion is that it's important for enterprises to use an OpenJDK that keeps pace with Oracle.**

Oracle is a well-resourced company and does a great job ensuring security and stability.

Trouble is their support costs way more than it should.

Now that you better understand the risks of free, unsupported Java -- and you've thought about how they apply to your IT team and entire enterprise -- let's explore how you can mitigate them.

# Reduce Risk with the Right Java OpenJDK and Support Subscription

If your organization has applications that use or depend on Java, an OpenJDK support subscription may be worth the investment.

**For some, a support subscription is like buying an insurance policy cushioning against disaster. For others, it's an advisory service that saves development and app teams' time. And for others, it's intellectual property protection and operational assurance.**

Regardless of your reason, make an informed decision on the right OpenJDK support for you.

**For complete support and peace of mind, it's important to remember a few things as you choose a Java partner:**

- 1. Your OpenJDK annual subscription should come complete with service level agreements.** Quarterly updates or random security patches are not enough. Security fixes must be delivered as soon as possible once security vulnerabilities are published, while out-of-cycle fixes are also essential.
- 2. Speeding security updates to production is everything.** It's imperative that stable builds be deployed rapidly into your production environment. Furthermore, your OpenJDK partner should have a track record of success that backs up their security and stability claims.
- 3. Your OpenJDK subscription shouldn't come with financial, intellectual property and legal risk. It's non-negotiable.** Your OpenJDK partner's binary should be verified compliant with the Java SE specification using the OpenJDK Community Technology Compatibility Kit licensed from Oracle. Your partner also needs to have signed the OCTLA agreement for at least Java 8 and 9+, and ideally 7, as well as provide guarantees that Java classes and APIs are not contaminated
- 4. Your Java support partner should be able to support your entire Java estate, regardless of operating system or version, whether it is on-prem, on a virtual machine or in the cloud.** This means they need to have longstanding, deep knowledge of Java, and one which is not just focused on one specific Linux platform or an individual cloud OS. They must have the range of skills to support your organization's heterogenous Java needs.

Most importantly, support should always be available when you need it.

You should be able to easily reach your support team, even on weekends and public holidays regardless of your time zone. After all, easy accessibility is the only way you can truly rely on your Java partner's experience, dedication, consistency and expert knowledge.

# Not all Java Support is Equal

Some companies simply offer support. Others live, breathe, and love Java... and only Java. **At Azul, Java is our heart and soul and we love Java.** This is true today as it was back at the company's foundation in 2002.

## **Azul has a longstanding commitment to the Java community.**

Over the years, Azul has been continuously committed to the Java ecosystem and OpenJDK community to evolve and improve production Java.

We proudly participate as an:

- OpenJDK project key contributor
- OpenJDK Vulnerability Group member (4 Azul employees amongst 28 members from 13 companies) to ensure coordinated releases of critical security updates across the industry
- Java Experts Group member to decide features for 9 releases of Java SE (JDK9-17)
- Java User Group participant with 2 Azul employees serving as leaders of groups and 7 who actively present at them
- Long-standing Java Community Process (JCP) member since 2011, driving the future of Java

Azul-employed engineers also lead projects for OpenJDK 7, 13 and 15 (including support for Apple silicon), more than any other organization. In addition, Azul team members are active on the OpenJDK community platform, [foojay.io](https://foojay.io).

This platform, short for **friends of OpenJDK**, is a web site with user-focused Java and OpenJDK technical dashboards. Alongside others, [Azul participates on its advisory board](#).

## **Azul has a long and strong track record of supporting and powering mission-critical deployments.**

Our founders' involvement in Java stretches back to the mid-1990s and the early days of Java.

Since starting Azul to advance enterprise Java, they've pioneered many innovations including:

- Continuously Concurrent Compacting Collector (C4)
- Java Virtualization
- Elastic Memory
- Various managed runtime and systems stack technologies
- Back-ported Mission Control, TLS 1.3 and other security protocols to OpenJDK 8
- The OpenJDK build that powers the Apple M1 silicon

Together, these innovations combine to deliver the expertise that underpins the most scalable and robust Java platforms.

## **With Azul Platform Core, there's no need to compromise value for high security.**

There are only two choices for security-only updates and stabilized builds: Oracle or Azul. And only Azul provides open source OpenJDK.

By going the Oracle route, enterprises get great security but at very high cost.

But by going with Azul, organizations get a track record of secure, stable OpenJDK delivery for more versions with support customers love - all without breaking the bank. In addition, security-only releases are suitable for immediate deployment and backed by strict Service-Level Agreements (SLAs).

## Azul Platform Core ensures certified compliance with the Java SE specification to reduce risk.

In addition, your intellectual property and smooth IT operations are safe with Azul's subscriptions. Be assured that builds shipped with Azul Platform Core:

- Are verified compliant with the Java SE specification using the OpenJDK TCK licensed from Oracle, a test suite of more than 120K unit tests.
- Carry additional intellectual property (IP) rights granted by passing the TCKs as defined by the Java Specification Participation Agreement and
- Provide extensive IP rights to compatible and specification compliant implementations.
- Get license non-contamination certification by Azul.

### How Does Azul Guarantee Non-Contamination?

1. In the process of certifying each specific binary package, Azul performs detailed analysis of each build artifact.
2. This analysis tracks and identifies any and all binary and object material that is accessible for linking by programs that may execute using the specific Azul Platform Core binary package.
3. The specific source code used to produce each accessible binary or object material portion is identified. Then the copyright and licensing associated with that source code is also identified and classified.
4. The detailed relationships between the various parts of the code and their respective licenses are fully analyzed.

The result of this formal process is verification and assurance that linking with accessible APIs of the specific binary will not cause IP contamination.

## **At Azul, access the most skilled JVM team on the planet.**

According to IDC, there are over nine million developers around the world. Of this army of developers, 69% use Java – more than any other programming language.

Amidst immense competition from a huge population of talented developers, Azul boasts more than 100 very experienced full-time engineers that work solely on JVMs and JDKs. In addition, of 30 Java Champions chosen each year, Azul is honored to have four Java Champions on staff, working directly with customers and the Java community.

When you subscribe to Azul Platform Core, know your organization has access to the industry's very best Java experts' deep knowledge of the JVM, memory management, Java performance issues, and usage of production application visibility tools.

Azul is ready and able to solve even your hardest problems – 24 hours a day, seven days a week.

### **Azul's independence is your Independence.**

As the only Java vendor that operates independently from all major independent software vendors (ISVs), operating system vendors and cloud providers, Azul has no hidden agenda.

It has no distractions from other product organizations. At Azul, it's only ever about Java, and not about locking you into an enterprise application suite or making it impossible to move away from an expensive cloud provider.



Best of all, Azul has the expertise and experience to support and secure your entire Java estate - regardless of what's in it including:

- [Windows](#), Linux [including Alpine], [macOS](#), and Solaris
- Docker and Linux containers for [cloud](#)
- Installers and package formats (DMG, MSI, RPM, DEB, APK, etc.),
- Processor architectures like x86, Arm, PowerPC, and SPARCv9
- 32- or 64-bit applications
- Any Java patch levels
- Any disk and memory sizes where JREs, modular builds or Compact Profiles (Java 8 only)
- Long-Term and MediumTerm Support Java Versions, including JDK 15, 13, 11, 8, 7 and 6

At Azul, you can choose between Standard, Premium, and Platinum Enterprise Plans, so you always get exactly what you need. No lock-in and no strings attached.

## Perfecting the Art of Java in Azul Platform Core

It's important to understand that Azul innovations do not change core OpenJDK code. Rather, our curation of Java perfects the art of OpenJDK.

In addition to fixing bugs, we organize, curate and oversee products that augment OpenJDK distributions with components that:

- 1) Ensure a smooth transition from the Oracle JDK while maintaining the same high standard of runtimes, for example the provision of key fonts once used by Oracle,
- 2) Facilitate easy deployment and easy production systems analysis, profiling and monitoring,
- 3) Reduce risk of using redistribution of OpenJDK-based runtime to include security vulnerabilities, IP contamination, patent litigation, and runtime compatibility with Oracle.

## **Azul Platform Core is ideal for all organizations with Java applications.**

If your apps and organization require the security that comes with regular Java security updates backed by strict SLAs, then Azul is right for you.

If you need support for selected older versions of Java, for example 7 or 6, then Azul is your only option.

If you need responsive and timely support from one of the largest groups of Java experts, then [enter the world of Azul](#).

## **About Azul**

Azul, the industry's only company exclusively focused on Java and the Java Virtual Machine (JVM), builds fully supported, standards-compliant runtimes that help enable Java-based businesses. Learn more at [www.azul.com](http://www.azul.com).

## **Contact Azul**

1-650-230-6500

[azul.com/products/core](http://azul.com/products/core)

The Azul logo consists of the word "azul" in a bold, lowercase, sans-serif font. The letter "z" is stylized with horizontal blue lines passing through it, creating a sense of motion or depth. The rest of the letters are solid dark blue.

azul

