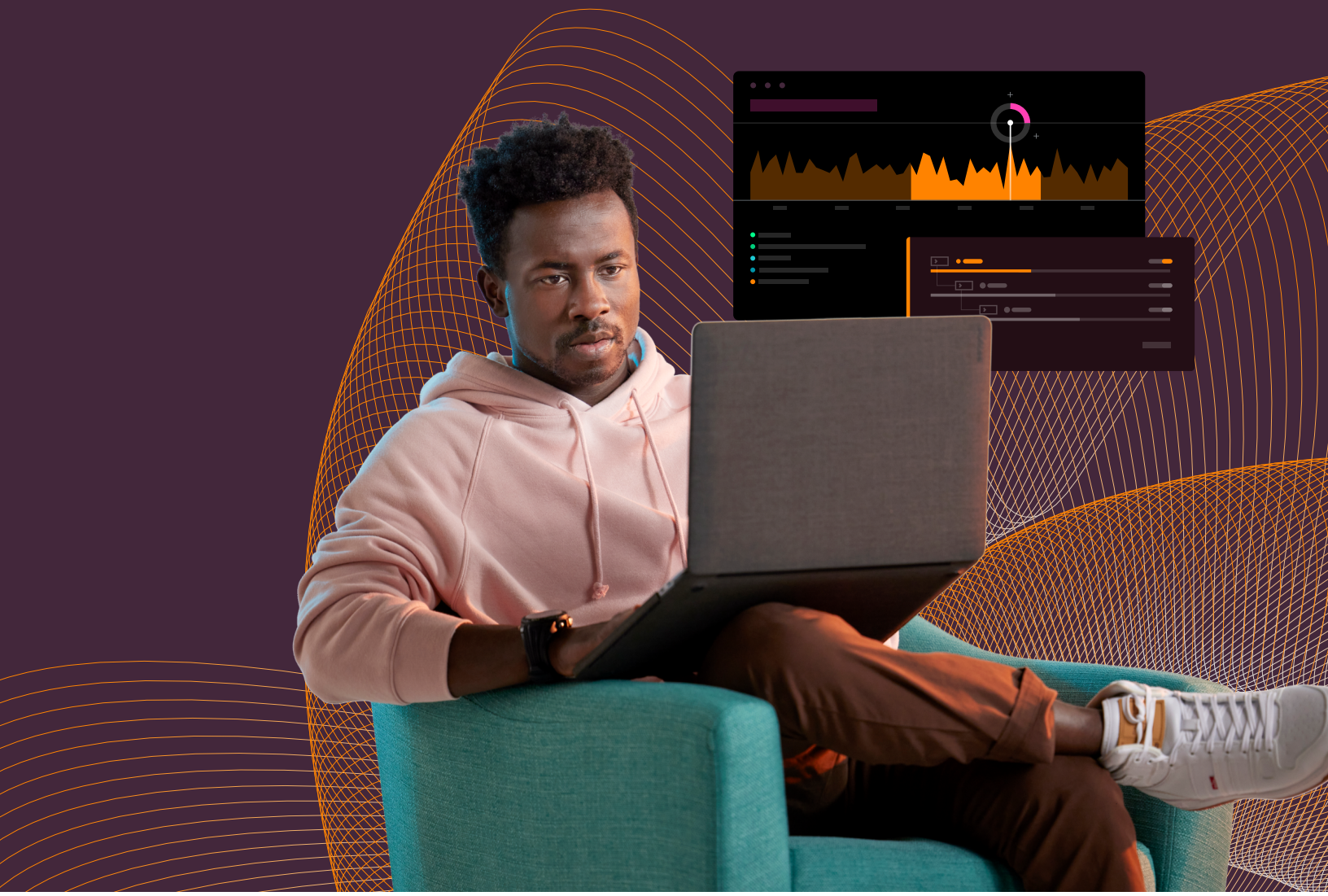
new relic



2023 State of the Java Ecosystem

An in-depth look at one of the most popular programming languages

Report / April 2023

Contents

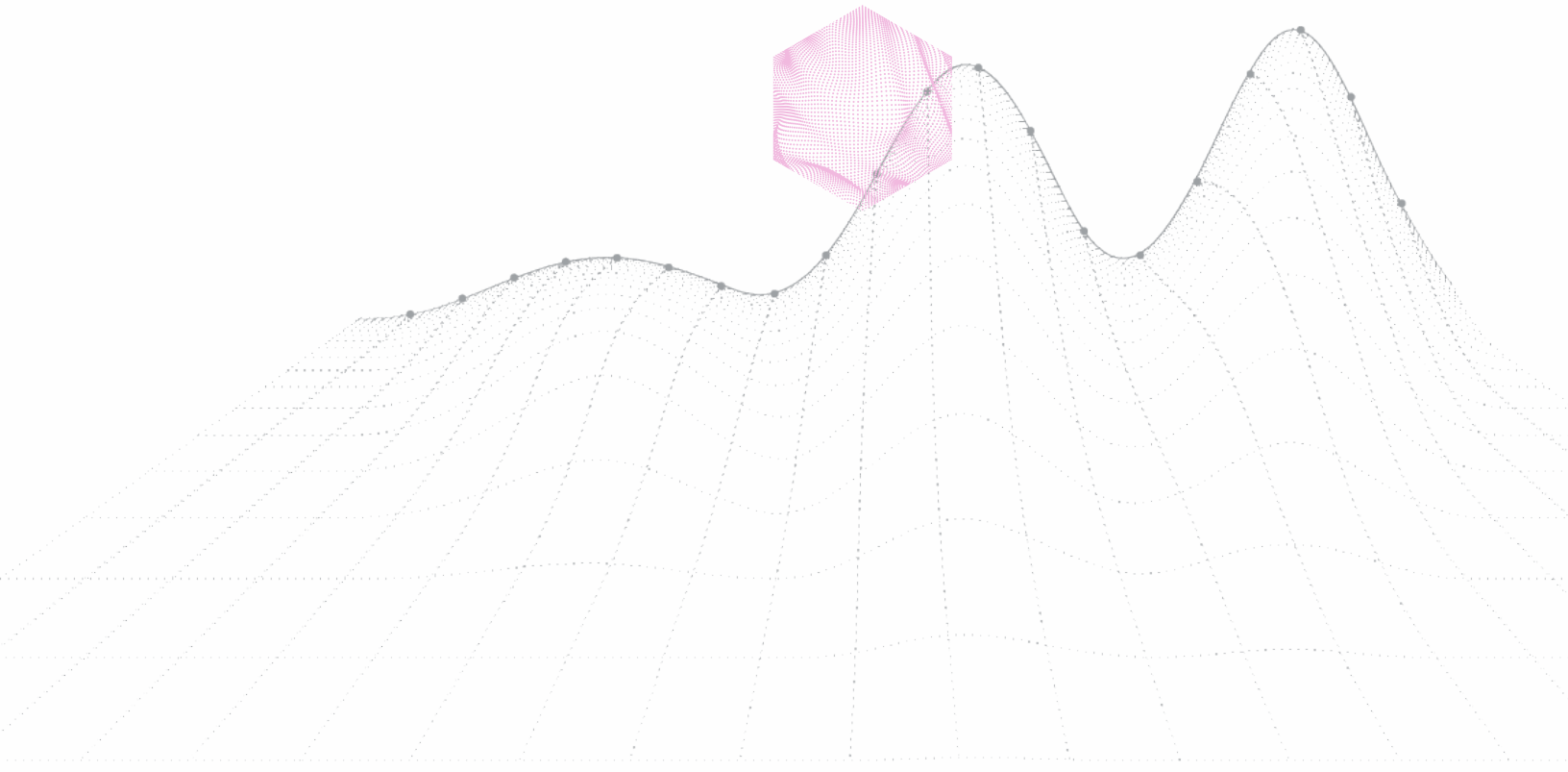
Overview

Sun Microsystem released the Java programming language in 1996 to provide a more portable and interactive way of developing modern multimedia applications. It has remained one of the most popular programming languages ever since. Java is used in almost every major industry and economic sector because it is platform-independent, offers thousands of libraries, and is well-supported.

New Relic has been monitoring the Java ecosystem for the past few years to uncover shifts in how it is being used due to new version releases and the rise of containers. The 2023 report provides context and insights into the current state of the Java ecosystem.

The following categories were examined:

- The most used Java versions in production
- The most popular JDK vendors
- The rise of containers
- The most common heap size configurations
- The most used garbage collection algorithms



Java 17 user adoption grew 430% in one year

Every two to three years a Java release is designated as long-term support (LTS) and receives quarterly stability, security, and performance updates only—not new features.

More than 56% of applications are now using Java 11 in production (up from 48% in 2022 and 11% in 2020). Java 8 is a close second with nearly 33% of applications using it in production (down from 46% in 2022).

While Java 11 has held the top spot for two years in a row, the adoption rate of Java 17 far exceeds what the developer world saw when Java 11 was introduced. More than 9% of applications are now using Java 17 in production (up from less than 1% in 2022), representing a 430% growth rate in one year. It took years for Java 11 to catch anywhere near that level.

Only 0.28% of applications are still using Java 7 in production, which makes sense since support for Java 7 ended in 2022. Most of the applications using Java 7 are legacy applications that have not been upgraded.

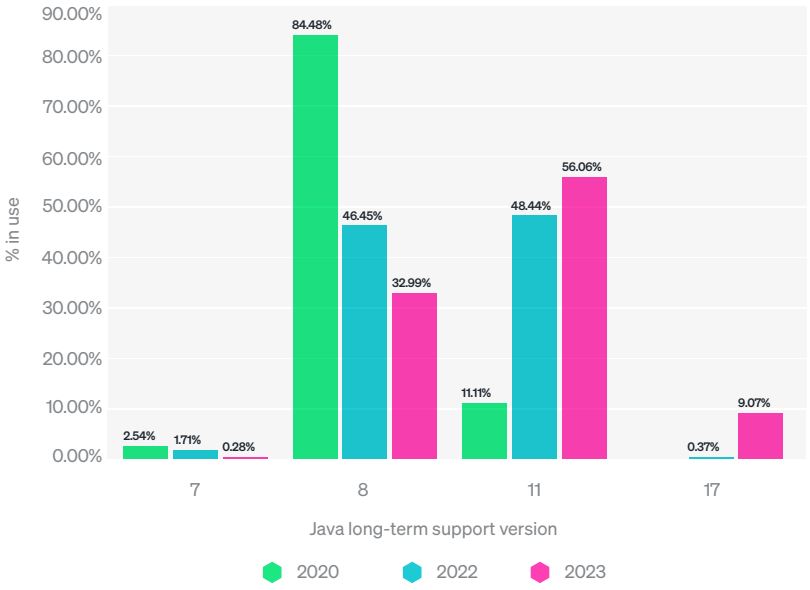


Figure 01. Percentage of Java LTS versions in use

Java 14 is the most popular non-LTS version

Starting with Java 9, the release pattern for the platform changed. Approximately every six months a new version of Java is available, but these versions are only supported until the next release. The intent is to make new features available more often.

Uptake for interim, non-LTS Java versions remains extremely low compared to LTS versions in production with only 1.6% of applications using non-LTS Java versions (down from 2.7% in 2022).

Some factors that could be impacting the decline in non-LTS version usage include:

- Lack of support
- Perceived attractiveness of the features
- Length of time until the next LTS version

Between version 8 and version 11, it was not clear precisely when the next LTS version would be released. Now there is a settled timeline, which has been reduced by two to three years. The next LTS version is expected to be 21, not 23, which may explain why some developers are willing to wait.

Out of the non-LTS Java versions in use, Java 14 is still the most popular (0.57%, down from 0.95% in 2022) with Java 15 a close second (0.44%, down from 0.70% in 2022).

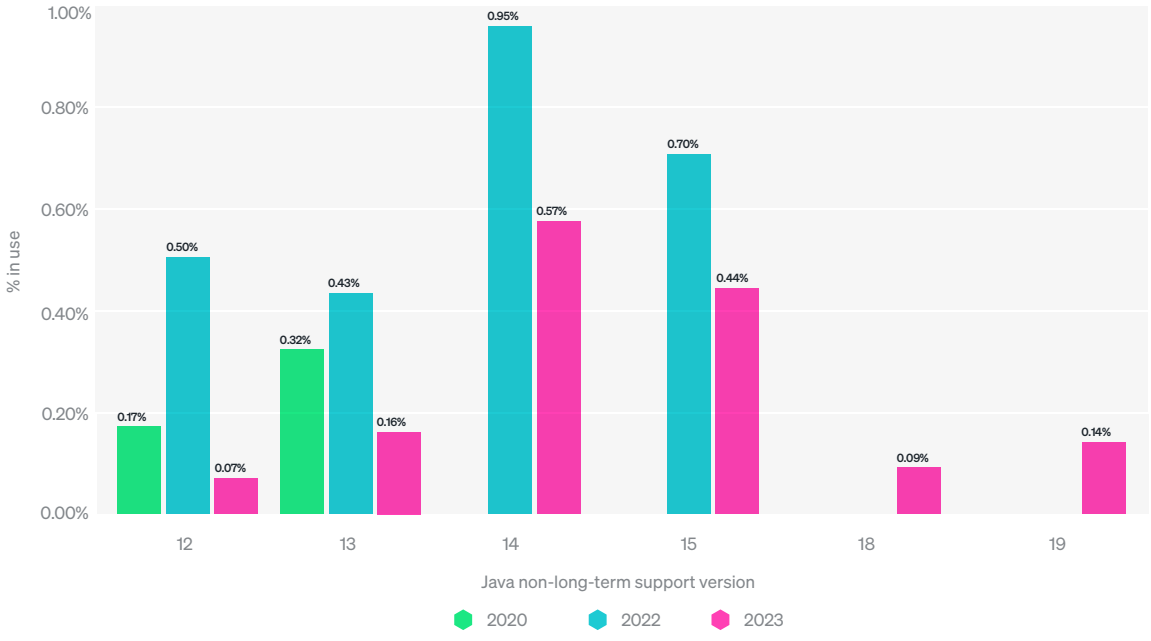


Figure 02. Percentage of Java non-LTS versions in use

Amazon is now the most popular JDK vendor

Recent years have seen changes in the source of Java Developer Kit (JDK) distributions in use. Many developers used to get their JDK from Oracle, but the open-sourcing of Java in the OpenJDK project has yielded a wealth of options.

In 2020, Oracle was the most popular JDK vendor, comprising roughly 75% of the Java market. There was a noticeable movement away from Oracle binaries after the more restrictive licensing of its JDK 11 distribution (before the return to a more open stance with Java 17). While Oracle retained the top spot in 2022 with 34%, it slipped to 28% in 2023.

The use of Amazon has increased dramatically to 31% of the market (up from 2.18% in 2020 and 22% in 2022), making it the most popular JDK vendor.

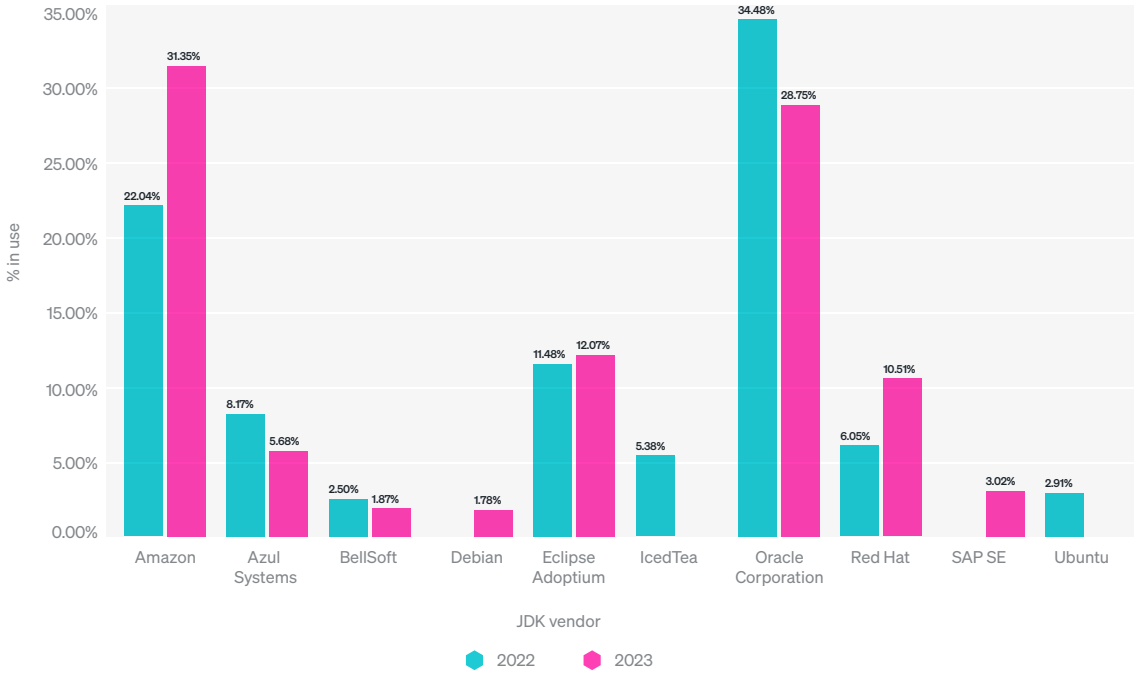


Figure 03. Percentage of JDK vendors in use

Containers rule everything around us

Containerizing applications has become mainstream—70% of Java applications reporting to New Relic do so from a container.

Compute settings in containers

Containers impact how engineering teams allocate compute and memory resources. For example, the New Relic data shows a much higher percentage of applications running with fewer than four cores when in containers.

Engineering teams are moving away from single-core settings in containers, with only 36% in use (down from 42% in 2022), and moving toward multi-core settings, with over 29% using an eight-core setting (up from 20% in 2022).

Engineering teams typically use smaller compute settings in cloud environments where they often deploy containers. However, this trend can pose unexpected issues for some applications, which could be contributing to the decrease in configuration. For example, if teams run with only one CPU, they may not get the garbage collector they expect—even if they set it explicitly.

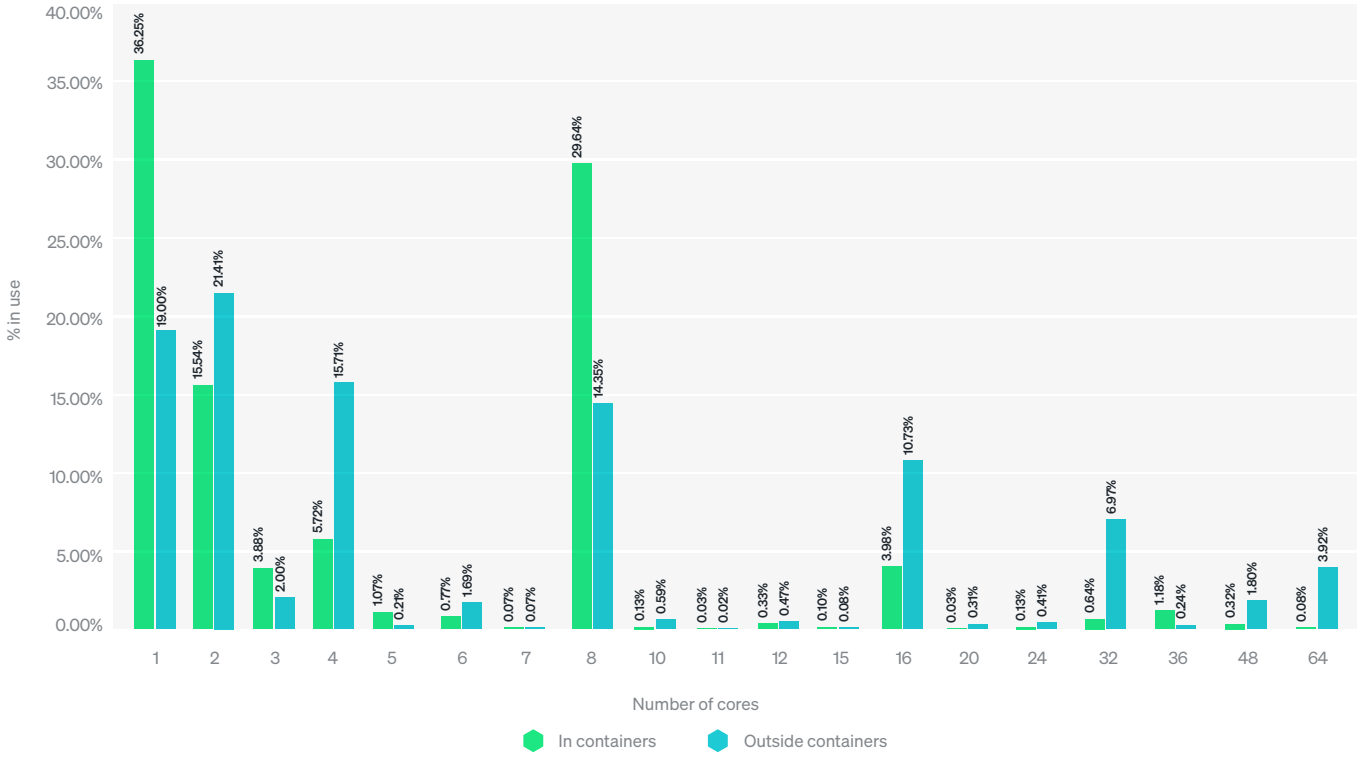


Figure 04. Percentage of Java applications in and outside containers by the number of cores

Memory settings in containers

Similar trends surface when comparing memory settings, with a tendency towards smaller instances in containers. The nature of deploying containers often makes developers more conscious of their footprint because limits are more strictly enforced.

Java 9 introduced a number of features to work better with containers, such as the `-XX:MaxRAMPercentage` startup flag that replaced specifying a precise heap size via `-Xmx`. The Java virtual machine (JVM) is aware of the container memory limits, so the `-XX:MaxRAMPercentage` easily scales the JVM to container sizes.

The New Relic data shows 30% of containerized applications explicitly request an upper bound on JVM memory through `-XX:MaxRAMPercentage` flags (up from 9% in 2022), so the flag is catching on.

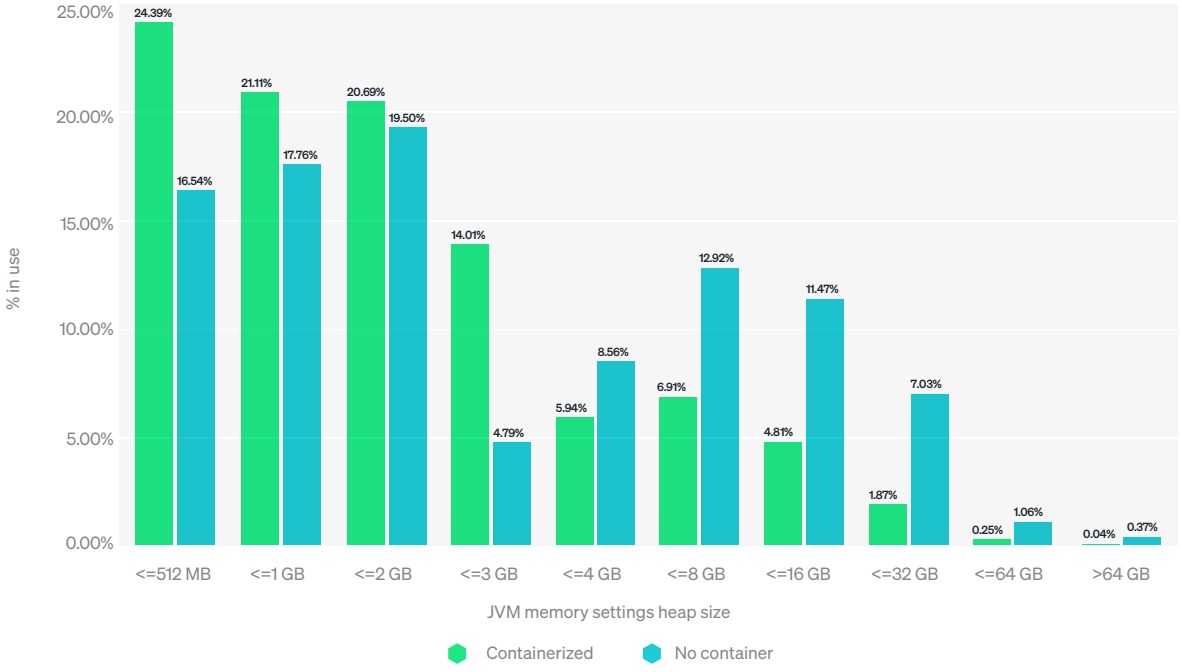


Figure 05. Percentage of Java applications with and without containers by JVM memory settings heap size

Garbage in, garbage out

Automatic garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting unused objects. Given its central role in JVM performance, garbage collection remains a hot topic in the Java community.

New Relic data shows that the Garbage-First (G1) garbage collector continues to be a clear favorite for those using Java 11 or later versions, with 65% of customers using it. One of G1's primary benefits is that it clears smaller regions instead of clearing large regions all at once, which optimizes the collection process. It also rarely freezes execution and can collect both the young and old generations concurrently, making it a great default for engineers.

Other experimental garbage collectors that have appeared post-Java 8 (ZGC and Shenandoah) still show small usage in production systems. Both have production-ready releases, but there is still a negligible uptake in general processing.

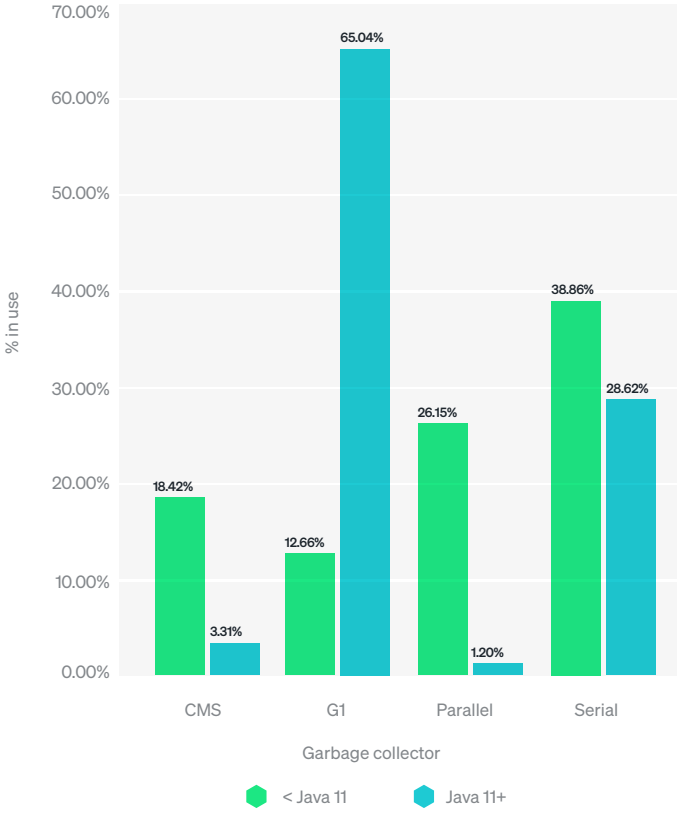
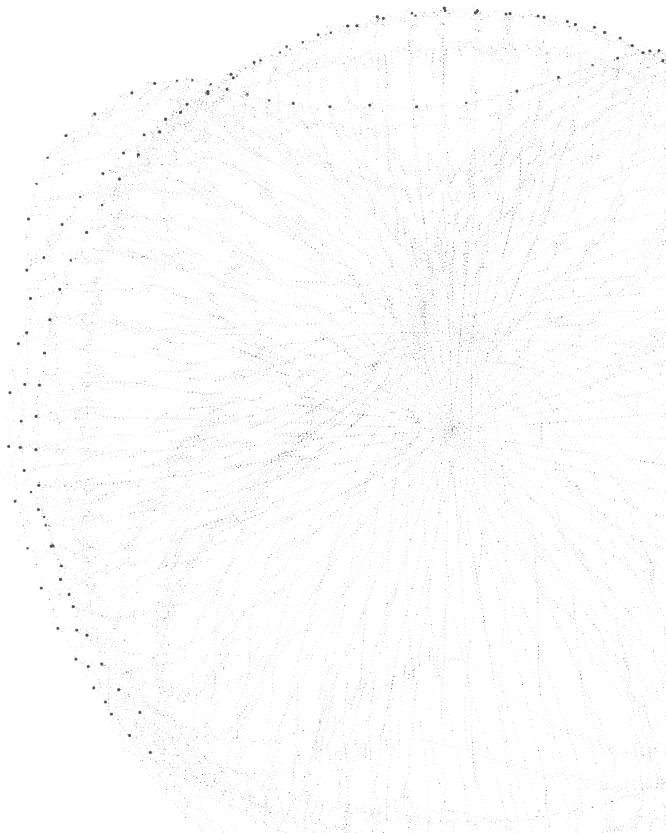


Figure 06. Percentage of garbage collectors in use by Java version



Methodology

The data for this report was drawn entirely from applications reporting to New Relic in January 2023 and does not provide a global picture of Java usage. New Relic anonymized and deliberately coarse-grained the appropriate data to give general overviews of the Java ecosystem. Any detailed information that could help attackers and other malicious parties was deliberately excluded from this report.

Start monitoring your Java data with New Relic today.

Install the Java Quickstart

About New Relic

As a leader in observability, New Relic empowers engineers with a data-driven approach to planning, building, deploying, and running great software. New Relic delivers the only unified data platform with all telemetry—metrics, events, logs, and traces—paired with powerful full-stack analysis tools to help engineers do their best work with data, not opinion.

Delivered through the industry's first usage-based pricing that's intuitive and predictable, New Relic gives engineers more value for their money by helping improve planning cycle times, change failure rates, release frequency, and mean time to resolution (MTTR). This helps the world's leading brands and hyper-growth startups to improve uptime, reliability, and operational efficiency and deliver exceptional customer experiences that fuel innovation and growth.

new relic.