new relic.

```
avg duration: 6.240ms | error rate:100.00% - 2
last 30 minutes
@app.route("/external/error")
def external_error():
    req = requests.get("http://localhost:8000/error")
    req.raise_for_status()

    return req.tex
```

2024 State of the Java Ecosystem

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

Contents

Overview

New Relic has been monitoring the Java ecosystem for the past few years to uncover shifts in how developers are using it, including whether they're adopting Java 21 at a faster rate than other Java versions, vendor- and community-supported Java Developer Kit (JDK) trends, and the types of Java-related questions asked by developers using the New Relic generative AI (GenAI) observability assistant.

This annual report provides context and insights into the current state of the Java ecosystem based on data from hundreds of thousands of applications reporting to New Relic monthly.

The 2024 report examines the following topics:

- The most-used Java versions in production
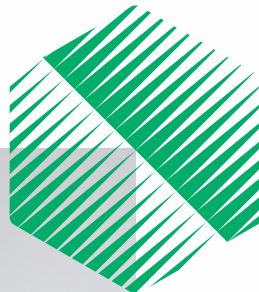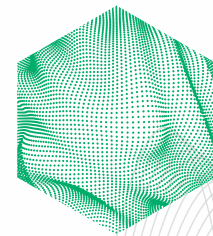- The most popular JDK vendors
- The use of compute and memory in Java applications
- The most popular Java frameworks and libraries for logging, encryption, and databases
- The most common types of Java-related questions and requests asked by developers

New Java versions being adopted faster

First, let's look at the most-used Java versions in production.

At one time, Oracle only released new versions with major updates and changes in the JDK. Now, Oracle releases new Java versions every six months—typically in March and September—with each release containing a handful of new features and bug fixes. Every two years, Oracle introduces a new Java long-term-support (LTS) version with updates to help improve stability, security, and performance, which developers often cite as one of the most important factors for upgrading Java versions.

Oracle released Java 21 in September 2023, which marked a significant milestone for Java with notable improvements to preview features like virtual threads and upgraded libraries, as well as advancements to syntax that put Java on par with many more modern languages.

In the six months after the release of Java 21, 1.4% of applications monitored by New Relic were using it. To put this into perspective, in the six months after Java 17 was introduced, only 0.37% of applications were using it, which is 287% fewer.

In addition, the adoption rate of Java 17 far exceeded what the developer world saw when Java 11 was introduced. About a tenth (9%) of applications were using Java 17 in production in 2023, and now 35% of applications are using Java 17, representing a nearly 300% growth rate in one year. It took years for Java 11 to reach anywhere near that level.

Less than 2% of applications were using Java non-LTS versions, which makes sense since they're usually not used in production.

35%

of apps are using Java 17
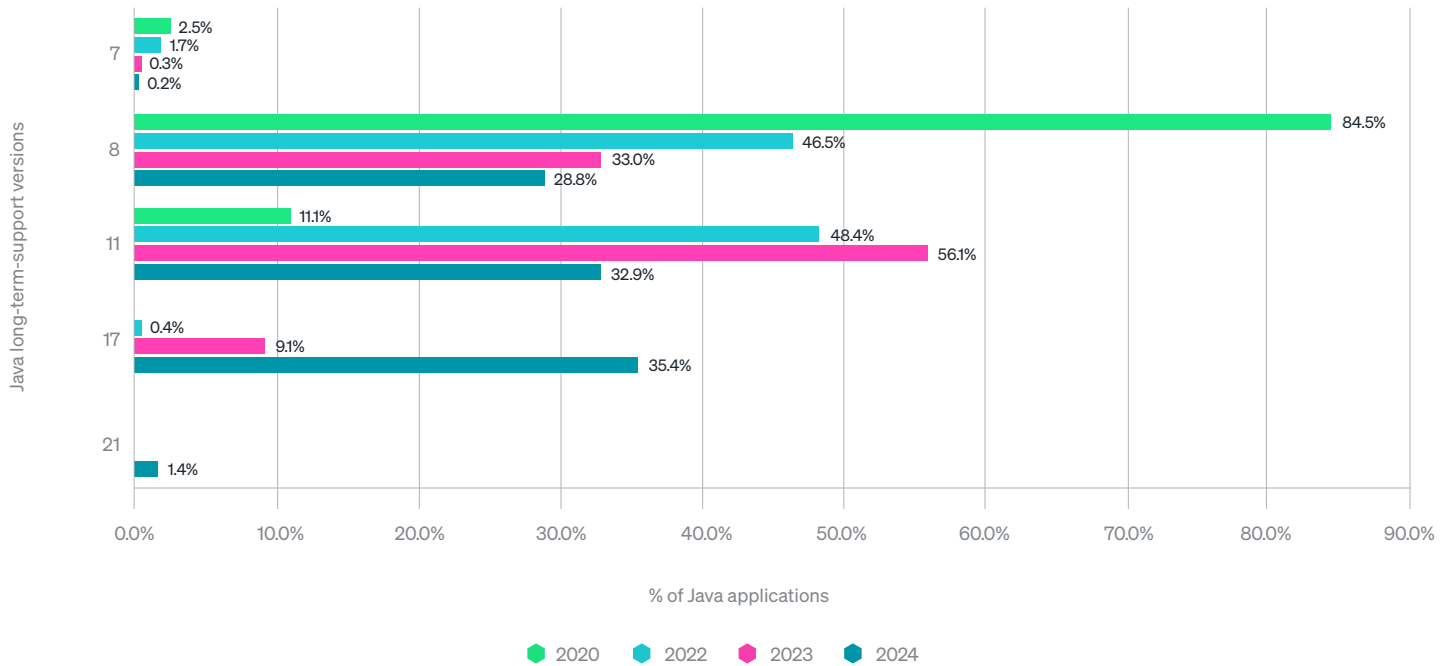
Java long-term-support version adoption by year
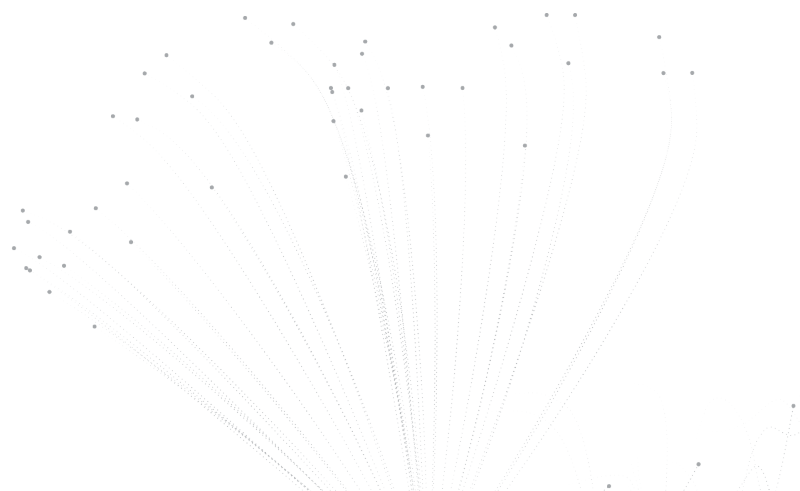
Eclipse Adoptium rising in popularity amongst JDK vendors

Now let's look at the most popular JDK vendors.

Java used to be closed-source or proprietary, which meant developers could only download the JDK directly from Sun Microsystems. However, roughly 10 years after Sun Microsystems released the first version of Java, Oracle released the first open-source Java version. OpenJDK maintains it and allows developer communities and vendors like Microsoft and Amazon to maintain other versions of the JDK.

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), and we've seen a steady decline year-over-year (YoY) ever since then. While Oracle retained the top spot in 2022 (34%), it slipped to 29% in 2023, and it's now at 21%—which represents a 28% decrease in one year.
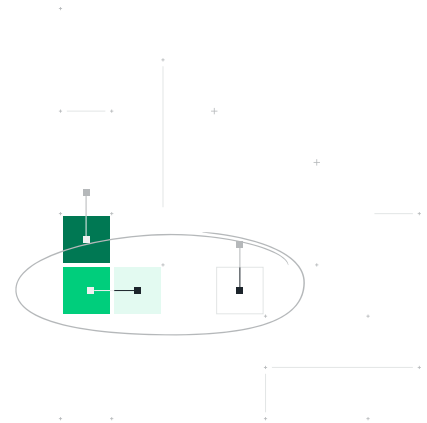
The use of Amazon increased to 31% of the market in 2023 (up from 2.2% in 2020 and 22% in 2022), but has dropped to 18% in 2024, which represents a 43% decrease YoY.

The rising star this year is Eclipse Adoptium, adoption of which rose 50% YoY from 12% to 18%. Because Eclipse Adoptium is community-managed, this JDK tends to be updated more frequently than the Oracle and Amazon JDKs.

18%

are using the Eclipse Adoptium JDK
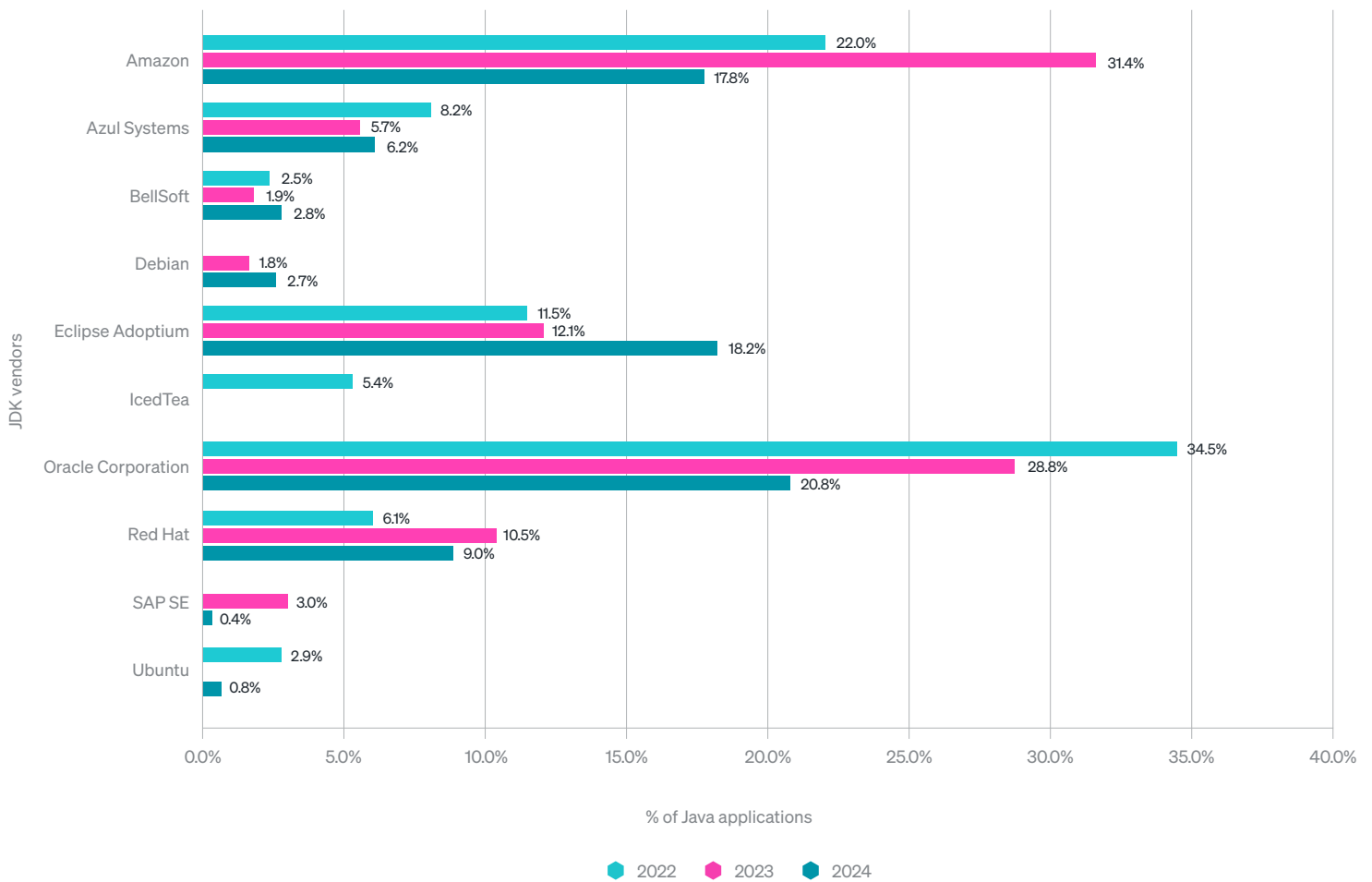
The most popular JDK vendors by year

The most common Java application configurations

Next, we look at the use of garbage collectors, compute, and memory in Java applications.

Garbage in, garbage out

Java garbage collectors (GCs) are memory management components used to prevent memory leaks, optimize memory usage, and ensure the overall performance and stability of Java applications.

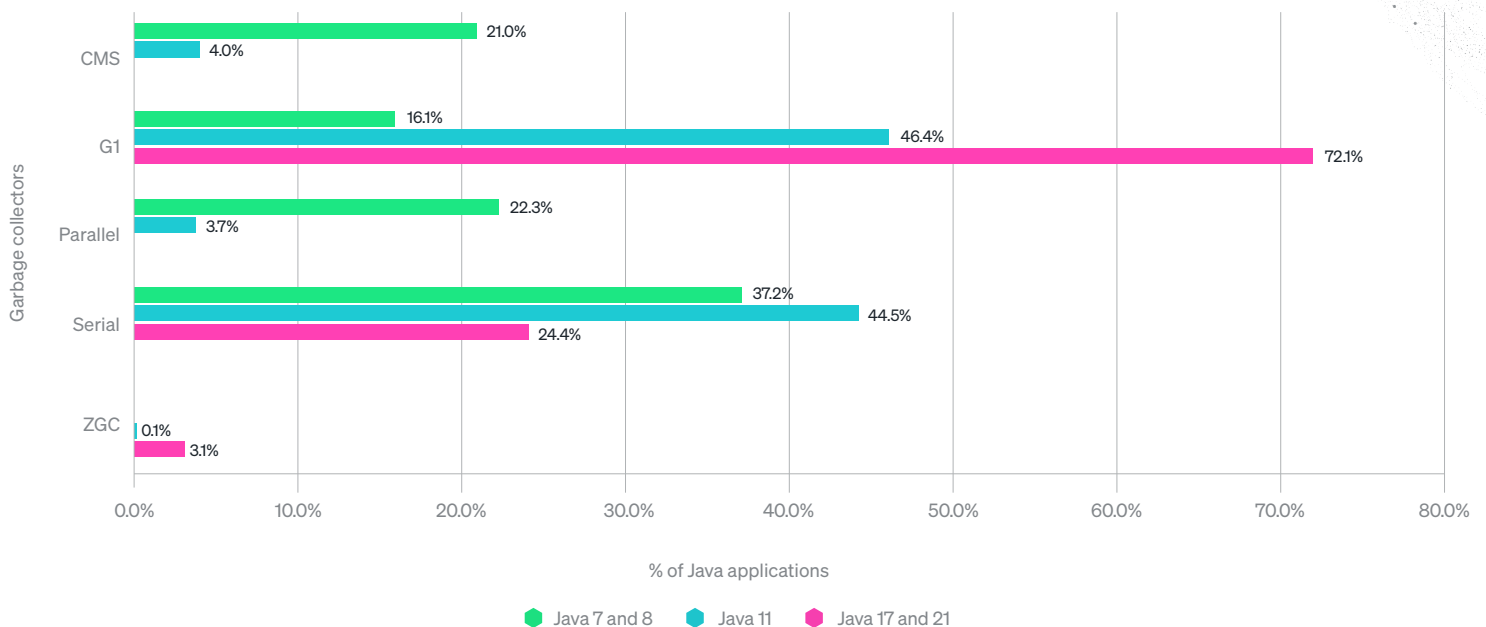Some Java versions only support specific GCs, so which GC developers use is partially dependent on their Java version.

Since Java 11, the Garbage-First (G1) GC has been the default. Being the default collector might explain why 43% of customers are using it and why there was a big jump in usage for Java 11, 17, and 21 compared to Java 7 and 8. In addition, 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 developers.

The second most popular GC is Serial (37%), which is ideal for applications or systems that run on a single processor or where a high number of Java virtual machines (JVMs) are running on the same machine. It also has lower CPU and memory overhead compared to more complex GCs, making it suitable for resource-constrained environments.

43%

are using the G1 garbage collector

Garbage collectors — % of Java applications

Garbage collectors	Java 7 and 8	Java 11	Java 17 and 21
CMS	21.0%	4.0%	
G1	16.1%	46.4%	72.1%
Parallel	22.3%	3.7%	
Serial	37.2%	44.5%	24.4%
ZGC		0.1%	3.1%

% of Java applications

● Java 7 and 8   ● Java 11   ● Java 17 and 21

Garbage collectors used by Java LTS version

Compute and memory settings

The New Relic data shows a 18% YoY increase in applications running with four or fewer cores.

The drive to run smaller makes a lot of sense in cloud environments where people are often deploying containers. But this trend can pose unexpected issues for some applications. In particular, many of the concurrent benefits from the default G1 GC on recent JVMs vanish when running with fewer than two cores. All those single-core instances may as well be using the Serial collector—and paying the performance cost of that.

68%

are using 1–4 cores
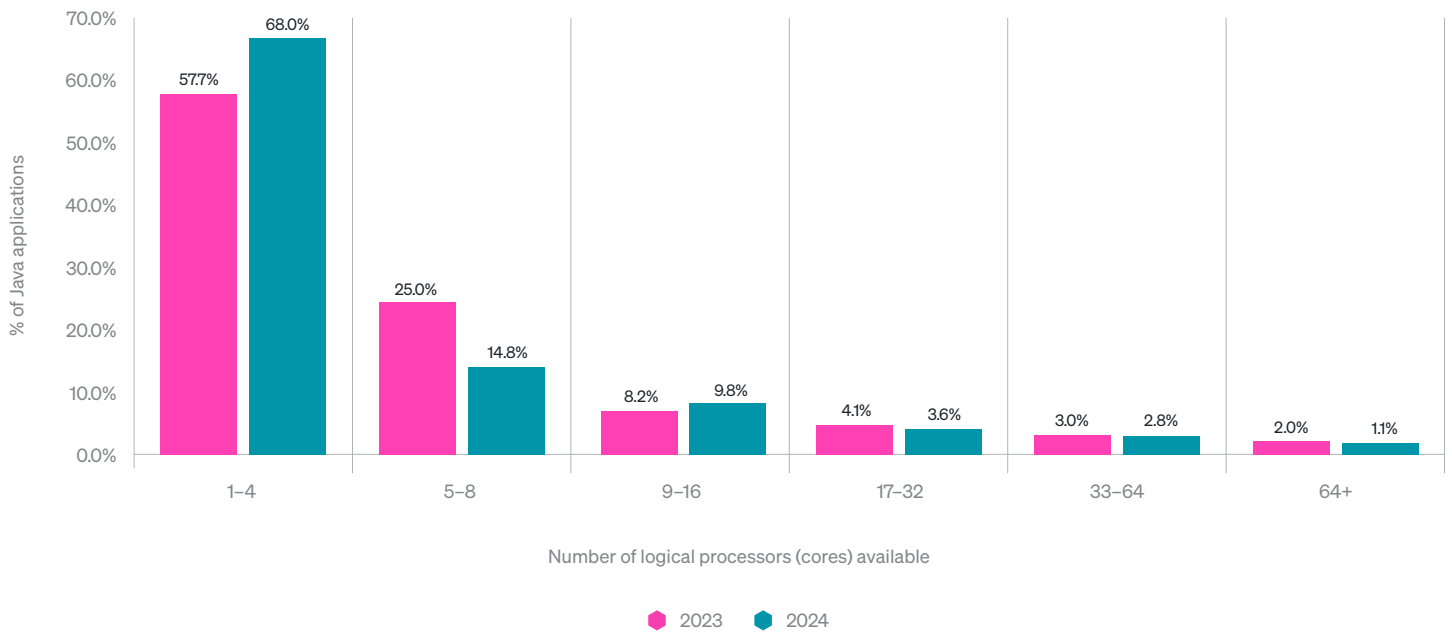
Logical processors (cores) available by Java applications in 2023 and 2024

When looking at JVM memory settings, 32% of Java applications use
1 GB or less and 68% use more than 1 GB. This equates to a 15% increase in
applications using more than 1 GB of memory YoY.

JVM memory settings heap sizes

Heap size	2023	2024
<=512 MB	21.9%	15.2%
512 MB but <=1 GB	19.0%	16.6%
1 GB but <=2 GB	22.5%	27.6%
2 GB but <=3 GB	9.0%	7.8%
3 GB but <=4 GB	6.6%	9.0%
4 GB but <=8 GB	9.3%	10.7%
8 GB but <=16 GB	6.5%	9.6%
16 GB but <=32 GB	4.2%	2.7%
32 GB but <=64 GB	0.8%	0.7%
64 GB	0.3%	0.2%

% of Java applications

● 2023   ● 2024
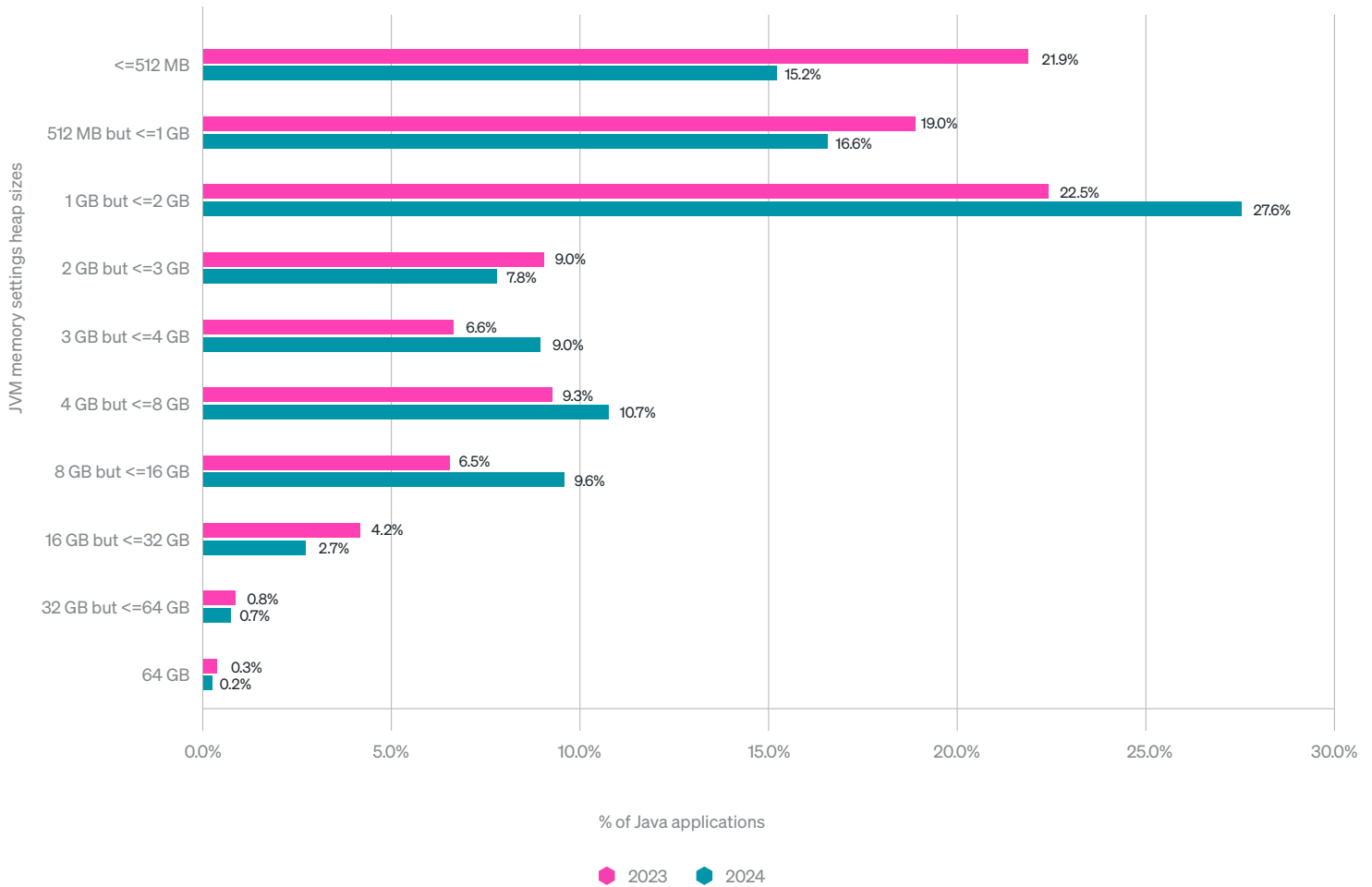
JVM memory settings heap size by year
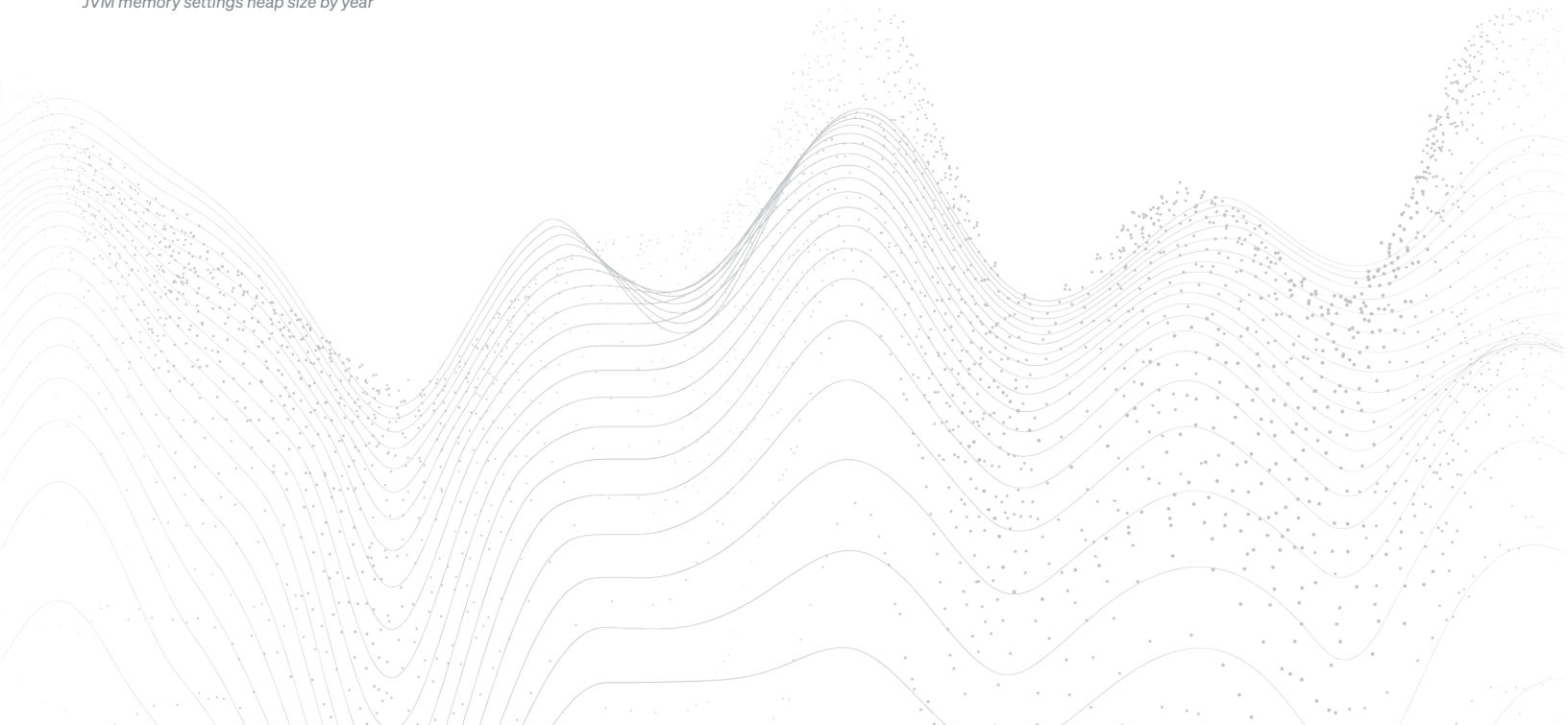
Popular Java frameworks and libraries for logging, encryption, and database

Every Java developer uses frameworks and libraries to streamline the application development process by using pre-written code for common tasks like web development and database connectivity. Libraries are used to enhance the functionality of an application. Here we examine the most popular frameworks and libraries for logging, encryption, and database connectivity.

Log4j is the most popular logging framework for Java applications

Any software application or system can have bugs and issues in the testing or production environments, and developers use logging tools to troubleshoot issues and fixes. However, logging is only useful when it provides the required information from the log messages without adversely impacting the application's performance. Software developers use various logging frameworks to solve these issues. In fact, 91% of the Java applications reporting to New Relic use logging frameworks.
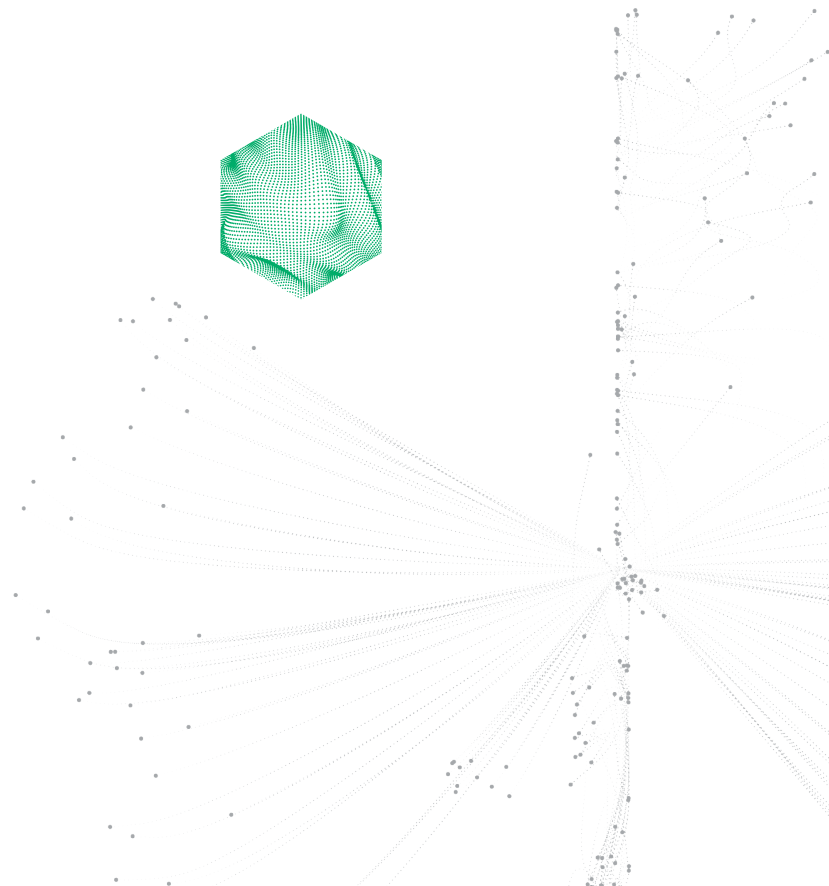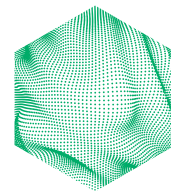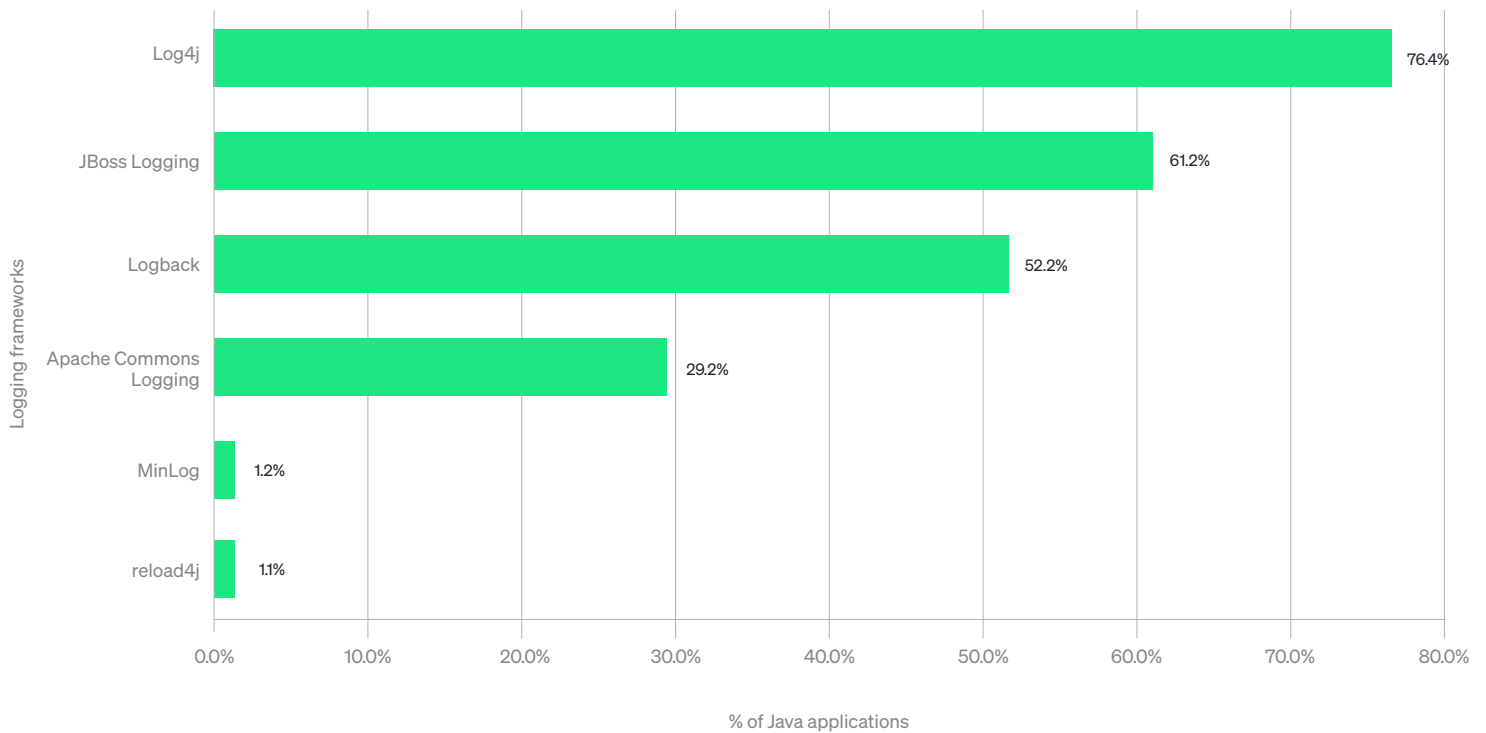
The most-used logging framework was Log4j with 76% of Java applications using it, followed by JBoss Logging (61%) and Logback (52%).

Also, most (83%) Java developers rely on SLF4j, which is a framework that acts as an abstraction for other types of Java logging frameworks. SLF4j enables software developers to use the logging framework of their choice and enables applications to switch to any Java logging framework interchangeably without impacting its implementations or doing any changes. Due to this functionality, SLF4j makes applications independent of logging frameworks, providing more flexibility and portability for logging across any part of the system. It also means that Java applications can use more than one logging framework.

76%

are using the Log4j logging framework

Logging frameworks	% of Java applications
Log4j	76.4%
JBoss Logging	61.2%
Logback	52.2%
Apache Commons Logging	29.2%
MinLog	1.2%
reload4j	1.1%

The most popular logging frameworks for Java applications
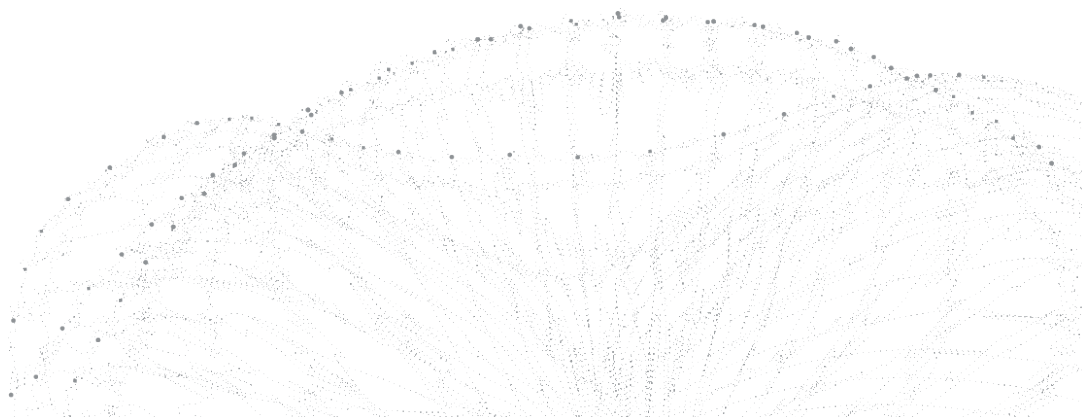
Bouncy Castle is the most popular encryption library for Java applications

More than one-third (41%) of the Java applications reporting to New Relic use encryption libraries: 17% use Bouncy Castle, 16% use Spring Security, and 6% use Jasypt. Bouncy Castle has been a popular library for encryption for many years due to its set of cipher suites and utilities.

While only 0.09% developers use the Amazon Corretto Crypto Provider (ACCP) library, we expect more applications to use it in the near future as companies and developers look to consolidate vendors and because it typically provides better performance.

17%

are using the Bouncy Castle encryption library

Bouncy Castle   17.1%
Spring Security   15.8%
Jasypt   6.4%
Tink   0.5%
Apache Commons Crypto   0.5%
Conscrypt   0.3%
Scrypt   0.1%
Amazon Corretto Crypto Provider 0.1%

Encryption libraries

0.0%   2.0%   4.0%   6.0%   8.0%   10.0%   12.0%   14.0%   16.0%   18.0%

% of Java applications
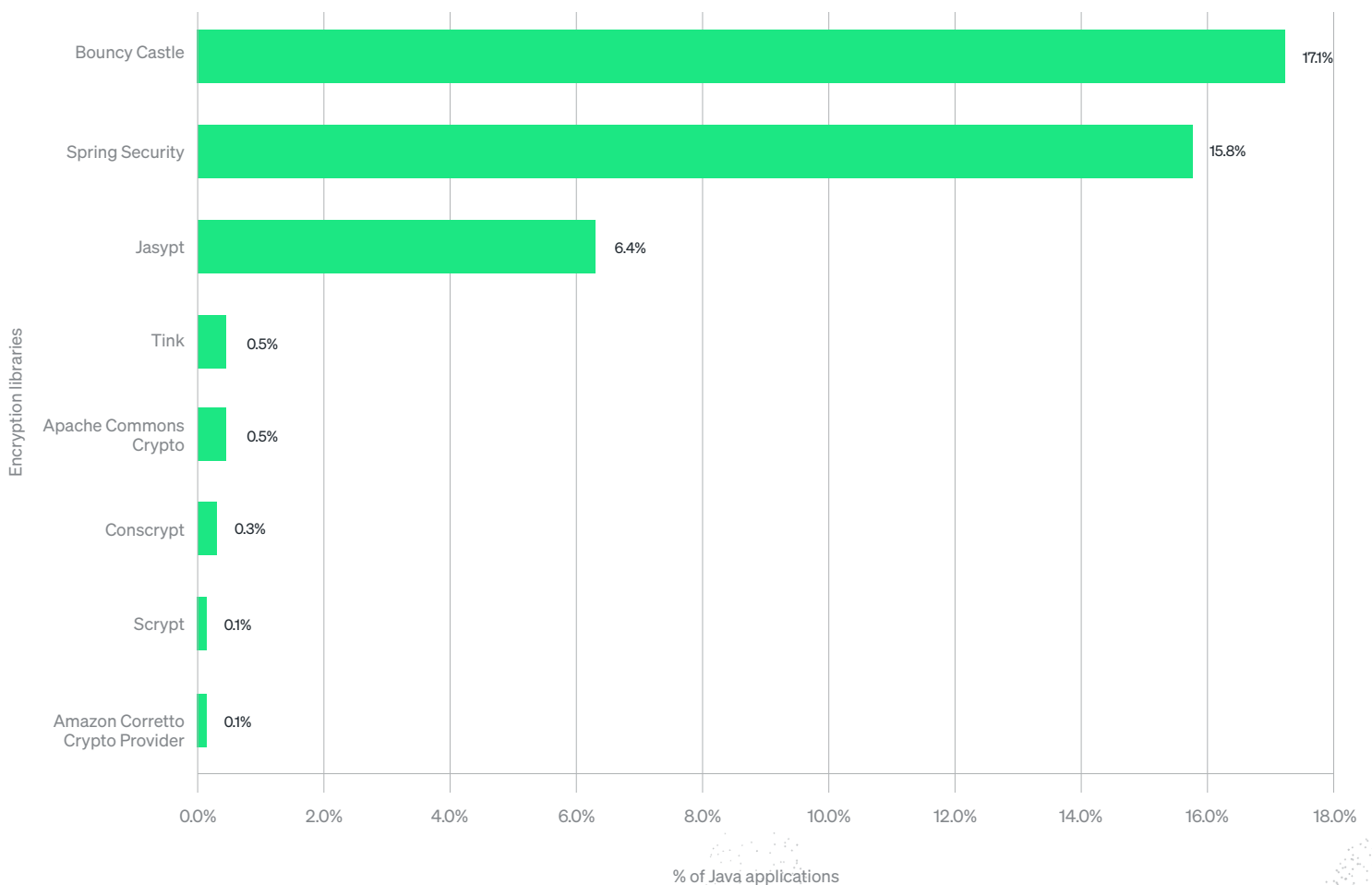
The most popular encryption libraries for Java applications

Oracle is the most popular database system for Java applications

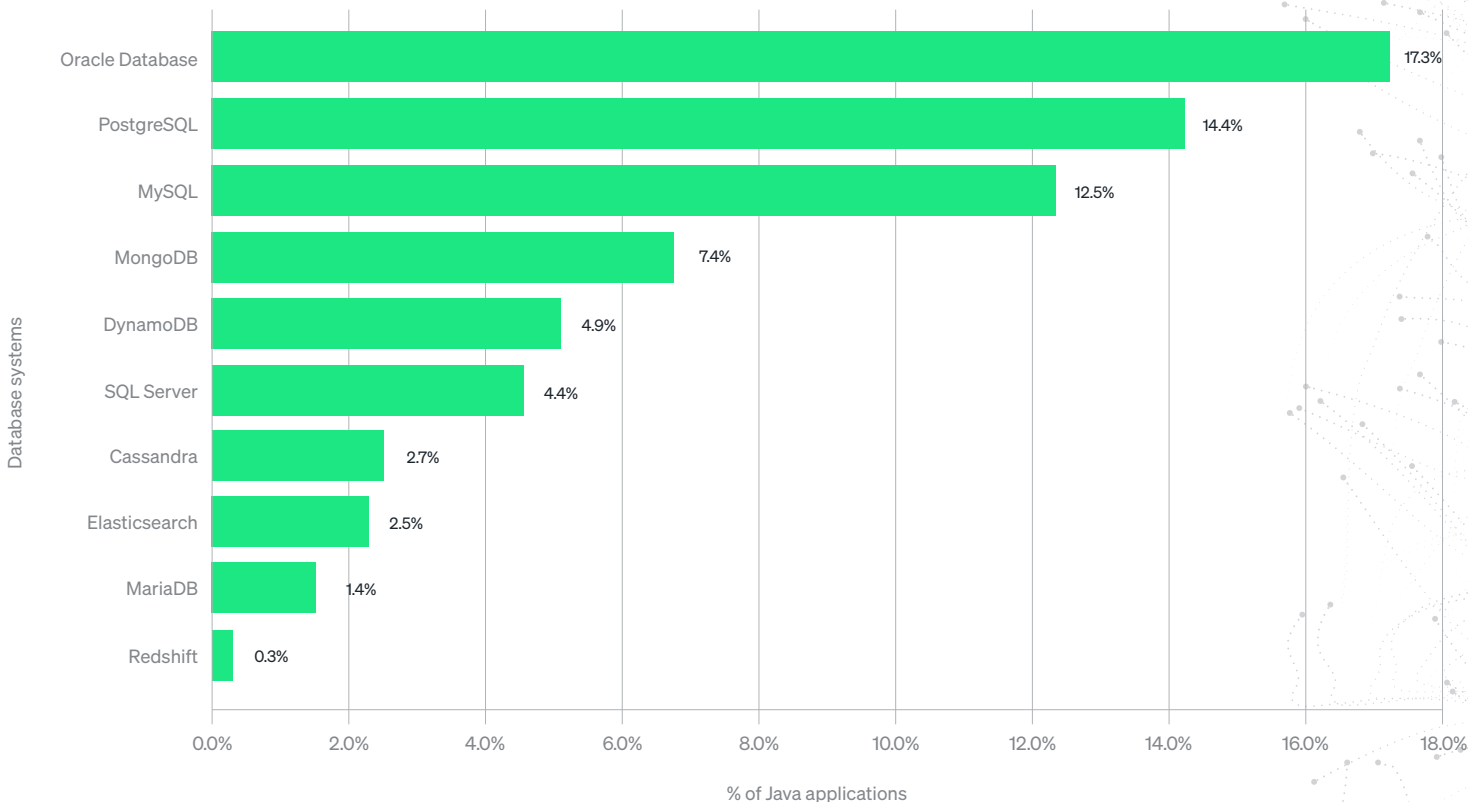When looking at databases, Oracle Database is the most widely used, with 17% of Java applications reporting to New Relic using it. Oracle Database is known for its scalability and ability to manage large amounts of data quickly and efficiently. As such, it tends to be the preferred database system for enterprises. Additionally, it offers customer support and a robust set of tools.

The second most popular database system is PostgreSQL, with 14% of Java applications reporting to New Relic using it. While Oracle Database is managed by Oracle directly and available through a license, PostgreSQL is an open-source database that's free to use and preferred for managing read-write operations and complex queries.

MySQL comes in third, with 13% of Java applications using it. MySQL is also an open-source database. It offers fewer features than Oracle Database and PostgreSQL, which makes it more stable and faster at processing, especially when processing read-only queries.

PostgresSQL has gained popularity in the last year because it's more compliant with the SQL specs and it supports more features than MySQL. MySQL is still a top database system, but we can see that its popularity is decreasing while PostgresSQL's popularity is growing.

17%

are using the Oracle Database

Database systems	% of Java applications
Oracle Database	17.3%
PostgreSQL	14.4%
MySQL	12.5%
MongoDB	7.4%
DynamoDB	4.9%
SQL Server	4.4%
Cassandra	2.7%
Elasticsearch	2.5%
MariaDB	1.4%
Redshift	0.3%

The most popular database systems for Java applications

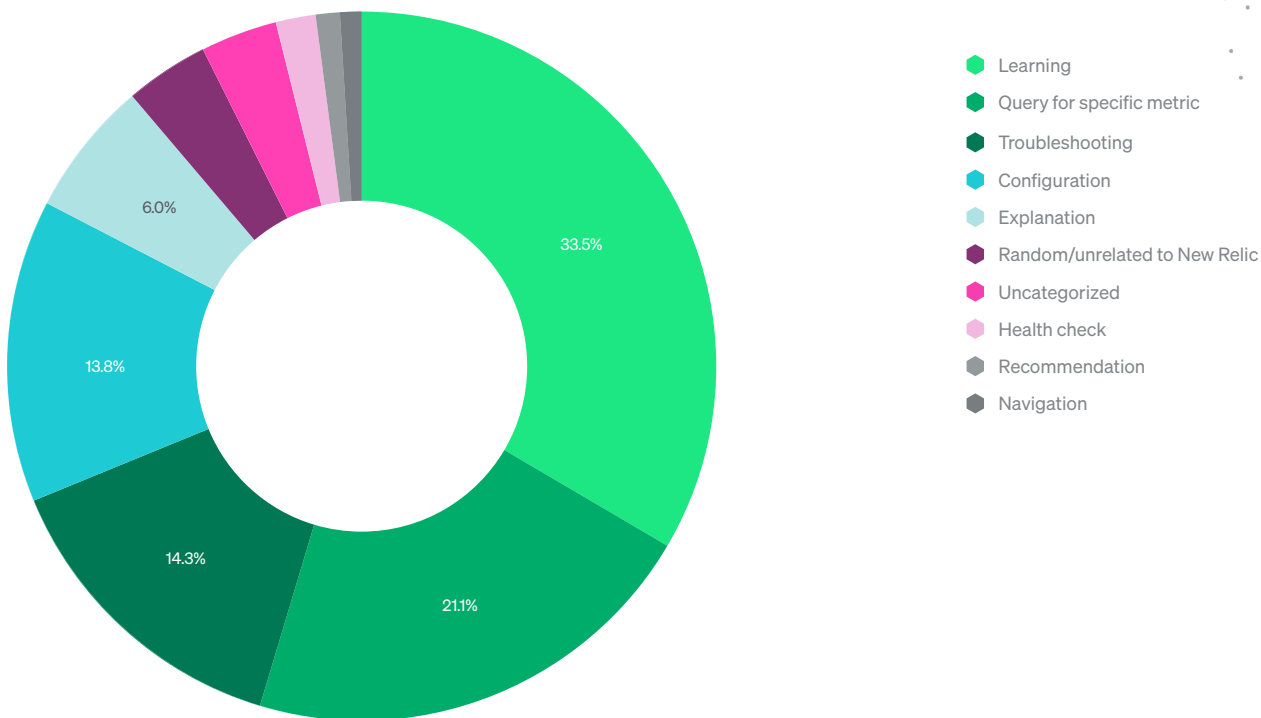# The most common types of Java-related questions and requests asked by developers

We also looked at what kind of Java-related questions and requests developers are asking via the New Relic AI GenAI observability assistant. Since January 2024, the data shows that 34% of the 483 Java-related questions were how-to type (learning) questions, 21% were related to querying for a specific metric, 14% were about configuration, and 14% were about troubleshooting.

Sample questions and requests include:

- How do I log API calls in my Java Spring Boot?
- If the Java framework changed from Tomcat to JBoss, do I need to reconfigure the agent?
- Create a dashboard for container memory utilization vs. JVM heap utilization.
- This monitored application uses the Vaadin framework in Java. It has high memory consumption. What could it be?
- In JVM, what does G1 Eden Space heap usage mean?

34%

of Java-related questions and requests are about how to do something

Legend:
- Learning
- Query for specific metric
- Troubleshooting
- Configuration
- Explanation
- Random/unrelated to New Relic
- Uncategorized
- Health check
- Recommendation
- Navigation

Pie chart values: 33.5%, 21.1%, 14.3%, 13.8%, 6.0%

The most common types of Java-related questions and requests asked by developers using New Relic AI

Methodology

This report is based on data gathered from hundreds of thousands of applications reporting to New Relic that provide performance information. Therefore, it doesn't provide a global picture of Java usage. All data was collected in 2024.

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 hypergrowth startups to improve uptime, reliability, and operational efficiency and deliver exceptional customer experiences that fuel innovation and growth.

new relic.