perforce

REPORT

# 2025 Java Developer Productivity Report

## Java Development Trends + Analysis

perforce
JRebel

# Contents

Rod Cope
CTO, Perforce Software

# Investing in Java Productivity Is More Important Than Ever

Dear Colleagues,

Welcome to the 2025 Java Developer Productivity Report. If you're reading this report for the first time, perhaps you found it in a desperate search to maximize your Java development resources as your business looks to do more with less.

Or perhaps you've been following the Java productivity benchmarks we've set each of the past dozen years. Welcome, or welcome back. Either way, you arrive as development leaders are at a turning point.

This year, we surveyed 731 developers, team leads, managers, and executives who work in Java about their current Java development environments, plans for their team's future, productivity challenges, and more. Among their responses, some key themes rose to the top.

Plans to add Java developer headcount are largely stagnant at only 51% as businesses wait to see what happens as economic headwinds continue, and the rise of AI converge. But still, your development teams are being tasked with adding more features, improving testing, accelerating time to market, etc. At the same time, 53% of respondents said that long redeploys and insufficient developer tools are their biggest barriers to Java development productivity.

How do you find the whitespace to succeed in an increasingly volatile business environment? While everyone is looking to AI to replace or reduce the need for developers, the answer may be simpler than that. Java productivity tools can help your developers save hours that can then be applied to adding true business value.

While economic uncertainty remains, the value of investing in Java remains steadfast as the language celebrates its 30th birthday this year. In a development world where languages rise and fall in popularity rapidly, Java has established itself as a stable backbone of enterprise applications across industries.

We hope the data presented in this report helps make your application development decisions easier by providing clear benchmarks on Java trends — as well as insights on how innovative companies can gain a competitive edge.

Enjoy the report,

Rod Cope, CTO, Perforce Software

# Investments in Java

In this section, we cover year-over-year spending trends for developer headcount and developer tools, and what those shifts mean for the broader Java ecosystem.

## Developer Headcount + Tool Budget

In 2025, 51% of respondents said their companies plan to add Java developers in the coming year, 16% did not plan to add any developer headcount, and 32% were not sure. Similarly, respondents were asked if their companies planned to increase their developer tool budget for 2025: 34% said yes, while 21% said there would be no tool budget increase and 45% were unsure.

That's a sharp decline from 2024 results for the same question, where 60% of respondents said they had plans to add Java developers in the coming year, and 42% said they intended to increase their developer tool budget.

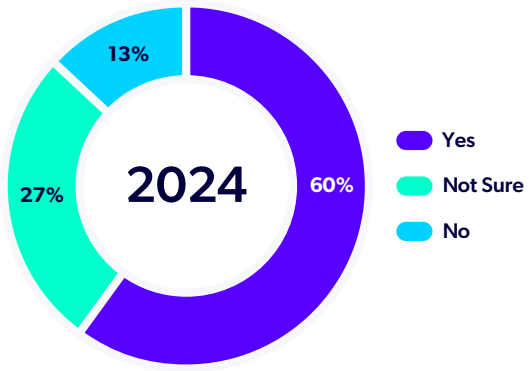
The outlook is brighter for companies with 1,000+ employees, however. Of that subset, 58% of respondents were planning to add Java developers in the coming year, with only 9% responding no and an additional 33% unsure of their companies' developer headcount plans. The situation was similar for developer tool budget, where 36% of respondents anticipated spending increases from their companies, 13% expected stagnant developer tool budgets, and 51% were unsure.

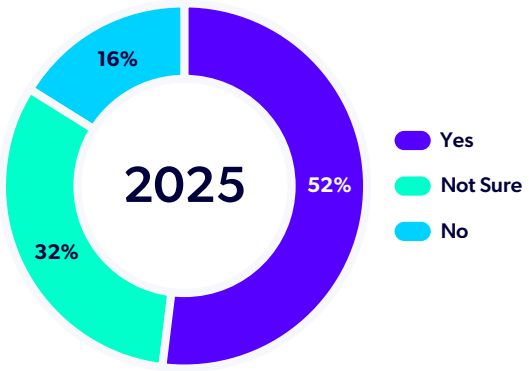
**KEY TAKEAWAYS:**

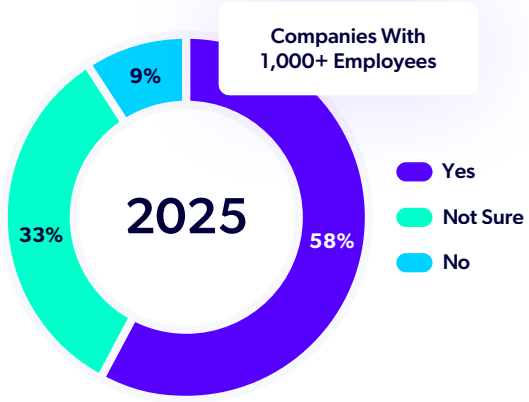
Long-forecasted economic headwinds have arrived in force for nearly all industries. While it's clear development spending has slowed across the board, enterprises are still carving out budgets for the people and tools necessary to maintain their mission-critical business applications. Of respondents from companies with 1,000+ employees, 58% plan to add developer headcount in 2025 — only a 2% decrease from 2024.

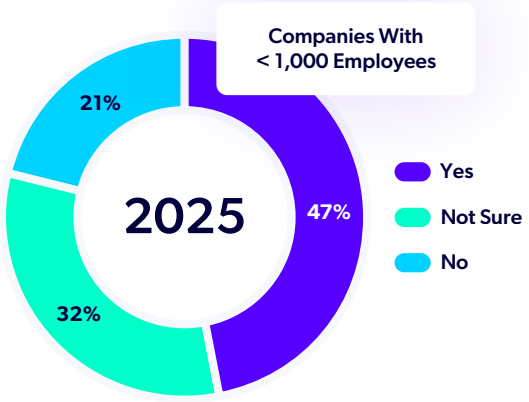
Regardless of company size, development teams are now being tasked with doing more with less — and in more onerous development environments. Some companies may also be facing hiring freezes coming from above. AI is the obvious answer, but in many cases this technology is not mature enough to truly save development teams time in a meaningful way. In these cases especially, productivity-focused development tools can help teams carve out whitespace to focus on mission-critical tasks.

# Does your company plan to add Java developers this year?

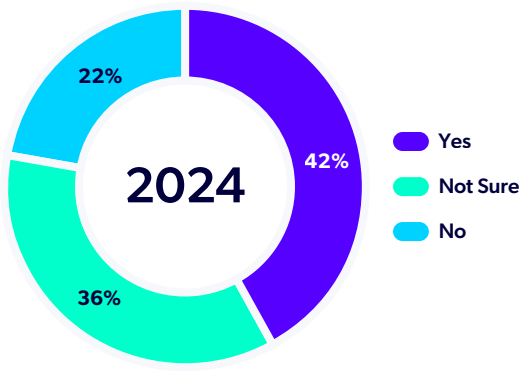


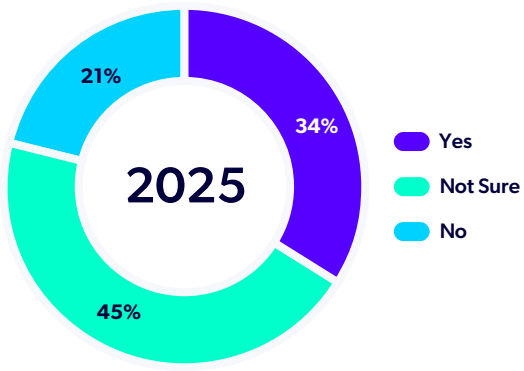



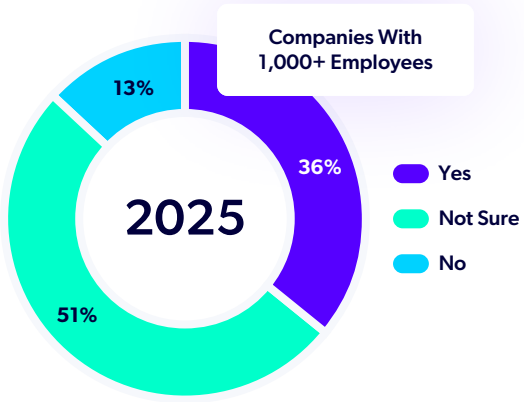



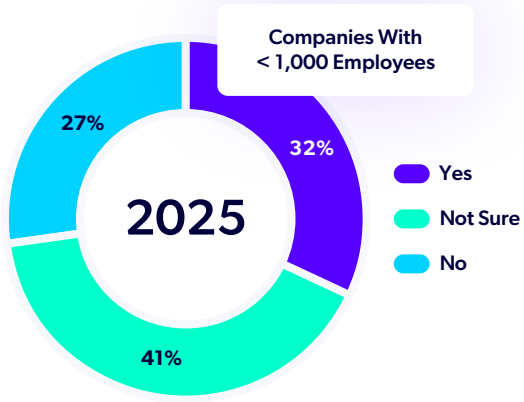



# Does your company plan to increase its Java development tool budget this year?



# Java Language + Technology Trends

In this section, discover what technologies are trending for Java development environments — and the factors that influence these tech stack decisions. While some results remained status quo, 2025 saw a definitive shift for IDE preferences and JDK adoption.

## JDK Adoption Shifts Amidst Changing Support Landscape

Respondents were asked which JDK programming language or languages they were using in their primary business application, and the results revealed a wholesale shift to JDK versions under long-term support (LTS).

61% of respondents said they use Java 17, 45% reported using Java 21, 5% reported using Java 22, and 9% reported using Java 23. Meanwhile, 72% of respondents are using Oracle distributions that are no longer supported, including 32% using Java 11, 35% using Java 8, and 5% using Java 7 or older versions. Additionally, respondents also reported using Kotlin (10%), Groovy (7%), and Scala (3%). These numbers are a stark contrast to results from two years ago, when 52% of respondents were using Java 11 or an older Java version.

Respondents were also asked their primary motivation(s) for upgrading JDK versions. LTS (64%), security (54%), performance (49%), new features (45%), and compliance (31%) topped the list. Among respondents from enterprise companies, however, those priorities were flipped, with security (68%), long-term support (65%), and performance (52%) topping the list.

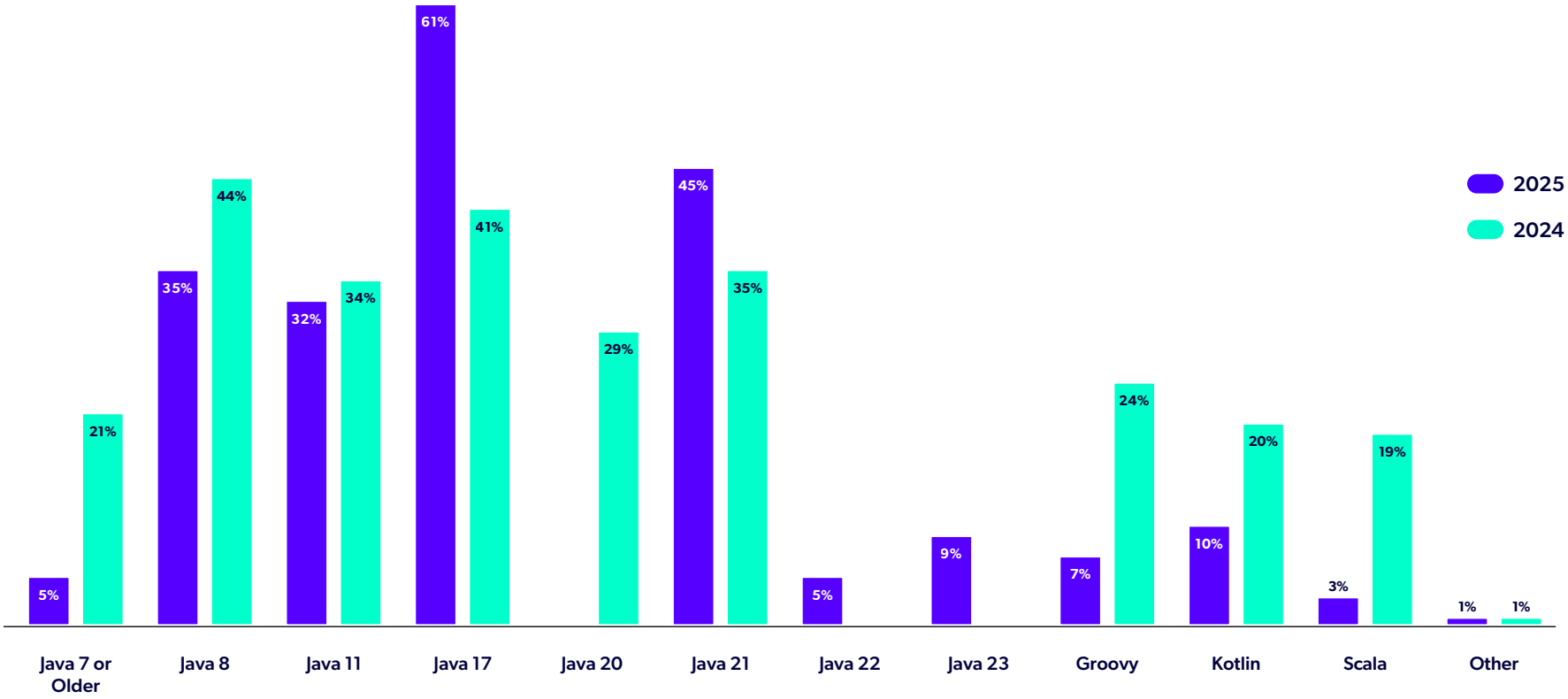
Results were mixed as to what JRE/JDK distribution respondents were using for their primary business application. While 45% of respondents are using Oracle JDK, 35% are using a generic OpenJDK distribution. Niche options like AdoptOpenJDK (24%), Amazon Corretto (23%), Azul Zulu (12%), GraalVM (10%), and OpenLogic OpenJDK (5%) trailed behind.

**KEY TAKEAWAYS:**

Upgrading JDK versions has traditionally been viewed as time-intensive and disruptive to business continuity. For that reason, many companies have stuck with Java 8 (or even older versions) for years. But the combination of Oracle's shift to releasing LTS versions every two years and changing pricing models seem to have compelled companies to upgrade to Java 17 and Java 21. Watch for that trend to continue as Java 25 is released in September 2025.

Respondents' preferences for security, compliance, and performance over new features could signal that the market has reached a saturation point for features, and developers are now prioritizing stability.

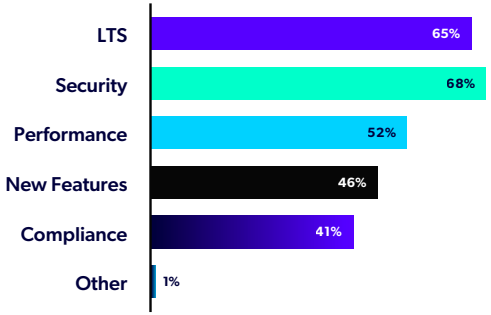
# What's your JDK programming language for your main application?



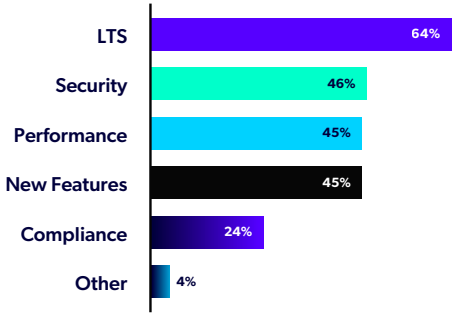

*Respondents could select all that applied.

## Why do you upgrade JRE/JDK versions?

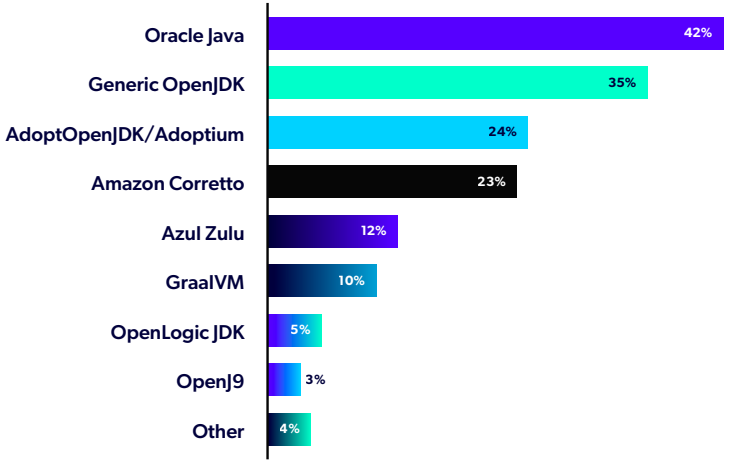
### 1,000+ Employees




### <1,000 Employees




## What's your primary JRE/JDK distribution?




*Respondents could select all that applied for all charts above.

# IDE Insights

Respondents were also asked to share which Java IDEs they were using. As expected, IntelliJ IDEA topped the charts with 84% of respondents. What's more interesting were the results behind: VS Code overtook for the second position (31%), with Eclipse slipping to third (28%). Other responses included, browser-based IDEs (1%), and JetBrains Fleet (3%).

Moreover, 42% of respondents are using more than one IDE in their Java development practice. Among respondents using IntelliJ IDEA as their primary IDE, 68% also reported using VS Code as a secondary IDE.

Development environments also look a little bit different for those working in remote, containerized, and cloud-based development environments: 86% use IntelliJ IDEA as their primary IDE, 73% use Tomcat as their application server, 71% use microservices as their primary application architecture, 64% use Java 17, and 47% use Java 21.

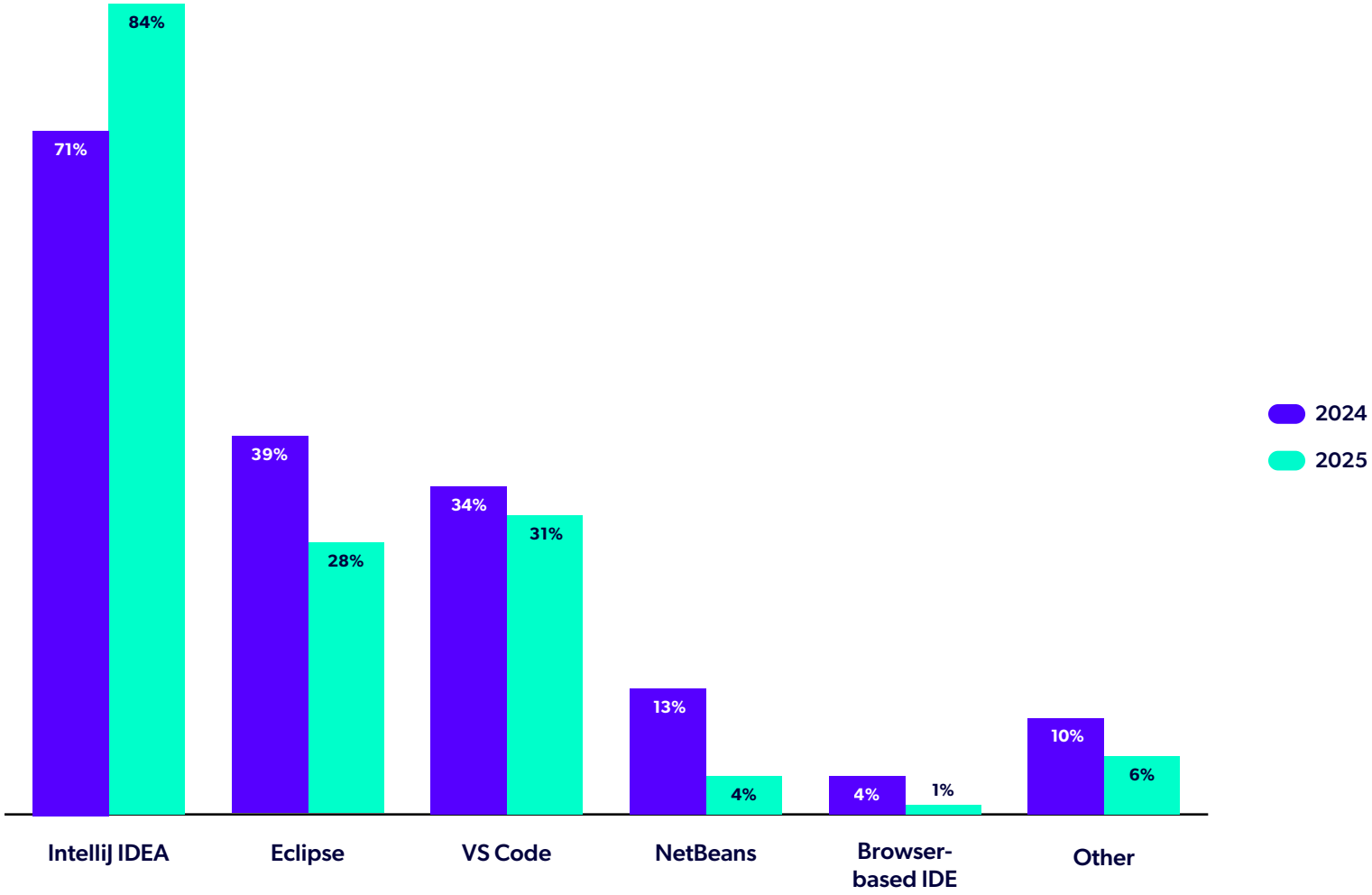
**KEY TAKEAWAYS:**

The IDE landscape is rapidly changing. Microsoft first released VS Code as a free solution in 2015, but it didn't take hold with the Java community until 2020. Fleet, an emergent code editor from JetBrains, is still in Beta but already gaining fans in the Java community.

These new developments are ballasted against the rest of the Java IDE landscape, where IntelliJ IDEA and Eclipse have been relied upon as stable backbones for Java development for decades.

Still, many developers are turning to more than one Java IDE to leverage specialty features, including debugging tools and AI assistants. Today, most third-party tools first provide features for VS Code first, then follow with support for IntelliJ IDEA and other Jetbrains IDEs. For example, of respondents who use GitHub CoPilot, 89% use IntelliJ IDEA as an IDE vs. 40% for VS Code. Whether this pick-and-choose model is a net gain or further contributes to productivity losses remains to be seen, but it's understandable why developers may choose between IDEs given their task at hand.

# What IDEs do you use in your Java development?




*Respondents could select all that applied.

# Application Server

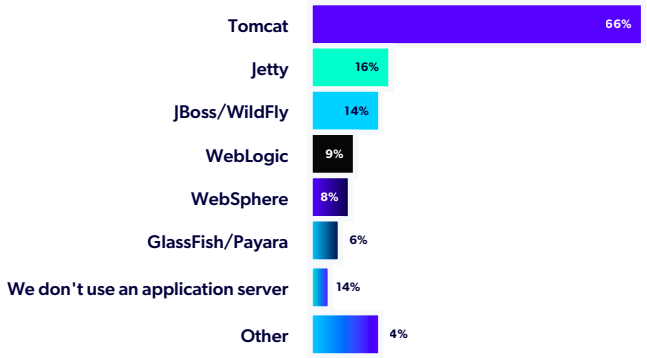
Respondents were asked what application server they use. Tomcat was the clear winner at 70%, with Jetty (18%), JBoss/WildFly (14%), WebLogic (9%), and WebSphere (8%) trailing. Interestingly, 13% of respondents reported they don't use an application server.

Among respondents from enterprises, those preferences are even more polarized, with 76% using Tomcat, 19% using Jetty, and 17% using JBoss/WildFly.

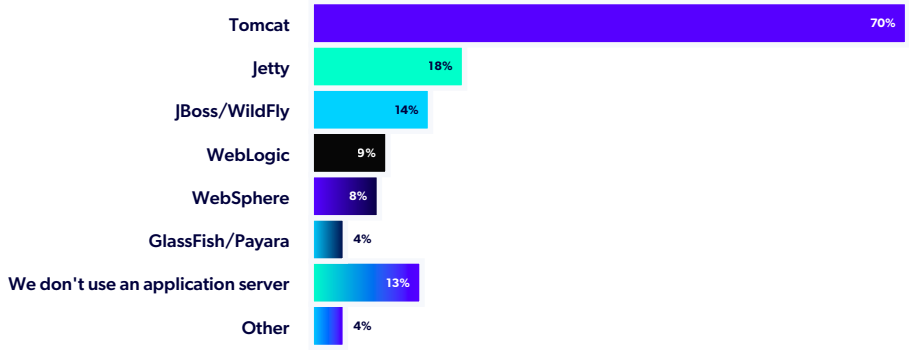
**KEY TAKEAWAYS:**

While other aspects of business' Java development environments are rapidly shifting, application server preferences remain steady. While application server options like Jetty, JBoss/Wildfly, WebLogic, and WebSphere have ebbed and flowed in popularity, Tomcat has remained the top choice of enterprises and smaller businesses alike, in part because it's lighter weight that JBoss/Wildfly and WebLogic, yet sufficient for most development environments.

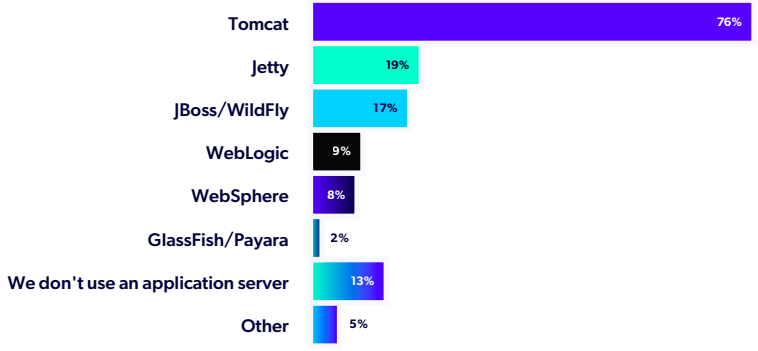
# What application server are you using?




## 1,000+ Employees




## <1,000 Employees




*Respondents could select all that applied for all charts above.

# Remote Development + Cloud

In this section, we cover responses to a series of questions asking respondents whether their companies use remote development environments, what cloud providers they use, and how their redeploy times compare between local and remote development environments.

## Cloud Usage Trends

Respondents were asked whether their companies were using remote, containerized, or cloud-based development environments. While 51% of respondents said their companies are using remote development environments, 27% are not, and 20% are planning to add remote development environments in the future.

Among enterprise respondents there is increased polarization in those numbers; 73% of respondents are using remote development environments, 21% are not, and only 6% have plans to add remote development environments in the future.

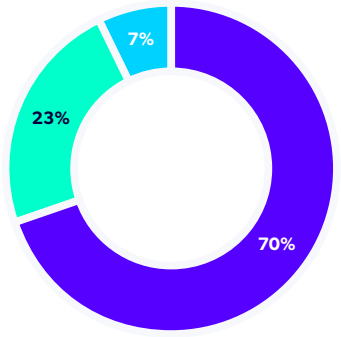
Respondents were also asked which cloud provider(s) their companies use. AWS (51%), Microsoft Azure (27%), and Google Cloud Platform (19%) won out, with tertiary cloud providers like Oracle Cloud (5%), Pivotal Cloud Foundry (5%), and IBM Cloud (2%) also seeing responses.

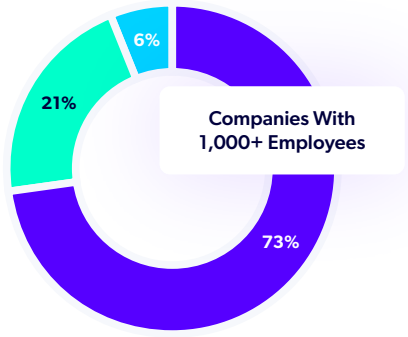
**KEY TAKEAWAYS:**

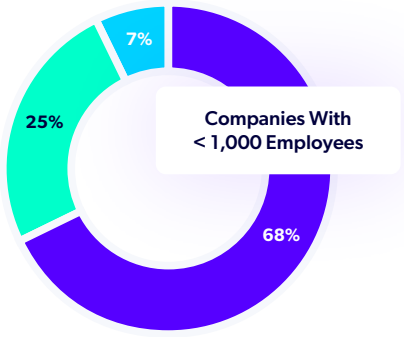
There's no denying that remote, containerized, and cloud-based development environments are increasing in popularity for businesses of all sizes. But increased complexities in these environments can create additional challenges for companies searching for development tools to ensure they are compatible and efficient within their own ecosystems.

While prior editions of the Java Developer Productivity Report saw a preponderance of respondents were using tertiary cloud providers as a part of a multi-cloud environment, this year's survey revealed that the multi-cloud makeup is dominated by the so-called "Big 3" of AWS, Microsoft Azure, and Google Cloud Platform.

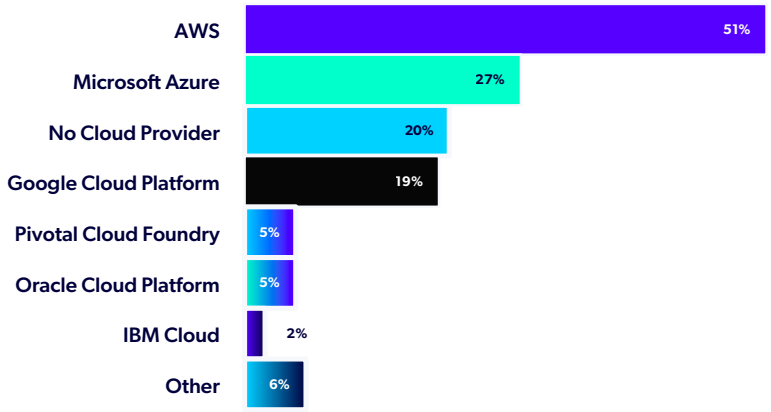
# Are you using remote, containerized, or cloud-based development environments?




# Which cloud platform(s) do




*Respondents could select all that applied.

## Remote Development Redeploy Times

In addition to cloud development plans and preferences, respondents were also asked to report their redeploy times for remote, containerized, and cloud-based development environments. 27% reported redeploys of 10 minutes or more, with another 25% reporting 5-9 minute redeploys and 18% with redeploys of 3-4 minutes.

Put this way, 23% of respondents say their redeploys are five minutes or longer for local development environments, but 52% saw redeploys of that length for remote development environments. Among enterprise companies (1,000+ employees) working in the cloud, that plight is even more pronounced, with 54% experiencing cloud redeploys of five minutes or longer.

Broken down by application architecture type, 64% of respondents using a monolithic application architecture saw redeploys of five minutes or longer in remote development environments — and 27% noted redeploys longer than 10 minutes. Those lengthy redeploys may be why 48% of that subset said that they're currently transitioning to microservices.

The situation improves slightly for respondents using microservices, with 53% experiencing redeploys that are five minutes or longer. But even then, 26% of that subset saw redeploys of 10 minutes or longer.

Cloud provider may have some bearing on remote development redeploy times. 60% of respondents using AWS reported remote redeploy times of five minutes or longer, whereas that figure was only 53% for respondents using Microsoft Azure and 43% for respondents using Google Cloud Platform (GCP). Cloud latency proved to be less of a barrier for respondents using tertiary cloud providers: Only 27% of respondents using Pivotal Cloud Foundry experienced remote redeploy times of five minute or longer, with similar figures for IBM Cloud (25%). That was not the case for Oracle Cloud, where 63% of respondents experienced remote redeploy times of five minutes or longer.

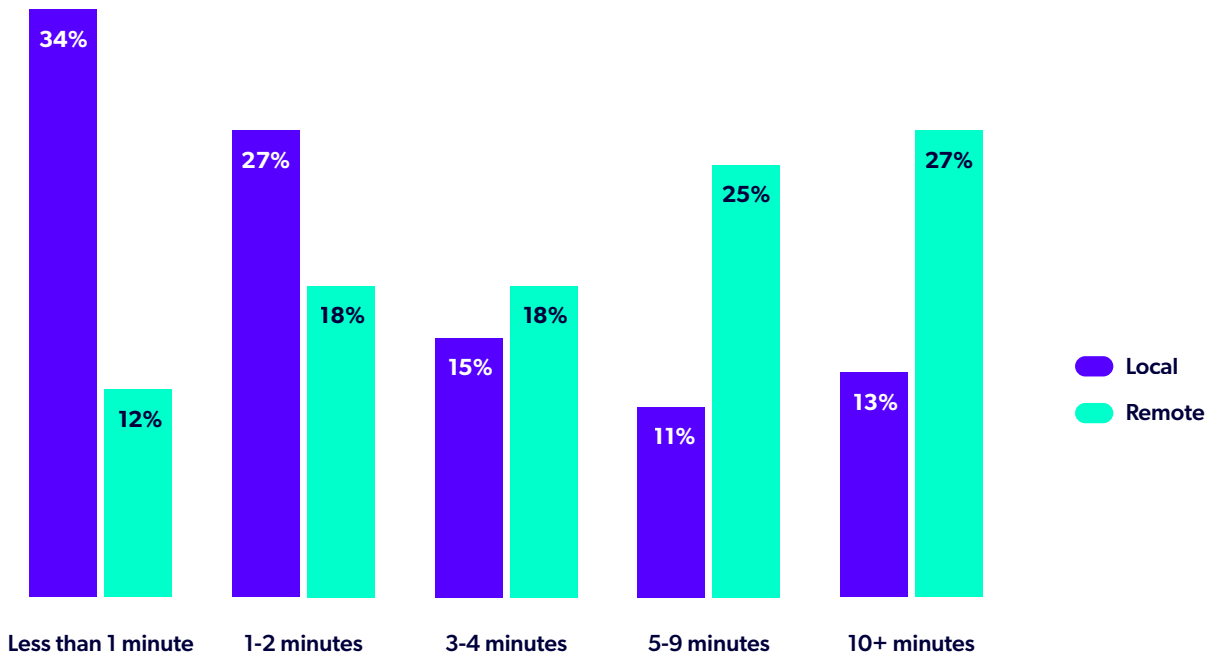
**KEY TAKEAWAYS:**

While we have been tracking local redeploy benchmarks for more than a decade, our dataset comparing redeploy times for remote vs. local development environments is just starting. That said, the trends are alarming, with redeploys of 10+ minutes common for remote development environments while the same respondents report local redeploys are predominantly less than two minutes.

While there are slight variations by development environment, application architecture type, business size, etc., remote development redeploy times appear to be a barrier across the board.

This cloud latency may cause frustration for development teams who are now burdened by more complex development environments. As a result, developers will get distracted while waiting, start doing other tasks, and get out of the flow — further magnifying those long redeploy times. The flexibility afforded by remote, containerized, and cloud-based development environments is undeniable, but the pain point of cloud latency cannot be ignored.

# Comparing Remote vs. Local Redeploy Times



# Application Architecture

In this section, we cover application architecture trends, adoption status, and increases in startup times for microservices.

## Microservices Give Way to Modular Architecture

While 64% of respondents use microservices and 36% of respondents use a monolithic application architecture, many respondents are using a mix of application architecture types that occupy the gray area between those polar opposites. Those midway options include modular monolith (24%), miniservices (19%), mobile (17%), SOA (16%), macroservices (15%), serverless (15%), and desktop app (14%).

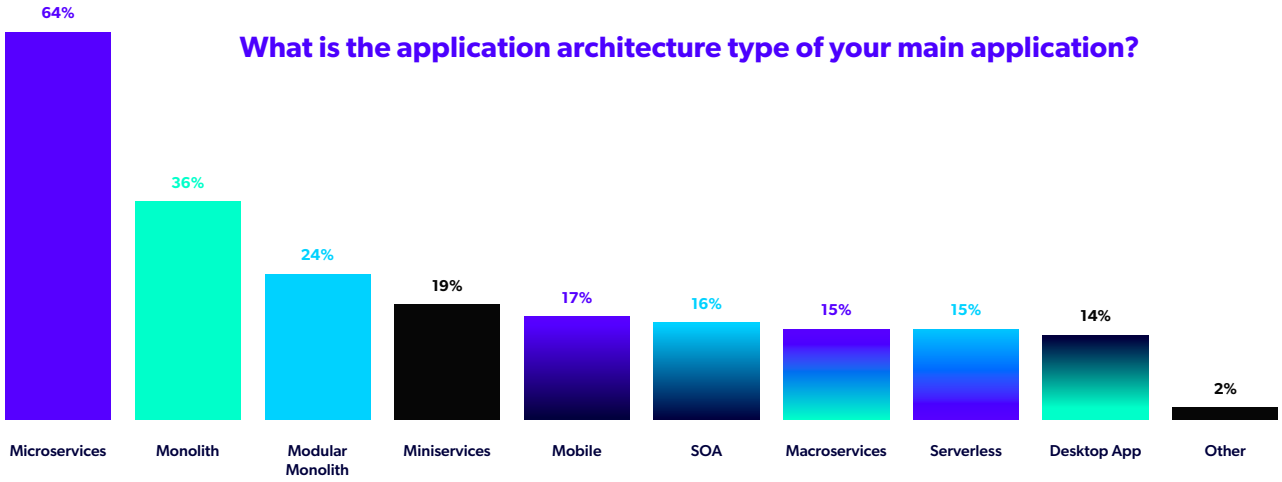
Among respondents using microservices, 23% have seen microservice startup times increase by 10% or more, while 23% have seen startup times increase by up to 9% and 54% have noted no increase. Results also showed that respondents are reevaluating their application architecture types. 43% are fully transitioned, 41% are in the process of transitioning, 12% are planning a transition, and 2% are currently reverting.

**KEY TAKEAWAYS:**

Enthusiasm for implementing microservices is often high, but companies find implementation difficult as they slide into miniservices or macroservices inadvertently — or make a conscious decision to revert from microservices entirely. Increasingly, more businesses are gravitating toward the best architecture fit for their application, rather than trying to force an application architecture that may no longer make sense.

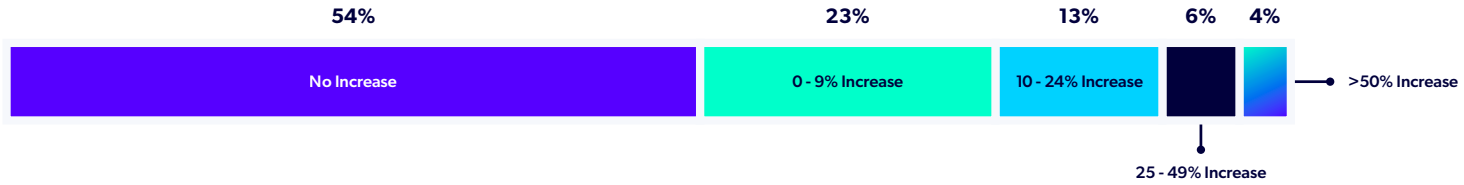
It is also important to note the role legacy environments play for business' mission-critical applications. 36% of respondents say they use a monolithic architecture for their main business application, with 39% of that subset reporting redeploy times that often exceed three minutes.

## What is the application architecture type of your main application?

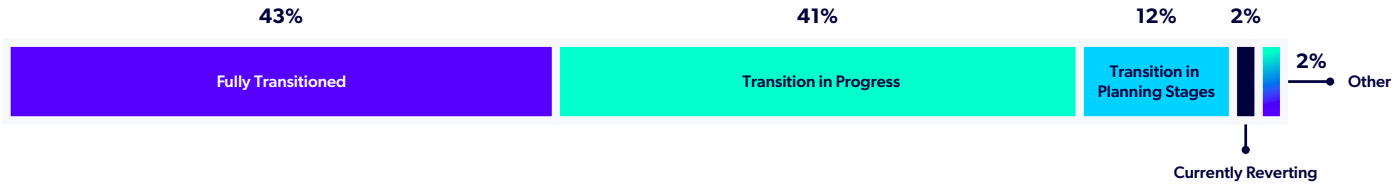



*Respondents could select all that applied.

## Have you experienced an increase in startup time for microservice applications?




## What is your application architecture adoption status?



# Definitions

**MICROSERVICES:**
An architectural style that structures an application as a collection of loosely coupled, independently deployable services.

**MINISERVICES:**
Similar to microservices but with slightly larger, more domain-driven boundaries. They are more substantial than microservices but smaller than monolithic services.

**MACROSERVICES:**
Larger services that encompass significant functionality, often representing a significant portion of an application, but are still smaller than a monolithic architecture.

**MONOLITH:**
A traditional architecture where the entire application is built as a single, indivisible unit.

**SOA (SERVICE-ORIENTED ARCHITECTURE):**
An architectural pattern where services communicate over a network to provide functionality. SOA emphasizes reusability and interoperability.

**DESKTOP APP:**
An application that runs on a desktop or laptop computer and does not rely on a web browser.

**MOBILE:**
An application designed to run on mobile devices such as smartphones or tablets.

**SERVERLESS:**
An architecture where the cloud provider manages the server infrastructure, allowing developers to deploy code without worrying about the underlying servers.

**MODULAR MONOLITH:**
An architectural style where the application is a single unit but is divided into distinct, self-contained modules.

# Productivity Trends

Read on for an in-depth look at the biggest time traps facing Java development teams and the tools and techniques they're using to solve them.

## Development Time

Respondents were asked what they would do with more development time in their day. Pragmatic answers included refactoring code and eliminating technical debt, while other respondents said they would contribute to open-source projects, go home early, or make coffee. Regardless of motives, it's clear that extra time is the ticket to increased business value.

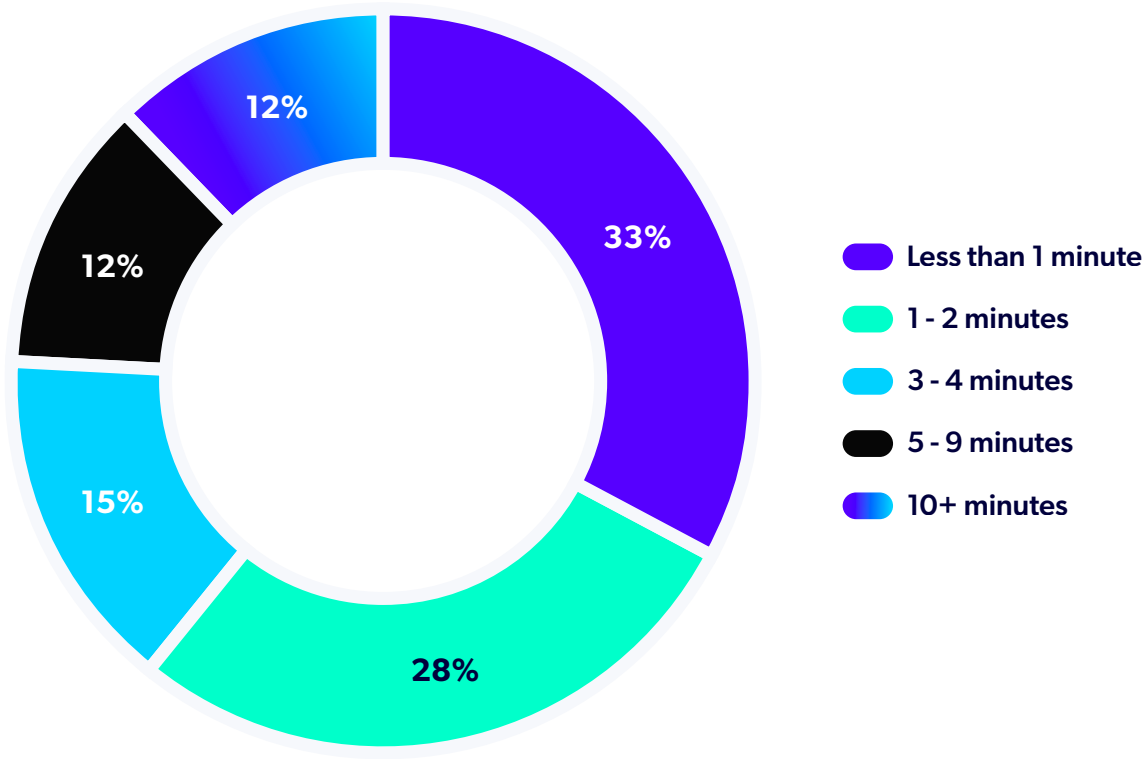
## Redeploy Benchmarks

Redeploys remain a threat to Java developer productivity, as chronicled in previous Java Developer Productivity Reports. 33% of respondents reported redeploy times of less than 1 minute, 28% reported redeploy times between 1-2 minutes, 15% reported redeploy times of 3-4 minutes, 12% reported redeploy times of 5-9 minutes, and an additional 12% reported redeploy times lasting more than 10 minutes.

**KEY TAKEAWAYS:**

The aforementioned redeploy times may seem inconsequential on their own, but redeploy times represents wasted minutes and a break in a developer's workflow. Without those interruptions, developers can dedicate their time toward mission-critical business tasks. Think of what extra development time could contribute to your team's ability to write better code, improve testing, add features, accelerate time to market, and more.

# How long does it take to redeploy your primary business application?



## AI in Java Development

AI is everywhere — and that includes Java development. Only 12% of respondents said they don't use AI tools, and an additional 12% said their companies don't allow the use of AI tools. Among enterprise respondents, however, the number who said their companies don't allow the use of AI tools climbs to 16%.

Top AI tools included generic juggernaut ChatGPT (52%), as well as developer-specific AI tools including GitHub CoPilot (42%) and IDE-integrated AI tools (25%). Enterprise respondents showed a preference for development-specific AI tools, with GitHub CoPilot clearly in the lead at 52% for this subset.

Among those using AI tools, there are clear trends in what tasks they help save time. Respondents were most likely to turn to AI tools for code completion (60%) and refactoring (39%). Error detection (30%), documentation generation (28%), debugging assistance (26%), and automated testing (21%) were also ripe use cases.

"AI coding assistants get better every month. Developers who tried AI a few months ago may think it's annoying or gets in the way. My advice is to keep trying AI tools at least once a quarter."

— Rod Cope, CTO, Perforce Software

**KEY TAKEAWAYS:**

As AI development tools continue to mature, they will create a clear competitive advantage for companies that buy in. Companies that are skittish about adopting AI tools or create blanket policies prohibiting their use will get left behind. That said, AI coding assistants are far from perfect today, and it's not easy to know which AI vendor and model will work best for a given programming language, codebase, use case, etc.

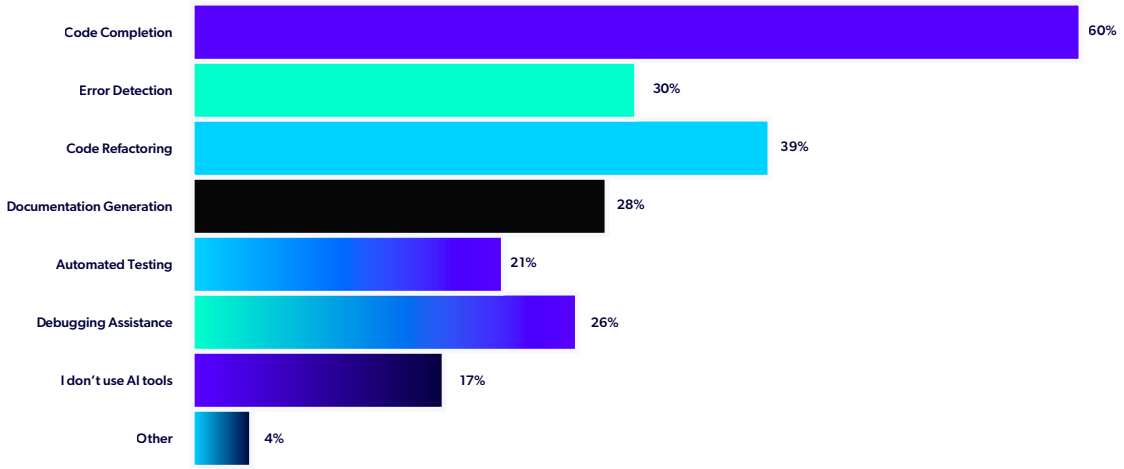
But the conversation around the AI advantage is more nuanced than that. Developers can choose between AI plugins for their IDE of choice, like GitHub CoPilot, a built-in IDE assistant like the JetBrains AI assistant, or a new IDE that's built with native AI integration, i.e., Cursor.

It's not enough to just use AI tools; your business needs to be using the right AI tools. And what's right is constantly changing based on use case and algorithm changes. Today, that might be an agentic IDE like Windsurfer, but tomorrow that could be something else entirely. It's critical for engineering organizations to continue monitoring this evolution, adjust their sails as they go, and continue to be open minded in how best to improve their efficiency, security, etc.

Moreover, it's important to note that AI cannot replace a human developer's eye, but it can help accelerate processes and reduce errors. Developers should lean into the areas where AI excels so they can apply their time and talents to tasks that make a direct impact on business value.

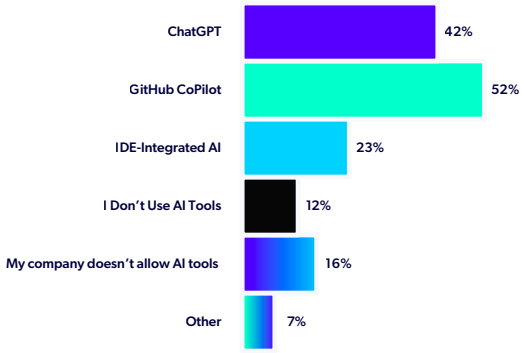
# AI in Java Development

## Which AI features do you use most frequently?

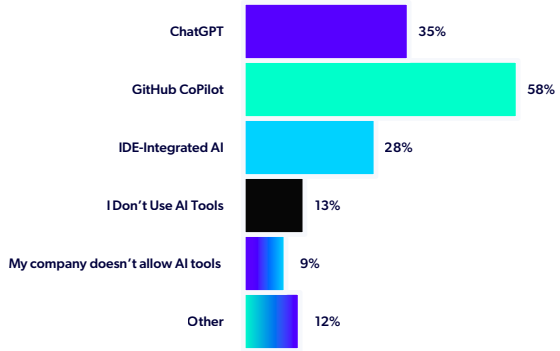



## Which AI tools are you using for Java development?

### 1,000+ Employees




### <1,000 Employees




*Respondents could select all that applied for all charts above.

## Encouraging Developer Productivity

Barriers to Java development productivity can take on many forms. Respondents reported that their biggest barriers to productivity were insufficient documentation (41%), communication issues between teams (38%), and mismanaged timelines (32%); long redeploy times (29%), developer turnover (26%), and insufficient developer tools (24%) also made the list.

In addition to asking respondents about their Java tech stack and development challenges, we also asked about the methods they are using to improve developer productivity. Results were across the board, with 47% tasked with individual efforts to investigate tools, 38% using work groups to investigate tools, 50% using AI tools, and 21% who said they are on their own.

Enterprises, however, are taking a less laissez faire approach to Java development. Among respondents from companies with 1,000+ employees, 49% have implemented workgroups to investigate tools and implement solutions. Usage of AI tools is also higher at 55% for enterprise respondents vs. 50% overall.

**KEY TAKEAWAYS:**

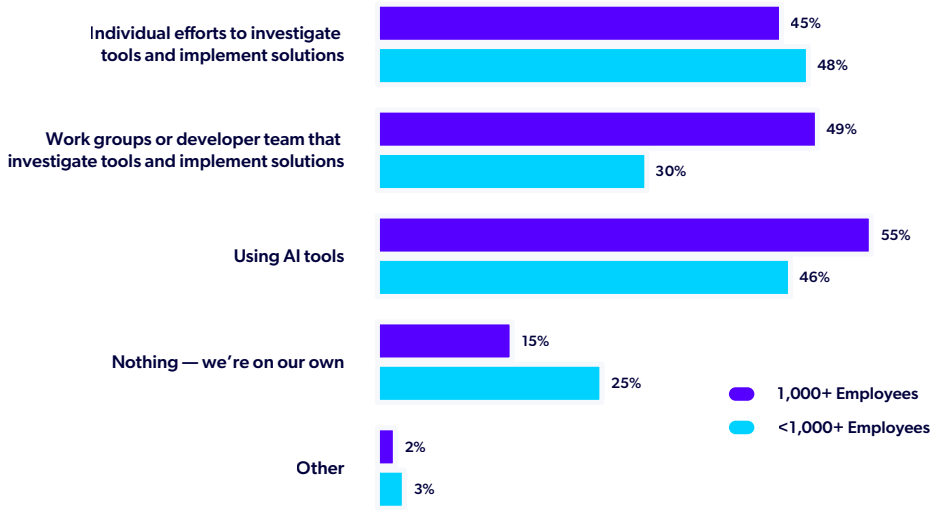
Developers are being tasked with doing more with less. And instead of trying to create more hours in the day, knowledgeable leaders are looking to a mix AI tools, individual efforts, and work groups to help pick up the slack. Among enterprise respondents, however, there is more coordination in those efforts: 49% of that subset uses work groups or developer teams to investigate tools and implement solutions, vs. only 30% of all other respondents from smaller companies.

Regardless of how companies are looking to overcome these development barriers, they strike with equal opportunity against businesses of all sizes. While issues like insufficient documentation can easily be addressed with AI tools, long redeploy times and insufficient developer tools require a more holistic approach to realize time savings and business value.

# Barriers and Solutions for Java Productivity

## What methods do you use to encourage developer productivity?


## Which methods do you use to encourage developer productivity?




## What are your biggest barriers to development productivity?


*Respondents could select all that applied for all charts above.

# Final Thoughts

Each year, the Java Developer Productivity Report produces powerful benchmarks on the challenges other businesses that rely on Java are facing — and the tools they're using to solve them. We hope you've found this report to be packed with valuable insights that inform how your business approaches Java development productivity.

Businesses that rely on Java should pay attention as Java users migrate en masse to Java 17 and Java 21 (61% and 41%, respectively), or as VS Code overtakes Eclipse as the second-most popular Java IDE. The Java landscape is constantly changing, even as the language celebrates its 30th birthday this year.

In the face of uncertain budgets and an imperative to do more with less, we hope you'll investigate Java productivity tools to extend the reach of the development resources you already have. AI assistants and the like might be stealing headlines at the moment, but keep in mind that 53% of respondents said that long redeploys and insufficient development tools are their biggest barrier to productivity.

Ultimately, the Java development teams that arm their teams with appropriate tools will be the ones that will succeed working in increasingly complex development environments and demanding business conditions.

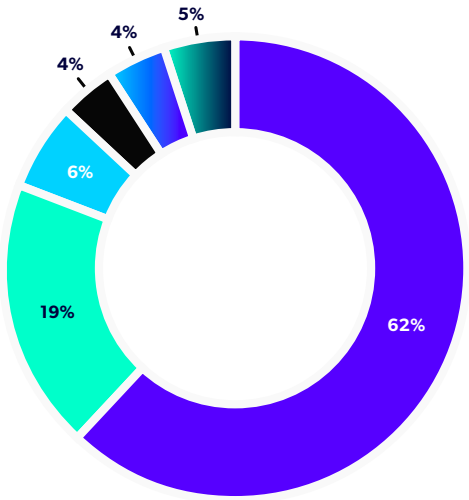
# Appendix

## About the Report

The 2025 Java Developer Productivity Report is based on an anonymous survey conducted between September 2024 and January 2025 that received 731 responses.

## Job Title

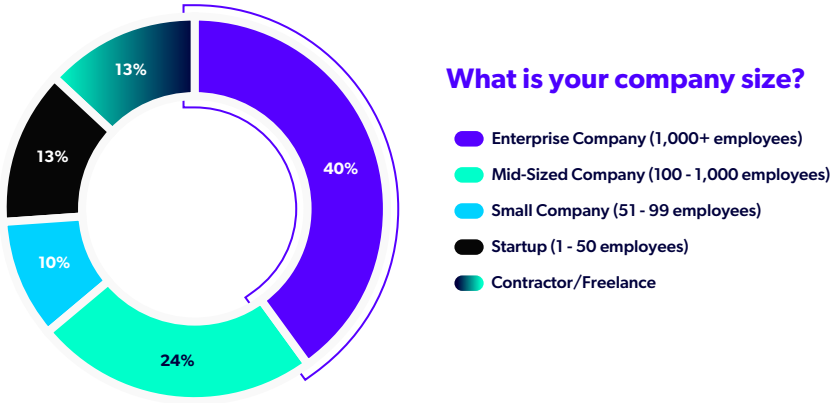
While most respondents listed their role as developer (62%), this year's survey saw more respondents in leadership roles compared to 2024, including team lead (19%), director/manager (6%), and vice president/executive (4%).

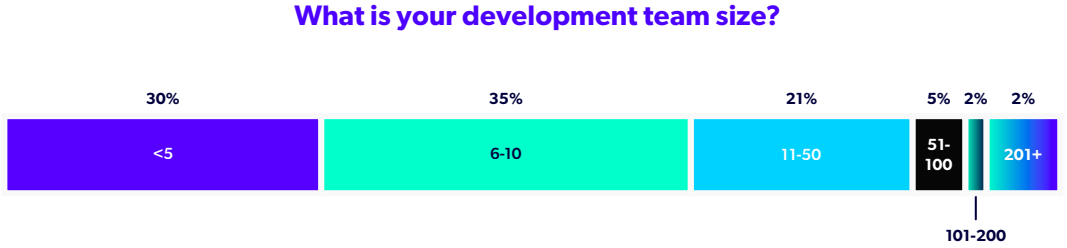


# Team and Company Size

Respondents represented a wide range of company sizes from enterprises and midsize organizations to startups and freelancers. Company size spread was mixed, with 40% from companies with 1,000+ employees, 24% from companies with 100-1,000 employees, 10% from companies with 51-99 employees, 13% from companies with 1-50 employees, and 13% describing their roles as contractor or freelance.

The size of respondents' development teams that works in Java was across the board. 7% of respondents were on teams with 201+ Java developers, 2% reported 101-200 Java developers on their teams, 5% are on teams with 51-100 Java developers, and 21% reported working with 11-50 other Java developers at their organizations. That said, the preponderance (65%) were on teams of 10 or less.



# About the Authors

### ROD COPE

Rod Cope is the CTO of Perforce Software. He provides technical vision and architectural leadership for the company's globally distributed development teams, including JRebel. For the past 25 years, Rod has spoken on various technical and business topics at dozens of conferences around the world, including JavaOne, OSCON, and Embedded World.

### JEFF MICHAEL

Jeff Michael is a product leader at Perforce Software responsible for the entire Development Tools portfolio, including JRebel. He has over 25 years of software delivery experience with a focus on OSS management, security, and risk compliance.

# About Perforce

Perforce powers innovation at unrivaled scale. With a portfolio of scalable DevOps solutions, we help modern enterprises overcome complex product development challenges by improving productivity, visibility, and security throughout the product lifecycle.

Our portfolio includes solutions for Agile planning & ALM, API management, automated mobile & web testing, embeddable analytics, open source support, repository management, static & dynamic code analysis, version control, and more. With over 9,000 customers, Perforce is trusted by the world's leading brands, including NVIDIA, Pixar, Scania, Ubisoft, and VMware. For more information, visit perforce.com.

**Learn More**