5 steps to upgrade from OpenJDK 8

Learn how to safeguard and scale your Java-based application portfolio

The Red Hat® build of OpenJDK 8 enters extended lifecycle support (ELS) November 2026. For those looking to stay on OpenJDK 8, Red Hat offers ELS with a level of support organizations need to help keep existing applications running, while maintaining compliance and focus on security and optimizing support and licensing costs.

For those looking to modernize, Red Hat provides the experience, tooling, and support to help streamline migration, mitigate migration-related risks, and optimize post-migration efficiency.

Read this checklist to learn about the key stages of migrating to a newer version of the Red Hat build of OpenJDK.

1 Assess your application portfolio and culture

Documenting your current environment and portfolio of Java-based applications, including detailed information on third-party dependencies and review release notes, and setting up a testing environment, will help you identify any potential issues before starting the migration process. Consider using the migration toolkit for applications, which can help you streamline the migration process by analyzing, scoping, and automating parts of the process.

Assessing your people and culture—including identifying key stakeholders, ownership of key technologies and processes, and any resistance to change—will help you understand how to keep your teams fully aligned as you evolve to meet changing needs and adopt more frequent updates. Assess your internal resources to see if consultation services might be required to help complete migration in a timely manner.

Red Hat Consulting offers migration-specific support to help you assess your culture and application portfolios.

2 Choose the right migration approach for your future

Critical upgrades are becoming more frequent in the Java ecosystem, which makes choosing the right migration approach a crucial step to help your organization mitigate risk, control costs, and facilitate an efficient migration.

By developing long-term maintenance plans as part of your current migration strategy, your organization can also better prepare for future updates.

Recommended migration approaches include:

- **A direct migration** to the latest long-term support (LTS) version—in this case Java 25. This offers connectivity to modern technologies (such as advanced AI tools, post-quantum cryptography, cloud-native capabilities, performance tuning, and multithreading) and helps your organization lower resource investment during migration.
- **A phased migration**, for example, progressing along some combination of Java 11, Java 17, Java 21, and Java 25. This allows your organization to ease the efforts of fixing missing features or dependencies by taking smaller jumps. However, this approach requires more time and resources, and introduces risk with each migration step.

3 Start migrating your Java-based applications

Migrating applications involves several steps that can help improve efficiency and focus on security, including:

- **Code refactoring.** Removing deprecated application programming interfaces (APIs), such as Applet and Security Manager, making changes to default garbage collector and Java virtual machine (JVM) options, and setting up a new default Hypertext Transfer Protocol Secure (HTTPS) provider.
- **Performance optimization.** Tuning JVM settings and profile application behavior, as well as addressing new performance characteristics, to optimize efficiency and stability after migration.

4 Prioritize automated, comprehensive testing

It is important to set up automated test cases for your application portfolio—including smoke, unit, integration, and functional tests—to check functionality is in place and help you adapt and validate changes more efficiently.

This can be supported with continuous integration and continuous delivery (CI/CD) pipelines to automate build and deployment—as well as incremental rollouts with fallback strategies—and facilitate feedback and integration into production environments.

5 Choose tools that help simplify your migration

Red Hat offers a range of tools that streamline your migration by reducing complexity and mitigating migration-related risks, including tools that help you:

- **Analyze and plan.** The migration toolkit for applications helps you prepare for migration with tools for identifying high-level issues, providing effort estimates, and more.
- **Refactor code.** Red Hat Developer Lightspeed offers AI-powered assistants that help streamline source code refactoring, including providing AI-generated code tailored to your applications.
- **Automate and scale.** Ansible® Middleware offers tools that help efficiently scale your Java-based applications, including automated deployment, upgrades, and management.

Get the help you need to modernize your Java-based application platform

Speak to your Red Hat account team or reach out to Red Hat support to discuss your Java modernization strategy and how Red Hat can help streamline migration, mitigate migration-related risks, and optimize post-migration efficiency.