How to Future-Proof Your Java Stack

Migration Strategies for JBoss EAP Users

Power Up Your Enterprise Java

User Guide

Contents

Guide Updated: **December 2025**

For decades, Red Hat JBoss Enterprise Application Platform (EAP) has served as a cornerstone of enterprise Java ecosystems, providing a stable, feature-rich and commercially supported runtime for mission-critical applications. Its reputation for reliability made it a prudent choice for organizations building highly transactional, web-scale systems. This very stability, however, has created a significant paradox. The platforms that were once bastions of enterprise security and performance have, for many organizations, become sources of profound inertia. This inertia transforms a once-valuable asset into a source of escalating technical debt and critical business vulnerabilities, particularly for those still operating on older, unsupported versions.

The Support Lifecycle Reality Check

The JBoss EAP support lifecycle isn't arbitrary. It's a structured policy designed to focus engineering resources on modern, secure platforms. Understanding these timelines is essential because ignoring them introduces significant, unmanaged risk.

Red Hat's support phases each represent a material degradation in coverage. Full Support provides comprehensive services including bug fixes and security patches for all severity levels. Maintenance Support narrows this considerably: only patches for critical issues, no new features. Then comes Extended Life Support (ELS), which sounds reassuring but operates in two phases that few organizations understand correctly.

ELS-1 provides patches only for vulnerabilities Red Hat deems "critical." Moderate and low-severity issues remain unpatched. ELS-2 offers virtually nothing: just migration guidance while your platform accumulates vulnerabilities. You're paying premium prices to get advice about leaving.

Here's where major JBoss EAP versions stand today:

JBoss EAP Version	Full Support End	Maintenance Support End	ELS-1 End	Final End of Life
5.x	Nov 30, 2013	Nov 30, 2016	Nov 30, 2019	ELS-2 Not Offered
6.x	Jun 30, 2016	Jun 30, 2019	Jun 30, 2022	ELS-2 Not Offered
7.x	Dec 31, 2023	Jun 30, 2025	Oct 31, 2027	Oct 31, 2030
8.x	Feb 05, 2028	Feb 05, 2031	Feb 05, 2033	ELS-2 Not Offered

JBoss EAP versions 5 and 6 are long past any form of support. Any organization still running these versions is operating with extreme risk. For JBoss EAP 7 users, maintenance support concluded in mid-2025, and the final deadline for security patches under ELS-1 is October 31, 2027. Waiting until the final months forces you into a high-pressure, reactive migration of an already insecure platform. These emergency projects are invariably more expensive, disruptive, and prone to failure than planned modernization efforts.

The Real Cost of Staying Put

The decision to defer migration introduces a cascade of interconnected risks that permeate every layer of your technology stack and business operations. These aren't theoretical concerns. They're tangible vulnerabilities that grow more severe over time.

Your Attack Surface Is Growing Daily

Once a product exits its support lifecycle, the vendor stops issuing security patches. Vulnerability discovery doesn't stop. This creates a one-way accumulation of risk where your attack surface expands daily with no corresponding mitigation.

The threats are diverse and severe. Remote code execution vulnerabilities in older JBoss versions, particularly through deserialization flaws, allow attackers to execute arbitrary commands and completely compromise systems. Path traversal issues let attackers access privileged files: configuration files, database credentials, private keys. Denial of service flaws can be exploited to exhaust resources and crash applications. Cross-site scripting vulnerabilities in administrative components enable session hijacking and user impersonation.

Log4Shell serves as the perfect case study. When this critical vulnerability in Apache Log4j surfaced, Red Hat promptly patched all supported EAP versions. Organizations on end-of-life (EOL) versions had no official patch. They scrambled with complex, incomplete workarounds: modifying runtime configurations, manually replacing libraries. These were risky procedures that could break functionality. This single event illustrated the untenable position of operating end-of-support (OES) or EOL software in a world of zero-day threats.

Performance Is Suffering From Architectural Decay

Older JBoss EAP versions impose significant performance penalties due to dated architecture. The evolution from EAP 5 and 6 to EAP 7 and beyond wasn't incremental. It was a fundamental re-architecture that yielded substantial gains in efficiency, startup time, and scalability.

Early versions were built on a monolithic architecture with complex, hierarchical classloaders and sprawling configuration spread across dozens of XML files. The server had to load and initialize all subsystems at startup, whether applications needed them or not. The result? Notoriously slow boot times, large memory footprints, and configurations that were difficult to manage and troubleshoot.

Modern application servers have moved beyond these limitations. Modular class loading provides better isolation and dependency control. Centralized configuration consolidates those myriad XML files into unified formats. Lazy loading means subsystems load on-demand, only when applications require them. This dramatically reduces startup times and lowers baseline memory consumption. Modern web servers like Undertow offer non-blocking I/O designed specifically for high-throughput, high-concurrency environments.

Organizations clinging to older versions actively forego these architectural benefits. They battle OutOfMemoryError exceptions due to less efficient garbage collection in older JVMs, requiring constant reactive tuning. Slow startup times hinder agility, lengthening development cycles and delaying recovery after outages. Most importantly, they're running on web containers that are architecturally inferior for handling modern, high-traffic applications.

Compliance Violations Are Guaranteed

For any organization handling sensitive data (particularly payment card information), running EOS or EOL software isn't just a technical risk. It's a direct compliance violation.

PCI DSS Requirements 6.1 and 6.2 explicitly mandate that organizations establish processes to identify security vulnerabilities and ensure all system components are protected from known vulnerabilities by installing applicable vendor-supplied security patches. By definition, an EOL product for which the vendor no longer supplies patches cannot meet this requirement.

While PCI DSS allows "compensating controls" in certain circumstances, the PCI Security Standards Council has stated these should be considered "temporary solutions only" and that organizations must have an "active migration plan" to upgrade to supported versions. Implementing effective compensating controls (dedicated network segmentation, advanced host-based intrusion prevention, application whitelisting, intensive real-time monitoring) is often far more complex and expensive than performing the necessary migration.

The situation becomes even clearer for internet-facing systems. A system running an unsupported platform will be flagged as an automatic failure during required scans by Approved Scanning Vendors. This makes achieving and maintaining PCI DSS compliance practically impossible.

You're Stranded on a Technical Island

Perhaps the most insidious long-term cost is the gradual descent into technical obsolescence. An outdated application platform acts as an anchor, preventing the entire development ecosystem from moving forward and adopting modern tools, practices, and architectures.

JBoss EAP 6, for example, is only officially supported on Java 8, with no support for modern Long-Term Support releases like Java 11, 17, or 21. This forces development teams to work with an aging version of the Java language, depriving them of significant language features, performance improvements, and security enhancements built into newer JDKs. This also creates a secondary security risk: the underlying JDK itself will eventually become end-of-life and stop receiving security updates.

Legacy platforms were designed for a pre-cloud, pre-container world of vertically scaled, manually configured servers. Attempting to force-fit them into modern infrastructure paradigms is inefficient and counterproductive. The resulting deployments are often economically inefficient on cloud platforms optimized for stateless, horizontally scalable applications that can take advantage of auto-scaling and pay-as-you-go pricing models.

These risk categories feed into one another, creating a self-reinforcing cycle of decay. Architectural limitations make platforms harder to patch and secure. Security vulnerabilities lead directly to compliance failures. The inability to upgrade prevents adoption of modern patterns that could help mitigate security and performance issues. The total risk grows exponentially the longer you wait to act.

Three Migration Pathways Forward

The optimal migration strategy depends on your specific business context, technical capabilities, and long-term architectural vision. Here are three distinct pathways representing different approaches to modernization.

Pathway 1: Strategic Migration to Payara Platform

For organizations seeking a modern, fully supported Jakarta EE platform without the complexity of complete re-architecture, migrating to Payara Platform Enterprise offers the most direct path to security and compliance.

Payara Platform Enterprise is a production-ready application server derived from GlassFish, enhanced with enterprise features, performance optimizations, and full commercial support. It's designed specifically for organizations that need the stability of Jakarta EE with the agility of modern cloud infrastructure.

For JBoss EAP 7 users, migrating to Payara Platform Enterprise 5 offers immediate benefits with minimal disruption. Both platforms are based on Java EE 8, which means your applications can continue using the `javax.*` namespace without code changes. You avoid the painful javax to jakarta namespace migration entirely while immediately regaining full commercial support including regular security patches, bug fixes, and access to enterprise-grade tooling. This isn't a stopgap. It's a strategic decision to move to a platform that's actively maintained, fully compliant, and designed for modern infrastructure.

For organizations ready to adopt Jakarta EE 10, Payara Platform Enterprise 6 provides a clear migration path with comprehensive tooling support. The platform includes Docker images for both Full Profile and Web Profile deployments, native Kubernetes support, and integration with modern monitoring and observability tools. Unlike staying within the Red Hat ecosystem, you're not locked into a single vendor's timeline or pricing model.

Payara Platform Enterprise also offers features that JBoss EAP lacks or charges extra for:

- Advanced health checking and notification systems for proactive monitoring
- Request tracing for debugging production issues
- Payara Micro for microservices architectures
- Extensive Docker and Kubernetes support out of the box
- Flexible deployment options from traditional servers to cloud-native containers

Migration tooling exists to help assess and execute the transition. The Payara team provides migration guides, professional services, and the Payara Accelerator service to support adoption, assessment, and reconfiguration work. For organizations with complex infrastructure or limited in-house expertise, this professional support significantly reduces migration risk and accelerates time-to-production.

Pathway 2: Incremental Upgrade Within Red Hat Ecosystem

For organizations deeply committed to Red Hat's product portfolio with significant existing investments, upgrading to JBoss EAP 8 remains an option. However, this path presents substantial challenges that organizations should carefully weigh.

The primary obstacle is the transition from Java EE to Jakarta EE 10, which requires the namespace migration from `javax.*` to `jakarta.*`. Every Java import, configuration reference, and XML descriptor must be refactored. This requires meticulous audit and modification of the entire application codebase and deployment descriptors.

Red Hat provides tools to facilitate this process. The Migration Toolkit for Applications scans application source code and binaries, generating detailed reports identifying all necessary migration tasks. The JBoss Server Migration Tool can read configuration files from older EAP servers and automatically migrate settings to the EAP 8 format.

However, this pathway keeps you locked into Red Hat's licensing model and support terms. As the lifecycle table shows, even EAP 8 will eventually reach end-of-life, requiring future migrations. Organizations should carefully evaluate whether the investment in Red Hat-specific tooling and expertise delivers long-term strategic value, or whether migrating to a more flexible platform makes better business sense.

Pathway 3: Cloud-Native Re-architecture

For organizations ready to fundamentally transform their application architecture, decomposing monolithic applications into microservices represents the most forward-looking approach. This pathway involves moving beyond traditional application servers entirely, adopting lightweight frameworks designed specifically for cloud-native environments.

Frameworks like Quarkus and Spring Boot follow an "embedded server" model, packaging application code with only necessary libraries and a minimal embedded web server into a single executable artifact. These frameworks are optimized for fast startup times and low memory footprints, making them well-suited for containerized and serverless environments.

However, this is a re-architecture project, not a migration. It requires applying domain-driven design principles to identify bounded contexts within existing monoliths and incrementally rewriting them as independent microservices. There's no automated tool to perform this core domain decomposition. It demands significant time, expertise, and organizational commitment.

For most organizations with production JBoss EAP applications, this represents a multi-year transformation journey rather than a tactical migration. It's a valid long-term vision, but it shouldn't prevent you from addressing immediate security and compliance risks. Many successful organizations adopt a hybrid approach: migrating existing applications to a modern, supported platform like Payara first, then selectively re-architecting high-value components as microservices over time.

Why Payara Makes Sense for JBoss EAP Users

Organizations evaluating migration options often overlook a fundamental question: why stay locked into a single vendor's ecosystem when alternatives offer equal or better capabilities with more flexibility?

Payara Platform Enterprise provides full Jakarta EE compatibility, enterprise support, modern infrastructure integration, and competitive pricing without vendor lock-in. You're not betting on a single vendor's product roadmap or accepting their licensing terms indefinitely. You're choosing a platform backed by a company focused exclusively on application server excellence, not one managing dozens of product lines with competing priorities.

For organizations on JBoss EAP 7, the path to Payara Platform Enterprise 5 is particularly compelling. You maintain your existing Java EE 8 codebase, avoid the namespace migration complexity, and immediately regain security patch coverage and compliance standing. Your development team can continue working in familiar territory while you plan longer-term architectural evolution on your timeline, not a vendor's support calendar.

For organizations ready to adopt Jakarta EE 10, Payara Platform Enterprise 6 offers a modern, cloud-ready runtime with full Docker and Kubernetes support, advanced monitoring capabilities, and flexible deployment models from traditional servers to microservices.

Transform Debt Into Opportunity

The evidence is conclusive: continuing to operate on end-of-life JBoss EAP versions is unsustainable and high-risk. The compounding dangers of an ever-growing unpatched attack surface, degrading performance from archaic architectures, guaranteed non-compliance with critical security standards like PCI DSS, and an innovation bottleneck that prevents adopting modern technologies create formidable barriers to business agility and resilience.

However, the necessity of migration shouldn't be viewed solely as a burdensome cost center or reactive technical chore. It represents a pivotal strategic opportunity: a chance to pay down years of accumulated technical debt, fundamentally modernize the Java application stack, and break free from vendor lock-in that constrains future flexibility.

The path forward begins with a clear-eyed assessment. IT leaders must conduct a thorough inventory of the entire JBoss EAP estate, use the end-of-life timelines to triage risk, and initiate a formal analysis and planning process. The future viability, security, and performance of the enterprise Java portfolio depend on the strategic decisions and actions taken today.

Need Immediate Protection? Payara Lifetime Support Bridges the Gap

We understand that not all workloads can be migrated immediately. Complex infrastructure, small development teams, mission-critical systems with limited downtime tolerance, and resource constraints are real challenges that organizations face.

That's why Payara offers Lifetime Support for Payara Platform Enterprise 4. This isn't about indefinitely postponing necessary modernization. It's about giving you the time and stability you need to plan and execute migrations properly, without leaving your systems exposed while you work.

Lifetime Support provides:

- Continued security fixes and critical bug patches through December 2030 (aligned with Java 8 EOL)
- Enterprise-level support for business-critical applications
- Time to plan strategic migration to Payara Platform 5 or 6 on your timeline
- Professional services support to guide your modernization journey

Think of Lifetime Support as your safety net while you build your future. You maintain security and compliance standing for systems that can't move immediately, while systematically planning migrations to Payara Platform 5 or 6. This phased approach reduces risk, spreads costs over time, and ensures you're never forced into emergency migrations under pressure.

Whether you're currently on JBoss EAP 7 looking for a migration path, on Payara Platform Enterprise 4 needing continued support, or on Eclipse GlassFish seeking a commercially supported alternative, we're here to help you navigate the journey to modern, secure Java infrastructure.

Learn more about Payara Lifetime Support and discover how we can support your migration to Payara Platform Enterprise 5 or 6.

sales@payara.fish

UK: +44 800 538 5490
Intl: +1 888 239 8941

www.payara.fish