



# From Stagnation to Acceleration: The Application Server Migration Executive Guide



# Contents

Guide Updated: **July 2025**

<b>Executive Summary</b>	<b>1</b>
<b>Why Migrate to a Different Application Server</b>	<b>2</b>
<b>Common Migration Stoppers and How to Address Them</b>	<b>3</b>
Perceived Risks & Fear of the Unknown	4
How to Address Risks and Fears:	4
Stagnation: If it ain't broke, why fix it?	4
How to Address Organizational Inertia & Stagnation	4
Cost of Migrating	4
How to Drive Cost-Effectiveness:	5
Vendor Onboarding Complexity	5
How to Simplify Vendor Onboarding:	5
Vendor Lock-In & Bundling Concerns	5
How to Avoid Future Vendor Lock-Ins:	5
<b>Overcoming Key Barriers</b>	<b>6</b>
How can I Ensure Business Continuity?	6
Best Practices to Ensure Uptime:	6
Is my Team Skilled Enough to Address the Complexity of a Migration?	6
Best Practices to Address Any Skill Gaps:	7
Is Migrating Secure?	7
Best Practices to Ensure App Security:	7
<b>Understanding Your Options</b>	<b>8</b>
<b>Planning Your Migration</b>	<b>9</b>
<b>Regular Reviews and Evaluations</b>	<b>9</b>
<b>Take the Next Step Now</b>	<b>10</b>

# From Stagnation to Acceleration: The Application Server Migration Executive Guide

**Successfully move to an enterprise Java runtime that improves application performance, reduces risk and empowers your team**

## Executive Summary

Not all application servers or runtimes are created equal, and over time, the one you started with may no longer be the best fit for where your business and application strategy are heading. In these cases, migrating to a better suited solution can deliver key benefits.

However, migrating application servers or runtimes is a critical yet complex process, thus the idea of migrating to a new platform can feel overwhelming for the risks, costs and uncertainties such activity brings.

This guide demystifies the runtime migration process. It explores why organizations choose to migrate, what holds them back, and how to confidently plan and execute a successful transition.

## Why Migrate to a Different Application Server

As organizations advance on their digital transformation journeys, advanced, reliable and effective applications become crucial to drive business success. To do this, the infrastructure behind your software needs to support innovation, speed, robustness and security.

An application's specific middleware requirements are not static but evolves over time. As such, what may have been able to support your applications and your business in the past may no longer be offer the best solution to address your needs. In these instances, sticking with your existing application servers or runtimes can hold your business back. Key drivers for migration include:

- **High Costs:** Many teams continue to pay premium licensing or support fees without seeing a return in performance, reliability or service quality. When vendors bundle in additional tools and features you don't use or lock you into long contracts, you may feel like you are not getting value for money. What you can gain by switching:
  - Transparent and predictable pricing models
  - Elimination of unnecessary costs
  - Better alignment between what you pay for and what you use
- **Poor Developer Experience:** If your developers are spending more time understanding, configuring, fixing or actually fighting your runtime than innovating, you are also likely experiencing frustration, high attrition and slower or delayed release cycles. What you can gain by switching:
  - A system that better aligns with your team needs
  - Higher productivity and less developer overhead
  - Shorter time-to-market
- **Vendor Lock-In:** Application server vendors may have organizations tightly coupled to a proprietary ecosystem, making it hard to adopt new technologies or benefit from a more interoperable framework by locking you into proprietary APIs, management tools or other constraints that limit your flexibility. If you feel any change to your system requires heavy rework, your current platform may be dictating more of your architecture than it should. What you can gain by switching:
  - Increased portability and control over your tech stack
  - Ability to adopt tools that works best for you instead of being boxed into a bundle
- **Limited Technical or Technological Support:** Fast ticket response times, access to timely updates, extensive documentation and knowledgeable guidance from engineers when you need it the most is essential. If you're consistently encountering delayed bug or security fixes, shallow knowledge bases or inability to reach the support time, or if you are forced to

modernize your application before you are ready to do so, the “assistance” you’re paying for may not be supporting your goals. What you can gain by switching:

- Faster issue resolution and proactive troubleshooting
  - A transparent product roadmap and frequent release cycle
  - Commercial support that actually meets your needs
  - A say in the features to be included in the middleware
- **Slow Evolution and/or Limited Feature Growth:** Some application servers evolve slowly or not at all, leaving teams unable to take advantage of newer development paradigms. Even performance improvements or observability features may lag behind industry expectations. What you can gain by switching:
    - Access to runtimes optimized for modern enterprise Java, such as low memory footprint, fast startup
    - Built-in support for distributed tracing, metrics and auto-scaling as well as other monitoring and observability tools
    - A platform that evolves alongside your applications and team
  - **Need for Regulatory-Compliant and Auditable Technology:** Development teams in certain sectors face stringent compliance, data protection and auditability requirements. Not all application servers offer the level of transparency, control, or alignment needed to meet these obligations. What you can gain by switching:
    - Runtimes with built-in features for logging, auditing and access control
    - Transparent and certifiable update cycles aligned with regulatory frameworks
    - Documentation and controls that simplify compliance audits and evidence gathering
    - Service and support that aligns with current legislations

## Common Migration Stoppers and How to Address Them

Even when the benefits of migration are clear, hesitation is common for many organizations and teams. In fact, even the most well-intentioned plans can be stalled by a number of perceived road-blocks, such as:

- Fear of the Unknown
- Organizational Inertia & Stagnation
- Cost of Migrating
- Vendor Onboarding
- Vendor Lock-In & Bundling

## Perceived Risks & Fear of the Unknown

For a number of organizations, the biggest issue when it comes to application server migrations is facing the unknown and various ‘what ifs.’ Concerns around stability during as well as after the move are frequent and valid, especially for teams managing high-availability systems. Stakeholders may resist change if they believe migration could interrupt business operations or cause irreversible failures. However, the longer a team wait to migrate, the more challenging the move will be.

### How to Address Risks and Fears:

- Prepare a comprehensive migration plan and engage all involved parties early on
- Use phased rollouts starting with less-critical services or environments
- Adopt blue-green or canary deployment strategies to ensure redundancy and minimize disruption
- Engage with your application server vendor to validate behavior pre-production
- Leverage observability tools to instantly flag anomalies and monitor for early signs of regression

## Stagnation: If it ain't broke, why fix it?

Somehow related to the fear of the unknown, the belief that “staying put is safer” is also common. However, this often masks deeper issues. Operating on outdated or suboptimal infrastructure silently incurs technical debt, higher costs, and a slower delivery pace. Teams end up working around limitations rather than moving forward. In fact, while the current application server may be functional, it could be holding the team back from making progress in key areas like automation, scalability, security or integration with cloud services.

### How to Address Organizational Inertia & Stagnation

- Conduct a total cost of ownership (TCO) analysis that includes licensing, maintenance, support and developer productivity
- Quantify how technical limitations affect delivery velocity, innovation capacity and uptime
- Highlight missed opportunities, such as inability to support new features, technologies or practices

## Cost of Migrating

Concerns around migration cost go beyond licenses. They often include operational expenses, maintenance, time investment, required retraining and potential short-term productivity dips. Also, many teams assume migration will distract from roadmap delivery. Teams need to avoid the perception of migration as a costly, monolithic undertaking that offers unclear returns and stall feature development, strain resources or drag on indefinitely.

### **How to Drive Cost-Effectiveness:**

- Engage with the selected vendor to conduct a TCO analysis
- Frame migration as a strategic investment with clear returns
- Break the effort into incremental, value-driven phases
- Redefine the initiative as a series of manageable, low-risk transition

### **Vendor Onboarding Complexity**

Switching application server often involves learning new tooling, deployment models and APIs, which can slow down teams initially. With development teams already under pressure to deliver, adding learning curves without clear short-term payoffs can lead to resistance.

### **How to Simplify Vendor Onboarding:**

- Select a vendor with robust documentation, an active community and clear upgrade paths
- Run a pilot project or “proof of concept” before the migration project
- Provide targeted internal enablement
- Partner with a vendor that offers extensive onboarding, early enablement support and ensure SLAs are aligned with team goals

### **Vendor Lock-In & Bundling Concerns**

Long-term agility is a critical objective for most technical teams. No one wants to end up locked into proprietary solutions when the reason behind migration was greater flexibility. While migrating to a new runtime may feel like trading one vendor dependency for another, there are a number of actions you can take to maximize operational freedom.

### **How to Avoid Future Vendor Lock-Ins:**

- Choose runtimes and frameworks that align with open standards or open ecosystems
- Package and deploy apps using container standards to preserve infrastructure portability
- Avoid using proprietary SDKs or APIs unless there’s a compelling business reason
- Build abstraction layers or service contracts where possible to isolate platform-specific dependencies

## Overcoming Key Barriers

Even with compelling reasons to migrate and solid answers to common migration fears, dev leads often face practical, high-stakes questions that can halt or slow application server migrations. These concerns must be addressed proactively, though careful planning to support the successful move to a different runtime. The most common barriers revolve around:

- Uptime & Business Continuity
- Team Expertise
- Application Security

### How can I Ensure Business Continuity?

Minimizing risk to service availability during an application server migration is non-negotiable, especially for customer-facing or revenue-critical applications. Unexpected outages or degraded performance can quickly erode trust across both internal and external stakeholders while affecting profitability.

#### Best Practices to Ensure Uptime:

- **Blue-Green Deployments:** Run both the old and new environments in parallel. Route production traffic to the new runtime only when confidence thresholds are met
- **Parallel Environments & Observability:** Gradually expose subsets of users or transactions to the new environment and decouple feature deployment from code deployment. Implement fine-grained metrics and distributed tracing to detect early anomalies before full cutover
- **Immutable Infrastructure & Infrastructure-as-Code:** Define the migration infrastructure using tools like Terraform, Pulumi or Ansible. This ensures reproducibility and eliminates inconsistencies between staging and production.
- **Automated Rollbacks:** Define rollback strategies in CI/CD pipelines to revert deployments quickly in the event of failure. This can speed up recovery and doesn't rely on manual intervention.
- **Pre-Migration Performance Benchmarking:** Compare CPU/memory utilization, size of queues, thread pool usage and startup time between the current and target runtimes to set clear expectations.

### Is my Team Skilled Enough to Address the Complexity of a Migration?

While application server migrations offer a unique opportunity to upskill, even experienced teams can hesitate as these are complex projects per se. Unfamiliar frameworks, platforms, APIs or technologies can make the shift even more challenging.



## Best Practices to Address Any Skill Gaps:

- **Targeted Skills Uplift:** Map the delta between current and future state tech, such as how to transition from Java EE to Jakarta EE or from Oracle WebLogic to Payara Server Enterprise, and plan key training activities around these areas
- **Shadowing & Pair Programming:** Assign seasoned engineers to pair with team members. Encourage mob programming during the early phases to build shared context
- **Pilot Projects:** Select a low-risk application to migrate first. Use this as a template to refine the migration process. Document learnings and templates for use across teams
- **External Expertise:** Consider engaging with your application vendor to discuss customized training and onboarding activities. Thanks to the extensive product expertise and hands-on capabilities, a such partner can help accelerate early decisions, enforce best practices and perform code reviews
- **Self-Service Migration Playbooks:** Create and share living documentation that outlines migration steps, code modification patterns, key processes and FAQs

## Is Migrating Secure?

Nobody wants to expose an application during or as a result of a migration, especially if the software is mission-critical. Thus, security during and following a runtime transition must be proactive, not reactive, and your security posture must be tightly managed across the full software development life cycle (SDLC).

## Best Practices to Ensure App Security:

- **Focus on Teamwork:** Engage security and compliance teams early to review architecture and policy gaps
- **Compliance & Certification Checks:** Confirm your new runtime and hosting environment comply with your industry's regulatory frameworks. Some commercial runtimes or platforms may offer attestation out of the box.
- **Conduct a Security Risk Assessment Before You Begin:** Evaluate risks associated with the transition from your current application server to the new runtime
- **Harden the Migration Environment:** Treat staging and transitional environments with the same security rigor as production, applying the principle of least privilege
- **Scan Everything You Move & Set Up Observability with a Security Lens:** Vulnerabilities in code, containers, and dependencies can migrate if unchecked, but you can set up tools to detect anomalous activity throughout the migration.
- **Plan for Secure Rollback:** Ensure that failure scenarios during migration can be reversed without exposing data or leaving systems in an inconsistent state with version-controlled deployments

## Understanding Your Options

To successfully choose a runtime that better aligns with your application and operational needs, first ask the big question: what are the main results you want to achieve. Is it lower costs? Better performance? Cloud-readiness? Developer experience? Or something else?

Once these goals are defined, it's important to take into account the core elements that the right solution should offer, such as:

- Cost aligns with allocated budget
- Level of compatibility, interoperability and portability required
- Level of support required
- Developer tools and runtime technologies that can effectively help your team
- Future plans for your application
- Regulations and industry standards that your runtime must support

In addition, it can be beneficial to consider the framework, architecture and deployment model that aligns with your requirements. In particular, since migrations are substantial, broad-scope projects, they offer an opportunity to modernize or rethink current applications. For example, it's worth weighing options like:

- Should we stay with Java EE (or shift to the more recent Jakarta EE) or adopt Spring or Quarkus?
- Should our application rely on a monolithic, microservices or modulith architecture?
- Should we have the application on-premises, in the cloud, change the type of cloud or take a hybrid approach?
- Should we self-manage runtimes or offload to a PaaS or fully managed platform?
- How can we better integrate and unify this application with the others we use?

Answering these questions will help you further refine the eligibility criteria for your ideal runtime.

## Planning Your Migration

You have the application, you have the runtime of your choice, your goals and objectives. Now you can start planning a migration. Don't skip this step, as it is where successful migrations start. A well-structured plan reduces surprises and helps to promptly address to any challenge. After clearly defining your objectives and identifying the right middleware solution for your project, make sure you include these steps in your migration plan:

- **Inventory and Assessment:** Document all applications, dependencies, runtime versions and configurations. Identify tightly coupled components and legacy integrations
- **Create a Migration Roadmap:** Break down the effort into manageable chunks
- **Communication Plan:** Keep all stakeholders informed, including your team and the vendor's, as they should all be aligned
- **Execute in Phases:** Avoid migrating all at once or rushing through the project. Instead, migrate one service or module at a time, validate success and iterate.

## Regular Reviews and Evaluations

Migration is not a one-time event, and your current runtime needs may change or evolve as your applications grow or your business adapts to new market needs. Sometimes it's best to adopt a continuous improvement mindset when it comes to the technologies supporting your software, including the middleware it relies on. Post-migration, establish regular reviews to assess:

- Performance and Reliability Metrics
- Developer Feedback
- Support Quality
- Security Posture
- Cost versus Projected Savings
- Runtime Alignment with your Application Needs and Strategy

## Take the Next Step Now

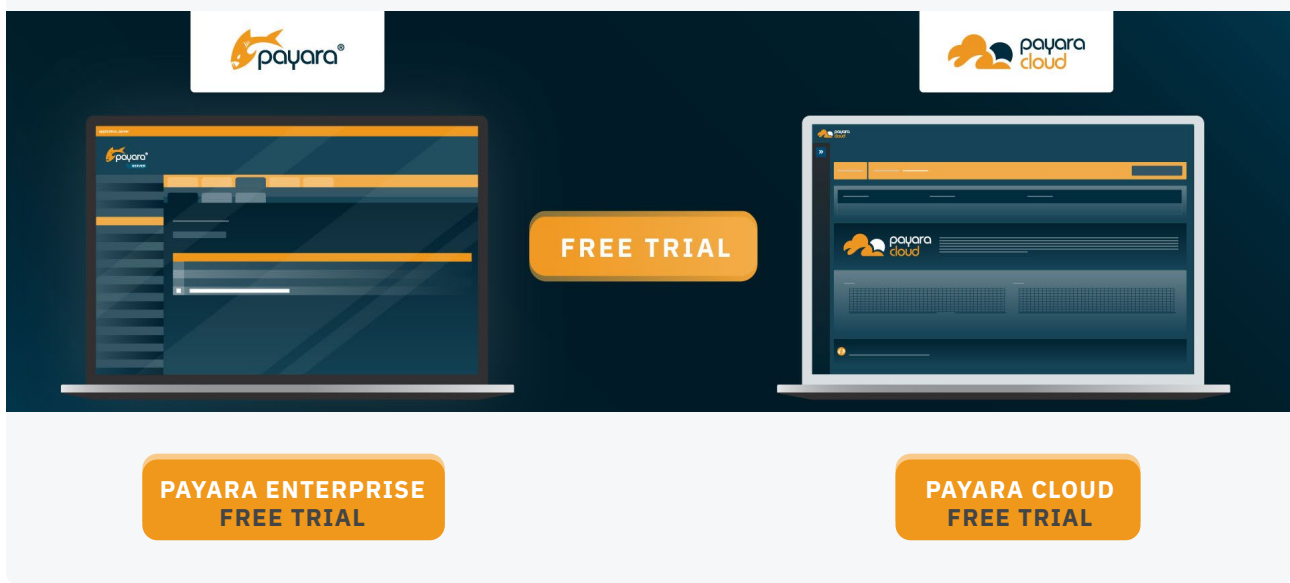
Sticking with a runtime you are unhappy with may feel safe, but it's often the riskiest choice in the long run. Application server and runtime migrations, when planned and executed correctly, can deliver significant value.

If your current application server is holding your business back, it's time to explore better options and move to a better suited solution. Start by evaluating your system's status and limitations and reach out to an experienced vendor offering compatible solution to evaluate your options, investigate about the migration support offered and how to set up develop a suitable roadmap. A strategic approach to application server shifts can turn a daunting task into a transformative opportunity.

At Payara Services, we are committed to help our customers benefit from seamless application server migrations to give them the peace of mind that their business will experience no disruptions while transitioning to any of the Payara Platform Enterprise suite of middleware technologies. We complement our broadly compatible solutions that rely on open standards with comprehensive technical support as well as customizable consulting services. As a result, your team can confidently embrace change and achieve unparalleled enterprise Java application performance.

Ready to power up your enterprise Java application with the right runtime? Let's tackle your migration fears together. Contact Payara today!

## Interested in Payara? *Try Before You Buy*



The banner features two laptops on a dark blue background. The left laptop displays the Payara Enterprise interface, with the Payara logo above it. The right laptop displays the Payara Cloud interface, with the Payara Cloud logo above it. Between the laptops is a central orange button labeled 'FREE TRIAL'. Below each laptop is an orange button: 'PAYARA ENTERPRISE FREE TRIAL' on the left and 'PAYARA CLOUD FREE TRIAL' on the right.



[sales@payara.fish](mailto:sales@payara.fish)



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



[www.payara.fish](http://www.payara.fish)

Payara Services Ltd 2025 All Rights Reserved. Registered in England and Wales; Registration Number 09998946  
Registered Office: Malvern Hills Science Park, Geraldine Road, Malvern, United Kingdom, WR14 3SZ