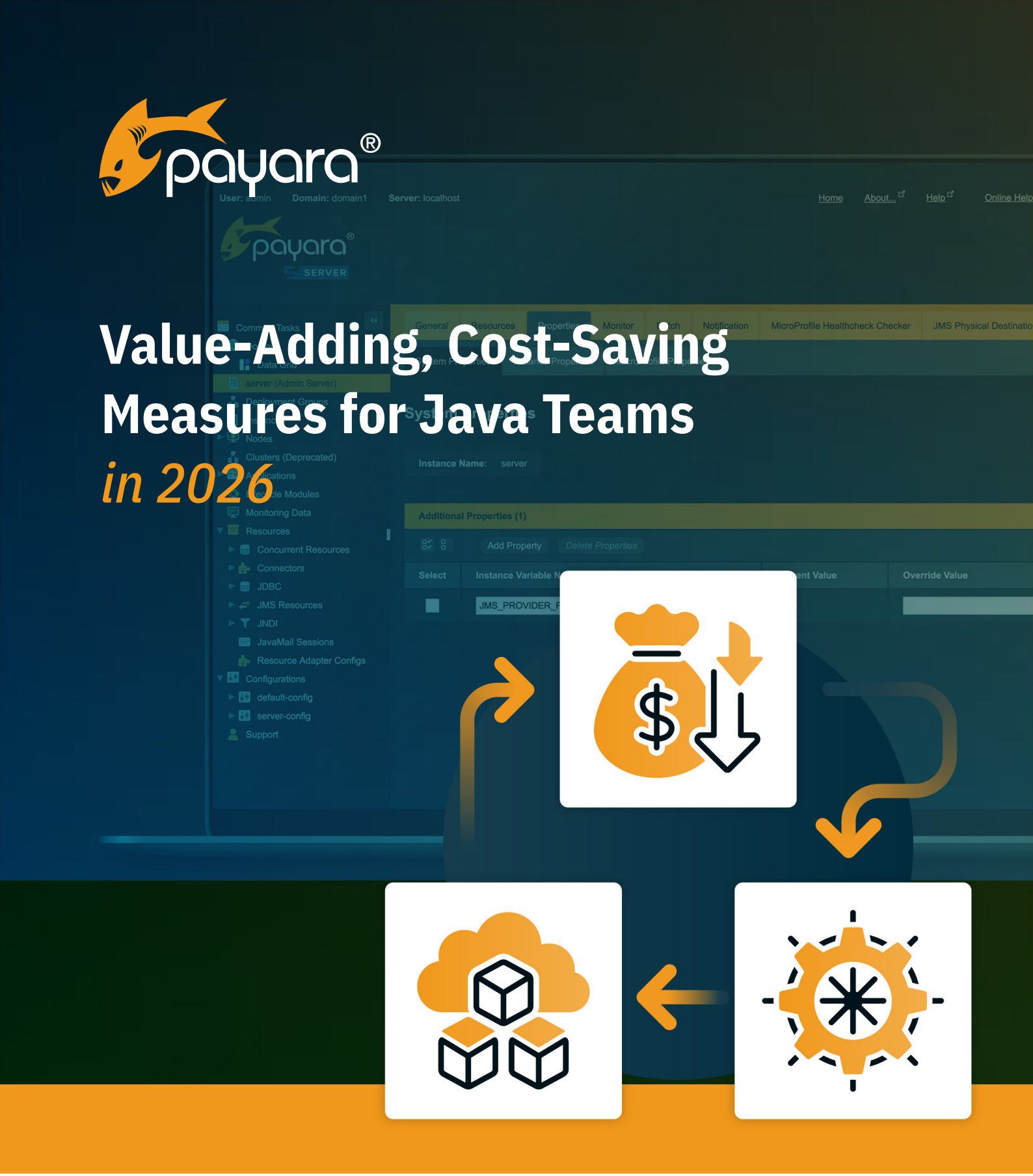
Value-Adding, Cost-Saving Measures for Java Teams *in 2026*



Power Up Your Jakarta EE

Contents

Executive Summary **1**

Introduction **1**

A Complex Landscape **2**

Cost Saving Measures That Add Value To Java Operations **3**

- Cloud Lift-and-Shift Migration 4
  - Move to Cloud Hyperscale 4
- From Monolith to Microservices: Containerization 5
- Kubernetes Orchestration 5

Maximizing the gains **6**

- Full Cloud Migration in Minutes 6
  - Hyperscale Cloud Adoption Without Complexity 7
- Rapid Containerization of Java Applications 7
- Simplified and Automated Kubernetes Management 7

Beyond Cost Savings: Enabling Strategic Efficiency **7**

Conclusions **8**

Executive Summary

Enterprise Java-based systems remain ubiquitous and foundational to most organizations, supporting anything from on-premise, legacy monoliths to next-generation cloud microservices. Yet the cost of delivering, deploying and running them is rising and every team managing Java apps, especially at scale, faces growing complexity. These challenges are primarily driven by the use of multiple Java applications, which are based on different frameworks, environments, geographies. In addition, most organizations have been driving key containerization, cloud migration and modernization projects.

To address the rapidly growing expenses, technology leaders are re-evaluating their budgets. While the reflexive answer has often been to cut costs by reducing licenses or downsizing environments, the focus is now shifting towards achieving measurable savings that directly benefit development teams and strengthen their ability to deliver.

In effect, traditional cost cutting measures have often led to hidden overhead, with Java development teams having to spend considerably more time to support complex environments, repetitive deployment scripts and disjointed toolchains. By identifying how to spend smarter through investments in value-adding, cost-saving solutions that optimize Java workloads, technology leaders can give time back to developers, so that they can succeed in ramping up innovation and release cycles.

This white paper analyzes the underlying cost drivers behind today's enterprise Java ecosystem and offers actionable strategies to optimize both expenditure and operations. Practical use cases illustrate key solutions that can yield considerable annual savings while empowering development teams to focus on rapid innovation and value delivery.

Introduction

Enterprise Java application costs and software delivery performance, especially deployment time, have become critical boardroom issues. Recent research and surveys confirm that companies face mounting financial pressure from direct licensing, runaway cloud infrastructure costs and container management.

While trimming budgets may provide short-term relief, they often impose new burdens on engineering teams. Developers find themselves maintaining a wider surface area with fewer resources, manually orchestrating deployments, and supporting multiple runtime configurations. The cost savings that appear in financial reports can re-emerge elsewhere in the form of slower release cycles and diminished productivity.

In addition, teams worldwide have been involved in large-scale projects, also impacting overhead. In particular, enterprises have invested heavily in modernization initiatives, as cloud deployments, containerization and microservice adoption have become popular features of the enterprise IT landscape. More precisely, the 7th Annual Nutanix Enterprise Cloud index (ECI) Report estimates

that nearly every company (94%) agrees that their organization benefits from adopting cloud native applications/containers and 98% have containerized applications or are at least in the process of containerizing some of their workloads.[1]

The shift has delivered agility and scalability, yet it has also revealed a hidden layer of complexity and, in turn, rising costs, which are some of the issues technology leaders are currently facing. In fact, most organizations need to balance cost savings measures while ensuring teams can innovate at pace. The answer to address both challenges lies not in reducing capacity but in redefining value, finding ways to achieve cost savings that enhance performance and empower developers rather than restrict them.

To succeed in this, companies should first identify the issues and bottlenecks that their enterprise Java teams are experiencing. Currently, these often stem from the breadth of the Java ecosystem.

A Complex Landscape

Java has been around for decades, and many of its early implementations in companies worldwide still support mission-critical operations. As such, virtually any organization maintains portfolios of Java applications at different levels of maturity. These workloads are based on a variety of frameworks, environments and tools that are not necessarily interoperable and compatible. The ubiquity and longevity of Java are part of the reason, as modern engineering teams inherit decades of architectural diversity.

This, in turn, gives organizations flexibility to choose the right tool for each purpose. In effect, each framework, environment, tool offers unique benefits and addresses specific goals, empowering teams to match technologies to business requirements.

While this broad ecosystem enables flexibility and allows teams to choose the most appropriate technology for a given workload, it also introduces substantial operational complexity. This complexity multiplies when modern agile and DevOps practices are applied, as these involve continuous development and integration as well as frequent deployments – all factors that amplify the number of moving parts that must work seamlessly.

Currently, teams are managing multiple pipelines, dependency trees and runtime configurations simultaneously. As a result, DevOps cycles and infrastructure configuration require considerable effort and impose substantial operational overhead, with remote redeployment times considered a barrier across the board. In effect, while there are slight variations among organizations, e.g. influenced by development environment, application architecture type and business size, **redeployments that**

1 Nutanix. (2025). 7th Annual Nutanix Enterprise Cloud Index. Available at: https://www.nutanix.com/enterprise-cloud-index-#key-findings [Accessed 06/11/2025]

exceed ten minutes are common for remote development environments, according to JRebel's 2025 Java Developer Productivity Report.[2]

In effect, enterprise Java teams are often sitting on a paradox. The more applications, frameworks, tools and environments are used, the more the technology stack looks like innovation. The same diversity, however, can lead to complexity that slows the speed at which everything actually moves. Even more, while some new layers were meant to simplify existing ones, they can actually contribute to making the stack grow denser.

These are some of the key reasons why many organizations have been struggling to keep pace with modernization and optimization of their Java stacks. For example, a Gartner survey from 2025 showed that 46% of government CIOs reported their organizations have slowed, paused or rethought modernization investments in the past 12 months.[3] These challenges underscore the need for a more strategic approach to managing Java ecosystems.

Cost Saving Measures That Add Value To Java Operations

As companies look for ways to improve their Java enterprise IT, related activities, and costs, the focus on operational efficiency has intensified. Leading organizations are focusing on achieving sustainable savings, which arise when optimization efforts also enhance the overall performance, scalability, and maintainability of Java workloads.

Within this context, cost-saving measures that add value are those that streamline processes and automate repetitive or resource-intensive tasks while improving the developer experience through solutions that are customized to support Java-based operations. These approaches seek to address current challenges as well as eliminating inefficiencies in deployment, monitoring, and infrastructure management, resulting in measurable productivity gains and faster delivery cycles.

In addition, Java-centric methods and solutions offer a valuable asset, as they can leverage the functionalities and capabilities that are specific to Java while offering opinionated setups that further reduce configuration.

In practice, these value-driven approaches can be applied across several operations and activities. The next sections explore concrete scenarios where organizations can realize value-added savings, illustrating how valuable solutions can deliver both financial and operational benefits.

2 perforce JRebel. (2025). 2025 Java Developer Productivity Report. Available at: https://www.jrebel.com/resources/java-developer-productivity-report-2025 [Accessed: 06/11/2025]

3 Robert Stoneman. (2025). Build Momentum for Application Modernization in Government. Available at: https://www.gartner.com/en/articles/government-modernization [Accessed: 21/10/2025]

Cloud Lift-and-Shift Migration

Lift and shift cloud migrations involve taking a Java workload exactly as it is, without any changes, and moving it to the cloud. On paper, this migration strategy offers an excellent option for teams that want to upgrade their applications quickly, with as little rework and disruption as possible.

Unfortunately, lift and shift projects aren't necessarily straightforward. Also, without any automation tool, **the process can take months** and incur frequent roadblocks, as teams conduct assessments, plan migrations, provision cloud resources, execute the migration, validate functionality and retire legacy systems.

A well-designed, Java-centric Platform Engineering solution that is easy to integrate and offers built-in application monitoring, management as well as automation tools can support developers in such operations while reducing time and resources needed. In practice, this means that **operations that would traditionally take weeks or even months can be completed in a fraction of the time, within an hour**, while eliminating many of the common roadblocks associated with cloud transitions. Beyond time savings, this efficiency translates into substantial financial benefits. Estimates indicate potential cost reductions exceeding **USD 150,000 per migration**.

Moreover, the use of such platforms allows development and operations teams to focus on higher-value activities, such as performance optimization and feature enhancements. As a result, the investment also contributes to long-term operational agility and resilience in cloud-based Java environments.

Move to Cloud Hyperscale

Similar to cloud migrations are modernization projects involving the move to hyperscale cloud providers, which can help transform application agility and scalability. While hyperscale cloud platforms deliver agility and scalability, the transition process often introduces hidden overhead, particularly around infrastructure setup, service configuration and tuning.

Typically, **these migration projects can take from a few days for smaller or simpler workloads, up to multiple weeks for complex, data-intensive environments**, often due to extensive infrastructure configuration. On average, **it is estimated that 264 hours are spent on such modernizations**. These projects also often involve several hours of planned downtime, impacting application availability and user experience.

Teams can substantially optimize the process by utilizing a platform that abstracts infrastructure setup complexity and application monitoring through meaningful automation. As a result, it is possible to **slash delivery time from hundreds of hours to minutes**, leading to considerable cost savings, while reducing overhead and cognitive load on developers.

From Monolith to Microservices: Containerization

Besides lifting and shifting to the cloud, most organizations are involved in more ambitious and resource intensive Java application modernization projects. One of the most popular consists of the move from monolithic to more modern microservices architectures, which involves the containerization of existing applications to create smaller, independently deployable services.

However, traditional containerization of Java applications involves multiple manual steps: configuring the application, creating Docker files, building Docker images, running containers, testing as well as managing multi-container deployments and pushing the image to a registry. **This process typically spans several weeks**.

The use of automated, developer-centric tools can **reduce containerization time down to three to five days**, saving an estimated **USD 69,581 per application**. For example, users can benefit from automatic clustering and data grid operations that simplify the creation, testing and management of containers.

Kubernetes Orchestration

As the popularity of containers in Java workloads continues to rise, more development teams begin to adopt container management/orchestration solutions, such as Kubernetes (K8s), to deploy their applications. While the technology certainly helps, its management, traditionally through YAML, is itself time- and resource-intensive.

For example, teams must build images, configure deployments and services, implement health probes as well as integrating monitoring. Even more, developers are likely deploying multiple microservices and variations, considerably raising the workload. When using conventional workflows, the process can take weeks.

Currently, there are a number of platform engineering solutions that can automate Kubernetes management and deployment. For example, teams can use templating tools or technologies that offer a higher abstraction, such as Infrastructure as Code (IaC) or managed runtimes.

These options can free up considerable time for developers, who can shift their focus from infrastructure setup to application enhancement. By empowering developers, organizations can benefit from productivity gains while driving valuable cost savings.

Maximizing the gains

The use cases discussed in the previous section highlight how value-adding tools can deliver considerable cost savings while improving operations as a whole, with the estimates provided clearly showing what good solutions can offer. More refined technologies can build on these results and extend the benefits further, enhancing both efficiency and operational resilience.

Payara Qube exemplifies such an approach. It addresses the common challenges faced by enterprise Java teams: complexity, fragmented toolchains, lengthy deployment cycles and escalating resource needs to handle infrastructures. It does so through a Java-centric platform designed to help teams deploy, monitor and scale workloads across different environments and frameworks through a unified interface. By integrating capabilities that traditionally require multiple products or manual coordination, Payara Qube consolidates operations across the entire Java application lifecycle, from deployment and monitoring to scaling and optimization.

This unified architecture allows teams to execute tasks that previously involved separate environments and extensive configuration within a single interface. Automated provisioning, built-in monitoring, and managed runtime support minimize the need for custom scripts or third-party integrations, significantly reducing both operational overhead and cognitive load on developers. Even more, as a Java-oriented solution, it provides an opinionated platform that can further abstract complexity.

In addition, by centralizing observability, scaling, and performance tuning, Payara Qube enhances operational stability and resilience, helping organizations maintain service quality even as they accelerate delivery cycles. This combination of cost efficiency, speed and control enables teams to focus on innovation rather than infrastructure maintenance. As a result, organizations can further improve the time- and cost-savings compared to traditional platforms.

Full Cloud Migration in Minutes

A streamlined lift-and-shift approach can reduce cloud migration times and save substantial resources. With Payara Qube, those benefits reach their full potential.

A complete cloud migration can be achieved in under 15 minutes while reducing risks and bottlenecks. Automation manages the provisioning, scaling and optimization of the target environment, while integrated monitoring ensures immediate visibility. The estimated savings for organizations adopting this solution are **USD 191,248 per migration**.

Hyperscale Cloud Adoption Without Complexity

Payara Qube also streamlines the adoption of cloud hyperscale solutions. Teams simply package and deploy their Java applications, and **within 15 minutes, the workloads are fully operational at hyperscale capacity**, complete with automated monitoring and scaling policies. This reduction in complexity leads to estimated savings of **USD 19,798 per application** and helps teams to deliver elasticity without compromising on architectural control.

Rapid Containerization of Java Applications

Payara Qube also compresses containerization activities into minutes. Teams upload a WAR file, deploy it and immediately benefit from automated clustering, image creation and runtime monitoring. The result is a reduction in effort equivalent to **USD 82,479 in savings per application**. Perhaps more importantly, they can benefit from a predictable, repeatable way to containerize applications at scale without bottlenecks or configuration drift.

Simplified and Automated Kubernetes Management

Finally, with Payara Qube, the complexity of Kubernetes and YAML configuration, service definitions and health checks is handled automatically.

Deployments that previously took weeks can be completed in minutes, supported by built-in monitoring, scaling and self-healing features. These efficiencies equate to estimated savings of **USD 82,723 per application**, while freeing Java teams to focus on reliability and optimization rather than low-level infrastructure tasks.

Beyond Cost Savings: Enabling Strategic Efficiency

The examples so far have shown how abstraction-oriented technologies can deliver targeted improvements that can simultaneously provide meaningful savings and productivity gains. However, these only addressed individual issues. To address the whole picture and achieve maximum impact, organizations need a solution that not only accelerates each individual process but unifies and optimizes them end to end.

While a good solution can reduce effort and cost at specific stages, a great solution transforms the entire Java operations lifecycle. It eliminates repetitive configuration across all deployments and redeployments, integrates monitoring and observability while offering developers an immediate, unified, and consistent path from build to production.

By removing the operational friction that slows down development and deployment, organizations not only save money but also improve agility, with teams able to deliver more frequent, more value-adding releases, respond faster to change, and maintain predictable, consistent performance at scale.

This is where a solution like Payara Qube demonstrates the difference between incremental savings and full-scale efficiency. In effect, the Java-centric, fully managed runtime offers a comprehensive platform that can simultaneously address the top challenges that teams handling enterprise applications face. Even more, it does so while delivering superior results. Ultimately, true cost efficiency comes from smarter systems that allow teams to do more with less effort.

Conclusions

Across sectors, technology leaders are looking at ways to reduce costs, with the most strategic realizing that the goal is not simply to spend less, but to spend smarter. Intelligent cost savings that stem from simplification, automation, and platform consistency produce gains that compound over time. Instead of demanding more from developers, they remove friction, freeing teams to focus on innovation and problem-solving.

This is particularly relevant for Java-based systems, where operational variance is high. Containerization pipelines, deployment scripts, and monitoring stacks differ not only by framework but by environment and team. Each layer introduces manual effort and risk. A dedicated, holistic platform, such as Payara Qube, that simplifies developer workflows and reduces cognitive load is therefore an asset.

The positive impact extends beyond operations. When friction decreases, creativity increases. Developers gain the bandwidth to refine applications, optimize performance, and experiment safely. Ultimately, organizations can gain a culture of efficiency grounded in confidence. What begins as a financial decision, e.g. reducing infrastructure costs, becomes a strategic, value-adding approach.

info@payara.fish

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

www.payara.fish