10 reasons to build your next Java application with Quarkus

Contents

Modernize your Java application development

There are an estimated 16.5 million active Java developers worldwide.[1] In fact, 49% of all active software developers use Java, making it one of the most popular languages for software development.[1]

Java frameworks—templates of prewritten, reusable code—let you create Java applications with less effort. They can include everything from libraries, compilers, and software development tools to predefined classes and functions that process input, manage hardware, and interact with system software. Starting with a framework, developers write and add custom code that implements the specific business logic of their applications.

Even so, traditional Java frameworks were built for monolithic applications running on dedicated servers in on-site datacenters. These frameworks often result in long startup times and large memory requirements for applications. They can also be difficult to use for workloads that will run within Kubernetes containers in hybrid cloud environments.

That's where Quarkus comes in. Quarkus lets Java developers use modern, industry-standard libraries and specifications to build applications for today's hybrid cloud infrastructure. A Kubernetes-native Java framework, Quarkus is tailored for OpenJDK HotSpot and GraalVM. With extremely fast boot times and low memory use, it offers near-instant scaling and high-density deployment on Kubernetes-based container platforms. Quarkus was introduced to the open source community in 2019 with the goal of making Java the leading platform in Kubernetes and serverless environments. Since then, many organizations have benefited from Quarkus and joined the development effort.

There are 2 versions of Quarkus: the community version and the supported Red Hat® build of Quarkus. You can always access the latest developments in the community version, and new features are released in the Red Hat build approximately every 6 months. Red Hat provides enterprise support for each release until the next major release is delivered, followed by 6 months of maintenance support. Fixes for selected common vulnerabilities and exposures (CVEs) and bugs are regularly backported to supported streams.

This e-book discusses 10 reasons to use Quarkus for your next Java application.

> With Quarkus, [we] could run **3 times denser deployments** without sacrificing availability and response times of services.[2]
>
> **Thorsten Pohl**
> Lufthansa Technik AVIATAR Product Owner Automation & Platform Architect

1 *SlashData. "State of the Developer Nation Q3 2022, 23rd Edition," November 2022.*

2 *Quarkus user story. "Lufthansa Technik AVIATAR experiences significant cloud resources savings by moving to Kubernetes-native Quarkus," February 2020.*

Reason 1:

Develop with a framework designed for the cloud

Quarkus is purpose-built to run applications in hybrid and multicloud environments, helping you accelerate cloud adoption initiatives and meet corporate modernization mandates. Quarkus optimizes applications for low memory use and fast startup times, making it ideal for modern, cloud-based microservices and serverless architectures. In fact, compared to traditional cloud-native stacks, first response times for applications running on Quarkus can be more than 200 times faster, while using almost 90% less memory.[3] These memory optimization capabilities also let you run more applications using fewer cloud resources, helping to reduce your overall cloud spend.

AWS, Microsoft Azure, and Google Cloud extensions make it easier to integrate Java applications with these services and benefit from your existing investments and relationships. And because Quarkus is fully integrated with Red Hat OpenShift®, it is easy to deploy Quarkus-based applications on Red Hat OpenShift Service on AWS and Microsoft Azure Red Hat OpenShift.

Improve Java application performance

- Learn how Quarkus helps **optimize application performance**. →
- Read about the methodology used to **measure Quarkus performance**. →

Reason 2:

Create Kubernetes-native applications

Quarkus simplifies deployment of containerized applications, allowing operations teams to streamline processes and give developers more self-service capabilities. Quarkus is a Kubernetes-native framework designed, built, and optimized for containers, so you can deploy applications without a detailed understanding of the underlying infrastructure. Extensions for Kubernetes and Kubernetes-based container platforms let you perform single-step deployments using Jib, Docker, and Source-to-Image (S2I) with minimal configuration required. And if you have Function-as-a-Service (FaaS) projects, this single-step deployment model lets you deploy Quarkus applications as Knative serverless functions.

3 Quarkus website, *accessed 4 December 2023.*

Red Hat's build of Quarkus delivers production-ready capabilities for cloud-native applications running in Red Hat OpenShift. It is built, tested, and verified with Red Hat OpenShift and Red Hat Enterprise Linux®, giving you a trusted software foundation for critical Java workloads. Red Hat experts provide production and development support for tested integrations and supported configurations.

Quarkus extensions provide native Kubernetes and Red Hat OpenShift experiences. For example, the kubernetes-config extension lets you use Kubernetes ConfigMaps and Secrets as a configuration source. Using the Kubernetes client, applications can read ConfigMaps and Secrets directly from the Kubernetes application programming interface (API) server. Other extensions let you generate Red Hat OpenShift resources from annotations, build container images of your application using Red Hat OpenShift, and develop Red Hat OpenShift Operators.

Learn about and access Quarkus extensions

A large catalog of Quarkus extensions is available to enhance your applications. Learn about key extensions for Red Hat OpenShift and Kubernetes environments:

- **kubernetes-config →**
- **quarkus-openshift →**
- **quarkus-container-image-openshift →**
- **quarkus-openshift-client →**

Reason 3:

Streamline development

Quarkus removes many inconveniences that developers often face, so you can rapidly write high-quality, innovative Java code. The Quarkus Language Server supports popular integrated development environments (IDE) like Microsoft Visual Studio Code, Eclipse IDE, and IntelliJ IDEA. You can use your preferred IDE in the development process that works best for you.

Quarkus Tools for Visual Studio Code brings Quarkus features directly to Visual Studio Code to help you develop faster. Generate Maven-based Quarkus projects via wizards. Create new Quarkus resources and tests using snippets. Simplify debug sessions with commands that create and add Visual Studio Code task and debug configuration, and automatically start Quarkus applications.

Using the Dev Services capability, Quarkus can automatically provision your application's required services—including databases, Apache Kafka Streams, and single sign-on (SSO)—in development and test mode. Simply include an extension and Quarkus will automatically start the related services and connect them to the application correctly, allowing you to focus on the business logic of your application.

Access Quarkus tools for your preferred IDE

- Download **Quarkus Tools for Visual Studio Code**. →
- Download **Quarkus Tools for Eclipse IDE**. →
- Download **Quarkus Tools for IntelliJ**. →

A developer user interface (UI) included with Quarkus, **Dev UI** provides visibility into your application as you develop it. With it, you can quickly understand all loaded extensions, including current status and documentation. You can also view and change configurations; manage and visualize continuous testing; and see Dev Services and build information, as well as other real-time logging.

The **Quarkus Live Coding feature**—enabled during development mode—makes recompiling and redeploying applications nearly instantaneous. When you refresh your browser, Quarkus scans your workspace to identify any modified Java, resource, or configuration files; compiles any changes; and refreshes the application. The refreshed application then services your request in seconds, while Quarkus reports any compilation or deployment issues in an error page. You can also run development mode **remotely** in a container environment like Red Hat OpenShift. This lets you develop and debug applications in the same environment used for production. Changes made locally are immediately visible in the container environment, so you can test your application under realistic conditions.

Finally, Quarkus integrates with popular **continuous integration/continuous deployment (CI/CD)** tools like **Red Hat OpenShift Pipelines** to automate builds, testing, and deployment of Quarkus applications.

Reason 4:
Take your skills to the next level

Quarkus helps you apply your existing Java knowledge, skills, and expertise to deliver new cloud-based applications based on modern software architectures and traditional Java libraries and standards. Developers familiar with Spring and Java EE can quickly get started with Quarkus because all are Java application frameworks for complex, high-performance applications. All 3 frameworks use similar concepts like dependency injection and offer a variety of components for application development.

There are many resources available for organizations that want to move applications from Spring to Quarkus. **Red Hat's Migration Toolkit for Applications** analyzes existing Spring Boot applications and recommends migration actions using Quarkus Spring Extensions. Applications that use these extensions are small, fast, and memory efficient. And, in some cases, you may not need to change the source code at all.

Learn more about moving from Spring to Quarkus

- Read the **Quarkus for Spring Developers** e-book. →
- Read the **Quarkus for Spring Developers** blog. →
- Download the **Migration Toolkit for Applications**. →

Reason 5:

Build with proven standards

The Quarkus programming model builds on proven, official standards and well-known frameworks—like Jakarta EE, Eclipse MicroProfile, and Eclipse Vert.x—to support compatibility and scalability across your environment. Using these frameworks, you can spend less time learning and more time innovating. For example, Quarkus uses a dependency injection solution based on Contexts and Dependency Injection (CDI). You can define REST endpoints with Jakarta RESTful Web Services (JAX-RS) annotations, map persistent entities with Jakarta Persistence (JPA) annotations and declare transaction boundaries with Jakarta Transactions (JTA) annotations, or configure and monitor applications with MicroProfile.

Reason 6:

Join a community of professionals

The Quarkus ecosystem is built, powered, and supported by a dynamic, fast-moving community of developers and backed by Red Hat, a leader in open source technologies. Since its first release in 2019, the open source Quarkus community has grown to more than 850 contributors worldwide.

Today, Quarkus includes more than 600 extensions to help you configure, boot, and integrate frameworks and technologies into your applications. Extensions also provide information to GraalVM so your applications can compile natively. The Quarkus extension framework offers guidelines and examples for creating your own extensions. You can contribute your extensions to the Quarkiverse GitHub, a repository including build, continuous integration (CI), and release publishing for community-developed Quarkus extension projects. Extensions hosted in Quarkiverse can be included in the code.quarkus.io extensions catalog and the Quarkus command line tools.

Participating in the Quarkus community allows you to influence the direction of future releases. See the public Quarkus roadmaps and learn more about release plans on the Quarkus release planning Github page. →

Reason 7:

Boost Java application security

Quarkus can help you enhance the security of your Java applications. The Red Hat build of Quarkus—including the core platform and all supported extensions—is created via **security-focused software delivery processes**. It also ships with software bill of materials (SBOM) files that include both supported and community extensions. With SBOMs, you can perform vulnerability and license analyses to evaluate the risk of using software, ensure all components are up to date, and avoid potentially harmful software.

Red Hat's Dependency Analytics extension for Visual Studio Code automatically checks your application dependencies, flags common vulnerabilities and exposures (CVEs), and suggests remediation actions to help you increase the security of your Quarkus applications. It can also suggest project level licenses and check for conflicts between dependency licenses to help you ship your software with the correct licensing.

Finally, the **Quarkus Security framework** provides cloud-based identity and access management capabilities with multiple built-in authentication mechanisms and security extensions. Supported **security mechanisms** include Open Authorization 2 (OAuth2), Security Assertion Markup Language (SAML), OpenID Connect (OIDC), Web Authentication (WebAuthn), Kerberos, and secure sockets layer/transport layer security (SSL/TLS). Quarkus provides Spring compatibility via the **Quarkus Extension for Spring Security API**.

Reason 8:

Meet sustainability goals

Quarkus can help reduce the environmental impact of IT operations and meet corporate sustainability goals. Quarkus-based applications feature low memory use and fast startup times, so you can use fewer hardware resources to run your workloads. Accordingly, the power required—and corresponding carbon emissions—is significantly less with Quarkus for the same application performance, compared to traditional Java frameworks.

To show how **Quarkus helps reduce carbon emissions**, Red Hatdevelopers deployed an application on cloud resources and loaded it with 800 requests per minute over 20 days. Using a traditional cloud-native framework, the application required an instance with 2 processors and 4GB of memory. Running on Quarkus, the same application needed an instance with only 1 processor and 1GB of memory. Based on estimates using publicly available data sets, over the 20 day experiment, the application running on Quarkus saved 1,313 kilograms of CO_2 emissions compared to the traditional framework. And because the application running on Quarkus used fewer resources, the cloud costs associated with it were less than half the cost of running on the traditional framework.

Read **How to write greener Java applications** to learn more about this experiment. →

Reason 9:

Build applications for the edge

Quarkus lets developers use familiar open source Java technologies to build applications for resource-constrained edge environments. Quarkus applications use resources efficiently, which is crucial when memory and processing power are limited. They also consume much less energy than traditional Java applications, making them ideal for battery-powered devices. Fast boot times mean that applications that process real-time data can start up and respond to events quickly. And support for real-time and event-driven architectures lets you process data and immediately respond to critical events.

Reason 10:

Integrate AI/ML models

Quarkus helps Java developers rapidly and simply create intelligent applications that incorporate artificial intelligence and machine learning (AI/ML) models. For example, you can integrate data storage systems, message queues, or cloud services with your AI/ML models. Create data pipelines to process and transform data for use in AL/ML workflows. Deploy your AI/ML models as Quarkus services in containerized cloud environments, including AWS, Microsoft Azure, and Google Cloud. Respond to events or stream data in real time with support for event-driven architectures. Or integrate Java ML libraries like Deeplearning4j and Weka into Quarkus applications to perform inferencing tasks.

Plus, with Red Hat OpenShift AI, you can deploy your Quarkus-based applications on the same platform that you use to train, deploy, and monitor your AI/ML workloads. Red Hat OpenShift AI is a fully supported environment based on machine learning operations (MLOps) best practices. It provides cross-infrastructure consistency, so you can deploy your applications in containers in datacenter, edge, and cloud environments.

Learn about integrating generative AI into Quarkus applications

- Read the **When Quarkus meets LangChain4j** blog. →
- Watch the **Langchain4j and Quarkus fireside chat** video. →

Ready to get started with Quarkus?

Quarkus lets Java developers use modern libraries and specifications to build applications for today's hybrid cloud environments. Boost application performance and security, streamline development, and extend your Java skills to new levels. With Quarkus, you can be ready for whatever the future holds.

Access the Red Hat build of Quarkus.

Download Quarkus

Generate your first project using Quarkus.

Generate an app

Try Quarkus in a developer sandbox.

Try Quarkus

See Quarkus in action

Many organizations are already experiencing the benefits of switching to Quarkus.

vodafone

Vodafone Greece cut memory resource consumption by 50% and startup times by nearly 25% for a key component in their telecommunications architecture.

Read the blog post. →

bankdata

Bankdata found that an application running natively on Quarkus had up to 194x faster boot-up times and used up to 4x less memory per call compared to Spring Boot.

Read the blog post. →

CANTON DU VALAIS
KANTON WALLIS

The cantonal administration Etat du Valais increased developer productivity by 3x and reduced cluster usage by 40% when they adopted the Red Hat build of Quarkus.

Read the case study. →

Red Hat