How It Started

How It's Going

I got my first programming job while I was still in high school and it was writing C++ code for a XENIX system connected to a Xerox high-speed laser printer. That was 1992

At the turn of the millennium I was working for tech start-ups using PHP and Java and Linux, where I first started collaborating with the Open Source community.

Since then, I have expanded my experience to be a contributor to Apache Camel, Eclipse Vert.x, and Quarkus among many other open source projects.

So, 30 years of experience in the industry and I still learn tons of new things every day, and I hope each of you do as well. It's one of my favorite parts of this job!

Application Servers are expensive and we cannot innovate as quickly...

Hopefully by understanding a little about my background, you'll be more open to understand why I am so excited about what is changing in our industry over the last decade.

PUBLIC

SOA is dead, nobody does Enterprise Service Bus anymore, but integration needs remain

Red Hat

When was the last time you had someone talk about SOA or ESBs? More likely you hear about Lambda and Step Functions, but orchestrating a large number of functions can be difficult and add lots of operational overhead. Yet, we still need a way to connect disparate systems, exchange and transform data, handle events, and more… Is there a way we can leverage what's good about modern cloud capabilities without giving up what was great about application server platforms like Karaf, ServiceMix, OSGi, etc…?

PUBLIC

Apache Camel & Quarkus

Supersonic, Subatomic Integration

Deven Phillips
Senior Architect
Runtimes Practice

I'd like to suggest that Apache Camel using the Quarkus runtime could be just the solution which balances those value propositions… Especially when you deploy using a container runtime like Kubernetes where you can have lots of scheduling and deployment flexibility while retaining the centrally configured simplifications of legacy application servers.

PUBLIC

Unpopular Opinion....

Kubernetes is the Enterprise Application Server for the Containerized world

Centrally controlled security policies

SSL/TLS offloading

Centrally controlled configuration and secrets

Centralized observability

Scaling, clustering, load balancing

Service discovery

Centrally controlled security policies

SSL/TLS Offloading

Centrally controlled configuration and secrets

Centralized observability

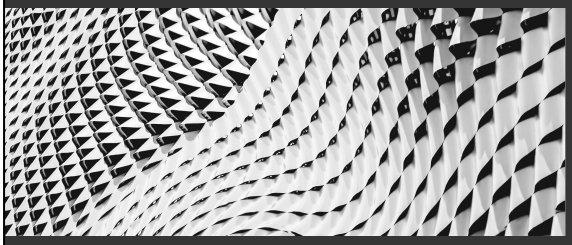
Scaling, clustering, load balancing

Service discovery

Red Hat

Many of you might say “But, Kubernetes is so much more than . . . “ Sure, but so were app servers… Most organizations failed to make use of all of the capabilities of their app servers and still do to this day. The main difference between app servers and Kubernetes is added flexibility… We can run any language, any framework, and we can be more granular in our scheduling of CPU, memory, and storage resources.

Apache Camel Evolves



Camel today is the same as it always has been from a developer perspective.

What has changed is the underlying complexity hidden behind the simple APIs and DSLs of Camel

Slow and steady innovation in products is generally more sustainable than big bang disruption

Open Access News | Research & Innovation News

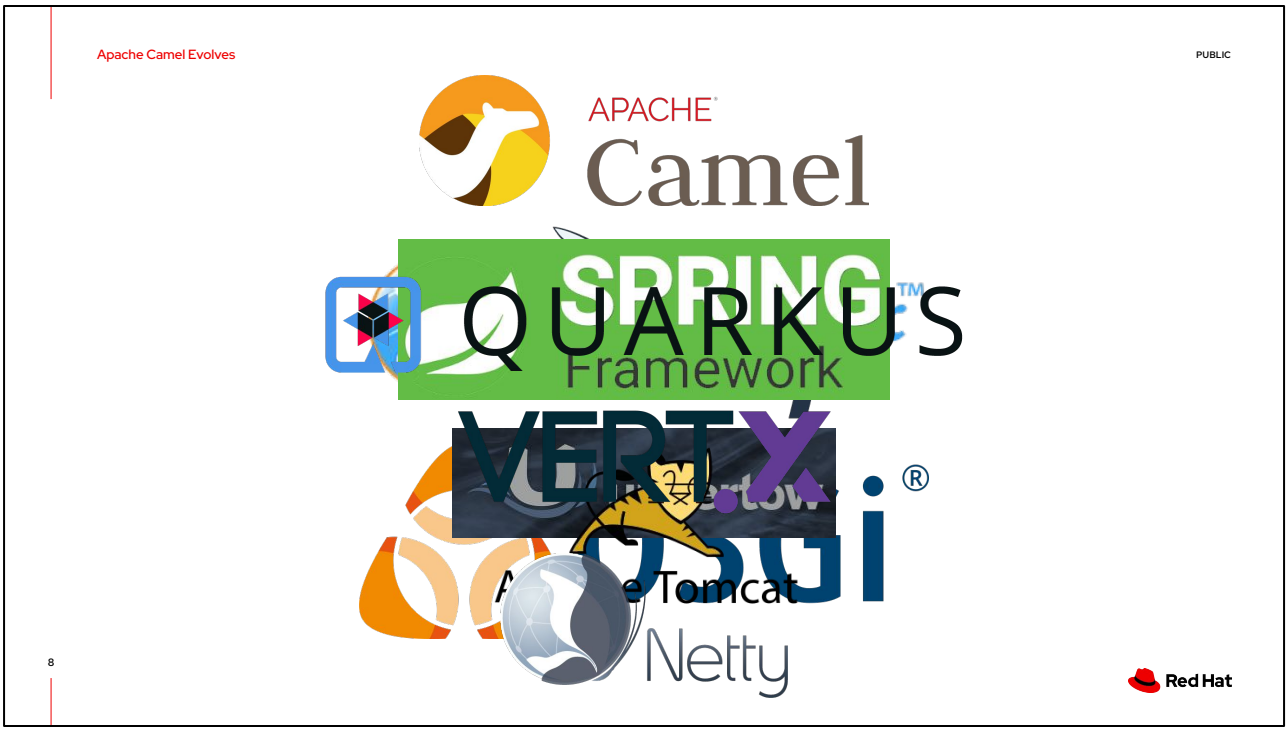
Why slow and steady wins the innovation race

June 22, 2021

© *Blackboard373*

Source:
https://www.openaccessgovernment.org/why-slow-and-steady-wins-the-innovation-race/113605/

Projects like Apache Camel are a wonderful example of this concept. Apache Camel doesn't change massively in large burst, but slowly and steadily while allowing it's users to adopt new features as they are able. Disruption in the industry typically means that the development teams currently using one solution have to stop work for a time and learn an entirely new paradigm before they can begin to be productive again.



ANIMATED: Another thing which is important to keep in mind is that Apache Camel really isn't changing, it is just adapting to a new and arguably better runtime.

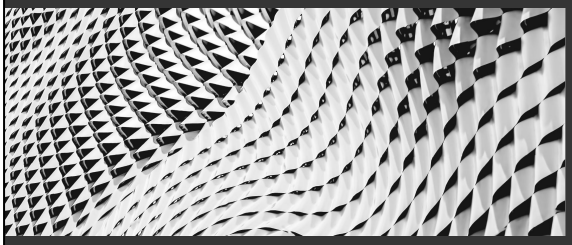
Camel in the old days on ServiceMix/OSGi was limited to thread pools

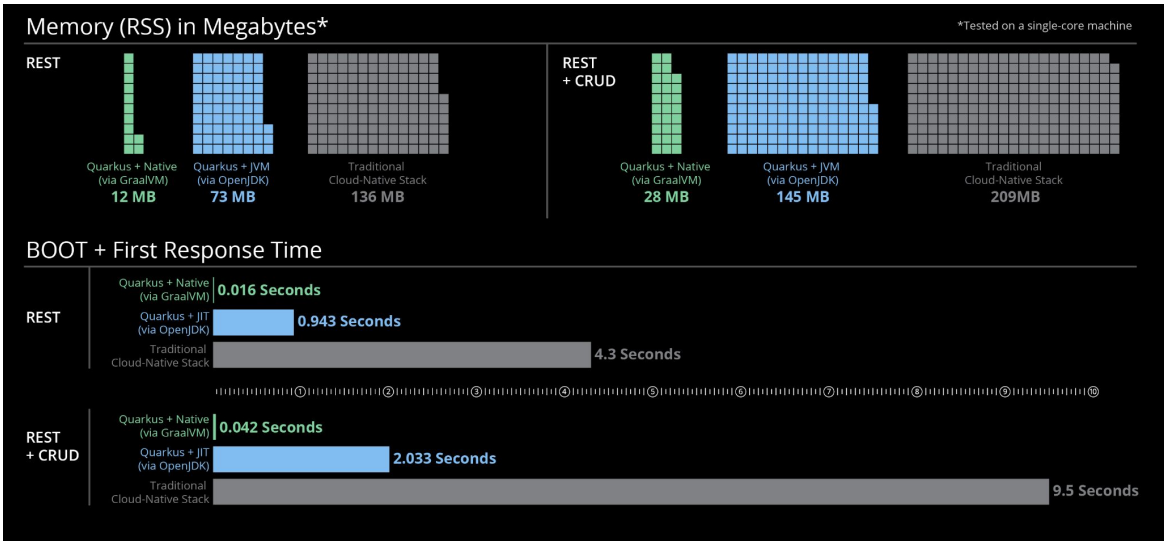
It Evolved to run on top of Wildfly, which was still limited to Thread Pools

Then onto Spring and Tomcat which is also limited by thread pools

But with Quarkus, we are operating on top of a stack which is completely non-blocking from top to bottom. Starting with NIO.2, with Netty, and Vert.x, there are such fewer limitations on how we process.

Quarkus Overview



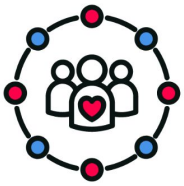


Source:
https://quarkus.io/

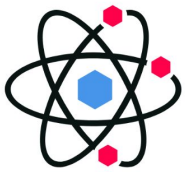
But sometimes it feels like these evolving and iterative innovations hit a critical mass and lots of things improve by a large amount and though it is a major change, it is not disruptive. This is how I see Camel on Quarkus. We get to use Camel in pretty much the same ways we always have, but we get to benefit from all of the evolution and innovation which has summed up into this new option. We get faster start times, lower memory footprints, better developer experiences, and more. It's the culmination of a number of different key technologies evolving and integrating to result in a major improvement.

Developer Joy

Kubernetes-native



Best of Breed Libraries and Standards

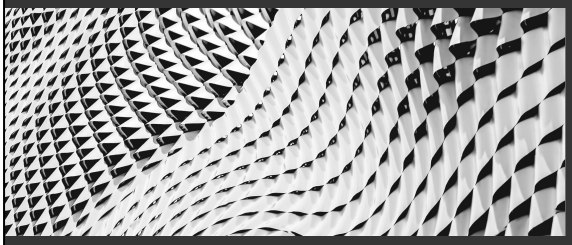


Imperative and reactive code

Source:
https://quarkus.io/

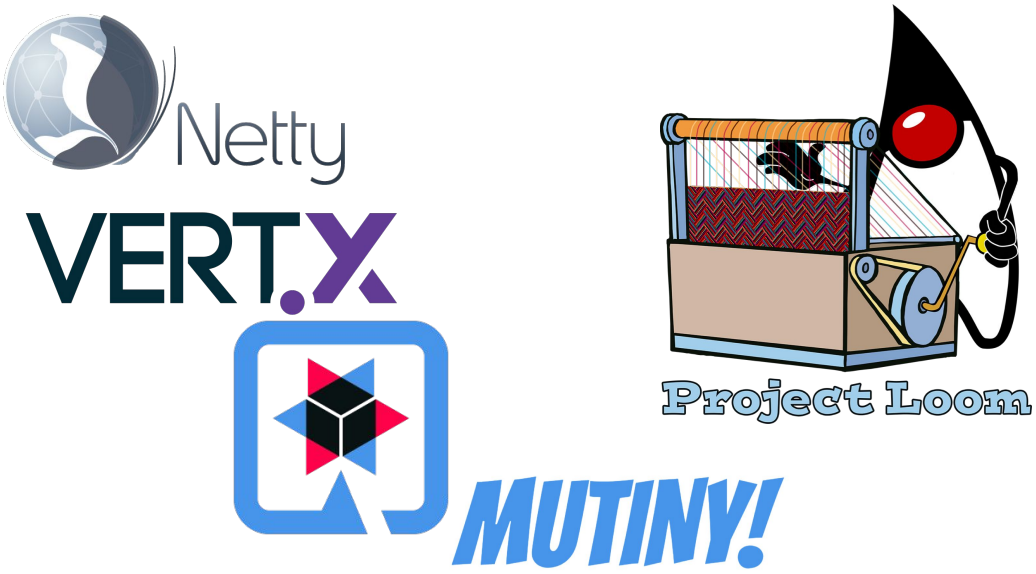
Quarkus brings a lot of the same capabilities we are used to in traditional Java development. Dependency Injection is handled automatically with CDI, we have access to the same Java libraries and ecosystems. The difference is that we have used those traditional interfaces and functionalities as a facade on top of a truly innovative runtime based on Eclipse Vert.x. Instead of the traditional thread and blocking I/O approach which underpins most of our existing technologies, Quarkus replaces the underlying I/O with a fully reactive toolkit but at the same time abstracts that complexity away from us behind the facade of familiar APIs. In addition, new features like live-reloading, DevServices, and native-image support allow us to be more productive with less effort.

Apache Camel Quarkus



The Apache Camel components and APIs that you already know and love running on a next-generation and future-facing runtime

Red Hat



Source:
https://quarkus.io/
https://quarkus.io/blog/quarkus-2-10-0-final-released/

Starting with Netty, which is a nice and lightweight abstraction on top of Java's NIO.2 APIs for non-blocking I/O operations.

Further improved by Eclipse Vert.x which gives us all of tools for a fully reactive Java application

Layer on Quarkus which presents a familiar and comfortable programming model based on JakartaEE Microprofile

And to go completely reactive it supports SmallRye Mutinty as a simplified experience for reactive streams.

All of this, and the underlying runtime is already providing support for the forthcoming Project Loom Virtual Threads.

This means that you are running on an extremely fast and efficient runtime today and ready to automatically take advantage of evolving features of the JVM tomorrow.

Demo Time!!!

https://bit.ly/supersonic-integration

Starting with Netty, which is a nice and lightweight abstraction on top of Java's NIO.2 APIs for non-blocking I/O operations.

Further improved by Eclipse Vert.x which gives us all of tools for a fully reactive Java application

Layer on Quarkus which presents a familiar and comfortable programming model based on JakartaEE Microprofile

And to go completely reactive it supports SmallRye Mutinty as a simplified experience for reactive streams.

All of this, and the underlying runtime is already providing support for the forthcoming Project Loom Virtual Threads.

This means that you are running on an extremely fast and efficient runtime today and ready to automatically take advantage of evolving features of the JVM tomorrow.

Thank You

And be sure to check out the other amazing Apache Camel content later this week!

Tuesday @ 12:10 CDT - Low-code Visual Integration Design with Camel Karavan

Wednesday @ 11:20 CDT - Saving Lives with Apache Camel K


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat