How Java Microservices Work

Microservices are a modern approach to software architecture. With microservices, application code is delivered in small, manageable pieces that are independent of others. In general, a microservices architecture represents a pattern of design in which each microservice is a small piece of the pie – in this case, the pie is the overall system. All microservices have their unique function, which is essential to the overall result. The idea behind a successful microservice is empowering the system to identify and recognize a unique subtask, or specific domain. Since each microservice will need to communicate with the other microservices often in a service mesh, the architecture requires a lightweight messaging system for such data transfer. There are a series of Java-based frameworks used to construct Java microservices. A few examples are:

Spring Boot:

Spring Boot is a well-known framework that helps build Java applications like microservices. It is effective as it makes the setup easy– everything is auto configured; no manual configurations are needed.

Quarkus:

Quarkus is a Kubernetes-native Java framework tailored for GraalVM and HotSpot, crafted from best-of-breed Java libraries and standards.

Micronaut:

Micronaut is a software framework for the Java virtual machine platform. It is designed to avoid reflection, thus reducing memory consumption and improving start times.

Akka:

Akka is an actor-based toolkit for building highly concurrent, distributed, and resilient message-driven applications for Java and Scala.

Jhipster:

JHipster is a development platform that utilizes Sprint Boot to quickly generate, develop, and deploy modern web applications and microservice architectures.

Play Framework:

Play is based on a lightweight, stateless, web-friendly architecture. Built on Akka, Play provides predictable and minimal resource consumption for highly-scalable applications.

MicroProfile:

An open forum to optimize Enterprise Java for a microservices architecture by innovating across multiple implementations and collaborating on common areas of interest with a goal of standardization.

Eclipse Vert.x:

Vert.x is a tool-kit for building Reactive applications on the JVM. Reactive applications are both scalable as workloads grow, and resilient when failures arise.

Eclipse Jersey:

This unique Java framework helps simplify the formation of REST web services. With this, communication between various microservice layers will be effective.

Swagger:

Swagger helps build REST APIs. It is a Java framework that facilitates interaction between various microservices.

vFunction

Request a Demo ▸

About vFunction

vFunction is the first and only AI-driven platform for architects and developers that intelligently and automatically transforms complex monolithic Java applications into microservices, restoring engineering velocity and optimizing the benefits of the cloud. Designed to eliminate the time, risk and cost constraints of manually modernizing business applications, vFunction delivers a scalable, repeatable factory model purpose-built for cloud native modernization. With vFunction, leading companies around the world are accelerating the journey to cloud-native architecture and gaining a competitive edge. vFunction is headquartered in Palo Alto, CA, with offices in Israel. To learn more, visit vFunction.com.