

Microservices with Quarkus

Day 1: Quarkus Fundamentals & RESTful Microservices

Course Introduction

- **Objective:** Master high-performance, reactive, cloud-native microservices with Quarkus.
- Covers the entire lifecycle: design, security, resilience, communication, deployment, and monitoring.
- **Duration:** 4 days (24 hours).
- **Structure:** Combination of theory and hands-on practice (minimum 60% practical work).
- **Prerequisites:** Strong Java background, familiarity with Maven, and basic knowledge of web services.
- **Resources:** All materials will be available in the course repository.

whoami ↩ Scott MESSNER

- **Software Career:** ☕ Java Developer, 🎓 Software Trainer, 🔧 Software Craftsman
 - ✉ Specialized in Messaging Systems (ActiveMQ, Kafka)
 - ☁ Affinity for distributed cloud systems (AWS, Azure, GCP)
- **Hobbies:** 🏃 Running, 🎧 Music, ₿ Bitcoin

Morning Session 1 (09:00 - 10:30)

- **Topic:** Getting Started with Quarkus
- **Activity:** Hands-on Lab 1
- **Goal:** Set up a complete development environment and build your first Quarkus application.

Lab 1 Objectives

- Create a new Quarkus REST application.
- Implement a custom REST endpoint for the train line's status.
- Experience key developer joy features like live reload and continuous testing.
- Test the custom REST endpoint.
- Manage the project's source code with Git.

Morning Session 2 (10:45 - 12:15)

- **Topic:** Debriefing Lab 1 & Core Concepts
- **Focus:** Reviewing the lab, understanding REST principles in Quarkus, and exploring the testing framework.

Debrief: Train Line Service REST

- **REST Endpoint:** `StatusResource` using Jakarta REST (JAX-RS) annotations (`@Path`, `@GET`).
 - [RESTEasy Reactive Guide](#)
- **Project Structure:** A closer look at `pom.xml` and the role of Quarkus extensions.
 - [Maven Tooling](#)
- **Testing:** `StatusResourceTest` validates the endpoint using RestAssured.
 - [Testing with RestAssured](#)

Debrief: Development & Testing Tools

- **Live Reload:** Quarkus automatically recompiles changes in dev mode.
- **Continuous Testing:** The benefits of instant feedback on code changes.
 - [Continuous Testing](#)
- **Dev Services:** Quarkus can automatically provision services like databases for development and testing (to be explored in later labs).
 - [Dev Services Overview](#)
 - [Azure Service Bus Extension](#)

Afternoon Session 1 (13:15 - 15:00)

- **Topic:** Introduction to Quarkus and Microservices
- **Goal:** Understand the philosophy behind Quarkus and its role in modern microservices architecture.

What is Quarkus?

- **Philosophy:** "Supersonic Subatomic Java" - A Kubernetes-native Java stack tailored for GraalVM & HotSpot, crafted from the best of breed Java libraries and standards.
- **Cloud-Native Approach:** Optimized for fast startup times and low memory usage, making it ideal for containers and serverless.
- **Standards-Based:** Built on standards like Jakarta EE and MicroProfile, with a rich ecosystem of extensions.

Quarkus in the Ecosystem

- **Comparison:**
 - **vs. Spring Boot:** Quarkus offers faster boot times and lower memory footprint due to its build-time optimizations.
 - **vs. Jakarta EE:** Provides a more modern, streamlined, and developer-friendly experience for cloud-native development.
- **Why Quarkus for Cloud?:** Excels in environments like Kubernetes and Serverless due to its resource efficiency and rapid scaling.
 - [Quarkus on Kubernetes](#)
 - [Quarkus on AWS Lambda](#)
 - [Quarkus on Azure?](#)

What are Microservices?

- **Architectural Principles:** A style of architecture that structures an application as a collection of loosely coupled, independently deployable services.
 - Key principles include Single Responsibility, Bounded Contexts, and Decentralized Governance.
- **Decoupling & Deployment:** Services can be developed, deployed, and scaled independently, enabling faster release cycles and greater resilience.

Introduction to MicroProfile

- **Purpose:** An open-source specification for building enterprise Java microservices, standardizing features across different vendor implementations.
 - [MicroProfile in Quarkus](#)
- **Quarkus & MicroProfile:** Quarkus provides a highly efficient and optimized implementation of the MicroProfile specifications.
- **Implementation:** Many of the MicroProfile APIs in Quarkus are provided by the [SmallRye](#) project.

Key MicroProfile & Quarkus APIs

- **MicroProfile Specs:**
 - **Config**: Externalize configuration.
 - **Health**: Expose health checks.
 - **Metrics**: Provide monitoring data.
 - **REST Client**: Type-safe REST invocation.
 - **Fault Tolerance**: Build resilient services.
- **Core Quarkus Extensions:**
 - **RESTEasy Reactive**: For building reactive REST APIs.
 - **Hibernate ORM with Panache**: Simplifies data access.
 - **OpenAPI & Swagger UI**: Generate API documentation.

Afternoon Session 2 (15:00 - 16:00)

- **Topic:** Deploying a Microservice
- **Activity:** Hands-on Lab 2
- **Goal:** Build and configure a simple REST microservice from scratch.

Lab 2 Objectives

- Add logging to a REST endpoint and debug a running application.
- Externalize and dynamically configure microservice properties.
- Override configuration for different environments.
- Build a native container image for the application.

End of Day 1

- Recap & Q&A