**Micro Services on QUARKUS**

**Building Cloud Native Apps**

**Duration 5 days**

**Prerequisite:**

* A minimum of 1 Year Experience in Programming
* Must have worked in Java 8 onwards
* Containers Knowledge - Docker

**Day 01**

**Micro Services Patterns Overview**

* Application architecture patterns
  + Monolithic architecture
  + Microservice architecture
* What is Micro Services
* Why Microservices
* Microservices patterns
  + Decomposition
  + Data Management
  + Deployment patterns
  + Communication Style Patterns
  + Service Discovery
  + Reliability
  + Observability
  + Security
* Testing Micro Services

**Micro Services Implementations on JVM**

* Microservice Implementation on JVM
  + Spring Cloud on Spring boot
  + Micronaut
  + Quarkus
  + Eclipse Vert.x
  + Eclipse MicroProfile

**Eclipse Micro Profile**

* + What is MicroProfile
  + MicroProfile Working Group && its Members
  + MicroProfile Implementations
    - QUARKUS
    - Open Liberty
    - Apache TomEE
    - Payara
    - Thorntail
    - WildFly
    - Lanucher
    - KumuluzEE
  + MicroProfile Microservices Spec features
    - MicroProfile Config
    - MicroProfile Health Check
    - MicroProfile Metrics
    - MicroProfile Fault Tolerance
    - MicroProfile Rest Client
    - MicroProfile Open API
    - MicroProfile JWT Authentication
    - MicroProfile Open Tracing

**QUARKUS**

* What is QUARKUS
* Why QUARKUS
* Problems in the Java Apps in Cloud
  + Not Cloud Native ready
  + Initial load Performance issue
  + Huge Bundle size
  + Not Containers ready
* Cloud Native Java Apps
  + Improve Cloud Native Ready
* Runtimes
  + Container with JVM
  + Container Without JVM – Native Image
* GraalVM
  + Increase application throughput and reduce latency
  + Compile applications into small self-contained native binaries
  + Seamlessly use multiple languages and libraries
* QUARKS Apps
  + Designed for GraalVM Ready
  + Based on JEE Spec Standards
  + Based on MicroProfile Spec for Microservices
  + Container first
  + Build time metadata processing
  + Reduction in Reflection use
  + Native Image pre boot

QUARKS Tech Stack

* + Quarks is collection of Community Projects
    - Eclipse Vert.x
    - SmallRye
    - Hibernate
    - Netty
    - RESTEasy
    - Apache Camel
    - Eclipse MicroProfile

**Day 02**

QUARKS Modules

* + Core
  + Web
  + Data
  + Messaging
  + Security
  + Business Automation
  + Integration
  + Command Line Applications
  + Cloud
  + Observability
  + Serialization
  + Frameworks and Languages
* Project Setup
  + Create maven project
  + Create Project via Quarks initializer
  + Bootstrapping an application
  + Creating a JAX-RS endpoint
  + Injecting beans
  + Functional tests
  + Packaging of the application
* Quarks Core
  + Core
  + Introduction core modules
  + CDI
  + Configuration
  + Loggers
  + Context Propagation
  + Initialization and termination callbacks
* Quarks Web
  + Building Restfull End Point
  + JAX-RS Spec and annotations
  + Spring Boot Integration and its annotations
  + REST JSON Services

**Day 03**

* Service Communication
  + [Setting up the model](https://quarkus.io/guides/rest-client-multipart#setting-up-the-model)
  + [Create the interface](https://quarkus.io/guides/rest-client-multipart#create-the-interface)
  + [Create the configuration](https://quarkus.io/guides/rest-client-multipart#create-the-configuration)
  + [Update the JAX-RS resource](https://quarkus.io/guides/rest-client-multipart#update-the-jax-rs-resource)
  + [Creating the server](https://quarkus.io/guides/rest-client-multipart#creating-the-server)
  + Service to Service Communication using Rest Client
* Data
  + Data source
  + Jdbc data source
  + Using Hibernate and JPA ORM
  + Hibernate ORM With Panche
  + Redis client
  + Caching
  + MongodB client

Reactive Programming ,Async, and Non blocking apis on Quarkus

* + What is Reactive Programming
  + Reactive programming Patterns
  + Imperative vs Declarative apis
  + Push vs Pull Programming
  + Reactive Streams
  + Reactive Manifesto
  + Blocking vs Nonblocking
  + Async
  + Callbacks
  + Vert.x Engine
* Vert.x Engine
  + What is Vertx
  + Architecture of Vert.x
  + Threading models
  + Worker threads and io Threads
* Reactive Implementation on Quarks
  + Using Quarkus reactive APIs
  + Using Vert.x clients
  + Using RxJava or Reactor APIs
  + Using Completion Stages or Publisher API
* WRITING REST SERVICES WITH RESTEASY REACTIVE
  + Declaring endpoints: URI mapping
  + Declaring endpoints: HTTP methods
  + Declaring endpoints: representation / content types
  + Accessing request parameters
  + Declaring URI parameters
  + Accessing the request body
  + Handling Multipart Form data
  + Returning a response body
  + Setting other response properties
  + Async/reactive support
  + Streaming support
  + Server-Sent Event (SSE) support
  + Accessing context objects
  + JSON serialization
* Micro Service Communication
  + Vert.x Web Client for Async Reactive http based communication

**Day 04 && Day 05**

**Micro Services Pattern Implementations**

* **Micro Service Implementation Projects on Quarkus**
  + **SmallRye Project**
* **SmallRye** 
  + MicroProfile based Project
  + Used by Quarks
* **QUARKUS -**Distributed Configuration for MicroProfile :SmallRye Configuration
* **Eclipse MicroProfile Fault Tolerance :SmallRye fault Tolerance**
  + Timeout: Define a maximum duration for execution
  + Retry: Attempt execution again if it fails
  + Bulkhead: Limit concurrent execution so that failures in that area can’t overload the whole system
  + CircuitBreaker: Automatically fail fast when execution repeatedly fails
  + Fallback: Provide an alternative solution when execution fails
* **QUARKUS -SMALLRYE HEALTH**
  + Running the health check
  + Creating your first health check
  + Adding a readiness health check procedure
  + Negative health check procedures
  + Adding user-specific data to the health check response
  + Extension health checks
  + Health UI
* **QUARKUS - USING OPENTRACING**
  + Creating the Maven project
  + Examine the JAX-RS resource
  + Create the configuration
  + Run the application
  + Tracing additional methods
  + Additional instrumentation
  + JDBC
  + Kafka
  + MongoDB client
  + Jaeger Configuration Reference
* **QUARKUS - MICROMETER METRICS**
  + Gauges
  + Counters
  + Summaries and Timers
  + Review automatically generated metrics
  + Using MeterFilter to configure metrics
  + Does Micrometer support annotations?
  + Using other Registry implementations
  + Creating a customized MeterRegistry
  + Support for the MicroProfile Metrics API
* **CENTRALIZED LOG MANAGEMENT (GRAYLOG, LOGSTASH, FLUENTD)**
  + Send logs to Graylog
  + Send logs to Logstash / the Elastic Stack (ELK)
  + Send logs to Fluentd (EFK)
  + GELF alternative: use Syslog
* **Deployment : Cloud**
  + Build Container Images
  + Deploying applications Quarkus on Kubernetes
  + Deploying applications Quarkus on Openshift
  + Serverless - Amazon lambda
* **Securing Micro Services**
  + Security overview
  + Security with .properties files
  + Security with jpa,jdbc,ldap
  + JWT Token based authentication