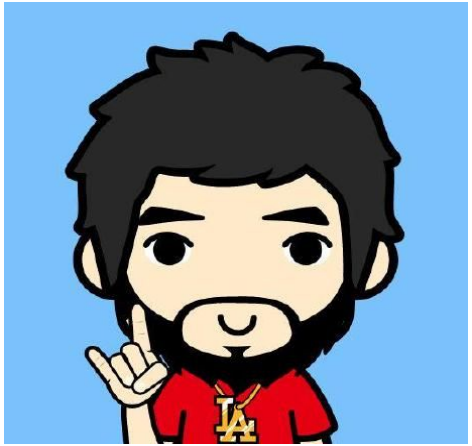
Microserviços Supersônicos e Subatômicos com Quarkus

Pedro Hos and William Siqueira

Pedro Hos @ github.com/pedro-hos

- Banco de Dados @ FATEC São José dos Campos 2012
- I’ve worked as Java Developer in some companies
- Since 2017 I am Software Maintenance Engineer @ Red Hat
- JUG Leader at JUG Vale jugvale.com
- Contributor at SJCDigital github.com/sjcdigital
- Blogger at pedrohosilva.wordpress.com
- Opensource and some source code samples github.com/pedro-hos

William Siqueira @ github.com/jesuino



- Banco de Dados @ FATEC São José dos Campos 2010
- Software Engineer @ Red Hat
- Colaborador do JUG Vale jugvale.com
- Colaborador do SJCDigital github.com/sjcdigital
- Escreve em alguns blogs
- Palestrante JavaOne, The Developers Conference, FISL e outros
- Opensource github.com/jesuino

Schedule

- Microservices overview
- Java and microservices
- Eclipse Microprofile
- Quarkus - numbers
  - First demo
    - Add hello endpoint - play with configuration - add json - add health check
    - add swagger - fault tolerance
- Quarkus features
  - Extensions - talk about all extensions
  - Second Demo - Database - security
  - Kogito Demo
- Quarkus Native - talk about how quarkus can be compiled to native
- Quarkus on Openshift/Kubernetes

What are microservices

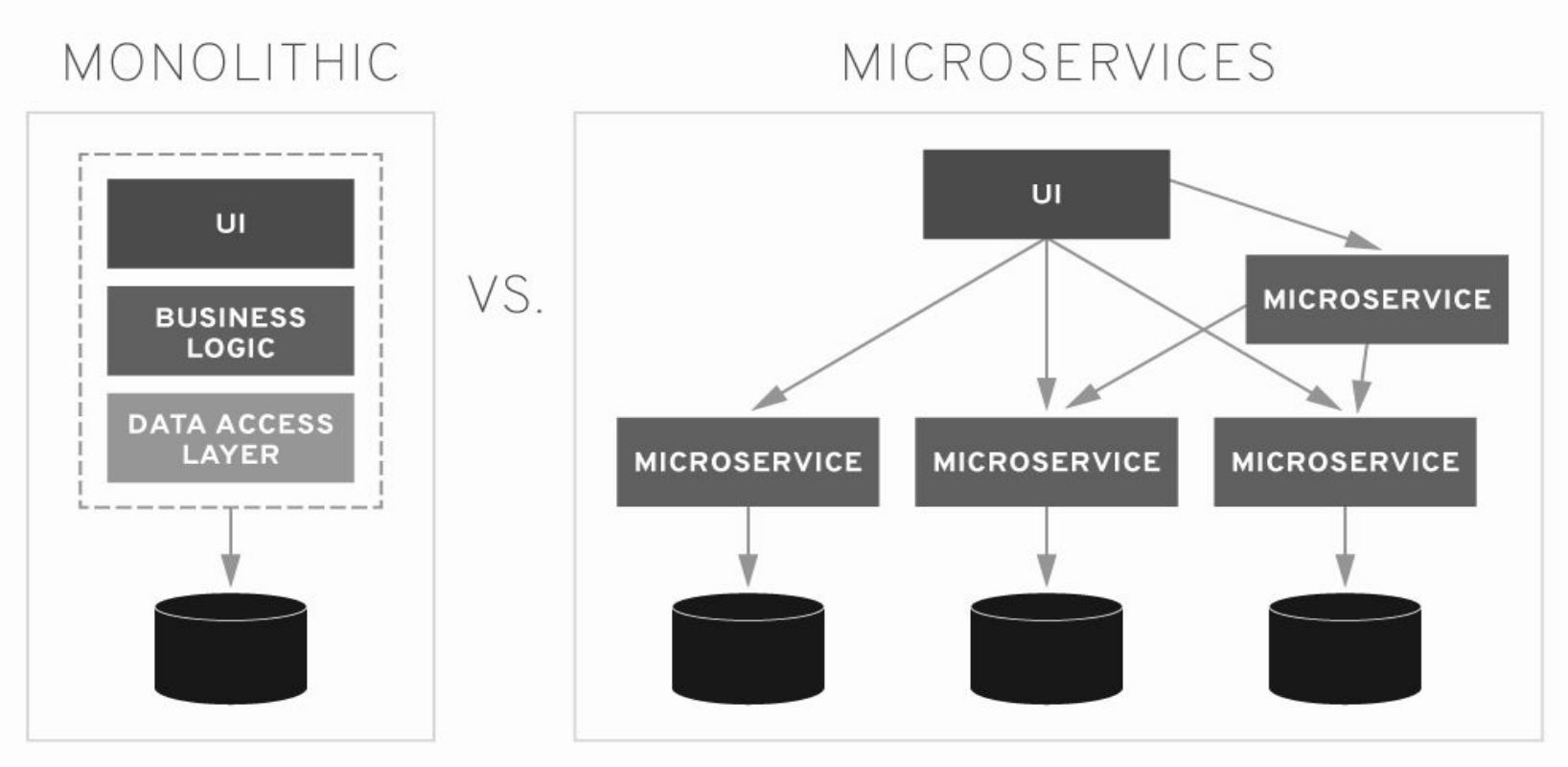
Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

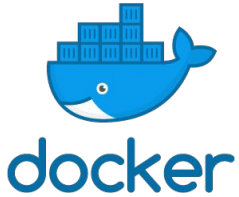
Reference: microservices.io

What are microservices



Microservices ecosystem

- Microservices are usually packaged in containers
  - Docker is a famous container solution



- Containers are used in production with a container orchestrator
  - Kubernetes is the most used container orchestrator solution

- Other tools are built for containers security, testing, monitoring and more
  - Around the microservices containers and infrastructure new tools can be used such as Istio, Prometheus...

Creating microservices with Java

- JakartaEE
  - Was known as Java EE before
  - Java Specification for Enterprise applications
  - Usually requires application server
  - Open specification focused in backwards compatibility (slow evolution)
- Spring boot
  - Not a specification
  - Unique vendor
  - FAT Jars approach
- Other libraries and frameworks
  - Play, Micronaut, Spark...
  - The list goes on… We need a specification for Java microservices

JAKARTA EE

Are all these libraries ready for use with microservices ecosystem?

Eclipse Microprofile

- Microservices Specification
  - Healthcheck
  - Circuit breaker
  - Fault Tolerance
  - API Documentation
- Focus on fast releases and innovation -
- May not have backwards compatibility
- Between Microservices implementations we have Quarkus

microprofile.io

Quarkus.io


QUARKUS
Supersonic Subatomic Java
A Kubernetes Native Java stack tailored for GraalVM & OpenJDK HotSpot,
crafted from the best of breed Java libraries and standards

Quarkus #Facts

- Container First
  - Ready for Docker and Kubernetes
- 10x lighter
  - The final JAR is optimized during and is faster than usual applications
- 100x faster
  - Can be compiled to a native package, resulting in high performance applications
- Low memory usage and fast Startup
- Live Reload
  - Run Quarkus in development and your changes are automatically reloaded, no need to server stop/start
- Microservices
  - Implements and extends Microprofile with Quarkus Extensions, built on top of mature libraries
- Developer Joy

How fast is Quarkus

it is supersonic, subatomic Java

Let’s code...

All source code are on github:
https://github.com/pedro-hos/javaee-pocs/tree/master/microservices/quarkus/tdc-recife-2019/quarkus-tdc-recife

Start Coding

- https://code.quarkus.io/
- Fill the Group and Artifact
- Choice the Extensions
- Download and run the application:
  - ./mvnw compile quarkus:dev
- Play with Configurations @ConfigProperty(...)
- List extensions and add JSON extension
- Add health check extension and hit the /health
- Create a new HealthCheck with @Liveness
- Add openAPI extension and add the property:
  - quarkus.swagger-ui.always-include=true

Start Coding

- Add fault tolerance extension
- Create a new resource to test fault tolerance
  - Test @Retry
  - Test @Timeout
  - Test @Fallback
- Configure MySQL Database
- Configure Keycloak security

And much more

- Reactive APIs
  - Vert.x, REST, Messaging and database drivers (Postgres and MySQL)
- Cloud
  - Serverless with AWS Lambda, Kubernetes
- Other Languages
  - You can develop Quarkus applications using Scala and Kotlin
- Messaging
  - Kafka, AMQ, MQTT, Artemis and Microprofile Messaging
- Business Automation
  - Kogito brings Drools and jBPM to Quarkus
- Misc
  - Email, Apache Tika, JGit, tasks scheduling and more to come
- Spring (!!!)

Quarkus Cheat Sheet -> https://lordofthejars.github.io/quarkus-cheat-sheet

Questions?