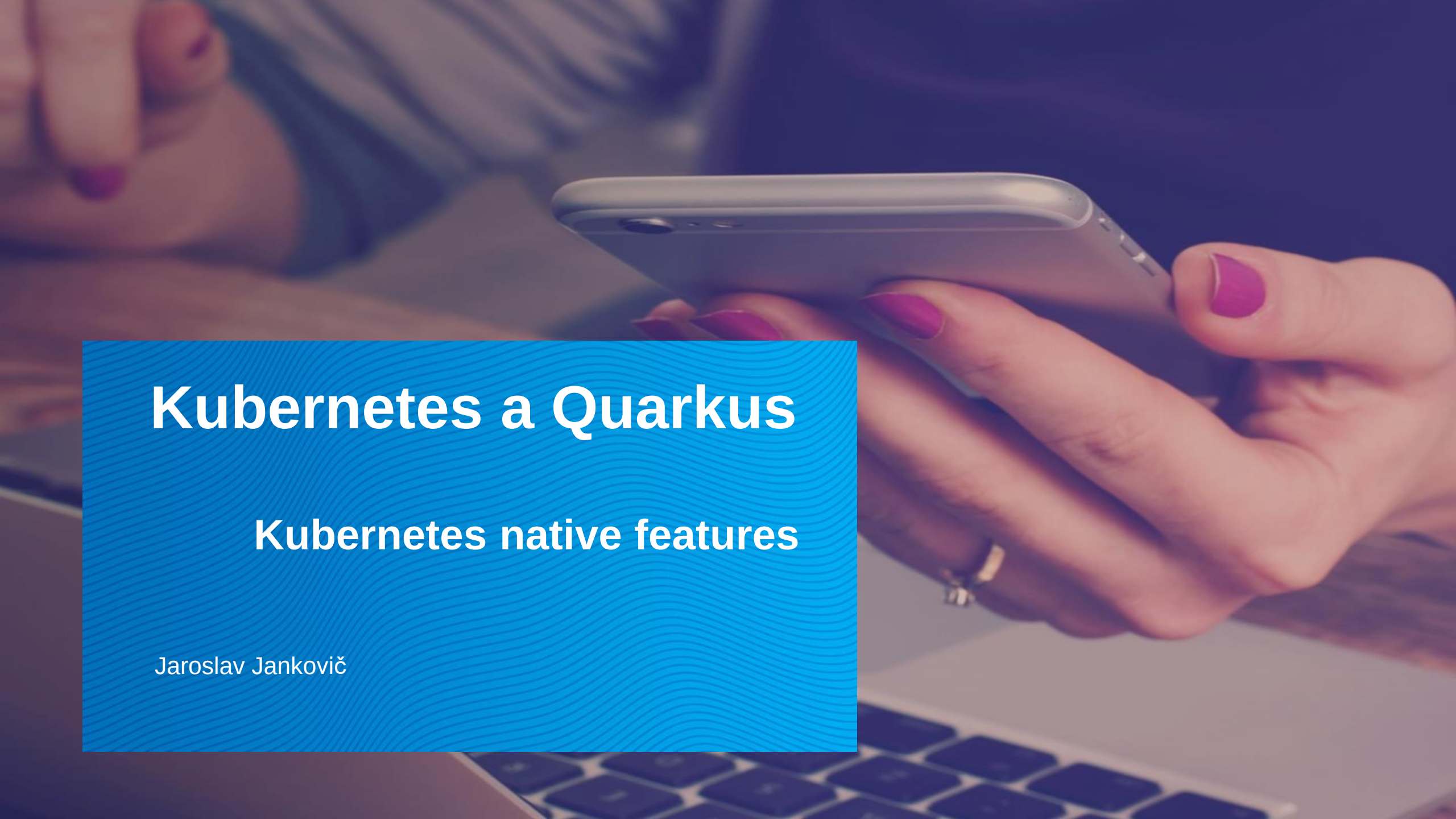

Kubernetes a Quarkus

Kubernetes native features

Jaroslav Jankovič

Zadanie

- Vyskúšanie pokročilých nástrojov v Quarkus
  - Builtin podpora Docker platformy
  - Nasadenie Quarkus na Kubernetes
  - Natívna kompilácia pomocou GraalVM

Hodnotenie Quarkus cvičení

+ 5 bodov

- dokončená **fee-service** služba (model, repository, backend interface, backend bean, REST, Config, Security)
- tzn. **nie je potrebné** dokončit: JUnit test, integračný test, OpenAPI, Health, Metrics

+ 5 bodov

- dokončené **nasadenie JVM** verzie **fee-service** na lokálny **Kubernetes** (Docker support, Kubernetes support, ConfigMap, Secrets)

+ 5 bonusových bodov:

- Nasadenie GraalVM skompilovanej verzie na Kubernetes
- Použitie pokročilých Kubernetes nastavení (replicas, liveness probe, readiness probe)

Termín odovzdania zadania je do **7.4.2023**

Pre získanie základných **10 bodov** je potrebné do školského Teams nahrať celý zdrojový priečinok projektu.

Pre získanie bonusových **5 bodov** je navyše potrebné:

- Nahrať súbor **kubectl_get_pods.txt**
  - súbor bude obsahovať výstup z príkazu: ***kubectl get pods***
- Nahrať súbor **kubectl_log.txt**
  - súbor bude obsahovať štartovacie logy z jedného podu (posledný riadok z logu bude obsahovať ***Installed features***)
  - logy získate napr pomocou príkazu: ***kubectl logs -f <pod_name>***

Príprava prostredia

- Príkazy môžete nájsť na https://github.com/jjankovi/fmfi/blob/main/2.cvicenie.md

- Prerekvizity z minulého Quarkus cvičenia
  - JDK 11+
  - Maven 3.8.6 (optional)
  - VS Code
  - Dokončená služba *fee-service*
- Prerekvizity z minulých cvičení:
  - Docker Desktop so spusteným Kubernetes

Build Quarkus

- Vytvorenie produkčného spustiteľného Quarkus artefaktu
  - *mvn clean package -DskipTests*
- Spustenie aplikácie
  - *java -jar target/quarkus-app/quarkus-run.jar* (ak v PATH mate JRE)
  - *%JAVA_HOME%/bin/java -jar target/quarkus-app/quarkus-run.jar*
- Nastavte aplikačnú premennú cez env variable daného OS a zopakujte
  - *WINDOWS: set minimal_fee_limit=10000*
  - *UNIX LIKE: export minimal_fee_limit=10000*

Docker

- Dockerfile
  - predpripravené Dockerfile (nonlegacy / legacy / native)
- Pridanie Docker podpory do Quarkus:
  - *mvn quarkus:add-extension -Dextensions="container-image-docker"*
  - Do aplikačnej konfigurácie pridajte:
    - *quarkus.container-image.group=fmfi*
    - *quarkus.container-image.name=fee-service*
    - *quarkus.container-image.tag=latest*
- Docker build:
  - použite *mvn clean package -DskipTests -Dquarkus.container-image.build=true*
  - Výsledkom je vytvorenie Docker image s označením ***fmfi/fee-service:latest***
- Spustite Docker kontajner
  - nezabudnite nastaviť potrebnú environment premennú a vystaviť port 8080
  - *docker run -p 8080:8080 -e "minimal_fee_limit=10000" fmfi/fee-service*

H2 – externá DB (cez Docker)

- Spustite standalone H2 DB (pomocou Docker)
  - *docker run -d -e H2_OPTIONS=-ifNotExists -p 1521:1521 -p 81:81 -v %cd%\h2_data:/opt/h2-data --name=H2Instance oscarfonts/h2*
  - DB dáta budú uložené aj po reštarte Docker kontajnera
  - otestujte DB konekciu http://localhost:81 (tabuľka ***fee*** ešte nebude existovať)
- Vytvorte schému
  - *create sequence hibernate_sequence start with 1 increment by 1;*
  - *create table Fee (*

    id bigint not null,

    acno varchar(255),

    amount numeric(19,2),

    creationDate timestamp,

    transactionId varchar(255),

    primary key (id)

  );

H2 – naplnenie databázy

- Spustite insert sql commands:
  - insert into fee values (1, '1987426353', '2', '2023-01-01', '83f3d5b6-7ca2-4f1d-838c-4d1cbfb1e1d8');
  - insert into fee values (2, '1987426353', '0.01', '2023-02-01', 'ceab21f0-fc20-4ed7-a6e8-6ac26ead5f39');
  - insert into fee values (3, '4444441111', '0.01', '2023-03-01', '791ba284-1491-40cd-a0f0-30589644ac2b');
  - insert into fee values (4, '4444441111', '0.01', '2023-03-02', '84ae8704-a0f0-4e0d-a1c7-919ba3a3ebdf');
  - insert into fee values (5, '4444441111', '2', '2023-03-03', '5cb19bcc-3212-4ba1-b2fb-3df59fb23379');
  - insert into fee values (6, '4200012345', '2', '2023-03-20', 'd6b6a7c8-c4cb-4f00-be22-46a688ddb01e');
- Overte naplnenie DB
  - select * from fee;

Docker – konekcia na externú databázu

- Upravte aplikačnú konfiguráciu
  - Hodnota premennej *quarkus.datasource.jdbc.url* bude **${db_url}**
  - Nová premenná *quarkus.datasource.username* s hodnotou **${db_username}**
  - Odstráňte premennú *quarkus.hibernate-orm.database.generation*
- Skompilujte, zdockerizujte a spustite aplikáciu v produkčnom móde (Docker kontajner) s potrebnými premennými
  - *docker run -p 8080:8080 -e "minimal_fee_limit=10000" -e "db_url=jdbc:h2:tcp://host.docker.internal:1521/test" -e "db_username=sa" fmfi/fee-service*

Kubernetes – príprava (1)

- Príprava aplikácie
  - Pridanie extensions:
    - *mvn quarkus:add-extension -Dextensions="quarkus-kubernetes"*
  - Úprava aplikačnej konfigurácie
    - *quarkus.kubernetes.image-pull-policy=never*
    - *quarkus.kubernetes.service-type=load-balancer*
    - *quarkus.kubernetes.ports."http".host-port=8080*

Kubernetes – príprava (2)

- Otvorte súbor *~/.kube/config*
  - Skopírujte hodnotu premennej ***client-certificate-data*** do env variable ***CLIENT_CERT_DATA***
  - Skopírujte hodnotu premennej ***client-key-data*** do env variable ***CLIENT_CERT_KEY***
- Konfigurácia Quarkus pripojenia na Kubernetes API
  - Úprava aplikačnej konfigurácie
    - *quarkus.kubernetes.namespace=default*
    - *quarkus.kubernetes-client.trust-certs=true*
    - *quarkus.kubernetes-client.master-url=https://kubernetes.docker.internal:6443*
    - *quarkus.kubernetes-client.client-cert-data=${CLIENT_CERT_DATA}*
    - *quarkus.kubernetes-client.client-key-data=${CLIENT_CERT_KEY}*

Kubernetes – nasadenie

- Zbuildujte a zdockerizujte projekt pomocou Maven
  - aký je rozdiel v target priečinku?
- Nasaďte Docker image na Kubernetes (2 spôsoby)
  1. *kubectl apply -f target/kubernetes/kubernetes.yaml*
  2. *mvn package -DskipTests -Dquarkus.kubernetes.deploy=true*
- Overte stav servera:
  - *kubectl get pods* – aký je stav pod-u?
- Zistite dôvod chyby
  - *kubectl logs -f <pod_id>*

Kubernetes – ConfigMap

- Vytvorte Kubernetes ConfigMap súbor *app-configmap.yml*
  - apiVersion: v1
  - kind: ConfigMap
  - metadata/name: app-configmap
  - do data/ pridajte všetky premenné aj s hodnotami
- Vytvorte v Kubernetes ConfigMap objekt pomocou *kubectl apply -f*
- Overte dostupnosť ConfigMap objektu
  - *kubectl describe configmaps app-configmap*
- Upravte aplikačnú konfiguráciu
  - ak sa všetky aplikačné premenné zhoduju s ConfigMap potom stačí:
    - *quarkus.kubernetes.env.configmaps=app-configmap*
  - pre každý aplikačný parameter, ktorý sa nezhoduje s ConfigMap pridajte (pozor na názov)
    - *quarkus.kubernetes.env.mapping.<param>.from-configmap=app-configmap*
    - *quarkus.kubernetes.env.mapping.<param>.with-key=<keyName>*
- Nasaďte aplikáciu na Kubernetes, overte stav servera

Kubernetes – Secrets

- Upravte aplikačnú konfiguráciu:
  - Hodnota hesla basic autentifikácie bude ***${user_password}***
- Vytvorte Kubernetes Secrets súbor *app-secret.yml*
  - apiVersion: v1
  - kind: Secret
  - metadata/name: app-secret
  - do data pridajte ***user_password.*** Hodnota bude enkódovaná pomocou base64 (použite inú hodnotu hesla ako ste mali v aplikačnom súbore)
    - Napr *https://www.base64encode.org/*
- Vytvorte v Kubernetes Secret objekt pomocou *kubectl apply -f*
- Upravte aplikačnú konfiguráciu tak, aby aplikácia konzumovala Kubernetes Secret
  - Totožný spôsob konfigurácie ako ConfigMap – len namiesto ***configmaps*** použijete ***secrets***
- Nasaďte aplikáciu na Kubernetes, overte stav služby (*kubectl get pods*)

Bonus – GraalVM Linux executable

- Dockerfile pre GraalVM
  - Predpripravené Dockerfile.native
- Vytvorenie Linux executable bez potreby nainštalovaného GraalVM (pomocou Docker image Mandrel) + vytvorenie Docker image pre Linux executable + nasadenie na Kubernetes
  - *mvn clean package -Pnative -DskipTests -Dquarkus.native.container-build=true -Dquarkus.container-image.build=true -Dquarkus.kubernetes.deploy=true*
  - V prípade ak Docker začne sťahovať iný image ako **ubi-quarkus-mandrel-builder-image:22.3-java17**
    - treba použiť premennú **quarkus.native.builder-image=quay.io/quarkus/ubi-quarkus-mandrel-builder-image:22.3-java17**
- Overte stav služby (*kubectl get pods*)

Bonus – Kubernetes pokročilé

- Zvýšte počet inštancií služby (Kubernetes service replicas) na **2** pomocou aplikačnej konfigurácie:
  - Použite premennú *quarkus.kubernetes.replicas*
- Pridanie podpory pre liveness/readiness probes:
  - *mvn quarkus:add-extension -Dextensions="smallrye-health“*
  - Upravte Kubernetes liveness/readiness probes pomocou aplikačnej konfigurácie:
    - *quarkus.kubernetes.[liveness/readiness]-initial-delay*, hodnota premennej = 0S
    - *quarkus.kubernetes.[liveness/readiness]-probe.period*, hodnota premennej = 10S
    - *quarkus.kubernetes.[liveness/readiness]-probe.timeout*, hodnota premennej = 10S
  - *mvn package -DskipTests*
    - Overte si nastavenie liveness/readiness probes v *target/kubernetes/kubernetes.yml*