Quarkus, développer ses microservices en Java dans le Cloud

Cours Pratique de 3 jours - 21h
Réf : QRK - Prix 2024 : 2 070CHF HT

Développé par Red Hat, Quarkus est un framework Java conçu pour les machines virtuelles Java (JVM) et la compilation native. Vous apprendrez à développer des applications Java découpées en microservices et en créerez des images pour les conteneurs que vous déploierez via un orchestrateur cloud comme Kubernetes.

OBJECTIFS PÉDAGOGIQUES

À l'issue de la formation l'apprenant sera en mesure de :

Concevoir une architecture fondée sur des microservices

Mise en place d'une communication orientée message entre deux microservices

Mettre en œuvre des systèmes de tolérance aux pannes

Sécuriser des microservices pour empêcher tout accès non autorisé

Tester et déployer une application microservices

Superviser une application en production

LE PROGRAMME

dernière mise à jour : 11/2023

1) Présentation

- Un nouveau framework Java ?
- La comparaison avec Spring Boot.
- Quid de Java EE et Jakarta EE ?
- Les architectures microservices.
- Le standard MicroProfile

2) Premiers pas avec Quarkus

- Création d'un projet "Hello world".
- L'outillage pour le développeur (Dev Services).
- Les processus de développement, débogage et build.
- Le framework de test Quarkus.
- L'environnement Docker.
- Tour d'horizon des extensions Quarkus.

Travaux pratiques : Prise en main de l'environnement et développement d'un premier microservice, expérimentation du live coding, du débogage, du test continu.

3) Communication HTTP/RESTful et GraphQL

- Rappels sur les principes de REST.
- Utilisation de Jakarta RESTful.
- Apports de GraphQL.
- Documentation des endpoints (Open API).
- Écriture d'un client HTTP.
- Sécurité et authentification.

PARTICIPANTS

Développeurs Java, chefs de projet Java/Java EE.

PRÉREQUIS

Bonnes connaissances de Java/Java EE.

COMPÉTENCES DU FORMATEUR

Les experts qui animent la formation sont des spécialistes des matières abordées. Ils ont été validés par nos équipes pédagogiques tant sur le plan des connaissances métiers que sur celui de la pédagogie, et ce pour chaque cours qu'ils enseignent. Ils ont au minimum cinq à dix années d'expérience dans leur domaine et occupent ou ont occupé des postes à responsabilité en entreprise.

MODALITÉS D'ÉVALUATION

Le formateur évalue la progression pédagogique du participant tout au long de la formation au moyen de QCM, mises en situation, travaux pratiques…
Le participant complète également un test de positionnement en amont et en aval pour valider les compétences acquises.

MOYENS PÉDAGOGIQUES ET TECHNIQUES

• Les moyens pédagogiques et les méthodes d'enseignement utilisés sont principalement : aides audiovisuelles, documentation et support de cours, exercices pratiques d'application et corrigés des exercices pour les stages pratiques, études de cas ou présentation de cas réels pour les séminaires de formation.
• À l'issue de chaque stage ou séminaire, ORSYS fournit aux participants un questionnaire d'évaluation du cours qui est ensuite analysé par nos équipes pédagogiques.
• Une feuille d'émargement par demi-journée de présence est fournie en fin de formation ainsi qu'une attestation de fin de formation si le stagiaire a bien assisté à la totalité de la session.

MODALITÉS ET DÉLAIS D'ACCÈS

L'inscription doit être finalisée 24 heures avant le début de la formation.

ACCESSIBILITÉ AUX PERSONNES HANDICAPÉES

Vous avez un besoin spécifique d'accessibilité ? Contactez Mme FOSSE, référente handicap, à l'adresse suivante psh-accueil@orsys.fr pour étudier au mieux votre demande et sa faisabilité.

- Implémentation des "Long Running Actions".
Travaux pratiques : Création de services RESTful et GraphQL, interrogation depuis un deuxième microservice Quarkus.

4) La tolérance à la panne
- Le principe de résilience.
- Pourquoi est-ce important ?
- La programmation défensive.
- Circuit Breaker, Bulkhead.
- Les autres patterns.
Travaux pratiques : Mise en pratique des 6 patterns de tolérance à la panne du standard MicroProfile.

5) Communication orientée message
- La programmation réactive.
- Les bénéfices et principales difficultés.
- Comment gérer les transactions ?
- Le pattern Saga.
- La communication asynchrone (ActiveMQ, Kafka...).
Travaux pratiques : Mise en place d'une communication orientée message avec Kafka entre deux microservices.

6) Mise en production
- Mécanisme de configuration pensé pour les conteneurs.
- Stratégie de construction des images.
- HotSpot et GraalVM.
- Compilation AOT, un changement radical.
- Déploiement sur Kubernetes.
Travaux pratiques : Construction des images Docker Open Container Initiative (OCI) pour un déploiement sur un orchestrateur cloud. : classiques (OpenJDK) et Ahead Of Time (AOT, GraalVM).

7) Supervision en production
- Définition des Health Check.
- Gestion des logs éparpillés.
- OpenTelemetry le nouveau standard.
- Les métriques systèmes et personnalisées.
Travaux pratiques : Définition d'un Endpoint Health check personnalisé, recueil et affichage des données de télémétrie dans Prometheus.

LES DATES

CLASSE À DISTANCE
2024 : 03 juin, 09 sept., 25 nov.