Rapport sur l’architecture de Uber

Groupe :

Amadou DIA Maxime NGUYEN

Introduction

Uber est une plateforme fondé en 2009 aux États-Unis, Elle permet de solliciter un chauffeur en intégrant un système de paiement directement via l'application. C'est une alternative pratique (et parfois moins couteuse) aux taxis traditionnels. L'entreprise propose également des services de livraison de repas et de colis.

I/ L'architecture microservices au cœur de l'entreprise Uber

A/ Définition des microservices

Les microservices sont une architecture logicielle qui consiste à diviser une application en services indépendants les uns des autres, chacun ayant leur propre responsabilité métier et communiquant entre eux via des API.

Chaque service est conçu pour être autonome, évolutif et déployable indépendamment des autres services.

Les microservices ont émergé comme une alternative à l'architecture monolithique traditionnelle, qui est plus difficile à gérer et à faire évoluer.

B/ L'adoption des microservices par Uber

Dans le cas d'Uber, l'architecture microservices a été mise en place pour répondre aux besoins de l'entreprise en matière de scalabilité et de flexibilité. Ce type d'architecture est super pratique pour Uber, car cela leur permet de gérer un grand volume de données et de requêtes tout en maintenant un service fiable.

II/ Comprendre l'architecture microservices

A/ Explication et avantages

L'architecture microservices est une approche visant à développer des applications qui se concentre sur la construction de services indépendants et autonomes communiquant entre eux via des API.

Chaque service est conçu pour résoudre un seul et unique problème. Elle présente également l'avantage de faciliter la maintenance de l'application, car chaque service est autonome et peut être mis à jour ou remplacé sans affecter les autres services.

Les services peuvent être développés en utilisant différents langages de programmation et des technologies, ce qui permet aux développeurs de choisir les outils qui conviennent le mieux à leur service.

B/ L'architecture microservices dans les grandes entreprises

De nombreuses grandes entreprises utilisent l'architecture microservices, y compris Netflix, Amazon, eBay, PayPal, et bien sûr Uber. Ces entreprises ont adopté cette architecture pour répondre aux besoins de leur entreprise en matière de flexibilité, de scalabilité, de fiabilité et de rapidité de déploiement.

III/ Les avantages de l'architecture microservices et sa structure en couches

A/ Les avantages globaux de l'architecture microservices

L'architecture microservices offre plusieurs avantages par rapport à l'architecture monolithique traditionnelle.

B/ La structure en couches de l'architecture microservices et leur détails

1. Couche de présentation :
   La couche de présentation est responsable de l'affichage des données à l'utilisateur. Elle peut prendre différentes formes, telles qu'une interface web, une application mobile ou une interface en ligne de commande. Les micro services ne gèrent pas directement la présentation des données mais fournissent les données nécessaires qui seront présentées à l'utilisateur par cette couche.
2. Couche de service :
   La couche de service contient les micro services eux-mêmes. Chaque microservice est spécialisé dans une fonctionnalité spécifique et fournit une API pour communiquer entre eux. Cette couche sert d'intermédiaire entre la couche de présentation et la couche d'accès aux données.
3. Couche d'accès aux données :
   La couche d'accès aux données est responsable de la lecture et de l'écriture des données dans les bases de données. Les micro services communiquent avec cette couche pour accéder aux données dont ils ont besoin pour effectuer leurs opérations.
4. Couche de base de données :
   La couche de base de données est en charge de la persistance des données et de la gestion des

transactions. Cette couche est généralement composée de plusieurs serveurs de bases de données pour garantir la disponibilité des données.

Les points forts de l'architecture microservices comprennent :

- L'adaptabilité : Les microservices fonctionnent de manière autonome, permettant ainsi d'ajuster les services en fonction des exigences de l'entreprise sans trop de difficultés.
- La souplesse : Chaque microservice peut être conçu dans un langage de programmation distinct et déployé sur des infrastructures variées.
- La robustesse : Les microservices opèrent indépendamment, assurant la continuité des autres services en cas de défaillance d'un d'entre eux.
- Le déploiement en continu : Les microservices se déploient de façon autonome, facilitant les mises à jour fréquentes et les déploiements en continu.
- La collaboration optimisée entre les équipes : Les équipes peuvent travailler indépendamment sur des services spécifiques, se concentrant ainsi davantage sur les besoins métier de chaque service.

Les points faibles de l'architecture microservices incluent :

- La complexité : L'architecture microservices est plus complexe que l'architecture monolithique en raison de la gestion de nombreux services autonomes.
- Les coûts de développement : La création et la maintenance de multiples microservices peuvent être coûteux, il est donc important de considérer ces coûts lors de la planification de l'architecture applicative.
- Les tests : Les tests d'intégration peuvent être plus compliqués à mettre en œuvre car ils doivent englober plusieurs services.
- La gestion des configurations : La gestion des configurations peut devenir plus ardue, car elle doit être effectuée pour chaque microservice individuellement.
- La complexité opérationnelle : La gestion de l'infrastructure de production pour un grand nombre de microservices peut être complexe et requérir des compétences spécialisées en architecture et en opérations.

En résumé, l'architecture microservices permet une meilleure flexibilité et une évolutivité plus facile en découpant l'application en petits services indépendants qui peuvent être développés, testés et déployés de manière autonome.

Chaque service est spécialisé dans une fonctionnalité spécifique et communique avec les autres services via une API.

Elle facilite également le déploiement, avec des services pouvant être déployés indépendamment les uns des autres. Cela permet des mises à jour plus rapides et faciles en réduisant les temps d'arrêt et augmentant l'efficacité et la productivité des équipes de développement.

IV/ Historique de l'architecture d'Uber

L'architecture d'Uber a évolué au fil des années pour répondre aux besoins d'une entreprise en pleine croissance.

Voici un résumé de l'historique de l'architecture d'Uber et les technologies impliqués :

1. Architecture Monolithique (2010-2012) : au début, Uber avait une architecture monolithique avec un backend en Python sur Postgres.
2. Architecture de Backend Python et Postgres (2012-2014) : Uber a conservé l'architecture de backend Python sur une base de données Postgres pour permettre une meilleure évolutivité.
3. Architecture de Backend à grande échelle (2014-2015) : avec l'expansion rapide d'Uber, l'entreprise a connu des problèmes de performance et a décidé de repenser son architecture backend. Uber a donc déplacé de nombreuses applications de Postgres vers MySQL.
4. Architecture Microservices (2015-2016) : pour répondre aux besoins de la croissance continue d'Uber, l'entreprise a finalement choisi d'adopter une architecture microservices pour son évolutivité, sa flexibilité et sa maintenabilité. Uber a également commencé à utiliser Docker pour simplifier la gestion des environnements de service locaux.
5. Utilisation de NGINX et HAProxy (2016) : Uber a commencé à utiliser NGINX et HAProxy pour gérer leur architecture front-end et à utiliser Buck pour son code iOS.
6. Architecture orientée événement et Kubernetes (2017-2019) : Au fil du temps, l'architecture microservices d'Uber est devenue plus sophistiquée avec une meilleure gestion des API une plus grande utilisation de Kubernetes avec une architecture orientée événement.
7. Uber Web (2020) : En 2020 Uber a annoncé une nouvelle architecture appelée "Uber Web" basée sur des microservices, Kubernetes, Go et Cassandra.

Ces étapes dans l'historique de l'architecture d'Uber montre les différentes étapes de sa croissance en montrant l'importance de choisir la bonne architecture pour répondre aux besoins spécifiques d'une entreprise en passant d'une architecture backend monolithique à une architecture microservices

V/ Comparaison entre Uber et Bolt

En tant que concurrent d'Uber, Bolt est aussi une plateforme de mobilité offrant des services de transport.+

1. Structure en microservices

Comme Uber, Bolt a choisi la structure en microservices pour gérer la complexité de son application. Les deux entreprises ont compris les avantages de diviser leurs applications en petits services indépendants pouvant être développés, déployés et mis à l'échelle séparément.

2. Langages de programmation et technologies

Uber utilise principalement des langages comme Python, Go et Java, ainsi que des technologies telles que Cassandra, MySQL et Kubernetes. De son côté, Bolt emploie surtout Java, Kotlin et Python pour ses services backend. Les deux entreprises se servent de langages et de technologies similaires bien qu'il puisse y avoir des différences dans leur mise en œuvre.

3. Gestion des données

Uber et Bolt doivent gérer un grand volume de données en temps réel pour offrir un service de transport efficace et fiable. Les deux entreprises se servent de bases de données distribuées pour gérer les données de leurs application avec Uber utilisant principalement MySQL et Cassandra et Bolt avec PostgreSQL.

4. Évolutivité

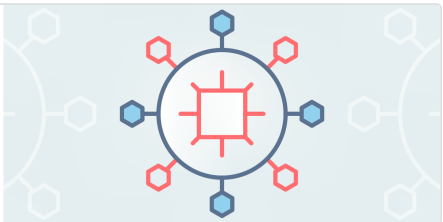
Uber et Bolt rencontrent des défis similaires en matière d'évolutivité en raison de leur croissance rapide et de l'augmentation du nombre de requêtes et de transactions à traiter. Les deux entreprises ont adopté des technologies et des approches similaires pour assurer l'évolutivité de leurs applications, notamment en utilisant des micro services et des outils de gestion de conteneurs comme Docker et Kubernetes.

Sources

Microservices : définition et architecture

Découvrez ce qu'est l'architecture en microservices et comment elle répond aux nouveaux enjeux de traitement de la donnée et d'évolutivité des systèmes.

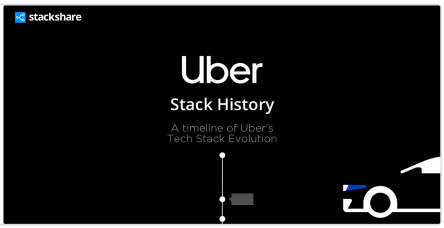
https://www.talend.com/fr/resources/guide-microservices/



Stack History: A Timeline of Uber's Tech Stack Evolution | StackShare

The story of Uber's major technology decisions over time. It's a story that underlies all other Uber narratives, and one that powers its meteoric growth even today.

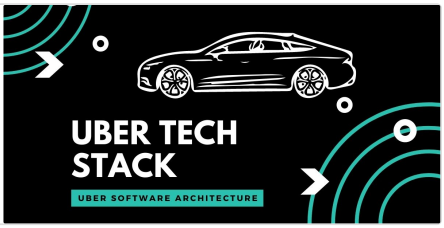
https://stackshare.io/stack-history-timeline-uber-tech-stack-evolution



Uber Tech Stack & Software Architecture - How Was Uber Built?

Check out this blog to know about uber technology stack and software architecure. Explore how uber was build.

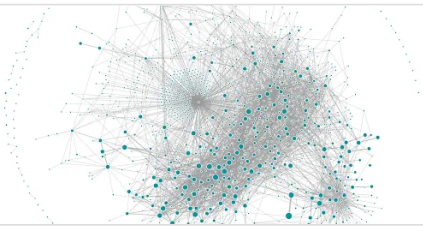
https://www.emizentech.com/blog/uber-tech-stack-software-architecture.html



Introducing Domain-Oriented Microservice Architecture | Uber Blog

Recently there has been substantial discussion around the downsides of service oriented architectures and microservice architectures in particular. While only a few years ago, many people readily adopted microservice architectures due to the

https://www.uber.com/en-FR/blog/microservice-architecture/



Qu'est-ce que l'architecture de microservices ? | Google Cloud | Google Cloud

L'architecture de microservices permet aux applications de séparer les applications en services indépendants. En savoir plus via les diagrammes d'architecture de microservices

https://cloud.google.com/learn/what-is-microservices-architecture?hl=fr