

Rapport écrit sur l'architecture logicielle de Twitter



Faty LO

Yassine BELLAGRAA

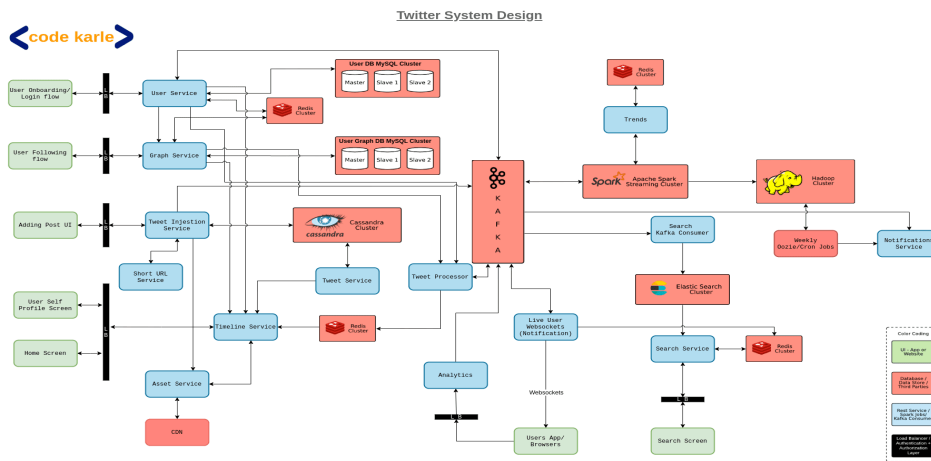
Sommaire

- I. L'architecture de Twitter
 - 1. Description générale
 - 2. Présentation de chaque service
 - 3. Evolution de l'architecture au cours du temps

- III. Concurrents de Twitter
 - 1. Entreprises concurrentes à Twitter
 - 2. Les avantages et inconvénients de cette architecture

I. L'architecture de Twitter

1. Description générale

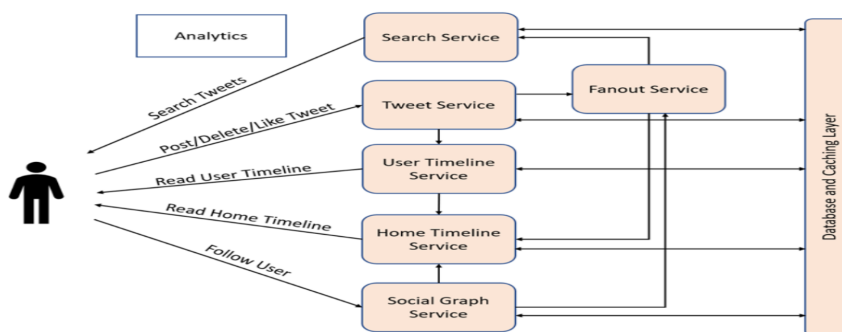


Twitter est un service de réseautage social en ligne où les utilisateurs publient et lisent des “tweets” se basant sur l’architecture de microservices.

L'architecture des microservices est une architecture utilisée dans le développement d'applications, permettant de décomposer celle-ci en composants indépendants avec chacun ses responsabilités.

Dans le cas de Twitter, pour répondre à une requête d’un utilisateur unique, l’application peut utiliser plusieurs microservices internes pour fournir une réponse que nous aborderons dans la partie suivante.

2. Présentation de chaque service



- Le service de Tweet

Un tweet est une information partagée à ses abonnés qui peut prendre la forme de photos, de GIF, de vidéos, de liens et de texte.

Il réceptionne les tweets d'un utilisateur, les redirige vers l'historique de ses abonnements et vers le service de recherche.

- Le service de graphe social

Il garde une trace des abonnements et des abonnés d'un utilisateur.

- Le service de l'historique utilisateur

Il retourne à l'utilisateur son historique, c'est-à-dire tous ses tweets rangés par ordre décroissant de création.

Ce service est aussi utilisé par le service de l'historique d'accueil ou un autre historique utilisateur.

- Le service de distribution

Il redirige les nouveaux tweets (qui viennent d'être créés) vers le service de recherche et le service de l'historique d'accueil.

Cela donne parfois lieu à d'autres microservices tels que le service des tendances ou le service des notifications.

- Le service de l'historique d'accueil

Il affiche l'historique d'accueil de l'utilisateur qui centralise tous les tweets de ses abonnements classés par ordre décroissant de création.

On en déduit donc une certaine similitude avec le service de l'historique utilisateur, mais c'est un peu plus complexe que ça.

- Le service de recherche

Il répond aux requêtes de l'utilisateur en lui présentant tous les tweets correspondant à ses critères de recherche.

3. Evolution de l'architecture au cours du temps



Le concept de séparer une application en plusieurs services n'est pas nouveau, l'architecture de microservices (*MOA : microservice architecture*) est une évolution de l'architecture orientée service (*SOA : service-oriented architecture*) qui existait depuis les années 2000.

L'une des principales différences réside sur l'indépendance des services dans la MOA, or dans la SOA un service peut être composé de plusieurs services. Ces derniers ne sont donc pas autonomes.

Au fur et à mesure de sa durée de vie, un projet va avoir tendance à grossir. Comme tous les grands services Internet, alors que le succès était grandissant, l'architecture mise en place aux débuts de Twitter, l'architecture monolithique, est devenue inadaptée dû à l'évolution exponentielle du nombre de tweets.

En effet, l'architecture monolithique qu'utilisait Twitter avait une large portée, étant conçue pour traiter de multiples tâches connexes. Elle peut contenir un serveur web, un équilibreur de charge, un catalogue qui propose des images de produits, un système de commande, une fonction de paiement et une composante d'expédition. C'est ce qui est à l'origine de l'adoption de la MOA pour faire face à l'augmentation inéluctable de la complexité du code et rendre souple un projet complexe en le développant sous forme de mini service simple.

II. Concurrents de Twitter

1. Entreprises concurrentes à Twitter

Parmi les principaux concurrents à Twitter, on peut citer Snapchat et TikTok qui utilisent également eux aussi l'architecture de microservices.

TikTok pour garder un système rapide et efficace en se basant sur une infrastructure cloud native et ainsi profiter de ses avantages.

Snapchat qui lui a permis de réduire le temps de latence et d'augmenter la fiabilité de l'application. Elle voulait également des services indépendamment déployables ce qui l'a conduit aux microservices.

Lorsque ces plateformes évoluent, cela devient difficile de se contenter d'une architecture monolithe.

Le passage à l'architecture de microservices est généralement dû au passage d'un nombre d'utilisateurs conséquent, mais dans d'autres situations ce n'est pas la raison principale...

2. Les avantages et inconvénients de cette architecture

Étant créé pour répondre à des besoins de traitements de la donnée et d'organisation auxquels les architectures SOA monolithiques ne pouvaient plus répondre, l'architecture microservice présente des avantages et des inconvénients pour Twitter.

Parmi les avantages, on peut citer :

- La réduction du temps de développement grâce à l'indépendance des services distribués, plusieurs services peuvent être développés simultanément. La durée du développement de l'application est de ce fait réduite, car plusieurs développeurs peuvent intervenir sur le projet en même temps sans se déranger et nuire au travail de l'autre.

- Une meilleure flexibilité, des microservices qui peuvent être développés en utilisant différents langages de programmation et peuvent être hébergés sur des serveurs différents. Cela permet à Twitter de choisir les meilleures technologies et infrastructures pour chaque service.
- Une réduction des risques du fait que les microservices soient indépendants. Lorsqu'un élément tombe en panne ou rencontre un problème, l'ensemble de l'application ne cesse pas de fonctionner contrairement aux applications monolithiques. Il permet également de réduire les risques liés à l'erreur humaine donc plus facile d'identifier et de résoudre une panne dans ce type d'écosystème.

Parmi les inconvénients, on peut citer :

- Une distribution proportionnée du système, car il va falloir gérer toutes les interconnexions entre composants et aussi avec la base de données
- Un ensemble de critères à prendre en compte au préalable tels que la configuration externalisée, la journalisation, les métriques, les contrôles de santé, etc.
- Une difficulté dans la réalisation de tests étant donné qu'on a une multitude de composants qui sont déployables indépendamment, il faut tester ces composants, mais en plus les interactions qu'ils ont avec les autres composants, cela rend la tâche un peu plus difficile

En résumé, l'utilisation d'une architecture de microservices pour Twitter présente des avantages significatifs, notamment en termes d'évolutivité, d'optimisation et de réductions de risques. Cependant, cela peut également présenter des défis tels que la complexité, les coûts de développement et de maintenance et la surveillance et le débogage plus complexe.

Références

<https://interviewnoodle.com/twitter-system-architecture-8dafce16aec4>

<https://cloud.google.com/learn/what-is-microservices-architecture?hl=fr#section-2>

<https://anceret-matthieu.fr/2021/07/larchitecture-micro-services/>

<https://www.talend.com/fr/resources/monolithic-architecture/>

<https://www.lavivienpost.com/how-tiktok-works-architecture-illustrated/#4>

<https://eng.snap.com/monolith-to-multicloud-microservices-snap-service-mesh>

<https://mobiskill.fr/blog/conseils-emploi-tech/microservices-ou-monolithes-quelle-architecture-choisir/>

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