Étude de cas

Vanguard

Refonte d'un système monolithique en microservices : la qualité du code monte en gamme avec les graphes

SECTEUR
Services financiers

CAS D'USAGE
Modernisation d'applications

OBJECTIF
Réagir rapidement pour rester compétitif, éviter les perturbations et être conforme à la réglementation

DÉFI
Refondre une importante base de code de production en microservices

SOLUTION
Recours à Neo4j pour saisir et analyser toutes les relations dans le code historique et les microservices

RÉSULTATS
- Métriques de qualité de code objectives, partagées et améliorées
- Risque réduit par des analyses d'impact et la mise en œuvre des meilleures pratiques

Vanguard Group devait remanier en micro-services un amalgame de code stratégique existant. En captant toutes les dépendances dans le graphe, l'équipe rationalise progressivement son architecture et effectue la refonte avec la collaboration de toute l'entreprise pour optimiser les métriques de qualité du code issues de l'analyse du graphe.

L'entreprise

Vanguard est l'une des plus grandes sociétés de gestion de placements au monde, avec plus de 3,5 billions de dollars en gestion. Elle est le plus important fournisseur de fonds communs d'investissement et le deuxième fournisseur de fonds négociés en bourse (ETF). Avec ses 17 600 employés, Vanguard propose ses services à plus de 20 millions d'investisseurs dans environ 170 pays. Son fondateur, John Bogle, est le créateur des fonds indiciels.

Le défi

Les entreprises de services financiers ont besoin d'une infrastructure informatique de pointe. La refonte des systèmes Java monolitiques en place chez Vanguard en de multiples microservices nécessitait une visibilité sur tous les composants et leurs dépendances. Certaines archives Java (jars) historiques de la société comptaient **3 à 4 millions de lignes de code**.

De plus, Vanguard subissait un déficit technique. Il lui fallait élaguer du code caduc, alors que les évaluations d'impact s'avéraient difficiles.

« Gérer du code est critique » note John Lavin, architecte logiciel chez Vanguard. « La priorité va souvent à la fourniture des fonctionnalités, à la commercialisation et au bouclage de la tâche. Mais si rien n'est fait pour que le code soit simple à gérer, aucun de ces objectifs n'est atteint. »

Pour commencer, l'équipe a essayé de visualiser les dépendances des jars dans un outil de poste de travail. Les tentatives pour télécharger toutes ces archives dans l'outil l'ont fait tomber, car il n'était pas conçu pour fonctionner à l'échelle de Vanguard.

Refondre des jars à plusieurs millions de lignes de code en microservices nécessite de gérer beaucoup de composants mobiles. Pour y parvenir, l'équipe a commencé à gérer les services dans un tableur compilé en un an.

« Nous avions de nombreux services différents avec divers statuts. Nous avons tenté de rassembler les choses et de documenter les dépendances au mieux dans le tableur. Mais ça ne pouvait pas fonctionner sur le long terme » relate John Lavin.

Vanguard devait modéliser tous ses services et leurs dépendances dans le graphe pour mener des analyses d'impact indiquant combien de services pouvaient être affectés par la défaillance de l'un d'eux. Les passerelles API qui ont recours au temps d'exécution pour capter les dépendances ne convenaient pas à Vanguard, car elles auraient omis certains services clés émettant des appels uniquement à certains moments de l'année.

Étude de cas

« Les gestionnaires vont directement dans les tableaux pour savoir si leurs indicateurs sont à la hausse ou à la baisse. La technologie des graphes nous donne un excellent moyen de tirer les métriques vers le haut et de garder un code propre, car tout le monde le consulte »

– John Lavin
architecte logiciel,
Vanguard Group

La solution

Il fallait à l'équipe une solution flexible, capable de s'adapter à des niveaux sans précédent. « Nous nous sommes rendu compte que la gestion de nos modules et services était une problématique de graphe » poursuit Johl Lavin. Vanguard a alors choisi Neo4j pour son évolutivité et sa flexibilité.

L'équipe a commencé par un modèle de données de graphe simple, avec l'ajout automatique des dépendances de jars lors de l'exécution d'une tâche de compilation. Ensuite, tous leurs attributs de code existants ont été ajoutés au graphe avec leurs dépendances.

En utilisant l'analyse de graphe, l'équipe a commencé à évaluer son code au regard des meilleures pratiques et des métriques extraites. Elle a ensuite ajouté des informations tirées de son tableur d'architecture pour enrichir son modèle.

L'équipe a mis au point des outils pour visualiser les relations et appliquer les meilleures pratiques, par exemple la restriction du nombre d'appels entre services pour réduire les risques. Un tableau d'évaluation de la qualité du code basé sur des métriques dérivées du graphe permet une collaboration entre managers et développeurs dans toute l'entreprise.

Les résultats

Vanguard a gagné une visibilité complète sur les dépendances entre jars existants et microservices, ce qui lui permet de mener des analyses d'impact et de moderniser ses systèmes monolithiques de façon incrémentielle.

Grâce à la saisie de toutes les relations et dépendances dans Neo4j, l'équipe résout de façon efficace le déficit technique au fur et à mesure de l'avancement et réalise des améliorations concrètes.

« Avec des métriques simples et un peu de maths, on peut obtenir de nombreuses métriques de module grâce à des requêtes Cypher simples » déclare John Lavin.

Vanguard visualise les métriques dérivées de Neo4j dans un tableau d'évaluation de la qualité du code qui donne un score à chaque module. Ce tableau est partagé dans l'entreprise, accompagné d'une visualisation du graphe sous-jacent sous forme de tableur. Ainsi, les métiers peuvent voir les tendances et évaluer l'avancée du projet de modernisation des applications.

« Les gestionnaires vont directement dans les tableaux pour savoir si leurs indicateurs sont à la hausse ou à la baisse.

La technologie des graphes nous donne un excellent moyen de tirer les métriques vers le haut et de garder un code propre, car tout le monde le consulte. »

Les architectes utilisent Neo4j non seulement pour gérer l'état actuel de tout le code Vanguard, mais aussi pour progresser vers l'état désiré en maximisant les efforts des développeurs et en favorisant la réutilisation.

« Lorsqu'une équipe projet entre dans sa phase de planification, nous fournissons des informations sur les services à utiliser, là où ils sont utilisés, ce qui est partiellement développé et l'endroit où les développeurs peuvent apporter leur logique pour que le tout ait du sens dans notre modèle architectural global » explique John Lavin.

Neo4j est la plus importante plateforme de bases de données de graphes qui permet à Airbus, Comcast, eBay, la NASA, UBS , Walmart et d'autres d'innover et de rester compétitifs. Des milliers de déploiements par la communauté et plus de 300 clients mettent à profit les données connectées avec Neo4j pour identifier la façon dont les personnes, les processus, les lieux et les systèmes sont interconnectés. Grâce à cette approche par les relations, les applications mises au point en utilisant Neo4j relèvent les défis associés aux données connectées, dont l'intelligence artificielle, la détection de fraude, les recommandations en temps réel et les données de référence.
Pour en savoir plus, consulter neo4j.com.

Des questions sur Neo4j ?

Contactez-nous :
info@neo4j.com
neo4j.com/contact-us