Conception et développement de la première version d'un annuaire de services

31 Janvier 2019

TELECOM SudParis

Contributors	**Organization**
Yuwei Wang	Télécom SudParis

Coordinators	**Organization**
Kavoos Bojnourdi	EDF Labs
Sophie Chabridon	Télécom SudParis
Denis Conan	Télécom SudParis

Document versions	
Version 1.0	27 Janvier, 2018

1 Introduction

L'architecture à base de microservices est un patron d'architecture qui décompose les systèmes répartis en services découplés. Chacun de ces services remplit une tâche spécifique qui peut être développée et déployée indépendamment [1]. Il aide à faciliter la capacité de montée en charge (*scalability* en anglais), l'agilité, la fiabilité et les autres aspects par rapport aux architectures monolithiques [2].

Cependant, ces avantages présentent aussi des challenges, la découverte de services est l'un d'entre eux. La découverte de services est le processus qui permet de déterminer et détecter les localisations des services dynamiquement, de les enregistrer et les gérer constamment [3]. Il considère aussi des stratégies de l'équilibrage de charge, la surveillance, les problèmes de disponibilité etc.

Ce document présente un projet d'un annuaire de services, un des composants importants de la découverte des services. La section 2 explique la conception du prototype. La section 3 détaille le développement du projet et les technologies utilisées. La solution du déploiement basé sur les conteneurs est présentée dans la section 4. Les résultats sont conclu dans la section 5.

2 Conception

La découverte des services joue un rôle important dans les architectures à base des microservices. En général, ses composants principaux sont [4]:

- **les fournisseurs des services** : sont les services qui offrent des fonctionnalités aux autres services. Ces services changent souvent en raison du changement de l'environnement dynamique, par exemple la défaillance, la montée en charge, le déploiement, etc.
- **les consommateurs des services** : sont les services qui demandent les fonctionnalités fournies par les autres services.
- **l'annuaire de services** : est un service qui peut être utilisé par d'autres composants pour récupérer des informations (les localisations avec des métadonnées du service). Les microservices parlent à l'annuaire de services pour enregistrer leurs localisations, alors que les consommateurs s'adressent à l'annuaire pour découvrir les services enregistrés.

Dans certain cas, un microservice est ne joue pas seulement un rôle des consommateurs mais également des fournisseurs. La figure 1 présente une structure basique de la découverte des services avec trois composants principaux.

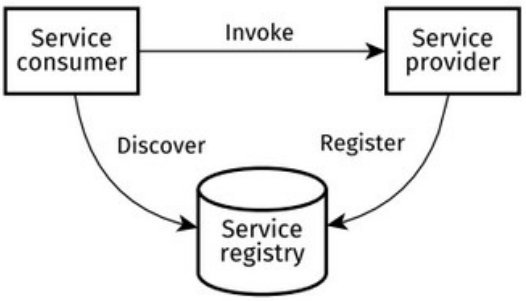

Figure 1 : La structure de la découverte des service [4]

Dans ce projet, un premier annuaire de services avec des fonctions de base a été réalisé. La figure 2 représente un point de vue de l'architecture ensemble du projet de l'annuaire de services. Dans cette figure, l'application suit la même philosophie que l'architecture dans le figure 1.

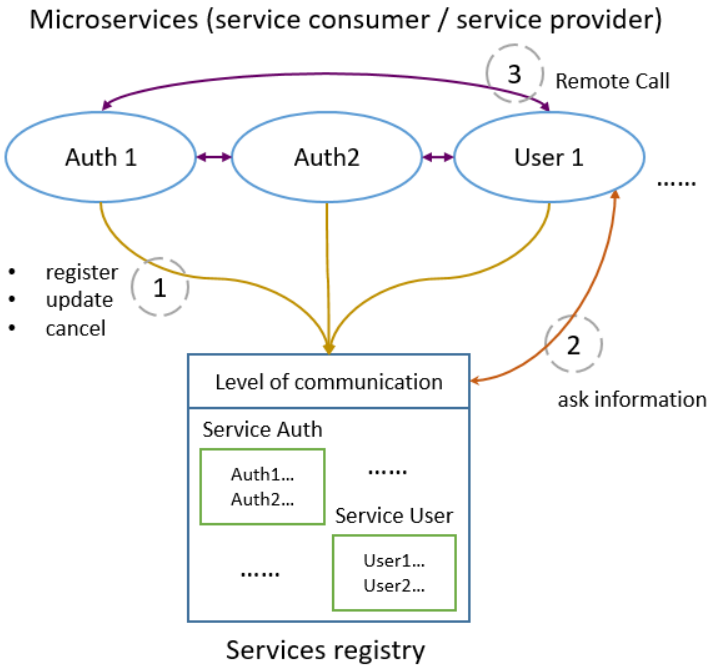

Figure 2 : L'architecture globale du projet

Il se compose de plusieurs composants :

- **les microservices en tant que fournisseurs des services** : Les microservices s'enregistrent dans l'annuaire de services lorsqu'il démarre. Ils envoient les informations d'enregistrement qui contiennent par exemple le nom du service, l'adresse IP, la version du service, les propriétés de la qualité du service, etc. à l'annuaire de services via une couche de communication. Un microservice possède plusieurs instances pour prévenir les imprévus (« Auth1 », « Auth2 » dans la figure 2 sont un microservice qui s'appelle « Auth » avec deux instances, et de la même façon pour « User1 », « User2 »).
- **les microservices en tant que consommateur des services** : Les microservices peuvent interroger l'annuaire de service aussi via la couche de communication. Ils envoient à l'annuaire le nom et la version de service comme la requête et ensuite reçoivent des retours avec les informations de fournisseurs des services et leurs TTLs (le temps de vie, *Time To Live* en anglais, abrégé TTL).
- **l'annuaire de services** : Après recevoir les informations d'enregistrement, l'annuaire de services va les stocker dans une dictionnaire de services. En plus, chacun des microservices dans ce projet doit pouvoir remonter les informations à l'annuaire, c'est-à-dire que l'annuaire peut interroger régulièrement les microservices pour savoir les qualités de services, par exemple la charge pour l'instant.

La couche de communication dans l'architecture est bien séparée pour qu'on puisse la remplacer plus facilement par différents protocoles de communication tels que HTTP, TCP, etc. et formats d'échange des données tels que JSON, XML, Protocol Buffers **, etc.

*** Protocol Buffers : un format de sérialisation développé par Google, documentation officielle https://developers.google.com/protocol-buffers/*

3 Développement

Dans un premier temps, ce projet de l'annuaire de services se communique basé sur une API REST avec le serveur HTTP pour la couche de communication. En plus, la base des données utilise PostgreSQL pour l'instant.

L'application est développée en langage Go (aussi appelé Golang*)*. Les microservices sont pris en charge par à peu près toutes les langues. On choisit le Golang comme le langage de programmation en plusieurs de raisons [5] [6]:

- Golang est un langage compilé, il construit directement un fichier en binaire plus petit connu par le processus dans le bas niveau de l'ordinateur, ce qui peut bien améliorer la performance.
- Les applications écrits en Golang facilitent le déploiement et la situation de la montée en charge parce qu'on a besoin de aucun dépendances et librairies d'exécution supplémentaires à gérer quand les déployer.
- Golang est conçu, supporté et utilisé par Google pendant dizaine années pour résoudre les problèmes des grandes infrastructures logicielles chez Google. Selon le retour d'expérience de Google, l'objectif de Golang est d'éliminer la lenteur et la maladresse du développement logiciel en face d'une croissante de cluster des machines et de rendre plus productif et évolutif [7].

La figue 3 présente une architecture logicielle du projet.

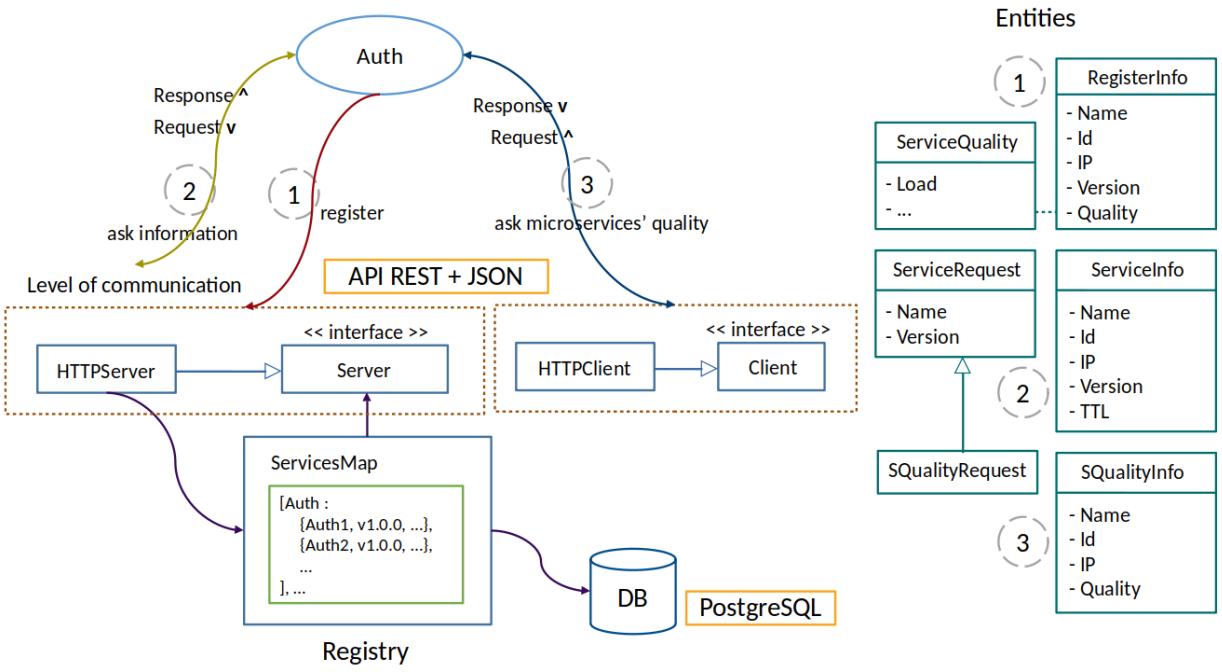

Figure 3 : L'architecture logicielle du projet

L'annuaire de service (« *Registry* » dans la figure 3) connecte une base de données (« *DB* » dans la figure 3) pour gérer tous les informations d'enregistrement venant de microservices enregistrés (l'opération « 1 *register* » dans la figure 3).

Il ouvre une connexion en utilisant une interface de serveur qui est implémentée par un serveur HTTP avec l'API REST en tant que la couche de communication. Les autres microservices

communiquent via cette couche au lieu de communiquer directement avec l'annuaire de services, par exemple demander les informations des autres microservices (l'opération « 2 *ask information* » dans la figure 3).

En même temps, l'annuaire de services a aussi une interface de client qui est implémentée par un client HTTP pour demander périodiquement la qualité de microservices (l'opération « 3 *ask microservices' quality* » dans la figure 3).

Le format des données d'échange est JSON. Dans le futur, les interfaces peuvent être implémentées par d'autres protocoles de communication et d'autre formats des données.

La figure 4 présente une diagramme de classe. Parmi ces classes, le paquet « rentities » contienne tous les entités de l'application. « *RegisterInfo* » expose les informations d'enregistrement. « *ServiceRequest* » et « *ServiceInfo* » présente la requête de microservices et la réponse correspondante de l'annuaire. « *SQualityReq* » et « *SQualityInfo* » sont la requête de l'annuaire et la réponse correspondante de microservices. « *Auth* » est une exemple de microservices qu'on a développé.

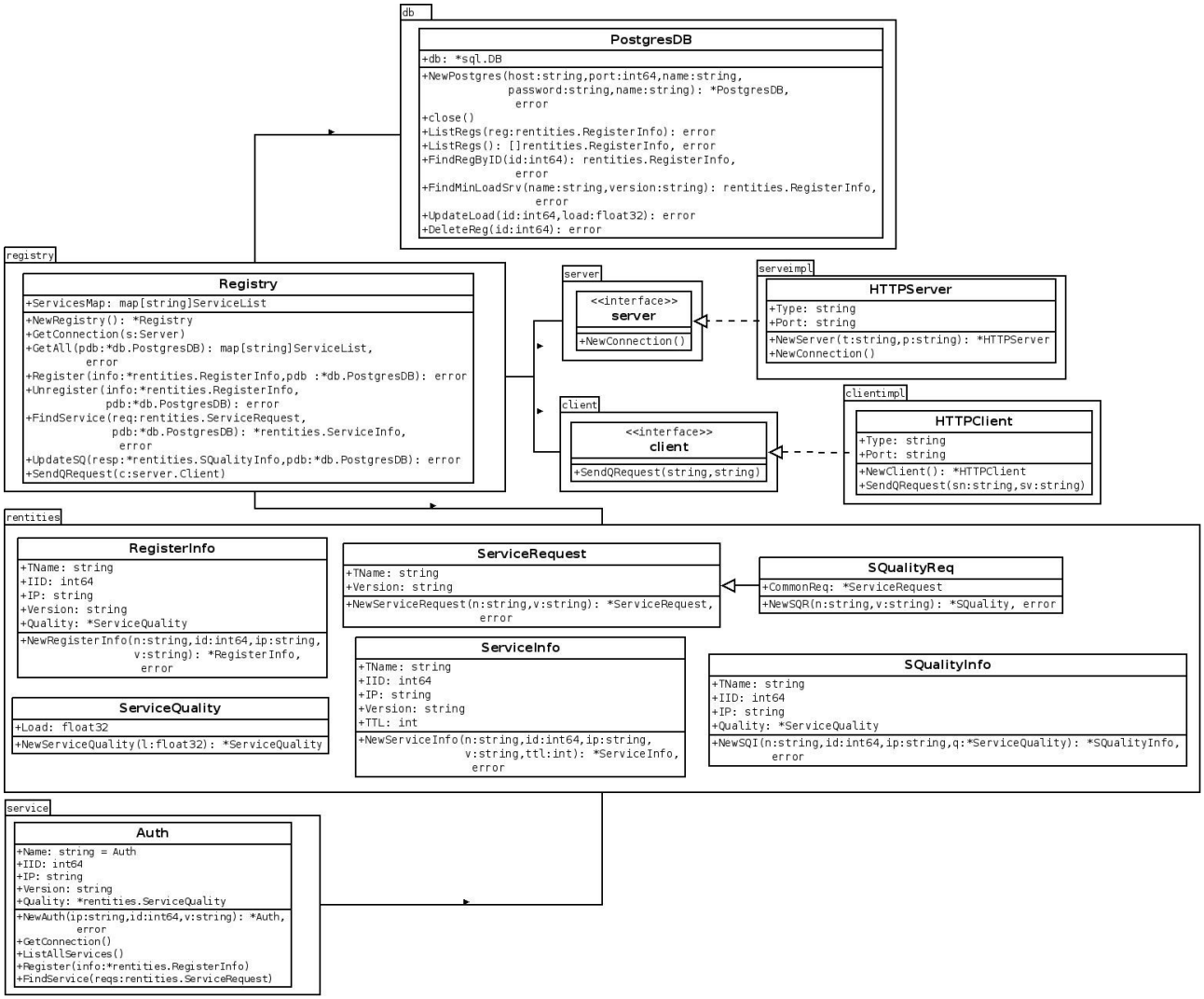

Figure 4 : Diagramme de classe

Après finir la programmation, il est nécessaire de faire des tests unitaires. D'abord, des tests en local dans une même machine pour simuler la processus des trois opérations (les cercles avec des numéros 1 2 3 dans la figure 3) sont effectués. L'annuaire de service utilise le port « 8080 » en « *localhost* » et les microservices « Auth » utilise le port « 8081 » en « *localhost* ». La base des données de PostgreSQL écoute le port « 5432 » en « *localhost* ».

4 Déploiement

La technologie de conteneurs est récemment utilisée fortement dans le déploiement des applications parce que les conteneurs sont plus légers et plus faciles à gérer comparés par les machines virtuelles traditionnelles [8]. Ils ont besoins moins de mémoire et moins de coûts d'infrastructure. Les architectures à base de microservices se composent de plusieurs petits services de manière modulaire. En utilisant Docker, chaque service de l'application peut être packagé dans un conteneur.

Donc, après passer les tests en local, on déploie l'application dans les conteneurs Docker. La figure 5 illustre le déploiement de l'application dans Docker.

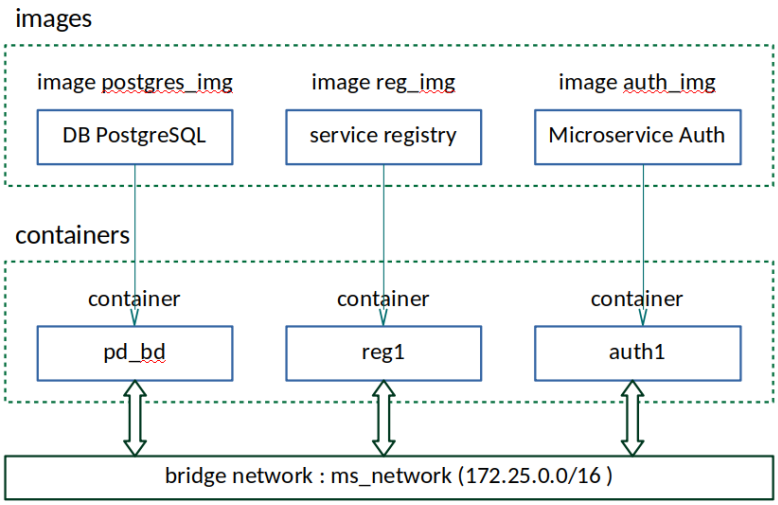

Figure 5 : illustration du déploiement dans Docker

Trois images sont construits. A partir de ces images, trois instances des images qui s'appellent les conteneurs sont créés : l'image « *postgres_img* » et le conteneur « pd_bd » pour la base des données PostgreSQL, l'image « *reg_img* » et le conteneur « reg1 » pour la partie de l'annuaire de services avec la couche de communication, l'image « *auth_img* » pouret le conteneur « auth1 » une exemple de microservices « *Auth* ».

Ces trois conteneurs connectent au même réseau de type passerelle (« *bridge network* » en anglais) pour communiquer. Donc un réseau de type passerelle en « 172.25.0.0/16 » appelé « *ms_network* » est défini. Il fournit une résolution DNS automatique entre les conteneurs, c'est-à-dire on peut utiliser le nom du conteneur au lieu de son adresse IP.

5 Conclusion

Pour conclure, cette première version de serveur de l’annuaire de services écrit en langage Go réalise les fonctionnalités basiques de la découverte de services : l’enregistrement de microservices et la demande d’un autre service. En plus, le déploiement dans les conteneurs Docker est aussi effectué. On définie cette version de l’annuaire de service comme la version 0.0.1.

L’étape suivante est de continuer à perfectionner cet annuaire de services, de faire des tests et chercher des défauts de cette version.

6 Références

[1] Z. Xiao, I. Wijegunaratne, and X. Qiang, "Reflections on SOA and Microservices", 2016 4th International Conference on Enterprise Systems (ES), pages 60–67, 2016.

[2] W. Hasselbring, G. Steinacker, "Architectures for Scalability Agility and Reliability in E-Commerce", Proc. IEEE Int'l Conf. Software Architecture Workshops (ICSAW 17), pp. 243-246, 2017.

[3] T. Cerny, M. J. Donahoo, M. Trnka, "Contextual Understanding of Microservice Architecture: Current and Future Directions", ACM SIGAPP Applied Computing Review, v.17 n.4, p.29-45, December 2017.

[4] C. Rotter, J. Illes, G. Nyiri, L. Farkas, G. Csatari, and G. Huszty, "Telecom Strategies For Service Discovery in Microservice Environments". In 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), pages 214--218, March 2017.

[5] F. S. Shoumik, M. I. M. M. Talukder, A. I. Jami, N. W. Protik and M. M. Hoque, "Scalable micro-service based approach to FHIR server with golang and NoSQL," in 2017 20th International Conference of Computer and Information Technology (ICCIT), 2017.

[6] M. Andrawos, M. Helmich, "Cloud Native programming with Golang", Birmingham: Packt Publishing, 2017.

[7] R. Pike, Google Inc, "Go at Google: Language Design in the Service of Software Engineering", https://talks.golang.org/2012/splash.article, SPLASH 2012 conference in Tucson, Arizona, October 25, 2012.

[8] V. Singh, S. K. Peddoju, "Container-based micro service architecture for cloud applications", 2017 International Conference on Computing Communication and Automation (IC-CCA), pp. 847-852, 2017.

9 rue Charles Fourier
91011 Évry Cedex
France
+33 (0)1 60 76 44 06

www.telecom-sudparis.eu