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

31 Janvier 2019

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, d'enregistrer ces services et de 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 de déploiement basée sur les conteneurs est présentée dans la section 4. Les résultats et la conclusion sont discutés 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** : ce sont les services qui offrent des fonctionnalités aux autres services. Ces services changent souvent en raison de variations dynamiques dans l'environnement, par exemple la défaillance, la montée en charge, le déploiement, etc.
- **les consommateurs des services** : ce sont les services qui demandent les fonctionnalités fournies par les autres services.
- **l'annuaire de services** : c'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 certains cas, un microservice ne joue pas seulement un rôle de consommateur mais également de fournisseur de service. La figure 1 présente une structure basique de la découverte des services avec trois composants principaux.

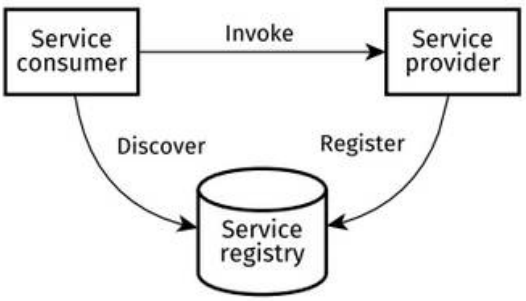

Figure 1 : La structure de la découverte des service [4]

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 de la figure 1.

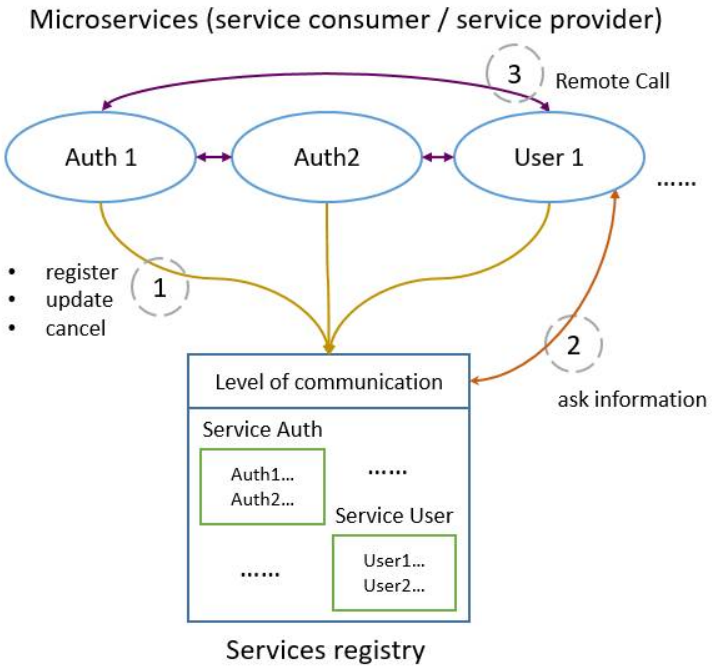

Figure 2 : L'architecture globale du projet

Il se compose de plusieurs composants :

- **les microservices en tant que fournisseurs de services** : Les microservices s'enregistrent dans l'annuaire de services lorsqu'ils démarrent. 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 représentent un microservice qui s'appelle « Auth » avec deux instances, et de la même façon pour « User1 », « User2 »).
- **les microservices en tant que consommateurs de services** : Les microservices peuvent également interroger l'annuaire de services via la couche de communication. Ils envoient à l'annuaire le nom et la version de service dans la requête et reçoivent ensuite en retour les informations des fournisseurs de services et leurs TTLs (le temps de vie, *Time To Live* en anglais, abrégé TTL).
- **l'annuaire de services** : Après la réception des informations d'enregistrement, l'annuaire de services va les stocker dans un dictionnaire de services. De 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 connaître la qualité de service, concernant 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 choisir également des 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 d'annuaire de services communique via une API REST avec le serveur HTTP pour la couche de communication. De plus, la base de 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 tous les langages de programmation. Nous avons choisi Golang comme langage de programmation pour plusieurs de raisons [5] [6]:

- Golang est un langage compilé, il construit directement un fichier en binaire plus petit connu par le processus de bas niveau de l'ordinateur, ce qui peut améliorer de manière importante les performances.
- Les applications écrites en Golang facilitent le déploiement et la gestion de la montée en charge car elles ne nécessitent aucunes dépendances ni librairies d'exécution supplémentaires à gérer au moment de les déployer.
- Golang est conçu, maintenu et utilisé par Google depuis une dizaine d'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 croissance de la taille des clusters de machines et de favoriser la productivité et évolutivité du développement logiciel [7].

La figure 3 présente l'architecture logicielle du projet.

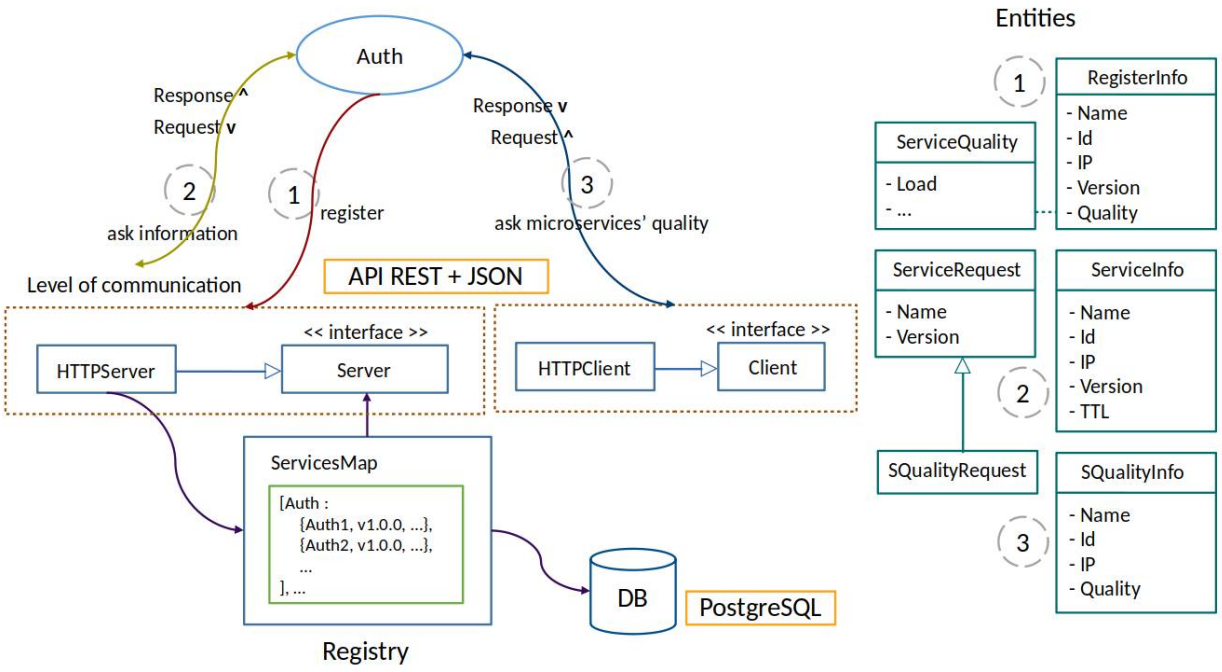

Figure 3 : L'architecture logicielle du projet

L'annuaire de services (« *Registry* » dans la figure 3) connecte une base de données (« *DB* » dans la figure 3) pour gérer toutes les informations d'enregistrement venant des microservices (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 couche de communication. Les autres microservices communiquent via cette couche au lieu de communiquer directement avec l'annuaire de services, par exemple pour demander les informations des autres microservices (opération « 2 *ask information* » dans la figure 3).

En même temps, l'annuaire de services a aussi une interface cliente qui est implémentée par un client HTTP pour demander périodiquement la qualité des microservices (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'autres formats de données.

La figure 4 présente un diagramme de classes. Parmi ces classes, le paquet « rentities » contient toutes les entités de l'application. « *RegisterInfo* » expose les informations d'enregistrement. « *ServiceRequest* » et « *ServiceInfo* » présentent la requête de microservices et la réponse correspondante de l'annuaire. « *SQualityReq* » et « *SQualityInfo* » sont la requête de l'annuaire et la réponse correspondante des microservices. « *Auth* » est une exemple de microservices que nous avons développé.

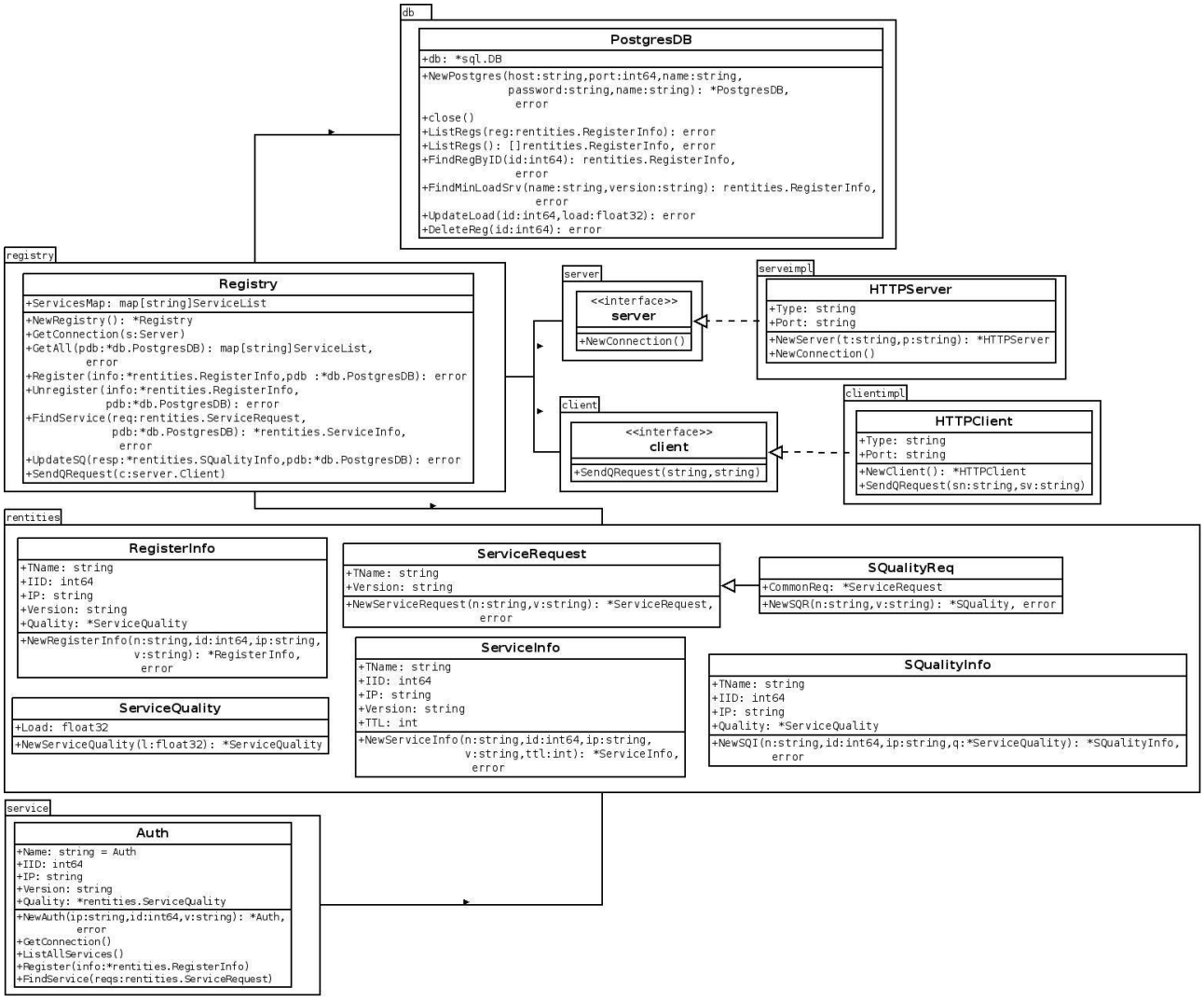

Figure 4 : Diagramme de classes

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

4 Déploiement

La technologie des conteneurs est utilisée fortement dans le déploiement des applications car les conteneurs sont plus légers et plus faciles à gérer comparés à des machines virtuelles traditionnelles [8]. Ils ont besoin de moins de mémoire et de 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.

Après avoir passé les tests en local, nous avons déployé l'application dans des conteneurs Docker. La figure 5 illustre le déploiement de l'application avec Docker.

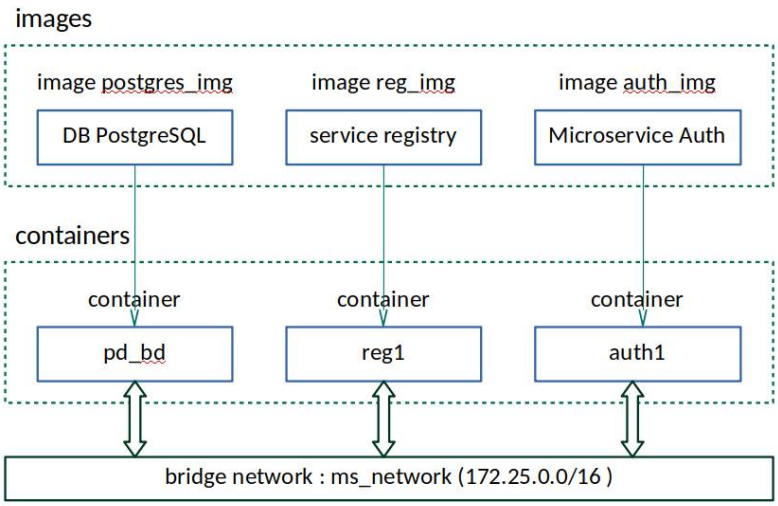

Figure 5 : illustration du déploiement avec Docker

Trois images sont construites. À partir de ces images, trois instances des images qui s'appellent les conteneurs sont créées : 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* » et le conteneur « auth1 » constituant un exemple de microservice « *Auth* ».

Ces trois conteneurs se 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, il est ainsi possible d'utiliser le nom du conteneur au lieu de son adresse IP.

5 Conclusion

Pour conclure, cette première version de serveur d'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. De plus, le déploiement dans les conteneurs Docker est aussi effectué. Cette version de l'annuaire de service correspons à la version 0.0.1.

L'étape suivante prévoit de continuer à perfectionner cet annuaire de services, de faire des tests et d'identifier les éventuels défauts de cette première 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