Architecture Microservices

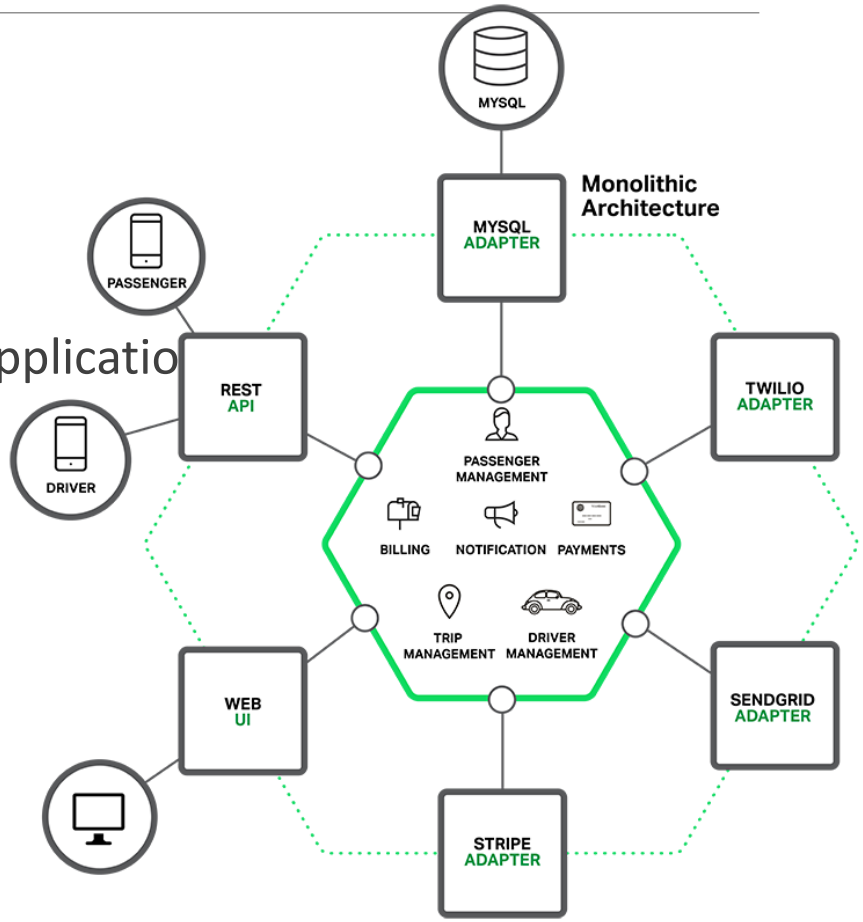
PLAN DU COURS

- DU MONOLITHIQUE VERS LES MICROSERVICES
- COMPOSANTS DE L' ARCHITECTURE MICROSERVICES:
  - API GATEWAY
    - RÔLE D'UN GATEWAY
    - TYPE DE GATEWAY
  - REGISTRY SERVICE
- DÉMO

Architecture monolithique

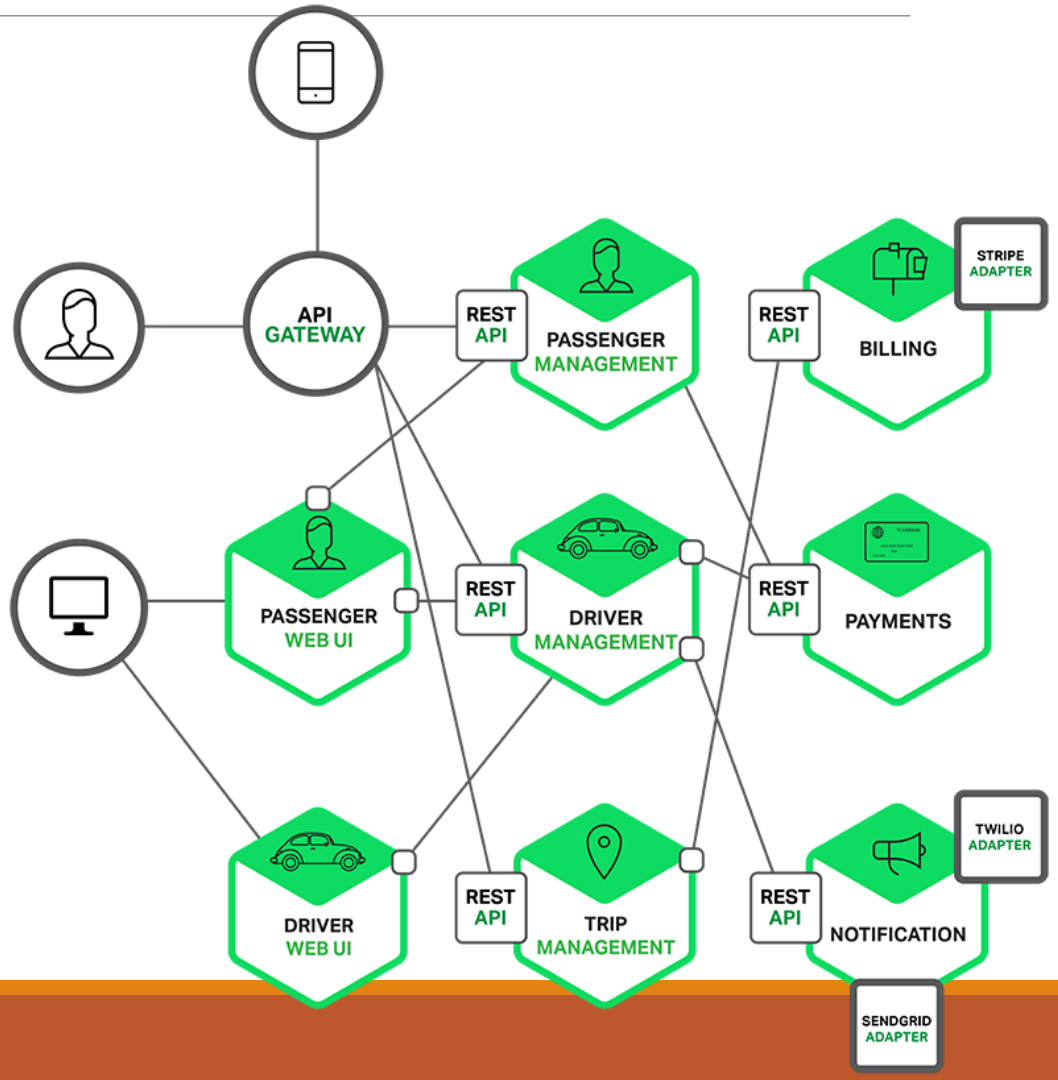
Inconvénients:

- Besoins fonctionnels centralisés dans un seul bloc
- Développé dans une seule technologie
- Chaque mise à jour nécessite le redéploiement de toute l'applicatio
- Difficulté de l'évolutivité



Solution: Architecture microservices

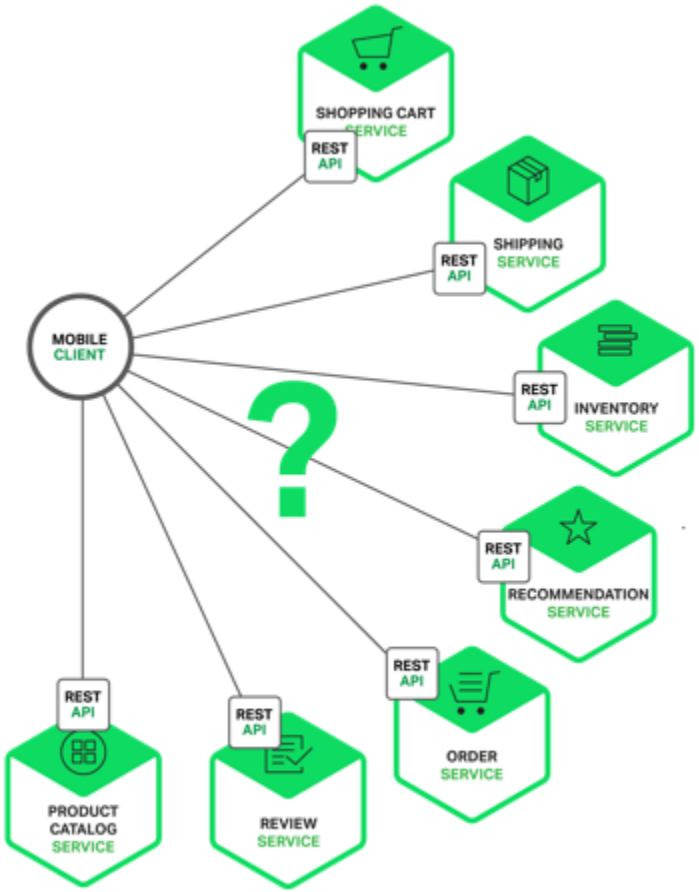
- Diviser l'application en plusieurs fonctionnalités, chacune sera développée comme étant un service indépendant de l'autre
- Déploiement de chaque service en plusieurs instances



Composants d’une architecture microservices

Problématique 1:

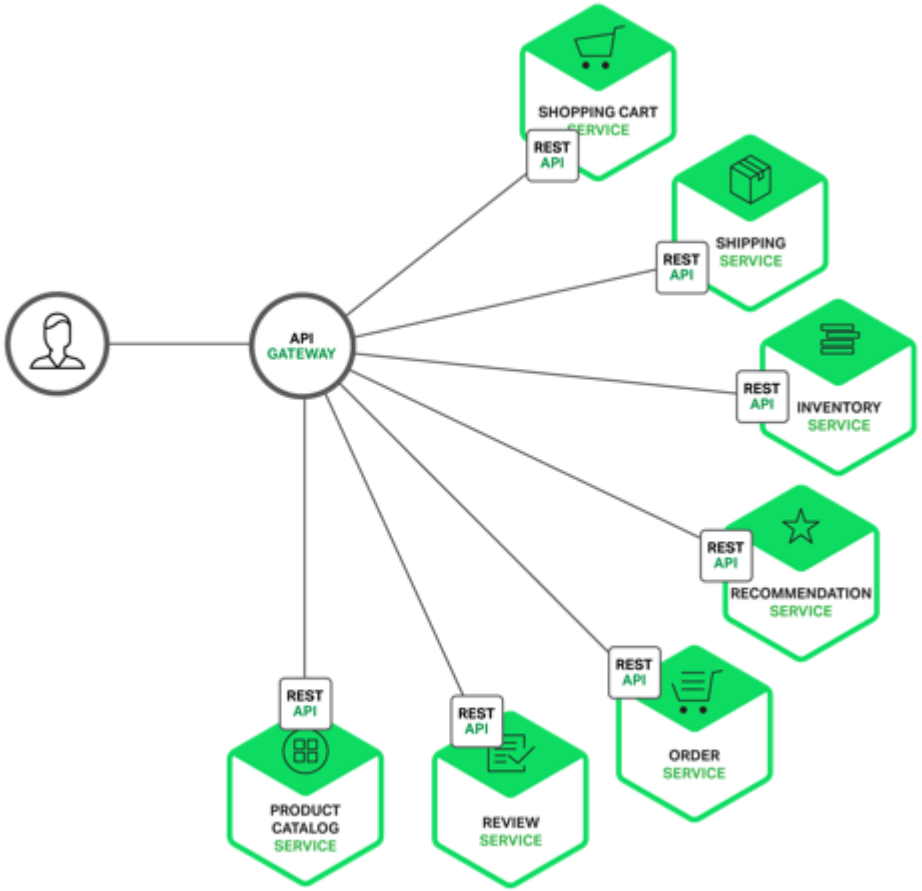
- Manque de sécurité
- Le client doit gérer tous les appels aux différents microservices
- Si l’application évolue et de nouveaux microservices sont introduits alors le client doit être notifié
- Couplage entre le client et les microservices



Composants d’une architecture microservices

Solution : utiliser une passerelle

- Point d’entrée unique dans un système d’information
- Encapsule la structure interne d’un système



Gateway: rôle

Routage des demandes:

- Un client doit d'abord envoyer sa demande au Gateway qui l'achemine au microservice approprié, pour retourner à la fin le résultat

Composition et Agrégation:

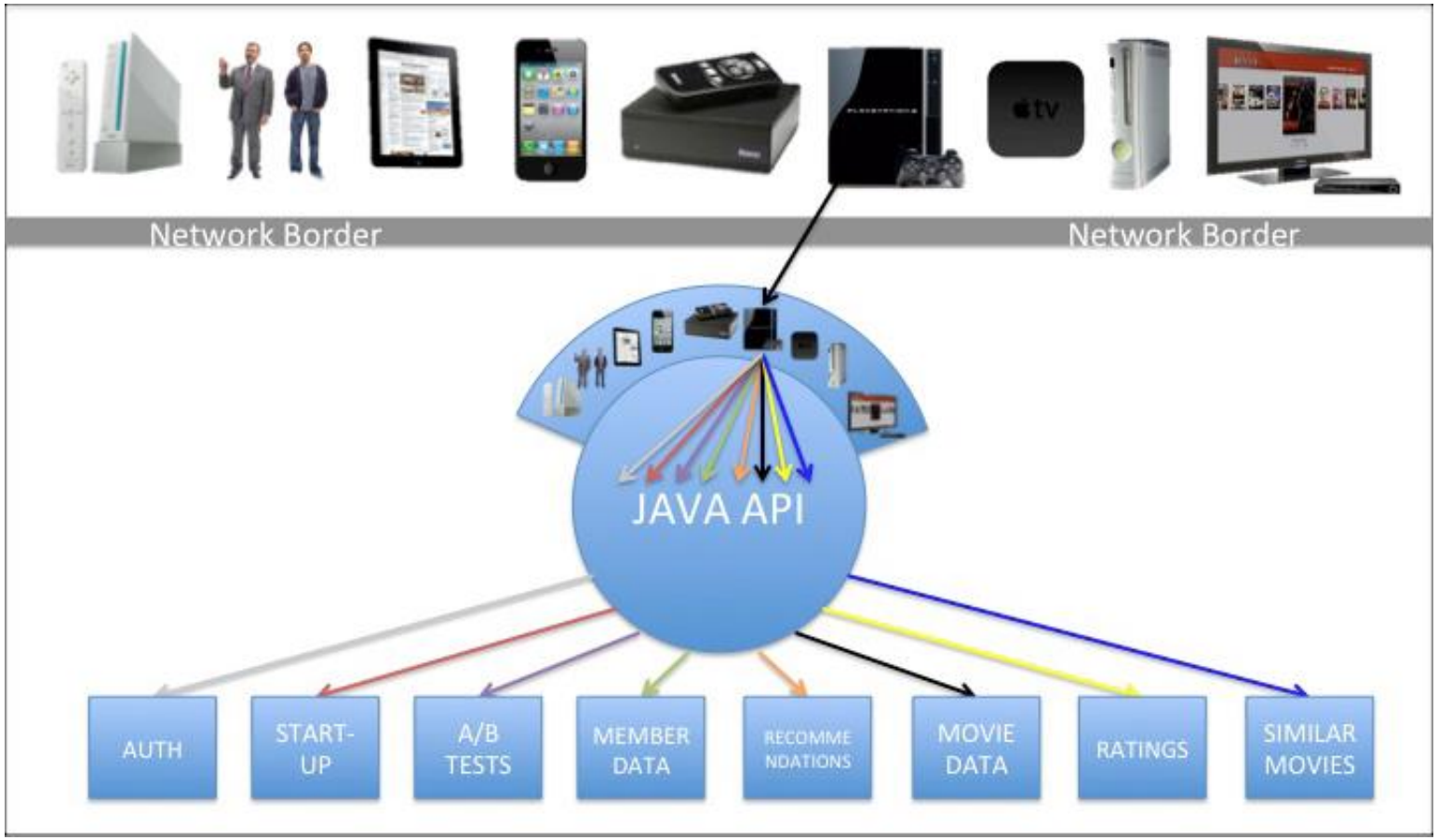
- Quand un client envoie une demande qui nécessite l'invocation de plusieurs microservices, le Gateway se charge de la composition de la demande et de l'agrégation des résultats,
- Exemple: si un client demande l'url « /produits/produit1 » pour afficher le détail du produit n°1 tout en affichant les différents avis reliés au produit demandé ainsi que les recommandations, le Gateway se charge de décomposer la requête et invoquer les microservices correspondants.

Type de Gateway

API Gateway:

- Au lieu de fournir une API unique, le API Gateway fournit une API différente pour chaque client,
- Exemple: API Gateway de Netflix
  - Le service de streaming Netflix est disponible sur des centaines de types d'appareils différents, y compris les téléviseurs, les décodeurs, les smartphones, les systèmes de jeux, les tablettes, etc.
  - Cette API fournit une API adaptée à chaque appareil en exécutant un code d'adaptateur spécifique à l'appareil pour gérer des milliards de demandes par jour,
- Inconvénient:
  - Haute disponibilité
  - Goulot d’étranglement

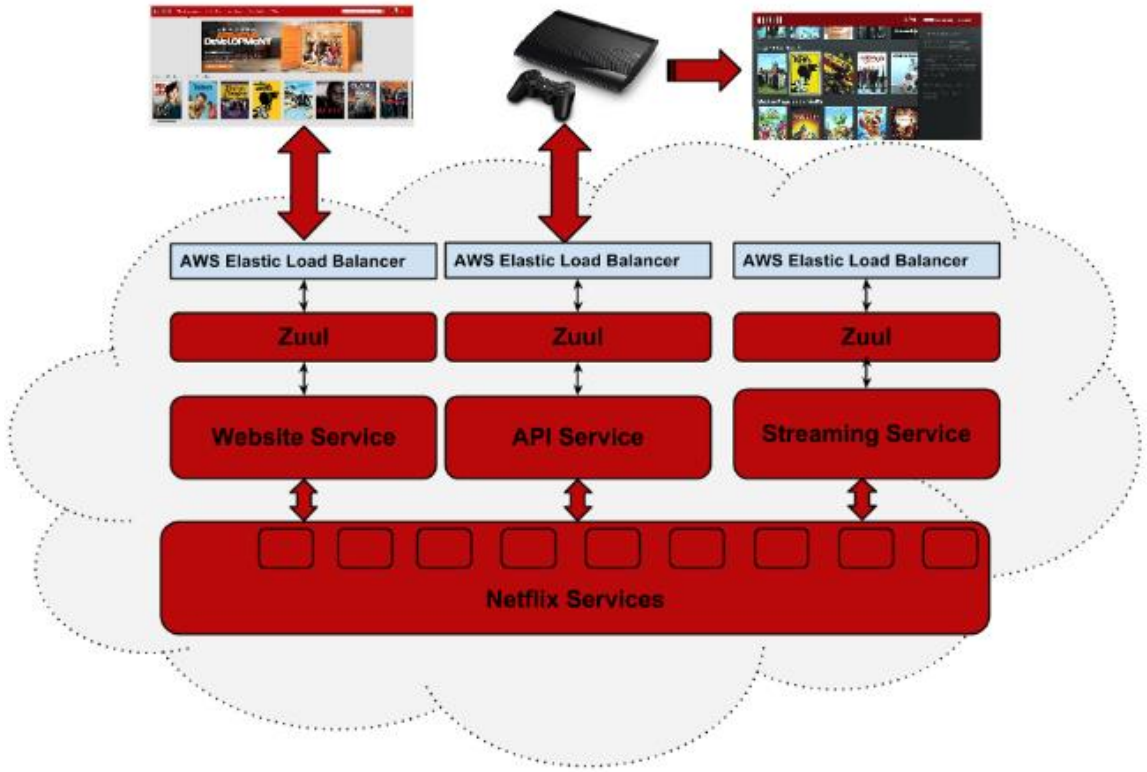
Exemple: Netflix API Gateway



Type de Gateway

BackEnd for FrontEnd Gateway:

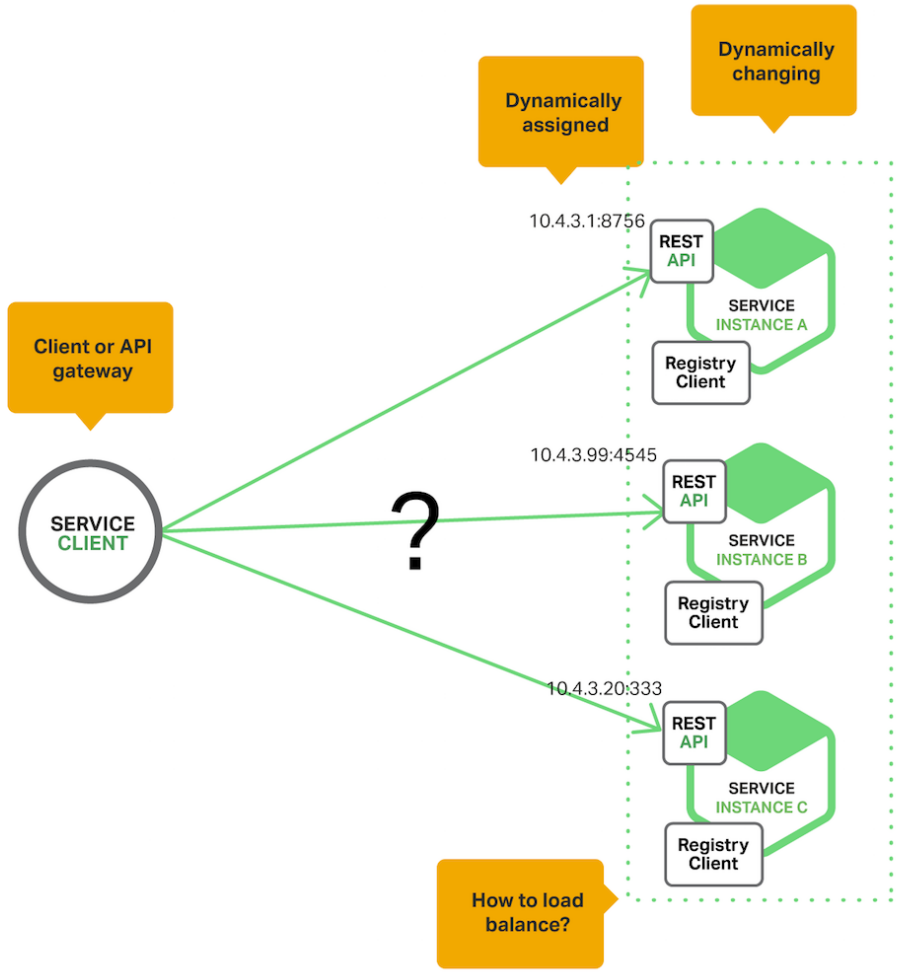
- Définir une passerelle API distincte pour chaque type de client,
- Exemple: Netflix Gatway API,
  - Netflix fournit plusieurs API, chacune gère les demandes d'un client spécifique



Composants d’une architecture microservices

Problématique 2:

- Pour faire une demande, le code client doit connaître l'emplacement réseau (adresse IP et port) d'une instance de service,
- Les instances de service ont des emplacements réseau attribués dynamiquement, au contraire des applications standards qui utilisent des adresses statiques,
- Les instances de service change dynamiquement en raison de la mise à l'échelle automatique et de mise à niveau,



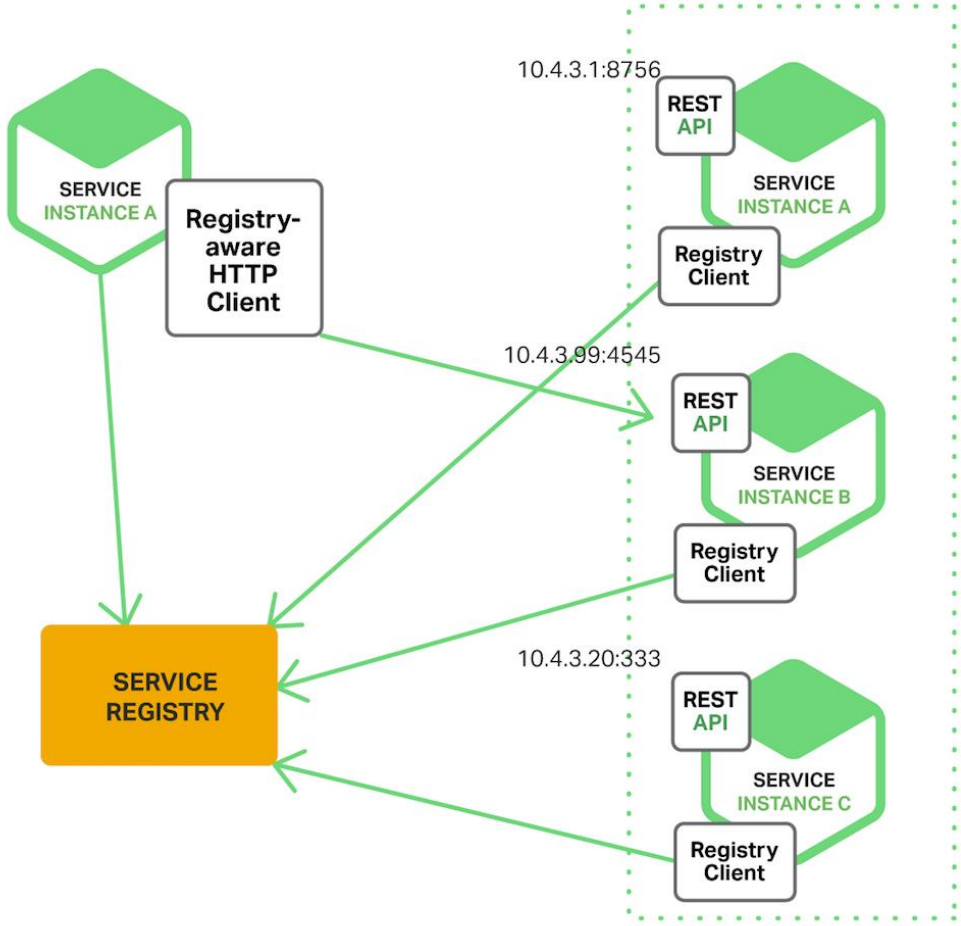
Composants d’une architecture microservices

Solution : utiliser un service de découverte (Discovery Service/Registry Service)

- Lors d’une demande d’un service, le client obtient l’emplacement d’une instance en interrogeant un « Service Registry » qui connait les adresses et les ports de toutes les instances,
- Les instances doivent s’enregistrer au démarrage, au près de ce registre, et supprimé à la fin

Modèles de Service Discovery

- Service Discovery côté client:
  - Le client est responsable de déterminer les emplacements réseau des instances des services disponibles
  - Le client interroge un registre de services,
  - Le client utilise ensuite un algorithme d'équilibrage de charge pour sélectionner l'une des instances de service disponibles et émet une demande
- **Exemple**: Netflix Eureka: fournit une API REST pour gérer l'enregistrement des instances de service et pour interroger les instances disponibles
- **Inconvénient**: couplage entre le client et le service de registre



Modèles de Service Discovery

- Service Discovery côté serveur:
  - Le client fait une demande à un service via un équilibreur de charge
  - L'équilibreur de charge interroge le registre de services et achemine chaque demande vers une instance de service disponible.
- **Exemple**: AWS Elastic: utilisé pour équilibrer la charge du trafic externe provenant d'Internet
- **Inconvénient**: disponibilité du composant