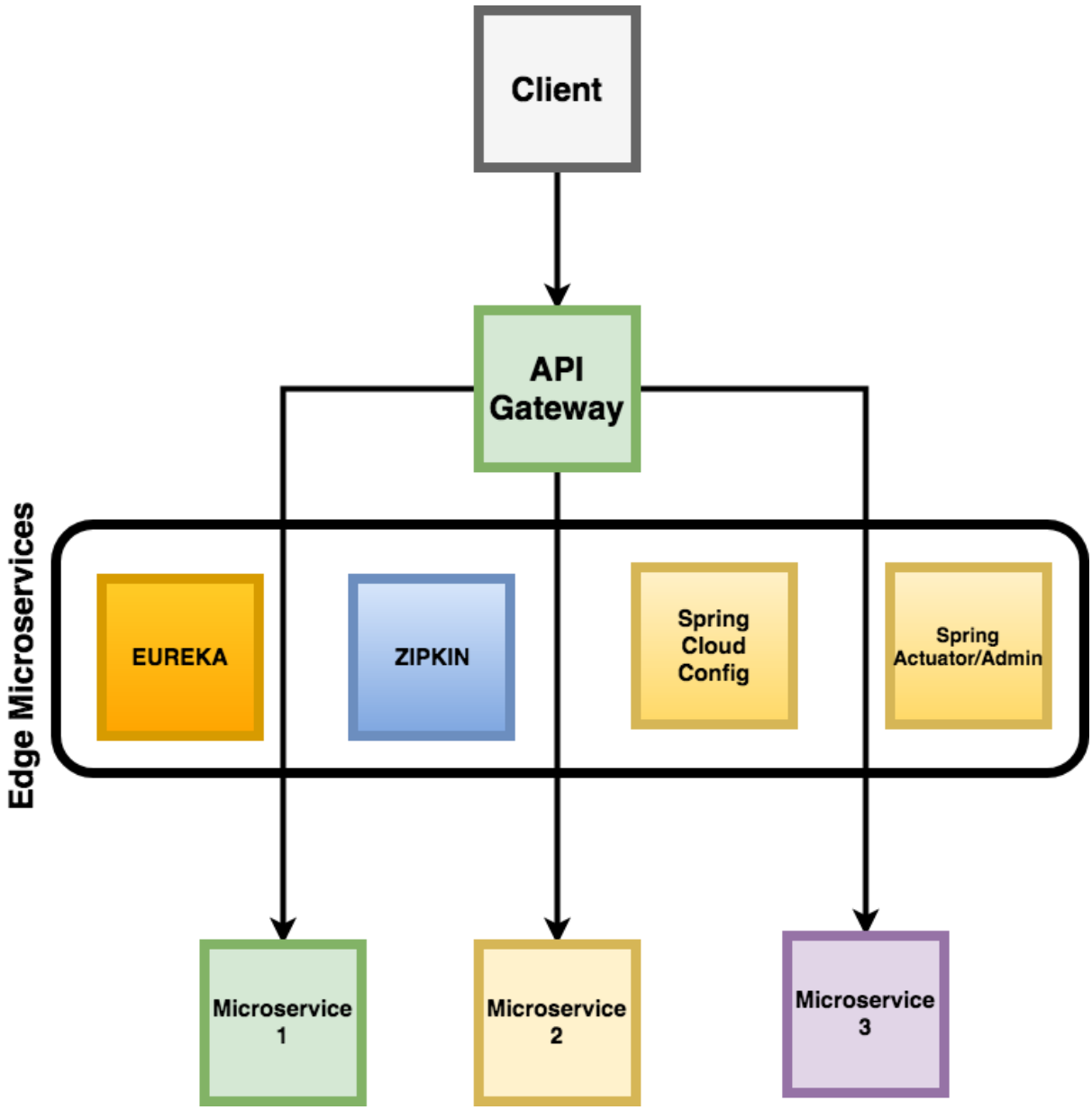
Atelier 3
Découvrir les Edge Microservices

Les **Edge Microservices** sont des Microservices spécialisés dans l'orchestration des Microservices centraux responsables de la logique de l'application.

Les Edge Microservices les plus populaires sont ceux publiés et utilisés par Netflix pour sa plateforme. Ils sont en grande partie regroupés sous Spring Cloud et disposent de fonctionnalités pour fonctionner nativement ensemble.



Ces Edge Microservices répondent chacun à un problème particulier. En voici quelques-uns auxquels nous allons nous attaquer dans les prochains chapitres.

La configuration

Dans une application grandeur nature, les Microservices et leurs différentes instances reposent énormément sur des fichiers de configuration tels que *application.properties.*

Imaginez que vous souhaitez changer un paramètre concernant une application en production. Il faut alors arrêter toutes les instances du Microservice, puis modifier et redéployer, en bloquant, de fait, l'application.

Comment éviter de bloquer l'application à chaque mise à jour de la configuration ?

La solution est **Spring Cloud Config.** Il va permettre de centraliser tous les fichiers de configuration dans un dépôt GIT et se positionner comme serveur de fichier de configuration.

Ainsi, quand vous mettez à jour un fichier de configuration dans votre dépôt GIT, *Spring Cloud Config* se met à servir cette nouvelle version, obligeant le Microservice à le prendre en compte à la volée.

La découvrabilité

En cas de grande charge sur notre application, nous souhaitons ajouter plusieurs instances d'un Microservice.

Comment garder la trace des URL des différentes instances disponibles, ainsi que leurs états ?

Pour répondre à ce problème, **Eureka** se propose en **naming server.** Grâce à une simple annotation, toute nouvelle instance de votre Microservice va être enregistrée auprès d'Eureka.

Votre client n'a plus qu'à consulter ce registre pour trouver les instances disponibles d'un Microservice donné.

Eureka s'occupe également de vérifier régulièrement si chaque instance enregistrée est toujours disponible afin de mettre à jour son registre en éliminant celles qui n'existent plus.

L'équilibrage de la charge (Load Balancing)

Nous souhaitons que nos Microservices soient les plus **découplés** et **autonomes** possible. Quand nous avons plusieurs instances d'un Microservice, il faut équilibrer la charge entre celles-ci.

On ne peut pas utiliser un équilibreur de charge central classique, car celui-ci limiterait la résilience de notre application en créant un point de centralisation.

Comment permettre à chaque Microservice de faire appel à un autre directement, sans être dépendant d'un Load Balancer central ?

La réponse est **Ribbon :** ce **load Balancer** côté client est capable d'indiquer directement, depuis le Microservice, quelle instance du Microservice distant appeler.

API Gateway

Imaginez que nous développons notre application e-commerce dans le cadre d'une véritable application professionnelle. L'affichage d'une fiche produit reposera alors sur des dizaines de Microservices. Il y aura par exemple des Microservices pour : les images, le pricing, les recommandations, les textes et caractéristiques des produits, un comparateur, la livraison, etc.

Quand notre client voudra afficher cette fiche produit, il devra faire appel à tous ces Microservices. Cela implique de les identifier, de repérer leurs instances, de s'authentifier auprès de chacun d'entre eux pour accéder aux ressources, de transformer le résultat reçu pour qu'il soit adapté au type d'appareil, etc.

Avec tout cela, vous imaginez que le client deviendra méchamment complexe, d'autant plus qu'il ne s'occupe pas uniquement des fiches produits !

Comment résoudre cette complexité ?

La solution est d'installer un point d'entrée unique vers les Microservices. C'est ce qu'on appelle une **API Gateway.**

L'API Gateway que nous allons utiliser s'appelle **ZUUL.** Elle va nous donner énormément d'avantages, notamment en ce qui concerne la sécurité : plus besoin de sécuriser chaque Microservice, il suffit d'imposer les règles d'authentification et de sécurité à ZUUL.

Traçage des requêtes

Reprenons l'exemple de la fiche produit : quand vous demandez celle-ci, votre requête doit traverser plusieurs Microservices avant d'aboutir. Pensez, par exemple, à notre application Mcommerce, dans laquelle la requête de demande de paiement envoyée par le client déclenche dans le Microservice-produits une autre requête vers le Microservice-commandes afin de mettre à jour le statut de la commande.

Si une erreur vous est retournée, il est difficile de savoir si elle s'est produite au niveau du Microservice-produits ou au niveau du Microservice-commandes.

Imaginez maintenant le problème quand votre requête traverse 25 Microservices avant d'aboutir !

Comment identifier le Microservice posant problème ?

Zipkin va vous permettre de **suivre vos requêtes de Microservice en Microservice** et de consulter les différentes réponses et interactions afin de cibler directement le problème.