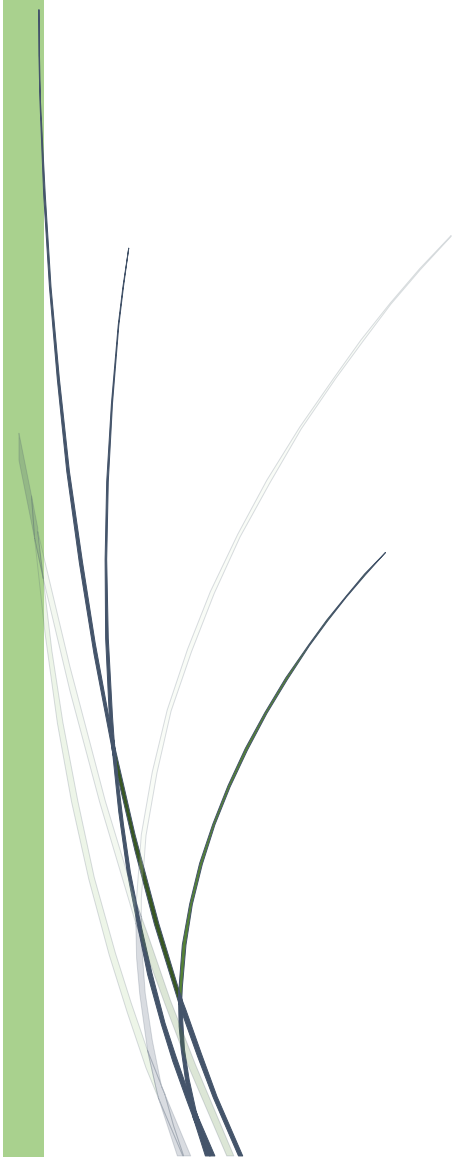
01/02/2025

API Gateway : Optimisation et Sécurisation des Microservices

Une passerelle essentielle pour les applications cloud-native



MAHDI KELLOUCH
ISMO TETOUAN

Table des matières

API Gateway

1. Introduction aux Applications Cloud Native

Une application cloud native est conçue pour tirer parti des architectures de cloud computing. Elle repose sur des microservices, le déploiement en conteneurs et une gestion dynamique via des API Gateway pour assurer performance, scalabilité et sécurité.

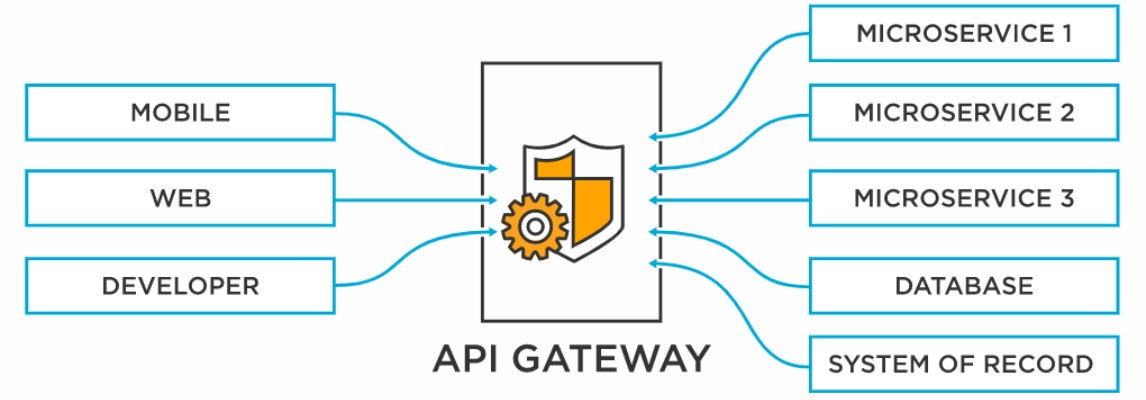
2. C'est quoi une API Gateway ?

Une **API Gateway** est un point d'entrée unique pour les clients souhaitant accéder à différentes API d'un système basé sur des microservices. Elle facilite la gestion des requêtes, la sécurité et l'optimisation des performances.

Fonctionnalités clés :

- Routage des requêtes vers les microservices appropriés
- Gestion des politiques d'authentification et d'autorisation
- Mise en cache des réponses pour optimiser la performance
- Surveillance et journalisation des requêtes
- Équilibrage de charge entre les microservices

Schéma d'une API Gateway :



3. Les avantages de l'API Gateway

Sécurité accrue

- Protège les microservices en servant de couche d'abstraction
- Permet l'authentification et l'autorisation centralisées (JWT, OAuth2, API Key, etc.)

Simplification du code

- Centralisation des règles métier dans l'API Gateway
- Diminution du code métier dans les microservices

Optimisation des performances

- Réduction des allers-retours entre le client et les services
- Mise en cache pour améliorer la rapidité d'accès aux données

Meilleure gestion de la charge

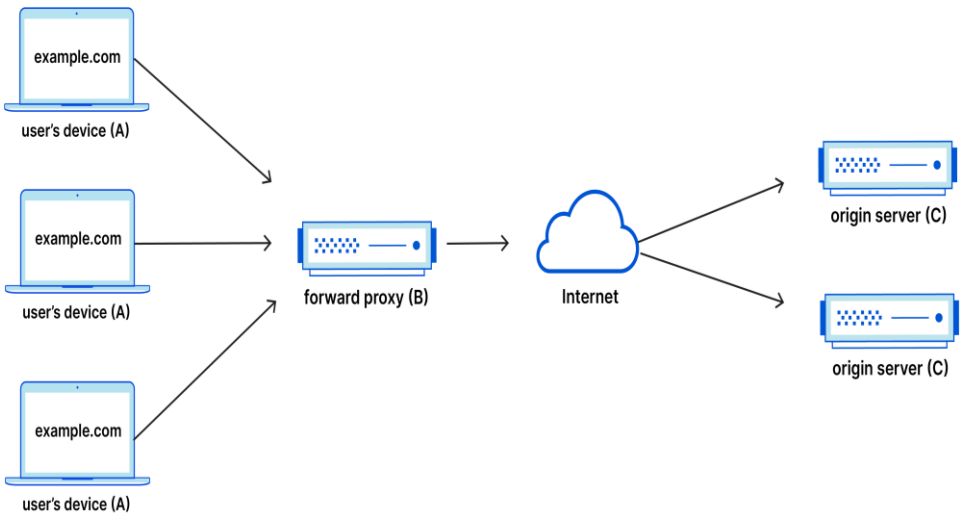
- Répartition des requêtes entre plusieurs instances
- Protection contre les attaques DDoS en limitant les requêtes

Collecte des métriques

- Suivi du trafic et des performances
- Analyse des logs et diagnostics facilités

4. C'est quoi un proxy inverse ?

Un **proxy inverse** est un serveur intermédiaire qui reçoit les requêtes des clients et les transmet aux serveurs web appropriés.



Différences entre Proxy et Proxy Inverse :

Type	Rôle
Proxy	Transmet les requêtes du client à Internet
Proxy inverse	Transmet les requêtes des clients aux serveurs backend

Exemple d'utilisation :

- Protection des services backend
- Mise en cache des réponses
- Gestion de la compression et de l'optimisation des requêtes

5. Introduction à NGINX

NGINX est un serveur web open source, performant et léger. Il est souvent utilisé comme **reverse proxy**, **load balancer** et **serveur d'applications statiques**.

Avantages de NGINX :

- Gestion efficace des connexions simultanées
- Optimisation du temps de réponse grâce au caching
- Support du protocole HTTP/2 et HTTPS
- Faible consommation de ressources

6. Installation de NGINX avec Docker

Nous allons utiliser l'image Docker officielle de NGINX pour le déploiement.

Installation et lancement :

```
docker pull nginx
```

```
docker run --name nginx-container -d -p 80:80 nginx
```

Vérification du fonctionnement :

Dans un navigateur, accédez à http://localhost.

7. Fichier de Configuration de NGINX

Le fichier principal est **/etc/nginx/nginx.conf**. Il contient la configuration du serveur web et du reverse proxy.

Exemple d'une configuration basique :

```
server {
    listen 80;
    server_name example.com;
    location / {
        proxy_pass http://backend_service;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Commandes utiles :

```
nginx -t            # Vérifier la syntaxe du fichier de configuration
nginx -s reload     # Recharger la configuration
```

8. Exercices d'application

Exercice 1 : Mise en place d'un Reverse Proxy avec NGINX

- Installez et lancez un serveur NGINX avec Docker
- Configurez-le pour rediriger les requêtes vers une API locale

Exercice 2 : Mise en cache des réponses API

- Configurez NGINX pour stocker les réponses en cache
- Vérifiez si le temps de réponse diminue avec le caching

Exercice 3 : Gestion des erreurs et des redirections

- Implémentez une redirection automatique vers une page d'erreur en cas d'indisponibilité du service backend

9. Conclusion

L'API Gateway est un composant clé des architectures modernes basées sur les microservices. Son rôle principal est de centraliser la gestion des API, d'améliorer la sécurité et de garantir une performance optimale. En utilisant NGINX comme reverse proxy, nous pouvons efficacement gérer la répartition du trafic et sécuriser les applications distribuées.