**Étape 1 : Comprendre et développer un simple microservice**

**Objectif** : Créer un premier **microservice basique** en utilisant **Spring Boot** (Java) ou **Node.js (Express.js)**.

✅ **Actions :**

1. Installez **Java + Spring Boot** ou **Node.js + Express**.
2. Créez un microservice simple qui expose une API REST :
   * Exemple : Un service de gestion des étudiants (etudiants-service).
   * Endpoint : GET /etudiants → Retourne une liste d’étudiants en JSON.
3. Testez avec **Postman** ou **cURL**.



**Étape 2 : Ajouter un deuxième microservice et communiquer entre eux**

**Objectif** : Créer un second microservice et le connecter au premier via une API REST.

✅ **Actions :**

1. Créez un **service des cours** (cours-service).
2. Ajoutez une API GET /cours qui retourne une liste de cours.
3. **Faites communiquer les deux services** :
   * Le etudiants-service appelle le cours-service pour afficher la liste des cours d’un étudiant.
4. Testez via Postman ou un navigateur.

🔹 **Exemple : etudiants-service appelle cours-service (en Spring Boot)**



**Étape 3 : Ajouter une API Gateway**

**Objectif** : Regrouper les services sous une seule URL pour simplifier l'accès aux clients.

✅ **Actions :**

1. Installez **Spring Cloud Gateway** ou **Traefik**.
2. Configurez l’API Gateway pour rediriger /etudiants et /cours vers les bons services.

🔹 **Exemple avec Spring Cloud Gateway (Java)**



**Étape 4 : Ajouter un Service Discovery (Eureka)**

**Objectif** : Permettre aux services de se découvrir automatiquement **sans utiliser d’IP statiques**.

✅ **Actions :**

1. Installez **Spring Cloud Netflix Eureka**.
2. Configurez **Eureka Server** (registre des services).
3. Faites en sorte que chaque service s’enregistre dynamiquement.

🔹 **Exemple de configuration pour enregistrer un service dans Eureka (application.yml)**

**Étape 5 : Ajouter un Circuit Breaker**

**Objectif** : Empêcher les erreurs d’un microservice de planter toute l’application.

✅ **Actions :**

1. Installez **Resilience4J** pour gérer les pannes.
2. Implémentez un mécanisme de secours si un service est indisponible.

🔹 **Exemple avec Resilience4J (EtudiantController.java)**