07/10/2023
Version 2

SPRING CLOUD

JÉRÉMY PERROUAULT

DISCOVERY

DISCOVERY / REGISTRY SERVICE

DISCOVERY

- Pour faire communiquer les 2 services précédents
  - Utilisation des URL, codées en dur ou dans un fichier de configuration …
  - Que faire en cas de changement de localisation ?
  - Que faire en cas d'environnements multiples (localhost, docker, plusieurs machines, etc.) ?
  - Comment faire du load-balancing
    - 2, 3 ou plus instances du même service accessibles depuis différentes localisations

DISCOVERY

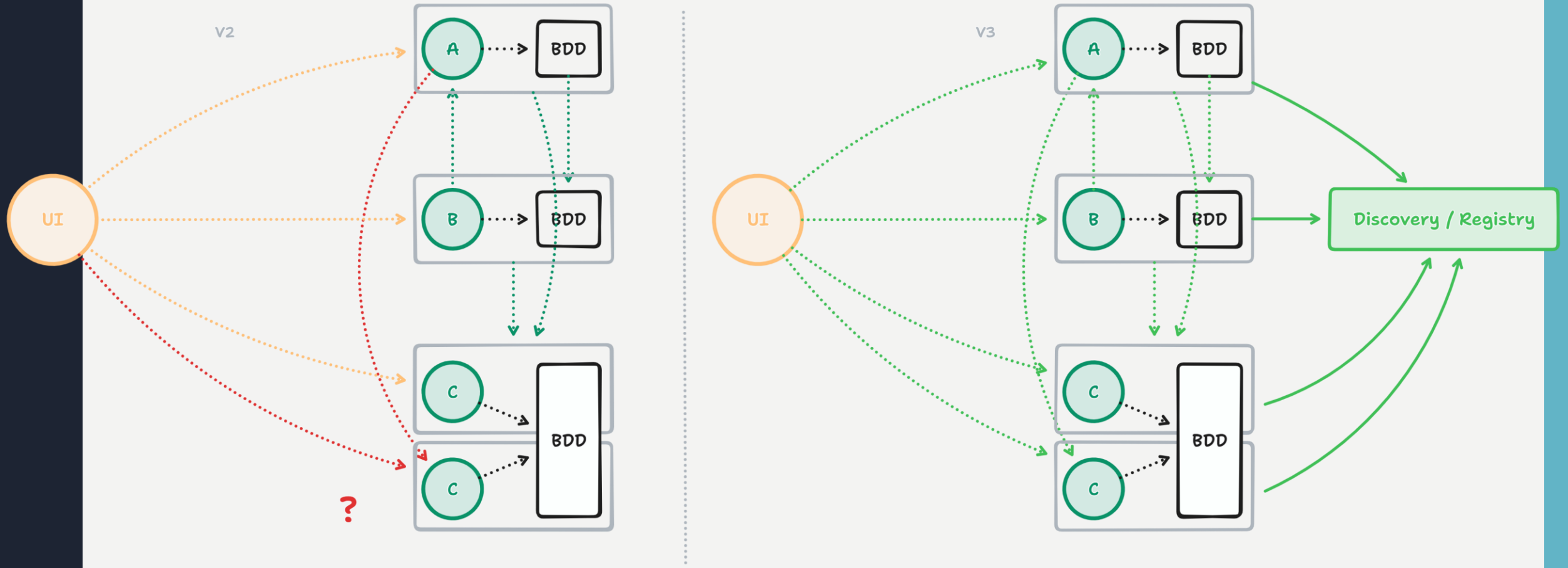
- Utilisation du Pattern **Service Discovery**
  - L'idée, c'est de forcer chaque instance de service à se déclarer opérationnelle sur un serveur dédié
    - Avec un nom de service
  - Chaque service pourra alors demander au serveur dédié
    - L'adresse ou les adresses disponibles (« load-balancing »)
    - Le protocole utilisé (HTTP ou HTTPS)
    - Le port utilisé
    - Etc.

DISCOVERY

- Utilisation d'une technologie existante …
  - **Eureka** (Netflix)
  - **Consul**
  - …

- Création d'un serveur de **Discovery** / **Registry**
- Chaque service est un client **Discovery**

DISCOVERY


V2
UI
A
BDD
B
BDD
C
C
BDD
?
V3
UI
A
BDD
B
BDD
C
C
BDD
Discovery / Registry

DISCOVERY

- Mettre en place un serveur **Eureka**
  - Avec **SPRING BOOT**

- Configurer les 2 services pour qu'ils deviennent des clients **Eureka**

SPRING BOOT

EXEMPLE D'IMPLÉMENTATIONS

SPRING BOOT – SERVEUR

- Ajouter la dépendance *spring-cloud-starter-netflix-eureka-server*

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
```

SPRING BOOT – SERVEUR

- Ajouter `@EnableEurekaServer` à l'application **SPRING BOOT**
- Configurer le server Eureka dans *application.properties*
  - Par défaut, le serveur **Eureka** s'enregistre lui-même en tant que client, on désactive ce comportement

```
server.port = 8761
spring.application.name = eureka-server

eureka.instance.hostname = localhost

eureka.client.register-with-eureka = false
eureka.client.fetch-registry = false
eureka.client.service-url.defaultZone = http://${eureka.instance.hostname}:${server.port}/eureka/
```

SPRING BOOT – CLIENT

- Ajouter la dépendance *spring-cloud-starter-netflix-eureka-client*

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

- Depuis **SPRING BOOT 3**, plus besoin de `@EnableEurekaClient`
- Configurer le client **Eureka** dans *application.properties*

```
spring.application.name = nom-service
eureka.client.service-url.defaultZone = http://localhost:8761/eureka/
```

SPRING BOOT – CLIENT

- Utiliser `RestTemplate` et l'injection de dépendance

```
@Bean
@LoadBalanced
public RestTemplate restTemplate() {
  return new RestTemplate();
}
```

```
restTemplate
  .getForObject("lb://nom-service/api/...", Type.class);
```

SPRING BOOT – CLIENT

- Utiliser **OpenFeign**
  - Avec la dépendance *spring-cloud-starter-openfeign* de **SPRING CLOUD**
  - L'annotation `@EnableFeignClients`
  - La création d'une interface dédiée

```
@FeignClient("nom-service")
public interface ServiceClient {
    @GetMapping("/api/endpoint/by-produit-id/{produitId}")
    public List<Response> findAllByProduitId(@PathVariable String produitId);
}
```

  - L'injection de la dépendance `ServiceClient`

```
serviceClient.findAllByProduitId("id-du-produit");
```