MICROSERVICES

JÉRÉMY PERROUAULT

DISCOVERY

DISCOVERY / REGISTRY SERVICE

DISCOVERY

- Pour faire communiquer les 2 services précédents
  - Utilisation des URL, codées en dur …
  - Que faire en cas de changement de localisation ?
  - Que faire en cas d'environnements multiples (localhost, docker, etc.) ?
  - Comment faire du load-balancing (2, 3 ou plus services identiques accessibles à différents endroits)

DISCOVERY

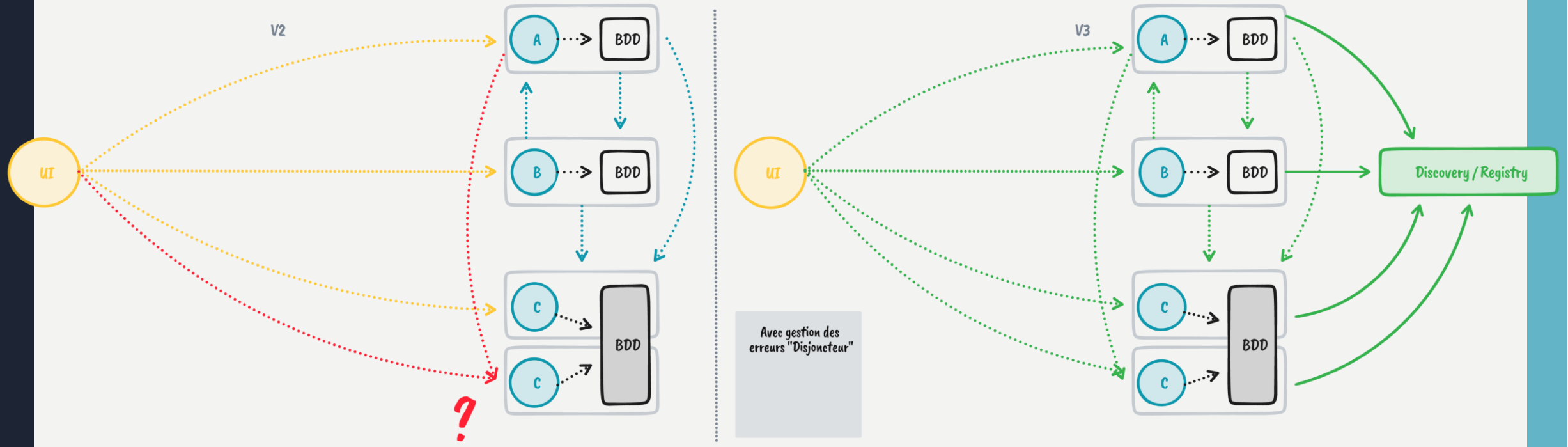
- Utilisation du Pattern « Service Discovery »
  - L'idée, c'est de forcer chaque instance de service à se déclarer opérationnelle sur un serveur dédié
    - Avec un nom de service
  - Chaque service pourra alors demander au serveur dédié
    - L'adresse ou les adresses disponibles (« load-balancing »)
    - Le protocole (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
Avec gestion des erreurs "Disjoncteur"

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>
```

- 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 = 9000

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>
```

- Ajouter **@EnableEurekaClient** à l'application Spring Boot
- Configurer le client Eureka dans *application.properties*

```
spring.application.name = nom-service

eureka.client.register-with-eureka = true
eureka.client.fetch-registry = true
eureka.client.service-url.defaultZone = http://localhost:9000/eureka/
```

SPRING BOOT – CLIENT

- Utiliser RestTemplate

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

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

.NET6

EXEMPLE D'IMPLÉMENTATIONS

.NET6

- Ajouter le package *Steeltoe.Discovery.Eureka*

```
<PackageReference Include="Steeltoe.Discovery.Eureka" Version="3.2.1" />
```

- Ajouter le service dans *Program.cs*

```
builder.Services.AddServiceDiscovery(options => options.UseEureka());
```

- Configurer le client Eureka dans *appsettings.json*

.NET6

```
"Spring": {
  "Application": {
    "Name": "nom-service"
  }
},

"eureka": {
  "client": {
    "serviceUrl": "http://localhost:9000/eureka/",
    "shouldRegisterWithEureka": true,
    "shouldFetchRegistry": true
  },

"instance": {
  "ipAddress": "127.0.0.1",
  "preferIpAddress": true,
  "nonSecurePortEnabled": true,
  "port": 5153,
  "securePort": 5153
  }
},
```

.NET6

- Utiliser HttpClientFactory & HttpClient

```
builder.Services.AddHttpClient("nom-service", client => {
    client.BaseAddress = new Uri("lb://nom-service/");
}).AddRandomLoadBalancer();
```