

**BCEPHAL-ARCH-001**

**B-CEPHAL**



**Architecture micro-services**

Moriset & co

www.moriset.com

Table des matières

[Table des tableaux iii](#_Toc45234574)

[Table des figures iii](#_Toc45234575)

[1. introduction 1](#_Toc45234576)

[1.1. But du document 1](#_Toc45234577)

[1.2. Cible 1](#_Toc45234578)

[1.3. Portée du projet 1](#_Toc45234579)

[1.4. Acronymes et abréviations 1](#_Toc45234580)

[1.5. Documents de référence 1](#_Toc45234581)

[2. Généralités 1](#_Toc45234582)

[2.1. Définition 1](#_Toc45234583)

[2.2. Quelques concepts 1](#_Toc45234584)

[2.3. Les avantages et bénéfices 2](#_Toc45234585)

[2.4. Les Problématiques 2](#_Toc45234586)

[2.5. Les Solutions 2](#_Toc45234587)

[2.6. Spring Cloud 2](#_Toc45234588)

[3. Implémentation 3](#_Toc45234589)

[3.1. La distribution : Srping Cloud Netflix Eureka 3](#_Toc45234590)

[3.2. La communication : Srping Cloud Netflix Feign 3](#_Toc45234591)

[3.3. Distribution des charges : Srping Cloud Netflix Ribbon 3](#_Toc45234592)

[3.4. Gestion des erreur : Srping Cloud Netflix Hystrix 3](#_Toc45234593)

[3.5. Exposition de l’API : Srping Cloud Netflix Zuul 3](#_Toc45234594)

[3.6. Configuration : Srping Admin 3](#_Toc45234595)

[4. Organisation 3](#_Toc45234596)

[4.1. Indépendance 3](#_Toc45234597)

[4.2. Attention au DRY 3](#_Toc45234598)

[4.3. Architecture 4](#_Toc45234599)

[4.4. Exploitation 4](#_Toc45234600)

[5. Schemas 4](#_Toc45234601)

[6. Catalogue 5](#_Toc45234602)

[7. Description par décomposition 5](#_Toc45234603)

[7.1. Décomposition par modules 6](#_Toc45234604)

[7.1.1. Interfaces du systèmes 6](#_Toc45234605)

[7.1.2. Interfaces utilisateurs 6](#_Toc45234606)

[7.1.3. Interfaces matérielles 6](#_Toc45234607)

[7.1.4. Interfaces logicielles 6](#_Toc45234608)

# Table des tableaux

[Tableau 1: Acronymes et abréviations 1](#_Toc45234097)

[Tableau 2: Document de référence 1](#_Toc45234098)

# Table des figures

Aucune entrée de table d'illustration n'a été trouvée.

# introduction

## But du document

Ce document décrit l'architecture micro-services mise en œuvre dans le cadres de l’application B-CEPHAL.

## Cible

Ce document s’adresse à l’équipe de développement de B-CEPHAL.

## Portée du projet

Cette architecture concerne la version 8 de B-CEPHAL. Elle architecture est conçue de façon à tirer avantage des micro-services.

## Acronymes et abréviations

|  |  |
| --- | --- |
| Acronyme/Abréviation | Définition |
|  |  |
|  |  |
|  |  |

Tableau 1: Acronymes et abréviations

## Documents de référence

|  |  |
| --- | --- |
| Référence | Titre |
|  |  |
|  |  |
|  |  |

Tableau 2: Document de référence

# Généralités

Cette section présente les concepts généraux d’une architecture micro-service.

## Définition

L’architecture micro-service est une approche de développement qui consiste à découper une application en petits services, tournant chacun dans un processus autonome et qui communique entre eux via un protocole de communication léger et faiblement typé.

## Quelques concepts

* **Le périmètre fonctionnel** : Le périmètre de chaque microservice doit être restreint et ne concerné qu’un domaine métier. La méthode « *Domain Driven Design* »[[1]](#footnote-2) peut s’avérée utile pour la définition et la délimitation du périmètre de chaque microservice.
* **L’indépendance des processus** : Chaque microservice doit tourner dans un processus indépendant. Ce qui permettra d’arrêter ou de redémarrer l’un sans impacter les autres.
* **Le couplage lâche** : Les microservices doivent êtres faiblement couplés entre eux pour qu’on puisse faire évoluer les différents microservices sans impacter les autres. Ceci passe par des protocoles de communication lâches (Messaging, API REST…).
* **Le déploiement** : Le déploiement doit être entièrement automatisé et si possible continu.

## Les avantages et bénéfices

* **La scalabilité** :
* **L’isolation** : Chaque microservice étant isolé, l’application va continuer à fonctionner (en mode dégradé) même si l’un des microservices n’est plus disponible.
* **L’indépendance** : On peut avoir plusieurs petits projets avec chacun leur cycle de vie. On va pouvoir releaser et déployer une portion de l’application sans avoir besoin de tout relivrer et retester. Cette indépendance permet également de choisir la solution de stockage, les technologies et les langages les mieux adaptés pour chaque microservice.
* **La productivité** : Les microservices étant plus petits, les équipes vont plus facilement monter en compétence sur un domaine métier précis. Le refactoring est également simplifier.

## Les Problématiques

* Système distribué : On peut se retrouver avec une multitude de services qui on chacun une multitude d’instances. Il va falloir être capable de cataloguer toutes ces instances (localisation, état).
* Communication : l’interconnexion de service et la répartition de charges.
* La tolérance aux pannes : Problème de timeout, défaillance en cascade.

## Les Solutions

* Airbnb : SmartStack
* HashiCorp : Consul
* **Netflix : Eureka, Feign,…**

## Spring Cloud

Framework mettant à disposition des outils et des librairies pour gérer les problématiques qui se posent dans le cloud (distribution, hébergement, …). Il permet aussi de faire une abstraction des outils de bas niveau utilisés en dessous. Spring-cloud-netflix

# Implémentation

## La distribution : Spring Cloud Netflix Eureka

Comment découvrir, localiser et cataloguer nos différents microservices.

Eureka est un service d’enregistrement près duquel chaque micro service à son démarrage va aller s’enregistrer en lui communiquant l’adresse de sa machine hôte et son port d’écoute. Chaque microservice communique également sont état de santé à Eureka à intervalle régulier. Chaque microservice peut s’adresser à Eureka pour avoir l’adresse sur laquelle joindre un autre microservice. Eureka possède un tableau de bord.

Spring-cloud-netflix

## La communication : Spring Cloud Netflix Feign

Feign est un client REST dynamique.

## Distribution des charges : Spring Cloud Netflix Ribbon

Repartie la charge entre les différentes instances d’un micro service. Il est présent dès qu’on intègre Feign.

## Gestion des erreur : Spring Cloud Netflix Hystrix

Coupe circuit, cascade d’erreur

## Exposition de l’API : Spring Cloud Netflix Zuul

Zuul est un proxy, vers les différents micro services. Par défaut, il expose chaque service sous une sous-URL et il s’occupe du routage. On peut y mettre des filtres (pour sécuriser une partie des services par exemple).

## CONFIGURATION: Spring Admin

Centralisation de la configuration des micro-services.

# Organisation

## Indépendance

* Un dépôt GIT pour chaque microservice
* Un build indépendant pour chaque microservice
* Un déploiement indépendant pour chaque microservice

## Attention au DRY

* Eviter de factoriser le code
* Privilégier la création de librairie
* Consumer-Driven Contracts (c’est le client qui définit ce qu’il veut au serveur)

## Architecture

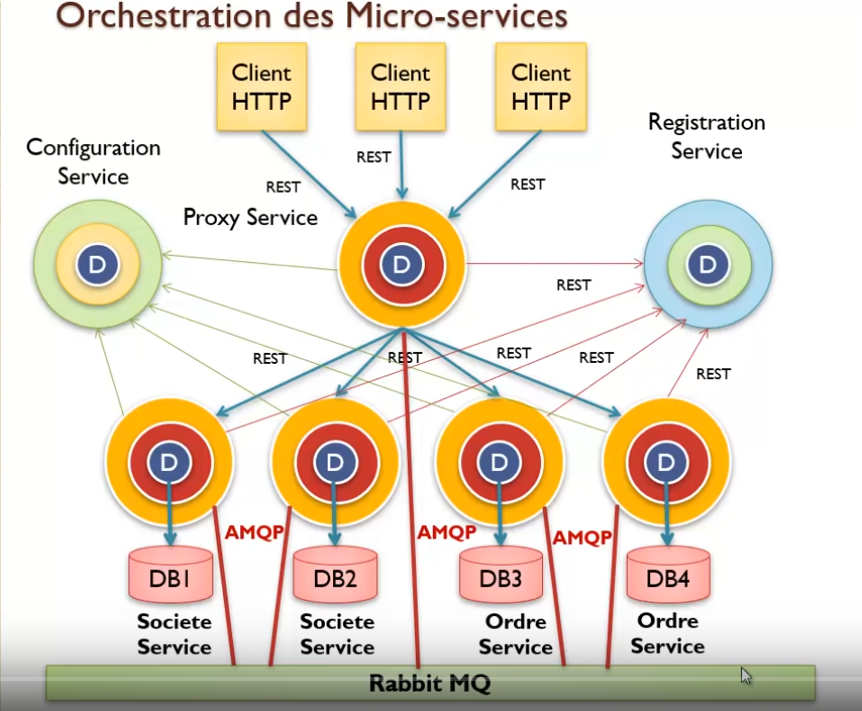
* Privilège le messaging (communication asynchrone)

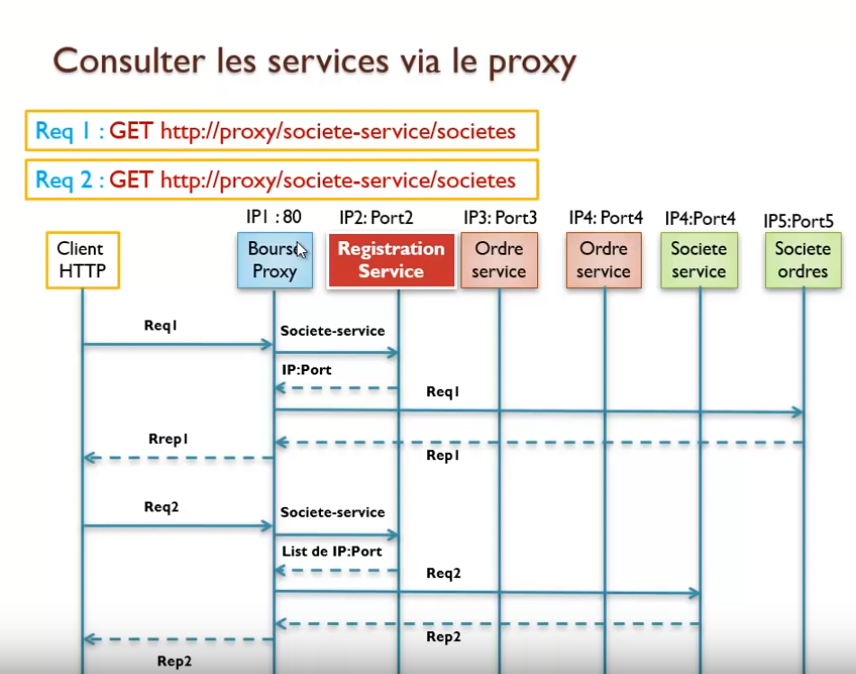
## Exploitation

* Conteneur / Docker : Dev ISO Prod
* Cloud
* Monitoring : Zipkin pour tracer les requêtes. ELK pour la centralisation des logs

# Schemas

https://www.youtube.com/watch?v=j8rnLUe3r4g&list=PLxr551TUsmAq2UEkfBidVmi21nP5KNk-K&index=2





# Catalogue

|  |  |  |
| --- | --- | --- |
| Project |  |  |
| B-CEPHAL Configuration Service | bcephal-configuration-service | http://saas1402fu.saas-secure.com/git/bcephal-configuration-service.git |
| B-CEPHAL Registration Service | bcephal-registration-service | http://saas1402fu.saas-secure.com/git/bcephal-registration-service.git |
| B-CEPHAL Proxy Service | bcephal-proxy-service | http://saas1402fu.saas-secure.com/git/bcephal-proxy-service.git |
| B-CEPHAL Security Service | bcephal-security-service | http://saas1402fu.saas-secure.com/git/bcephal-security-service.git |
| B-CEPHAL Project Service |  |  |
| B-CEPHAL Initiation Service |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Description par décomposition

L'architecture du programme B-CEPHAL est décrite à l'aide du modèle d'état, du modèle des classes et d'un diagramme de flux de donnée.

## Décomposition par modules

### Interfaces du système

### Interfaces utilisateurs

### Interfaces matérielles

### Interfaces logicielles

1. https://en.wikipedia.org/wiki/Domain-driven\_design [↑](#footnote-ref-2)