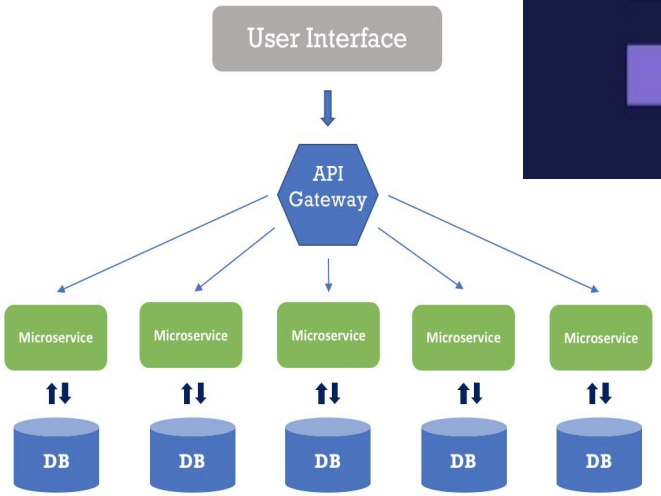
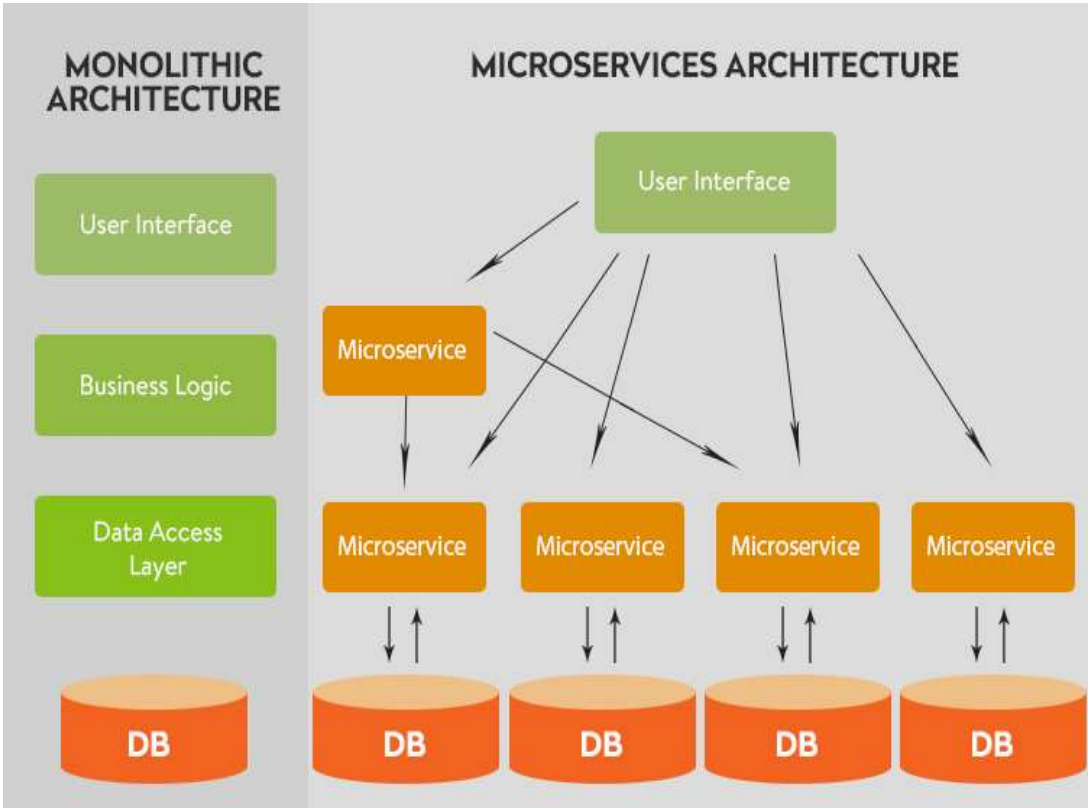
# FORMATION «Architecture» : Architecture des MicroServices







# SOMMAIRE

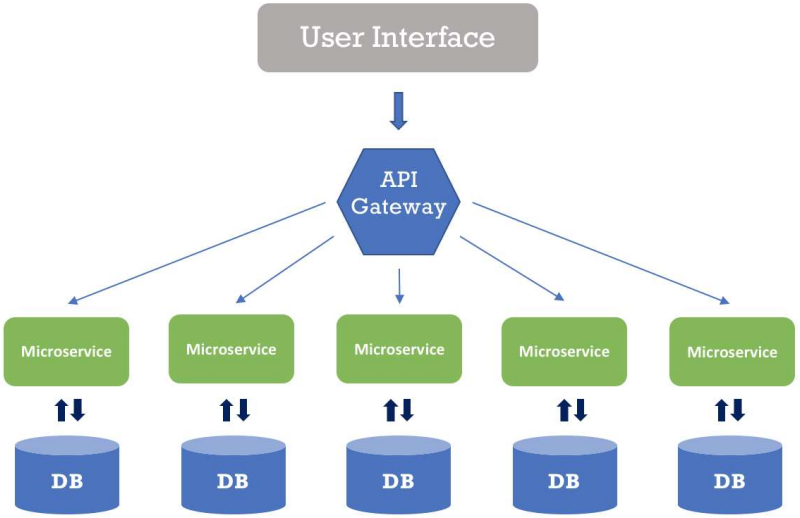
## Introduction

- 1/ Architecture : Principes
- 2/ JAVA JEE/Spring Boot
- 3/ API REST
- 4/ Microservices
- 5/ DevOps
- 6/ Travaux Pratiques
- 7/ Synthèse des bonnes pratiques

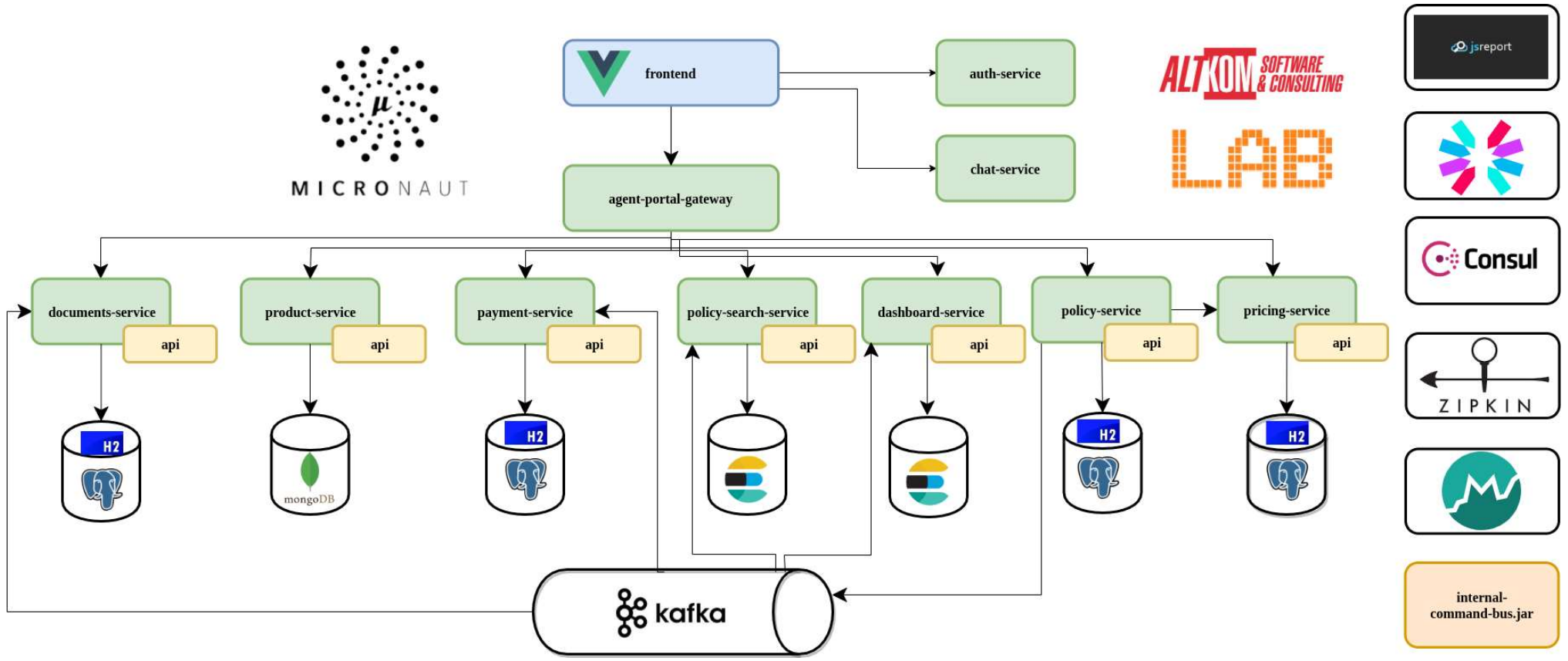
## Conclusion

# INTRODUCTION

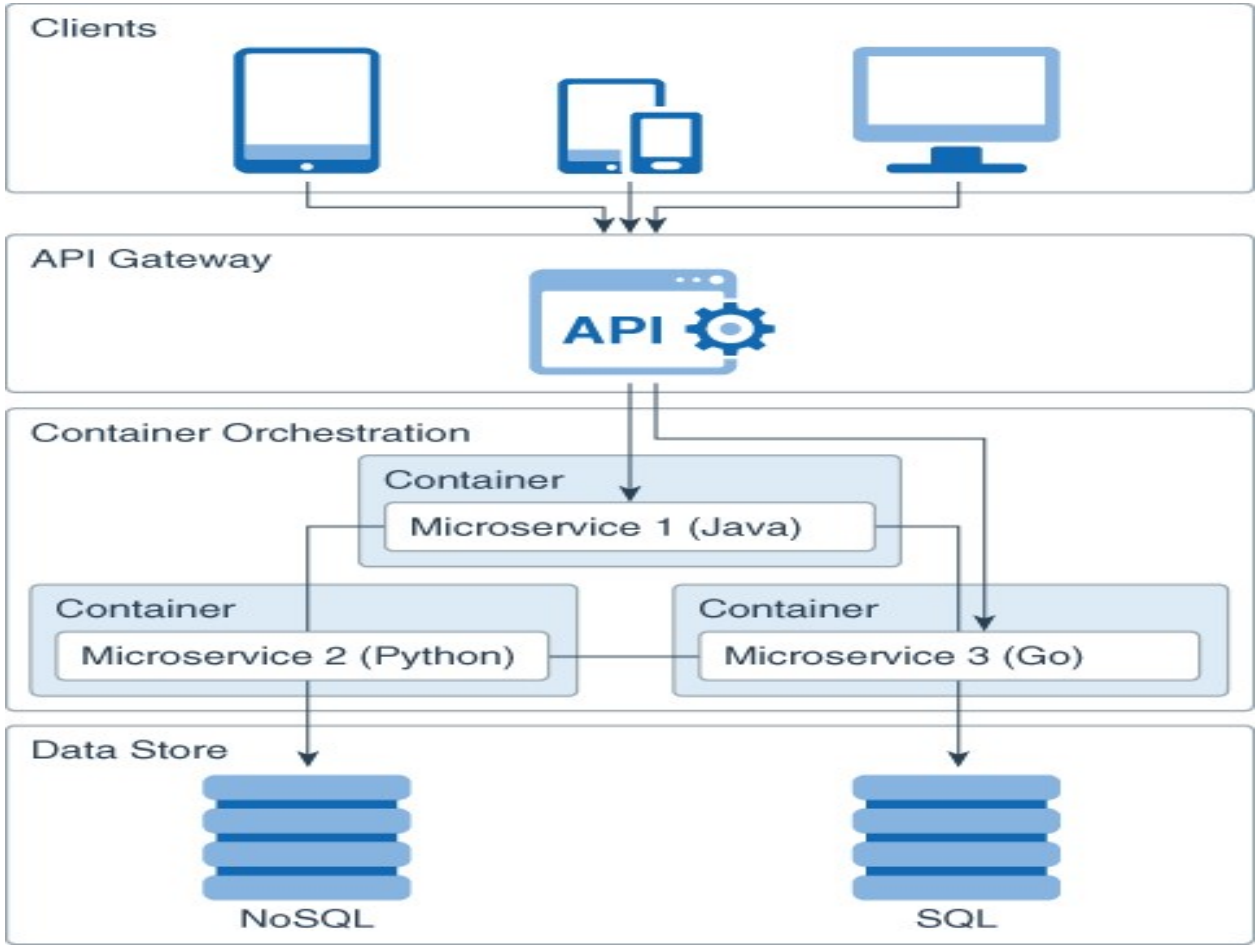
Dans l'architecture Microservices, **chaque service est conçu pour remplir et implémenter l'un des sous-domaines** du domaine de l'application. Par exemple, dans une application de commerce électronique, au lieu de créer des services basés sur les couches et les niveaux dans lesquels ils existent, nous créons les services sur les sous-domaines au sein de l'espace de domaine du commerce électronique comme le panier, le catalogue, la gestion des acheteurs et le Traitement des commandes, etc.




# INTRODUCTION

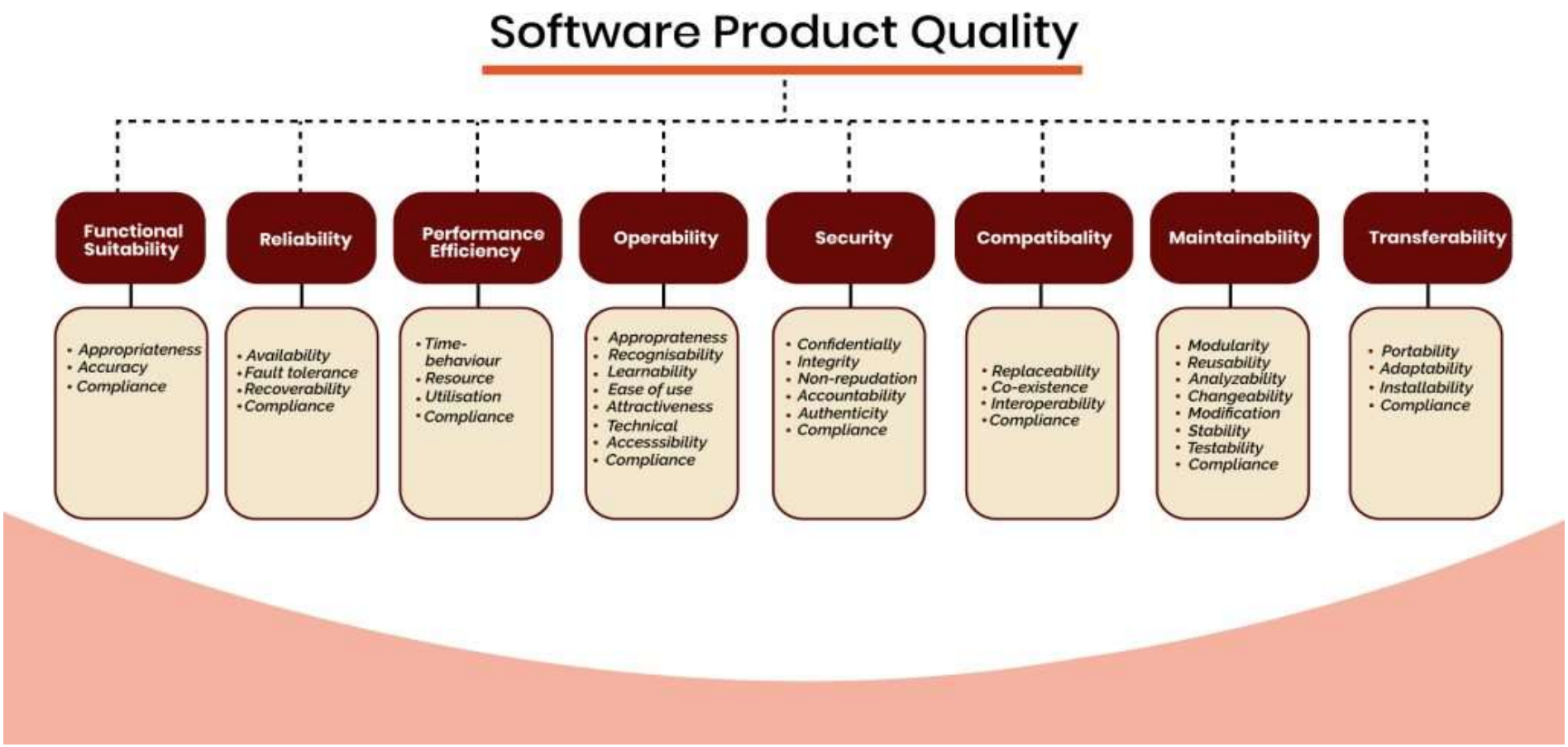


# INTRODUCTION



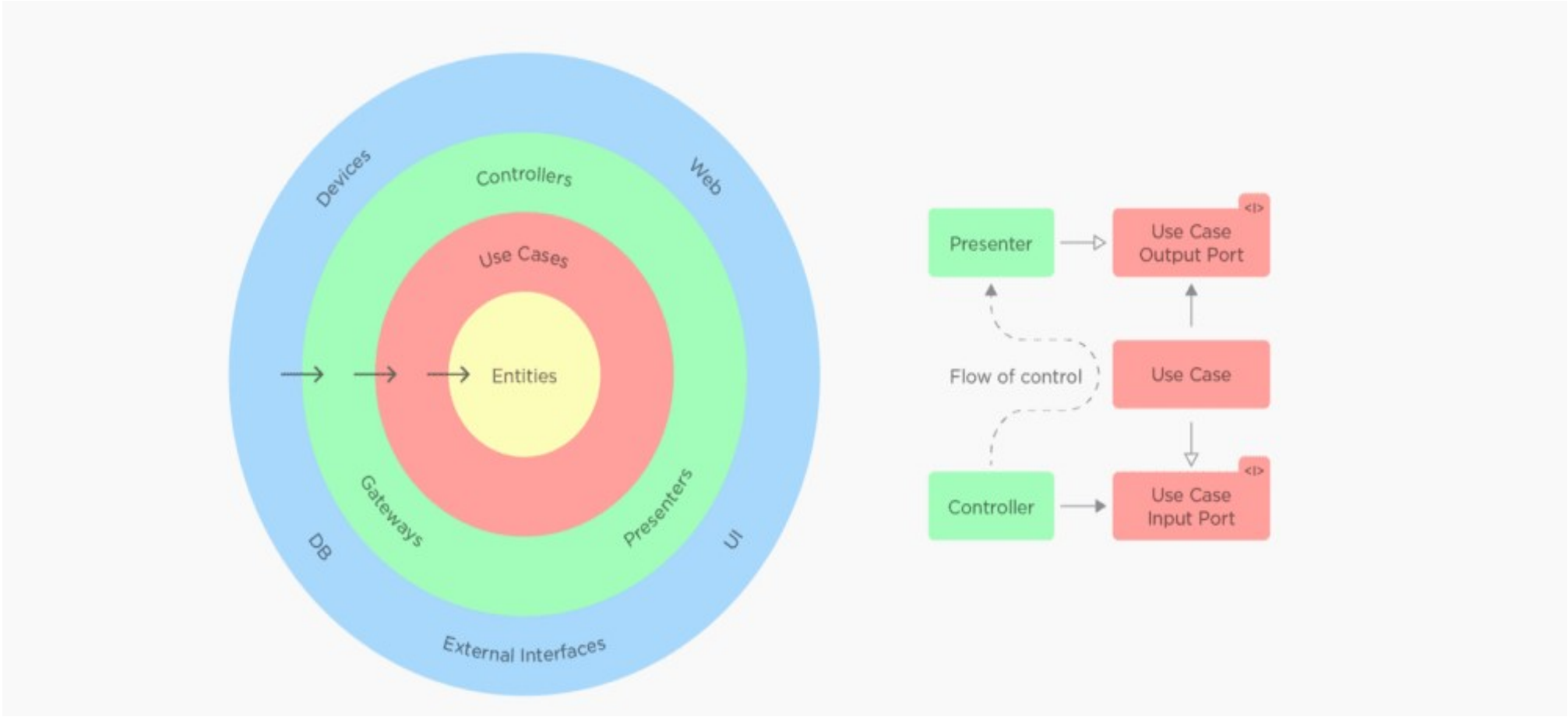
# 1/ Architecture

# Architecture : Software Quality

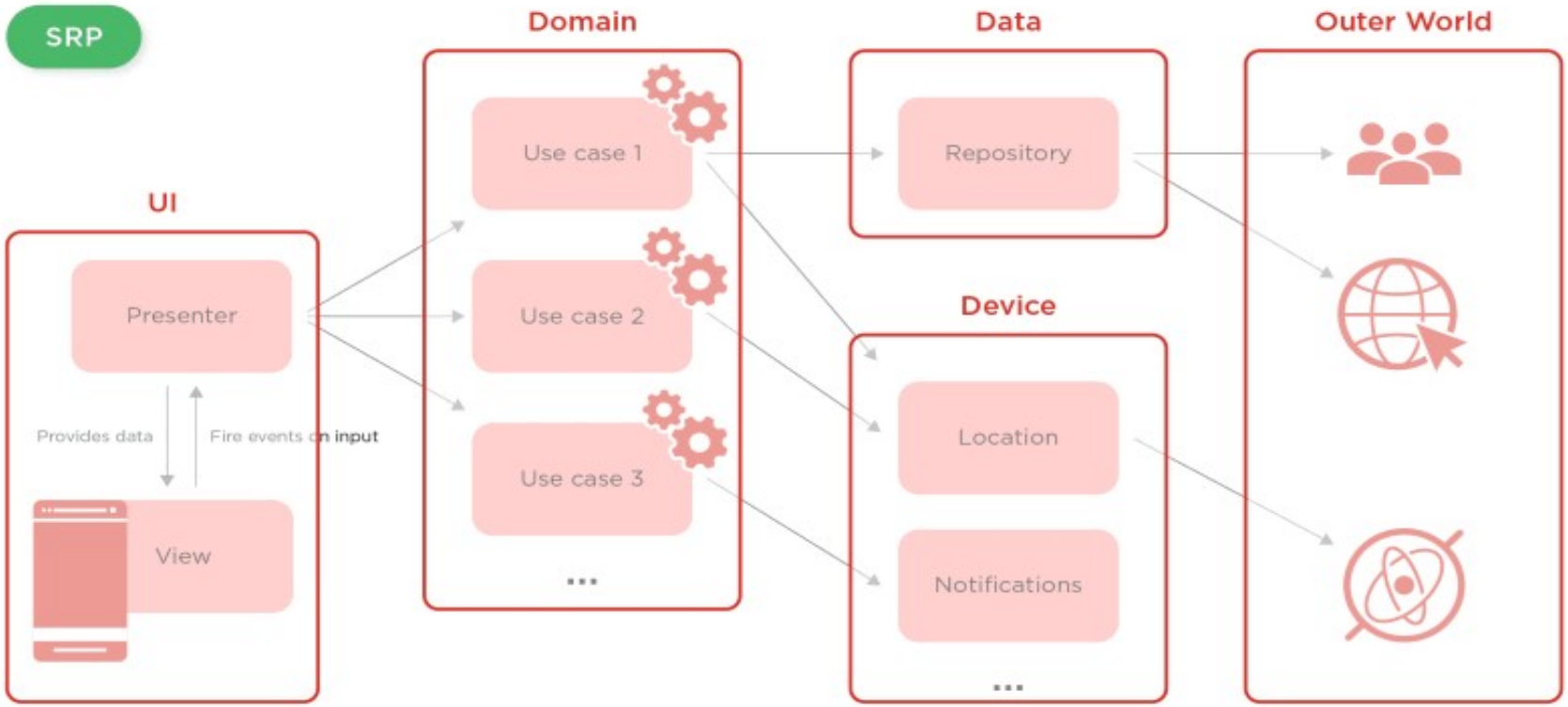


## Architecture : Software Quality

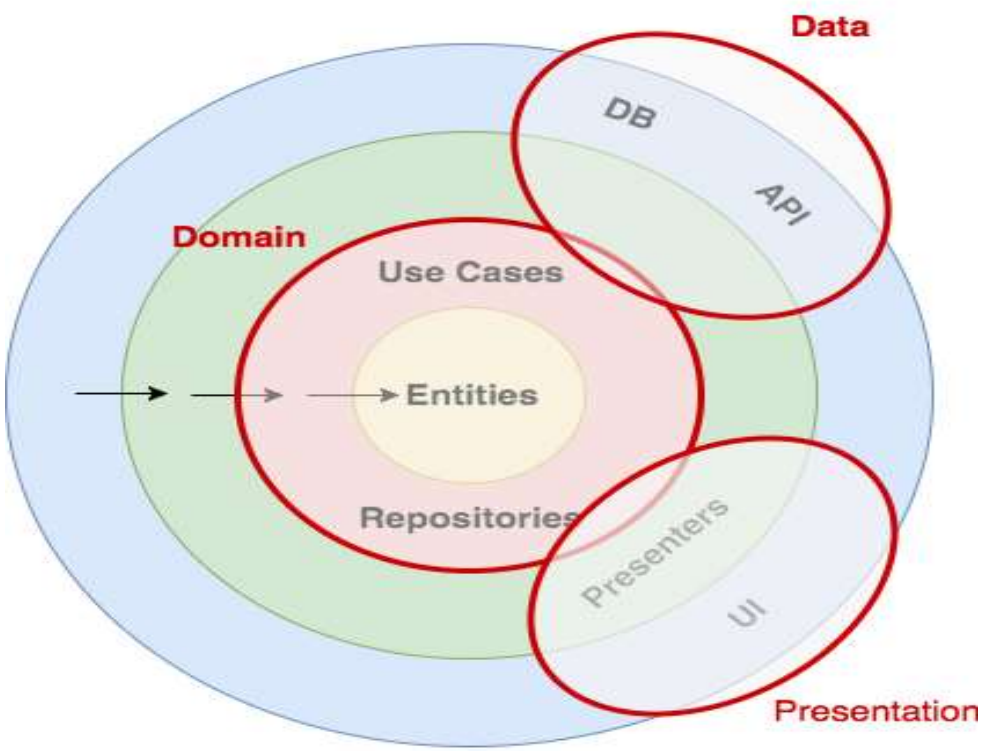
# Architecture : Clean Architecture

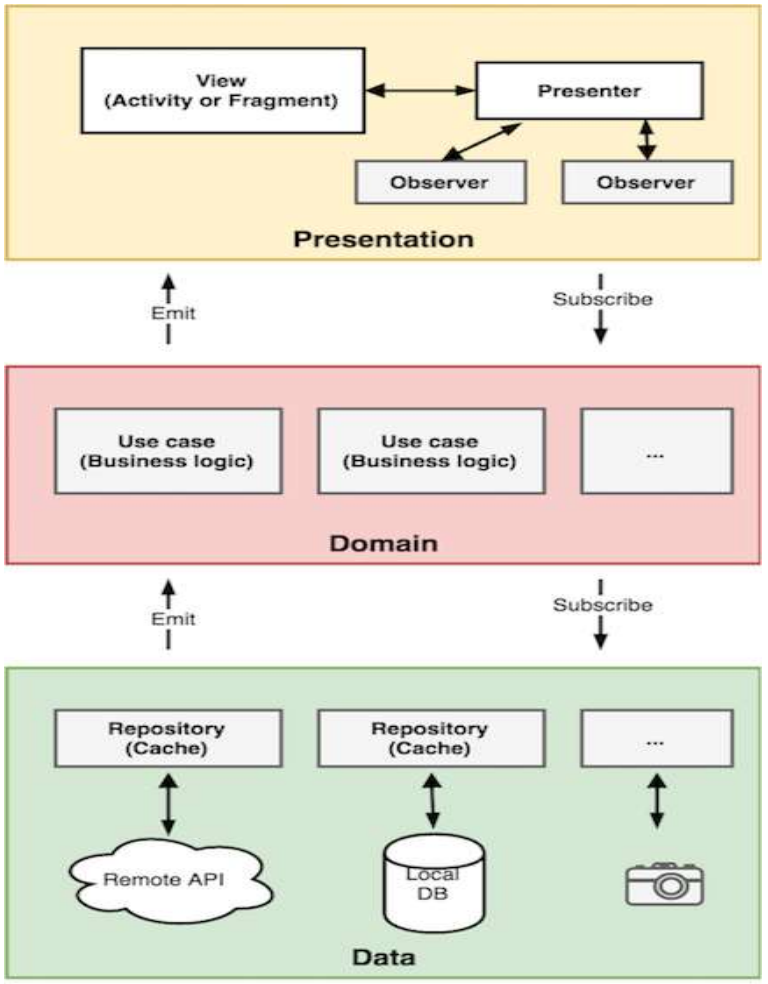


## Architecture : Clean Architecture



# Architecture : Clean Architecture




Patrick Yin

# 2/ JAVA JEE /Spring Boot

**Spring BOOT**

Norme JEE

# Core J2EE Pattern

**Presentation Tier**

- Intercepting Filter
- Front Controller
- View Helper
- Composite View
- Service to Worker
- Dispatcher View

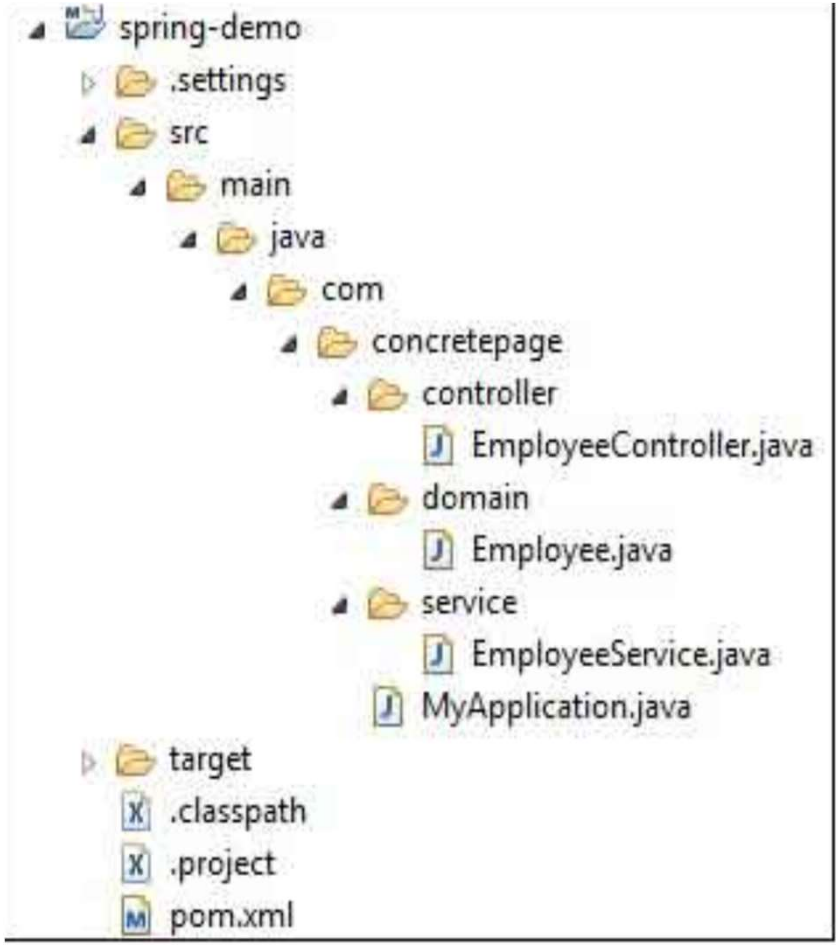
**Integration Tier**

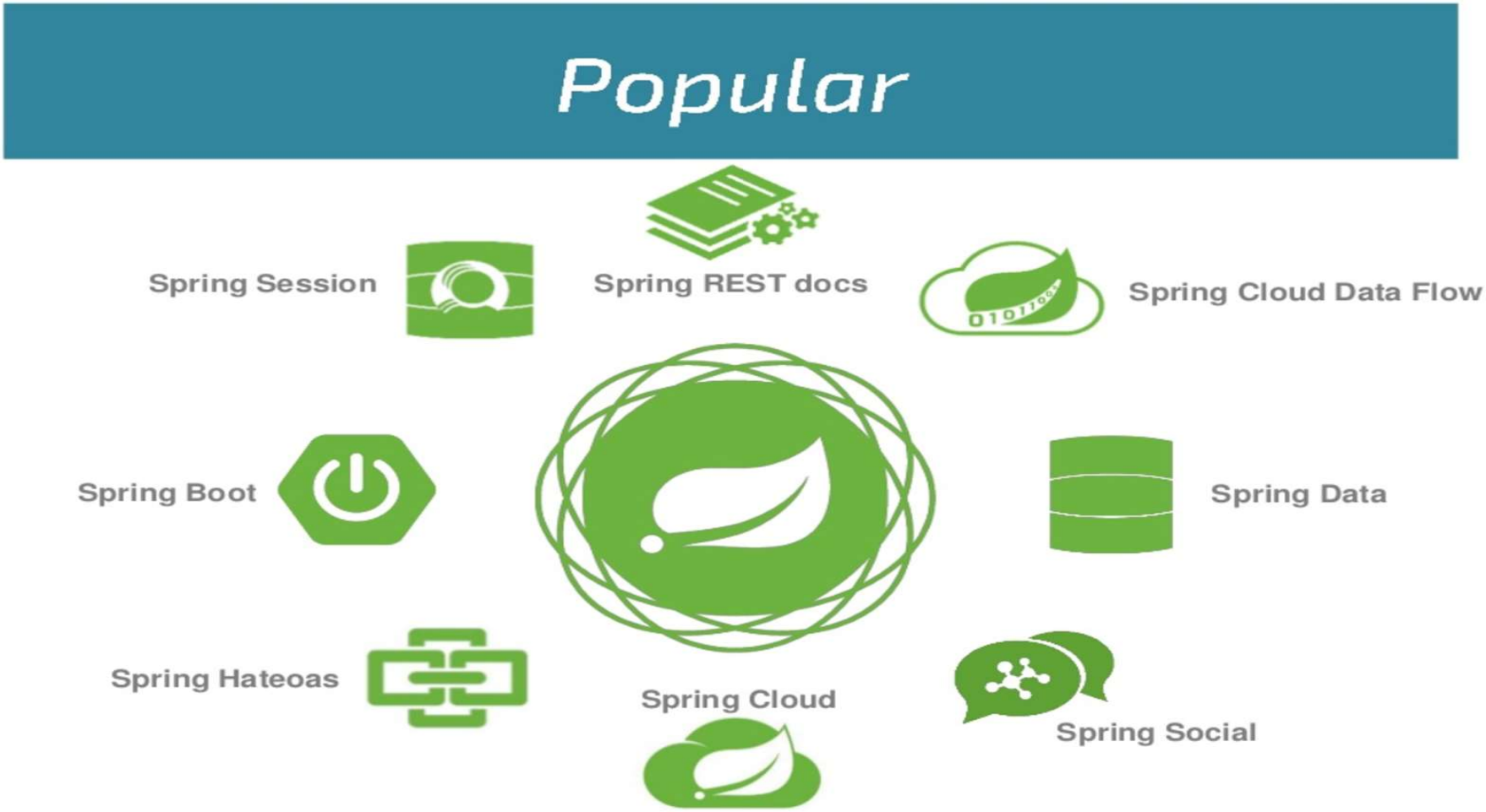
- Data Access Object
- Service Activator

**Business Tier**

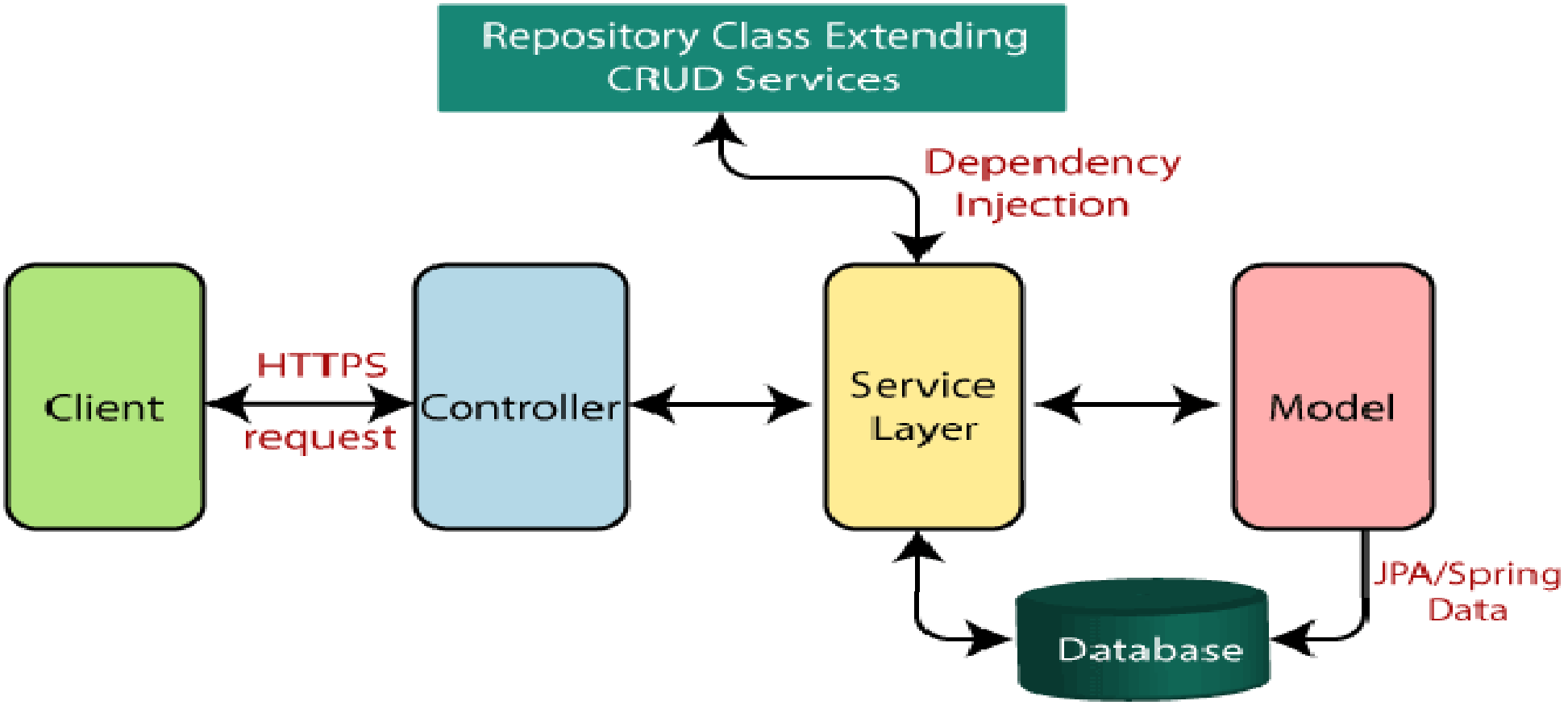
- Business Delegate
- Service Locator
- Session facade
- Transfer Object
- Transfer Object Assembler
- Value List Handler
- Composite Entity

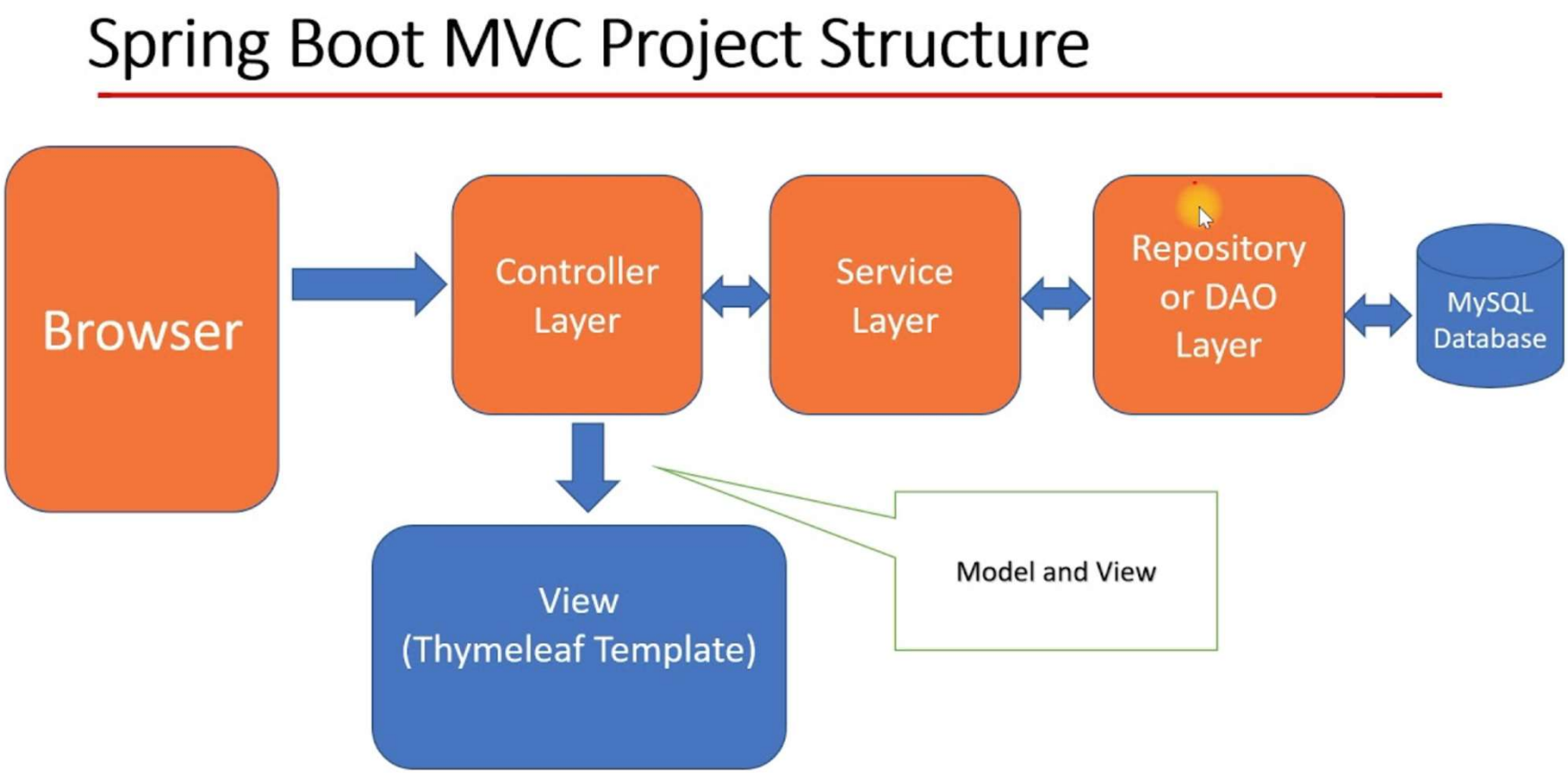
# MAVEN


spring-demo
.settings
src
main
java
com
concretepage
controller
EmployeeController.java
domain
Employee.java
service
EmployeeService.java
MyApplication.java
target
.classpath
.project
pom.xml

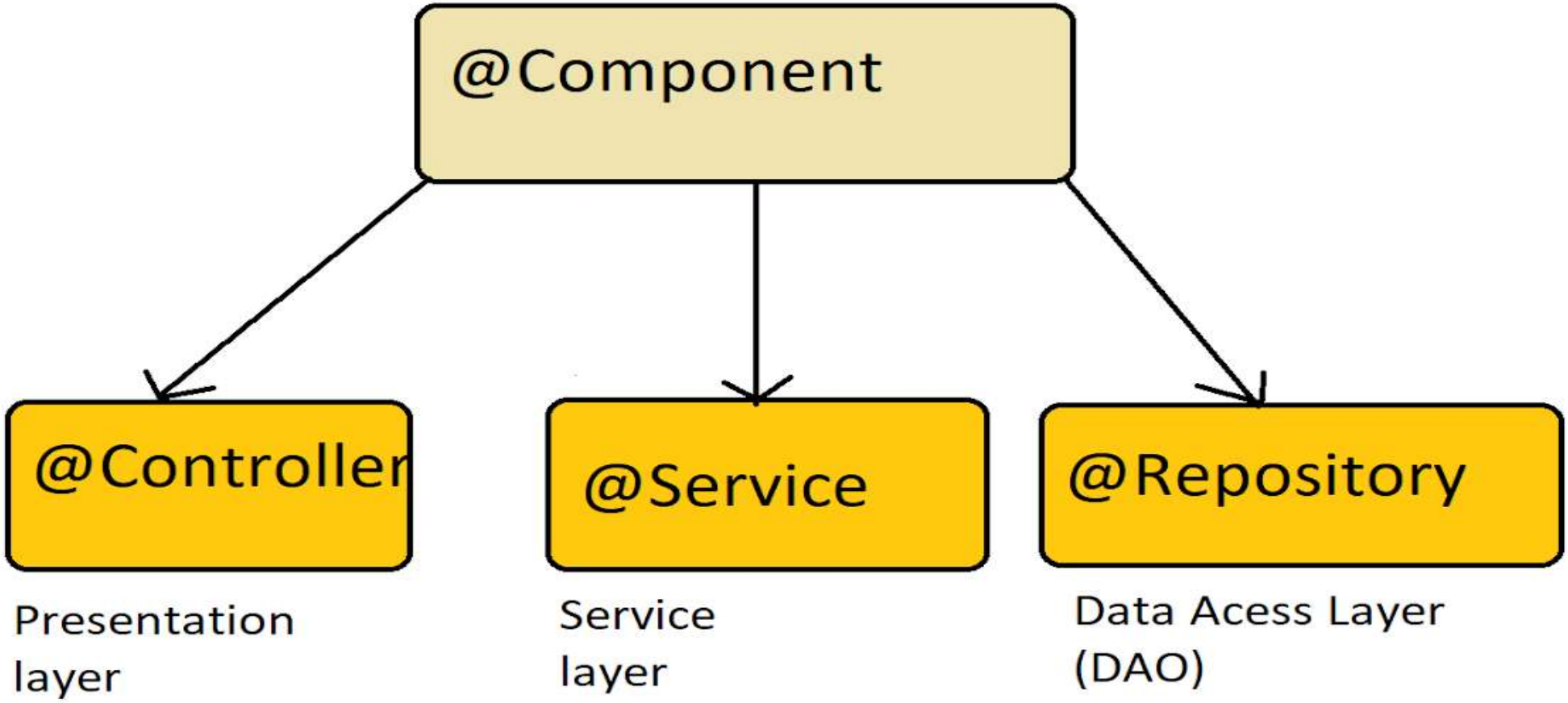

Popular
Spring Session
Spring REST docs
Spring Cloud Data Flow
Spring Boot
Spring Data
Spring Hateoas
Spring Cloud
Spring Social

# Spring Boot flow architecture



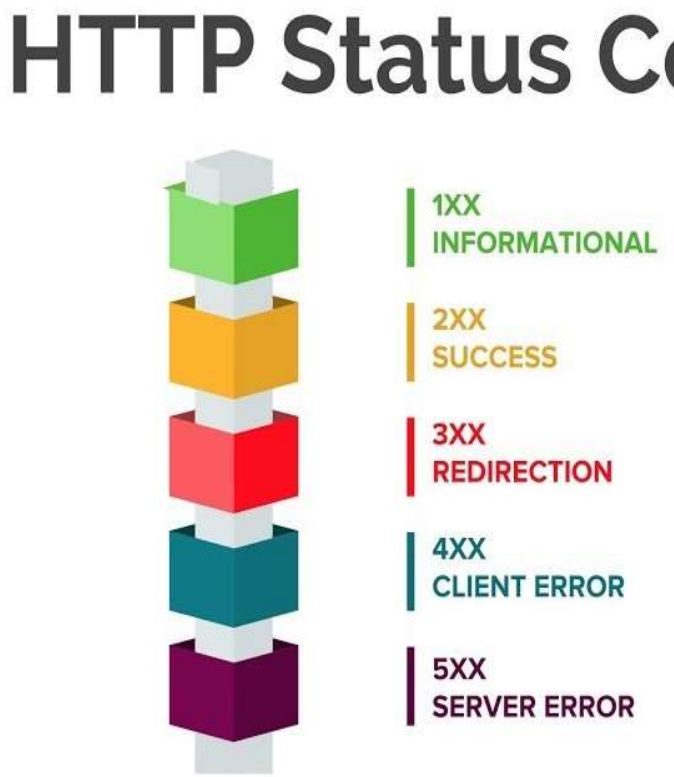

Spring Boot MVC Project Structure
Browser
Controller Layer
Service Layer
Repository or DAO Layer
MySQL Database
View (Thymeleaf Template)
Model and View

## Spring Boot  Component



# 3/ API REST

# HTTP Status Codes




## HTTP STATUS CODES

**2xx Success**

| | |
|---|---|
| 200 | Success / OK |

**3xx Redirection**

| | |
|---|---|
| 301 | Permanent Redirect |
| 302 | Temporary Redirect |
| 304 | Not Modified |

**4xx Client Error**

| | |
|---|---|
| 401 | Unauthorized Error |
| 403 | Forbidden |
| 404 | Not Found |
| 405 | Method Not Allowed |

**5xx Server Error**

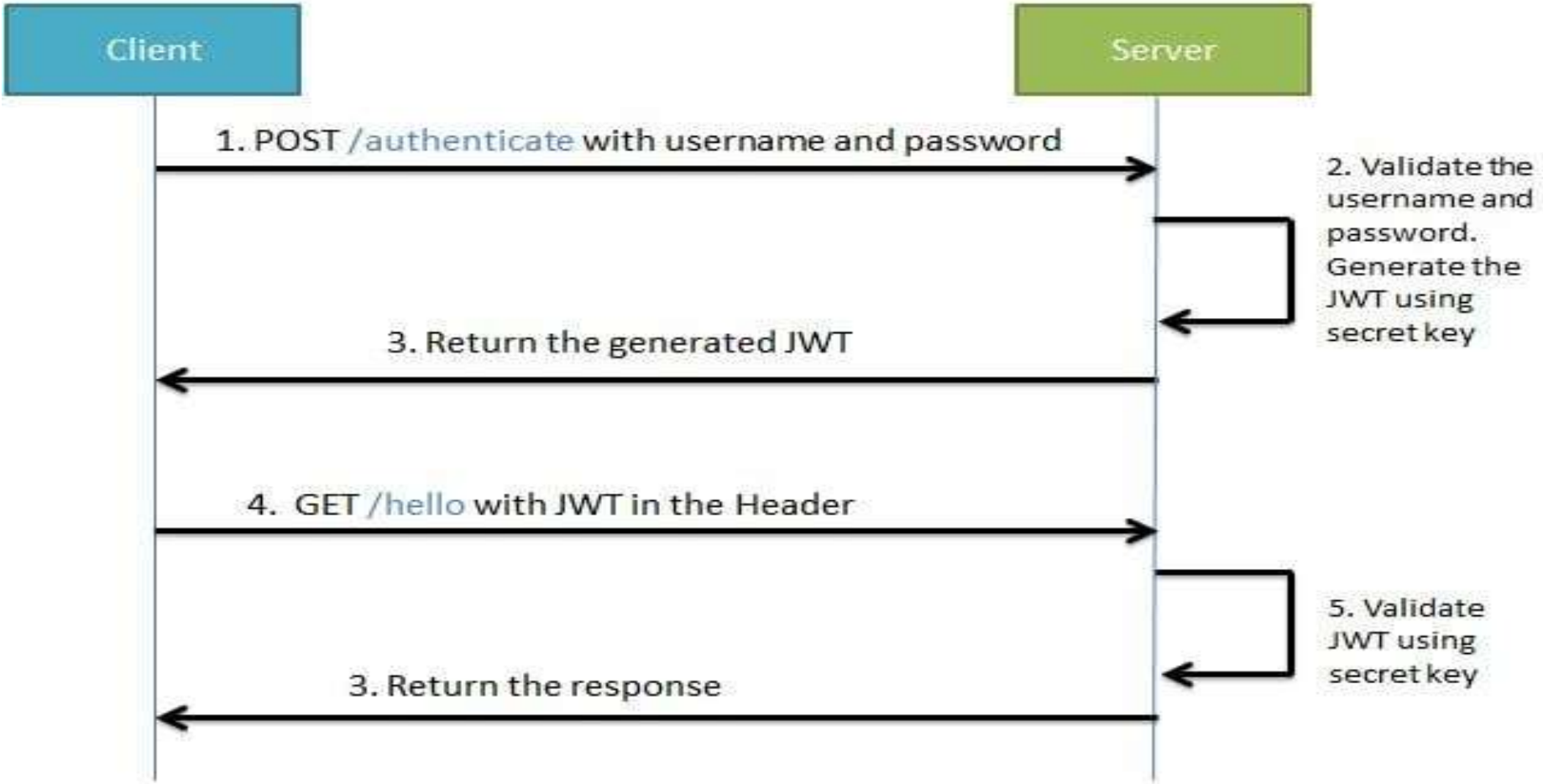
| | |
|---|---|
| 501 | Not Implemented |
| 502 | Bad Gateway |
| 503 | Service Unavailable |
| 504 | Gateway Timeout |

INFIDIGIT

# HTTP Status Codes

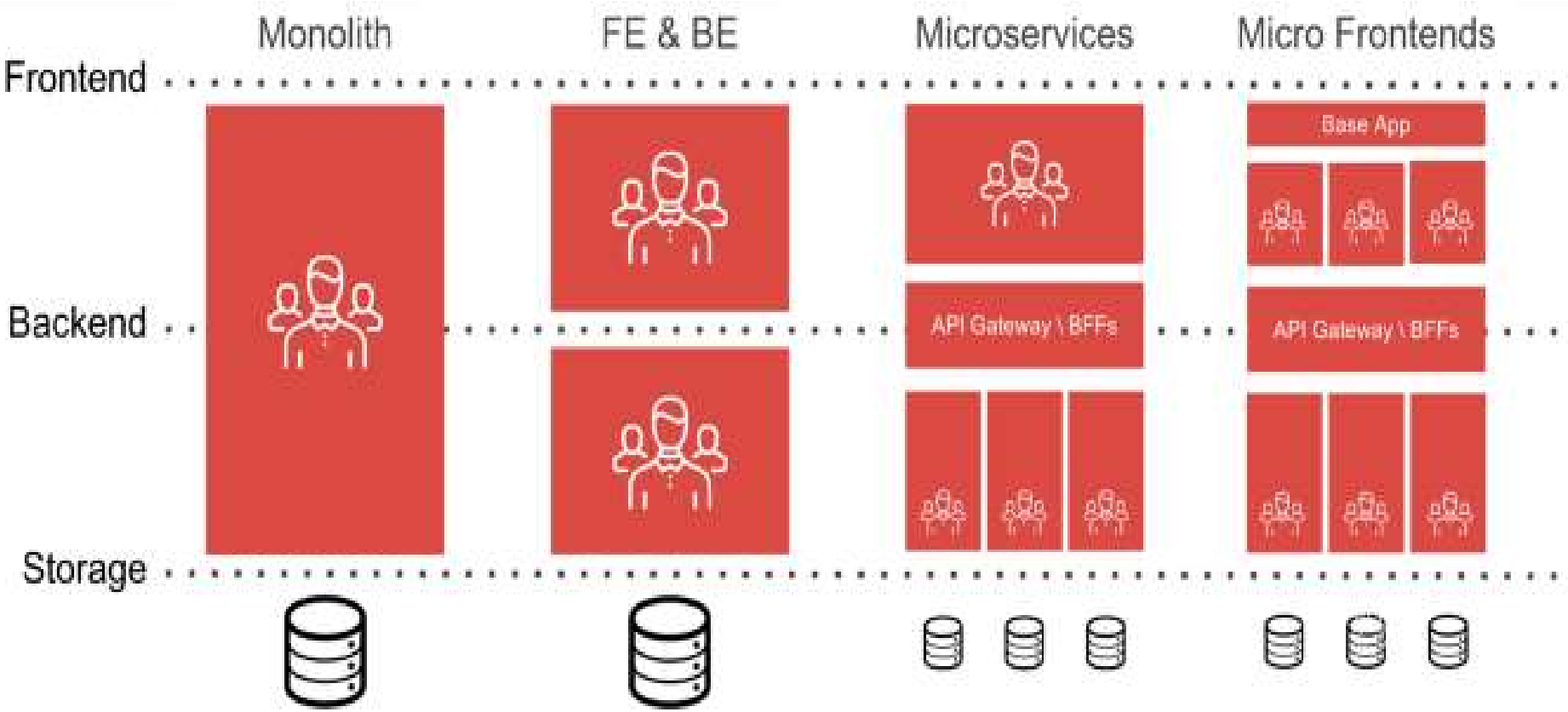
| Code | Description | Code | Description |
|---|---|---|---|
| 200 | OK | 400 | Bad Request |
| 201 | Created | 401 | Unauthorized |
| 202 | Accepted | 403 | Forbidden |
| 301 | Moved Permanently | 404 | Not Found |
| 303 | See Other | 410 | Gone |
| 304 | Not Modified | 500 | Internal Server Error |
| 307 | Temporary Redirect | 503 | Service Unavailable |

# API Sécurité avec JWT

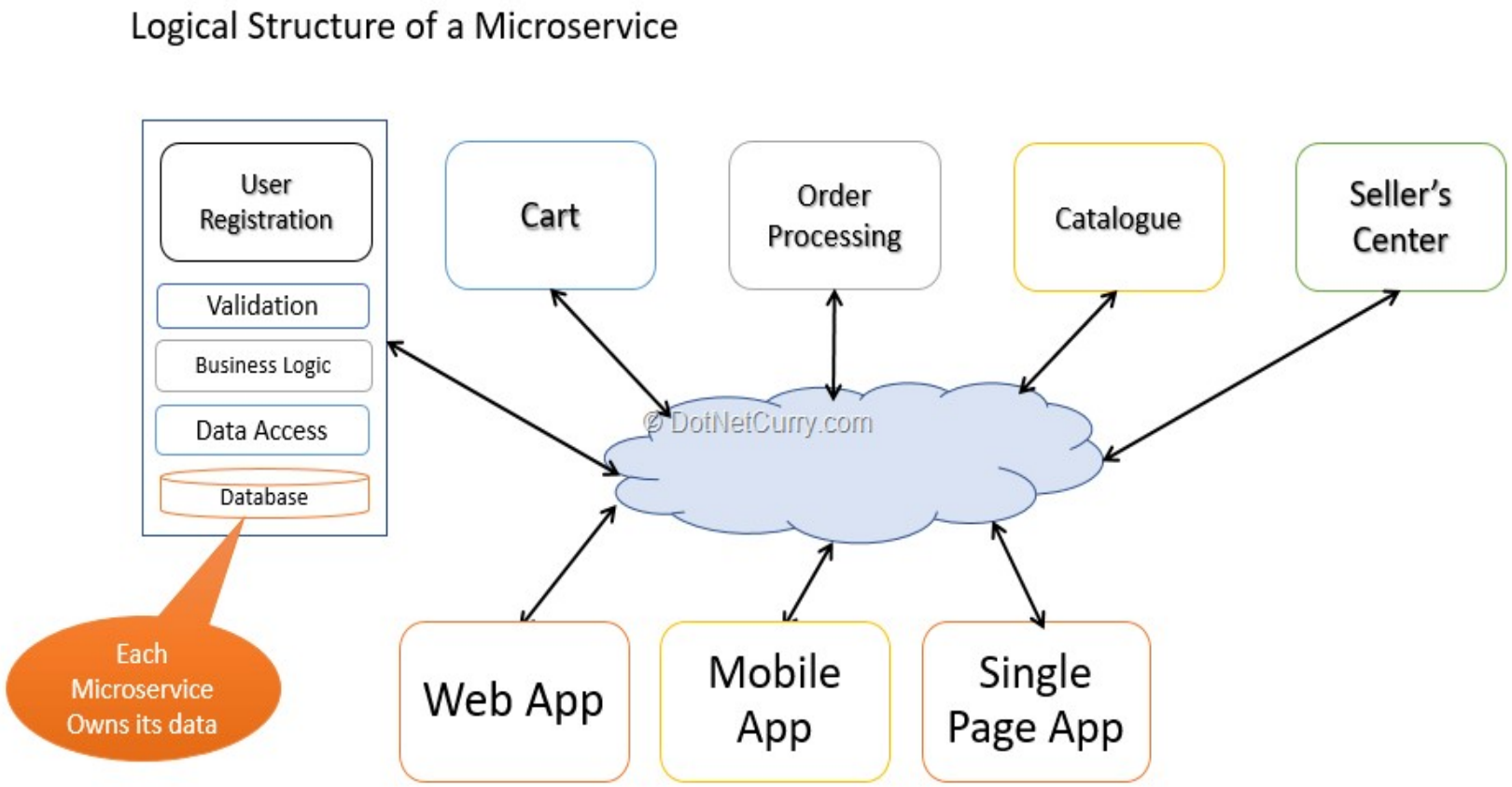


# 4/ MicroServices

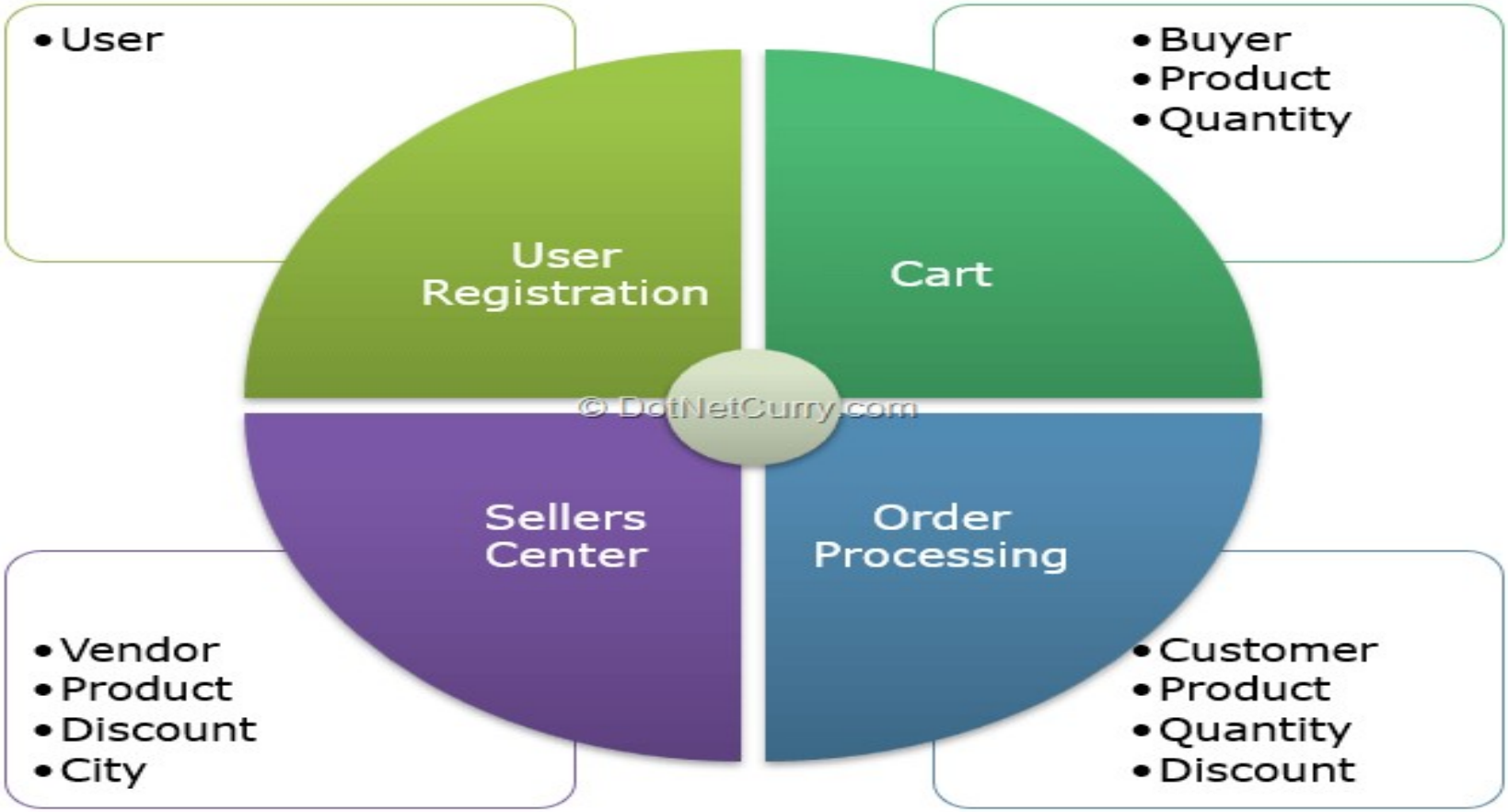
## Architecture et  Evolution du Monolith vers les MicroServices

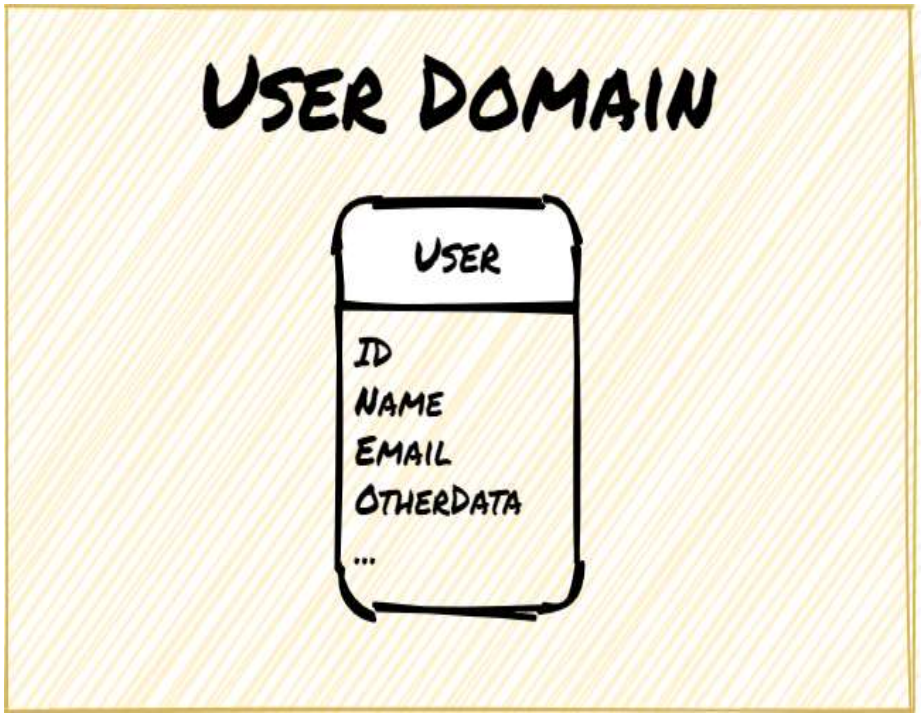


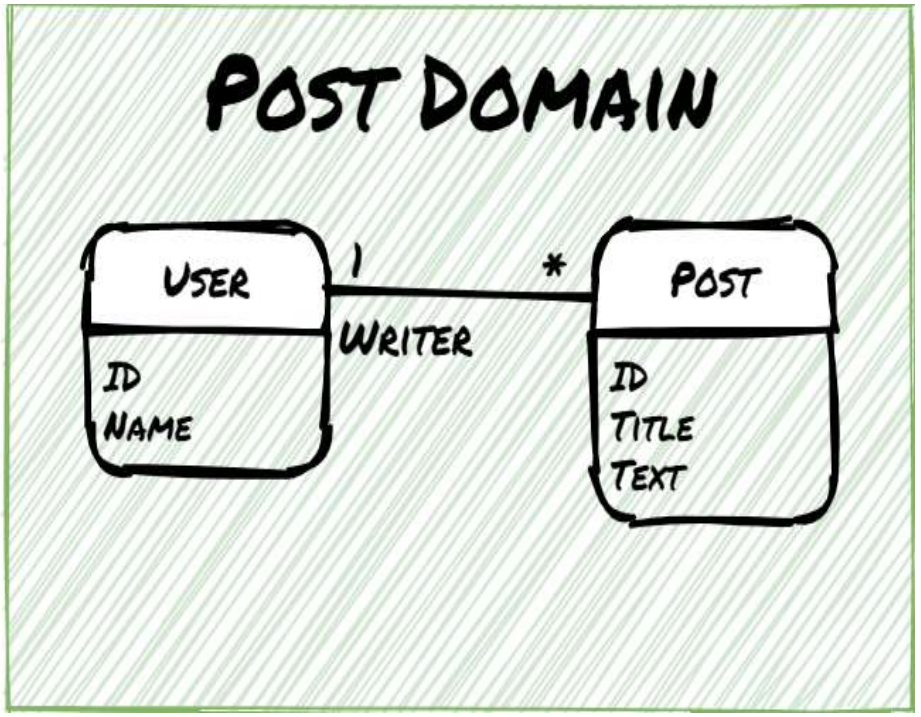
# Architecture Microservices / Découpage



## Architecture Microservices / Découpage



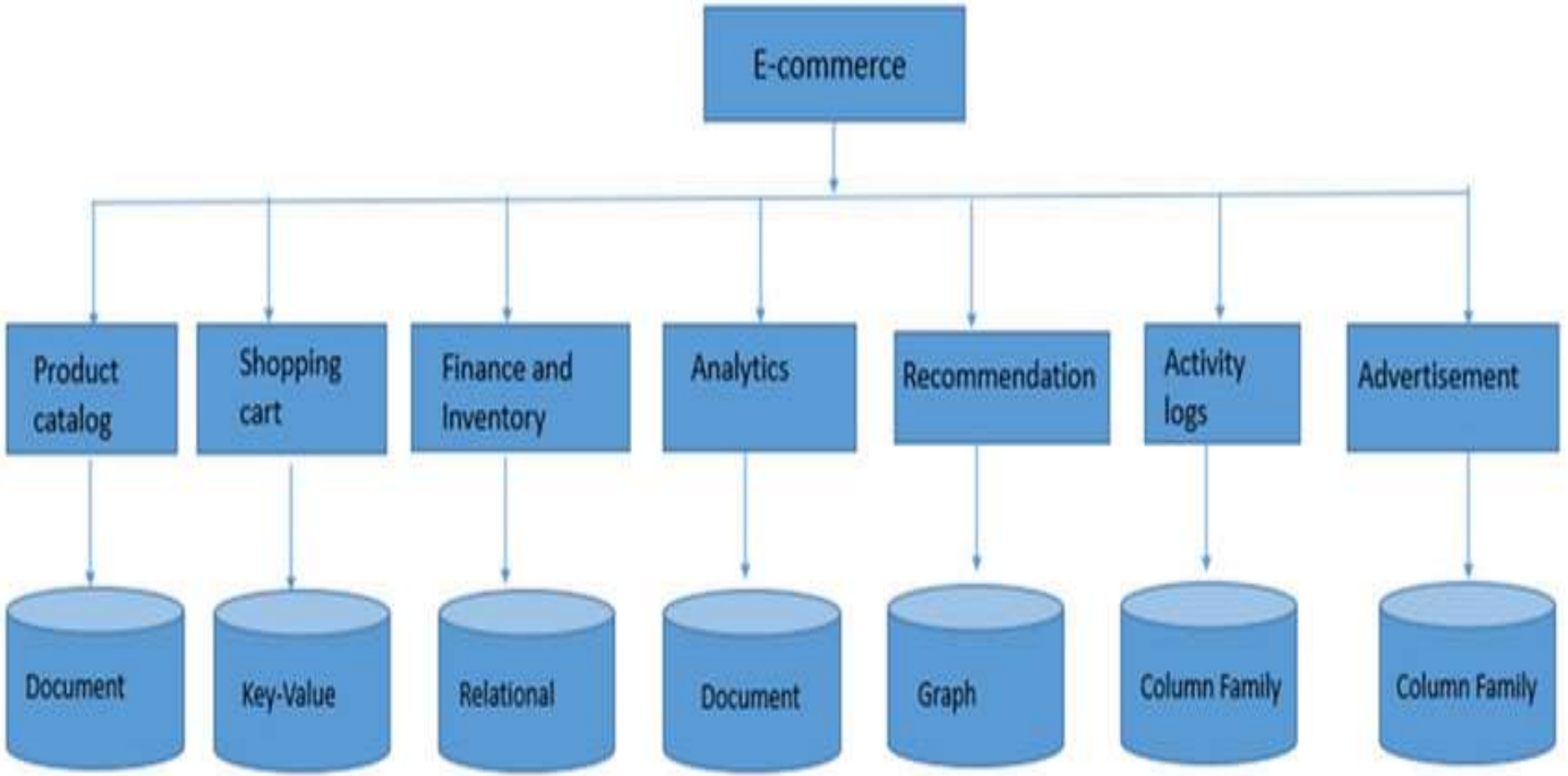

USER DOMAIN
USER
ID
NAME
EMAIL
OTHERDATA
...


POST DOMAIN
USER
ID
NAME
1
WRITER
*
POST
ID
TITLE
TEXT

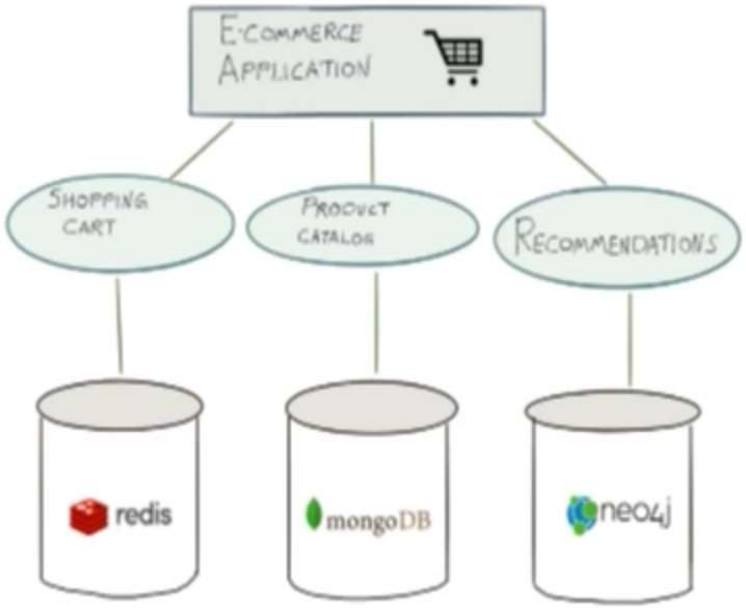
## Architecture Microservices : Polyglot  Persistence


- Different types of data in different ways
- Take advantage of strengths of different databases

## Architecture Microservices : Polyglot  Persistence

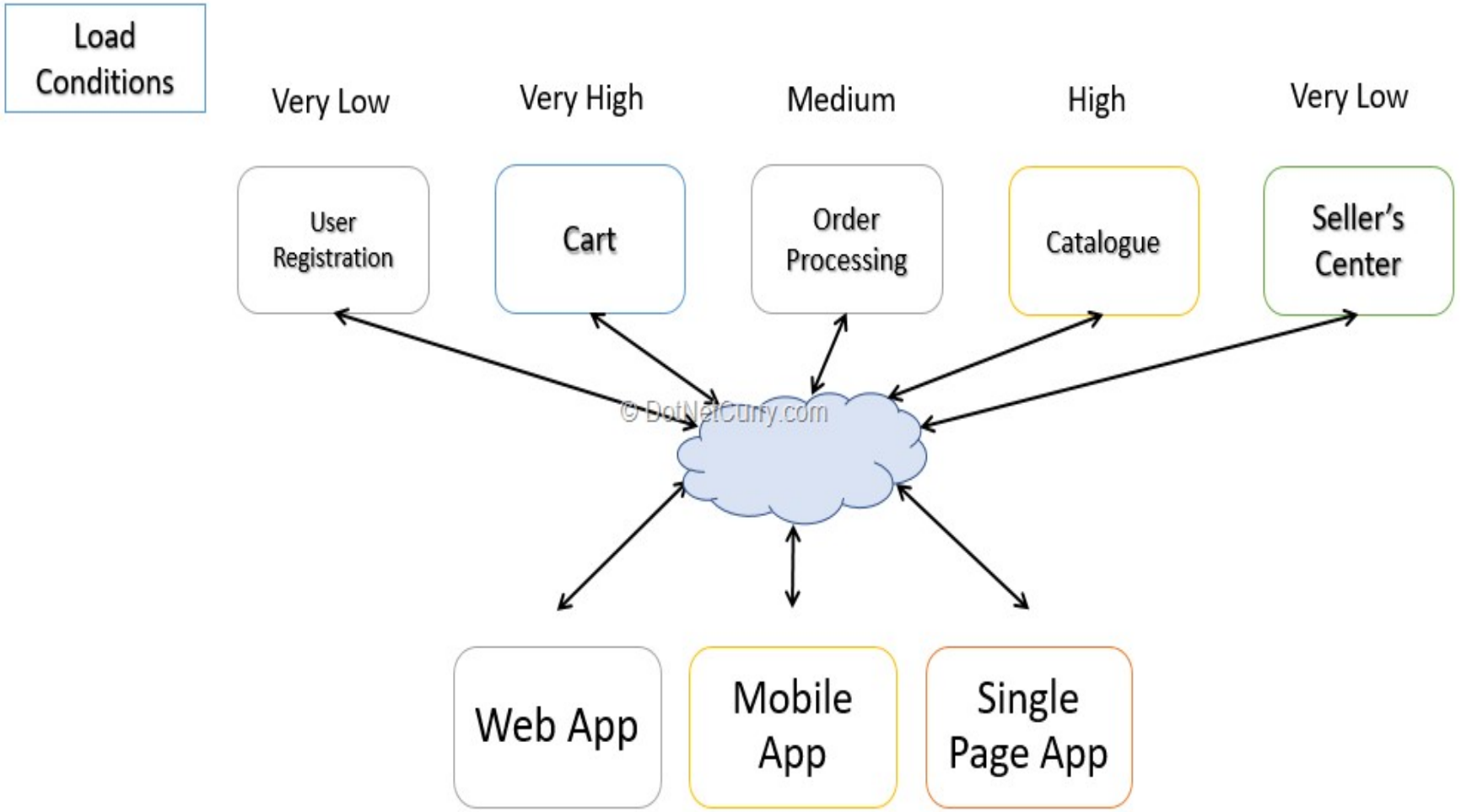


# Polyglot Persistence

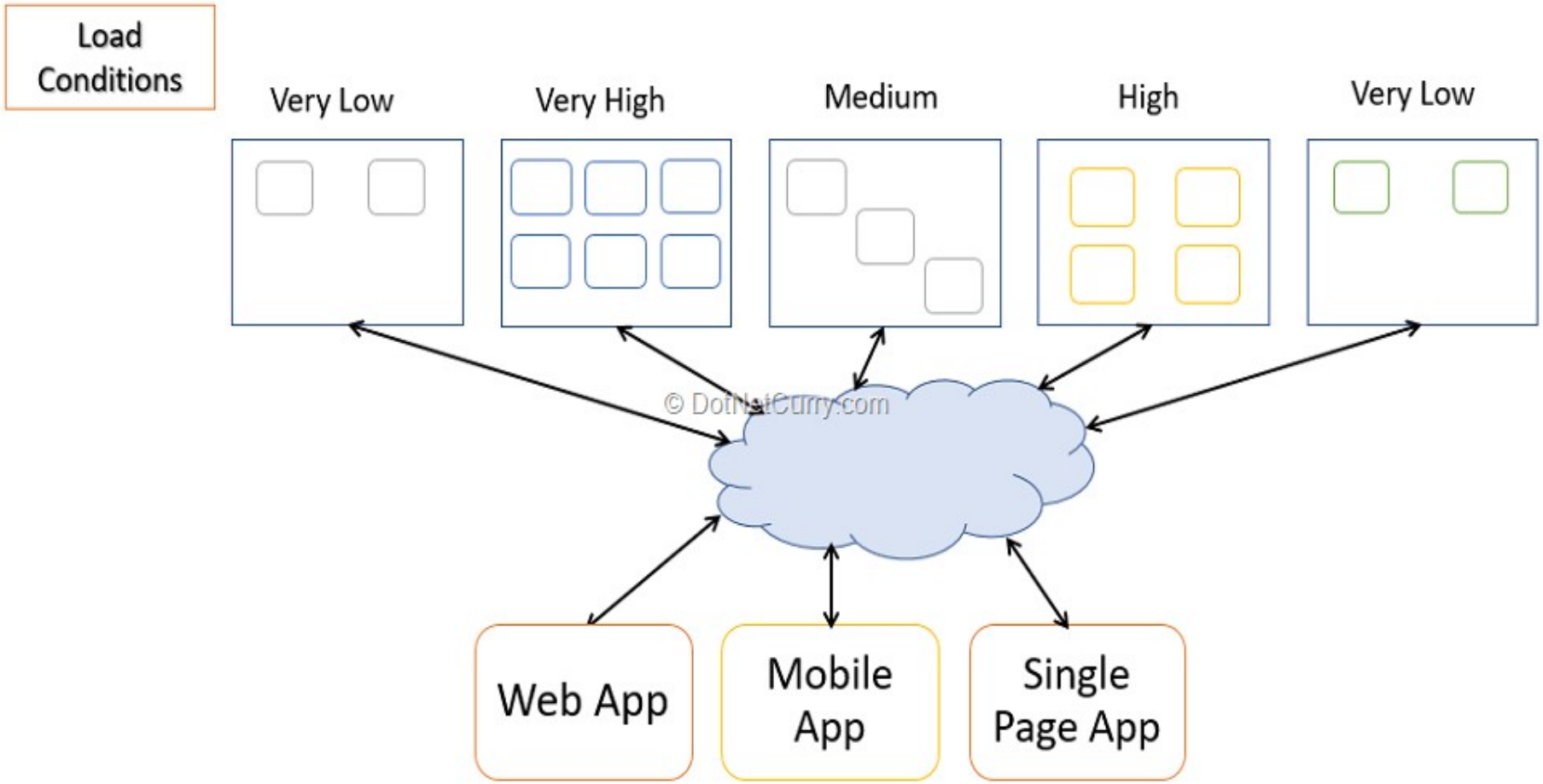



| | Functionality | Database type |
|---|---|---|
| Shopping Cart | Rapid session reads / writes | Key-value store |
| Orders / Product Catalog | Frequent reads | Document |
| Customer social graph | Recommendation | Graph |

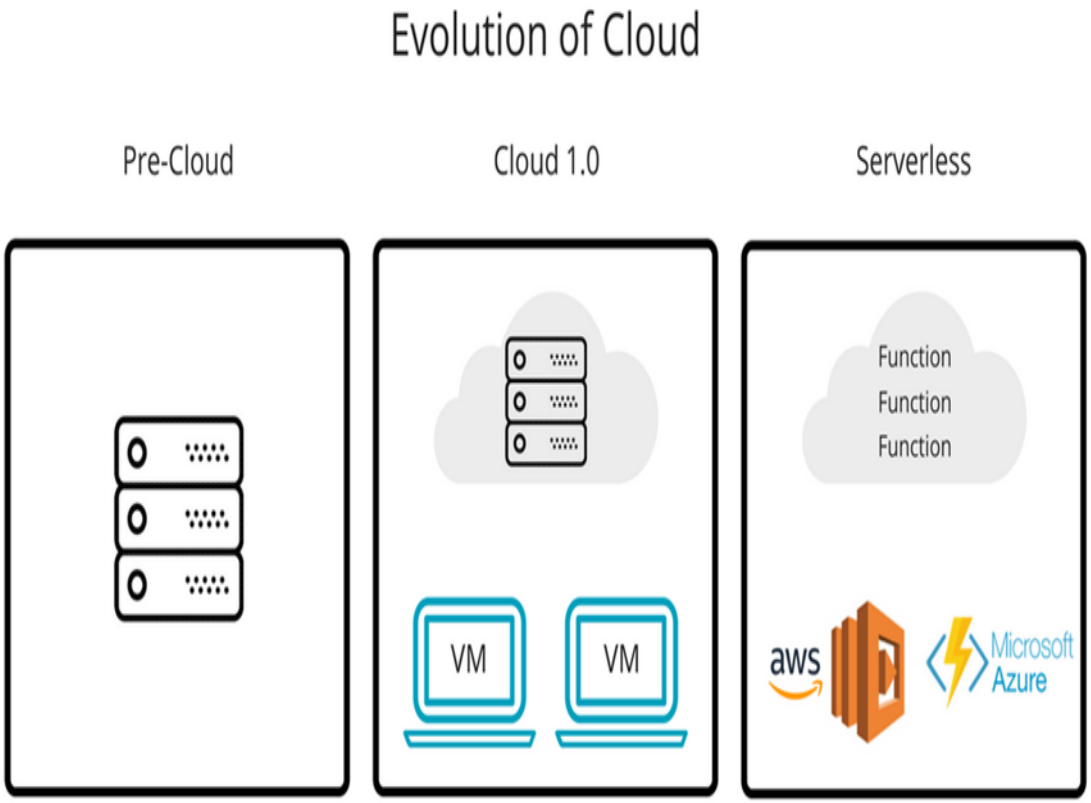
## Architecture Microservices : Charge et Scaling

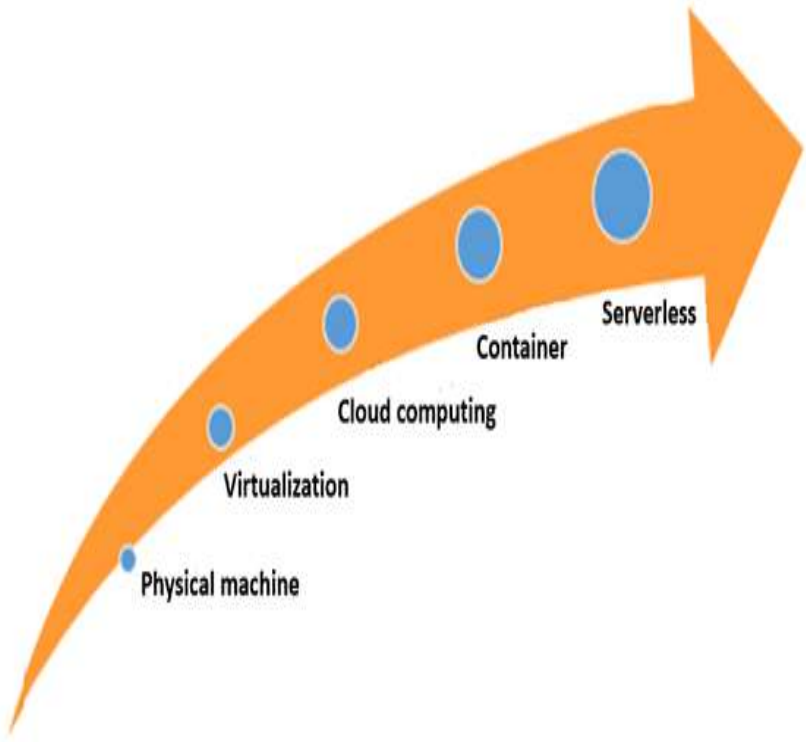


## Architecture Microservices : Charge et Scaling/ Instances

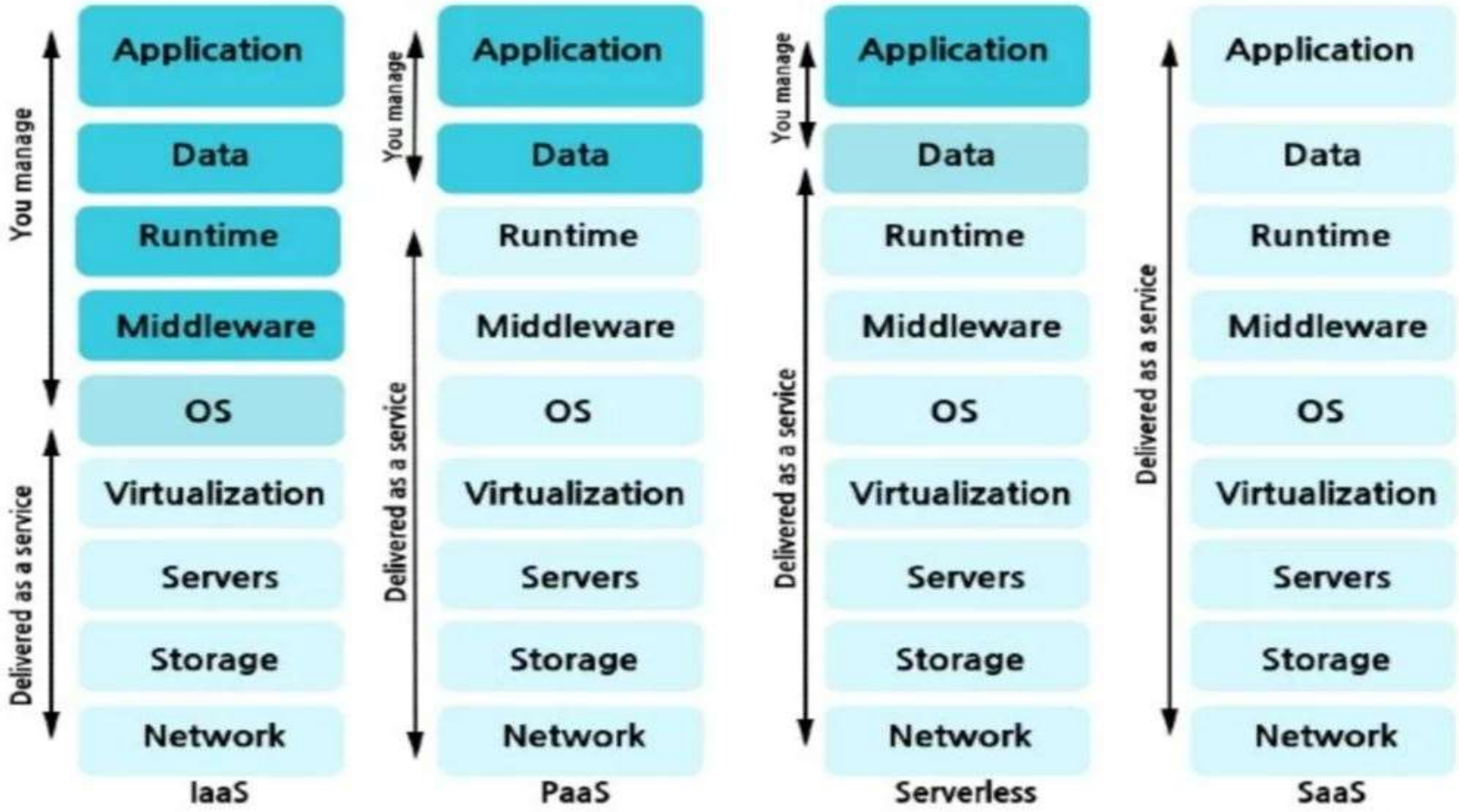


## Architecture : Traditional vs Serverless

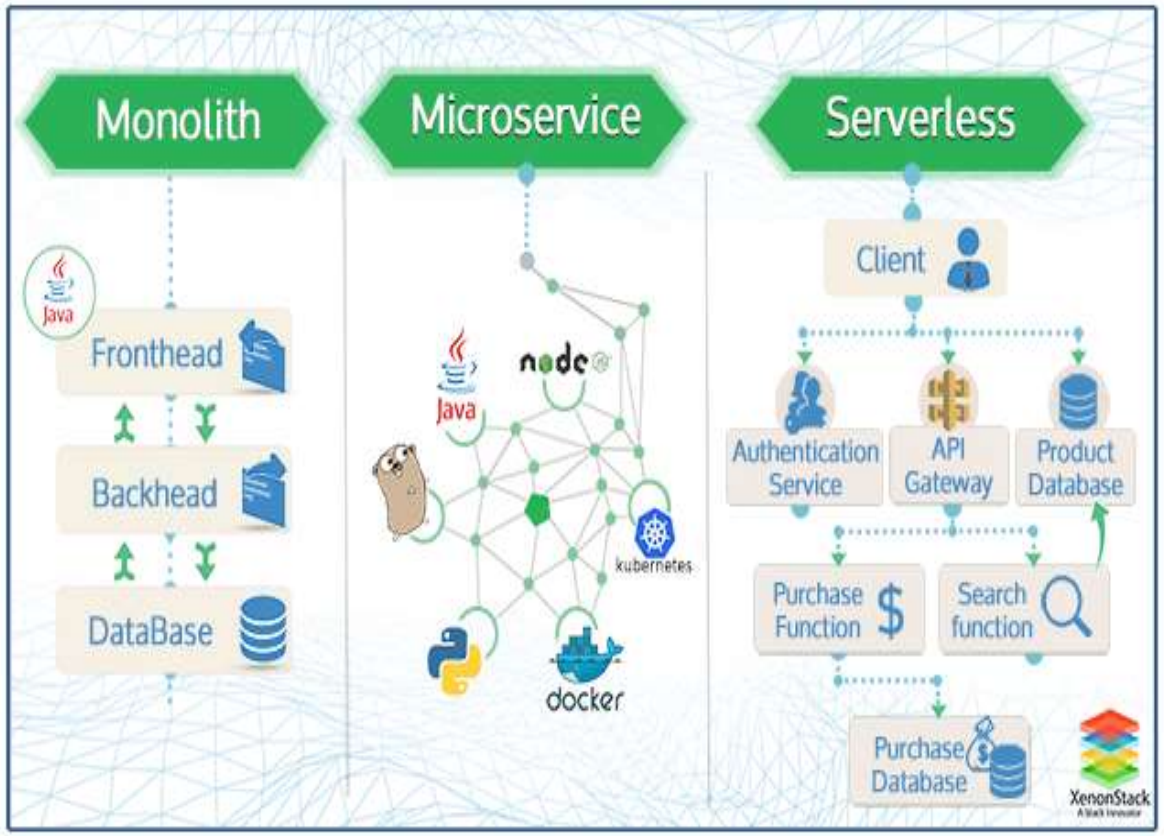


## Architecture : CLOUD Evolution



## Architecture : Traditional vs Serverless




# Architecture : Traditional vs Serverless

Avantages de l'architecture sans serveur

**•Productivité :**
les développeurs peuvent se concentrer sur leur produit principal au lieu de se soucier de la gestion et de l'exploitation des serveurs ou des environnements d'exécution. Cela réduit les frais généraux et permet aux développeurs de récupérer du temps et de l'énergie qui peuvent être consacrés au développement d'excellents produits évolutifs et fiables.

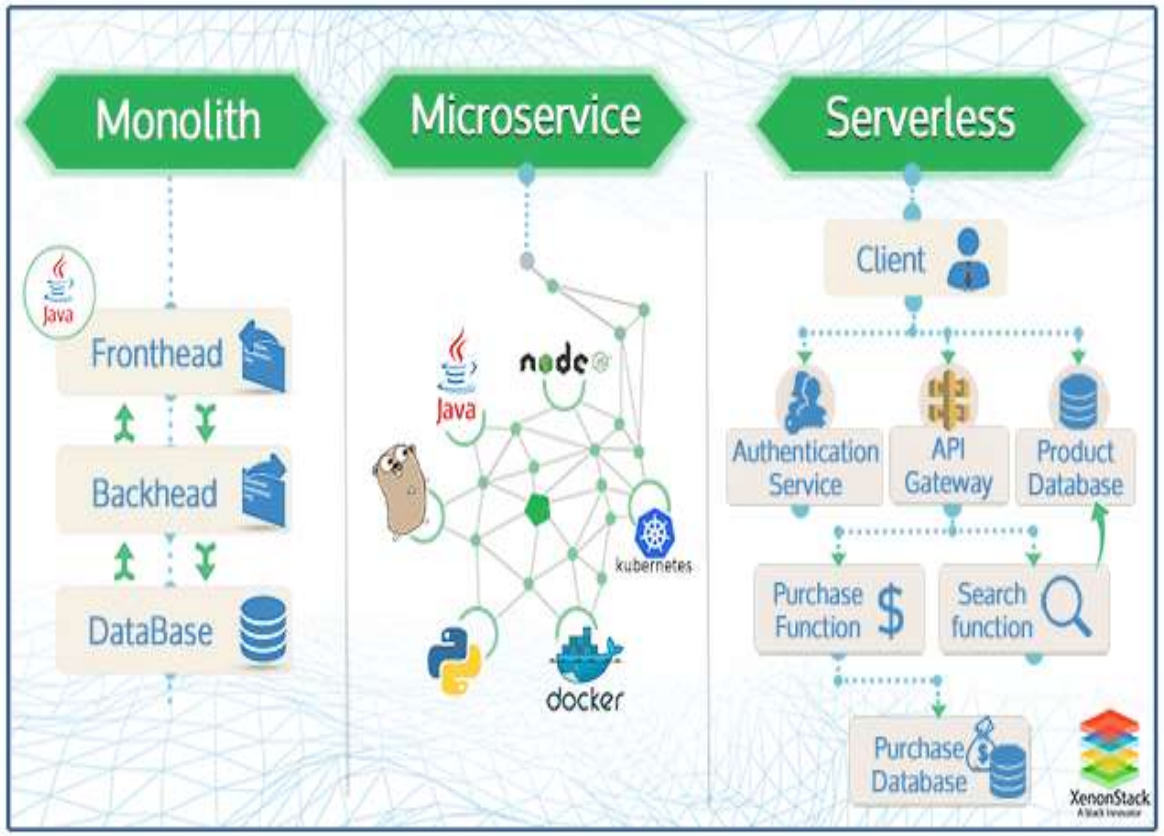
**•Mise à l'échelle :**
l'application peut être mise à l'échelle automatiquement ou en ajustant sa capacité via une valeur seuil, en fonction des unités de consommation (par exemple, débit, mémoire) plutôt que des unités de serveurs individuels. Et vous devez payer pour un débit ou une durée d'exécution constants plutôt que par unités de serveur individuelles. Pas besoin de s'inquiéter du nombre de requêtes simultanées.

**•Disponibilité :**
nous n'avons pas à nous soucier des capacités ou du besoin d'architecturer les choses, cela est automatisé du côté du fournisseur. ils offrent une disponibilité et une tolérance aux pannes intégrées pour les services.

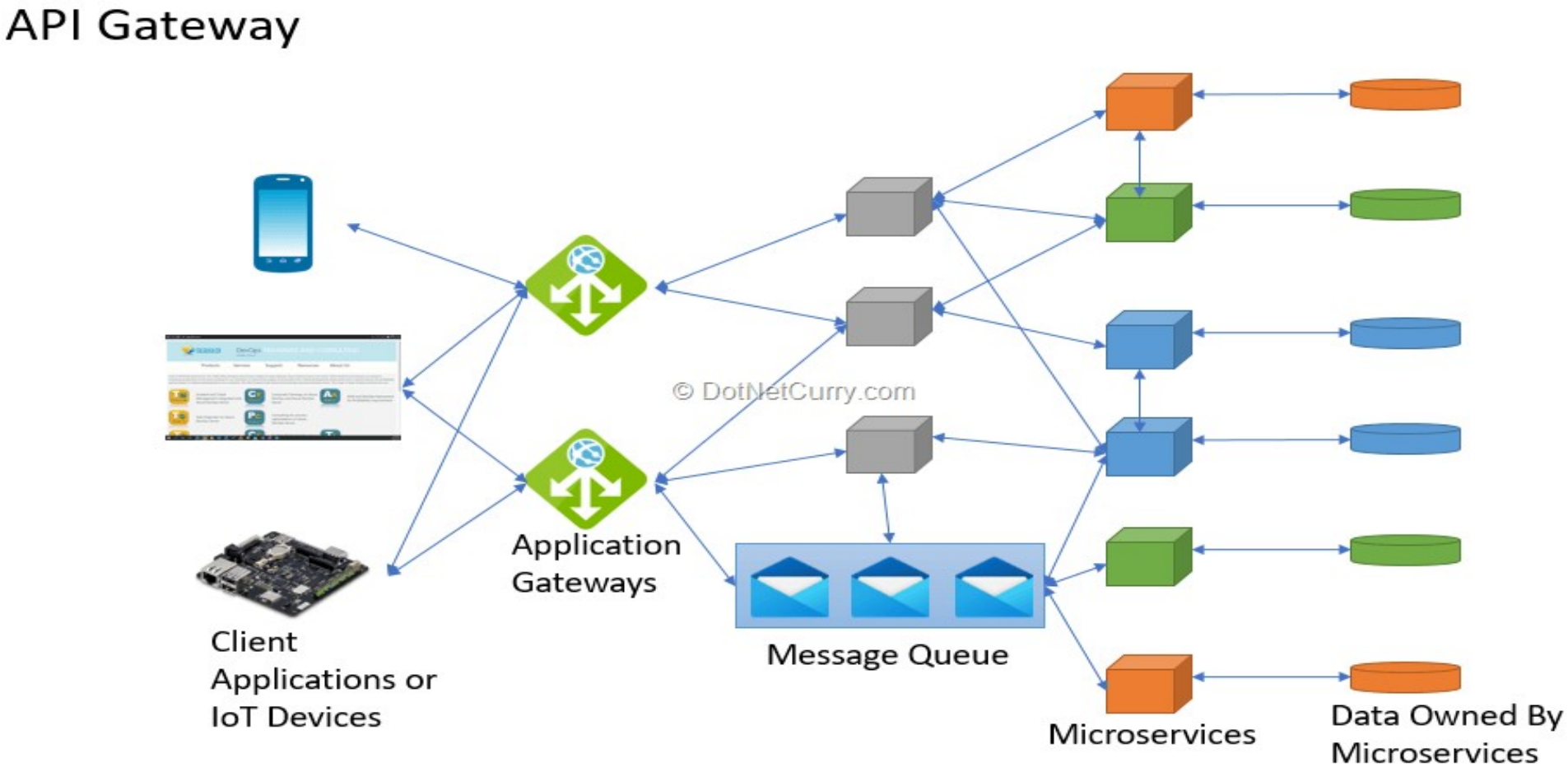
**•Payer à l'utilisation :**
cela ne vous coûte rien de courir si personne ne l'utilise. Inconsciemment, hors frais de stockage de données. Il réduit les coûts d'exploitation en termes d'infrastructure et d'exploitation. Le coût est basé sur le nombre d'exécutions de fonction, mesuré en millisecondes au lieu d'heures.

**•Agilité :**
Des unités déployables plus petites permettent une livraison plus rapide des fonctionnalités sur le marché, augmentant la capacité d'adaptation au changement.

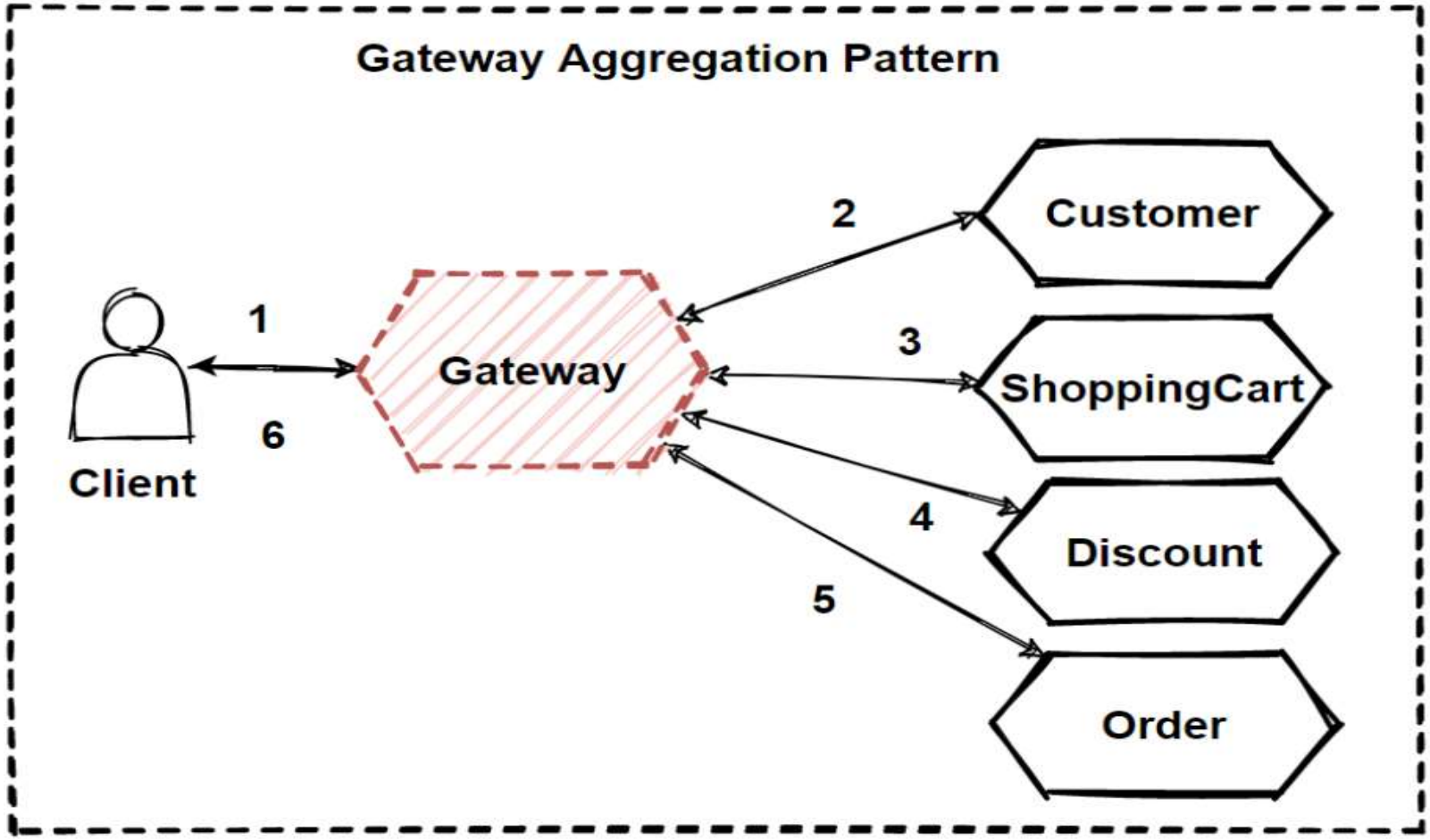
## Architecture : Traditional vs Serverless



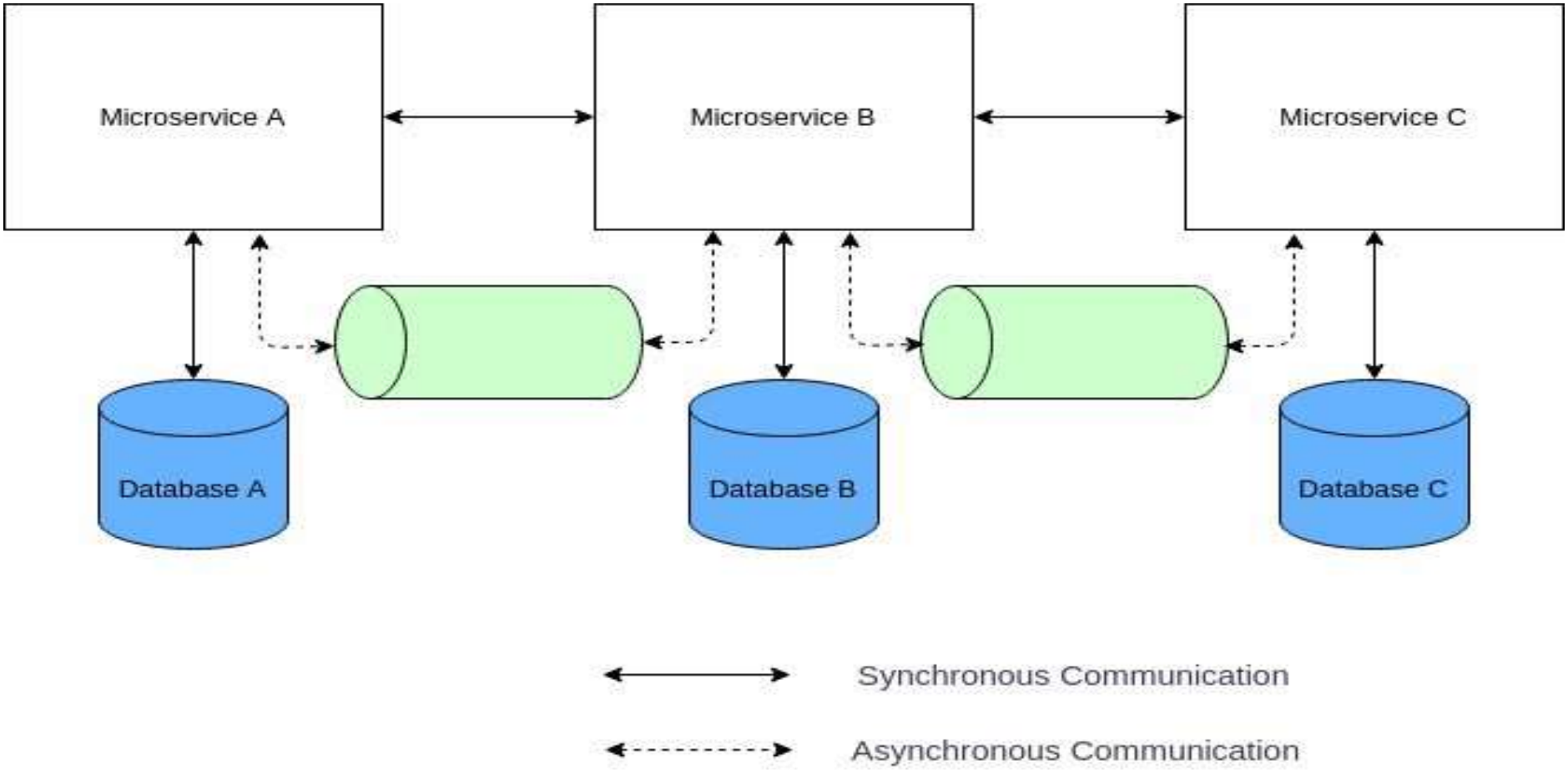

# Architecture Microservices : Application Gateways / API Gateways / Events Bus



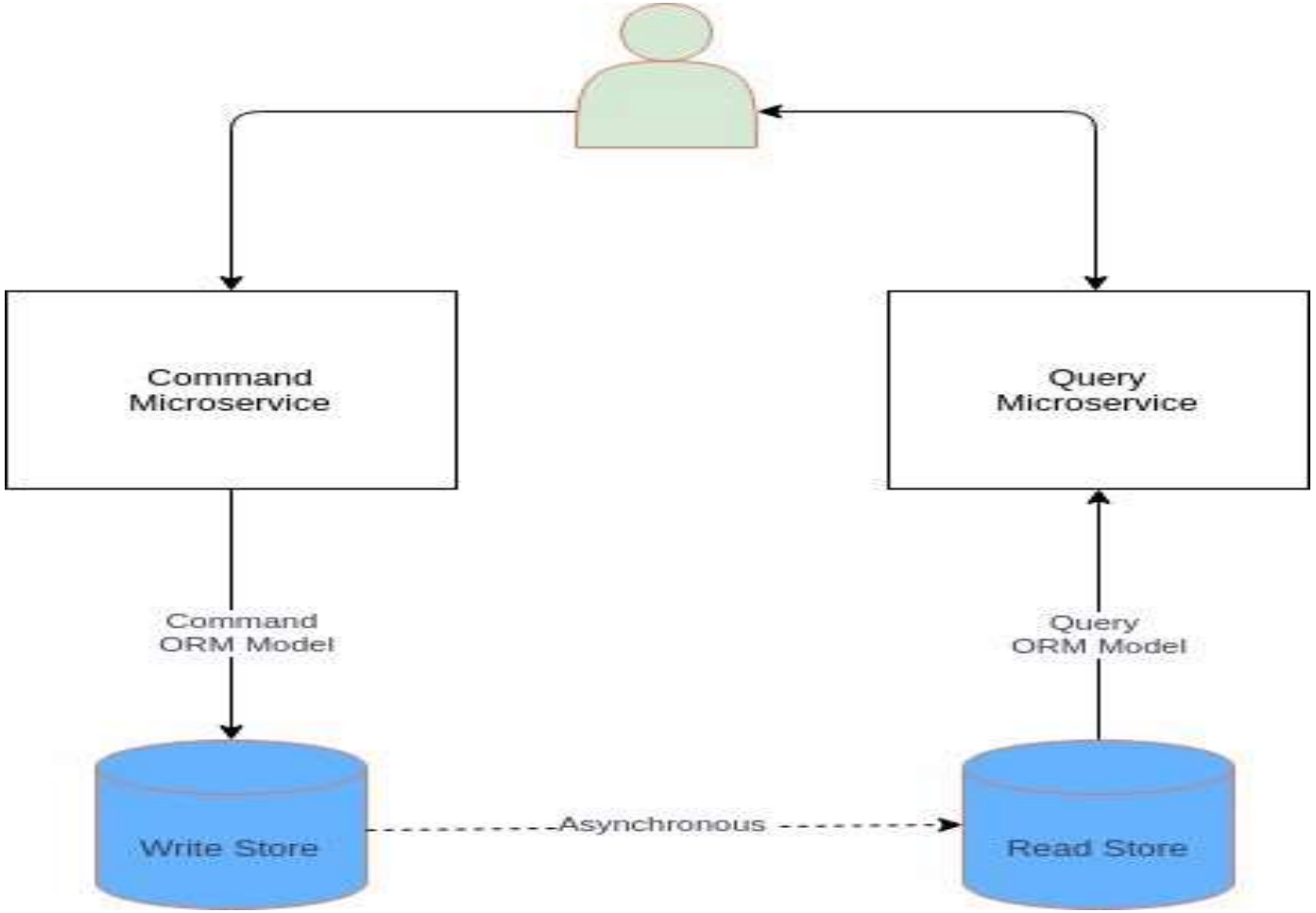
# Architecture Microservices : Application Gateways / API Gateways / Events Bus



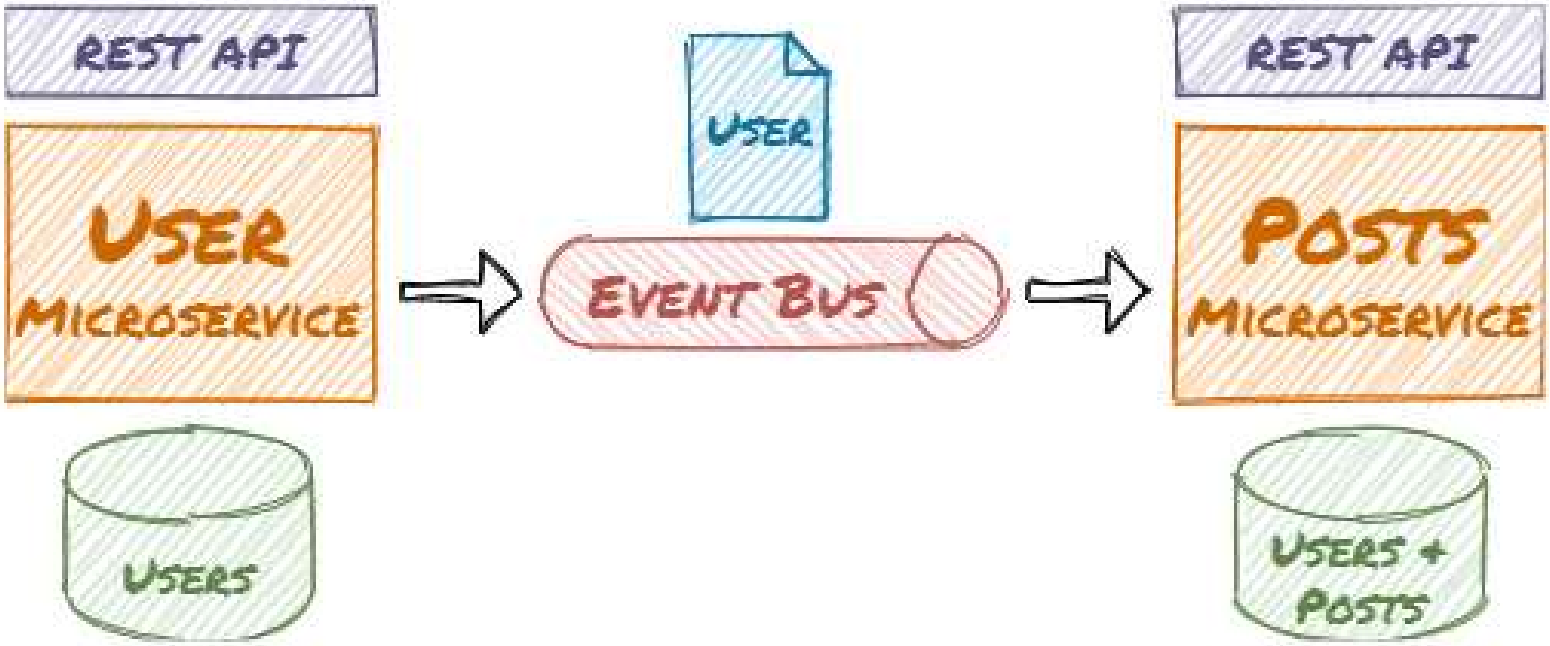
# Architecture Microservices : Application Gateways / API Gateways / Events Bus



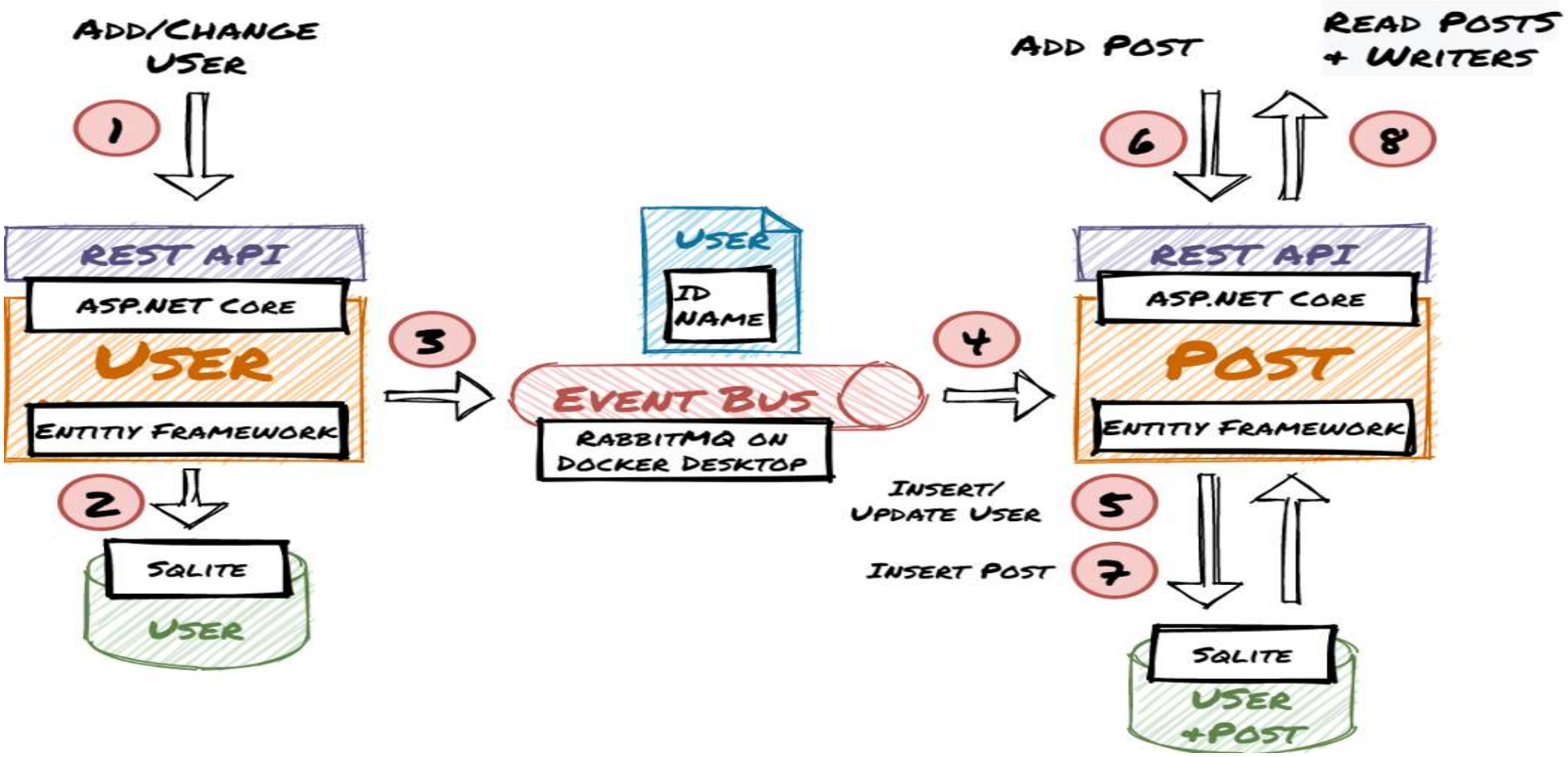
# Architecture Microservices : Application Gateways / API Gateways / Events Bus



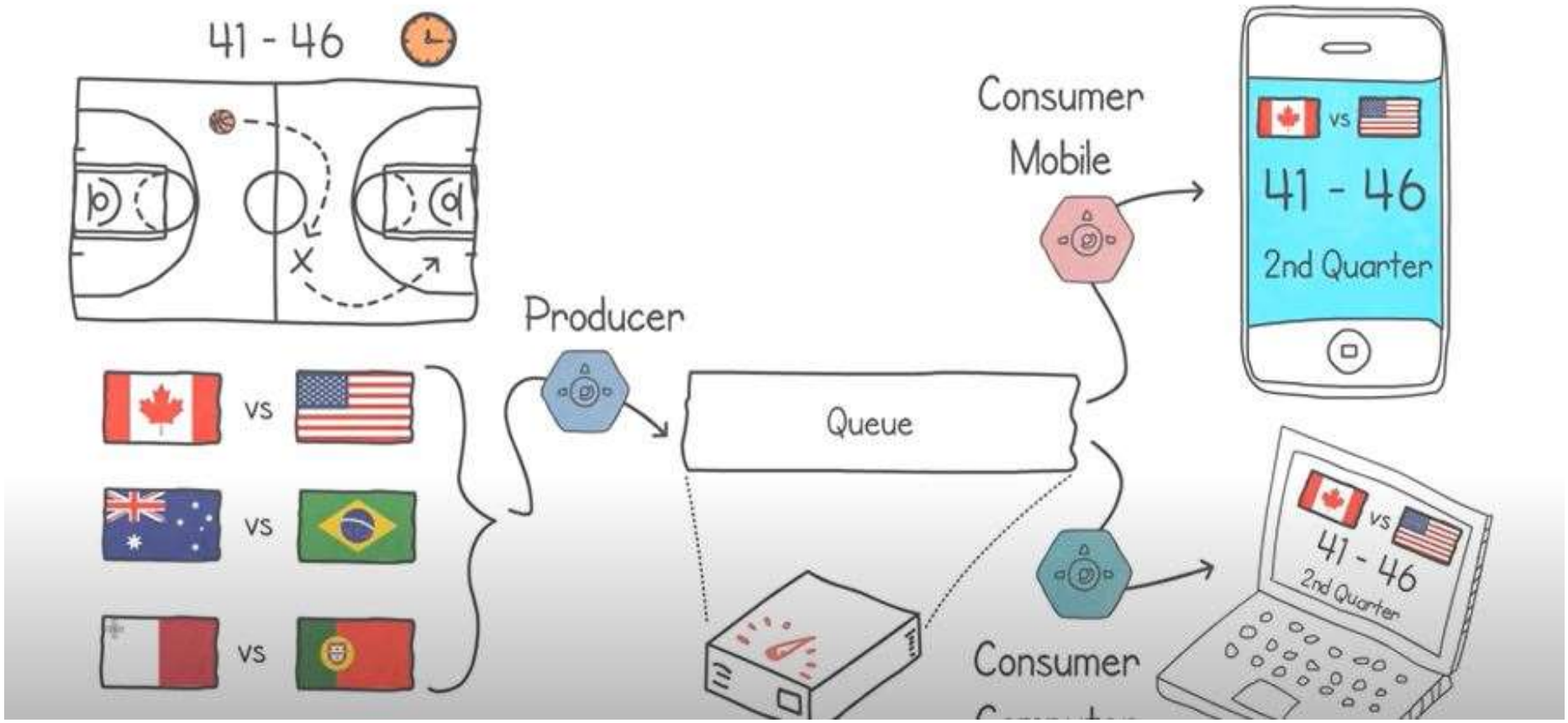
## Architecture Microservices Events



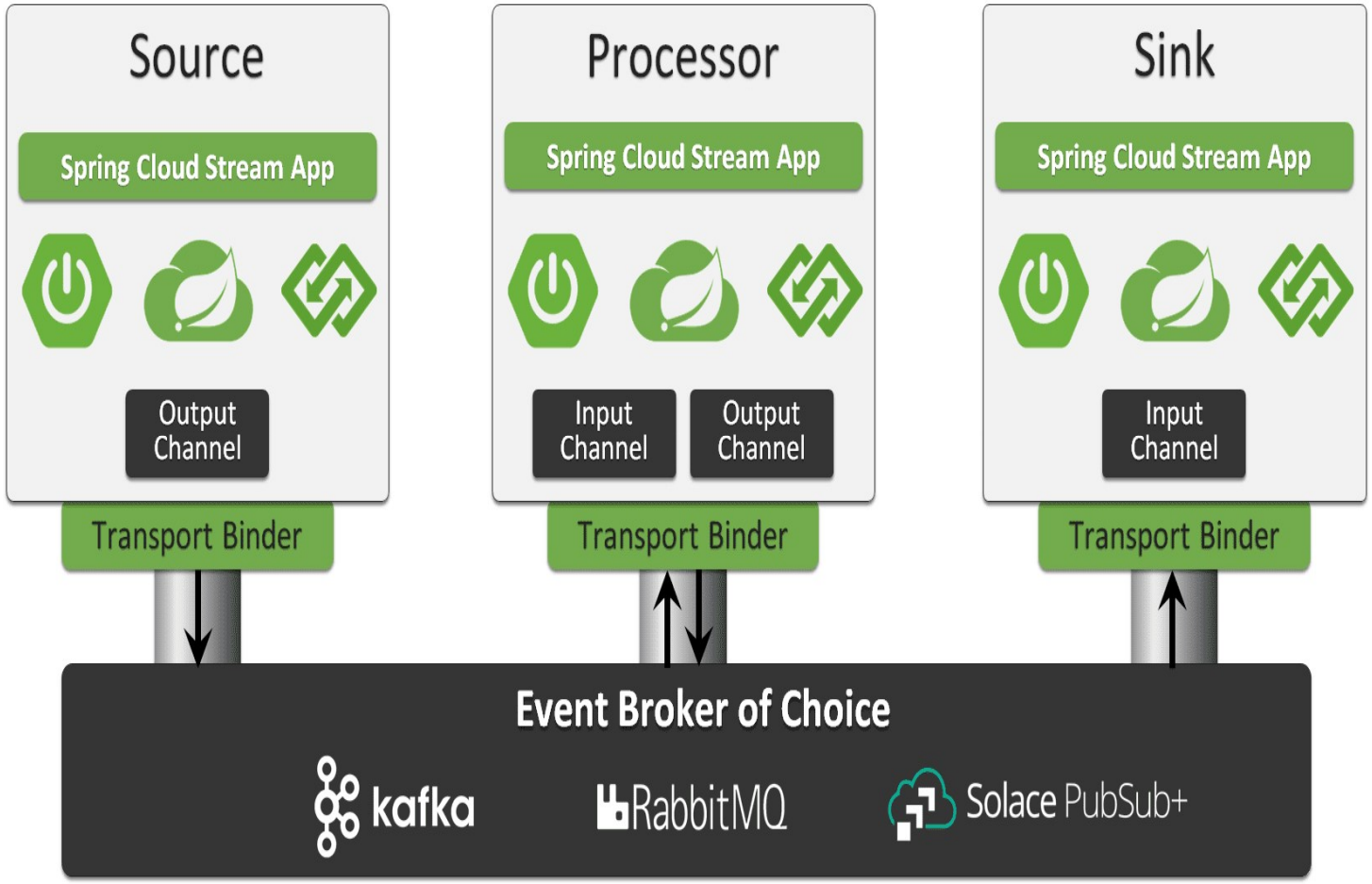
## Architecture Microservices Events



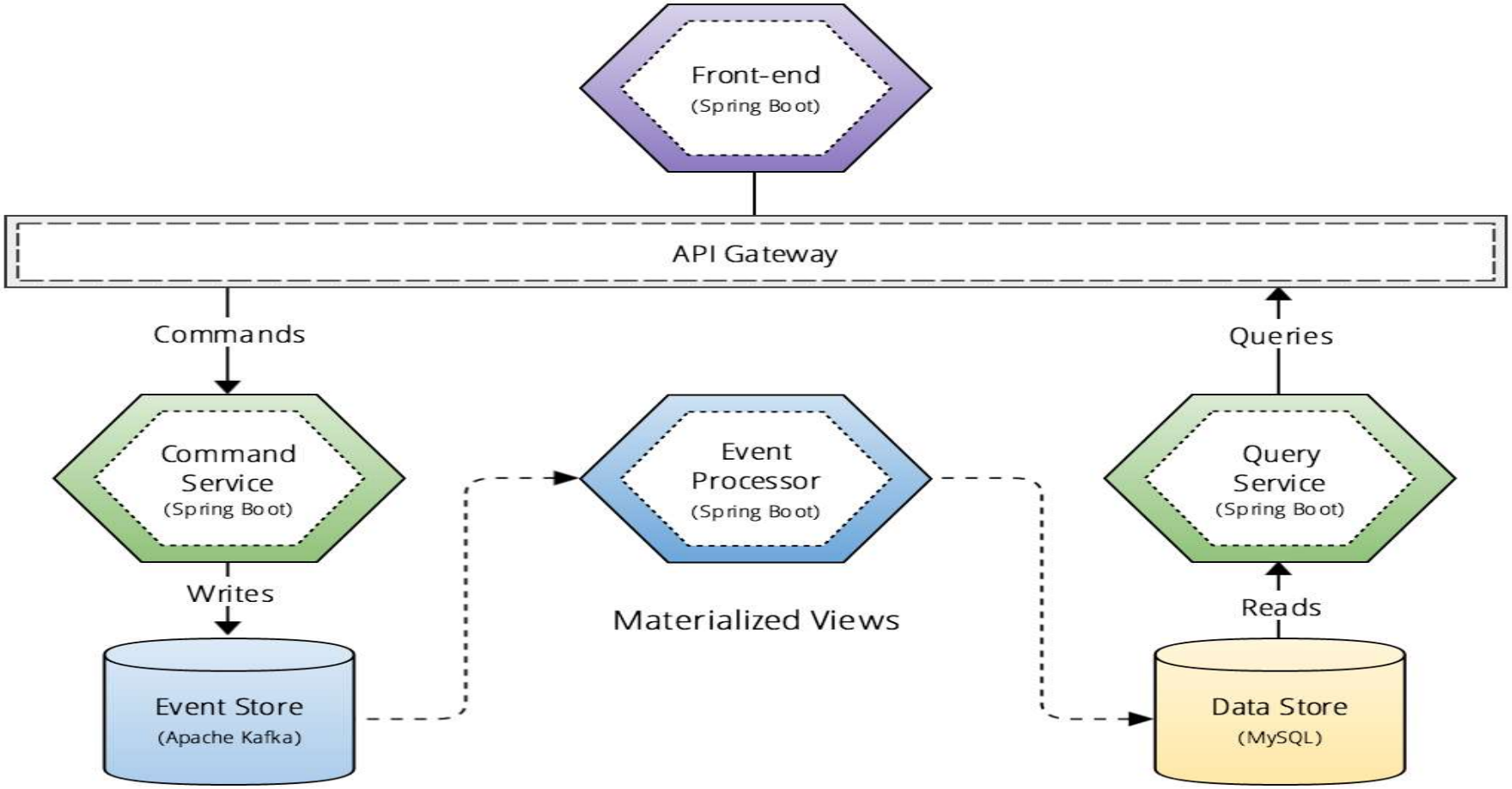
# Architecture Microservices Events with KAFKA



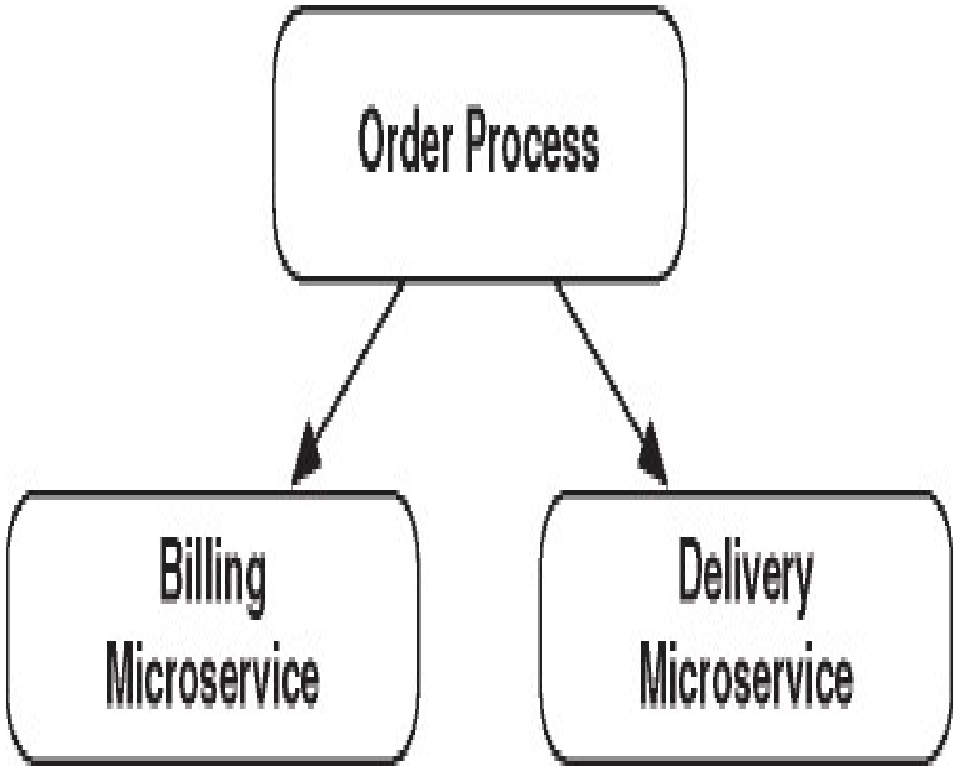
## Architecture Microservices Events

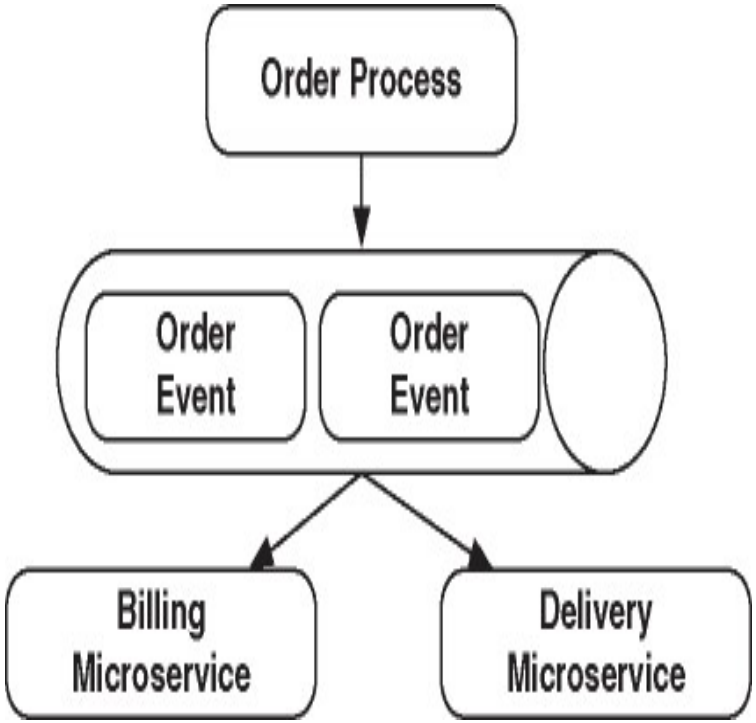


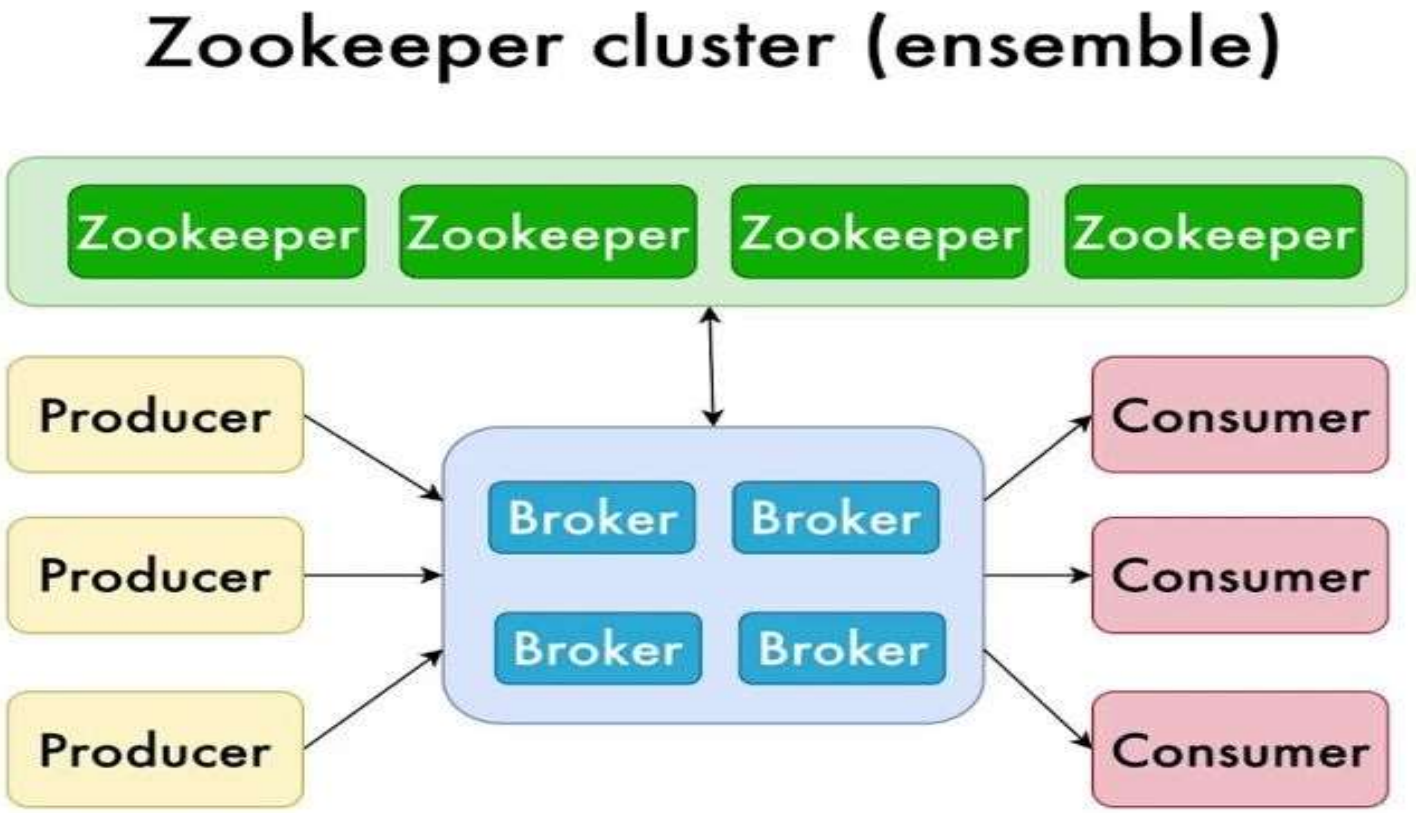
# Architecture Microservices Events



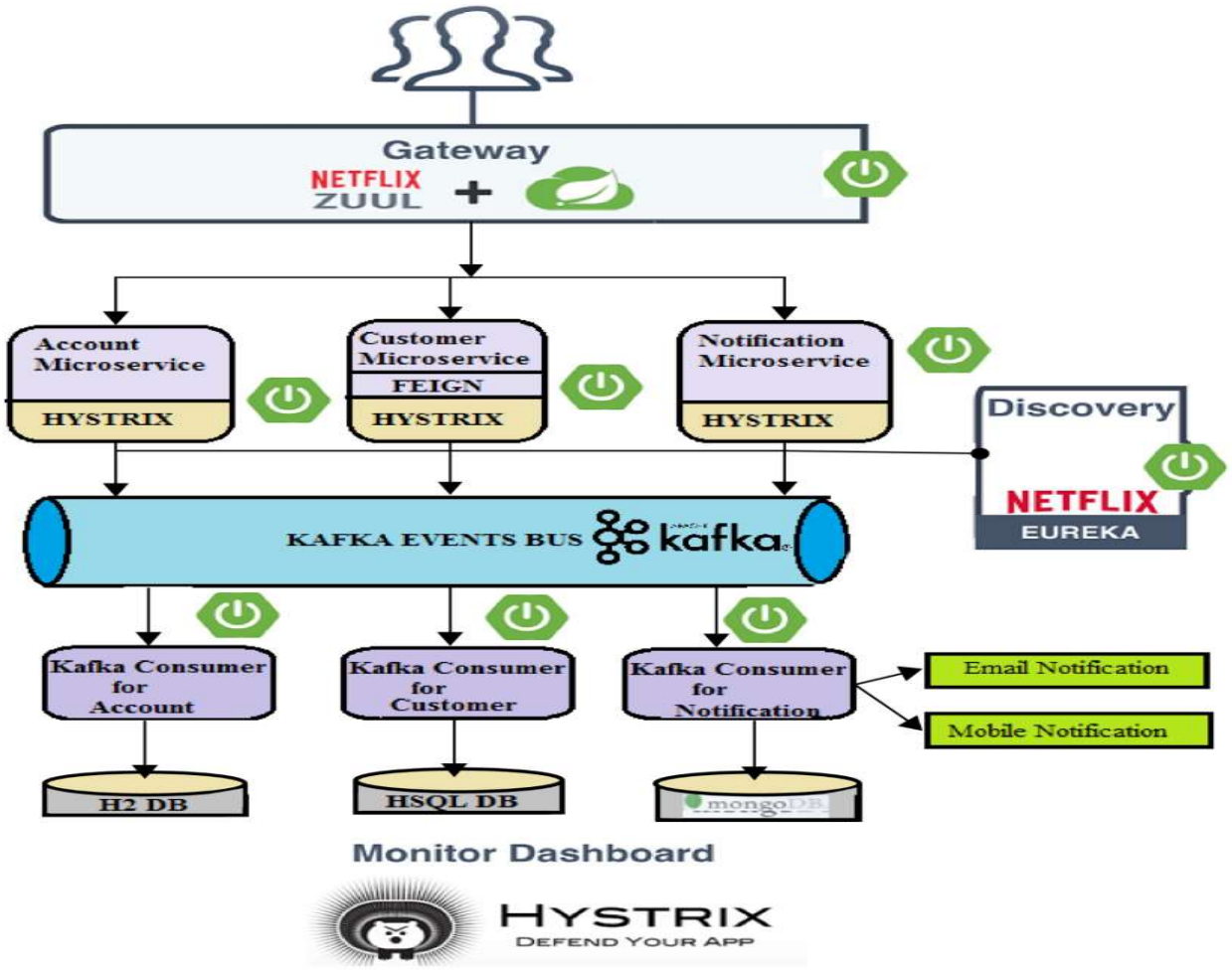
## Architecture Microservices Events



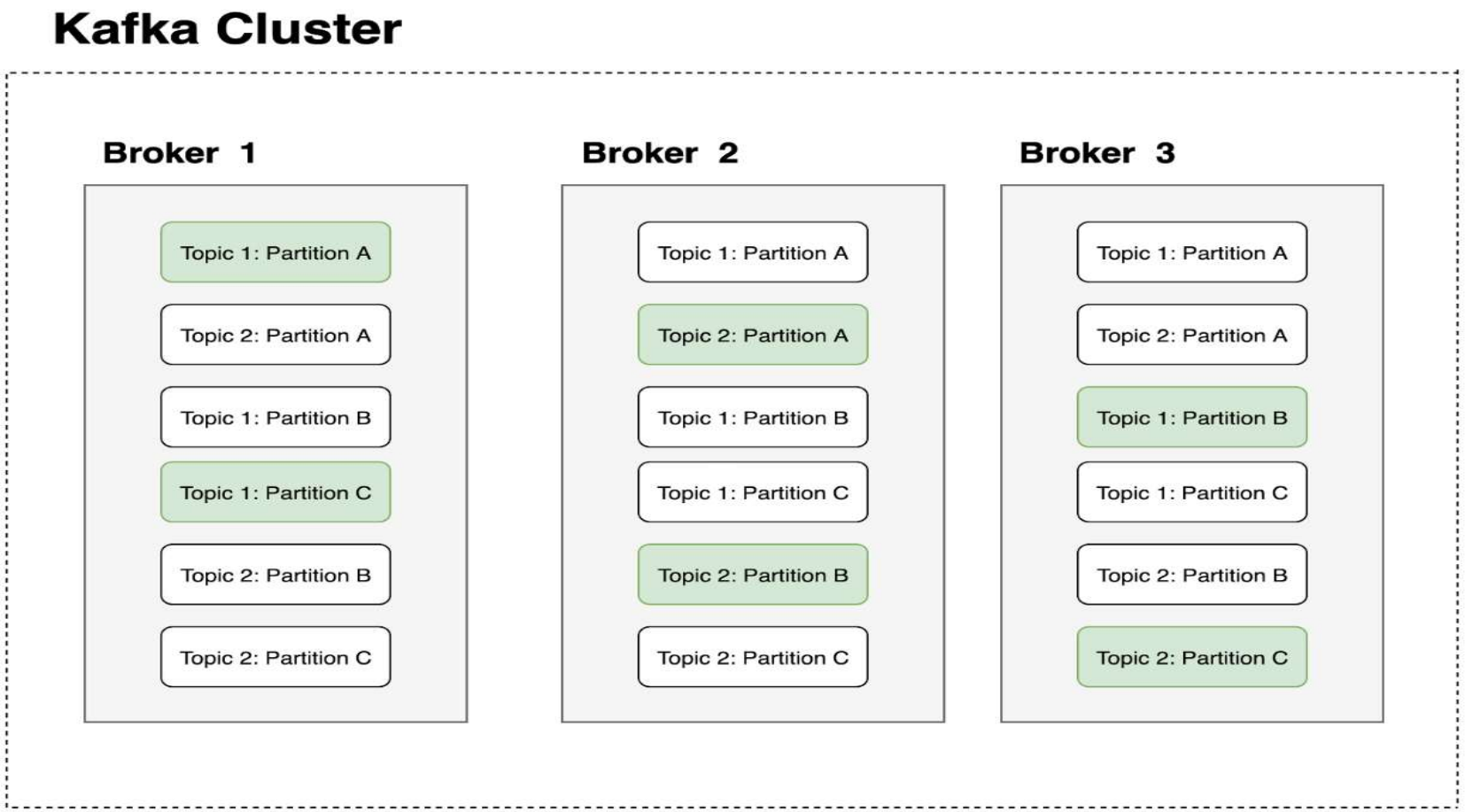




Zookeeper cluster (ensemble)
Zookeeper
Zookeeper
Zookeeper
Zookeeper
Producer
Producer
Producer
Broker
Broker
Broker
Broker
Consumer
Consumer
Consumer

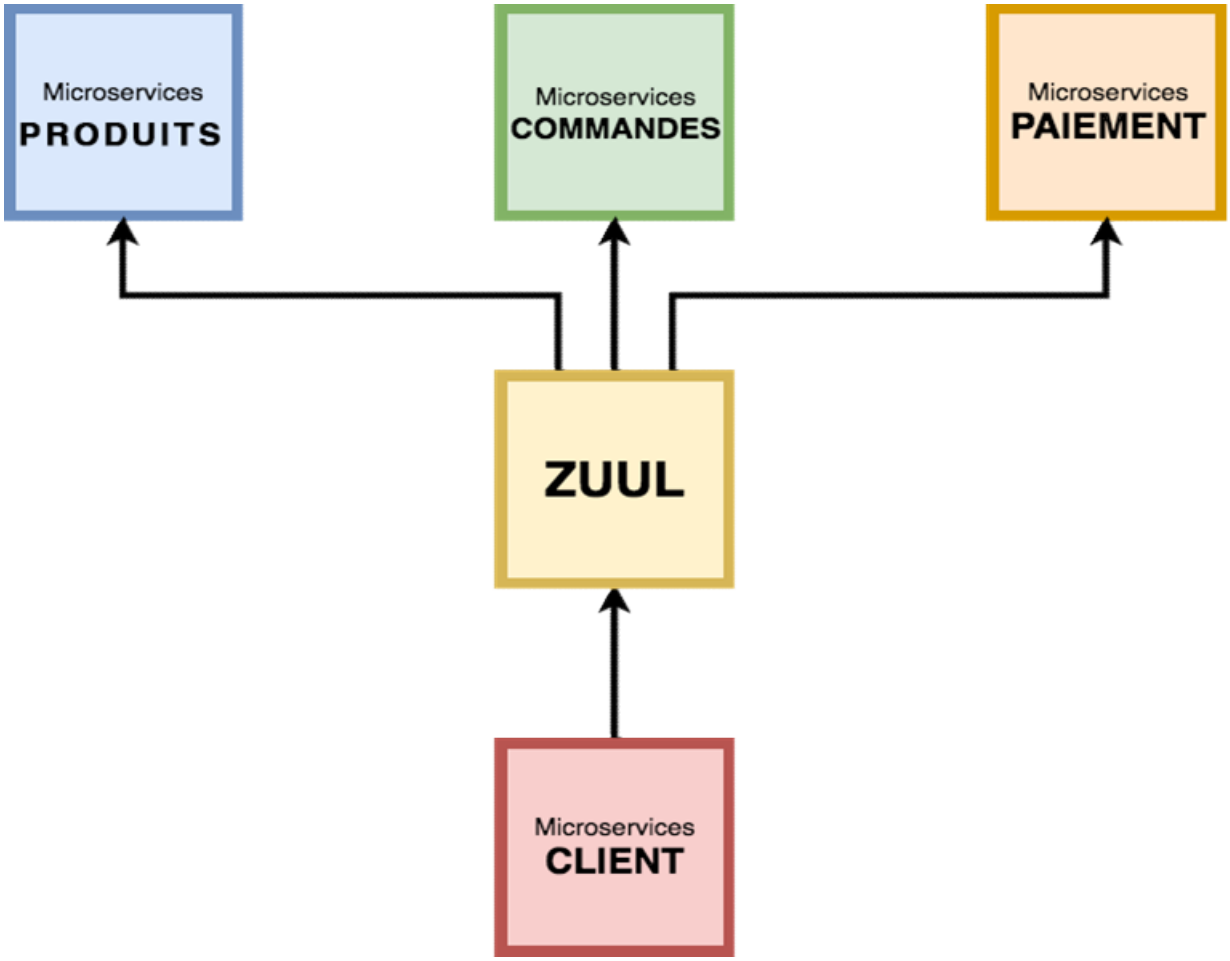
# Architecture Microservices Events with KAFKA



# Architecture Microservices Events with KAFKA

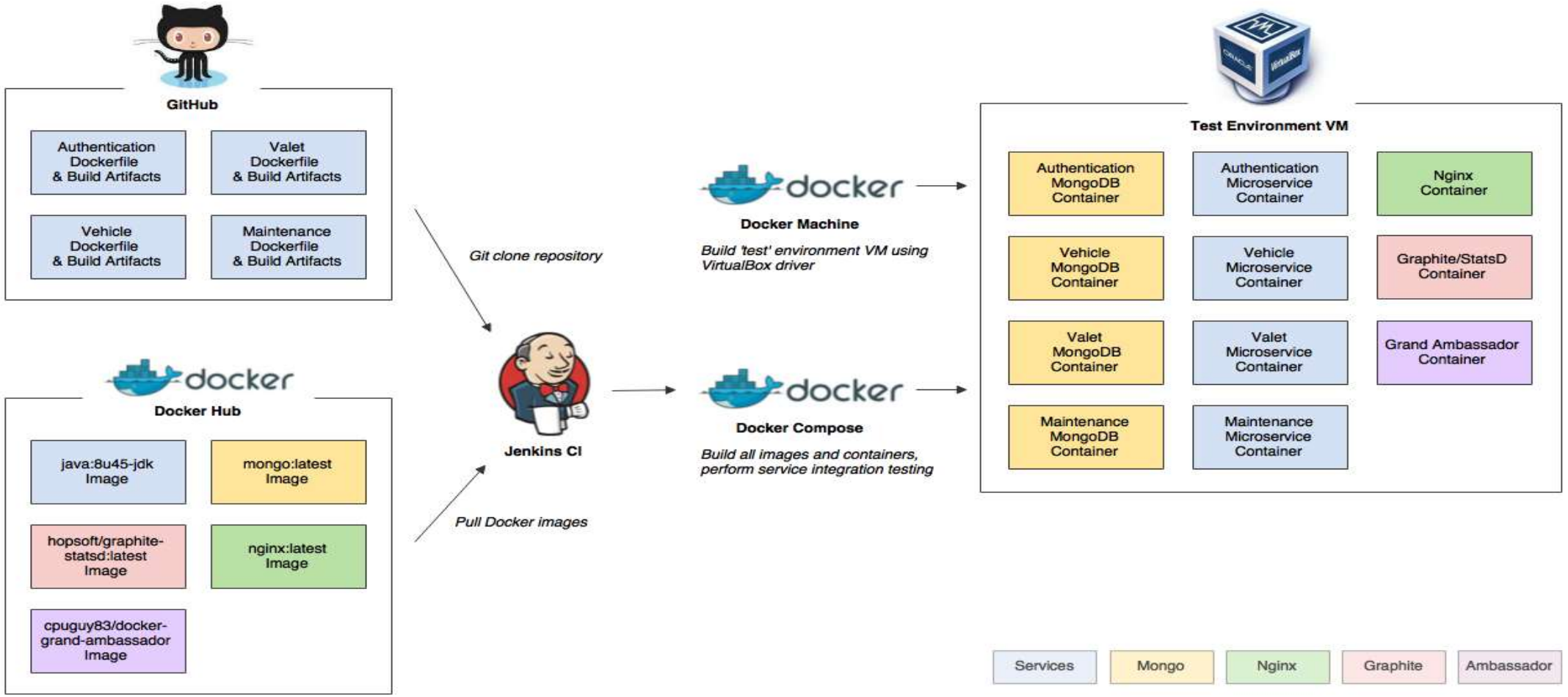


## Avec API GATEWAY

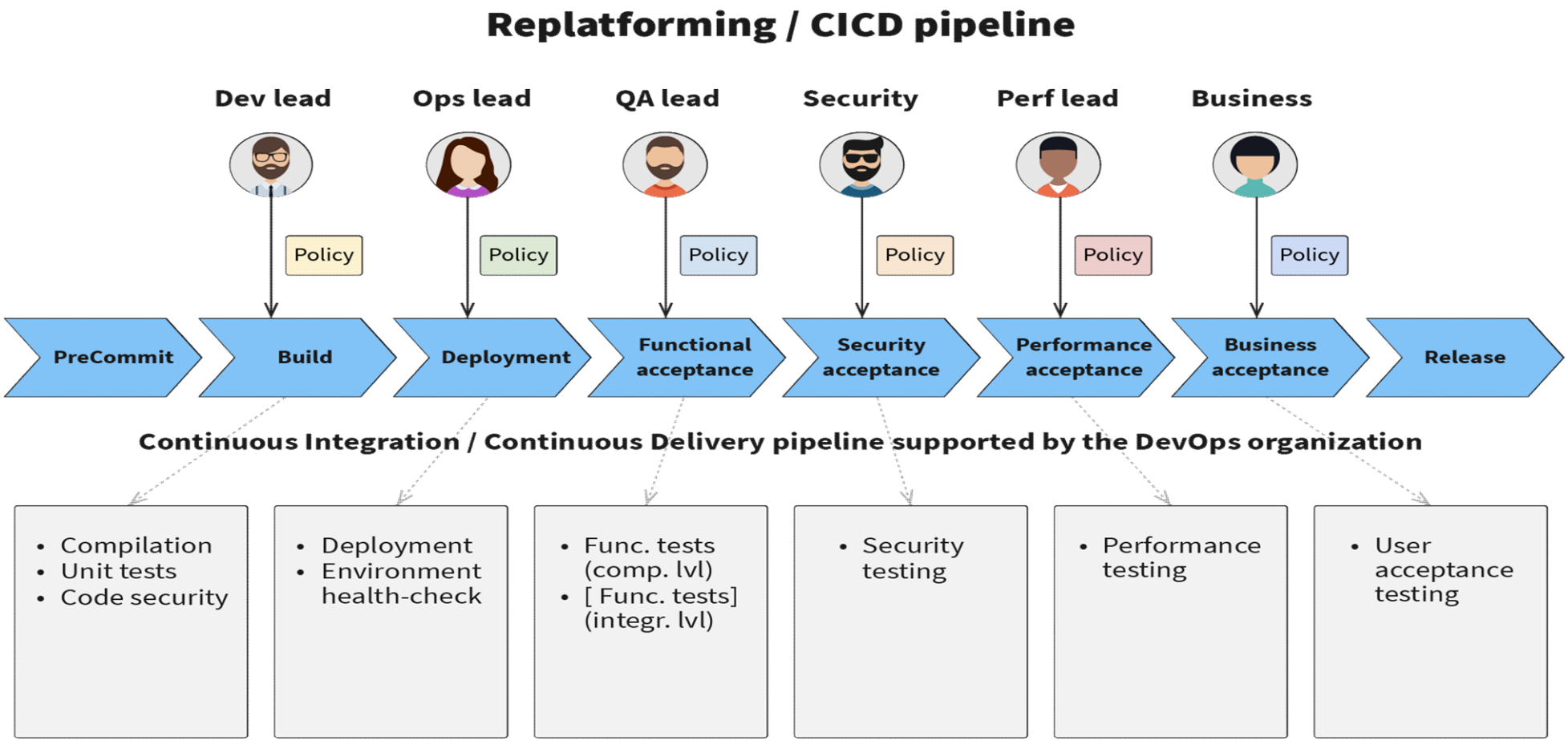


# 5/ DevOps

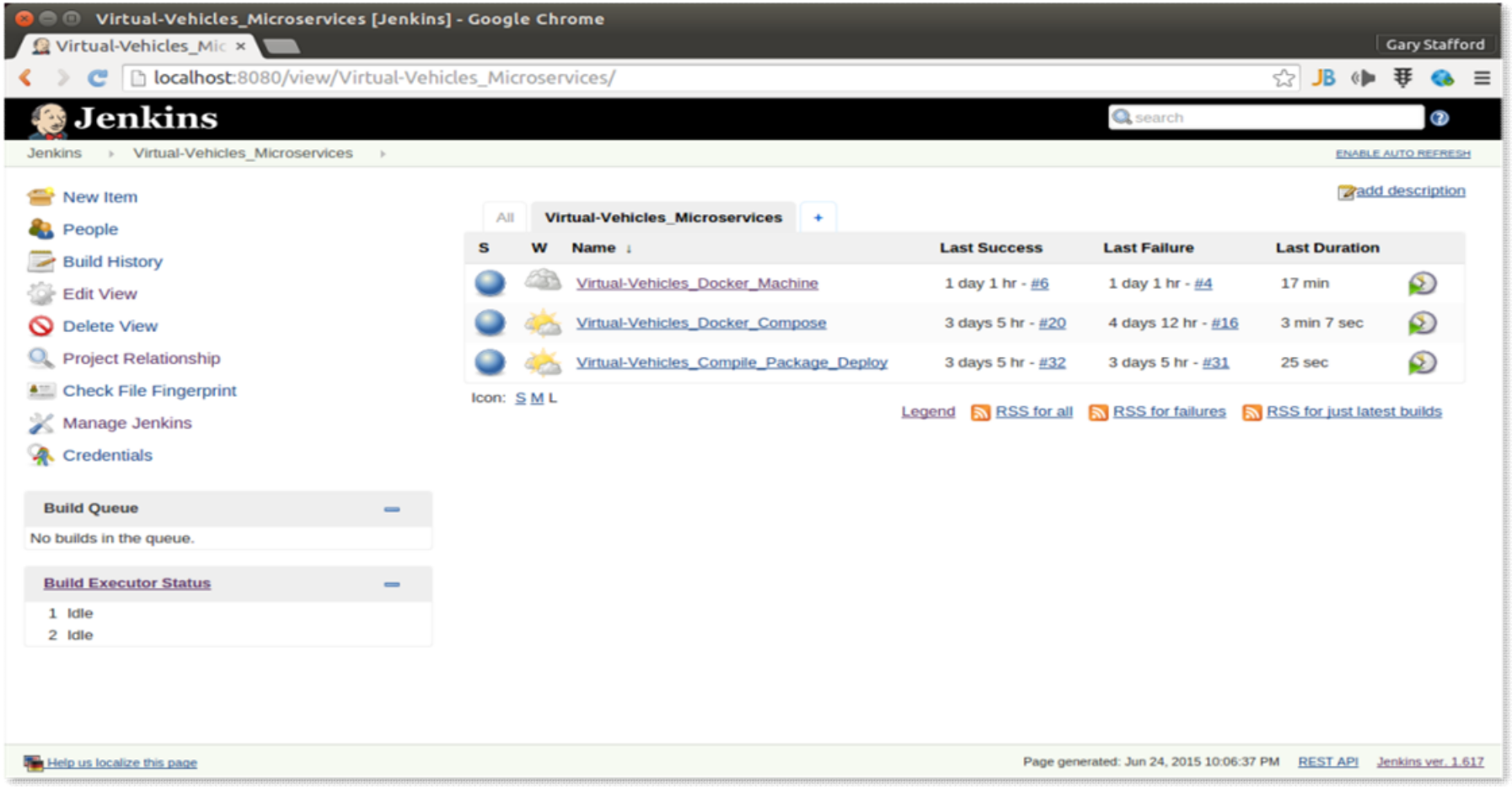
# DevOps: Docker compose



# DevOps: CI/CD Pipeline



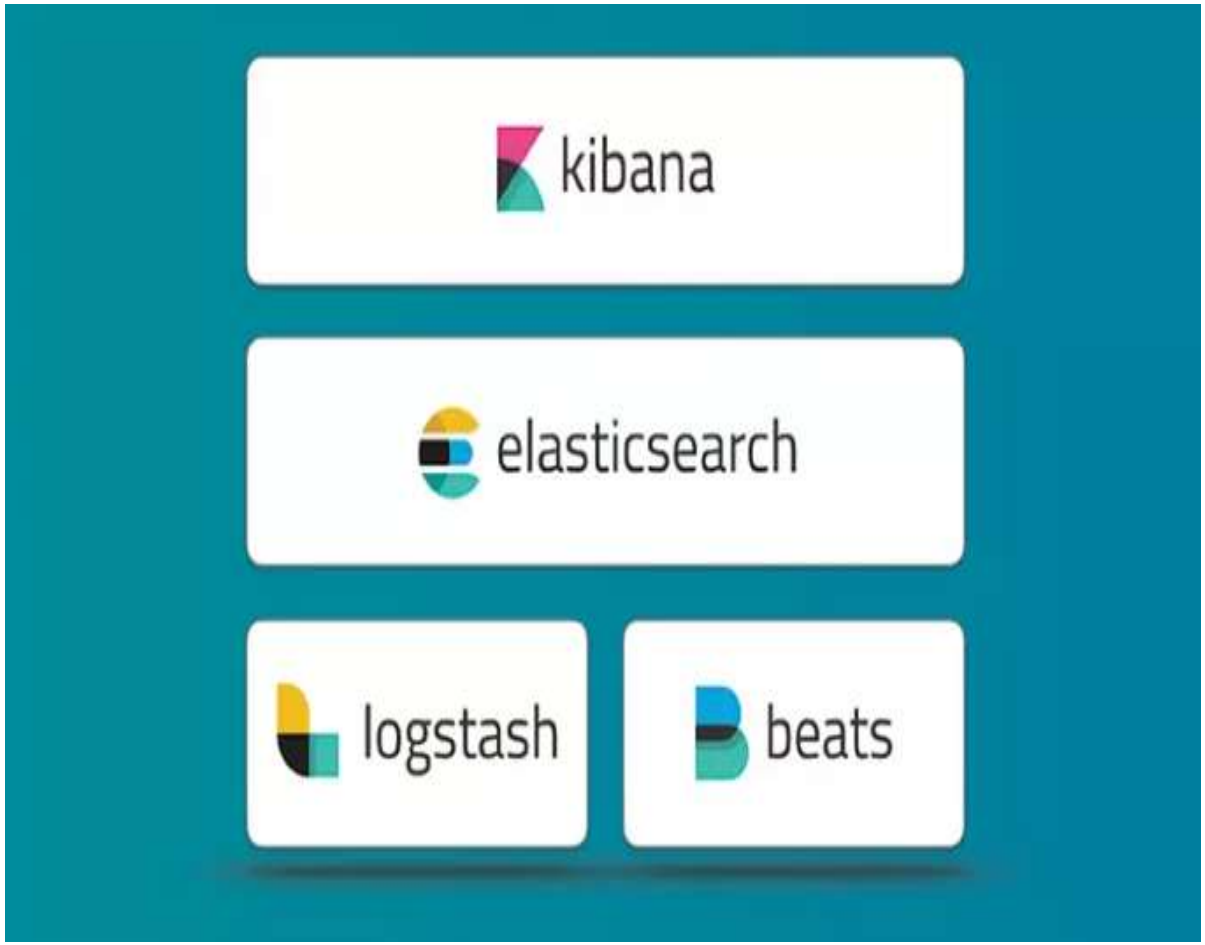
# PIC: Jenkins

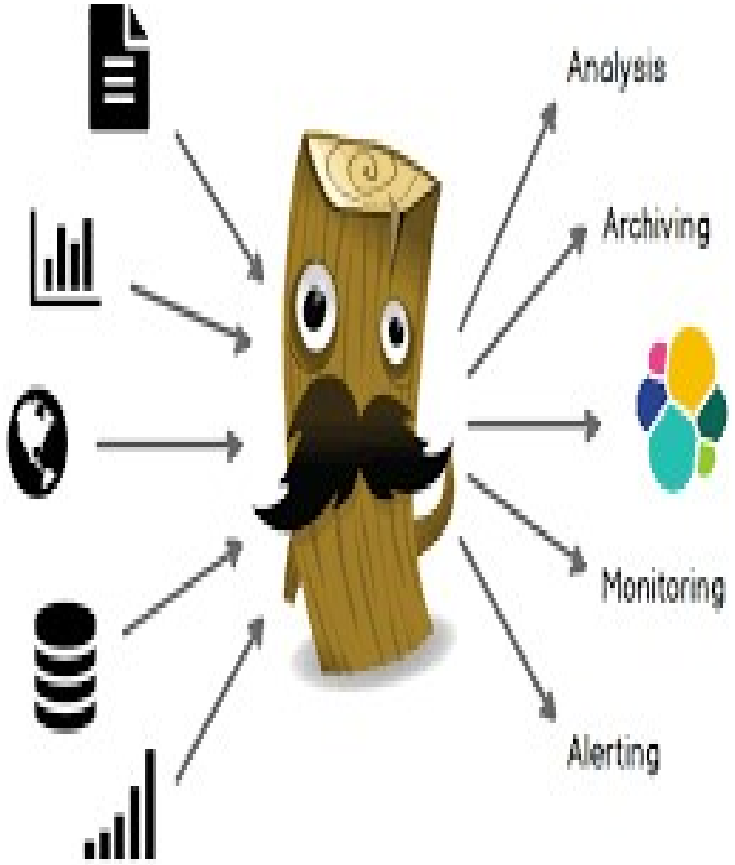

Virtual-Vehicles_Microservices [Jenkins] - Google Chrome
Virtual-Vehicles_Mic
Gary Stafford
localhost:8080/view/Virtual-Vehicles_Microservices/
Jenkins
search
Jenkins
Virtual-Vehicles_Microservices
ENABLE AUTO REFRESH
New Item
People
Build History
Edit View
Delete View
Project Relationship
Check File Fingerprint
Manage Jenkins
Credentials
add description
All
Virtual-Vehicles_Microservices
S
W
Name
Last Success
Last Failure
Last Duration
Virtual-Vehicles_Docker_Machine
1 day 1 hr - #6
1 day 1 hr - #4
17 min
Virtual-Vehicles_Docker_Compose
3 days 5 hr - #20
4 days 12 hr - #16
3 min 7 sec
Virtual-Vehicles_Compile_Package_Deploy
3 days 5 hr - #32
3 days 5 hr - #31
25 sec
Icon: S M L
Legend
RSS for all
RSS for failures
RSS for just latest builds
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Help us localize this page
Page generated: Jun 24, 2015 10:06:37 PM
REST API
Jenkins ver. 1.617

## PIC: Jenkins

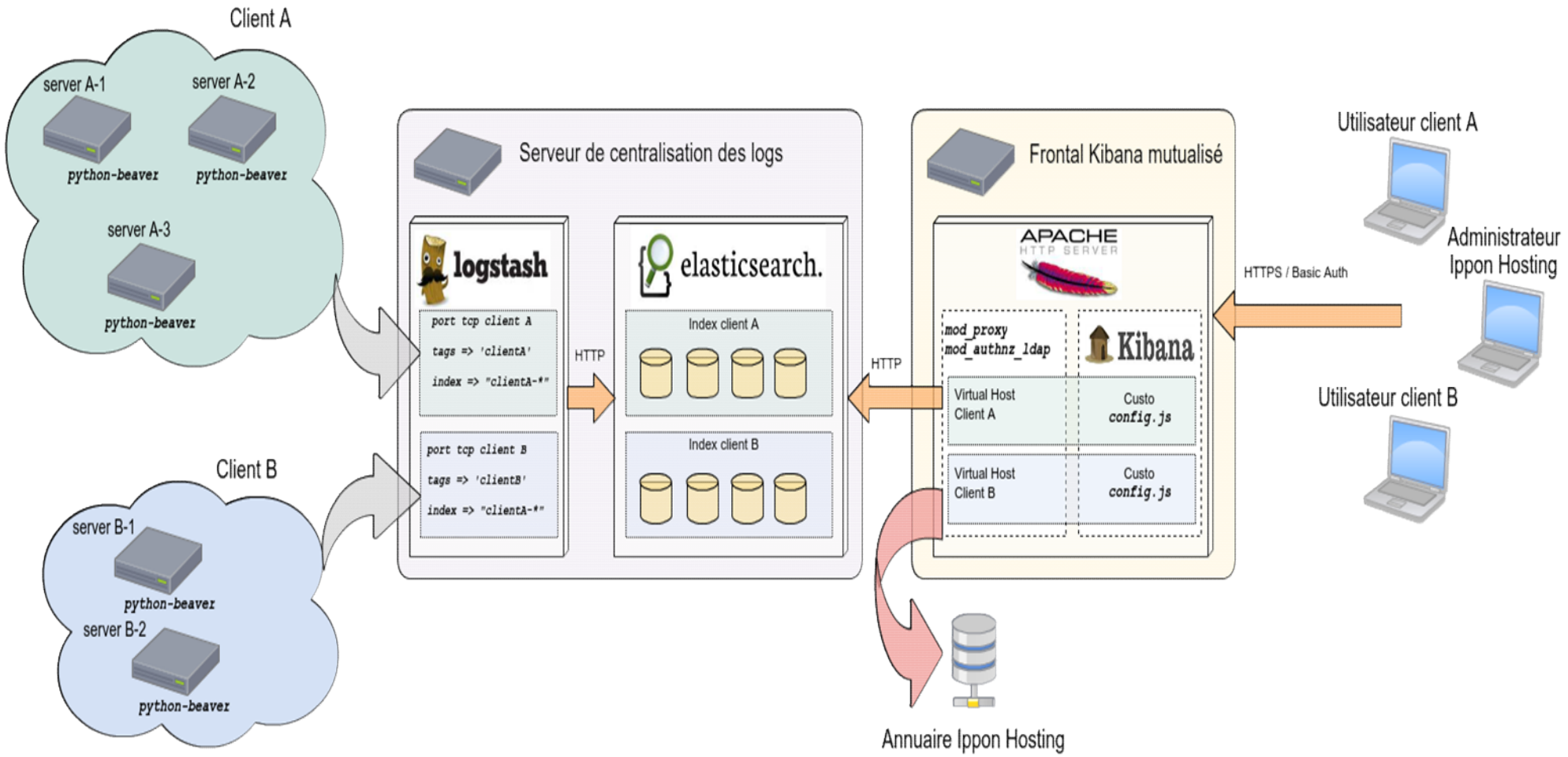
```
CONTAINER ID    IMAGE                                 NAME
============    =====                                 ====
17992acc6542    jenkins_nginx                         jenkins_nginx_1
bcbb2a4b1a7d    jenkins_vehicle                       jenkins_vehicle_1
4ac1ac69f230    mongo:latest                          jenkins_mongoVehicle_1
bcc8b9454103    jenkins_valet                         jenkins_valet_1
7c1794ca7b8c    jenkins_maintenance                   jenkins_maintenance_1
2d0e117fa5fb    jenkins_authentication                jenkins_authentication_1
d9146a1b1d89    hopsoft/graphite-statsd:latest        jenkins_graphite_1
56b34cee9cf3    cpuguy83/docker-grand-ambassador      jenkins_ambassador_1
a72199d51851    mongo:latest                          jenkins_mongoAuthentication_1
307cb2c01cc4    mongo:latest                          jenkins_mongoMaintenance_1
4e0807431479    mongo:latest                          jenkins_mongoValet_1
```

## PIC: ELK logs Centralisation and Alerting

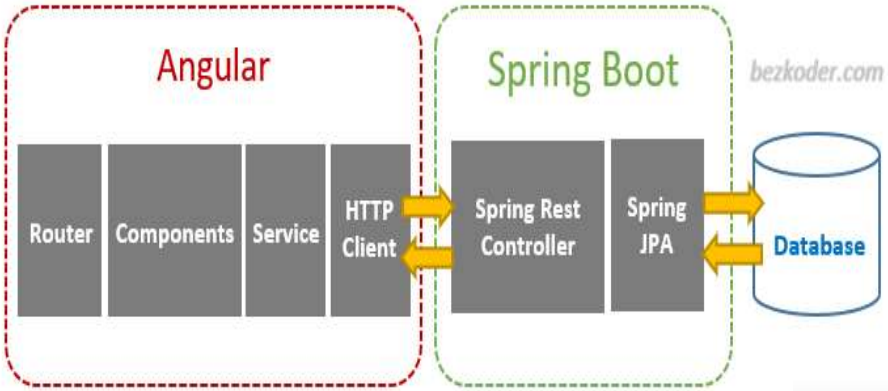


# PIC: ELK logs Centralisation and Alerting



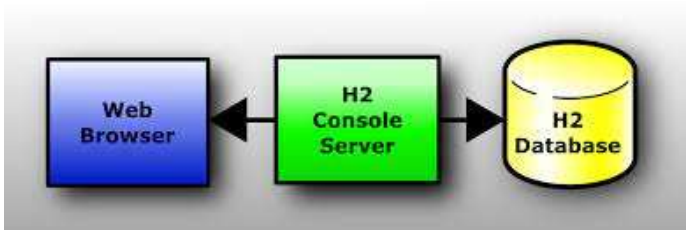
# 6/ TPs

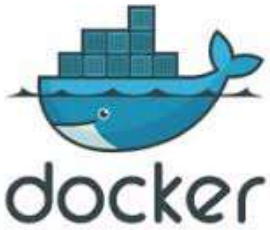

Angular
Spring Boot
bezkoder.com
Router
Components
Service
HTTP Client
Spring Rest Controller
Spring JPA
Database


APACHE
kafka


POSTMAN

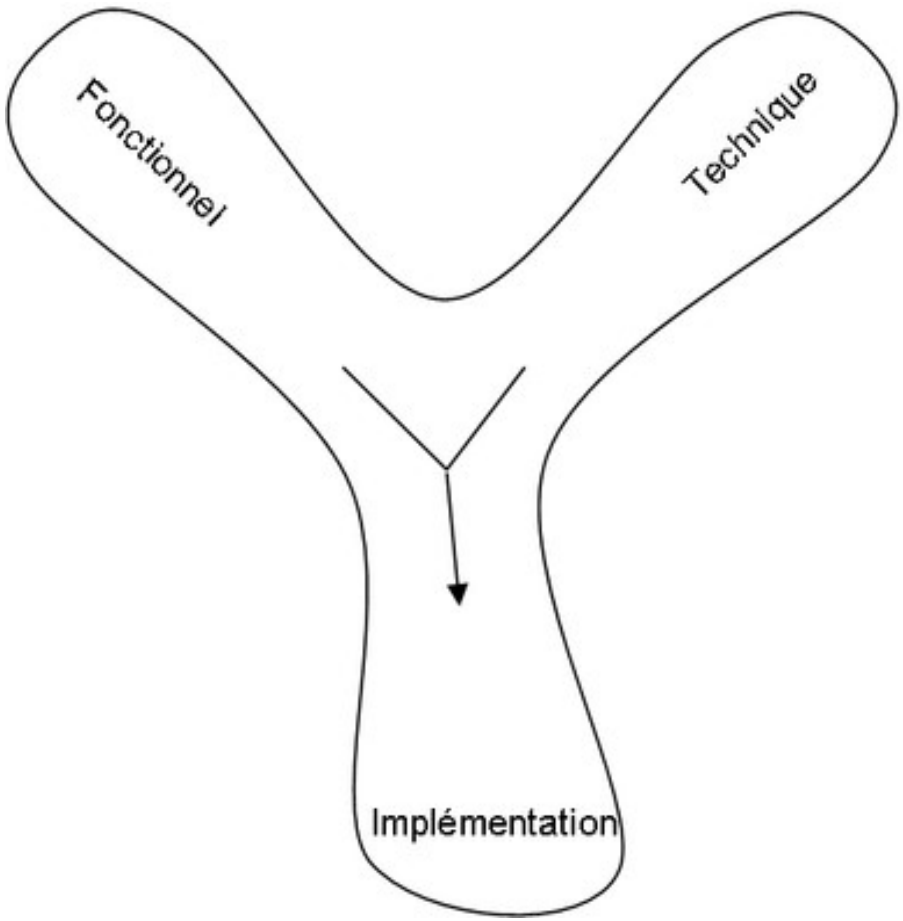

JHipster

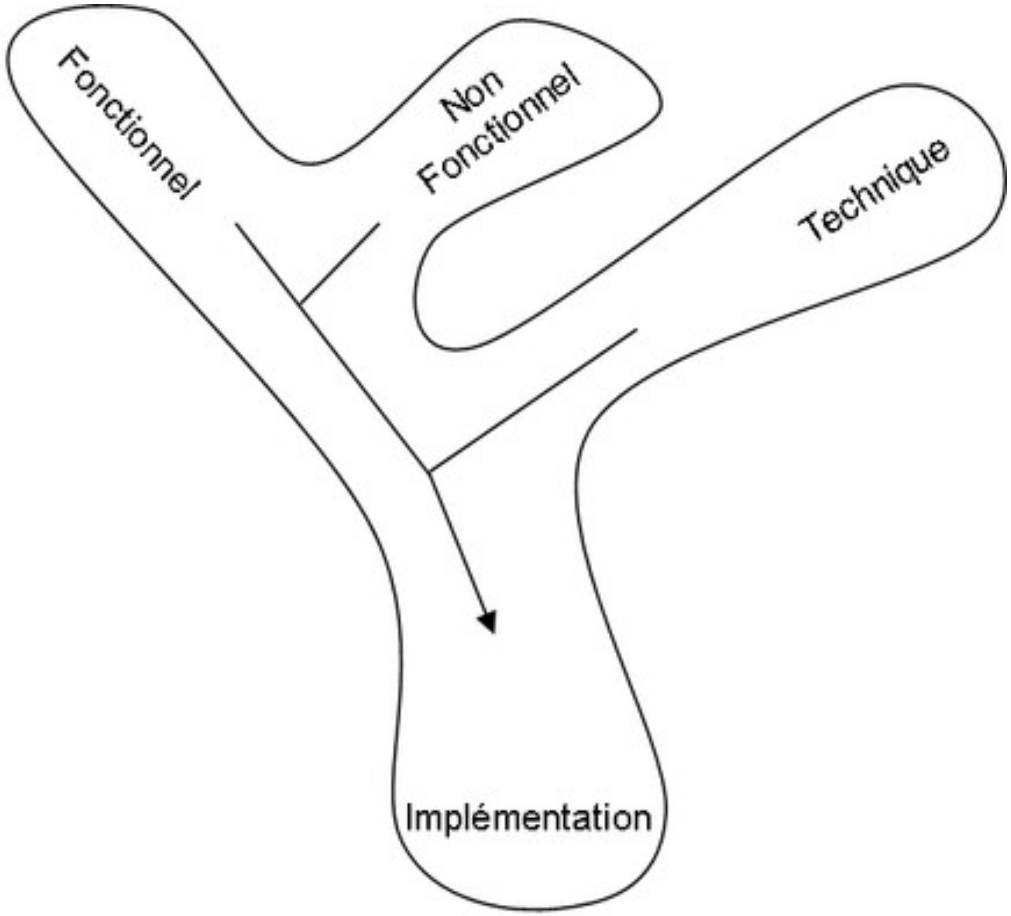

Web Browser
H2 Console Server
H2 Database


docker

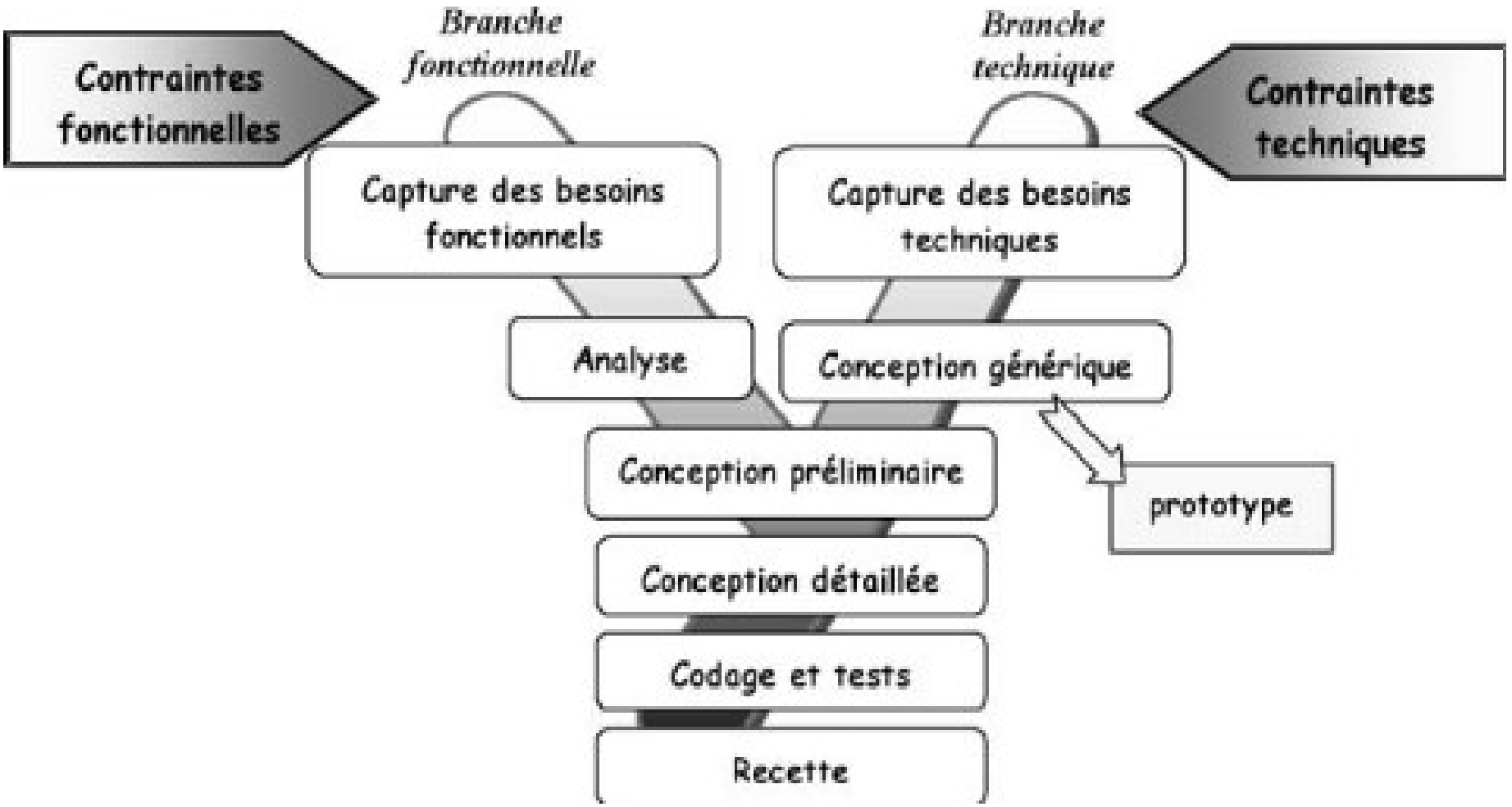
# 7/ Les Bonnes Pratiques

## Cycle en Y

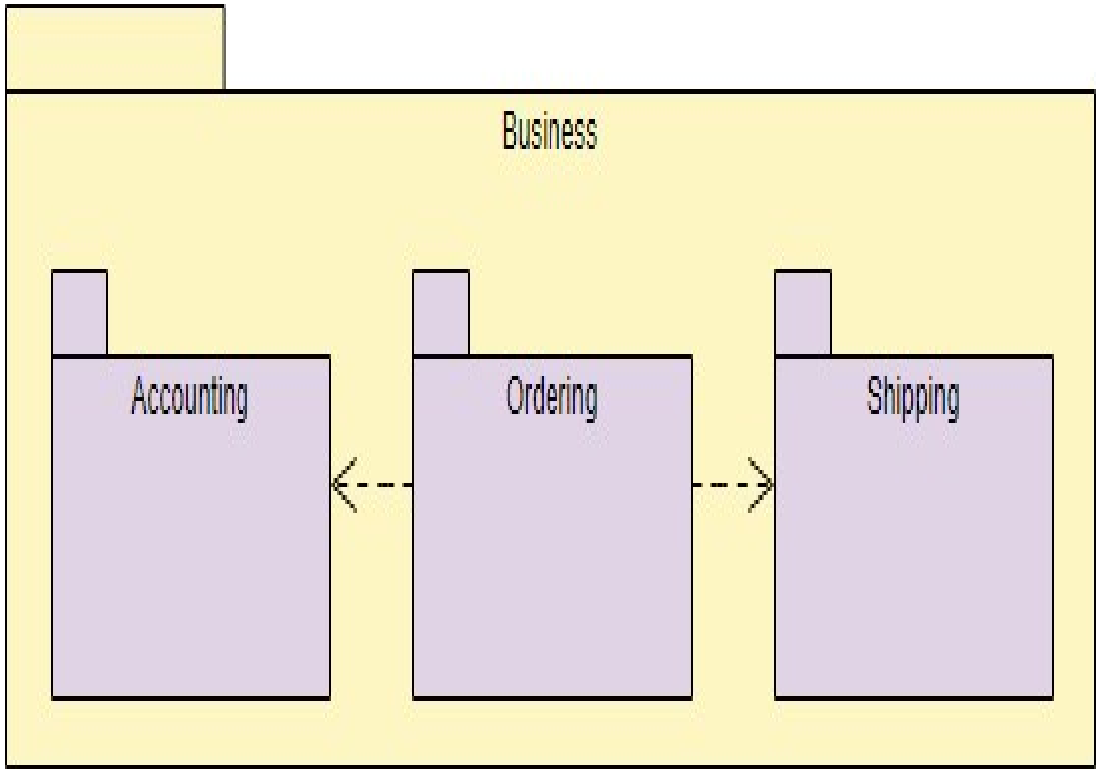

Fonctionnel
Technique
Implémentation

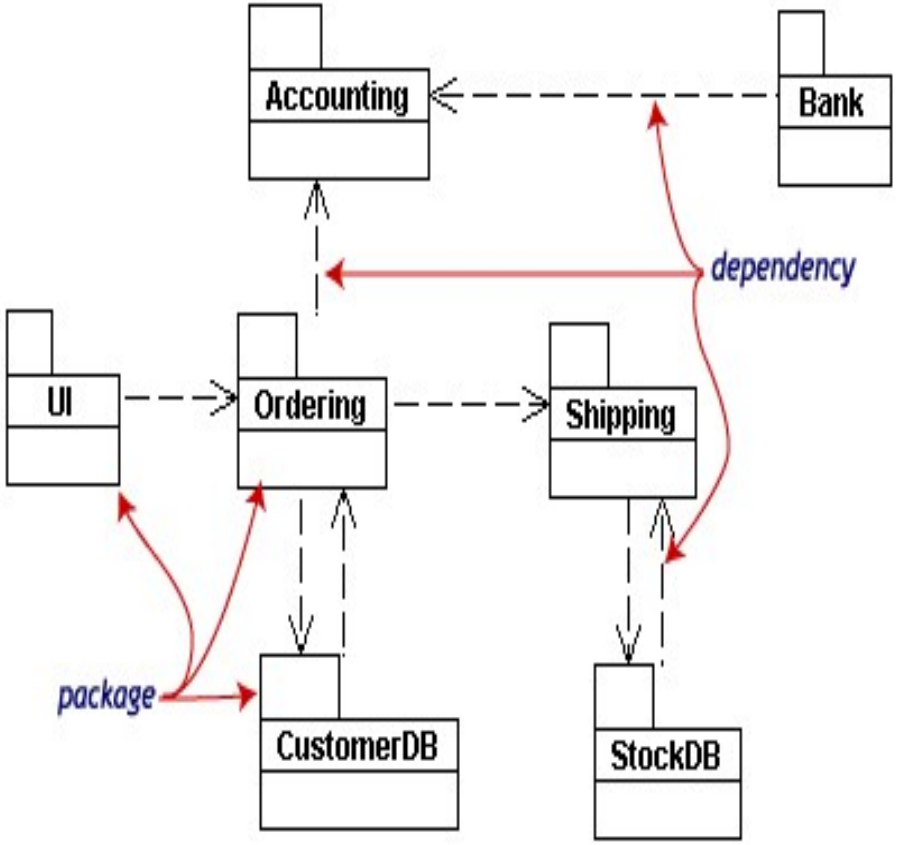

Fonctionnel
Non Fonctionnel
Technique
Implémentation

## Cycle en Y


Branche fonctionnelle
Branche technique
Contraintes fonctionnelles
Contraintes techniques
Capture des besoins fonctionnels
Capture des besoins techniques
Analyse
Conception générique
Conception préliminaire
prototype
Conception détaillée
Codage et tests
Recette

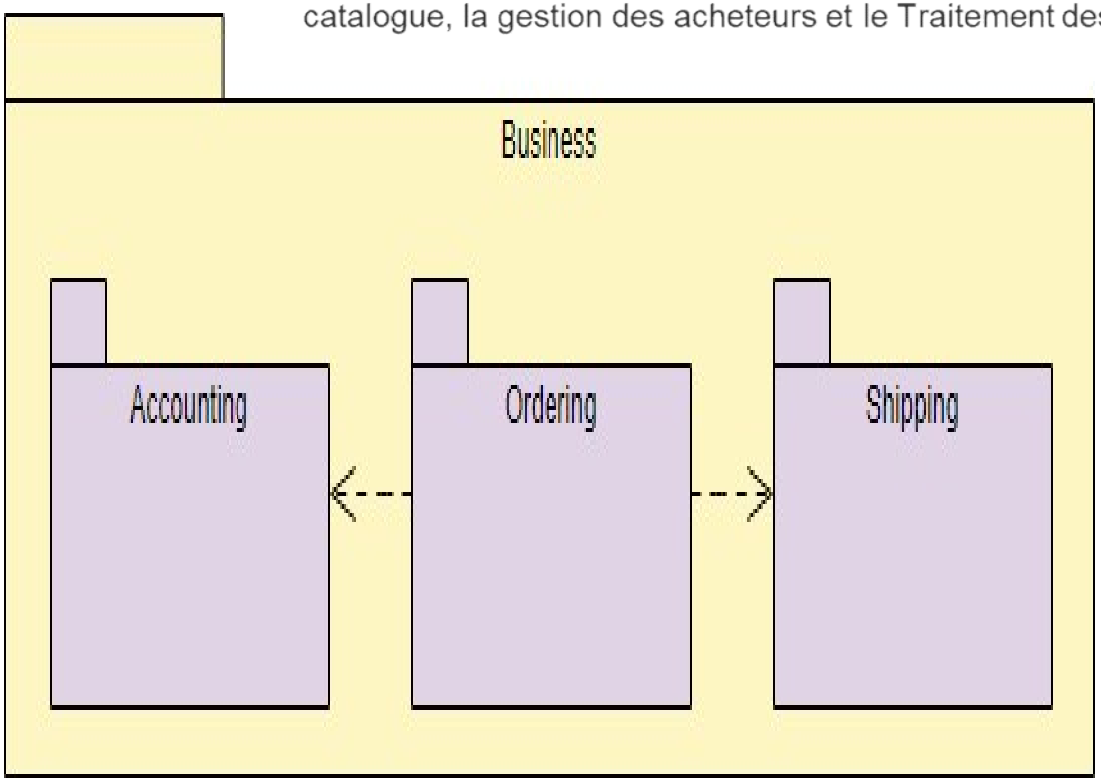
## Architecture Fonctionnelle / Processus Métier



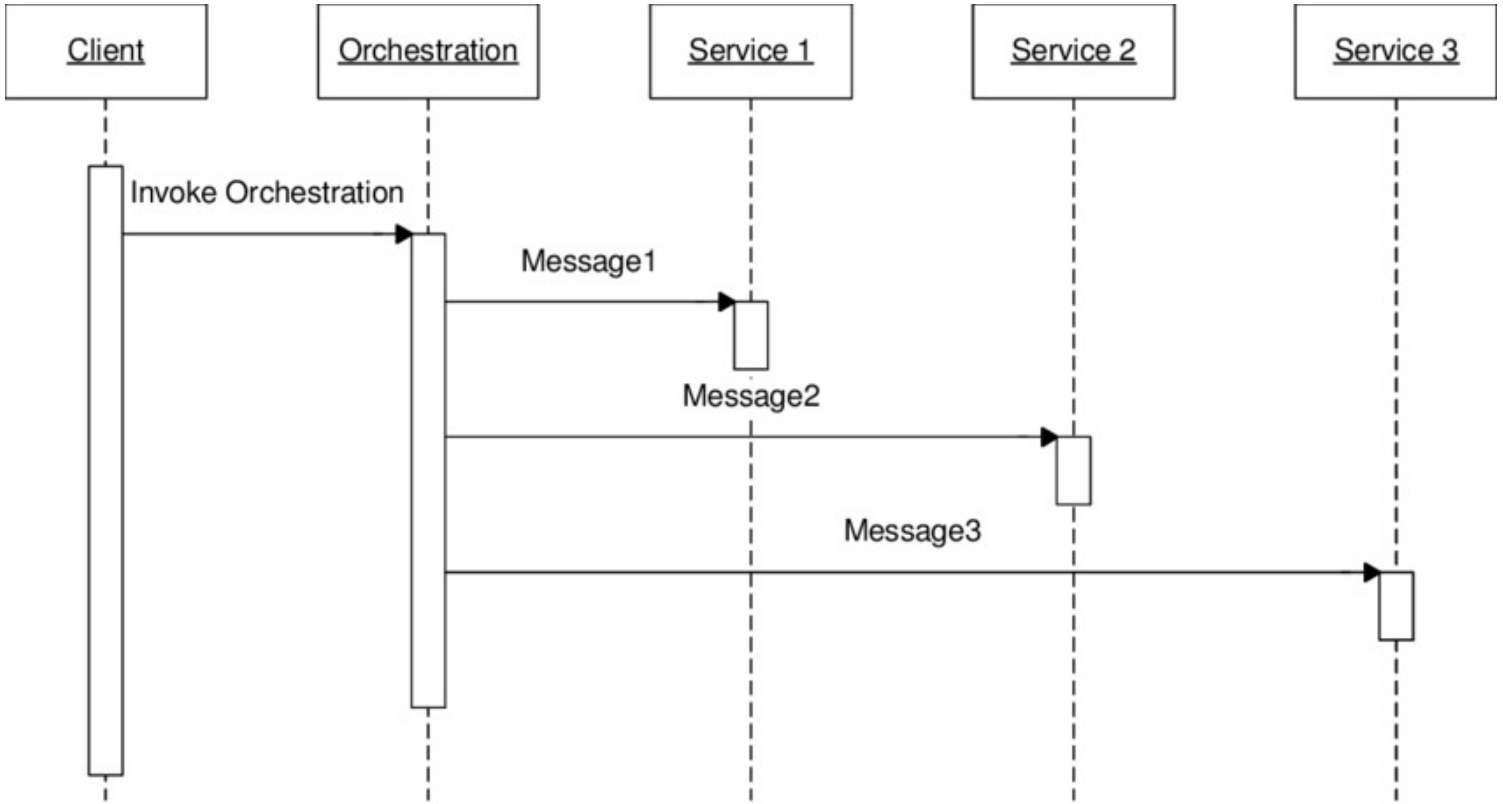



## Architecture Fonctionnelle / Processus Métier

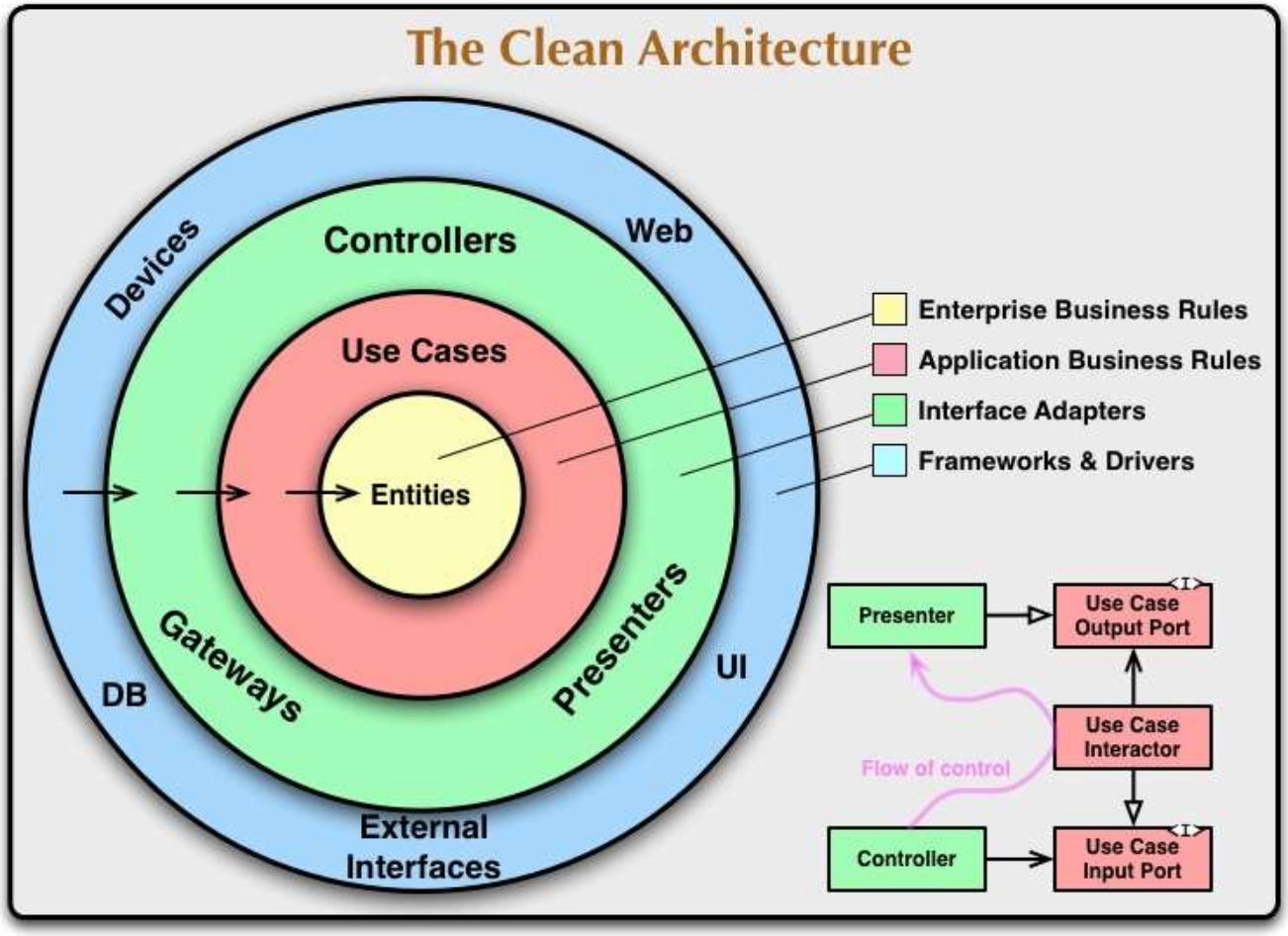
Dans l'architecture Microservices, **chaque service est conçu pour remplir et implémenter l'un des sous-domaines** du domaine de l'application. Par exemple, dans une application de commerce électronique, au lieu de créer des services basés sur les couches et les niveaux dans lesquels ils existent, nous créons les services sur les sous-domaines au sein de l'espace de domaine du commerce électronique comme le panier, le catalogue, la gestion des acheteurs et le Traitement des commandes, etc.



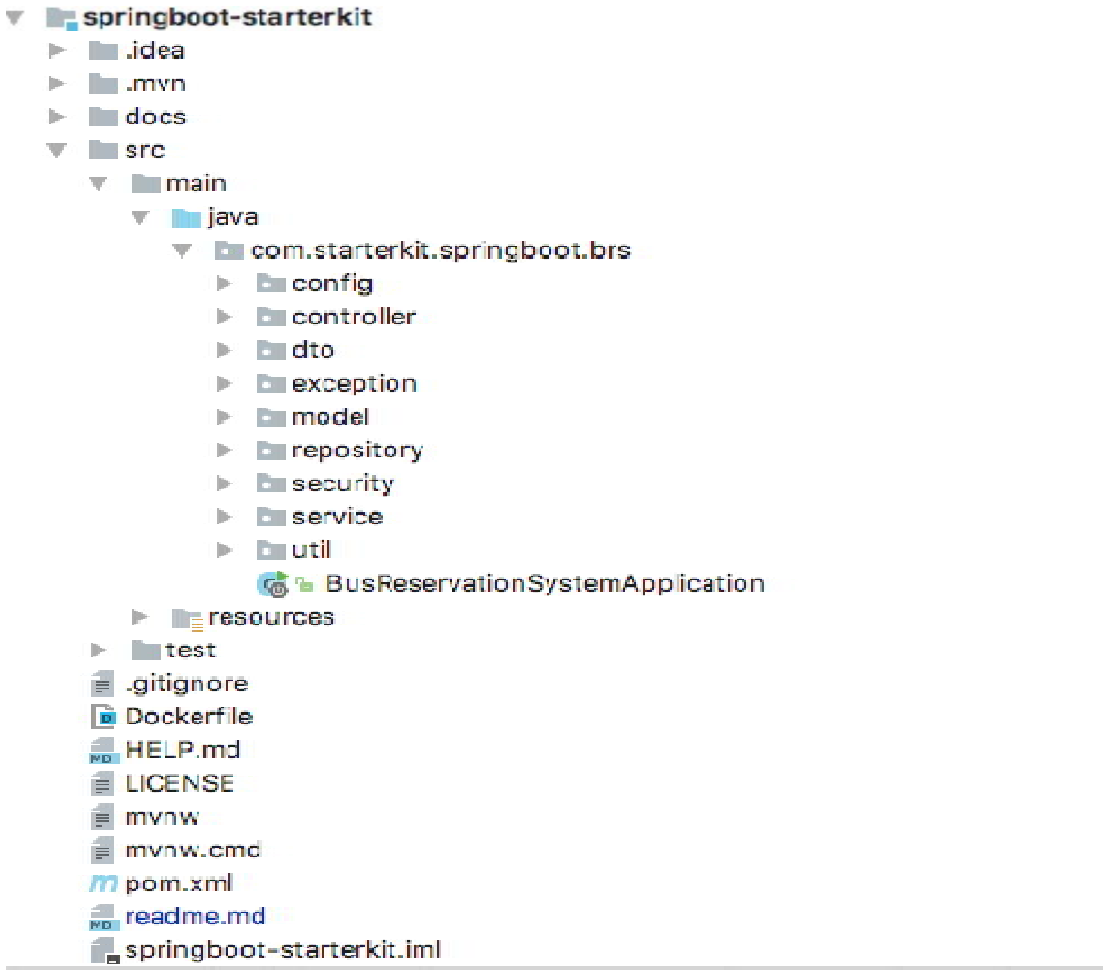
## Architecture Fonctionnelle / Processus Métier



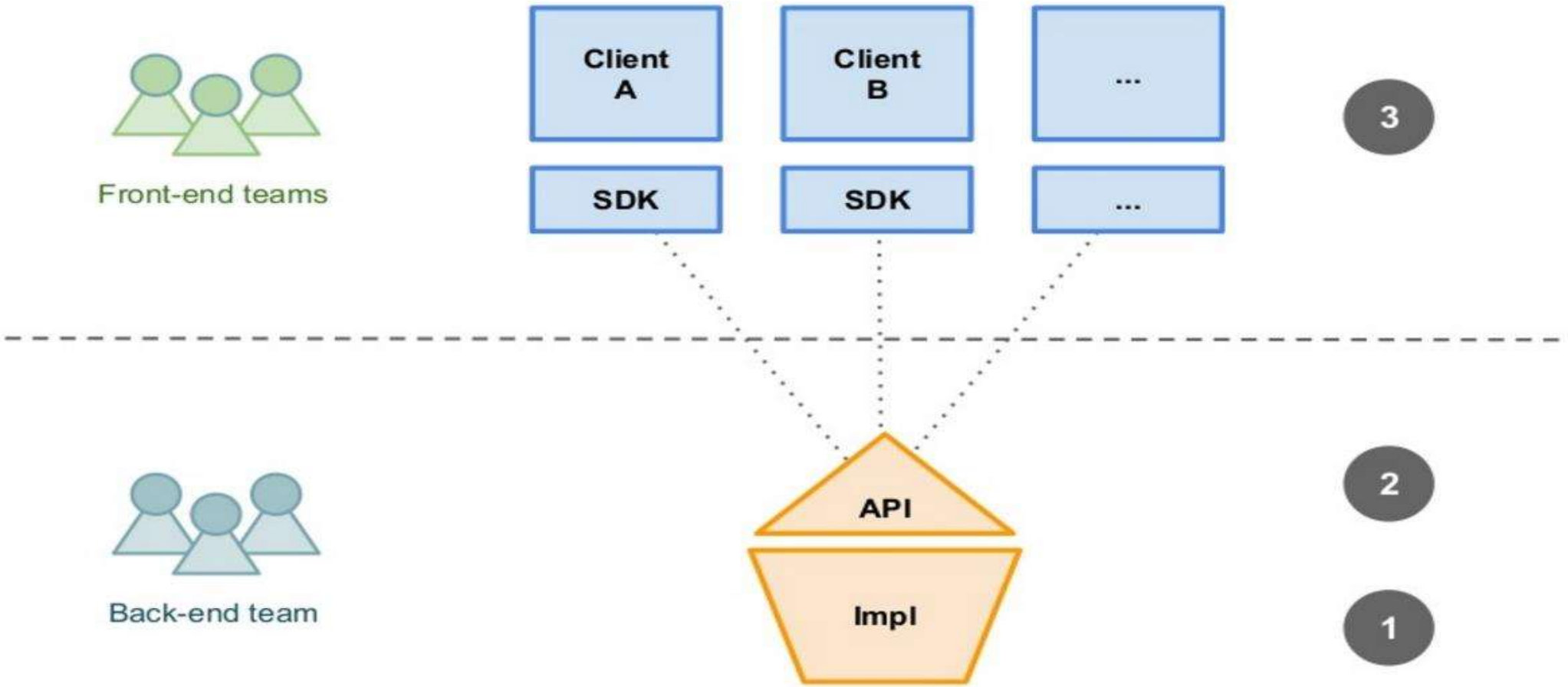
# Clean Architecture



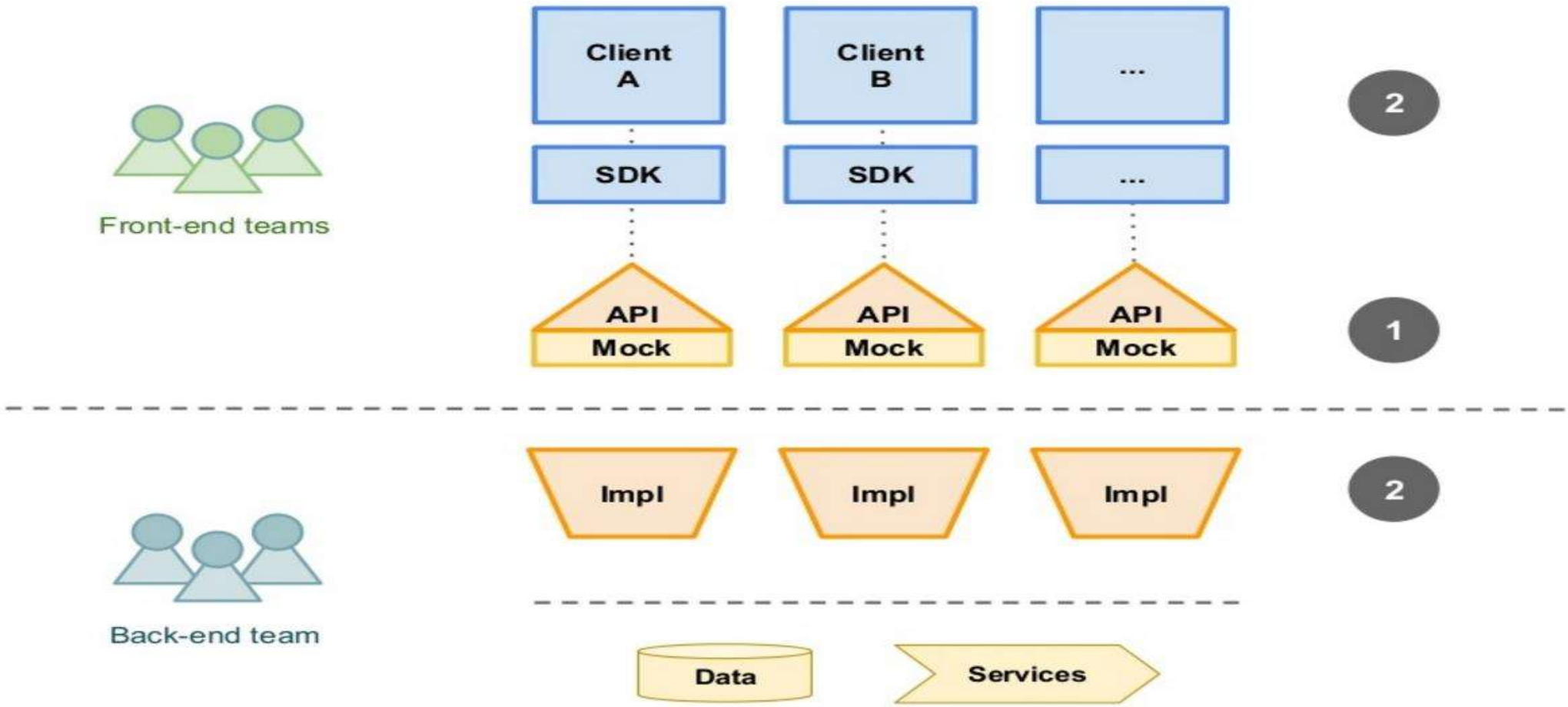
# Project Structure : Folder



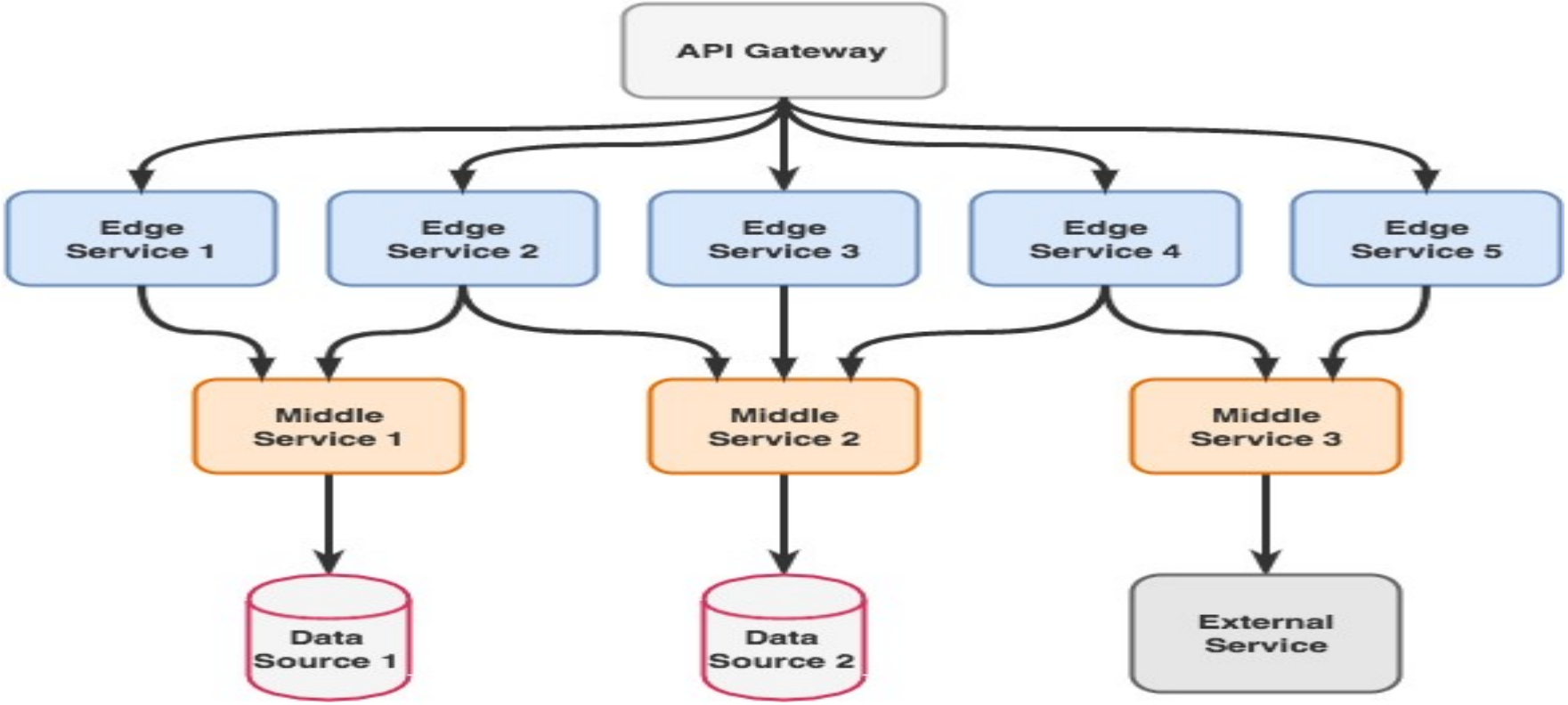
# Architecture Structure : API First



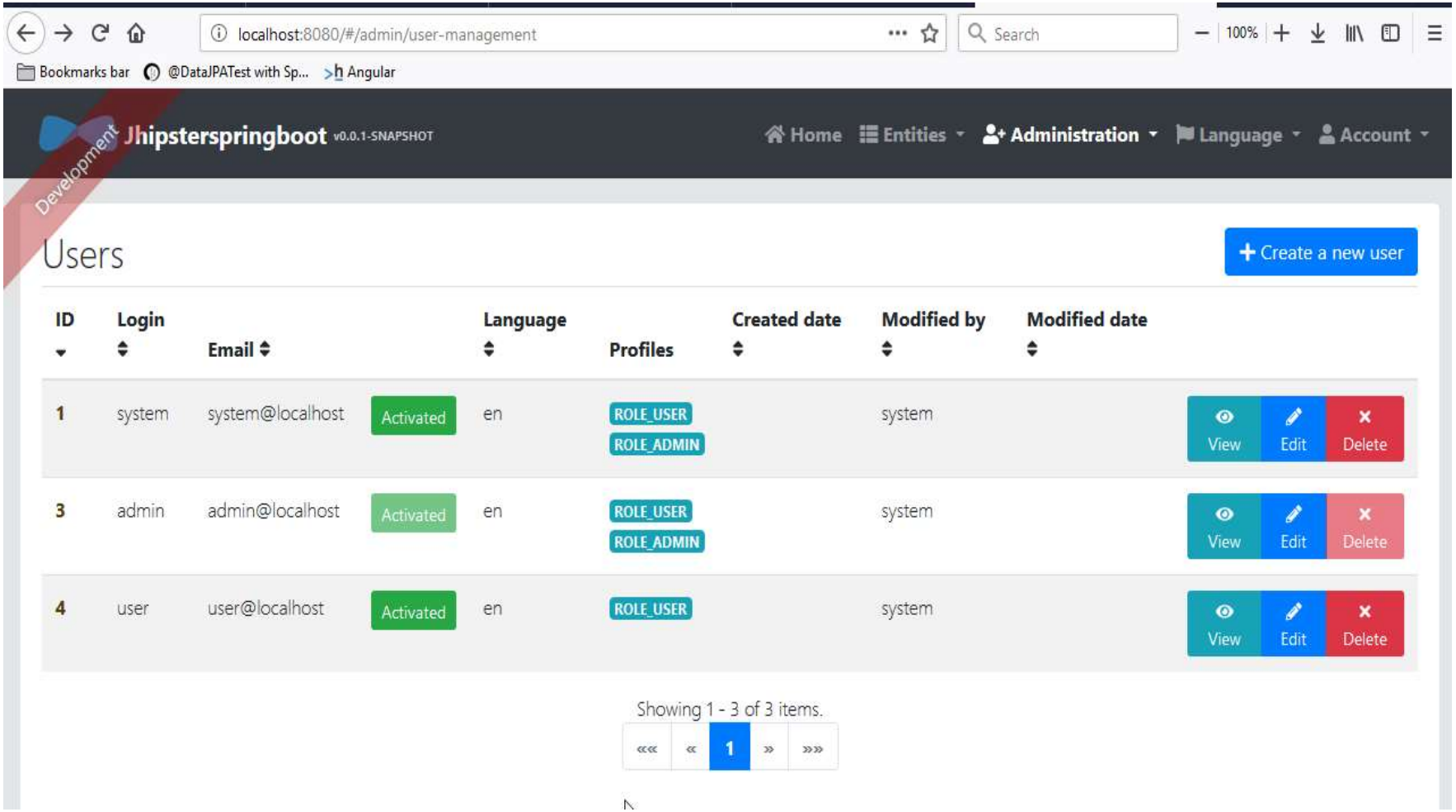
# Architecture Structure : API First



# Architecture Structure : API First



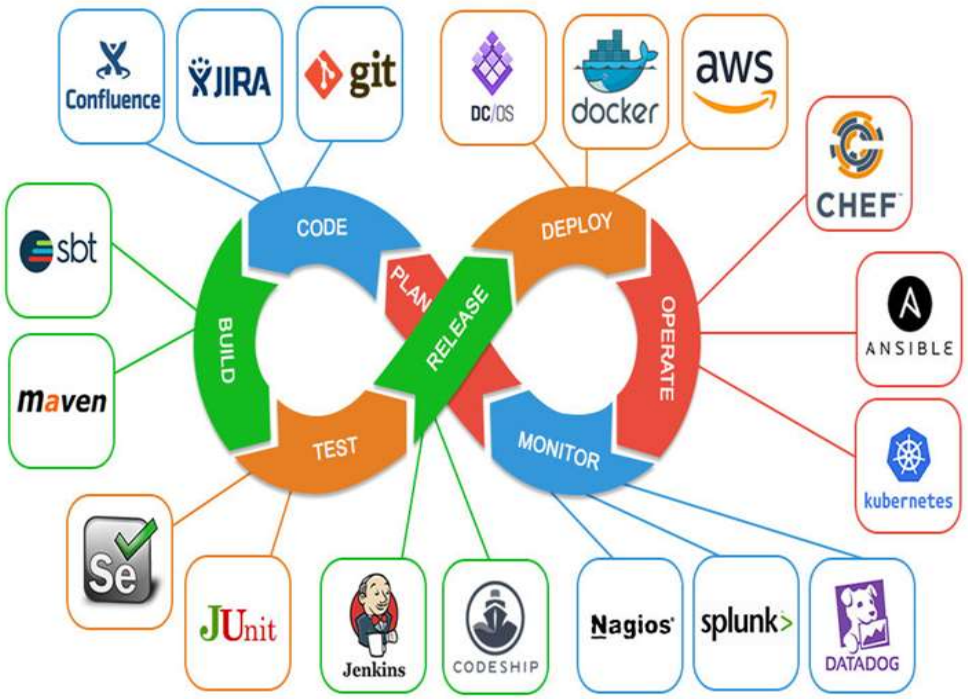
# JHIPSTER

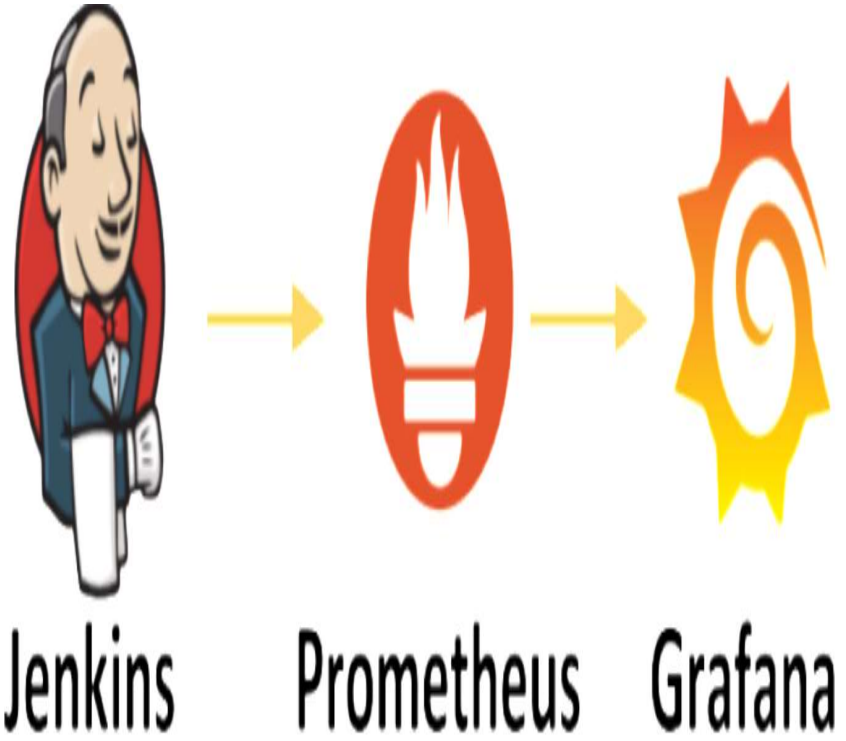

localhost:8080/#/admin/user-management
Search
100%
Bookmarks bar
@DataJPATest with Sp...
Angular
Development
Jhipsterspringboot v0.0.1-SNAPSHOT
Home
Entities
Administration
Language
Account
Users
Create a new user
ID
Login
Email
Language
Profiles
Created date
Modified by
Modified date
1 system system@localhost Activated en ROLE_USER ROLE_ADMIN system View Edit Delete
3 admin admin@localhost Activated en ROLE_USER ROLE_ADMIN system View Edit Delete
4 user user@localhost Activated en ROLE_USER system View Edit Delete
Showing 1 - 3 of 3 items.
«« « 1 » »»

# JHIPSTER: Architecture Code/ Generation

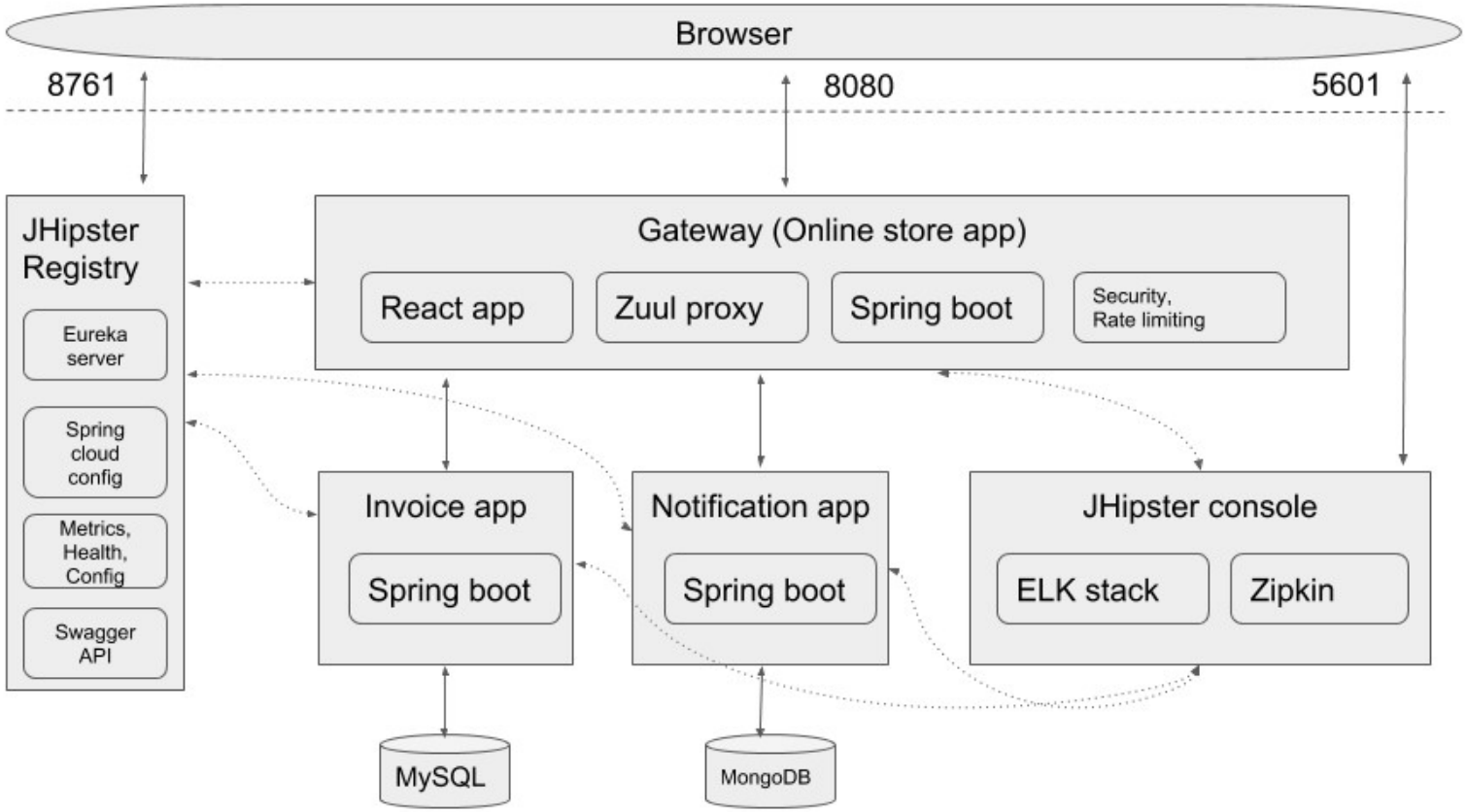

```
jhipster-beispiel — yo TERM_PROGRAM=Apple_Terminal SHELL=/bin/bash — 148×49
WKMDE5914919:jhipster-beispiel pfxadmin$ yo jhipster

     ██╗ ██╗  ██╗ ██╗ ██████╗  ███████╗ ████████╗ ███████╗ ██████╗
     ██║ ██║  ██║ ██║ ██╔══██╗ ██╔════╝ ╚══██╔══╝ ██╔════╝ ██╔══██╗
     ██║ ███████║ ██║ ██████╔╝ ███████╗    ██║    █████╗   ██████╔╝
██   ██║ ██╔══██║ ██║ ██╔═══╝  ╚════██║    ██║    ██╔══╝   ██╔══██╗
╚█████╔╝ ██║  ██║ ██║ ██║      ███████║    ██║    ███████╗ ██║  ██╗
 ╚════╝  ╚═╝  ╚═╝ ╚═╝ ╚═╝      ╚══════╝    ╚═╝    ╚══════╝ ╚═╝  ╚═╝

                    http://www.jhipster.tech

Welcome to the JHipster Generator v4.14.4
---------------------------------------------------------------------------------------------------------
  If you find JHipster useful consider supporting our collective https://opencollective.com/generator-jhipster
  Documentation for creating an application: http://www.jhipster.tech/creating-an-app/
---------------------------------------------------------------------------------------------------------

Application files will be generated in folder: /Users/pfxadmin/jhipster-beispiel


? May JHipster anonymously report usage statistics to improve the tool over time
? No
? Which *type* of application would you like to create? Monolithic application (recommended for simple projects)
[? What is the base name of your application? jhipstertest
[? What is your default Java package name? com.mycompany.myapp
[? Do you want to use the JHipster Registry to configure, monitor and scale your application? No
? Which *type* of authentication would you like to use? OAuth 2.0 / OIDC Authentication (stateful, works with Keycloak and Okta)
? Which *type* of database would you like to use? SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL)
? Which *production* database would you like to use? MySQL
? Which *development* database would you like to use? H2 with disk-based persistence
? Do you want to use the Spring cache abstraction? Yes, with the Ehcache implementation (local cache, for a single node)
[? Do you want to use Hibernate 2nd level cache? No
? Would you like to use Maven or Gradle for building the backend? Maven
? Which other technologies would you like to use?
? Which *Framework* would you like to use for the client? Angular 5
[? Would you like to enable *SASS* support using the LibSass stylesheet preprocessor? Yes
[? Would you like to enable internationalization support? Yes
? Please choose the native language of the application English
? Please choose additional languages to install
? Besides JUnit and Karma, which testing frameworks would you like to use?
>◯ Gatling
 ◯ Cucumber
 ◯ Protractor
```

# DevOps : Git + PIC Jenkins + Docker + ELK + Grafana/Prometheus + Jmeter + OWASP

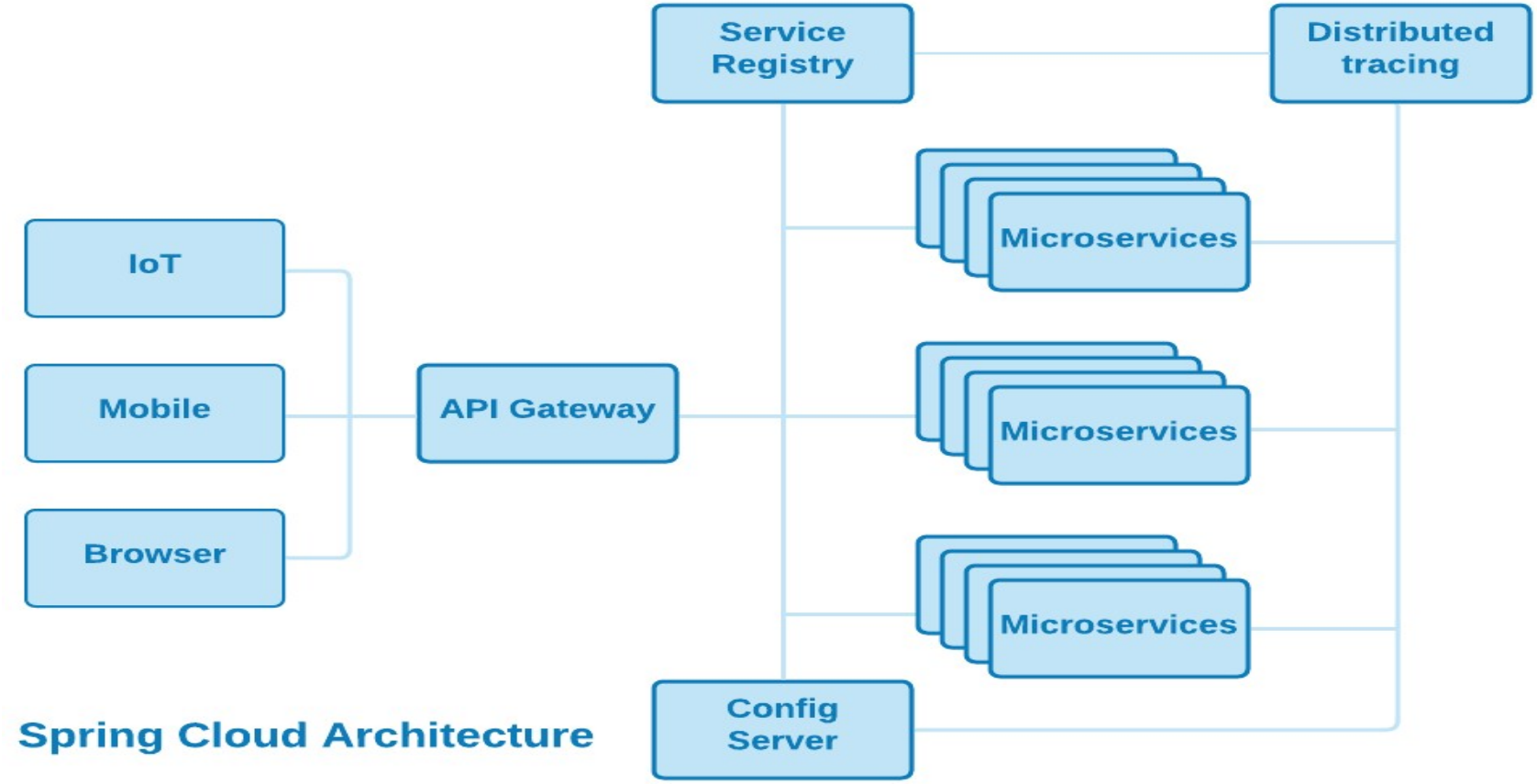


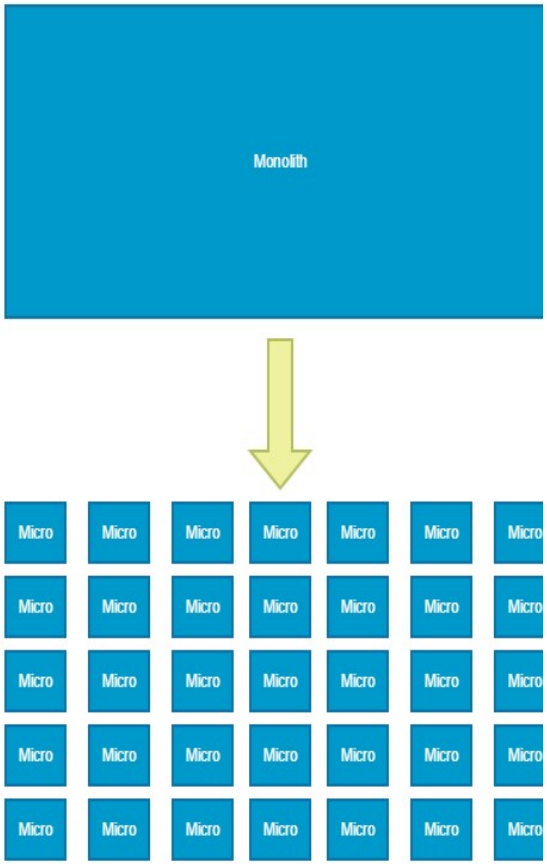
# JHIPSTER

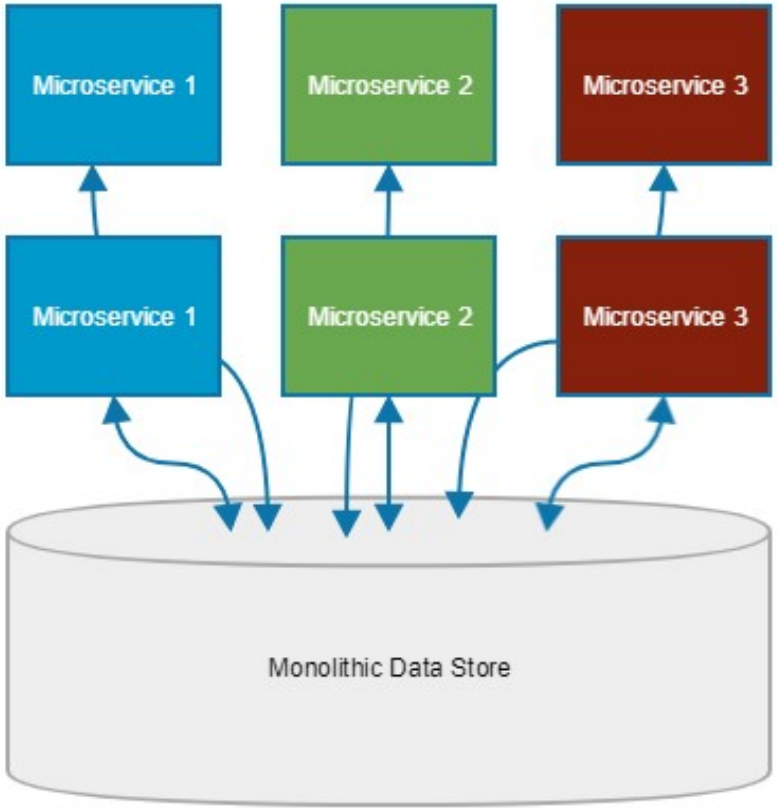

Browser
8761
8080
5601
JHipster Registry
Eureka server
Spring cloud config
Metrics, Health, Config
Swagger API
Gateway (Online store app)
React app
Zuul proxy
Spring boot
Security, Rate limiting
Invoice app
Spring boot
Notification app
Spring boot
JHipster console
ELK stack
Zipkin
MySQL
MongoDB

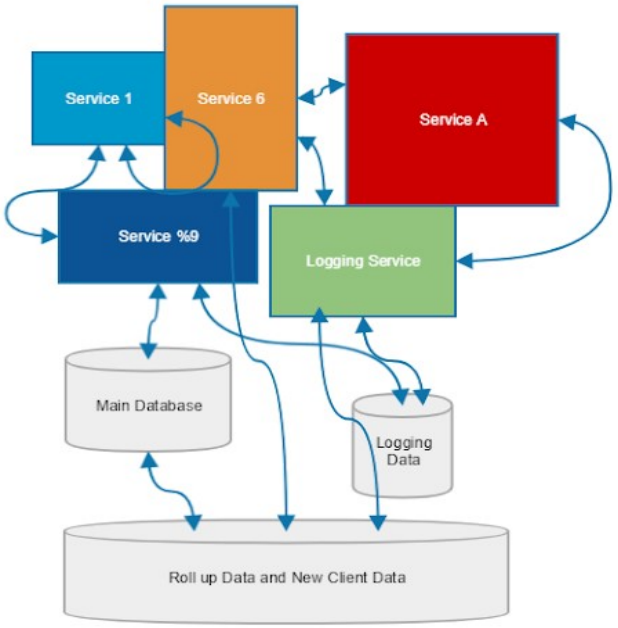
# JHIPSTER


Service Registry
Distributed tracing
IoT
Mobile
Browser
API Gateway
Microservices
Microservices
Microservices
Config Server
Spring Cloud Architecture

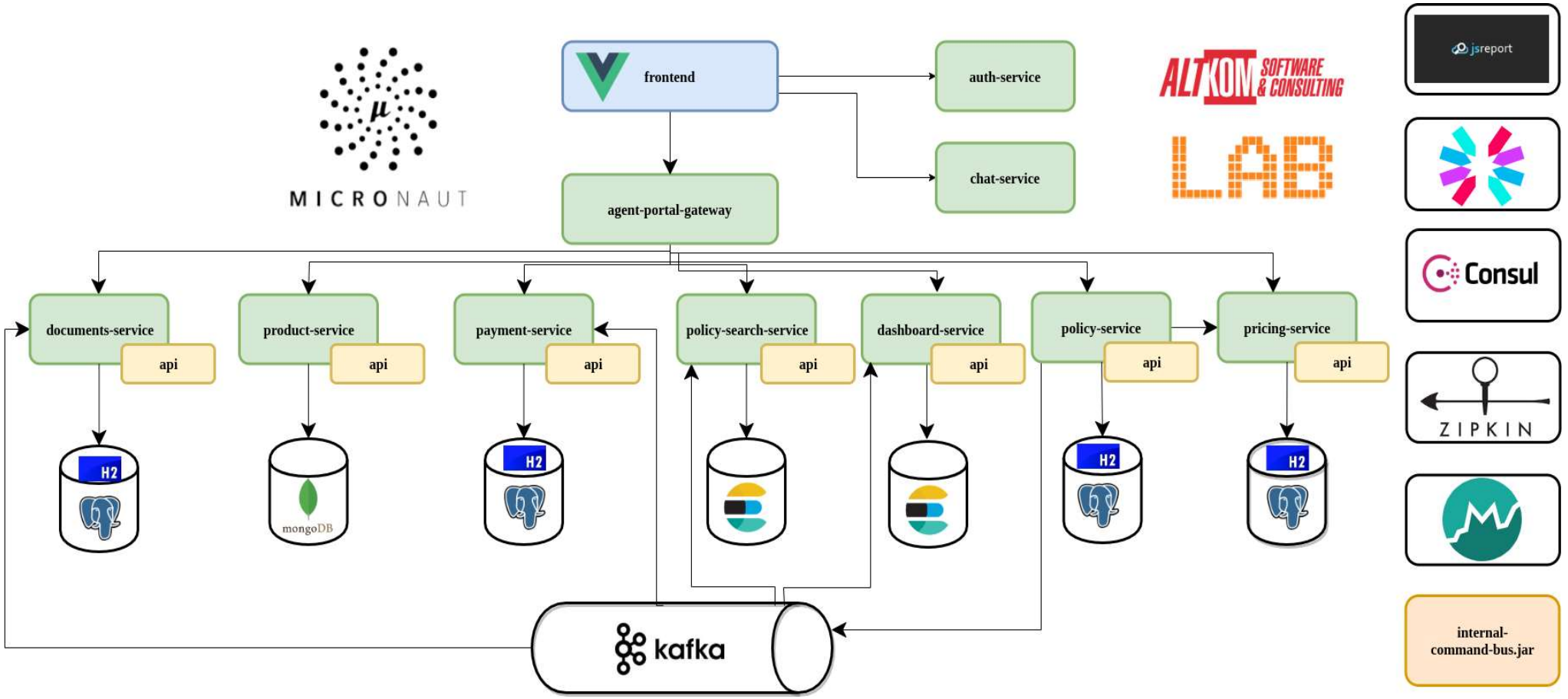
# MicroServices AntiPatterns



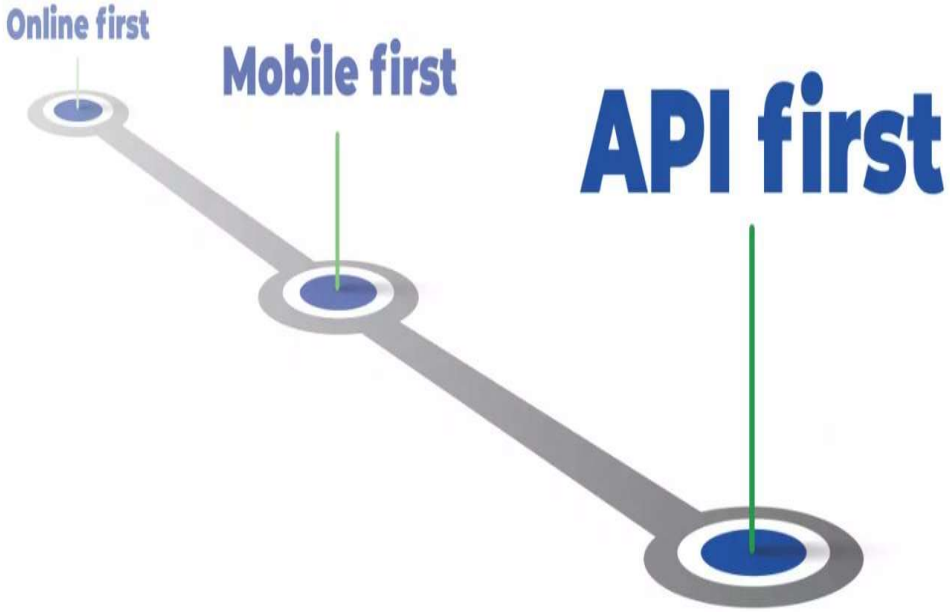



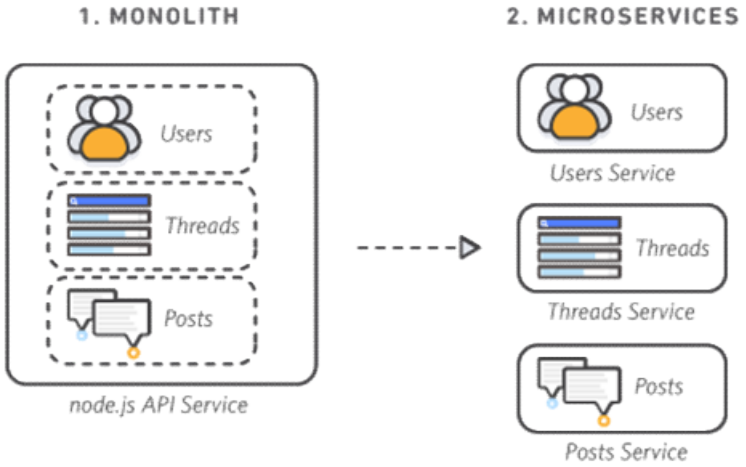



# Découpage des Micro services en 1, 2, 3, 5, 8, 13  Modules maximum

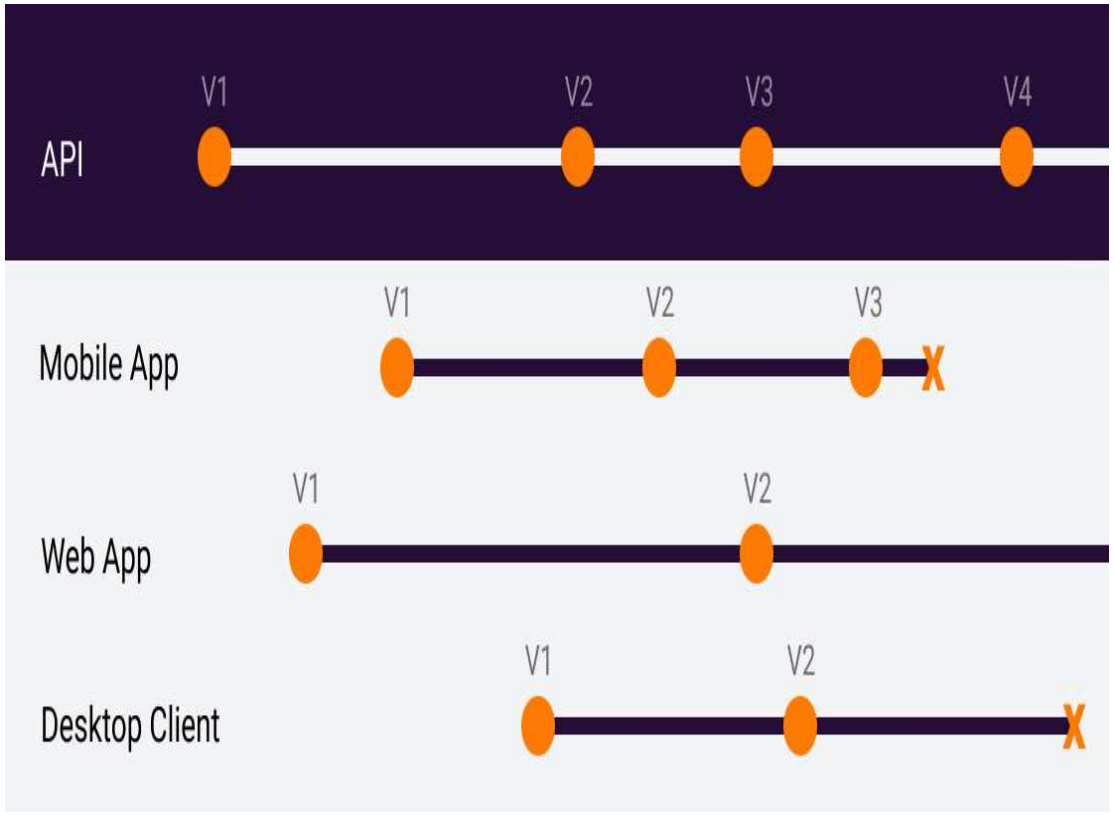


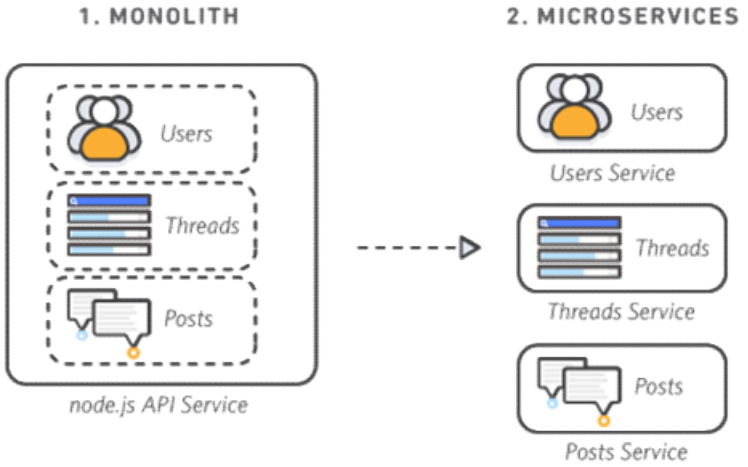
# Conclusion


Online first
Mobile first
API first

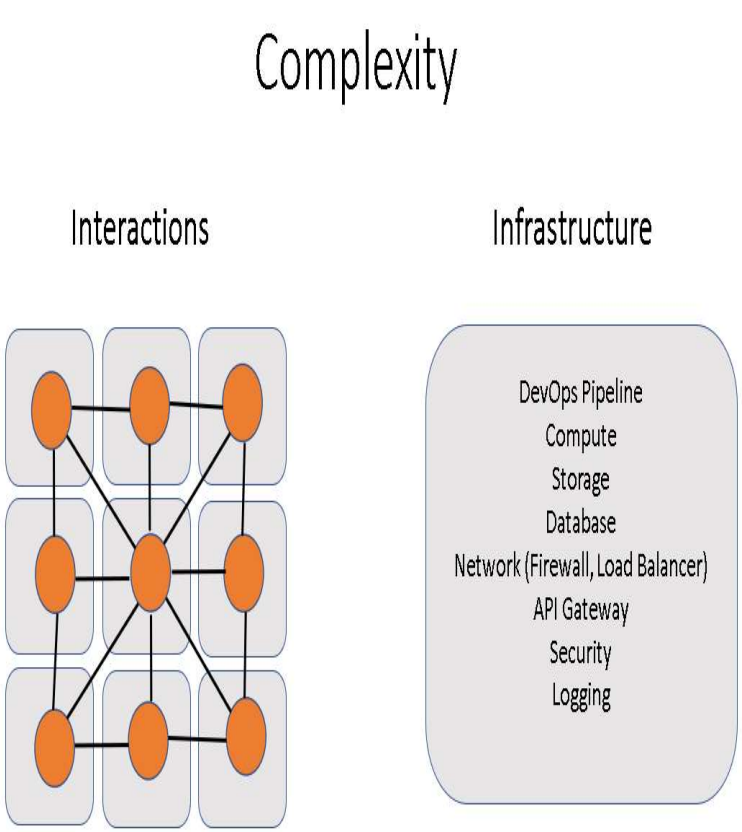

1. MONOLITH
2. MICROSERVICES
Users
Threads
Posts
node.js API Service
Users
Users Service
Threads
Threads Service
Posts
Posts Service

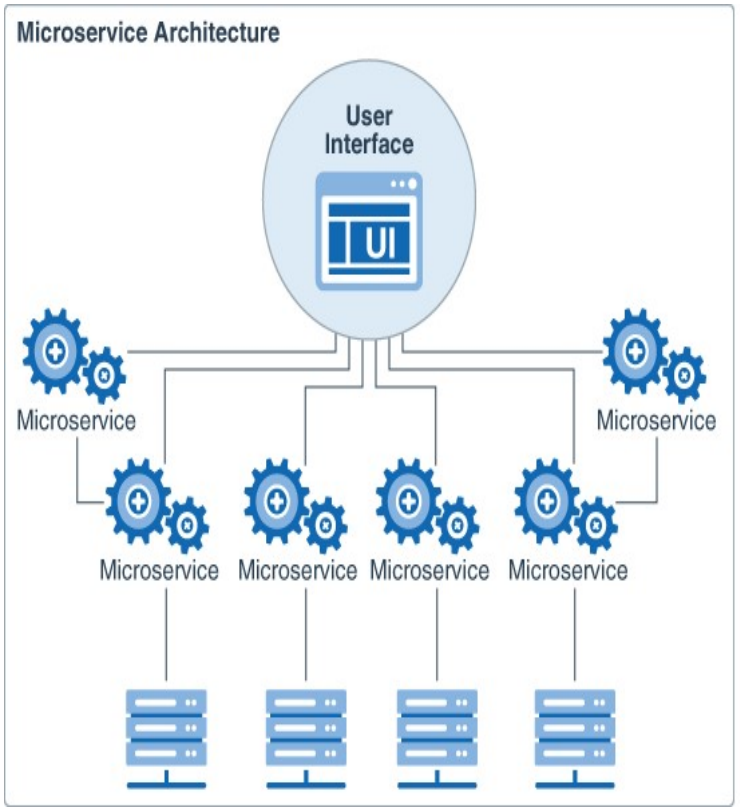
# Conclusion

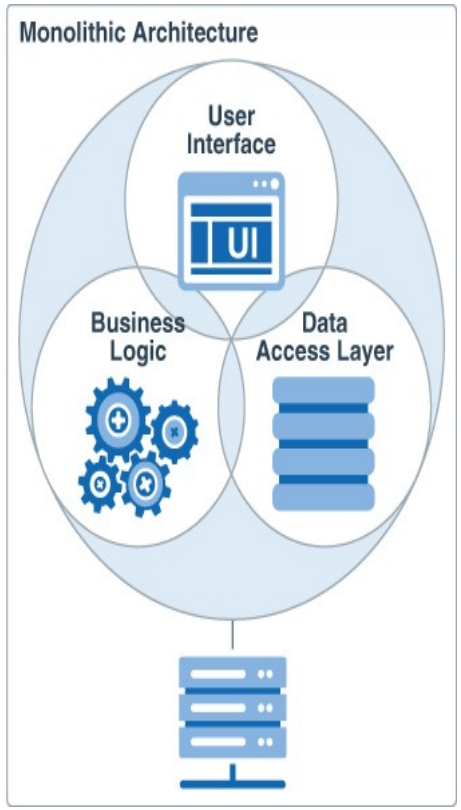

API
V1
V2
V3
V4
Mobile App
V1
V2
V3
X
Web App
V1
V2
Desktop Client
V1
V2
X


1. MONOLITH
Users
Threads
Posts
node.js API Service
2. MICROSERVICES
Users
Users Service
Threads
Threads Service
Posts
Posts Service

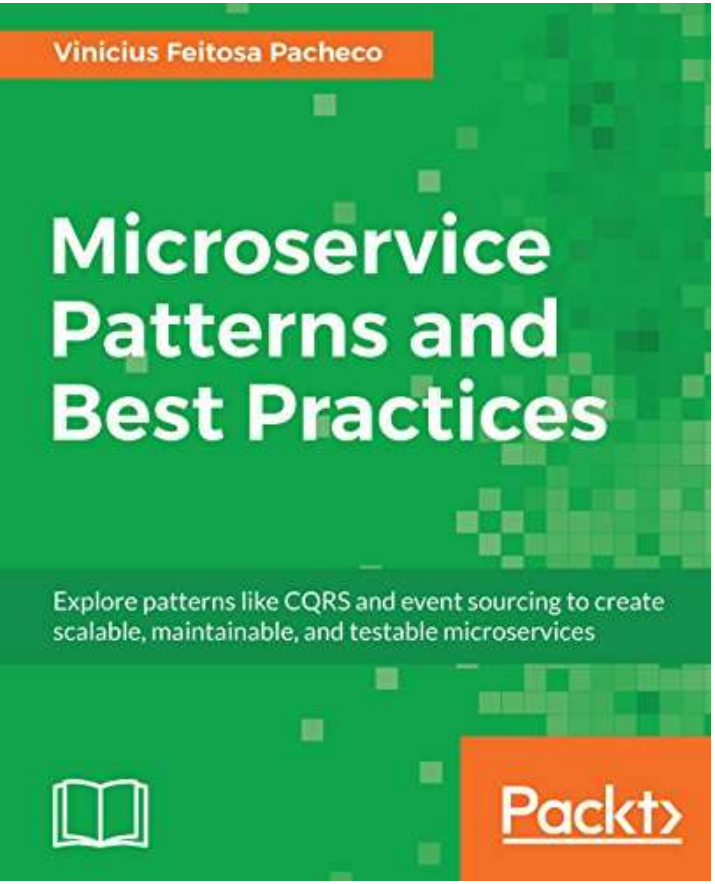
# MERCI !

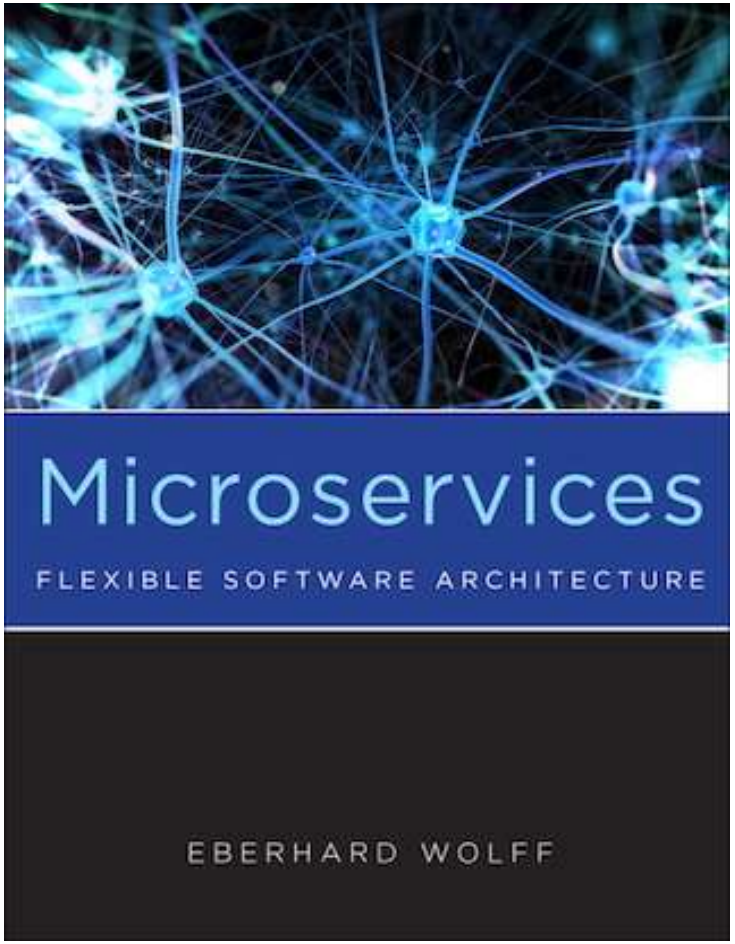

Complexity
Interactions
Infrastructure
DevOps Pipeline
Compute
Storage
Database
Network (Firewall, Load Balancer)
API Gateway
Security
Logging

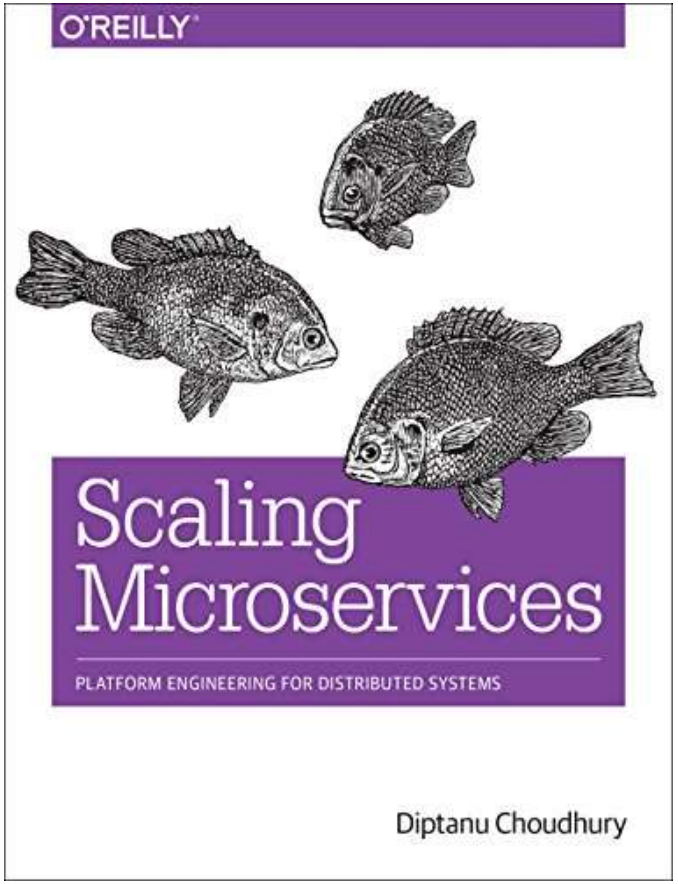

Microservice Architecture
User Interface
UI
Microservice
Microservice
Microservice
Microservice
Microservice
Microservice


Monolithic Architecture
User Interface
UI
Business Logic
Data Access Layer

# BIBLIOGRAPHIE



# BIBLIOGRAPHIE


https://github.com/sanogotech/microservicesFormationFull

https://baeldung-cn.com/jhipster-microservices

https://www.dotnetcurry.com/microsoft-azure/microservices-architecture

https://eventuallycoding.com/2015/04/09/zuul/

https://itnext.io/how-to-build-an-event-driven-asp-net-core-microservice-architecture-e0ef2976f33f

https://itnext.io/how-to-build-an-event-driven-asp-net-core-microservice-architecture-e0ef2976f33f

https://github.com/sanogotech/books-1

https://www.cloud-devops.fr/liste-des-codes-http/

https://github.com/sanogotech/jhipster-cheatsheet