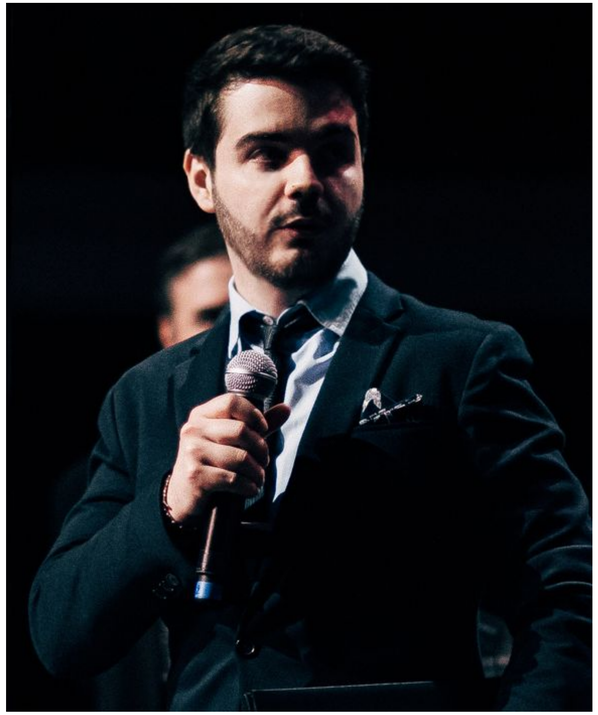


Samuel ANTUNES
Consultant Ingénieur DevSecOps
OCTO Technology
Email : contact@samuelantunes.fr

aws certified
Solutions Architect
Associate

aws training and certification
Authorized Instructor

ICEBREAKER TIME
https://laviedesreines.com/questions/questions-je-nai-jamais/

“ L’origine du DevOps ”

DevOps ?

Juin 2009 : John Allspaw et Paul Hammond posent les bases de DevOps lors d'une conférence :
10+ Deploys a Day: Dev and Ops Cooperation at Flickr.

Octobre 2009 : le mot DevOps a été inventé par Patrick Debois durant l'organisation des premiers devopsdays en Belgique.

RÉPONDRE AUX EXIGENCES DU MARCHÉ

Grosse pression du marché

On veut rapidement de nouvelles fonctions

On veut corriger plus vite les bugs voire les prévenir

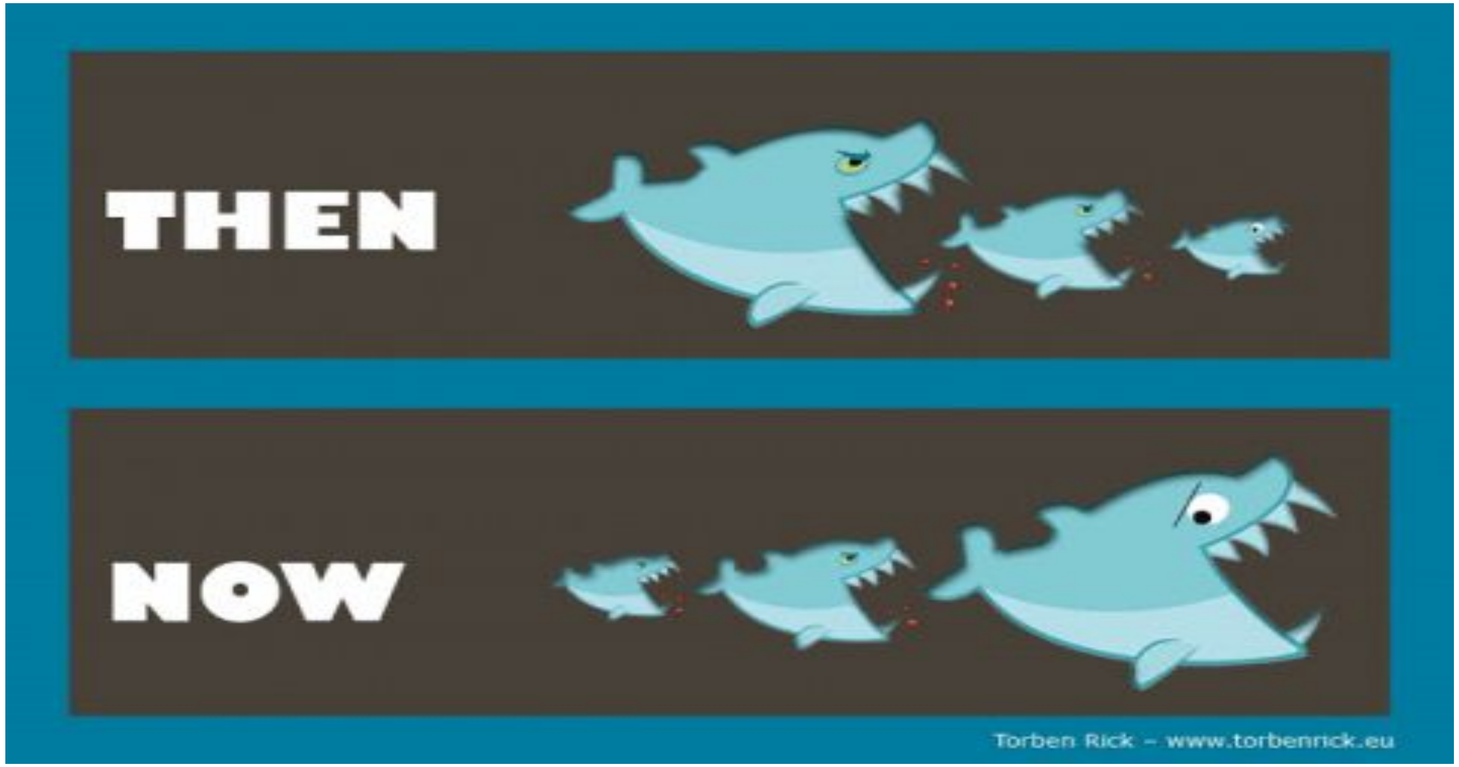
On veut un SLA « 24/24 7/7 »

On veut une meilleure UX

On veut pas payer cher

On veut pouvoir grandir vite

Ma boîte n’est **pas un Géant du Web**, alors pourquoi me parler de DevOps?



- Contexte concurrentiel tendu
- Nouveaux entrants sur les marchés historiques
- Enjeux de Time to Market de plus en plus importants
- Enjeux de qualité et de disponibilité de plus en plus importants
- La transformation digitale des métiers

“ La philosophie DevOps ”

UNE DEFINITION

DevOps est un ensemble de pratiques qui visent à réduire le Time to Market et à améliorer la Qualité en optimisant la coopération entre les Développeurs et la Production

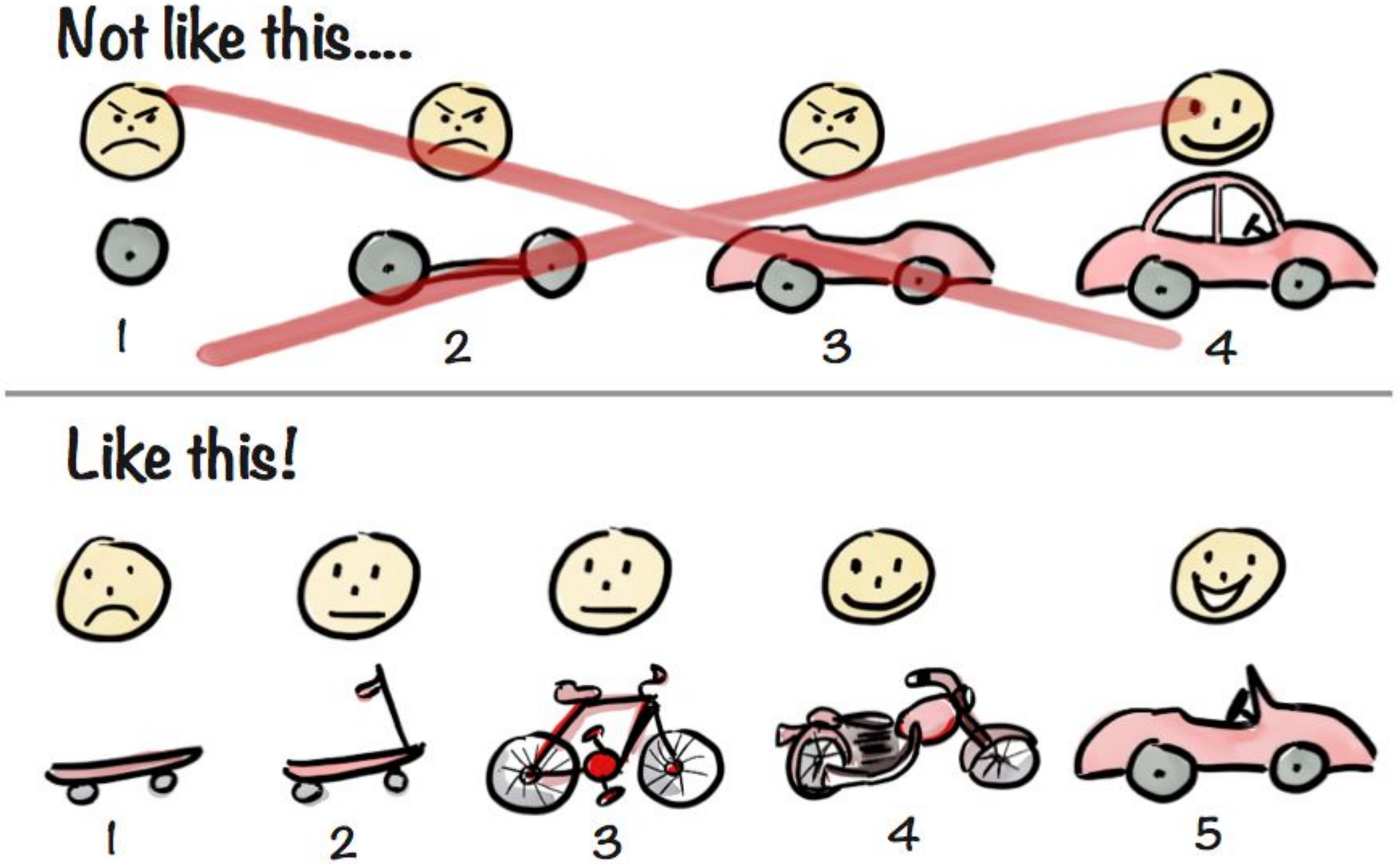
LES VALEURS FONDATRICES

Les méthodes Agile accordent une valeur plus importante (pas UNIQUE) à :

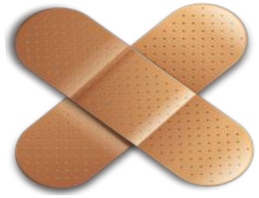
Personnes et échanges	>>	Méthodes et outils
Logiciel qui fonctionne	>>	Documentation exhaustive
Collaboration avec le client	>>	Négociation contractuelle
Adaptation au changement	>>	Suivi d’un plan établi

Source : http://agilemanifesto.org/

DICTÉE PAR LA #VALEUR

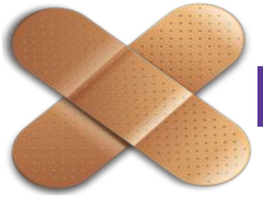


CONSTATS : DES DOULEURS RÉCURRENTES



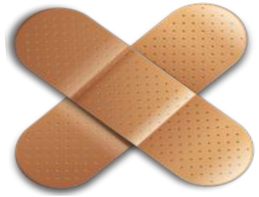
DOULEUR #1

Trop long pour provisionner des environnements



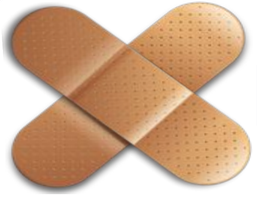
DOULEUR #2

C’est chaud entre les DEV et les OPS



DOULEUR #3

RIP la qualité qui provoque des anomalies en prod



DOULEUR #4

Le déploiement de l’appli est beaucoup trop long

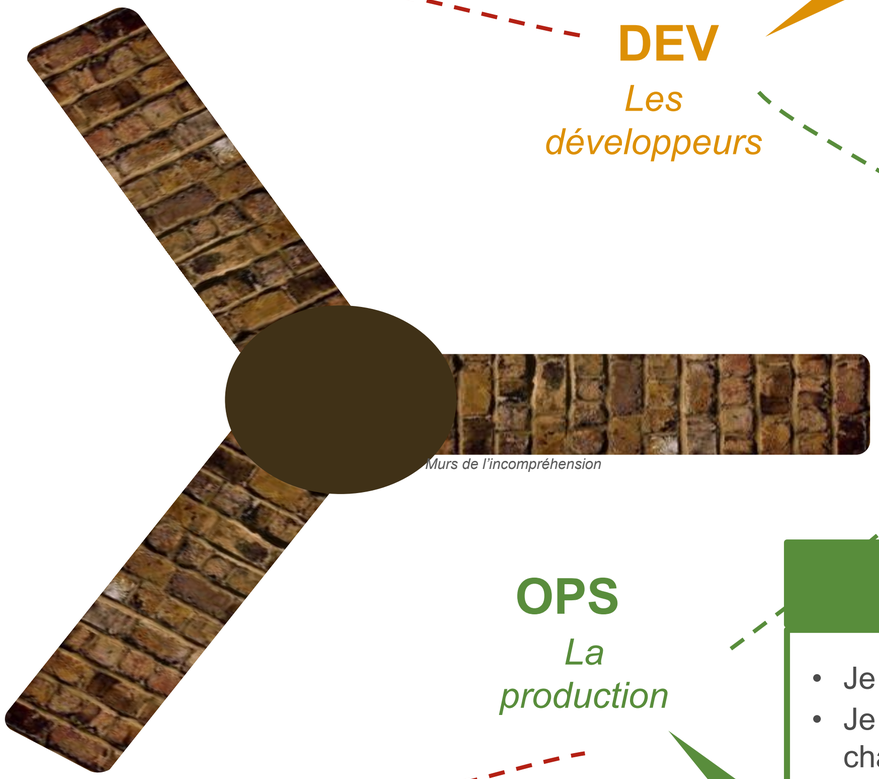
DES POINTS DE VUE DIFFÉRENTS …

Rapidité

- Je veux livrer **vite** des **nouvelles fonctionnalités**
- Je veux **innover**
- Easy le **changement**

DEV

Les développeurs



Murs de l'incompréhension

BIZ

Le métier

Rapidité Qualité Stabilité Coût

- Je veux donner du rêve très **rapidement** à mes clients
- Je veux qu'il n'y ait **aucun** problème.
- Et calmez-vous avez la **thune**, on n'est pas Amazon nous.

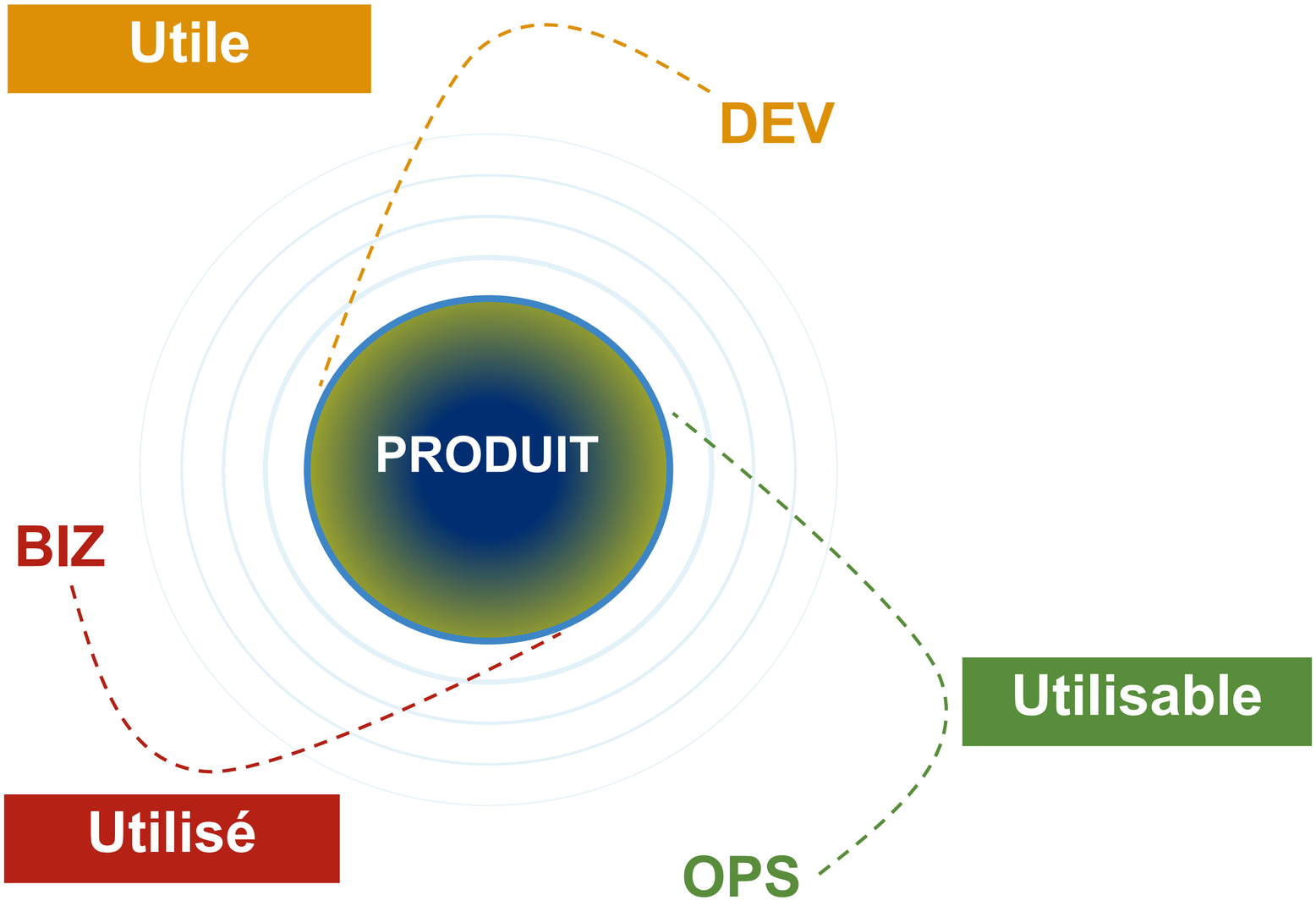
OPS

La production

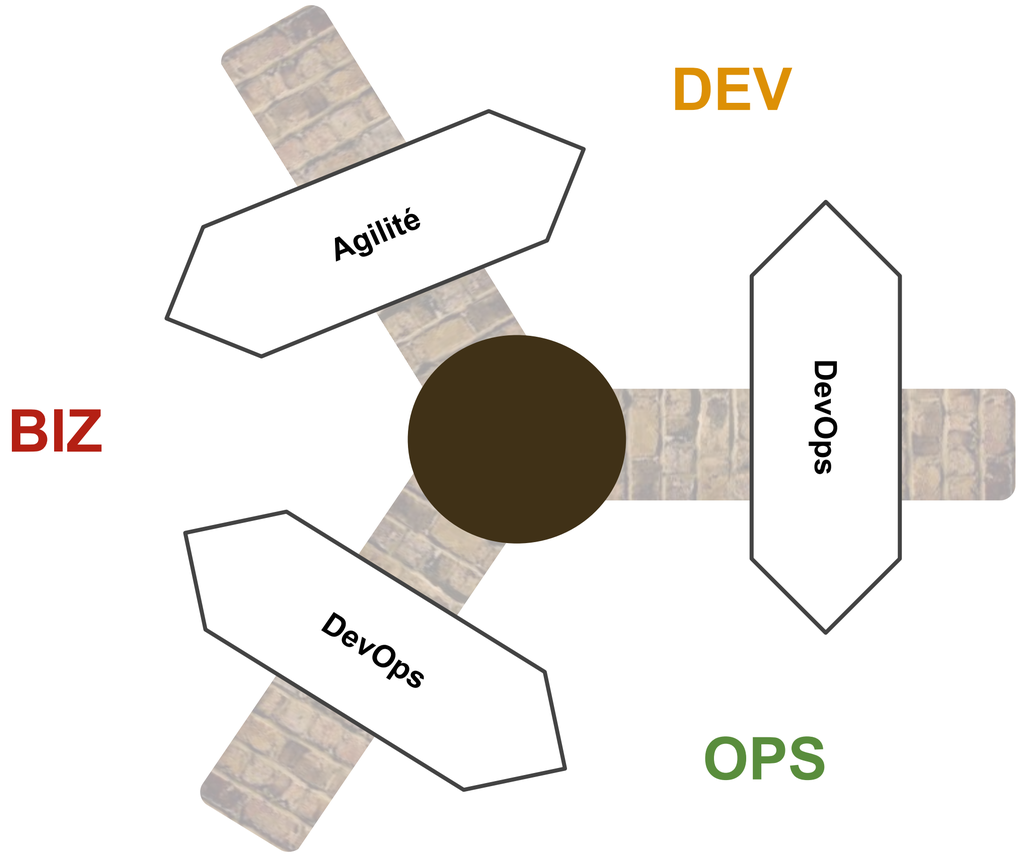
Stabilité Qualité Coûts

- Je veux que ce soit **stable**
- Je veux de la **qualité** à chaque changement
- Culture du **service** (SLA : bah oué c'est mort les gars, je vais pas me lever à 3h du mat parce que vous savez pas coder)

DES ENJEUX COMMUNS...

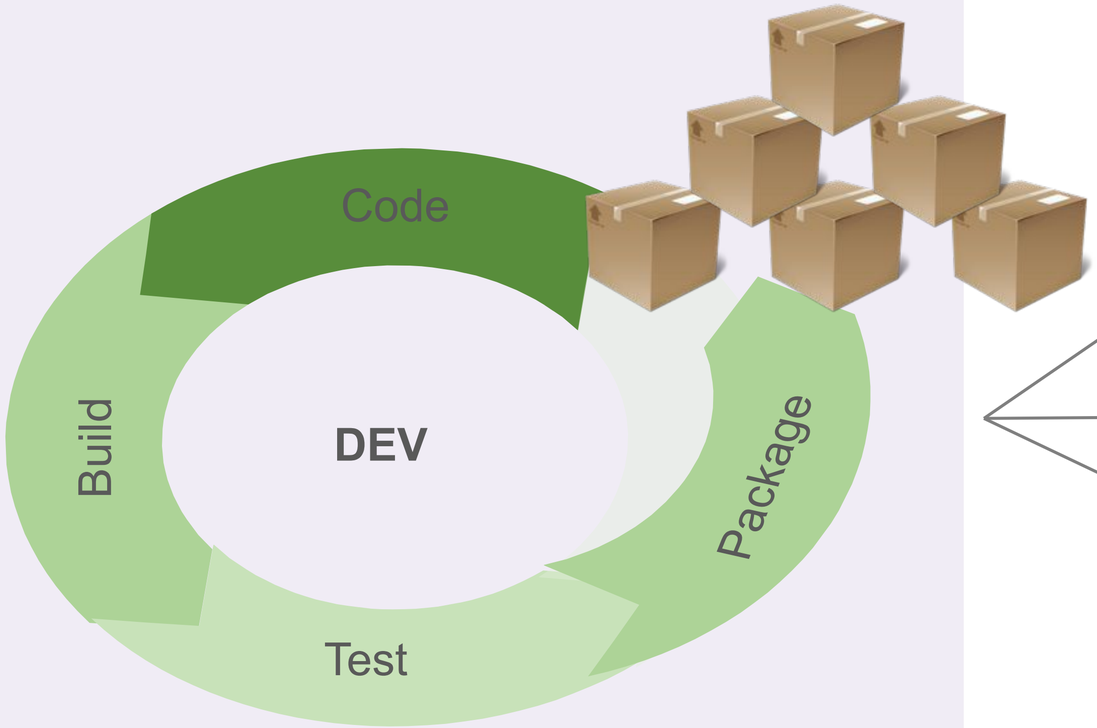


LES PRATIQUES POUR ROMPRE LES MURS...



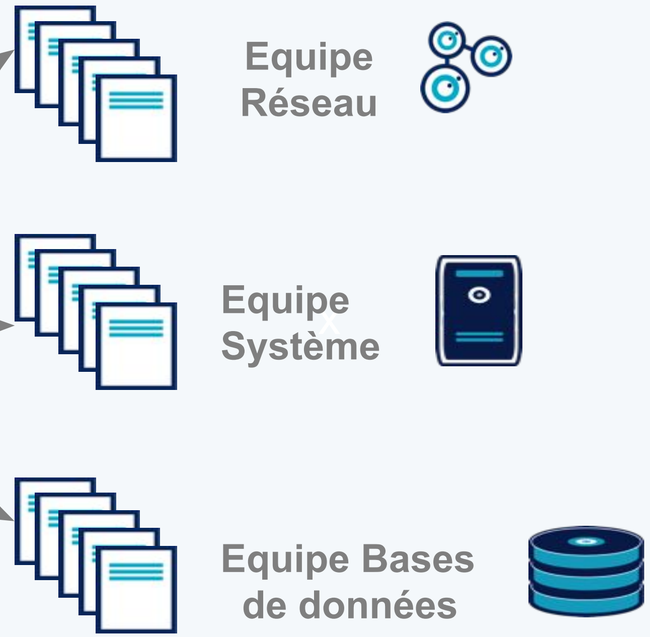
DES ORGANISATIONS DIFFÉRENTES

Développement Agile : équipe autonome



- Durée : 2 - 3 semaines

Ecosystème en silo



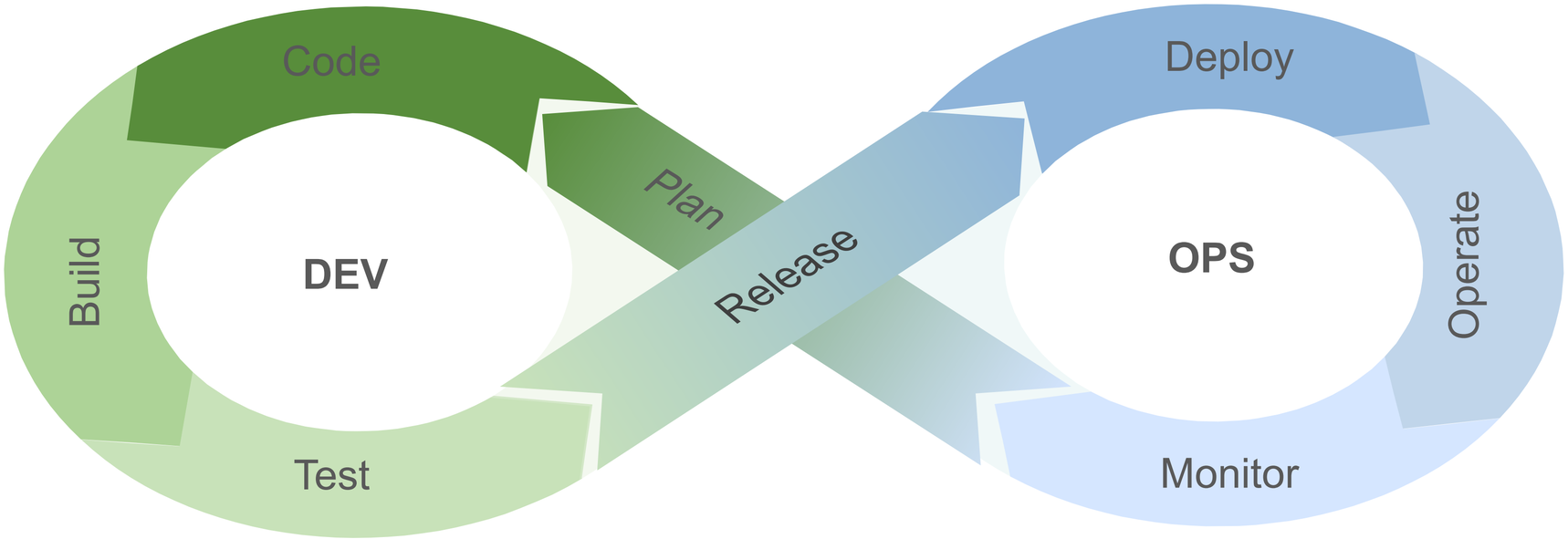
- Durée : 2 - 3 mois

ÉTENDRE L'AGILITÉ À LA PRODUCTION

Agilité des Développements

Agilité des Opérations

Convergence des processus : DEVOPS



- Durée : 2 - 3 semaines

WATERFALL - AGILE - DEVOPS

“ Les 4 piliers du DevOps ”

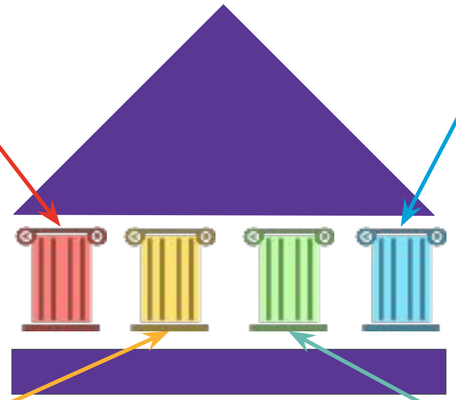
LES 4 PILIERS DU #DEVOPS

Culture, méthode et organisation

Culture, des modèles d'organisation et des méthodes favorisant le dialogue des Dev et des Ops

Architecture et patterns

Architectures et patterns répondant aux préoccupations des Devs et des Ops



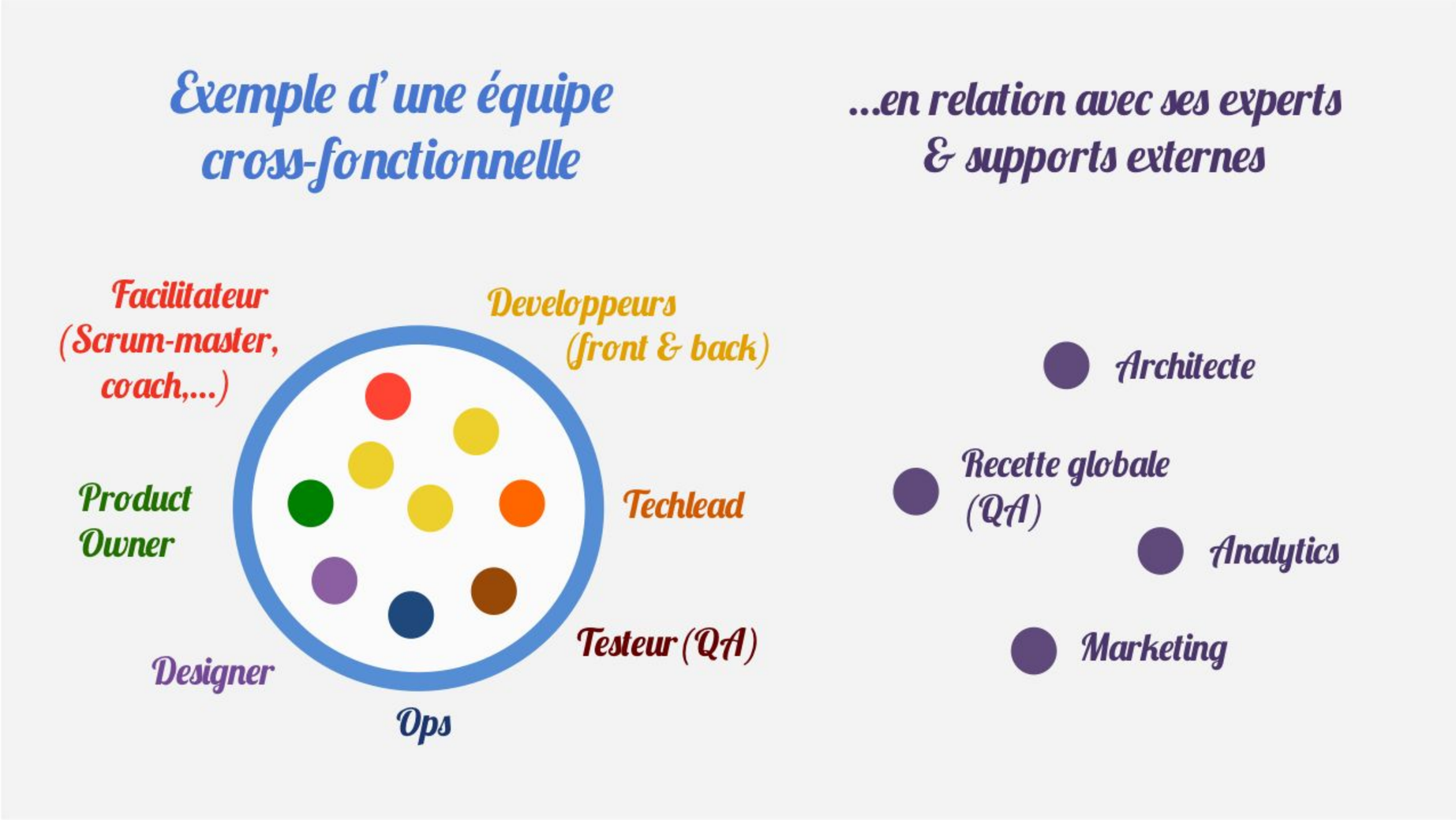
Construction et déploiement continu

Outils et processus de construction et déploiement continu

Infrastructure as code "IaC"

Outils d'automatisation pour construire et maintenir l'infrastructure par du code

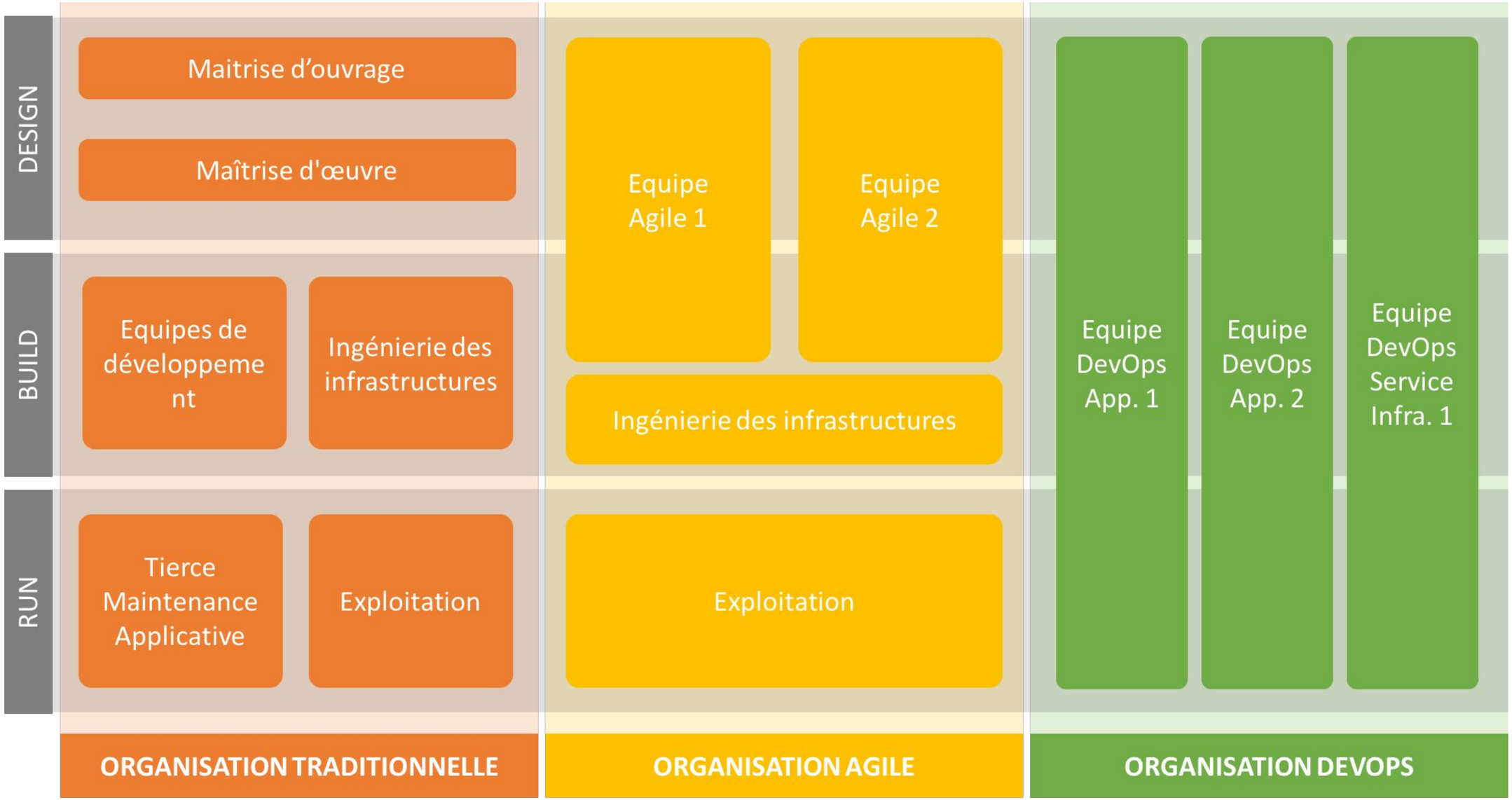
- **L'union des compétences pour atteindre l'objectif est un élément clef de DevOps**



Des modèles d’organisation favorisant l’autonomie et la responsabilisation (aussi appelé le Shift-Left)

- Décloisonnement des organisations (fin des silos techniques)
- « *You build it, you run it* »
- Autonomisation des équipes
- Les Pizza Teams (celles de chez PizzaHut avec le cheezy crust bien sûr, pas votre domino’s là)

LES ORGANISATIONS



CULTURE, MÉTHODE ET ORGANISATION

RITUELS

- VIS MA VIE (SHADOWING. Qui a dit #VDM ?)
- WAR ROOM
- PEER REVIEW
- BBL
- RÉTROSPECTIVE
- POST-MORTEM
- PAIR-PROGRAMMING
- HACKATHON

ATTITUDES

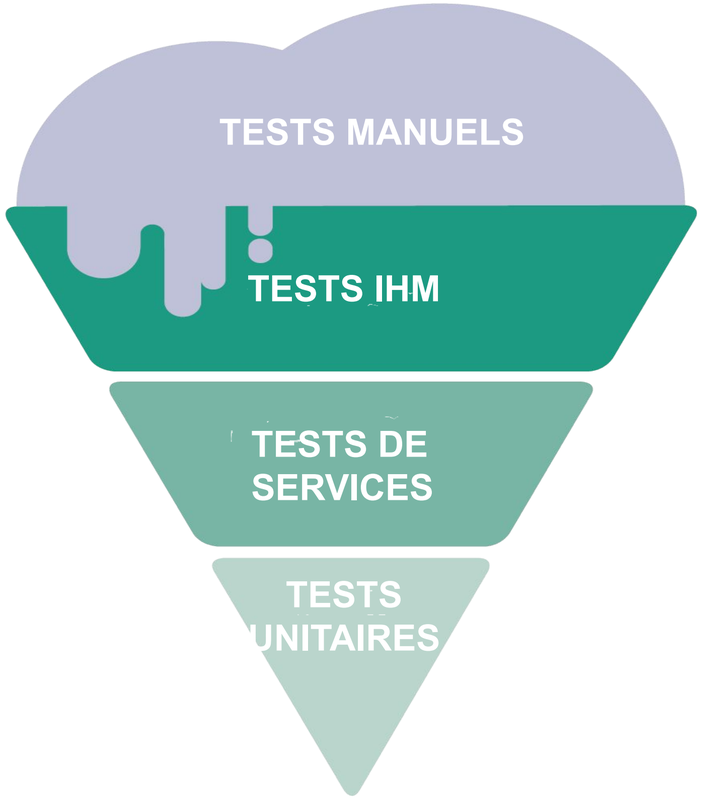
- TRANSPARENCE
- OPEN DATA
- DROIT À L'ERREUR
- SOLIDARITÉ

METHODO

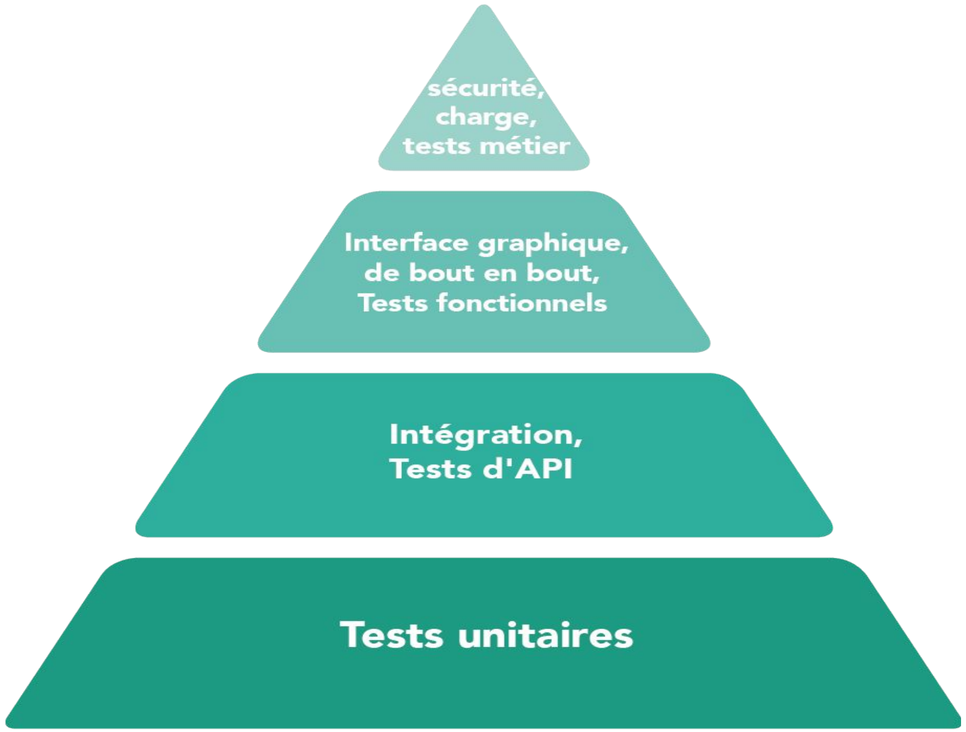
- KANBAN
- KAIZEN
- LEAN STARTUP
- 5 WHY

CULTURE, MÉTHODE ET ORGANISATION

Approche traditionnelle :
Trouver les bugs

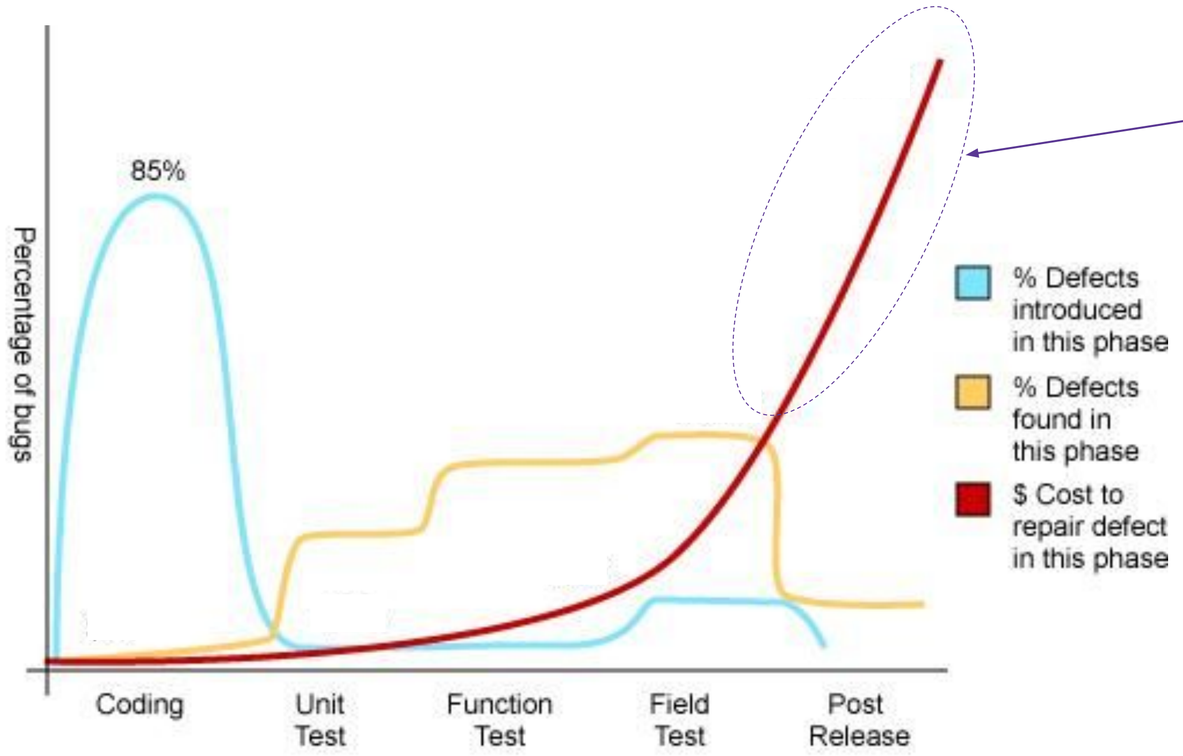


Approche agile:
Prévenir les bugs



CULTURE, MÉTHODE ET ORGANISATION

- Identifier les **bugs/régressions au plus tôt** permet d'**éviter des corrections coûteuses**.



Source: Applied Software Measurement, Capers Jones

Le **coût de correction** de bugs en **production** est le plus **élevé**.

“Une bonne stratégie de tests devrait offrir un ***feedback rapide et en continu****, afin d’****être efficace dans la correction des défauts détectés et minimiser la perte de temps****.”*

CULTURE, MÉTHODE ET ORGANISATION

Pratiquer les tests **directement lors du développement**

Se concentrer sur certains types de tests plus efficace

Effectuer les tests **au plus tôt**

Automatiser

CULTURE, MÉTHODE ET ORGANISATION

FONCTIONNEL

Tests des nouvelles fonctionnalités

—

Tests de non régression fonctionnelle

Tests d'acceptation utilisateur (UAT)

—

Tests exploratoires

—

Tests d'usabilité

CONSTRUIRE

"des tests pour guider l'équipe et supporter le développement"

CONTRÔLER

"des tests pour critiquer le produit réalisé"

Tests unitaires

—

Tests d'intégration

Tests de robustesse

—

Tests de charge

—

Tests de sécurité

TECHNIQUE

Des métriques partout ! Pour mesurer la performance :

- des **services** : supervision technique et applicative (Performance applicative et technique, fiabilité, sûreté de fonctionnement)
- de **l'organisation** : Coûts / compétences, Ratios de performance par technologie…
- des **processus** (Productivité, célérité des développements, taux d'erreur, blocages, SLA…)
- du **produit** : satisfaction client, taux de conversion, time to market

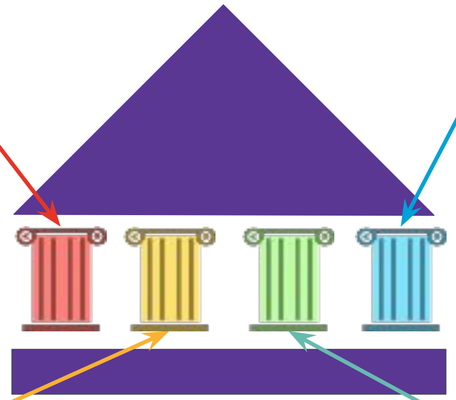
LES 4 PILIERS DU #DEVOPS

Culture, méthode et organisation

Culture, des modèles d'organisation et des méthodes favorisant le dialogue des Dev et des Ops

Architecture et patterns

Architectures et patterns répondant aux préoccupations des Devs et des Ops



Construction et déploiement continu

Outils et processus de construction et déploiement continu

Infrastructure as code "IaC"

Outils d'automatisation pour construire et maintenir l'infrastructure par du code

ARCHITECTURE ET PATTERNS

SCALABILITÉ / DISPONIBILITÉ

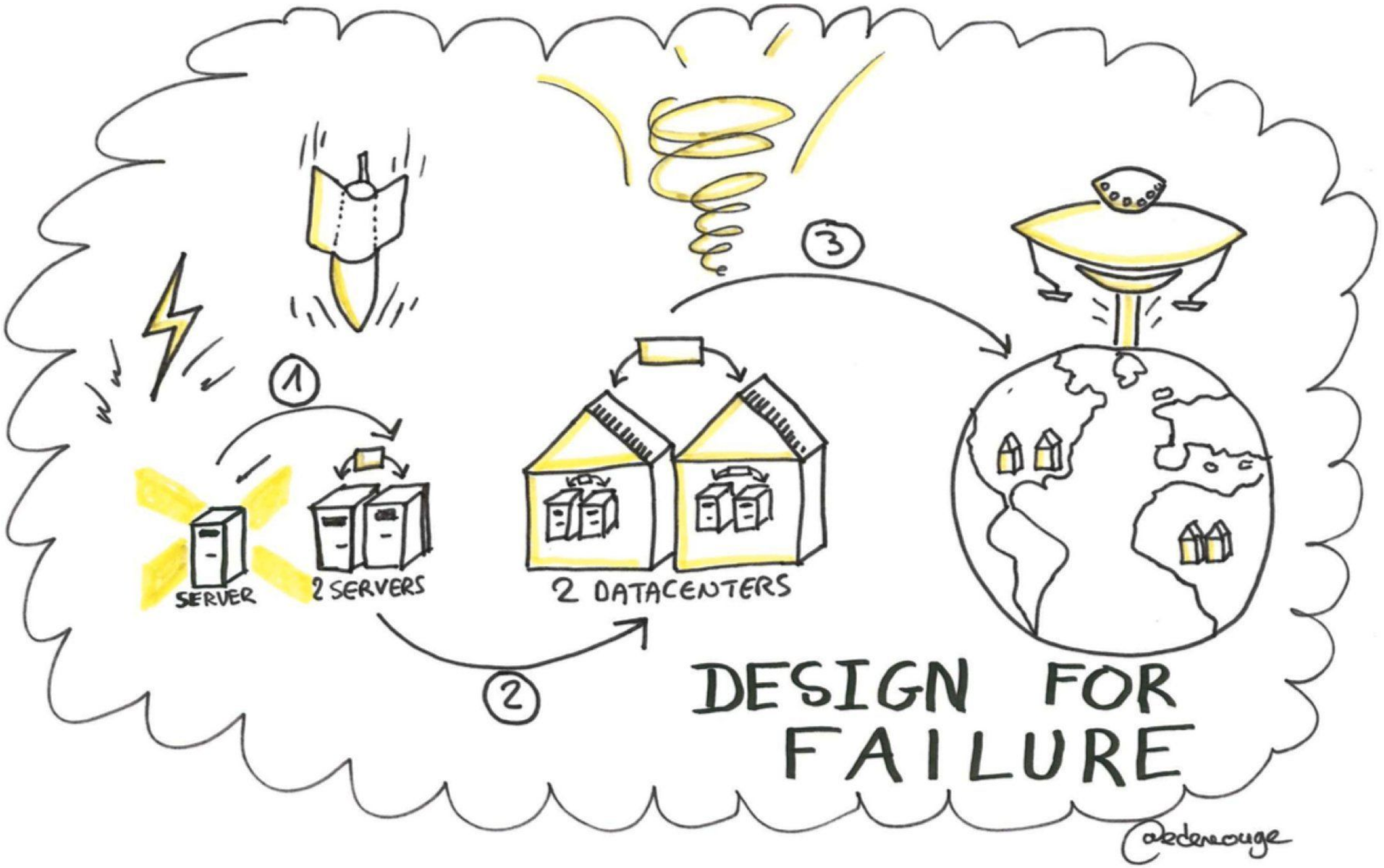
- DESIGN FOR FAILURE
- APPLICATION STATELESS
- DÉCOUPLAGE COMPOSANTS SCALABLES
- UTILISATION SERVICE TIERS PERSISTENCE

EXPLOITABILITÉ

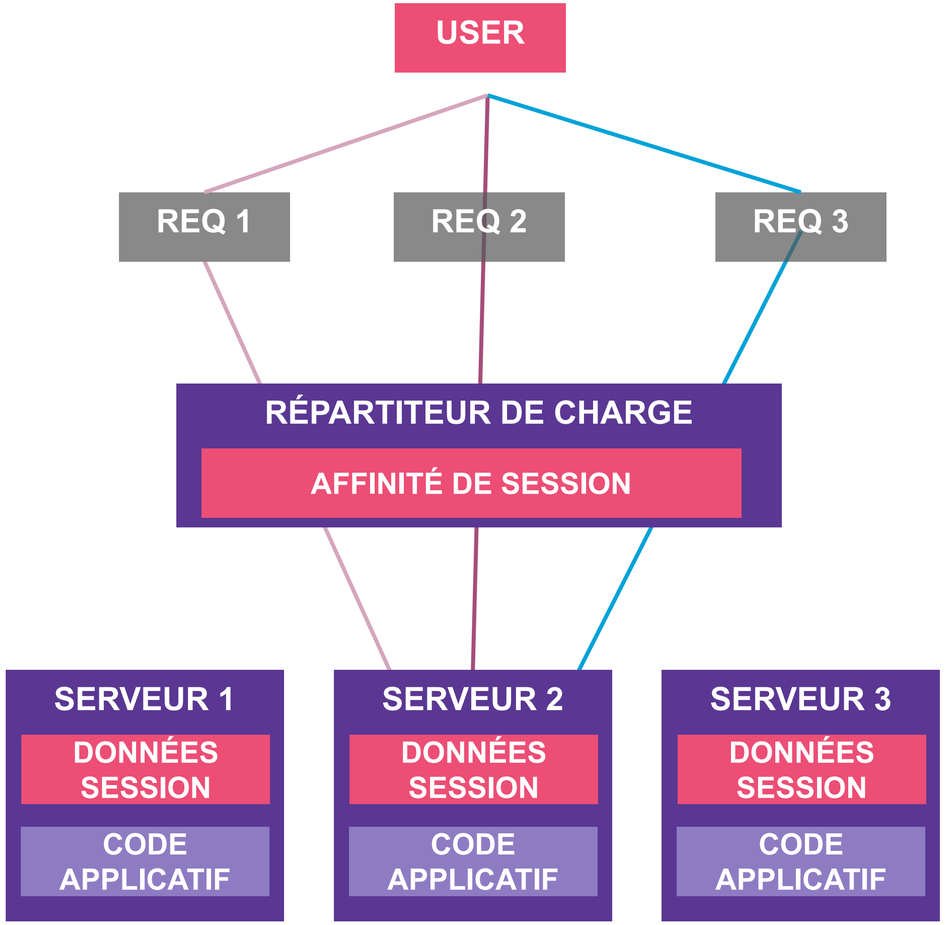
- APPLI / INFRA OBSERVABLES / MESURABLES
- LOGS CENTRALISÉS ET CORRÉLÉS
- INSTRUMENTATION DU CODE
- MONITORING / HEALTHCHECK APPLI
- TABLEAU DE BORD / BAM

DÉPLOIEMENT

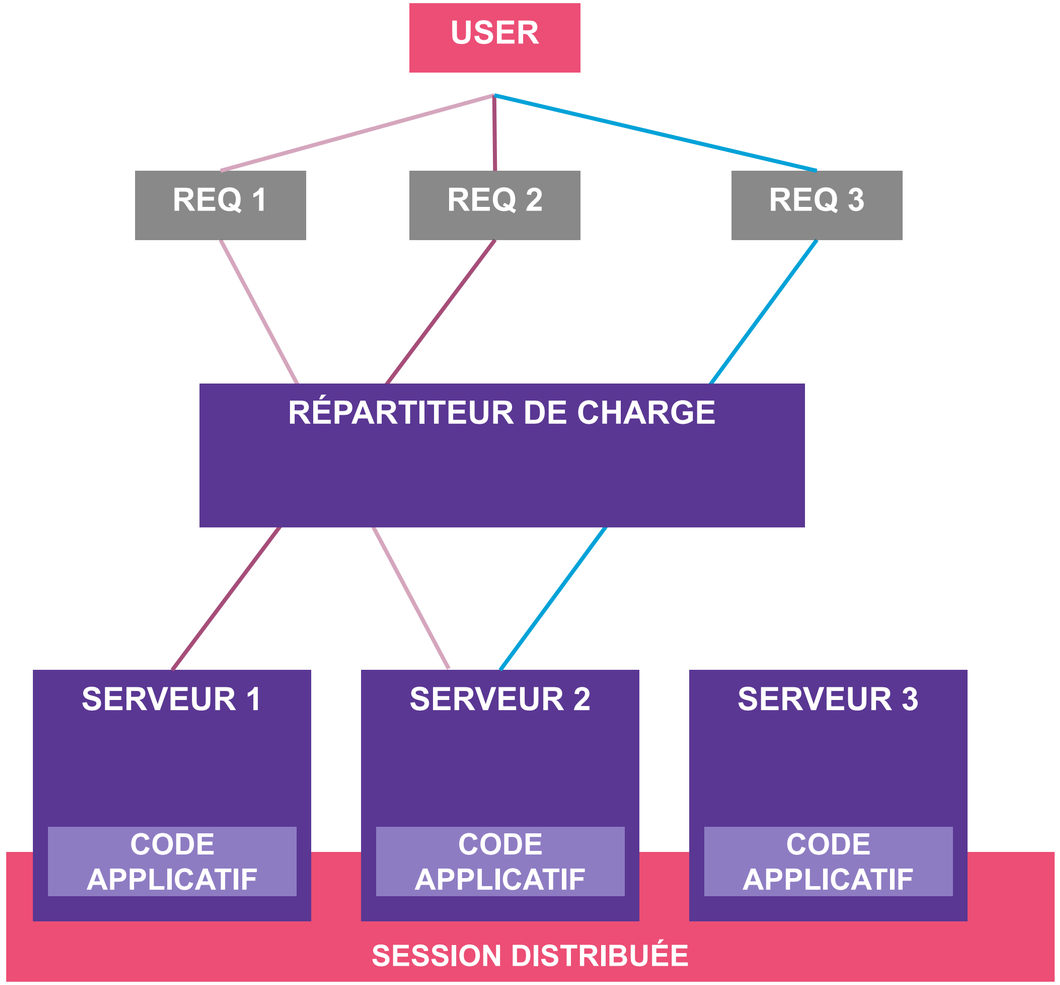
- FEATURE FLIPPING
- BLUE/GREEN DEPLOYMENT
- A/B DEPLOYMENT
- CANARI RELEASE


1
SERVER
2 SERVERS
2
2 DATACENTERS
3
DESIGN FOR FAILURE
@adrienrouge

ARCHITECTURE ET PATTERNS



STATEFUL



STATELESS

ARCHITECTURE ET PATTERNS

Machines virtuelles	→	Conteneurs
Datacenter	→	Orchestrateur de conteneurs
Redondance hardware	→	Design for failure
Serveurs	→	Services
IP Adresse, DNS	→	Service Discovery
Serveur de monitoring	→	Service de monitoring
Application monolithique	→	Microservice
TCP/IP	→	REST

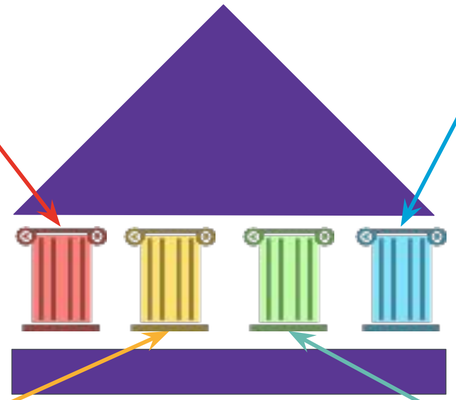
LES 4 PILIERS DU #DEVOPS

Culture, méthode et organisation

Culture, des modèles d'organisation et des méthodes favorisant le dialogue des Dev et des Ops

Architecture et patterns

Architectures et patterns répondant aux préoccupations des Devs et des Ops



Construction et déploiement continu

Outils et processus de construction et déploiement continu

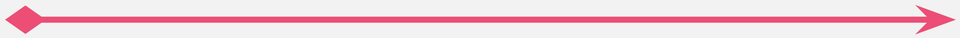
Infrastructure as code "IaC"

Outils d'automatisation pour construire et maintenir l'infrastructure par du code

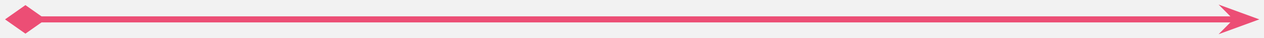
CONSTRUCTION ET DEPLOIEMENT CONTINU

- **L’automatisation du processus logiciel doit emmener le produit jusqu’à son environnement de production**

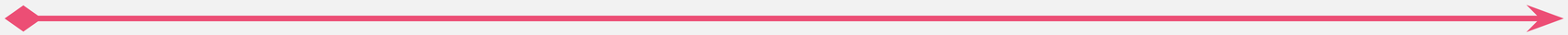
Intégration continue = Continuous Integration



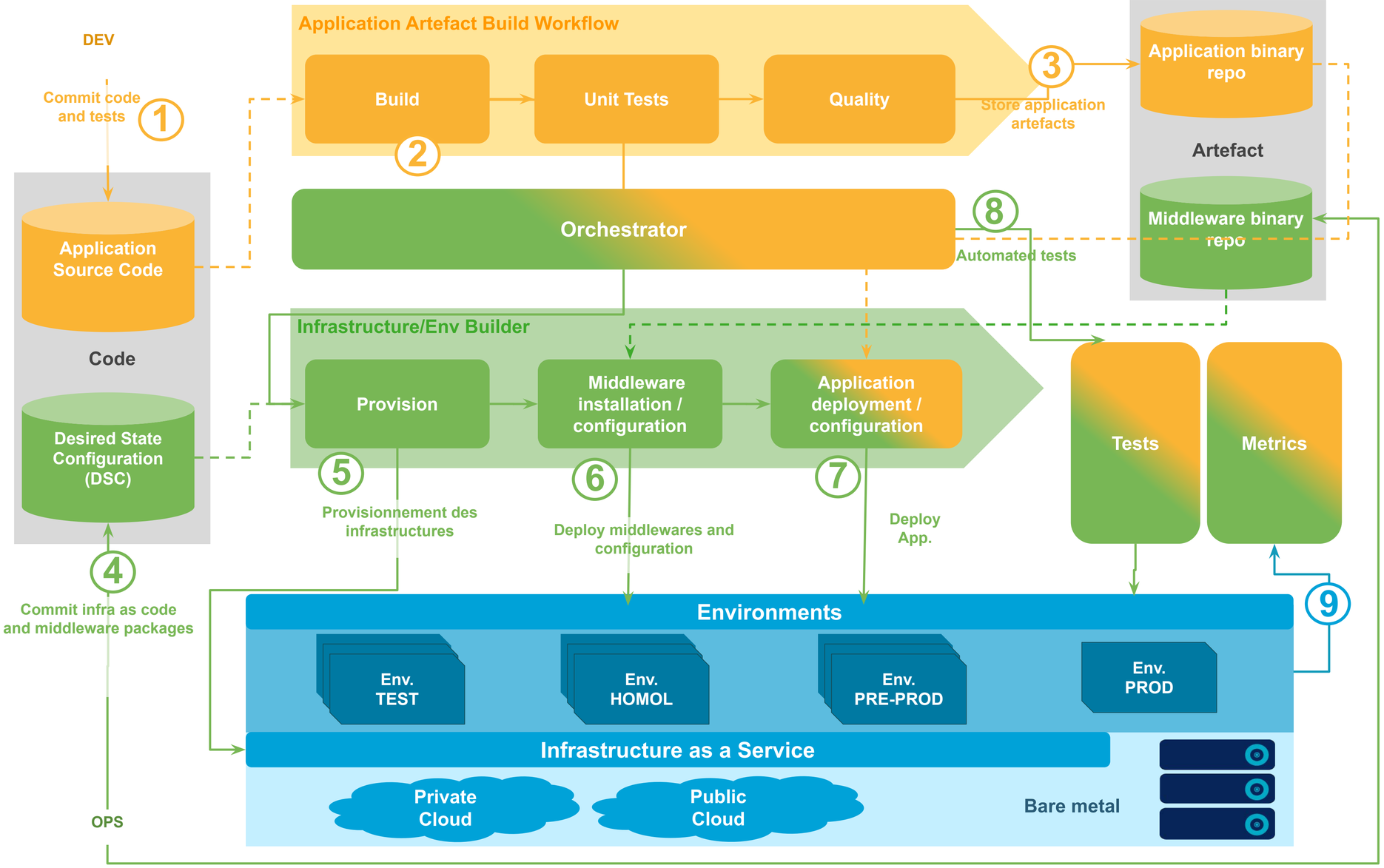
Livraison continue = Continuous Delivery



Déploiement continu = Continuous Deployment



INTÉGRATION ET DÉPLOIEMENT CONTINUS DANS LE IAAS



Cours CI/CD

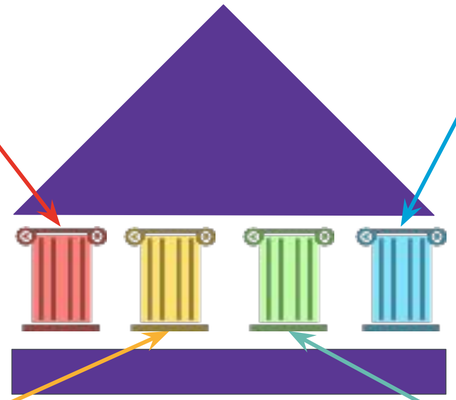
LES 4 PILIERS DU #DEVOPS

Culture, méthode et organisation

Culture, des modèles d'organisation et des méthodes favorisant le dialogue des Dev et des Ops

Architecture et patterns

Architectures et patterns répondant aux préoccupations des Devs et des Ops



Construction et déploiement continu

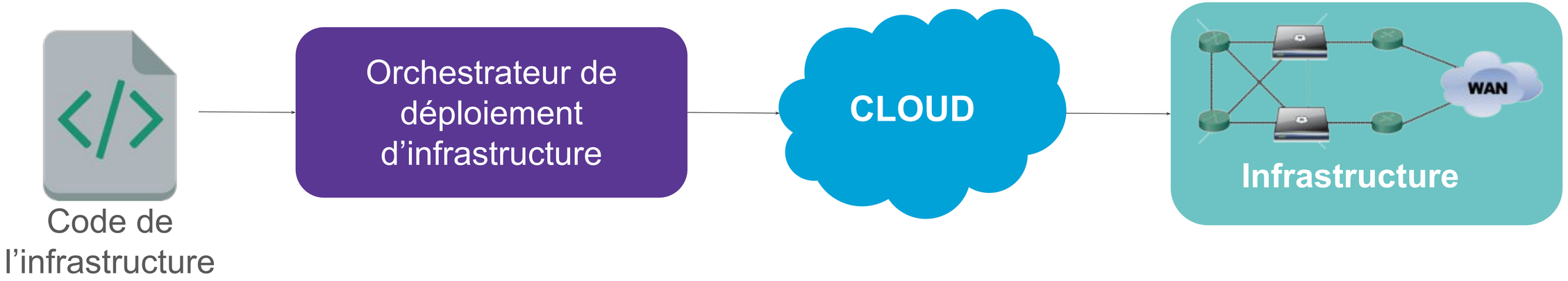
Outils et processus de construction et déploiement continu

Infrastructure as code "IaC"

Outils d'automatisation pour construire et maintenir l'infrastructure par du code

INFRASTRUCTURE AS CODE “IAC”

Décrire sous forme de code exécutable et testable la configuration de l’architecture technique d’infrastructure



INFRASTRUCTURE AS CODE “IAC”

- **Automatiser la construction des environnements et la répétabilité des processus pour :**
  - Garantir une infrastructure homogène
  - Assurer le respect des standards en place
  - Ouvrir l’exécution de certaines tâches aux développeurs
  - Configurer un environnement plus rapidement
  - Avoir un processus déploiement fiable, reproductible et portable

- **Appliquer les pratiques de qualité du monde logiciel à l’infrastructure :**
  - Tests unitaires
  - Qualité de code
  - Code review
  - Gestion de version

- **Gérer un parc important de serveurs avec peu d’administrateurs système**

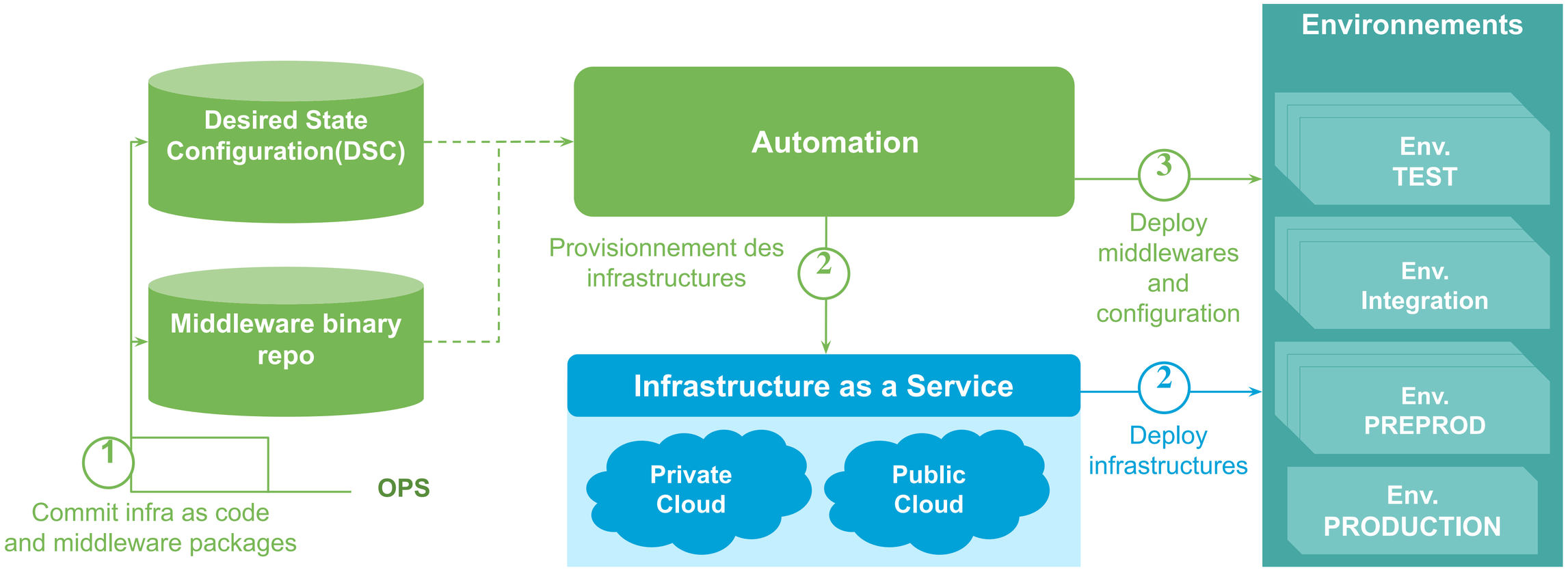
INFRASTRUCTURE AS CODE “IAC” 

- **Etat désiré (DSC : Desired State Configuration) :**
  - Seul l’état désiré est décrit, l’outil calcule comment faire pour atteindre l’état
  - Le code n’est pas impératif mais descriptif

- **Idempotence :**
  - Demander plusieurs fois le même état n’affecte pas le système

- **Conformité assurée :**
  - Audit de la conformité à l’état désiré
  - Correction automatique des écarts

INFRASTRUCTURE AS CODE “IAC”



INFRASTRUCTURE AS CODE “IAC”

Exemples de produits du marché :

Jenkins

Deployit

Capistrano

SERENA

Bamboo

Maven

ca technologies

NOLIO Zero Touch Deployment™

ANSIBLE

puppet labs

CFEngine

OPSCODE Chef

SALTSTACK

Microsoft Azure

vmware

docker

TERRAFORM

openstack™

OVH.COM

rackspace

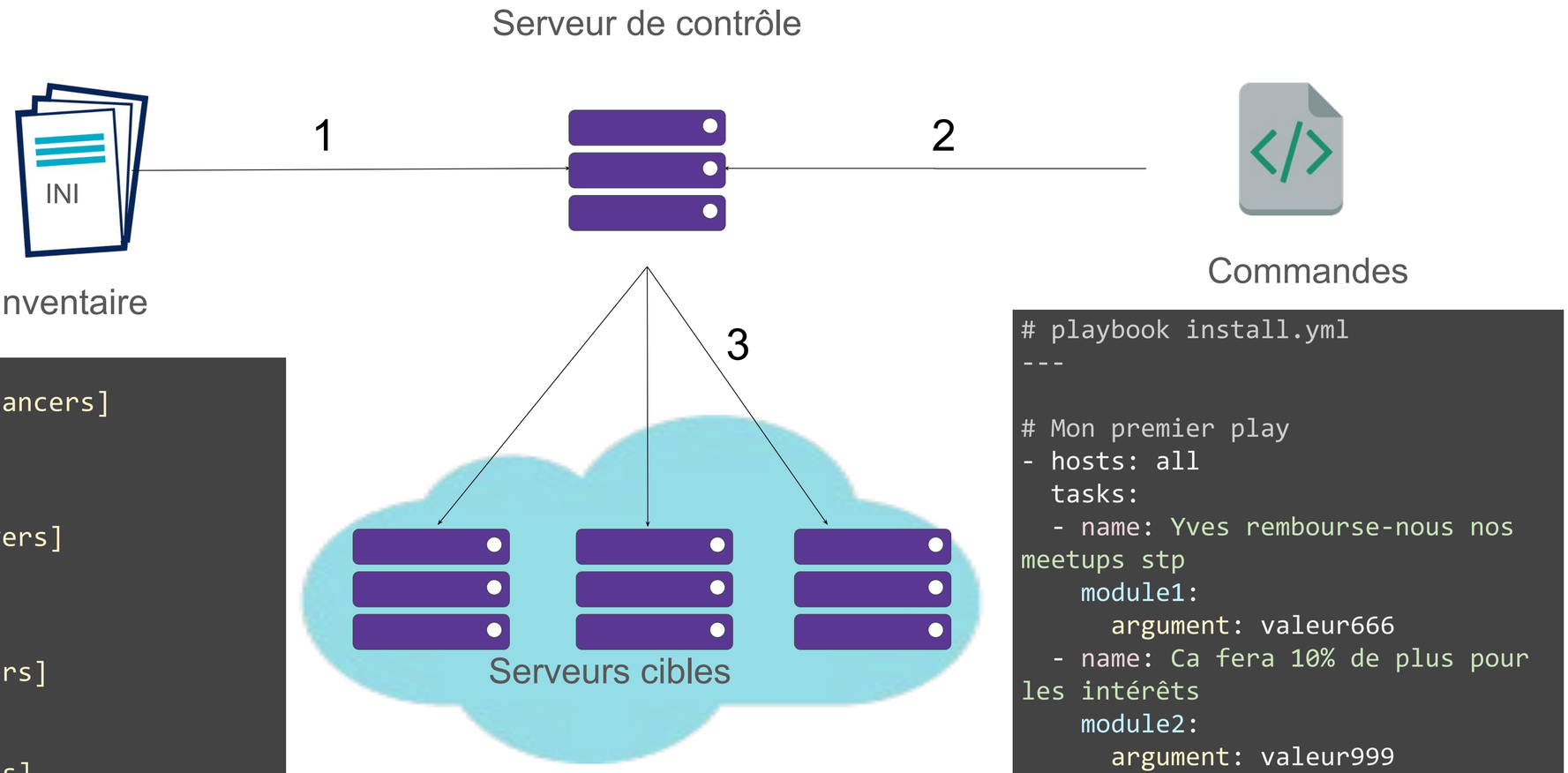
amazon web services

Déploiement / configuration des **Applications**

Installation / configuration des **Middlewares**

Provisioning de l’infrastructure (IaaS/PaaS/CaaS)

INFRASTRUCTURE AS CODE “IAC”




```
[load-balancers]
lb1-srv
lb2-srv

[web-servers]
app1-srv
app2-srv

[db-masters]
db1-srv

[db-slaves]
db2-srv

[db-servers:children]
db-masters
db-slaves
```

```
# playbook install.yml
---

# Mon premier play
- hosts: all
  tasks:
  - name: Yves rembourse-nous nos
meetups stp
    module1:
      argument: valeur666
  - name: Ca fera 10% de plus pour
les intérêts
    module2:
      argument: valeur999

# Un autre play
- hosts: web-servers:db-masters
  tasks:
  - name: DevOps c’est la champions
league
    module3:
      argument: valeur666999
```

INFRASTRUCTURE AS CODE “IAC”

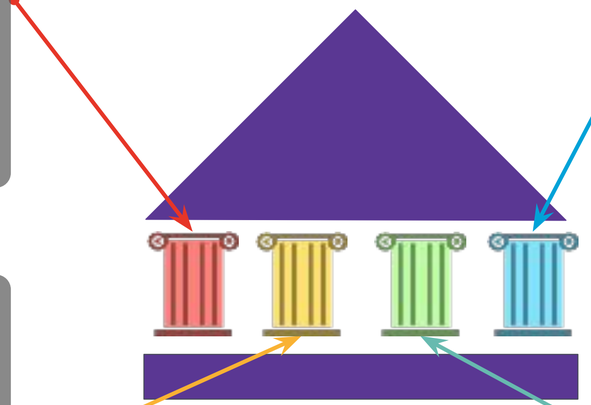
```
# Configure server front
---- host: Web-Server
  tasks :
  - name: Ensure apache server is installed
    package:
       name: ”apache”
       state: ”present”

  - name: Ensure if apache is started and enabled
    service:
       name: “apache”
       state: “started”
       enabled: true
```

INFRASTRUCTURE AS CODE “IAC” 

Cours TERRAFORM

Cours ANSIBLE

RAPPEL : LES 4 PILIERS DU #DEVOPS



Culture, méthode et organisation

Culture, des modèles d'organisation et des méthodes favorisant le dialogue des Dev et des Ops

Architecture et patterns

Architectures et patterns répondant aux préoccupations des Devs et des Ops

Construction et déploiement continu

Outils et processus de construction et déploiement continu

Infrastructure as code "IaC"

Outils d'automatisation pour construire et maintenir l'infrastructure par du code

“Introduction Cloud”

Les 9 caractéristiques du Cloud Computing

« API first » et self-service accessible depuis les outils du développeur (SDK, API, CLI, Portail)

Services d'infrastructure standardisés et rationalisés

« Bêta perpétuelle » et gestion forcée de l'obsolescence

Paiement à l'utilisation (à l'heure et la quantité de ressources)

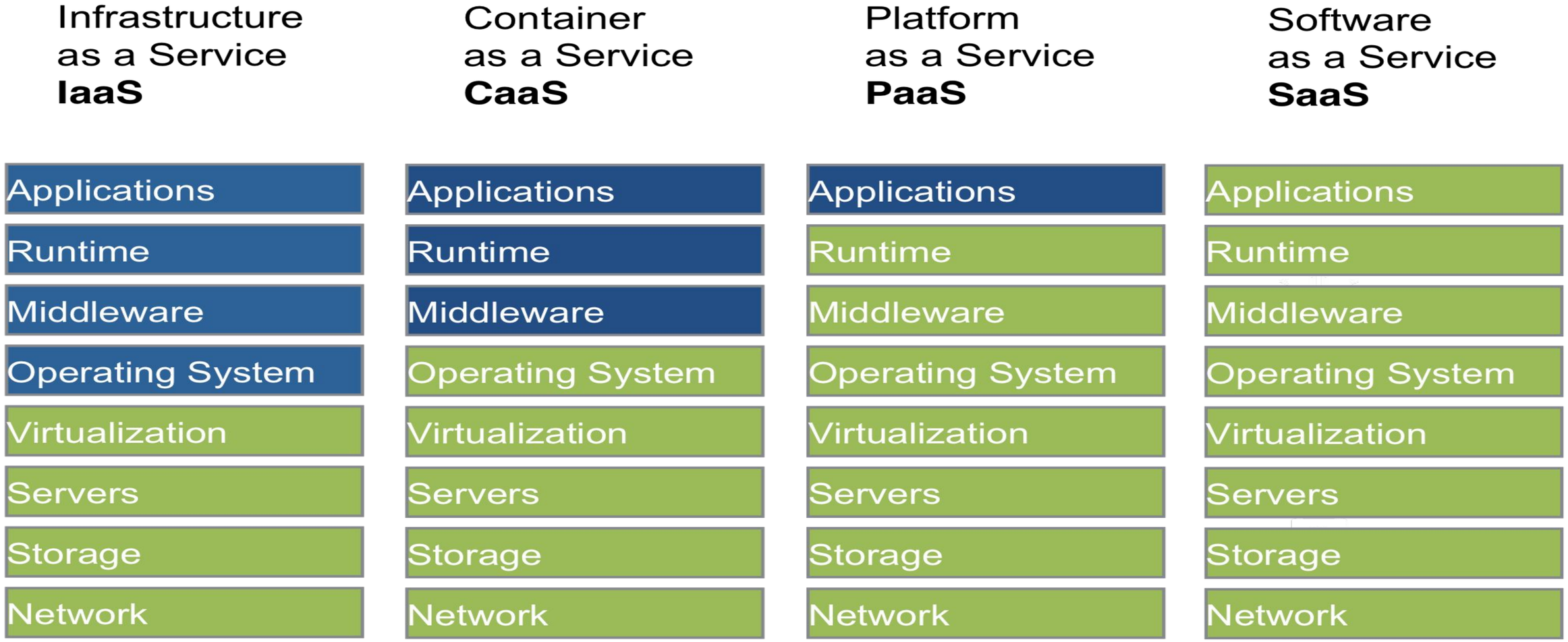
Elasticité et capacité infinie

Offre de services composable

Responsabilités partagées (gestion de la capacité, mise à jour, sécurité, etc.)

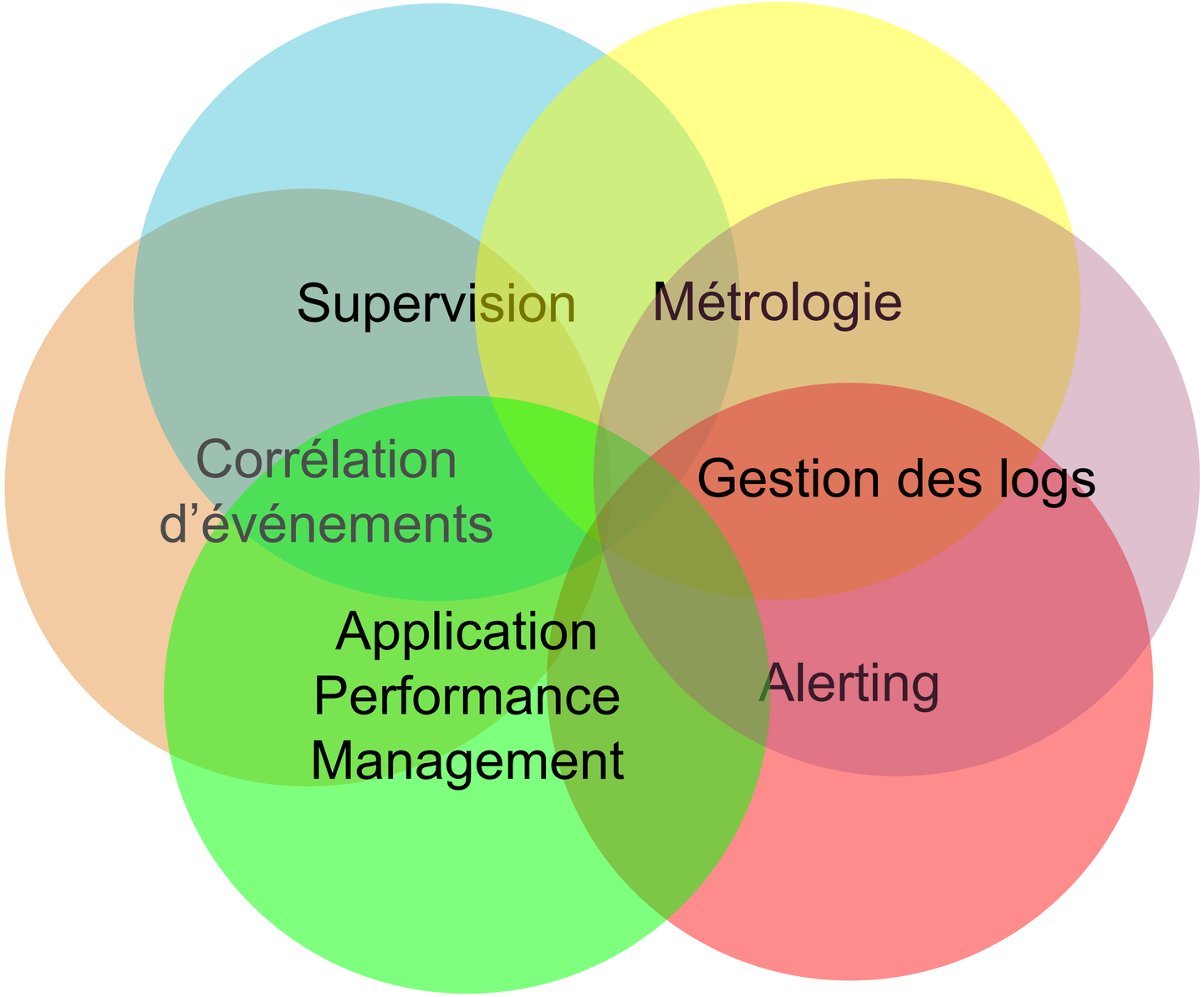
Réactif : disponible en quelques minutes

Multi-tenant et isolation des ressources


Infrastructure as a Service
IaaS
Container as a Service
CaaS
Platform as a Service
PaaS
Software as a Service
SaaS
Applications
Runtime
Middleware
Operating System
Virtualization
Servers
Storage
Network
Applications
Runtime
Middleware
Operating System
Virtualization
Servers
Storage
Network
Applications
Runtime
Middleware
Operating System
Virtualization
Servers
Storage
Network
Applications
Runtime
Middleware
Operating System
Virtualization
Servers
Storage
Network

“ Monitoring ”

QU’EST-CE QUE LE MONITORING ?



QU'EST-CE QUE LE MONITORING ?

- On n'améliore que ce que l'on mesure

- On a (parfois) besoin de se faire réveiller la nuit (prévenez votre copine/copain ou ça peut mal finir)

- On a toujours besoin de prendre le pouls du système (ce qu'on appelle aussi healthcheck)

- On a besoin de voir ce qu'il s'est passé
  - Pour comprendre ce qui a raté (encore la faute des dévs)
  - Pour anticiper ce qui va se produire (ça c'est nous #Ops <3)

- Monitorer permet des choses automagiques comme :
  - Auto-scaling
  - Self-healing

BRAINSTORM TIME !

(**Proposer des indicateurs qui vous paraissent pertinents sur un site de eCommerce**)

TYPES D'INDICATEURS

- Matériel (physique)
- Système
- Applicatif
- Fonctionnel
- Financier
- Sécurité

BEAUCOUP ~~TROP~~ DE SOLUTIONS DISPONIBLES

LA BONNE HYGIÈNE DU MONITORING

Risque d'un monitoring qu'on ne maintient pas :

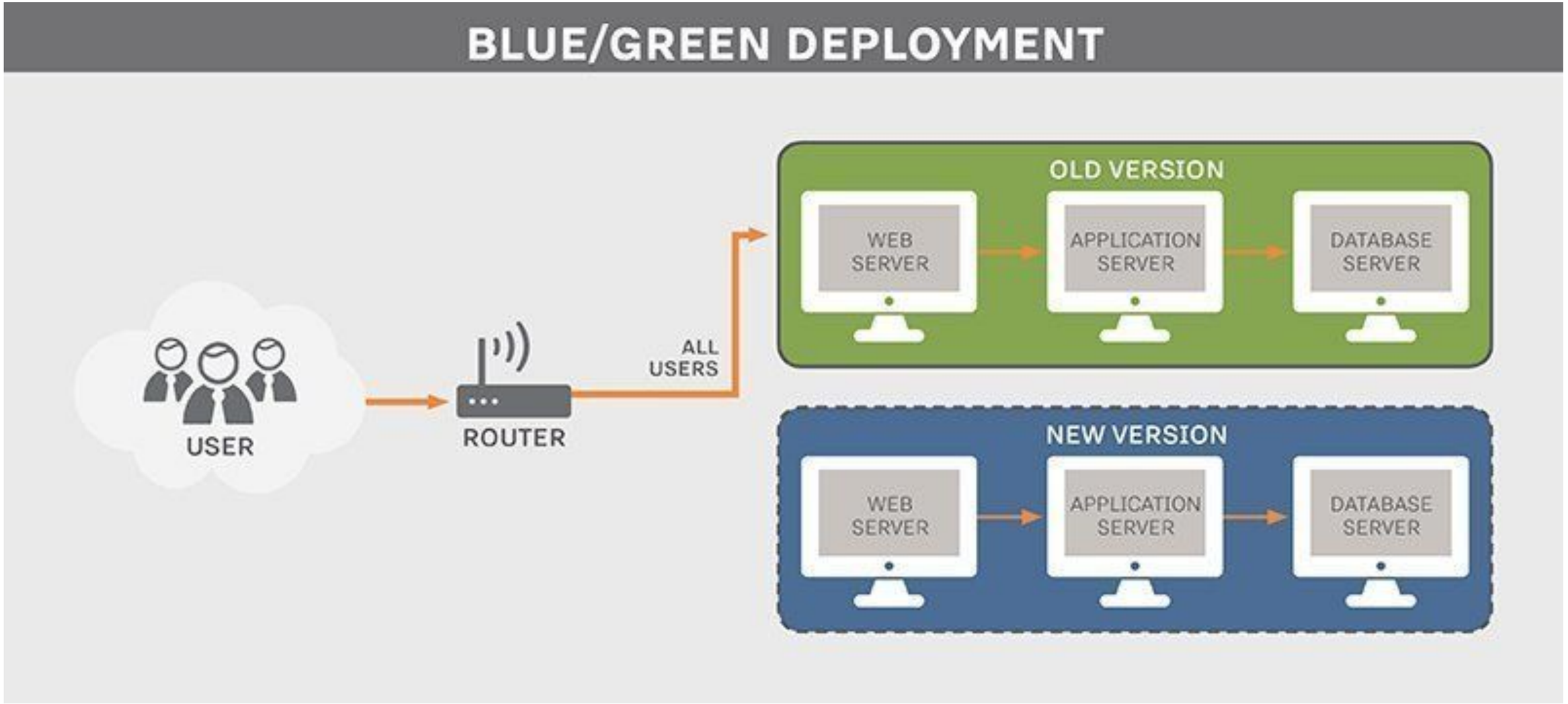
- Trop de données collectées inutiles
  => **Gâchis de stockage**

- Trop d'alerte dont certaines à ignorer
  => **Les vrais incidents passent inaperçus** dans la masse

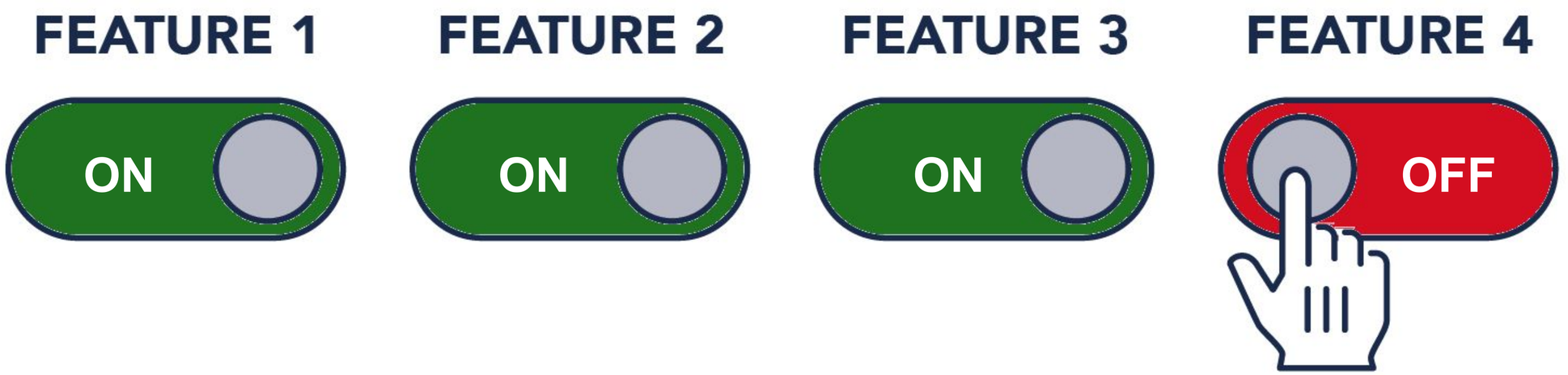
- Trop de fausses alertes ou avec des seuils mal réglés
  => **Astreintes inutiles**, manque de sommeil, anxiogène pour le projet

“ Haute dispo (ZDD) ”

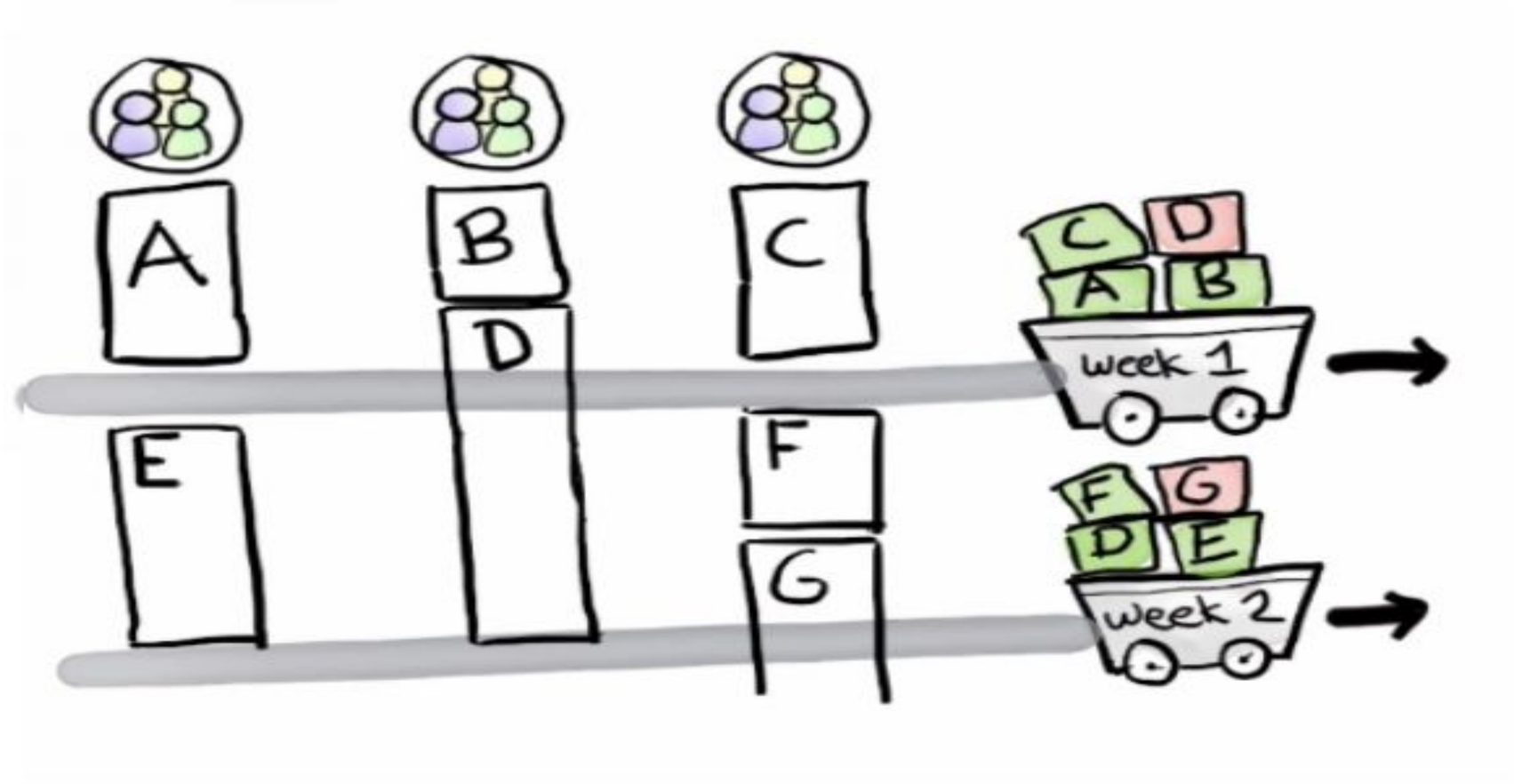
BLUE / GREEN DEPLOYMENT

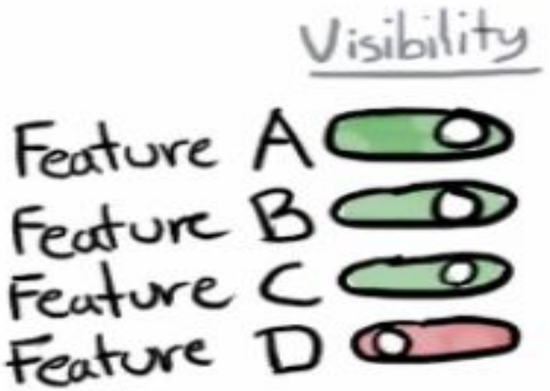


FEATURE FLIPPING



FEATURE FLIPPING





Source : The Unproject Culture, Spotify

VERTICAL / HORIZONTAL SCALING