Introduction aux métiers de l'intégration

Université de Nantes
M1 MIAGE 2024-2025

Bertrand Florat

Architecte Solution et intégrateur

bertrand@florat.net

Avis / suggestions

Objectifs du cours

- Expliquer le **rôle d’un intégrateur**
- Présenter les **différents métiers de l’intégration**
- Présenter l’**écosystème actuel**
- Fournir quelques **patterns** et **anti-patterns**

Agenda (à prioriser)

- **Le rôle de l'intégrateur**
- Les usines logicielles
- Les conteneurs
- La démarche DevOps
- Les documents pour l'exploitant
- La gestion d'un test de charge

Thèmes au choix

https://forms.gle/bwyLjYf6WhJLZjFa7

Le rôle de l’intégrateur

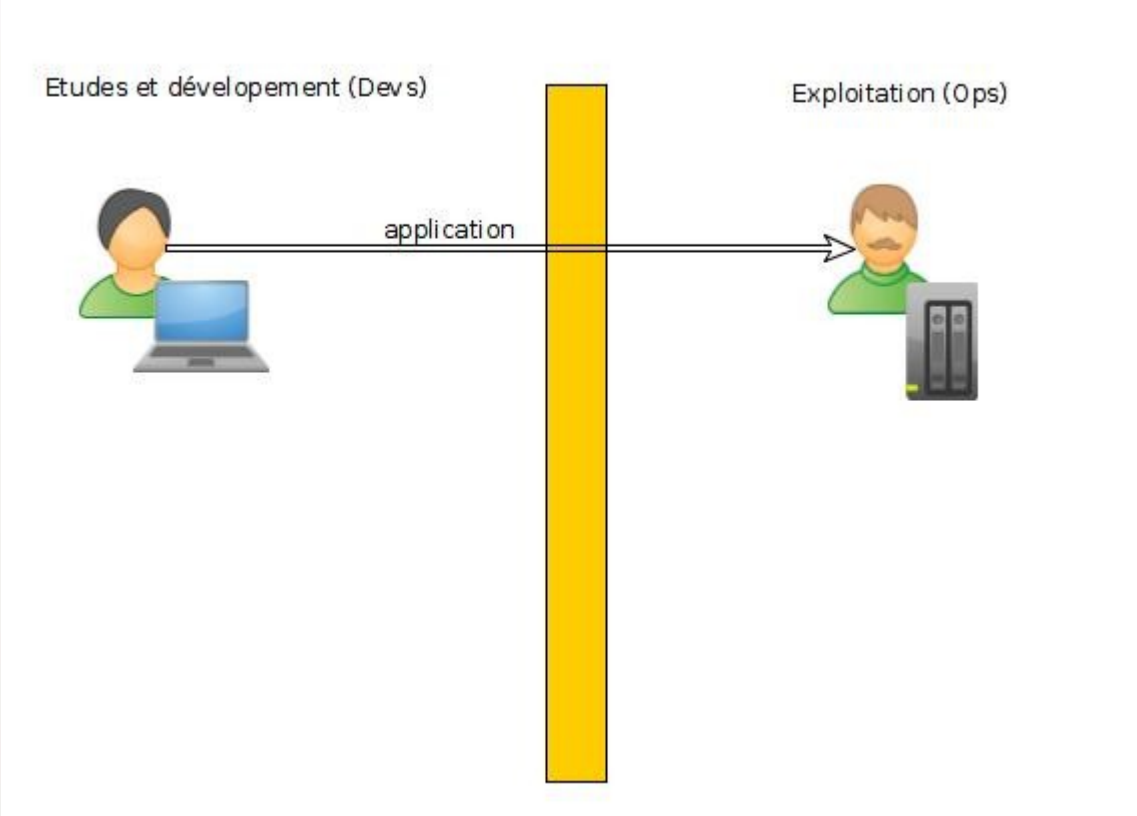
“

Instancie l’architecture en quelque chose d’utilisable

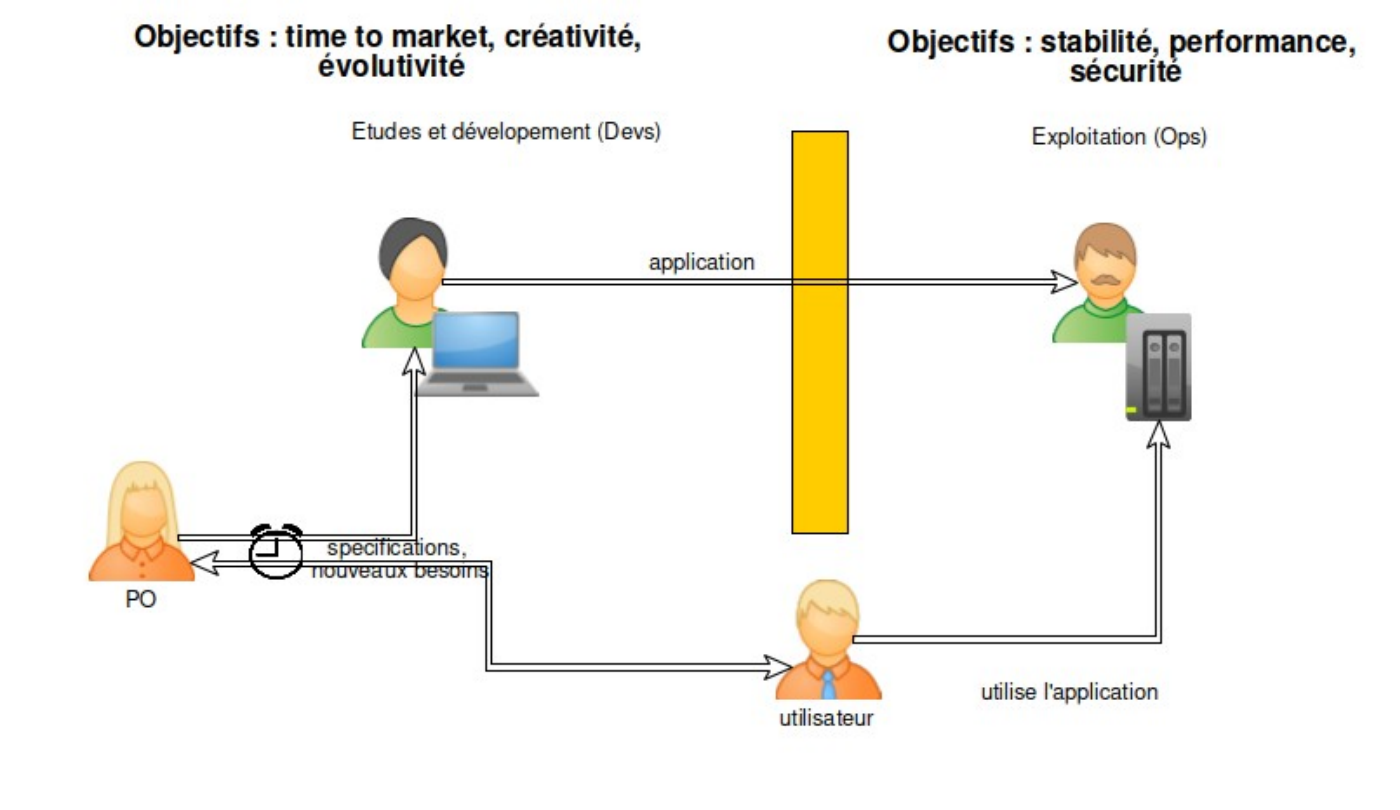
et c’est difficile ...

celui qui fait que tout marche ensemble

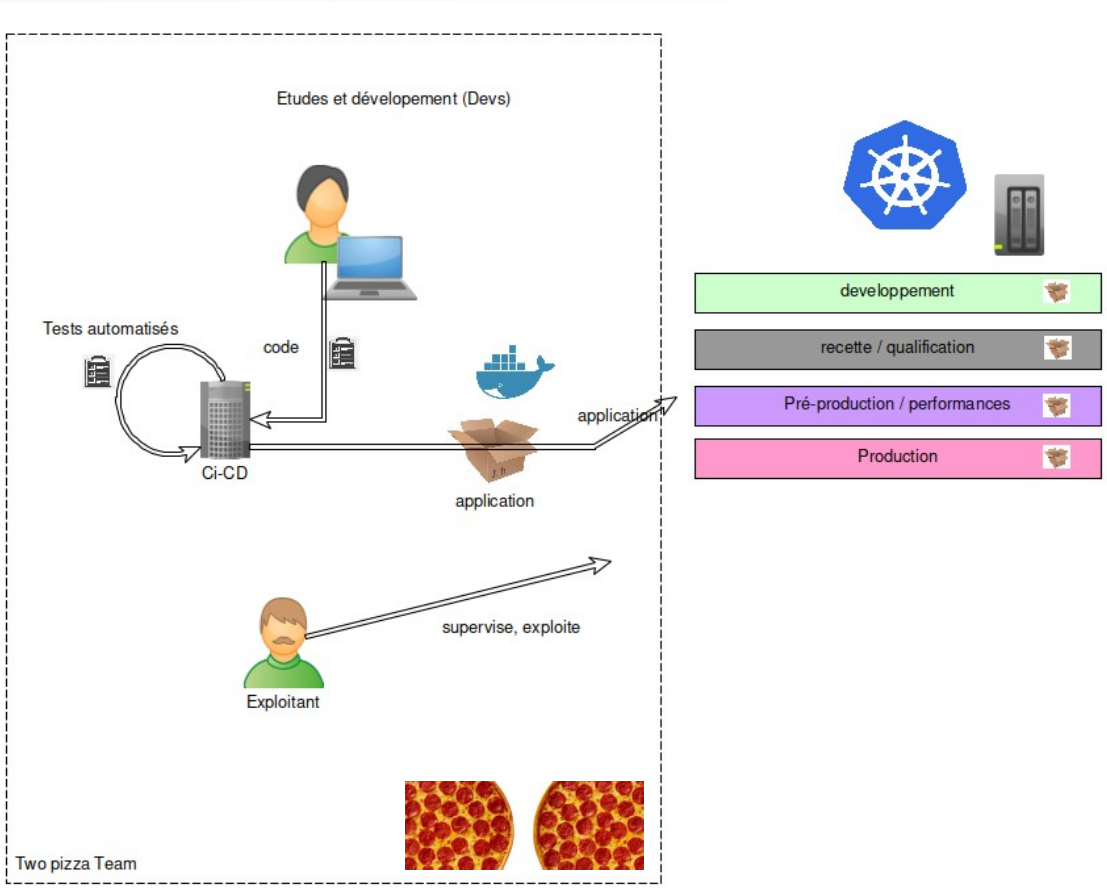
Un peu de contexte: les Devs et les Ops...



...ne parlent pas la même langue et ne partagent pas les mêmes objectifs:

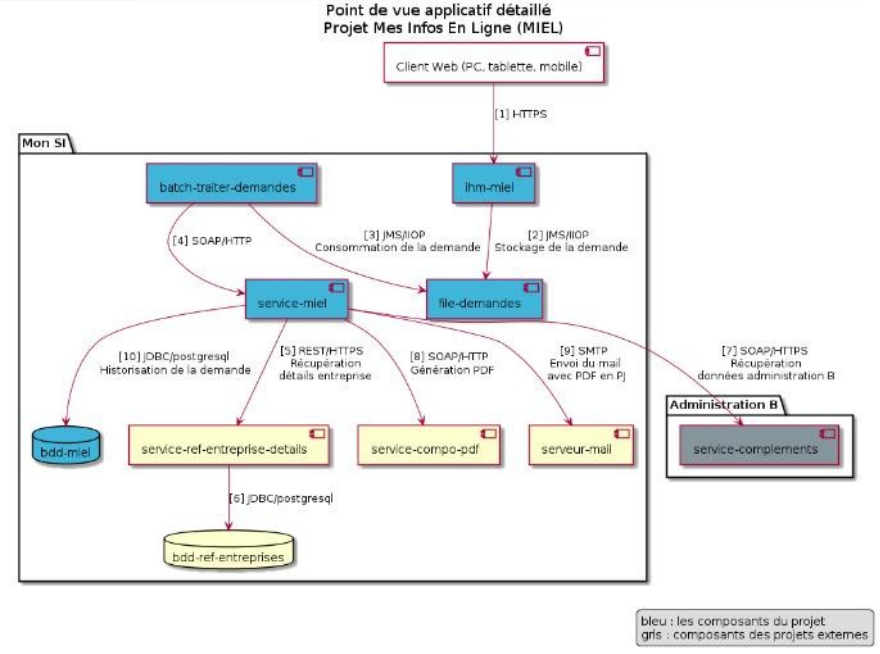


Mieux : l'approche DevOps

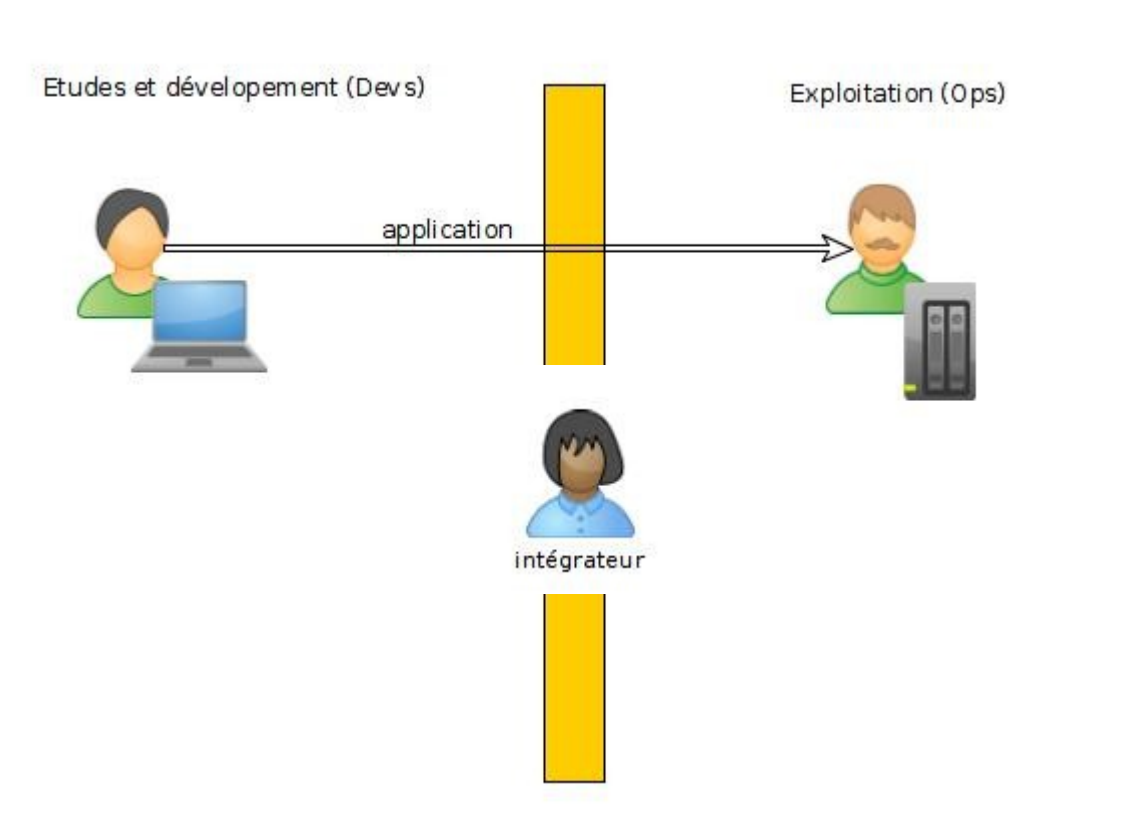


Architectures de plus en plus distribuées et complexes
SOA, micro-services

- Beaucoup de paramétrage
- Intégration complexe
- Besoin de coordination entre projets ou composants

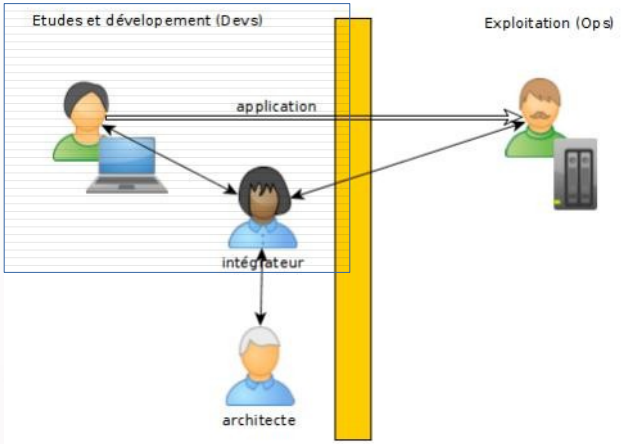


L'intégrateur.rice : interface entre les Devs et les Ops



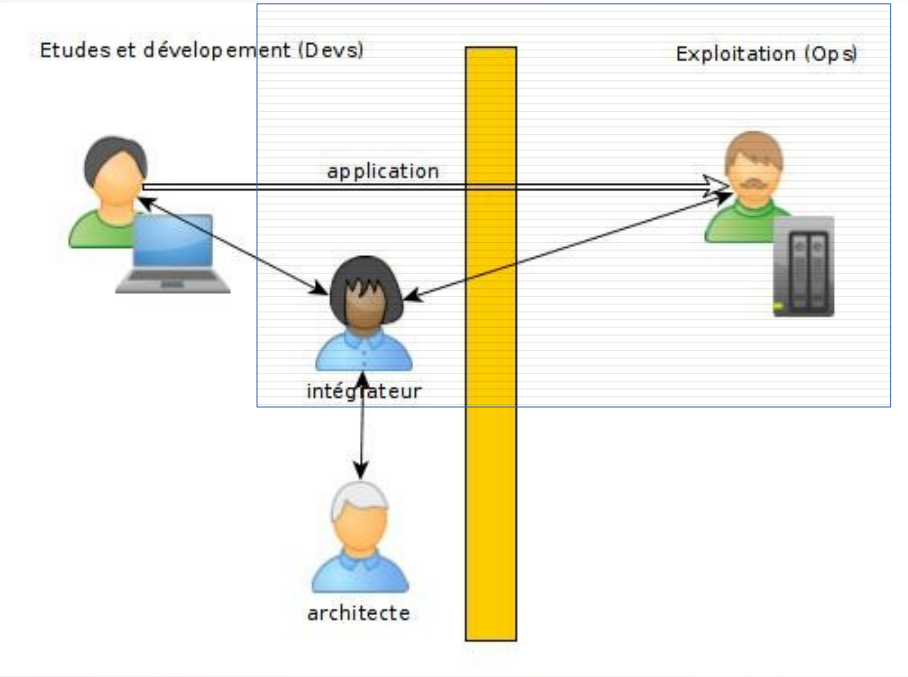
les principaux rôles de l'intégrateur
Face aux Devs

- Industrialisation développement via la **CI-CD**
  - Build (Maven, Gradle, npm, pip...)
  - Intégration continue
  - Déploiement/Publication continue
  - Inspection continue
- **Paramétrage** applicatif
- **Exploitation** DEV
- Groupware



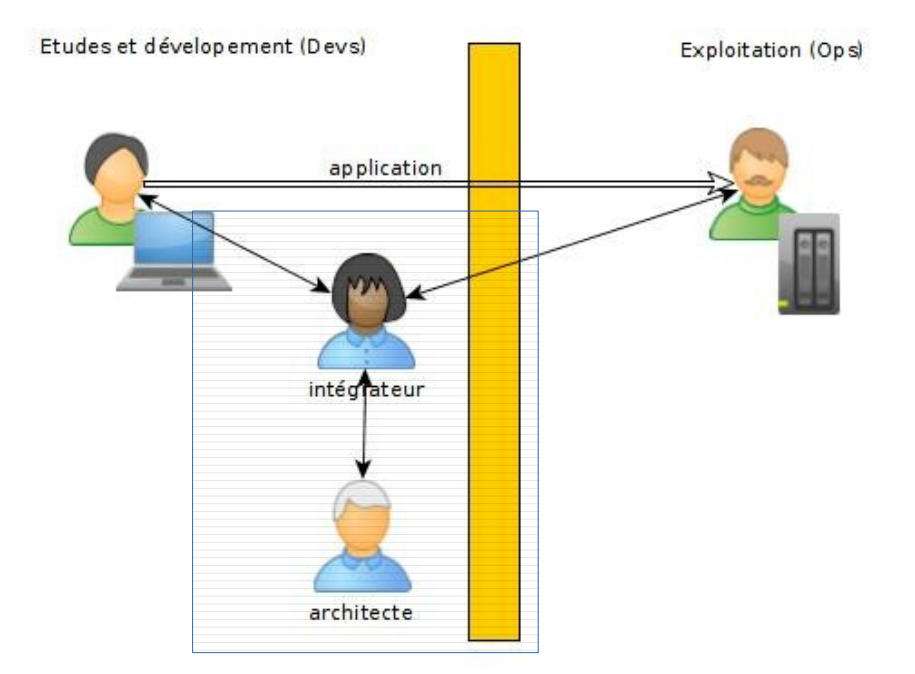
Face aux Ops

- **IaC** (Infrastructure As Code)
- Préparation **plan de production**
- Rédaction du **DEX** (Dossier d'EXploitation)
- Support de second niveau en cas de problèmes en PROD
- Rédaction du guide d'installation (si pas d'IaC ou partielle)



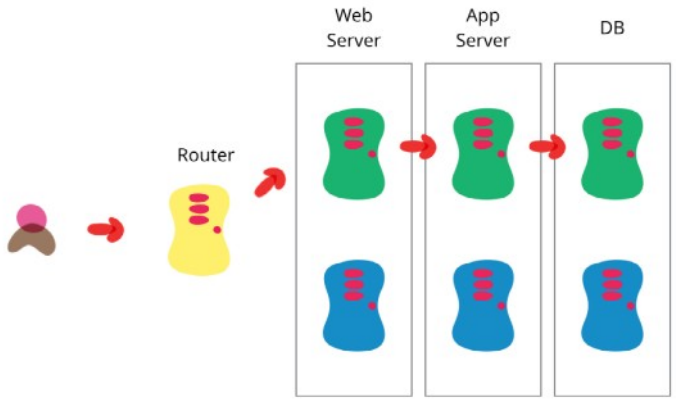
Avec les architectes

- Élaboration du **modèle de déploiement** (clusters...)
- Préparation/exécution **tests de charge**
- **Tuning/configuration** des composants d'infrastructure
- Intégration de **progiciels** dans le SI



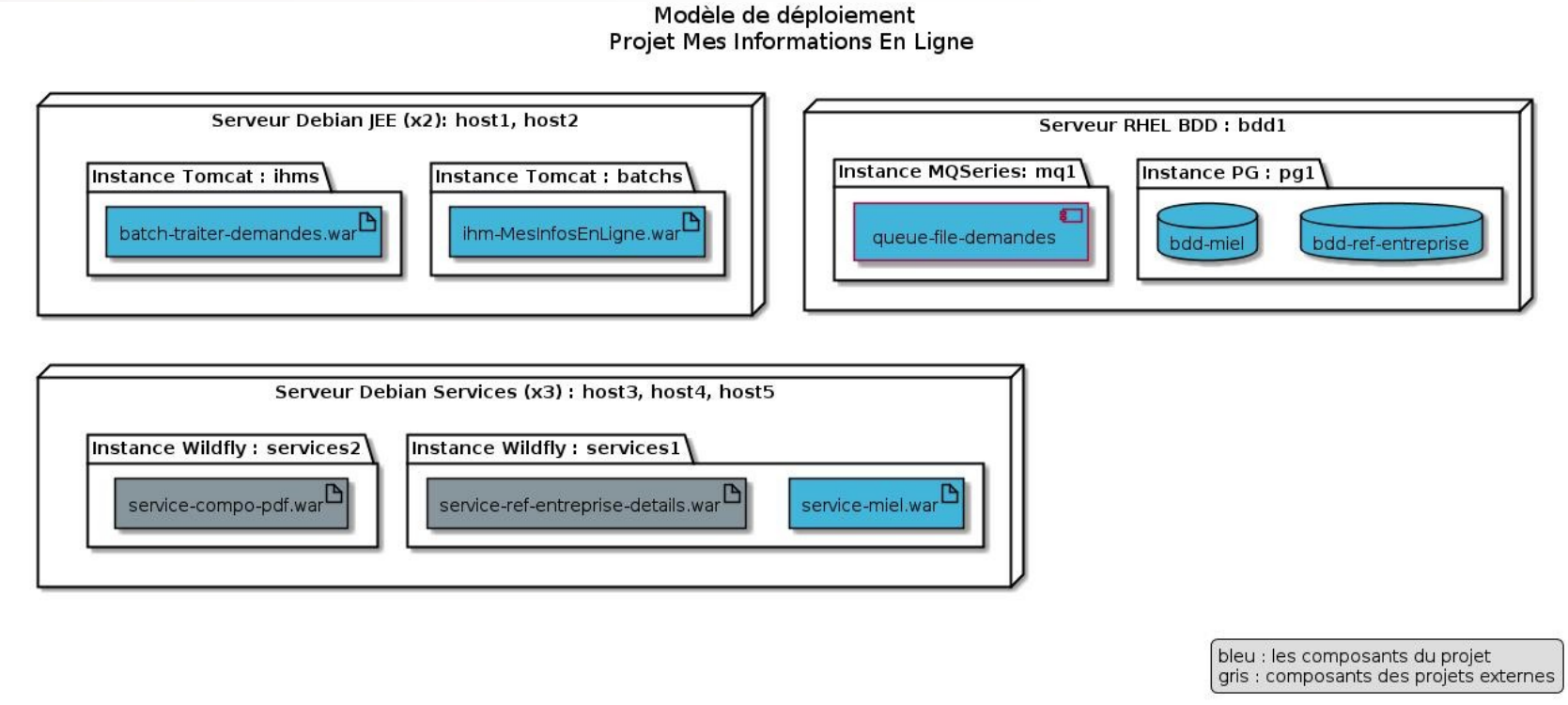
Utilisation de stratégies de déploiement avancées

- **Blue-Green**
  - Tout l'un ou tout l'autre
- **TBD (Trunk Based Dev.)**
  - Plus de branches de maintenance des anciennes versions supportées.
  - Utilisation des Feature Flags
- **Canary testing**
  - 1 % en B, 99 % en A
- **Rolling update** (orchestrateur de conteneurs)
  - 100% A, 0 % B -> 60 % A, 40 % B -> 100 % A, 0 % A

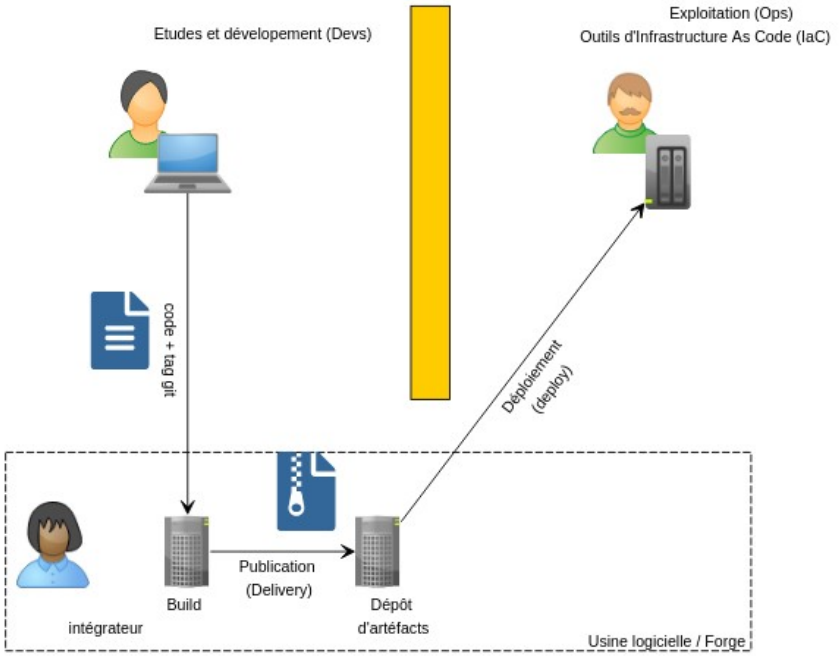


crédit : Martin Fowler, https://martinfowler.com/bliki/BlueGreenDeployment.html

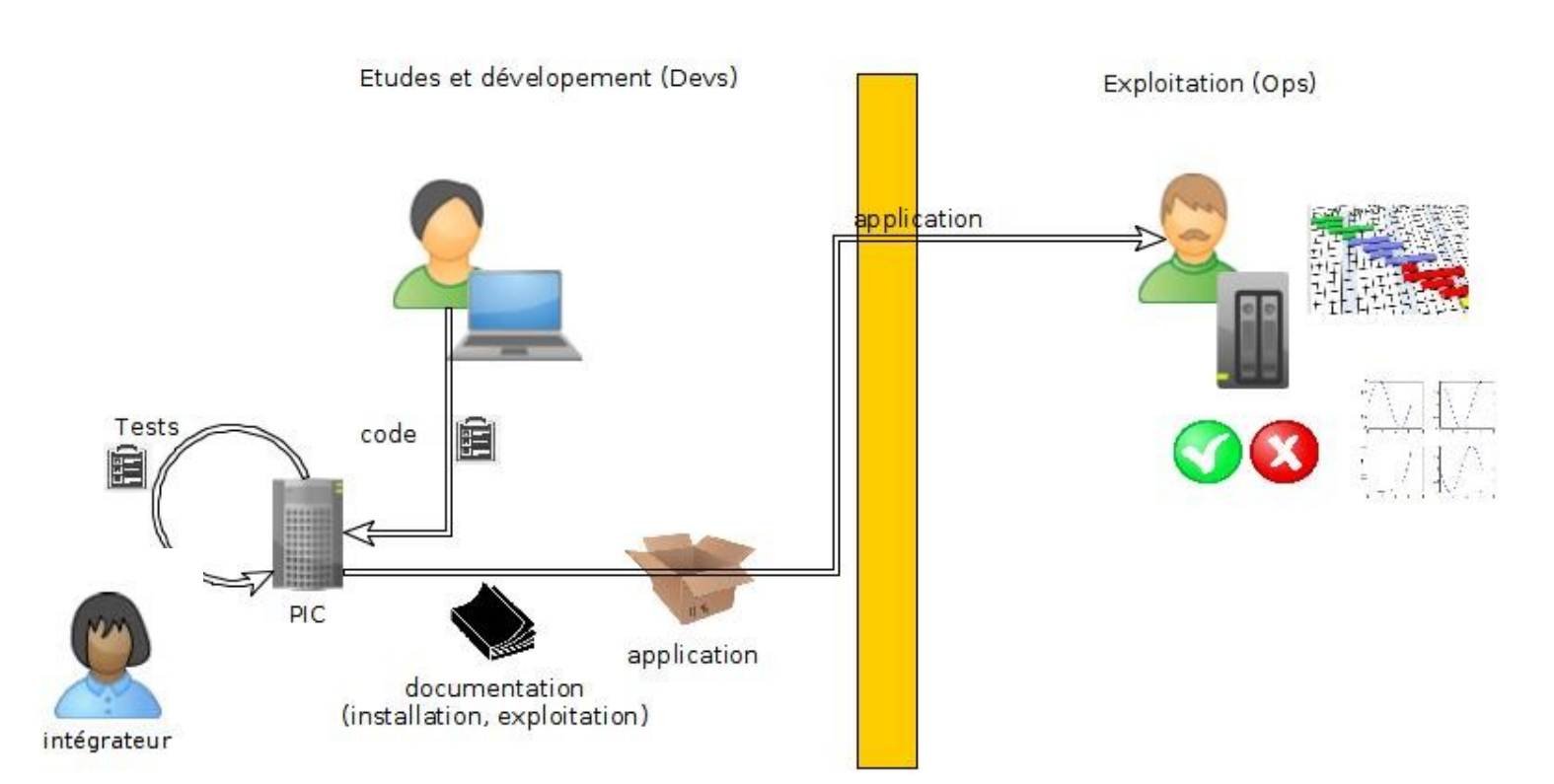
Définition des modèles de déploiement par environnement



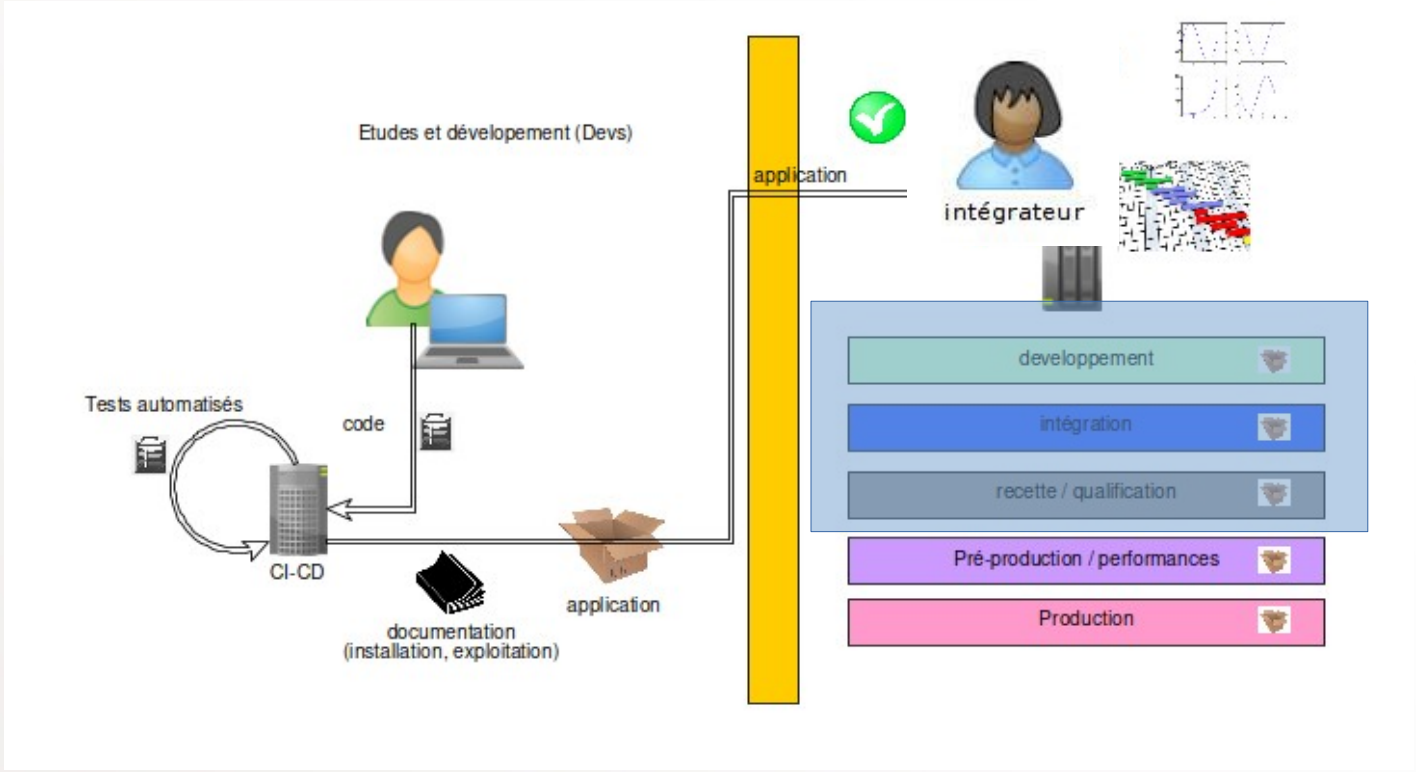
Gestion de l’usine logicielle



Documentation



Administration des environnements jusqu'à la PRE-Production



Un domaine, plusieurs métiers/rôles (1/4)

→ Intégrateur applicatif

Objectif principal : Assurer que les dépendances sont compatibles, à jour et sûres

- **Assemble** les applications et leurs dépendances (Maven, npm, gradle, Pypi, etc.)
- Gère les **conflits** et contient la **cohérence globale**.
- Analyse sécurité (Software Component Analysis - **SCA**)
- **Documentation**

Proche du DEV ? Très proche (+++)
Proactif/réactif ? : Proactive

Ce rôle peut être exercé par des DEV expérimentés

Un domaine, plusieurs métiers/rôles (2/4)

→Intégrateur d'Exploitation / SysOps

Objectif principal : **Assurer le bon fonctionnement quotidien** des systèmes informatiques

- **Surveillance** et gestion des systèmes et des applications
- **Résolution des incidents** et des problèmes techniques.
- Mise en œuvre des **mises à jour et des correctifs**.
- Lancement opérations manuelles d'IaC (Infra As Code).
- Gestion des **sauvegardes** et des restaurations.
- **Planification** avec équipes DEV du déploiement des applications.
- **Documentation**

Proche du DEV ? Moyennement proche (+)

Proactif/réactif ? : **Réactif**

OVHcloud

elasticsearch

Un domaine, plusieurs métiers/rôles (3/4)

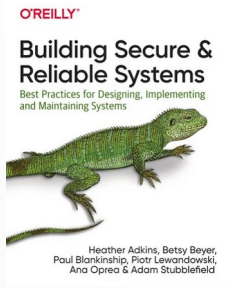
→SRE (Site Reliability Engineer)

Objectif principal : **Améliorer la fiabilité des systèmes** et des applications via des pratiques d'ingénierie logicielle.

- **Développer des outils** de supervision / alerting / automatisation.
- **Fiabiliser** le SI via l'IaC.
- Configurer environnements pour **robustesse** et **résilience**.
- Tests **résilience / chaos engineering**.
- Optimiser les performances et effectuer des benchmarks.
- Proposer aux DevOps les **bonnes pratiques IaC**
- **Écrire les post-mortems**
- **Documentation**

Proche du DEV ? Moyennement proche (+)

Proactif/réactif ? : **Proactif +++**



https://sre.google/books/

Un domaine, plusieurs métiers/rôles (4/4)

→DevOps

Objectif principal : Améliorer la **collaboration entre les équipes de développement et d'opérations** (SysOps, SRE…) pour accélérer le cycle de développement et de déploiement logiciel .

- **Automatisation** des processus de développement, de test et de déploiement (**CI/CD**).
- Écriture **IaC** (Kubernetes, Ansible, Terraform…)
- Aide les DEV à intégrer les **bonnes pratiques d'infra** (ex : Cloud Native, redondance, rejeux, etc.)
- Mise en place d'**amélioration continue.**
- **Documentation**

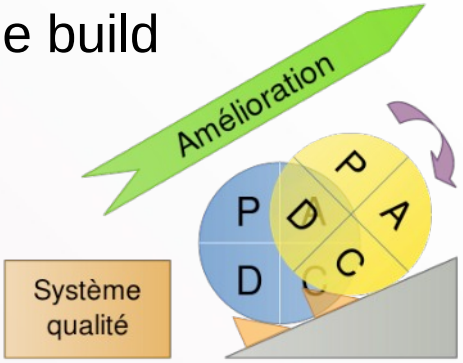
Proche du DEV ? Très proche (++)
Proactif/réactif ? : Proactif ++

Des métiers en pleine croissance

- **Non délocalisable** :
  - Développeurs remplaçables par IA/progiciel/SaaS
  - Sysadmins remplaçables par du cloud
  - MAIS toujours besoin de SysOps / DevOps / SRE
- Idéalement, **être multi-rôles**
  - Vision globale du code à la production
  - Alter ego de l’Architecte Solutions

Patterns

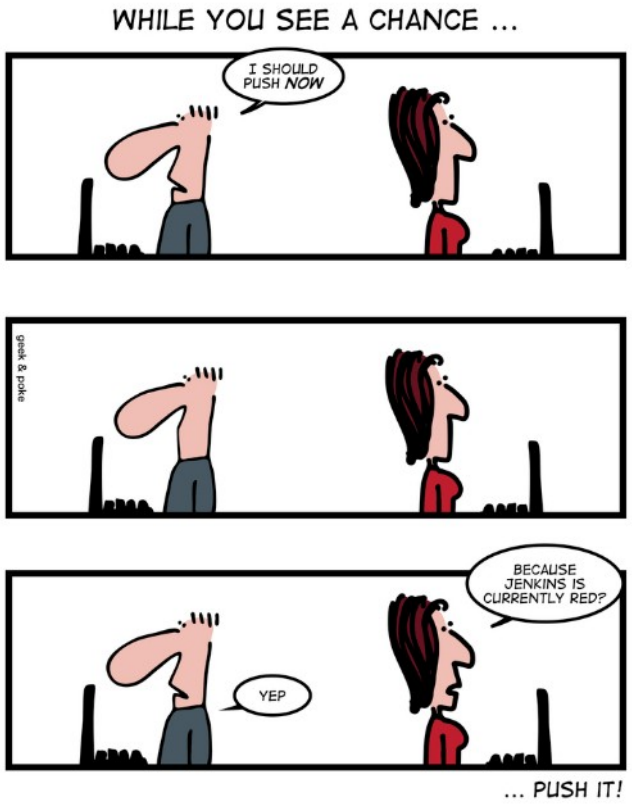
- **GitOps** (si ce n'est pas dans Git, ça n'existe pas)
- Convention over configuration (**on rails**)
- Jamais de configuration plate-forme dépendante dans les paquets applicatifs
- **Tout automatiser**, éliminer toute intervention humaine
- **Build au commit**
- **Feedback** le plus rapide possible, accélérer le build
- **TBD: une seule branche** si possible
  - (sauf branches topic pour les **Merge Requests**)
- Utiliser un **outil** d'IT Automation
  - Ansible (pour la configuration),
    Terraform (pour le provisionnement de ressources), ...



Roue de Deming, source : wikipedia

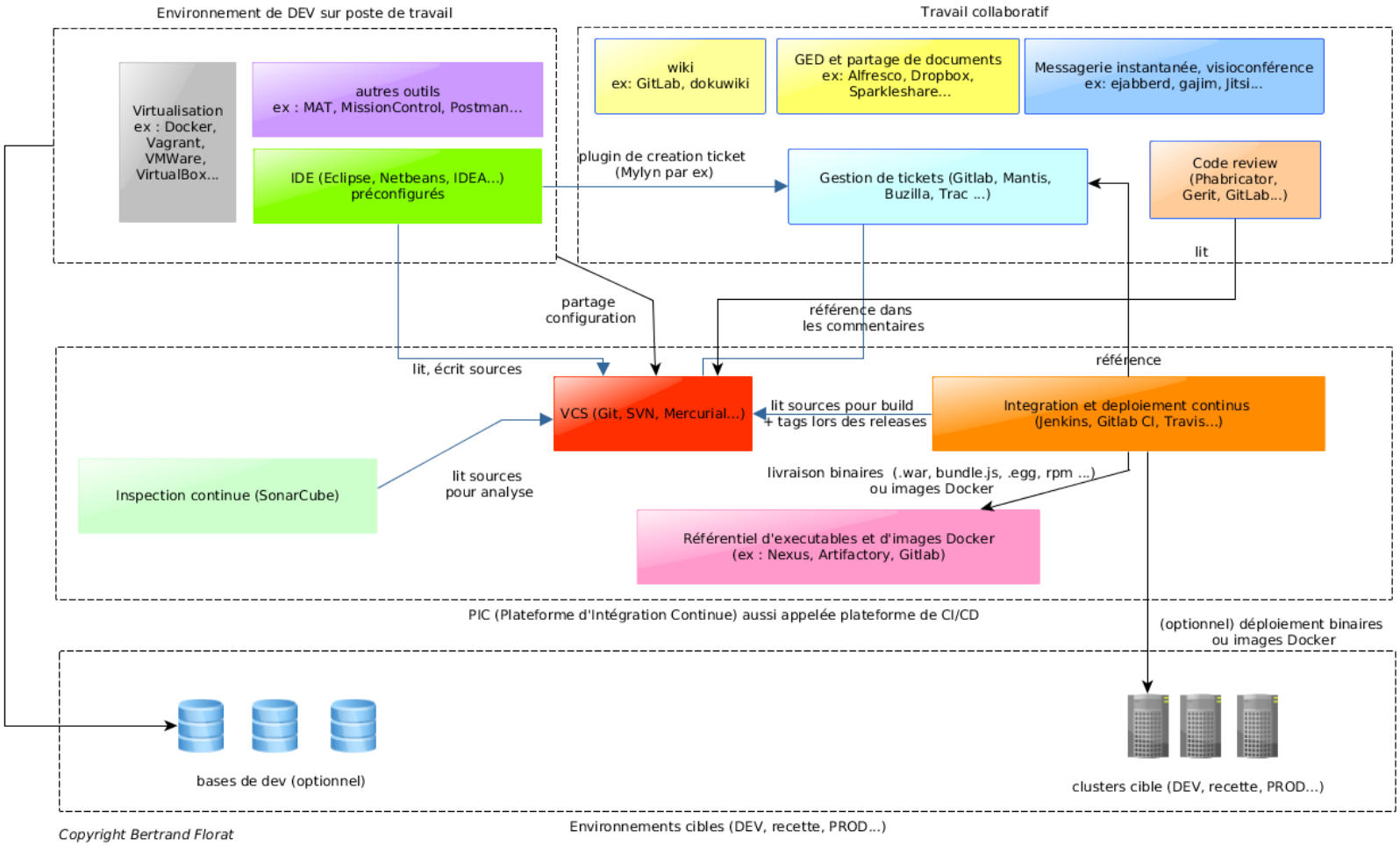
Anti-patterns

- Nombreuses interventions humaines, worflows humains
- Builds lancés manuellement
- Tests (trop) longs
- Perte confiance dans la CI
- Nombreuses branches
- Des paquets applicatifs différents par environnement



Les usines logicielles

Composants principaux de la plateforme de CI/CD

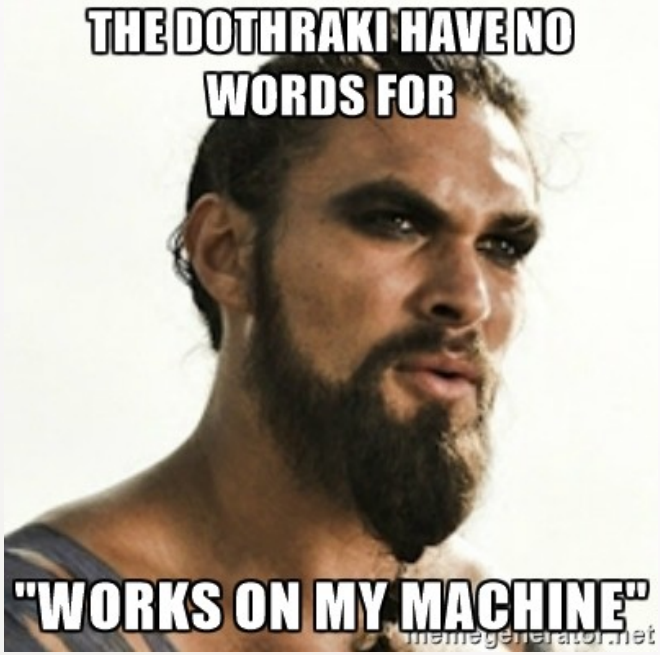


Pourquoi une plateforme de CI-CD ? pour automatiser !

- Intégration continue : pratique **XP** (eXtreme Programming) pour accélérer la boucle de retour
- « Bob the builder » : **collaborateur corvéable à merci (cobotique)**
- Lancement automatique des **builds**
- Lancement automatique des **tests**
- Analyse automatique de la qualité (**Inspection Continue**/qualimétrie)
- **Livraison** continue (Continuous Delivery)
- **Déploiement** continu (en PROD : le Graal)

… et tester en continu

- Évite le syndrome « Works on my machine »
- Évite d’exécuter tous les TU et TI en local



Feedback aussi souvent que possible pour limiter les coûts

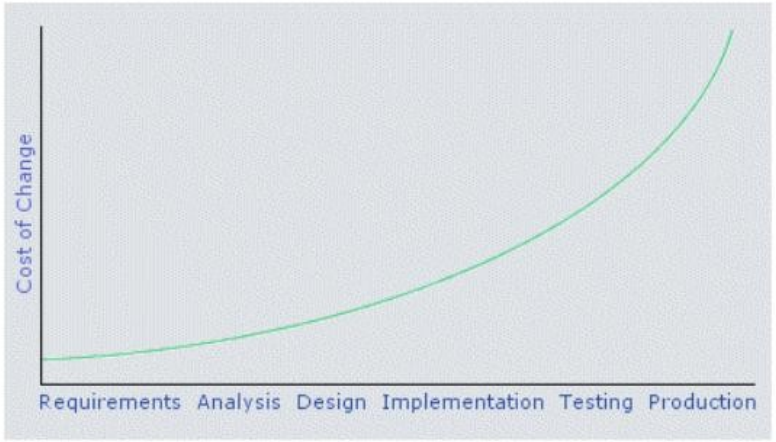
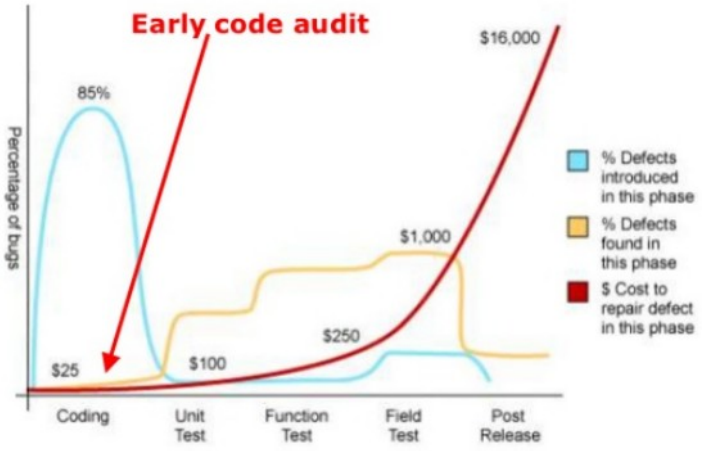
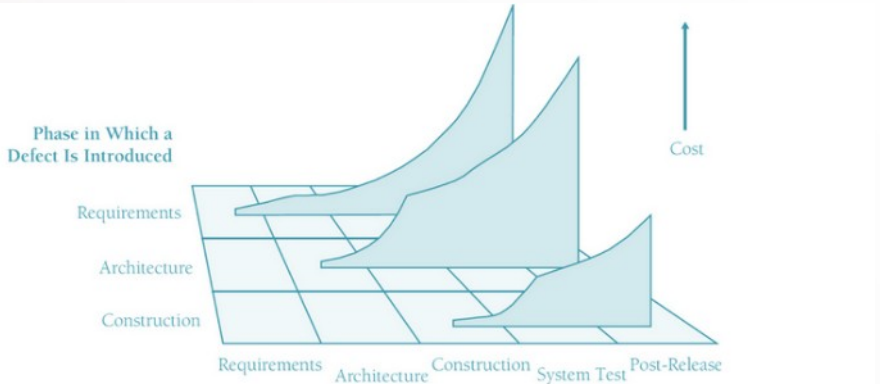


Fig. 4. The cost of change rising exponentially over time (Beck [1999])



Applied Software Measurement, Capers Jones, 1996
Building Security Into The Software Life Cycle, Marco M. Morana, 2006

Règle des 10 mins max : boucle de rétroaction maximale pour un pipeline



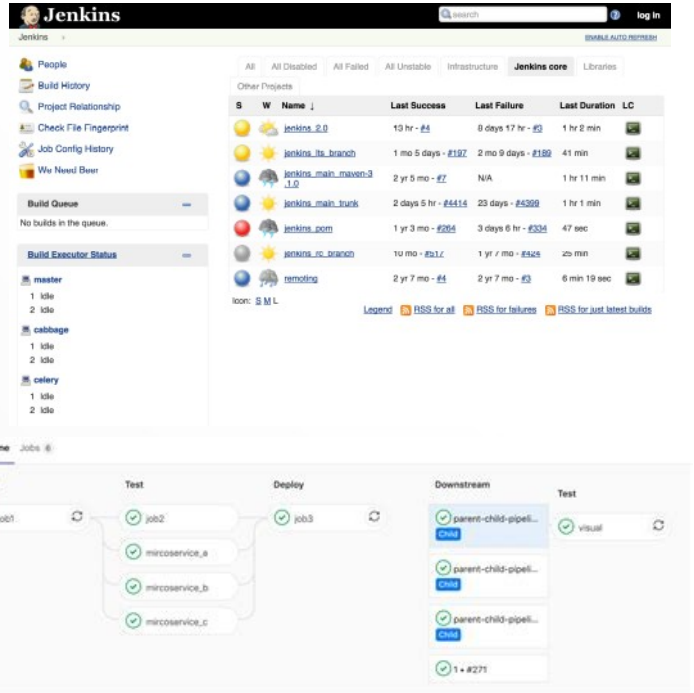
Source : Code complete, 2nd edition, by Steve McConnell [2]

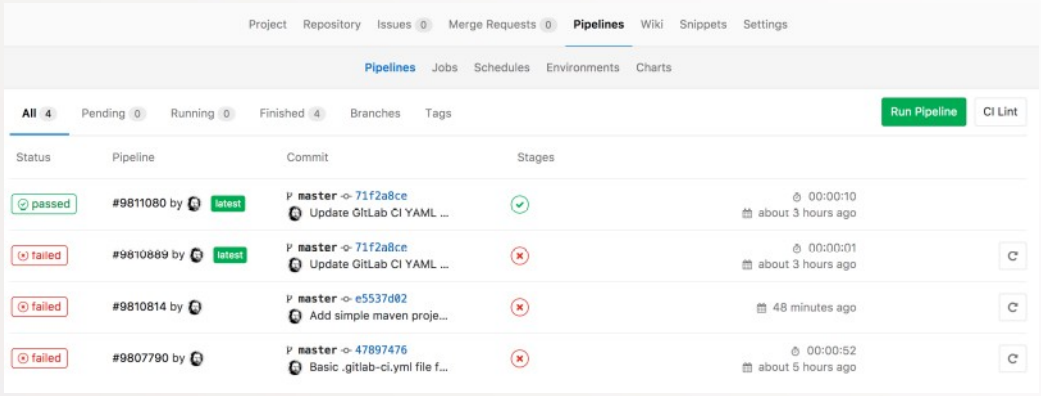
Le SCM : le point d'entrée

- **Git** : distribué, workflows complexes/MR, équipes distribuées, startups, université, Open Source
- **Build au commit**: L'outil de CI récupère le code et lance le build
- Possibilité de coder des **hooks** (commandes exécutées sur événements) pour ajouter des contrôles par exemple.
- En savoir plus : https://cours-git.florat.net/

L'outil de CI/CD : le cœur du réacteur
Mise en place de pipelines

https://landscape.cncf.io/





SonarQube pour analyser la qualité

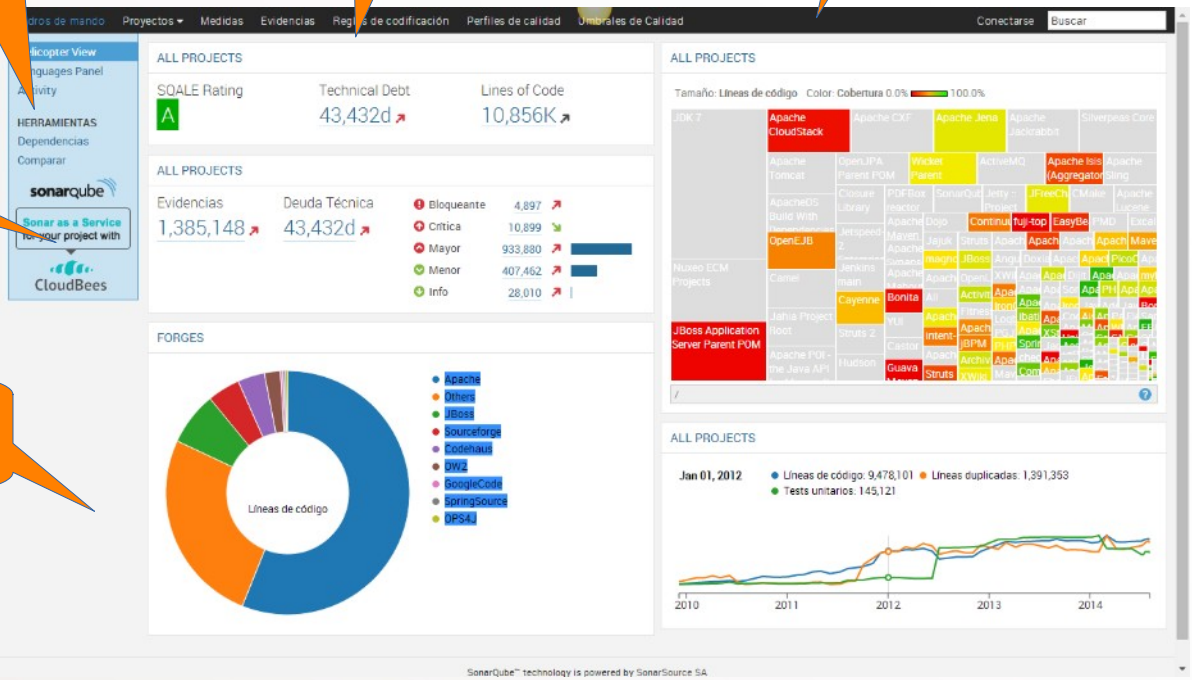
évolution dans le temps

calcul de la dette technique

auto-formation

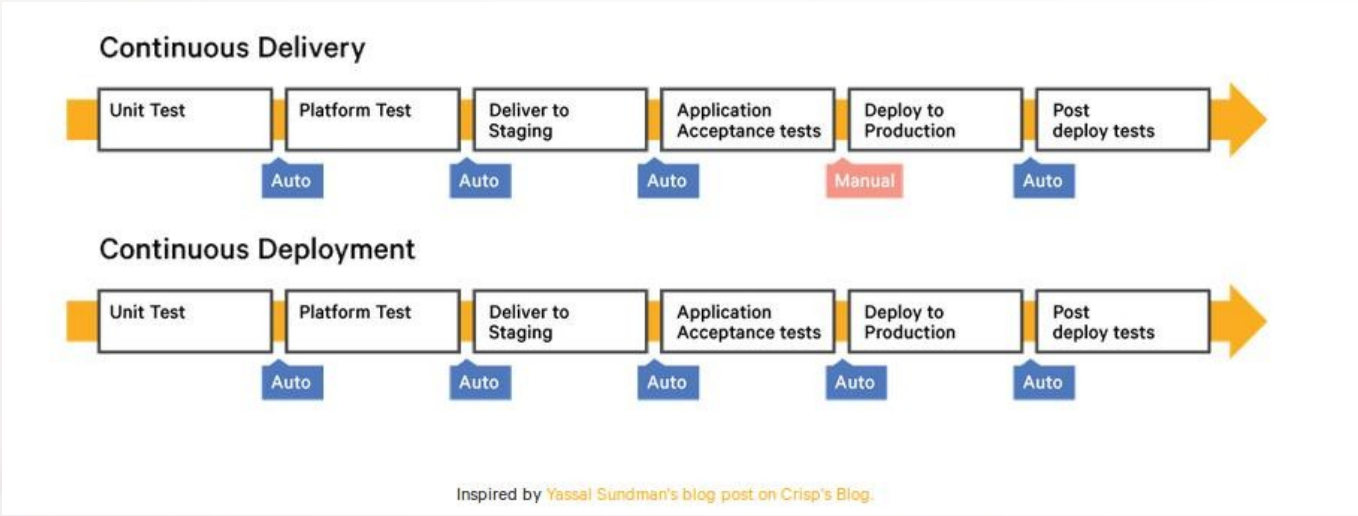
indicateurs haut niveau

profils d'erreurs et de seuils



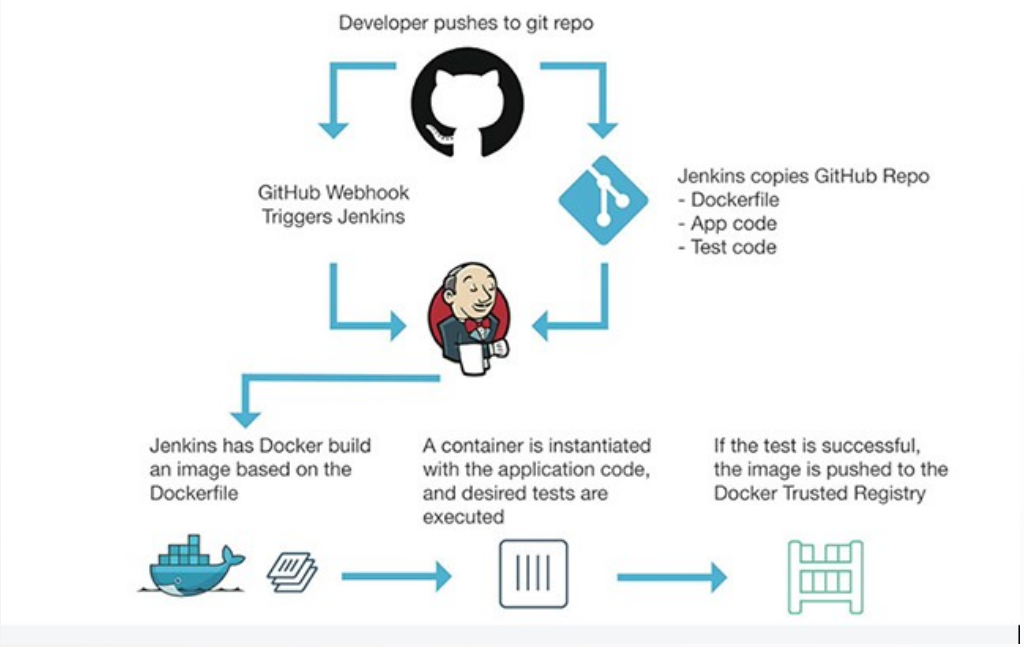
La livraison et le déploiement continu

- **Livraison continue** : prêt à livrer mais on déploie pas
- **Déploiement Continu** : mise à disposition aux utilisateurs sans intervention humaine
- Dans « CI-CD », le 'D' (**Delivery** ou **Deployement**) dépend du niveau de maturité de l'organisation



Déploiement

- **Génération de conteneurs** lors du build
- Déploiement vers cluster ou CaaS via un pipeline
- Hors containeurs, différentes architecture de déploiement :
  - Jboss CLI, SSH, Maven Wagon, Ansible, SaltStack ...

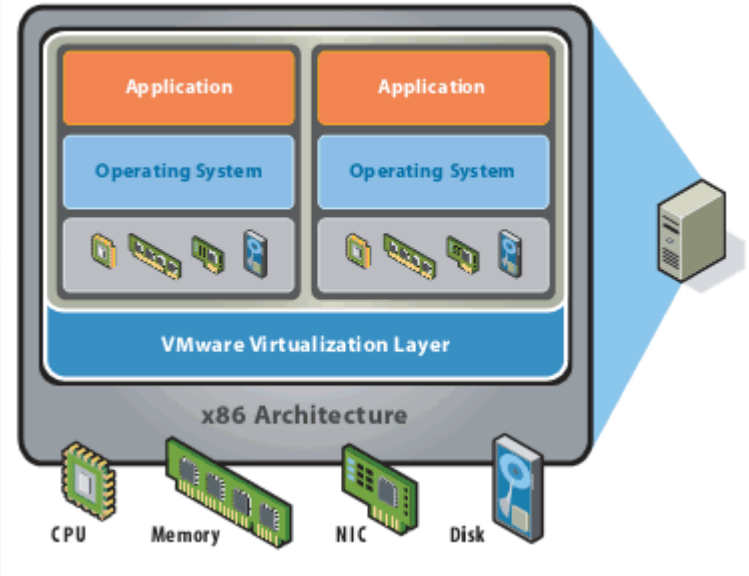


Source : docker.com

Les conteneurs

Aujourd'hui, la plupart des serveurs sont virtuels

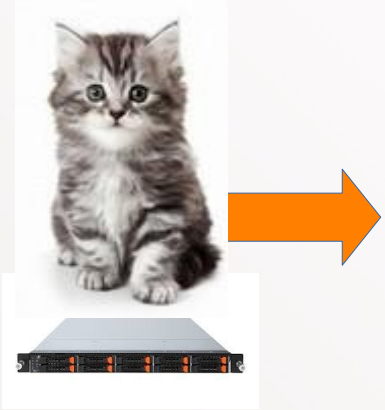
- **Partage mémoire, CPU et disque entre OS invités**
- Technologies : VMWare, KVM, VirtualBox, MS Virtual Server
- Tous environnements (du DEV à la PROD)
- Permet de densifier les datacenters : **on bourre !**
- Grande flexibilité

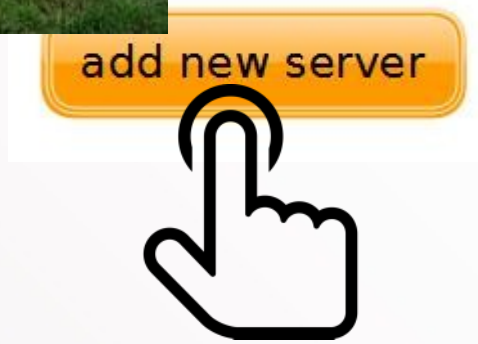


Source : VMWare

Approche « pet to cattle »

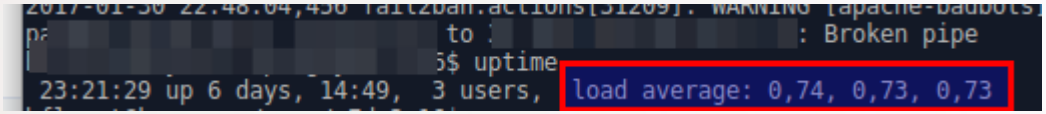
- Les serveurs deviennent une **commodité**
- Serveurs créés/modifiés/supprimés en un clic
- Un problème ? **on remplace**.





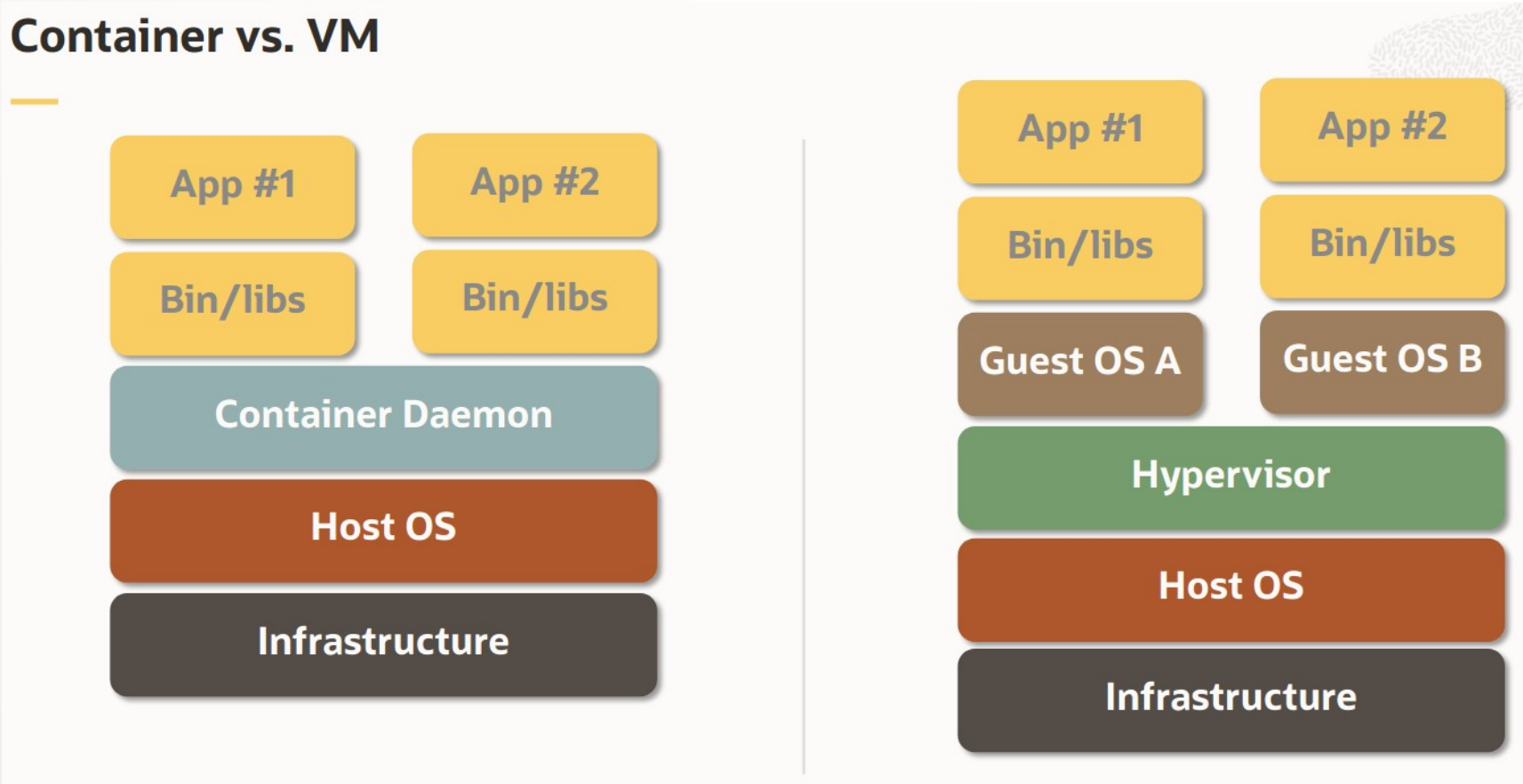
mais ne suffit plus

- Il faut toujours installer les applications :
  - installer les packages (RPM, .deb...)
  - dérouler le guide d'installation
  - … pour chaque déploiement
- Tuning composants complexe
- Dll Hell et problèmes de comptabilité
- **Gaspille** toujours mémoire et disque



→ **il faudrait un grain plus fin**

les conteneurs...



source : ZDNet

Approches complémentaires ?

- Ne pas opposer VM et conteneurs
- VM : premier découpage pour grosses machines
- Les **conteneurs** peuvent être **dans des VM**...
  - Mais pas toujours (**bare metal**)

mais c’est quoi un conteneur ?

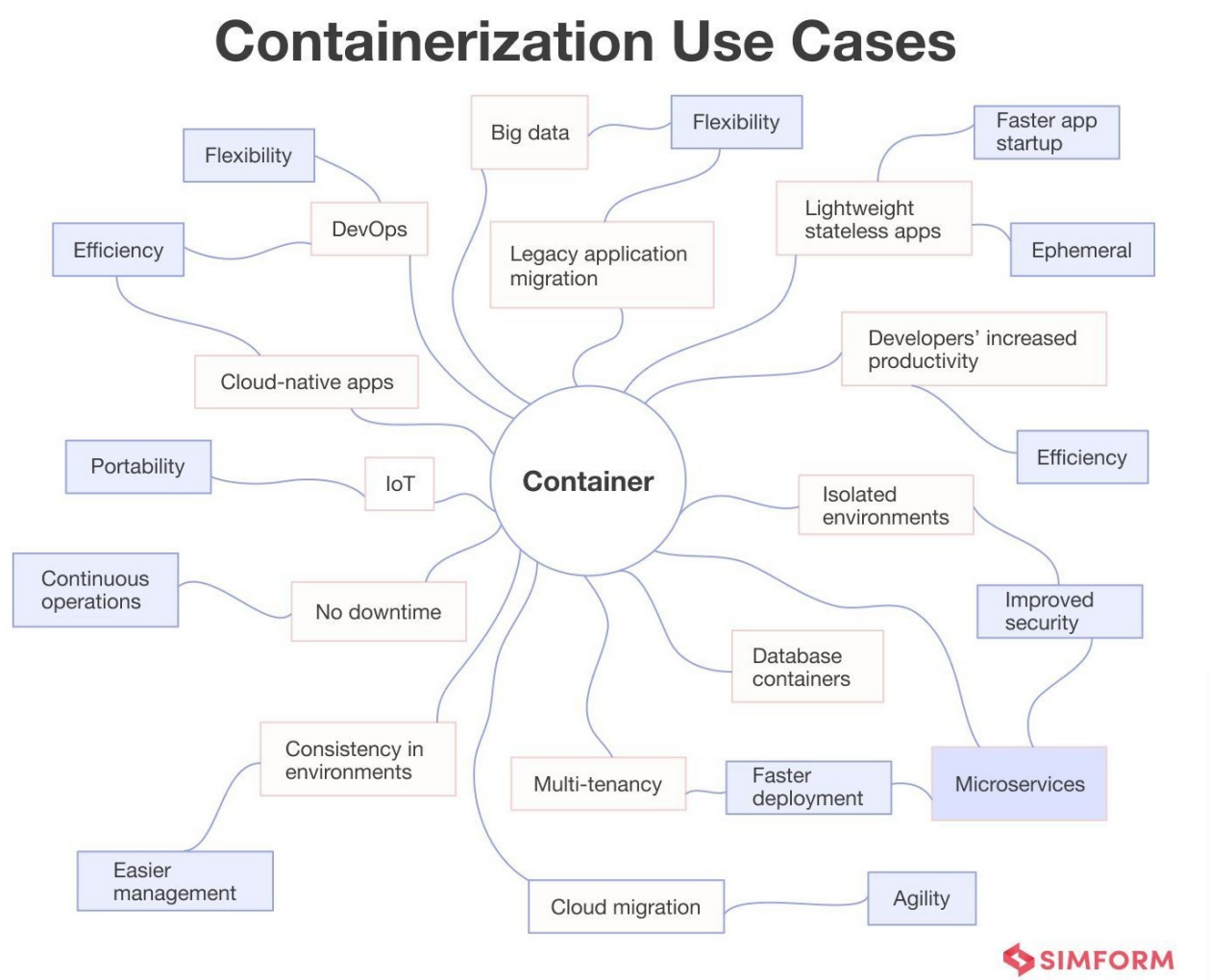
- **Normalisé**
- **Étanche** (processus isolés)
- **Auto-porteur**
- Une norme : Open Container Initiative (OCI)
- S’exécute via un **runtime de containeur** (containerd, CRI-O, rkt, Docker...)

Docker / containerd

- Basé sur des fonctionnalités du **kernel Linux** (cgroups, namespaces, union FS...)
- Exécution sur un même Linux mais en isolation complète
- Repository, gestion de versions
- Taille disque de **quelques Mo** (contre 1 Go pour une VM)
- Consomme **4 à 10 fois moins de RAM** qu'une VM
- Démarre en **quelques secondes**
- **Mono-application** (ex : une image docker mongodb)
- Ne stocke **pas de données** (volumes sur l'hôte)

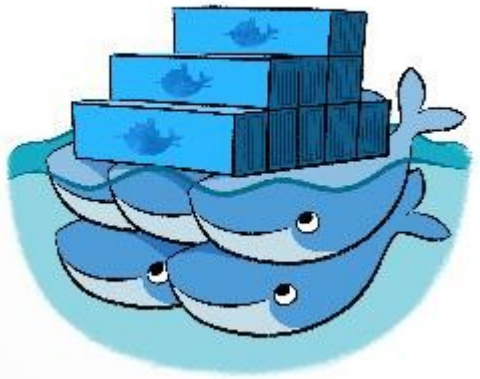
Solomon Hykes

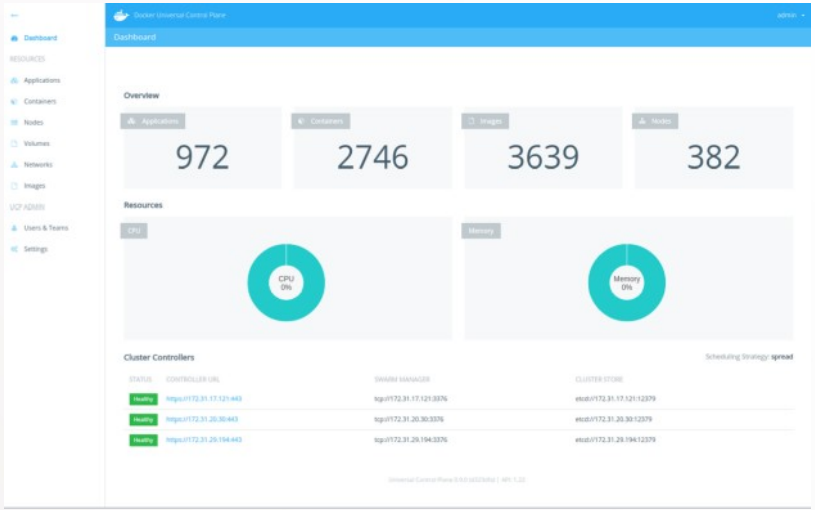
Cas d'usages


Containerization Use Cases
Container
Big data
Flexibility
Flexibility
DevOps
Efficiency
Legacy application migration
Lightweight stateless apps
Faster app startup
Ephemeral
Developers' increased productivity
Efficiency
Cloud-native apps
Portability
IoT
Isolated environments
Continuous operations
No downtime
Improved security
Database containers
Consistency in environments
Multi-tenancy
Faster deployment
Microservices
Easier management
Cloud migration
Agility
SIMFORM

Conteneurs et clouds

- Nouveau type de cloud :
  CaaS (Container As A Service) :
  OVH PublicCloud, GKE, EKS, ACS ...
- **Orchestrateurs** : Docker Swarm,
  Google **Kubernetes**, Rancher…
- Clouds publics ou privés
- **Infrastructure élastique**
- Outils d’exploitation et
  de supervision globaux





Kubernetes

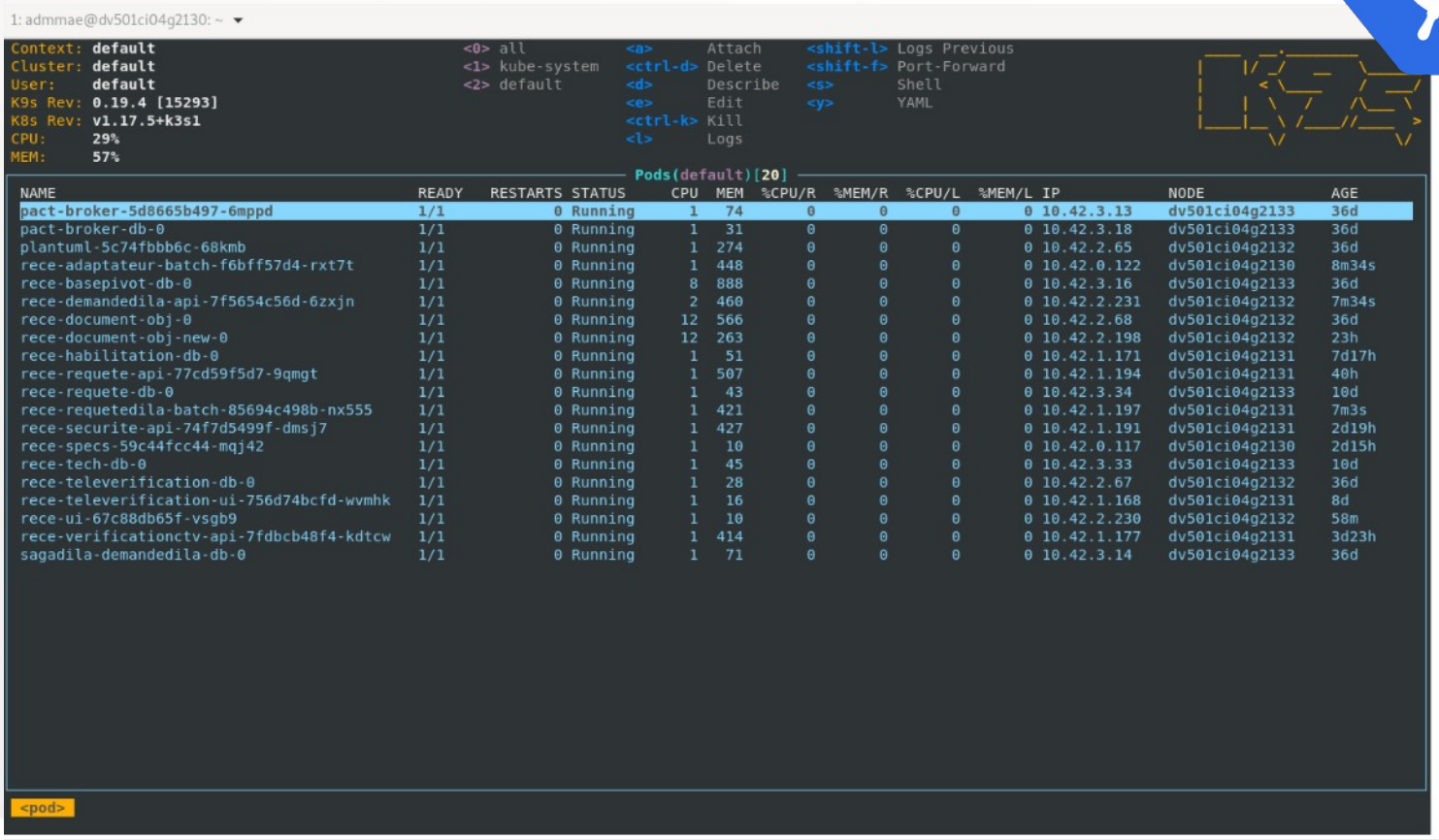
- kubernetes.io : « Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. »

Bref, c'est un orchestrateur de conteneurs

C'est une toute une infrastructure managée au dessus de l'infrastructure existante. conteneurs (compute) + stockage + réseau

Permet effectivement le développement et l'exploitation **d'applications** vraiment **cloud-native**

Kubernetes


1: admmae@dv501ci04g2130: ~
Context: default
Cluster: default
User: default
K9s Rev: 0.19.4 [15293]
K8s Rev: v1.17.5+k3s1
CPU: 29%
MEM: 57%
<0> all
<1> kube-system
<2> default
<a> Attach
<ctrl-d> Delete
<d> Describe
<e> Edit
<ctrl-k> Kill
<l> Logs
<shift-l> Logs Previous
<shift-f> Port-Forward
<s> Shell
<y> YAML
Pods(default)[20]
NAME READY RESTARTS STATUS CPU MEM %CPU/R %MEM/R %CPU/L %MEM/L IP NODE AGE
pact-broker-5d8665b497-6mppd 1/1 0 Running 1 74 0 0 0 0 10.42.3.13 dv501ci04g2133 36d
pact-broker-db-0 1/1 0 Running 1 31 0 0 0 0 10.42.3.18 dv501ci04g2133 36d
plantuml-5c74fbbb6c-68kmb 1/1 0 Running 1 274 0 0 0 0 10.42.2.65 dv501ci04g2132 36d
rece-adaptateur-batch-f6bff57d4-rxt7t 1/1 0 Running 1 448 0 0 0 0 10.42.0.122 dv501ci04g2130 8m34s
rece-basepivot-db-0 1/1 0 Running 8 888 0 0 0 0 10.42.3.16 dv501ci04g2133 36d
rece-demandedila-api-7f5654c56d-6zxjn 1/1 0 Running 2 460 0 0 0 0 10.42.2.231 dv501ci04g2132 7m34s
rece-document-obj-0 1/1 0 Running 12 566 0 0 0 0 10.42.2.68 dv501ci04g2132 36d
rece-document-obj-new-0 1/1 0 Running 12 263 0 0 0 0 10.42.2.198 dv501ci04g2132 23h
rece-habilitation-db-0 1/1 0 Running 1 51 0 0 0 0 10.42.1.171 dv501ci04g2131 7d17h
rece-requete-api-77cd59f5d7-9qmgt 1/1 0 Running 1 507 0 0 0 0 10.42.1.194 dv501ci04g2131 40h
rece-requete-db-0 1/1 0 Running 1 43 0 0 0 0 10.42.3.34 dv501ci04g2133 10d
rece-requetedila-batch-85694c498b-nx555 1/1 0 Running 1 421 0 0 0 0 10.42.1.197 dv501ci04g2131 7m3s
rece-securite-api-74f7d5499f-dmsj7 1/1 0 Running 1 427 0 0 0 0 10.42.1.191 dv501ci04g2131 2d19h
rece-specs-59c44fcc44-mqj42 1/1 0 Running 1 10 0 0 0 0 10.42.0.117 dv501ci04g2130 2d15h
rece-tech-db-0 1/1 0 Running 1 45 0 0 0 0 10.42.3.33 dv501ci04g2133 10d
rece-televerification-db-0 1/1 0 Running 1 28 0 0 0 0 10.42.2.67 dv501ci04g2132 36d
rece-televerification-ui-756d74bcfd-wvmhk 1/1 0 Running 1 16 0 0 0 0 10.42.1.168 dv501ci04g2131 8d
rece-ui-67c88db65f-vsgb9 1/1 0 Running 1 10 0 0 0 0 10.42.2.230 dv501ci04g2132 58m
rece-verificationctv-api-7fdbcb48f4-kdtcw 1/1 0 Running 1 414 0 0 0 0 10.42.1.177 dv501ci04g2131 3d23h
sagadila-demandedila-db-0 1/1 0 Running 1 71 0 0 0 0 10.42.3.14 dv501ci04g2133 36d
<pod>

The Big Picture (décomposition standard avec ordres de grandeur moyens)

1 planète
1..100 datacenters par organisation

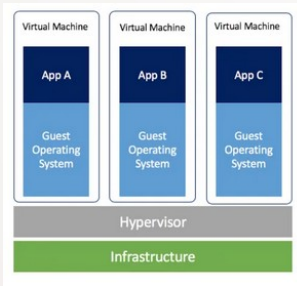
500 m² à 2 000 m²
~200 **armoires**

42U, *7* fonds de panier

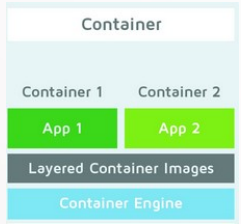
Hardware

2 sockets, 32 cœur, 250 Gio RAM

8U, 16 lames, ~512 ***cœurs***,
4 To RAM



128 VMs



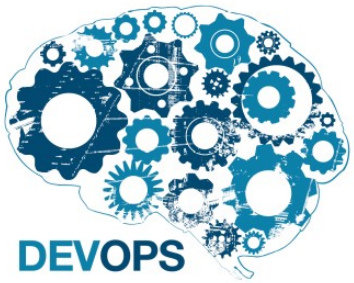
~10 à 100 conteneurs

Software

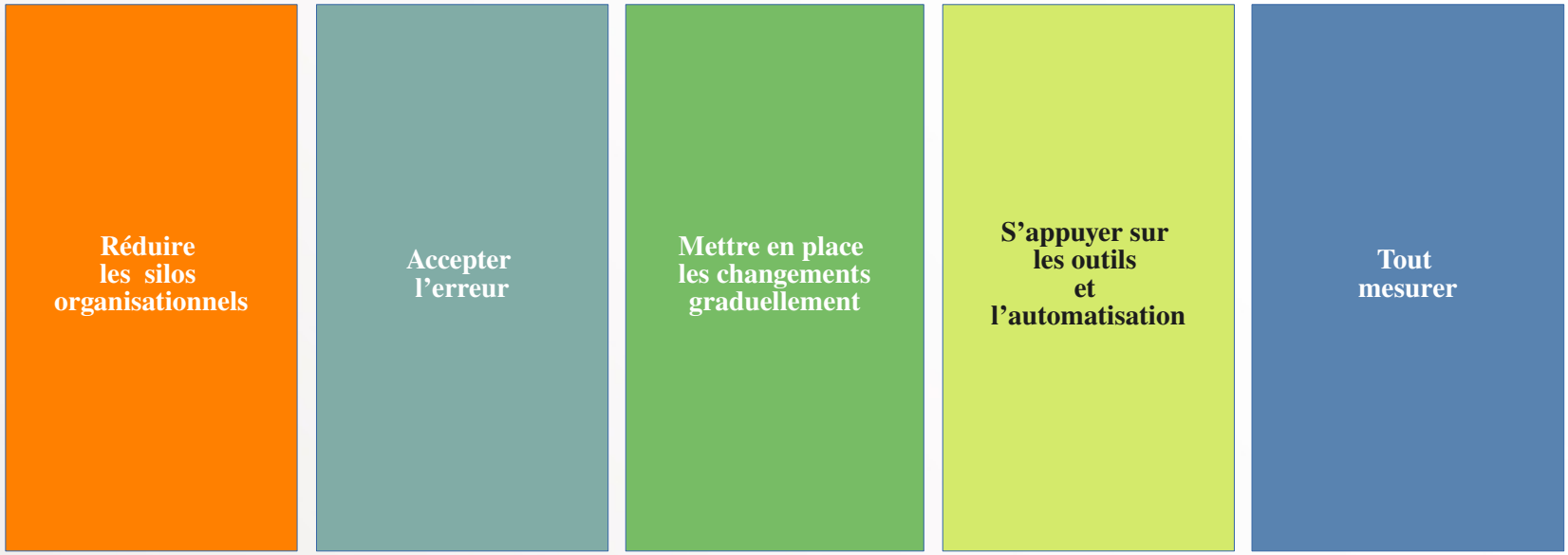
La démarche DevOps

Le DevOps...

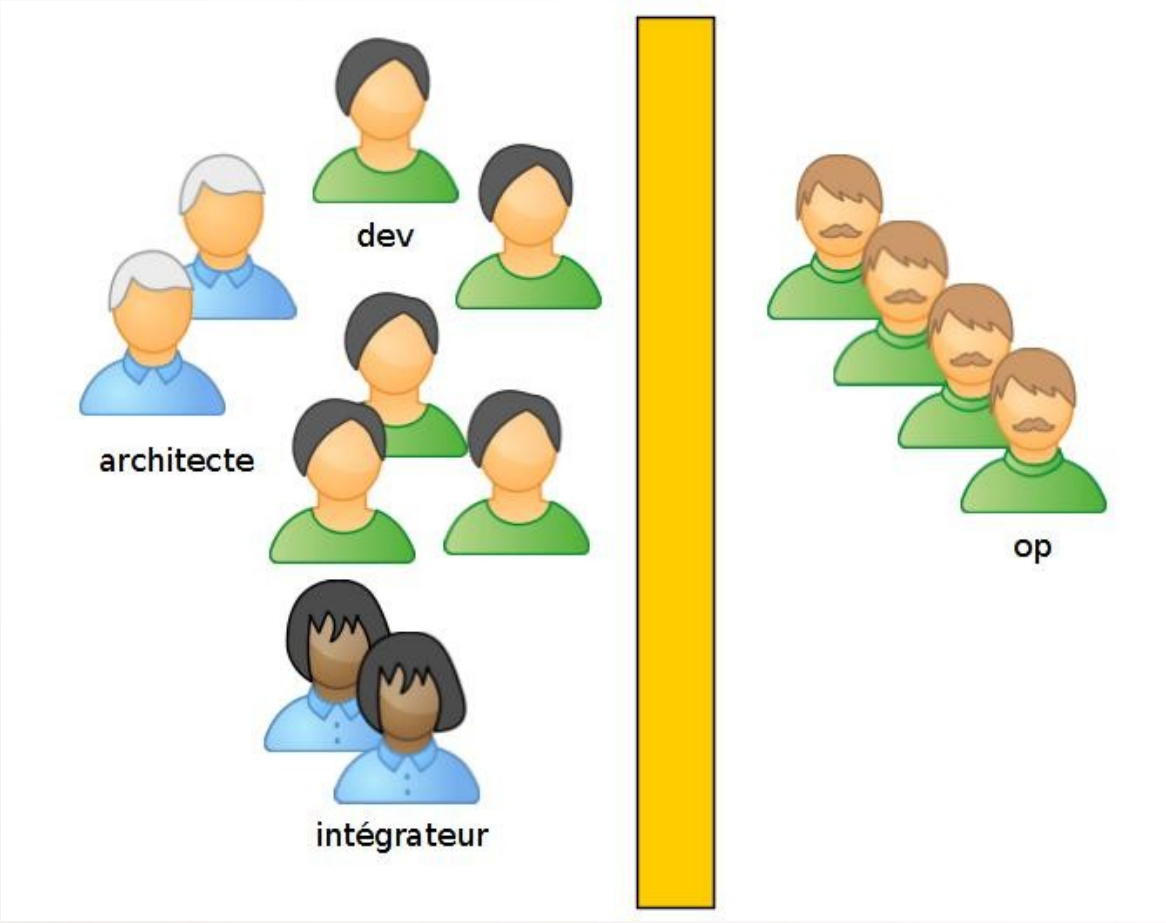
- Ensemble de principes et pratiques issus du **courant agiliste** (2008) : l'agile des ops
  - encore **embryonnaire dans les faits**
- Équipes **multi-disciplinaires** centrées sur l'apport métier : **le produit avant le process**
- Vise l'amélioration du **TTM** (Time To Market)
- Deux piliers :
  - La **collaboration**
  - **L'automatisation**



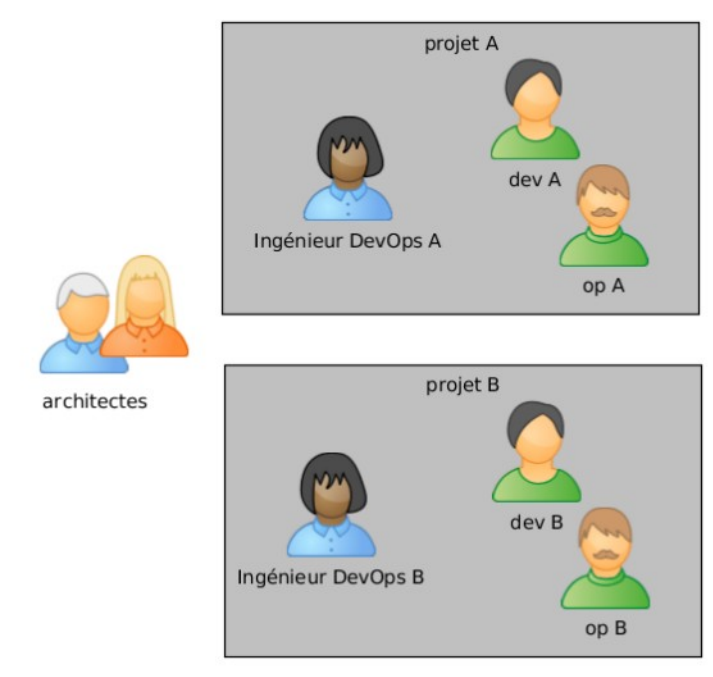
Les piliers du succès



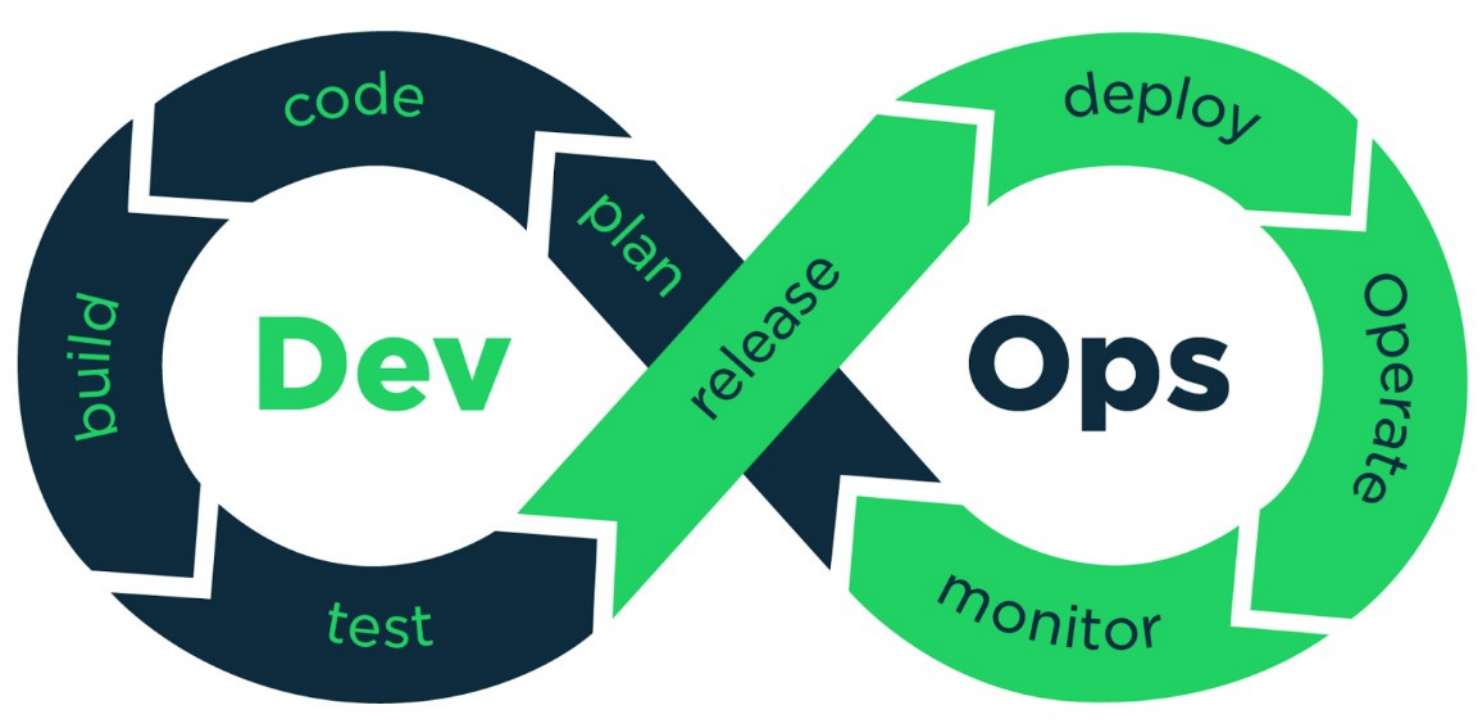
De ça :



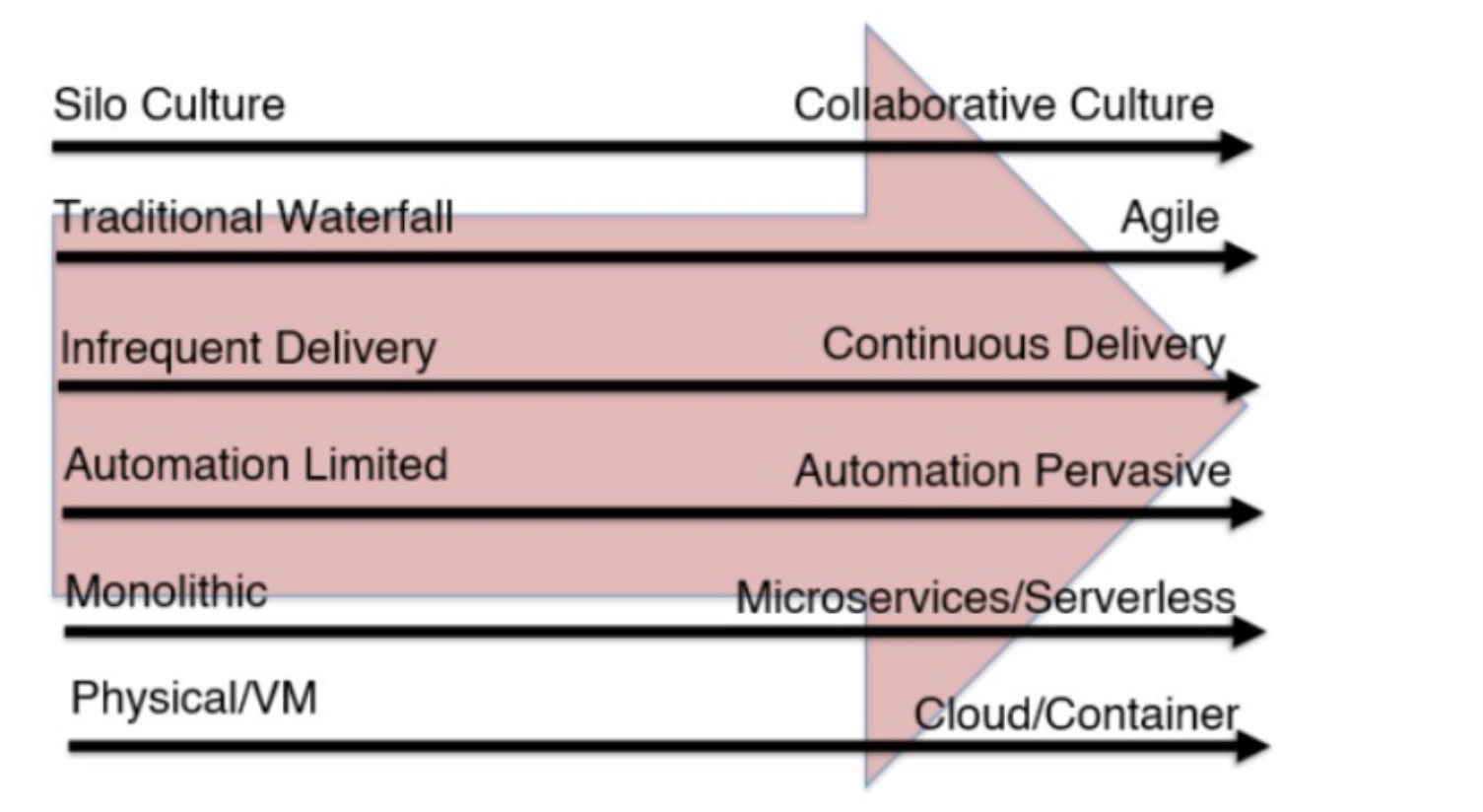
à ça (la two-pizzas team) :



Un cycle d'amélioration continue



Mix de changements architecturaux / process et humains



Source: DZone

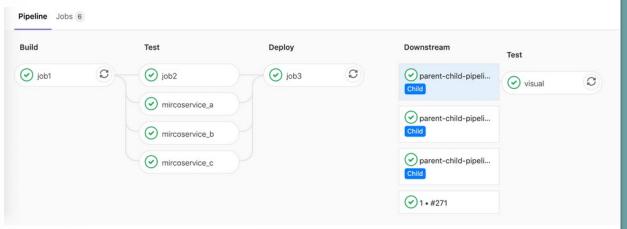
Collaborer, mais comment ?

- **Penser système**, pas équipe isolée
  - Communication globale
- **Feedback**
  - Source d'information commune (ex: daily)
- Innovation et **amélioration continue**
  - Ateliers
- **Partager les responsabilités**
- Casser les carcans
  - les **devs exploitent** (cloud)
  - les **ops codent** (IaC : Infrastructure as Code)

« You build it, you run it »
(Amazon)

Qu’est ce qu’on entend par automatisation ?

- **CI** : Intégration Continue
- **Tests automatisés** (TU,
  TI, E2E, spécifications exécutables...)
- Livraison continue
- Infrastructure as Code (**IaC**)
  - **Provisioning** des serveurs et composants
  - Minimiser le **MTTR** (temps de réparation) et le **MTTD** (temps de détection) plutôt que le MTBF
    (temps entre deux pannes)
  - **GitOps** : Si ce n’est pas dans Git, ça n’existe pas
- **CD** (Livraison Continue voire Déploiement Continu)



Pipeline Gitlab

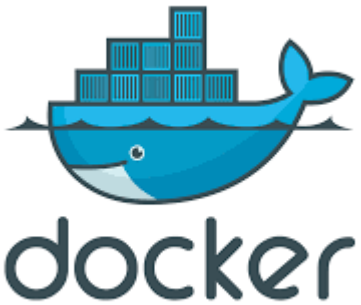
Les outils d'Infrastructure As Code

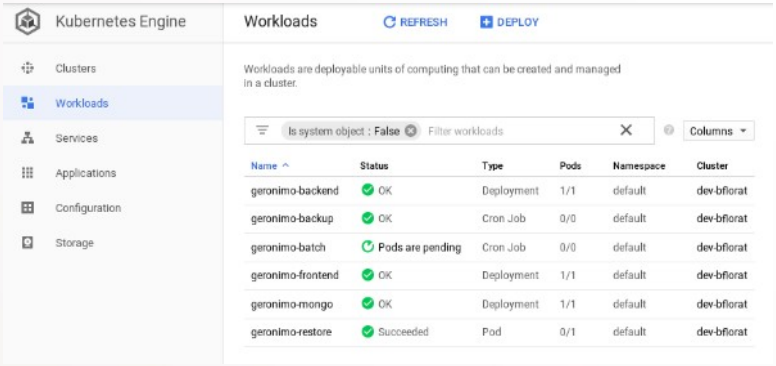
- Libère les Ops pour des **tâches à plus grande valeur ajoutée** (performance, sécurité..)

```
---
- hosts: webservers
  vars:
      http_port: 80
      max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
      yum:
        name: httpd
        state: latest
  - name: write the apache config file
      template:
        src: /srv/httpd.j2
        dest: /etc/httpd.conf
      notify:
      - restart apache
  - name: ensure apache is running
      service:
        name: httpd
        state: started
  handlers:
      - name: restart apache
        service:
            name: httpd
            state: restarted
```

Qu’apportent les conteneurs ?

- **Facilite grandement le déploiement** de systèmes complexes
- Facilite les **architectures micro-services**
- Remonte le **tuning** vers les équipes de **DEV**
- Référentiel de composants d’infrastructure sur étagère
- **Économies** de mémoire
- Plus besoin de guides d'installation
- **Exploitation unifiée**





DevOps : experts en automatisation

Compétences attendues :

- **Sysadmin** Linux (++)
- **Virtualisation** / **containeurs** / outils **d'IaC**
- Connaissances en **réseau, stockage** et solutions **Cloud**
- Maîtrise de quelques **langages** orientés ops (Bash, Python, GO...)
- **Soft skills** ++ (et bonne gestion du stress)
- Très bonne maîtrise des outil de **CI/CD**
- Bonne maîtrise des principes de **tests**
- Bonnes connaissances en **sécurité**
- Maîtrise des **process**

Quiz ! Niveau DevOps de la pratique (de 0 à 5) ?

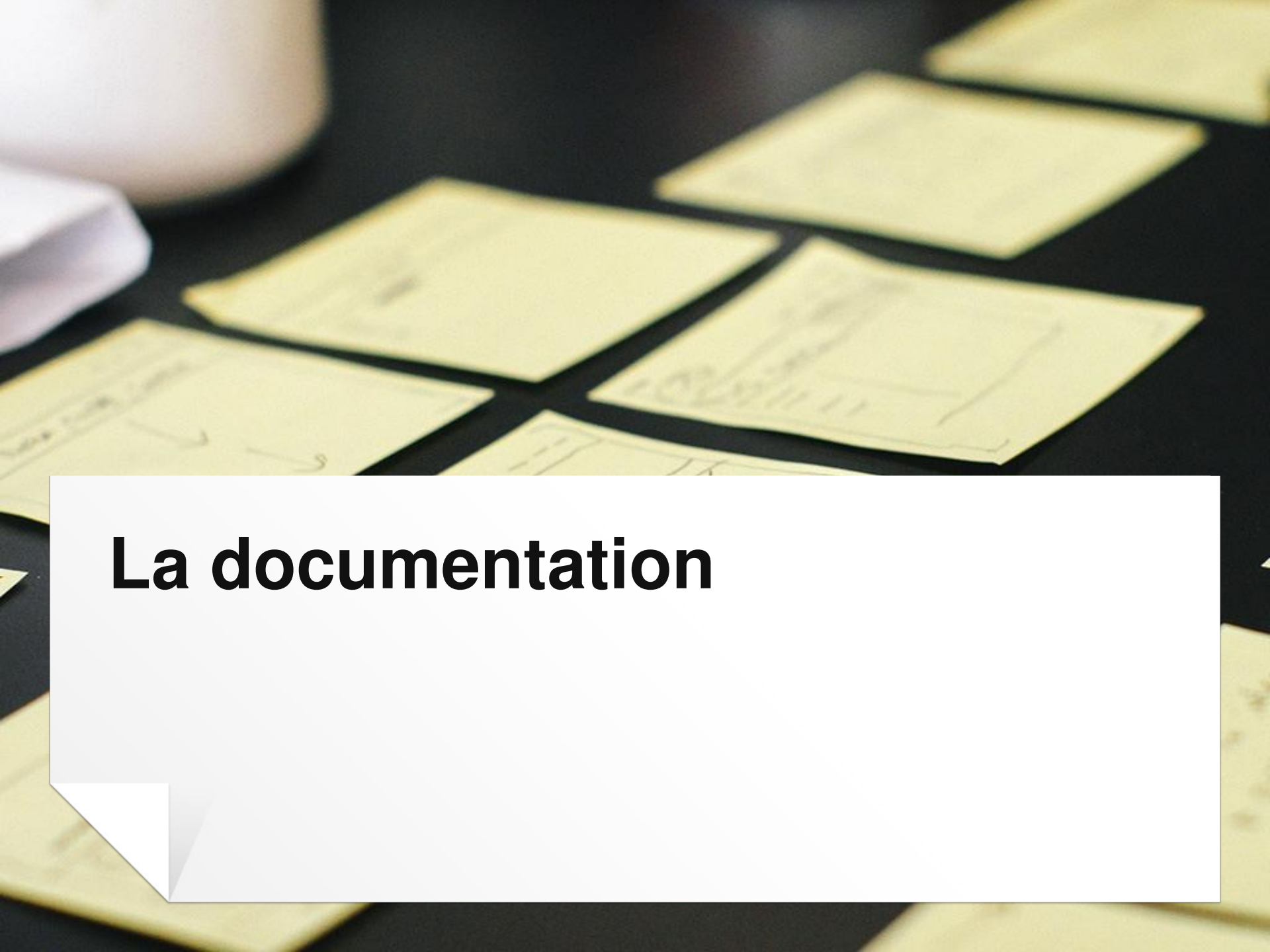
Les Ops fournissent aux Devs un dashboard Zabbix mais seuls les Ops reçoivent les alertes par mail.

Un canal de messagerie instantanée a été mis en place pour remonter les problèmes de PROD mais la plupart du temps, les Ops ne signalent que les indisponibilités programmées.

Chaque équipe produit dispose d'un DevOps dédié qui automatise tous les aspects du cycle de vie (build, déploiement, alerting…) et dispose des droit de PROD sur le provider Cloud.

Les modules sont conteneurisés en Docker mais lancé manuellement (pas d'orchestrateurs)

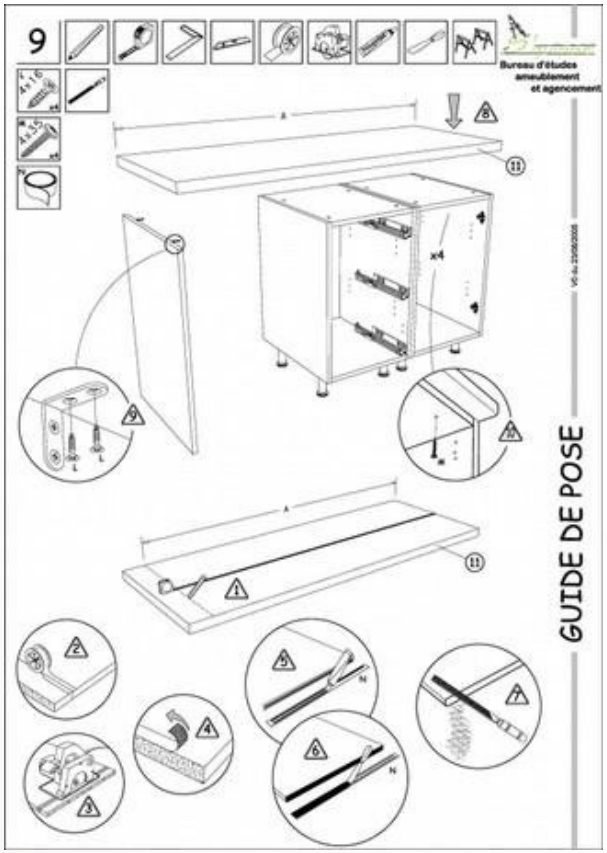
L'équipe de Dev rédige un document d'installation manuelle et l'envoie aux Ops par mail lorsque la version du produit est publiée.



La documentation

Le guide d’installation

- Doit permettre à un exploitant d’installer l’application
- De vérifier que l’application fonctionne
- De mettre l’application à jour
- De supprimer l’application

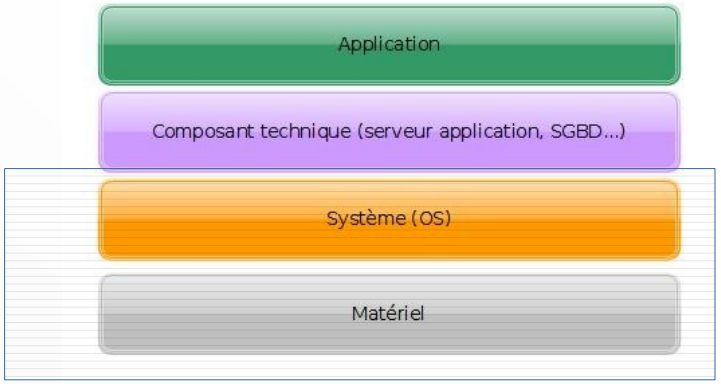


Le guide d’installation

avec les conteneurs,
ce guide est remplacé
par le déploiement
continu et des
tests automatisés

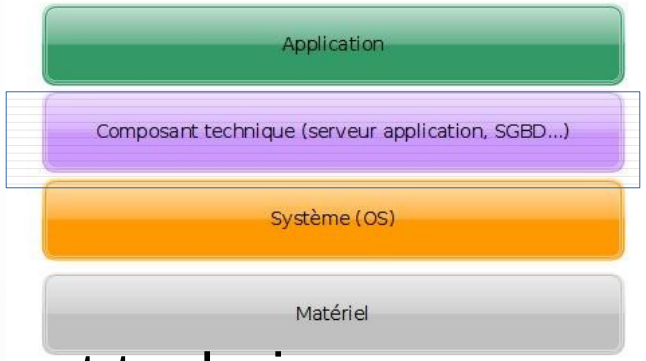
Le guide d'installation
Préparation du système

- Pré-requis :
  - version de l'OS
  - pré-requis matériels (CPU , mémoire, disque)
- Paquets requis
- Création de partitions ou volumes logiques
- Création des groupes et utilisateurs
- Configuration de la configuration globale du serveur



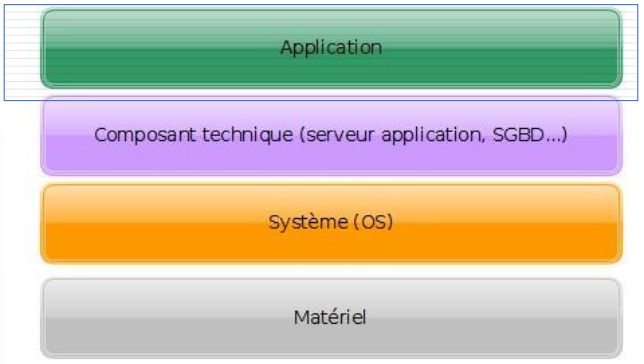
Le guide d'installation
Installation des composants techniques

- Pré-requis (composant et version)
- Installation du composant (SGBD, serveur d'application …)
- Configuration standard du composant technique (charset, sécurité, connectivité...)
- Tuning fin de l'intergiciel (Xmx, taille des caches…)
- Démarrages automatiques



Le guide d'installation
Installation des applications

- Copie / déploiement des binaires
- Configuration applicative
- Pré-chauffage
- Smoke tests bout en bout



Le guide d'installation
Mises à jour

- Depuis la version n-1
- Depuis la version n-x ? si exigé
  - Conserver plusieurs versions pour MAJ incrémentales
- Scripts de MAJ de bases
  - Scripts d'extension (vers état transitoire versions n et n+1)
  - Scripts de contraction (vers état n+1)
  - Scripts de rollback (vers état n)
  - Outils : voir par exemple Liquibase
- Optimisations (calcul stats, mise en cache...)

Migration

- Arrêt de l’ancien système
- Export des données
- Migration des données
- Import de données
- Tests

faire une migration incrémentale quand possible

Le guide d’installation
Désinstallation

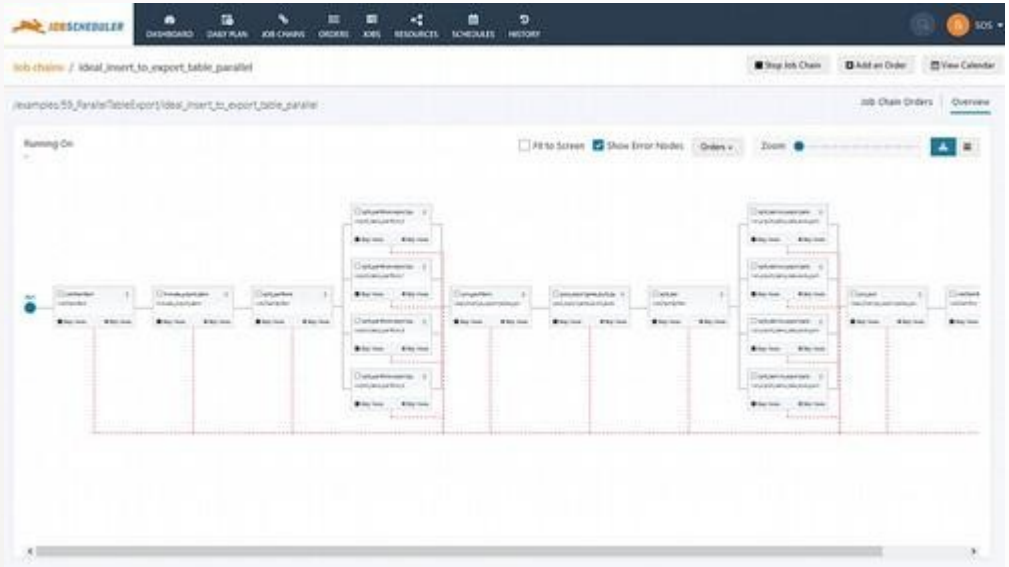
- Arrêt de l’application
- Archivage
- Suppression des données
- Suppression des intergiciels désormais inutiles
- Libration des ressources (disque)
- Suppression démarrages automatiques

Le Dossier d’EXploitation (DEX)

- Permet à un exploitant de faire fonctionner l’application
- Dans le respect des exigences non fonctionnelles

Le DEX
Commandes

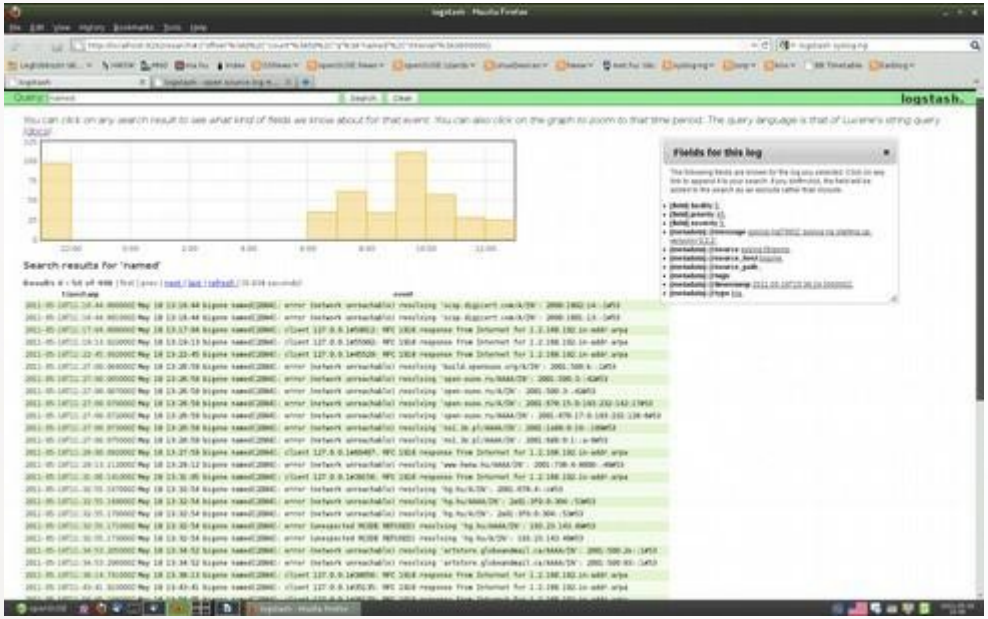
- Procédures d’arrêt / démarrage (dans le bon ordre)
- Liste des opérations programmées
- Affinités / anti-affinités des batchs
- Plan de production
- Procédures purges /archivage



Ordonnancement de batchs dans JobScheduler

Le DEX
Gestion des incidents

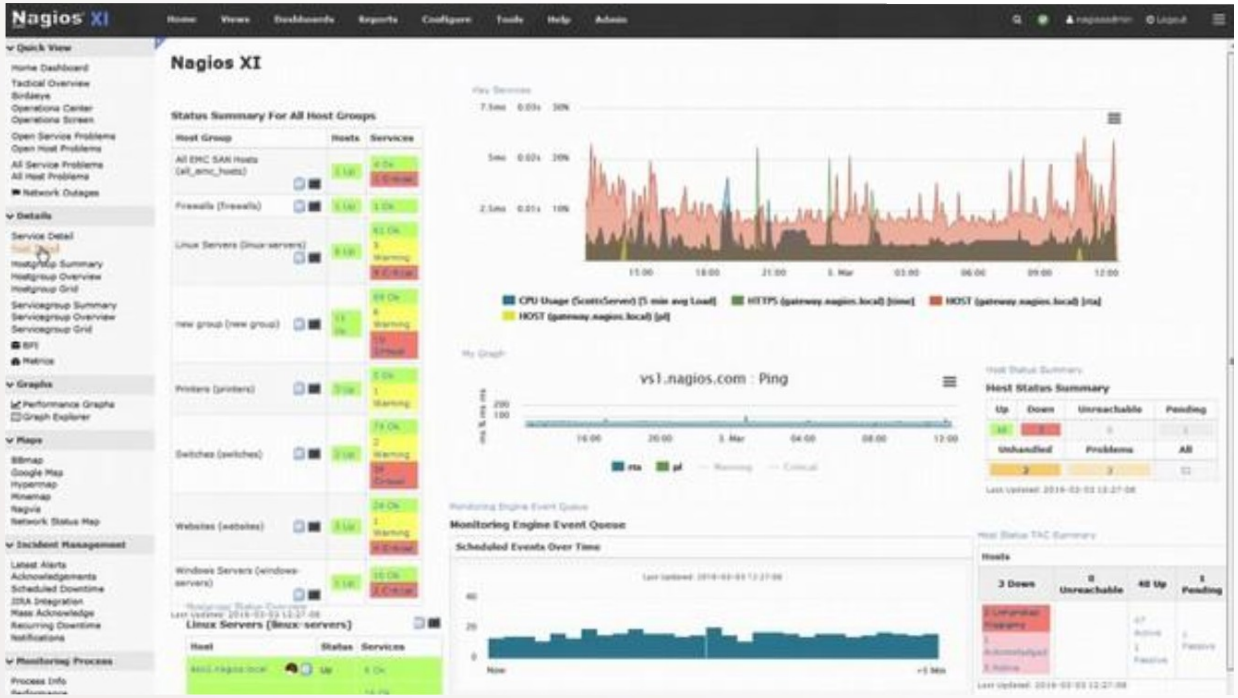
- Localisation des logs et aide l'interprétation
- Aide au diagnostique
- Procédures de sauvegarde et restauration
- Modes dégradés



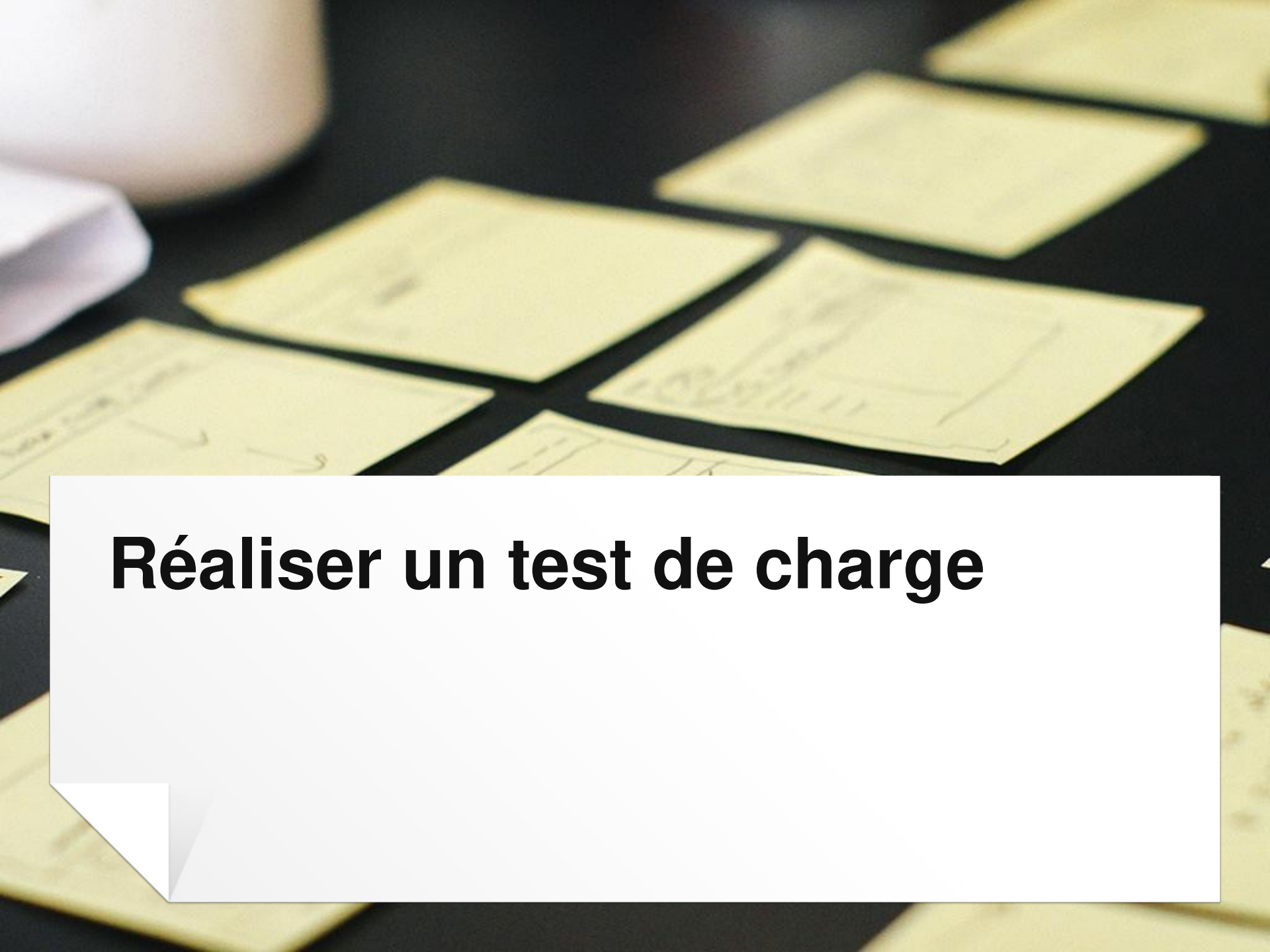
Logstash : consultation de logs centralisés

Le DEX
Supervision

- Métriques + seuils = indicateurs
- Indicateurs système
- Indicateurs applicatifs



Supervision avec Nagios



Réaliser un test de charge

Préparer un test de charge
Les tests d'acceptation / à la cible

- Charge le système comme prévu dans les exigences
  - Objectif : **vérifier que le système tient la charge demandé en pic**

- Pré-requis : recueillir le besoin (SLA)
  - Charge prévisionnelle en TPS
  - Exigences de temps de réponse au 95eme centile
  - Ratio d'erreurs
  - Disponibilité

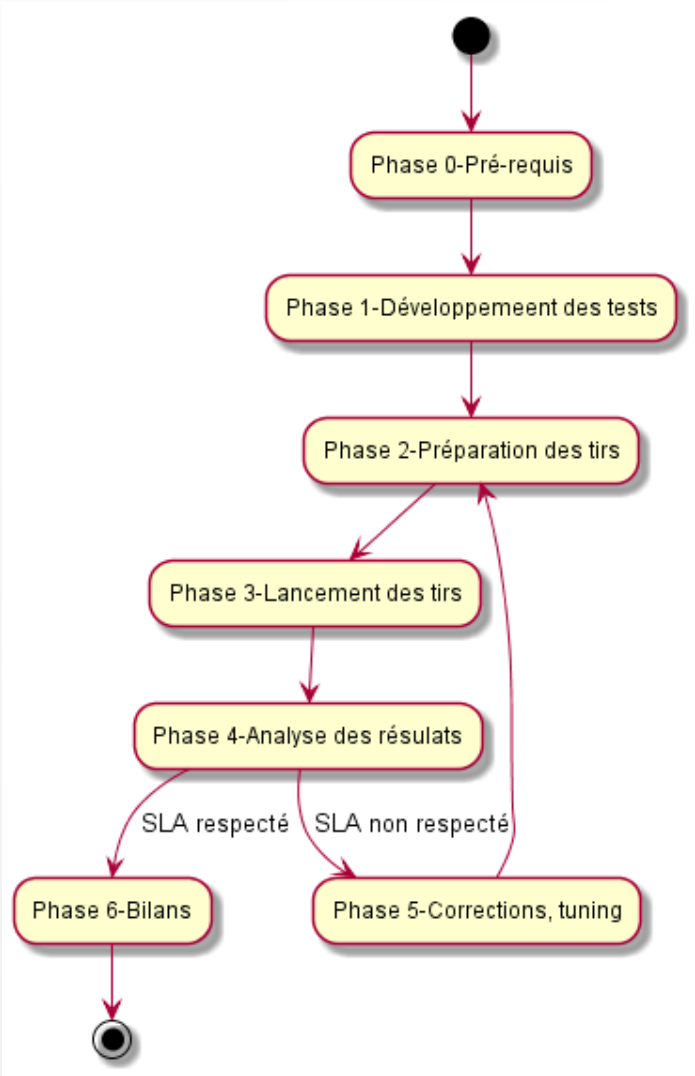
Préparer un test de charge
Les tests aux limites

- Charge le système jusqu'à ses limites
- Quelques limites intéressantes :
  - Décrochage : le ratio TR/nb utilisateurs commence à ne plus être linéaire (devient exponentiel)
  - Rupture (on ne respecte plus le TR ou ratio d'erreurs dépassé)
  - Tombe en panne (plus aucune réponse)
- Objectif : **déterminer la charge que peut supporter mon système**
  - De quelle marge je dispose ?

Préparer un test de charge
Les tests de robustesse

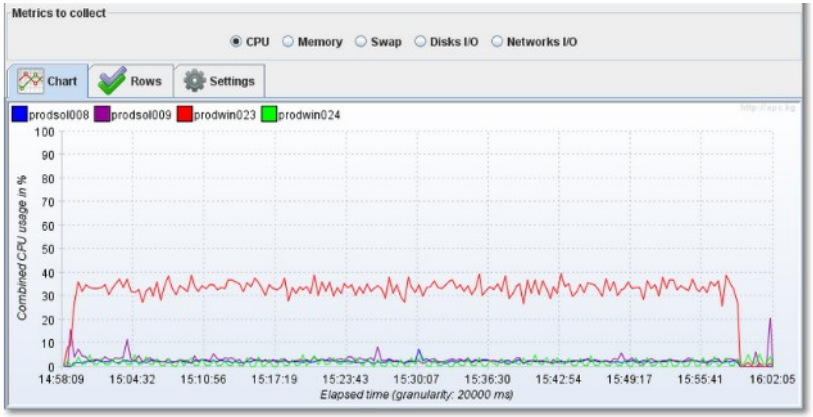
- **Charge** le système sur une **longue période**
  - exemple : une semaine
- Objectif : détecter les **dérives**
  - Fuites mémoire
  - Remplissage de systèmes de fichiers
  - Erreurs intermittentes

Préparer un test de charge
Exemple de workflow



Préparer un test de charge
Les outils : l'outil de supervision

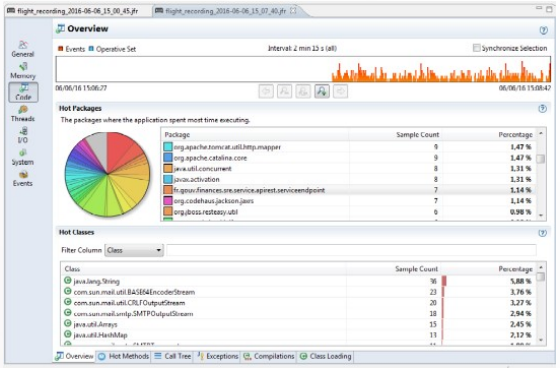
- Suivre/enregistrer
  la charge du système
- Corréler charge du système /charge injectée
- Métriques : CPU, mémoire, réseau, nb threads, ...
- Outils :
  - APM (Application Performance Monitoring)
  - Outils standards (top, vmstat.)
  - plugins de l'injecteur (injecteur HTTP de jmeter)



JMeter Servers Performance Monitoring

Préparer un test de charge
Les outils : le profiling

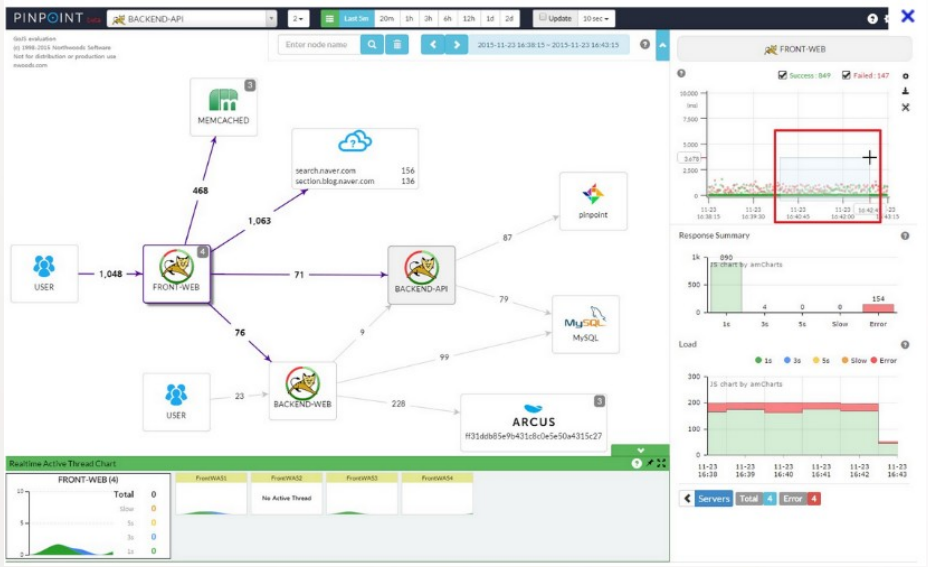
- En DEV ou en PROD selon les outils
- Analyser les problèmes de performance (hot spots)
- Temps de réponse détaillés
- Analyse de la mémoire
- Outils :
  - Orienté développement :
    JVisualVM, Chrome DevTools
    pour les applications web côté client.
  - Orienté production :
    Glowroot, Mission Control,
    PinPoint, nombreux APM propriétaires
    comme Dynatrace



Oracle Mission Control

Préparer un test de charge
Les outils : les APM (Application Performance Monitoring)

- Analyser tout le système (chaîne de liaison complète = trace)
- Peut être activé en production (avec prudence)
- Surtout pendant les campagnes de performance



Pinpoint APM

Références

- Jez Humble, David Farley-Continuous Delivery_ Reliable Software Releases through Build, Test, and Deployment Automation-Addison-Wesley (2010)
- Toby Clemson : https://martinfowler.com/articles/microservice-testing/
- « Site Reliability Engineering » par Google
- Cours gestion de version Bertrand Florat : https://florat.net/cours_vcs/cours.html