



DEVOPS

Démarche pour délivrer en continu

Programme

- **Qu'est-ce que DevOps ?**
- **La problématique DevOps**
- **Les principes DevOps**
- **Les pratiques DevOps**
- **DevOps Toolchains**
- **DevOps Toolchains & Industrialisation des Déploiements**
- **Le cloud et la virtualisation**
- **La virtualisation des environnements**
- **Automatisation du provisioning des environnements**
- **Monitoring applicatif**
- **Collaboration entre les équipes**
- **DevOps et les autres référentiels**

QU'EST CE QUE DEVOPS

Qu'est ce que DevOps

- **Bien qu'il existe de nombreuses interprétations de DevOps, la définition la plus commune est**



“Un mouvement culturel et professionnel qui met l’accent sur la communication, la collaboration et l’intégration entre les développeurs de logiciels et les professionnels des opérations informatiques tout en automatisant le processus de livraison de logiciels et les changements d’infrastructure.

Il vise à établir une culture et un environnement où la construction, le test et la mise en production de logiciels, peuvent se faire rapidement, fréquemment et de manière plus fiable.”

Wikipedia

*Améliorer la capacité de l’IT à produire des logiciels plus rapidement,
Offrir de la valeur aux clients plus rapidement*

- **Un titre**
- **Une équipe séparée**
- **Un outil**
- **Que de la Culture**
- **Que de l'automatisation**
- **Aucun opérationnel**
- **Anarchie**
- **Le grand Ouest**



5 Things DevOps is NOT
<https://devops.com/what-devops-is-not/>

Buts de DevOps

- **Versions plus petites, plus fréquentes**
- **Réduction des risques et des efforts**
- **Réduction des couts des itérations du produit et des délais**
- **Une culture de la communication et de la collaboration**
- **Consistance et rapidité grâce à l'automatisation.**

Amélioration dans:

- Time to market
- Intégration avec le métier
- Réactivité
- Qualité du code et des déploiement
- Productivité
- Visibilité
- Agilité

NoOps or NewOps

*Est-ce que la migration vers le cloud signifie la fin des opérationnels ?
Non ! Le rôle de l'opérateur change tout comme celui du développeur.*

- **Alors que les opérations d'automatisation peuvent sembler être l'objectif de NoOps, il y aura toujours un besoin de compétences et de rôles humains, y compris :**
 - Concevoir, mettre en œuvre et gérer des infrastructures internes et externes et les architectures d'entreprise
 - Fournir un soutien à la clientèle, gérer les alertes, le triage et l'escalade
 - Servir d'interface entre les fournisseurs de services cloud et les clients
 - Gestion de l'automatisation opérationnelle

Culture

Automation

Lean

Measurement

Sharing

Plus que tout autre, DevOps est un mouvement culturel basé sur les interactions entre l'humain et la technique pour améliorer leurs relations et les résultats.

DevOps va plus loin que les développeurs logiciels et les opérationnels.

- **Dev inclut toutes les personnes impliquées dans le développement de produit et services:**
 - Architectes, représentants métiers, Clients, responsable de produit, Chef de projet, assurance qualité (QA) testeurs and analystes, fournisseurs, etc.
- **Ops inclut toutes les personnes impliquées dans la fourniture et la gestion des produits et services**
 - Professionnel de la sécurité de l'informations, ingénieur système, administrateur système, ingénieur des opérations IT, release engineers, administrateurs de base de données (DBAs), ingénieur réseaux, le centre de support, fournisseurs, etc.

DevOps doit continuellement fournir des résultats en comblant et en améliorant presque tous les aspects de l'IT

- **L'informatique doit aller plus vite sans risquer la qualité**
- **Les investissements antérieurs ne fournissent pas de valeur de bout en bout**
- **Le développement agile est bon mais ne fournit pas la pleine valeur**
- **Les processus ITSM sont bons mais ne fournissent pas la pleine valeur**
- **L'automatisation est bonne mais ne fournit pas une valeur totale**
- **La culture en silo de l'IT limite le flux de création de valeur**

Le mur de la confusion

*Qu'en est-il de la sécurité, de la gouvernance,
de la gestion des risques et de la conformité? Que veulent-ils ?*

**Les Devs veulent
des changements**

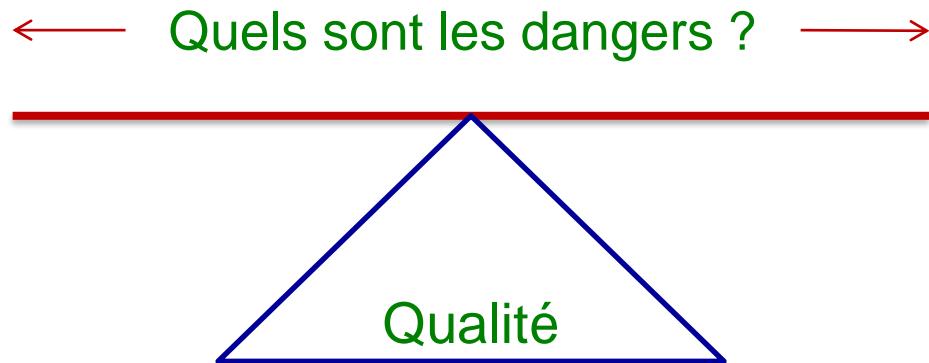


**Les Ops veulent de
la stabilité**



Le mur de la confusion

Orientation
extrême sur les
changements



Orientation
extrême sur la
stabilité

Dangers:

- Manque de qualité
 - Instabilité
 - Manque de professionnalisme
- ...

Dangers:

- Immobilisme
 - Retard sur les mises sur le marché
 - Les concurrents nous dépassent
- ...

Qu'est ce que veut le métier ? Tout !

Beaucoup de changements et de la stabilité ...

Développement



Opérations



- L'isolation en silos peuvent favoriser les stéréotypes et les idées fausses entre Dev, Ops et d'autres équipes informatiques.
- Cela peut également avoir une influence sur le flux de travail et sur la capacité de l'IT à fournir des innovations en permanence.

Jeux des divergences

- **Les divergences entre les équipes de développement et les équipes opérationnelles**
- **Les sources de conflits entre équipes (déploiement, technologies mises en œuvre...) (Même entre équipe Dev/Dev et Ops/Ops)**

Post-it

1 - Remplir des Post it :

- (une idée émotion par post-it)
- Indiquer si on vise un Dev ou Ops.
- (ex: à cause des Devs)

2 – Les coller sur un mur ou tableau blanc

3 – Organiser et classer ces post it.

4 – Débriefer sur le contenu

Exemple : Dev

- Le retour arrière n'est jamais possible.

Les 3 méthodes – (The Three Ways)

- **La première méthode – Le flux**
 - Comprendre et augmenter le flux de travail (gauche à droite)
- **La deuxième méthode – le Feedback**
 - Créer de feedback courts qui permettent une amélioration continue (de droite à gauche)
- **La troisième méthode – L'expérimentation et l'apprentissage continu**
 - Créer une culture qui va favoriser :
 - Expérimentation, la prise de risque et l'apprentissage des erreurs
 - Comprendre que la répétition et la pratique est la condition préalable à la maîtrise

La première méthode – Le flux



- **Comprendre le flux du travail**
- **Augmenter le flux en comprenant et en supprimant les contraintes**
- **Ne jamais passer un défaut connu à l'étape suivante**
- **Ne jamais autoriser un amélioration locale qui pourrait dégrader la performance globale**
- **Avoir une compréhension globale du système dans son ensemble**

Un but est d'avoir un flux de travail qui va rapidement de gauche à droite

La théorie des contraintes

*La théorie des contraintes a pour principe fondamental : le flux généré par une organisation est limité par au moins un processus, c'est-à-dire un **goulot** ou goulet d'étranglement. La production de valeur ne peut donc être augmentée qu'en augmentant la capacité de production au niveau de ce goulot d'étranglement.*
(Wikipédia)

■ La théorie des contraintes reconnaît que

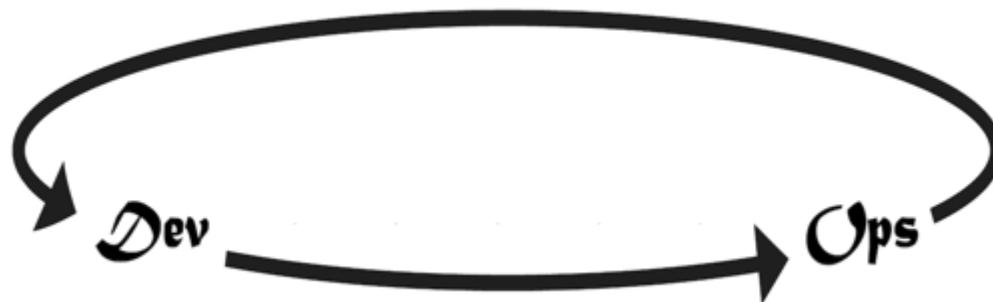
- Chaque processus a au moins une contrainte ou un goulot d'étranglement qui l'empêche d'atteindre ses buts constamment.
- La capacité du processus sera limité par son maillon le plus faible
- Améliorer cette contrainte est le moyen le plus rapide et efficace d'améliorer le système ou le processus dans son ensemble

- **Retard des développements**
- **Création des environnements (test/pré-prod, etc...)**
- **Déploiement du code**
- **Préparation et exécution des tests**
- **Audit QA ou de sécurité**
- **Architecture limité, fragile**
- **Gestion du produit, du projet**
- **Processus trop complexe or trop bureaucratique**



Quelle est la contrainte dans votre organisation ?

La seconde méthode – le feedback



- **Comprendre les besoins de tous les clients internes et externes**
- **Raccourcir et amplifier les feedbacks**
- **Créer et ancrer la connaissance là où il y en a besoin.**

Un des buts est de raccourcir et d'amplifier les feedbacks de droite à gauche, comme cela les améliorations nécessaires peuvent être faites continuellement.

Exemples de boucles de Feedback

- **Tests automatisés**
- **Peer review (Revue de paire) des changements en production**
- **Supervision/ Gestion des événements**
- **Tableaux de bord**
- **Logs de production**
- **Les mesures des processus**
- **Les post-mortems / revues**
- **Gestion des incidents, des changements, des problèmes, de la connaissance...**

La troisième méthode – L'apprentissage et l'expérimentation continue

*La troisième méthode encourage une culture qui favorise:
l'expérimentation continue, la prise de risque et l'apprentissage des erreurs;
Et de comprendre que la pratique et la répétition est un prérequis à la maîtrise*

- **Allouer du temps pour l'amélioration du travail quotidien**
- **Créer des rituels qui récompensent les équipes qui prennent des risques**
- **Introduire des erreurs dans le système pour augmenter la résilience**
- **Prévoir du temps pour des expérimentations et de l'innovation (ex:hackatons)**



- Encourager l'apprentissage quotidien et le partage des connaissances
- Créer des plans d'éducation axés sur la formation et les compétences
- Incorporer l'apprentissage dans les processus
- Utiliser la technologie pour accélérer l'apprentissage
- Expérimenter, résoudre des problèmes et faire des démonstrations
- Autoriser et utiliser les erreurs comme sources d'apprentissage
- Rendre les résultats de l'apprentissage visibles



"An organization's ability to learn, and translate that learning into action rapidly, is the greatest competitive advantage."

Jack Welsh

LES PRATIQUES DEVOPS

- **Intégration continue**
- **Livraison continue**
- **Déploiement continu**
- **Test continu**

CI - Intégration Continue (1)

L'intégration continue est une pratique de développement qui oblige les développeurs à intégrer le code dans un référentiel partagé au moins une fois par jour.

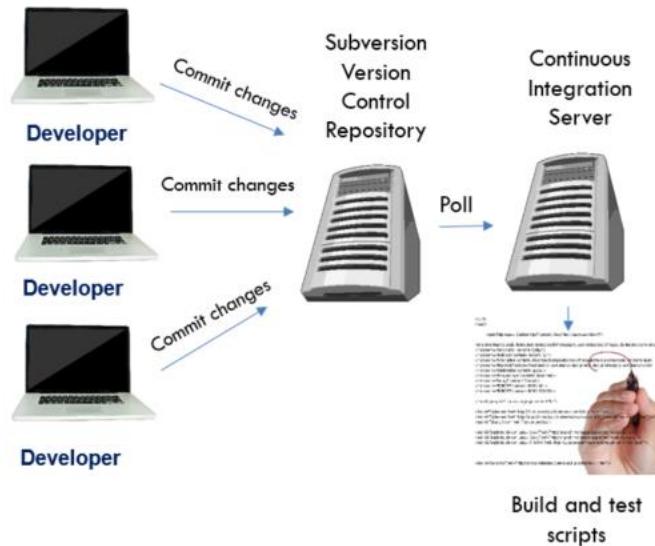
- **Chaque check-in est validé par :**
 - Un build automatique
 - Des tests unitaires, d'intégration et d'acceptance automatisés.
- **Dépendant de standard de codage consistant.**
- **Requiert des référentiels de contrôle de version et des serveurs CI pour collecter, créer et tester le code « commité ».**



L'intégration régulière dans des environnements de type production facilite la détection et la localisation rapides des conflits et d'erreurs.

CI - Intégration Continue (2)

- S'exécute sur des environnements “production-like”
- Intègre plusieurs branches en une (master/trunk)
- Doit passer les tests unitaires et d'intégration.
- Permet une détection et une correction rapide des erreurs des changements de code avant de passer en production



Bien qu'ils soient principalement associés au développement de logiciels agiles, les approches du type cycle en V (« waterfall») peuvent également profiter d'une intégration continue et de pratiques de développement axées sur les tests.

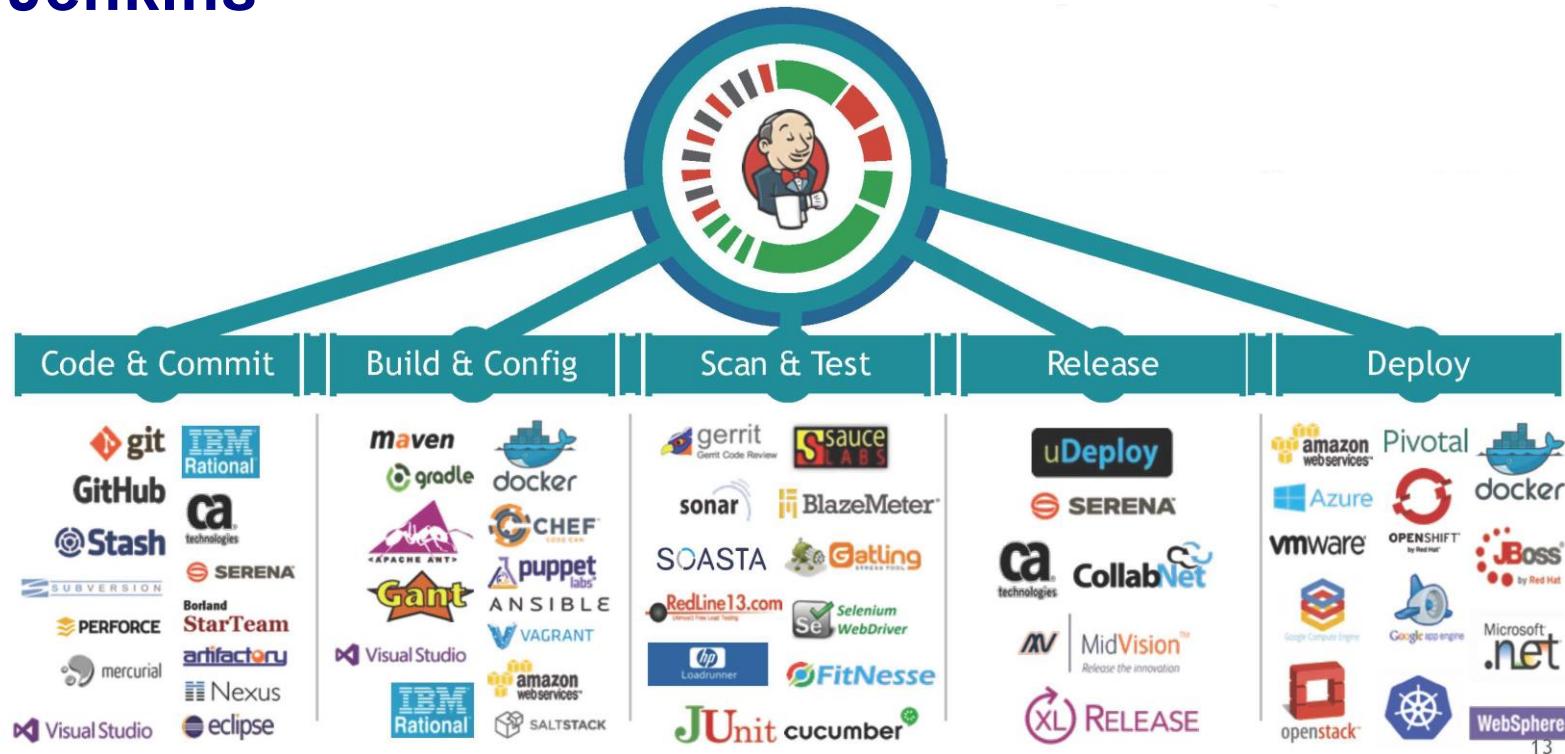
La livraison continue est une méthodologie qui met l'accent sur la certitude que le logiciel est toujours prêt à être mis en production, déployer tout au long de son cycle de vie.

- **Amène l'intégration continue CI au niveau suivant.**
- **Fournit un feedback rapide et automatique sur la possibilité de passer en production.**
- **Priorise le fait d'avoir un logiciel déployable plutôt que de travailler sur de nouvelles fonctionnalités**
- **S'appuie sur un pipeline de déploiement qui permet des déploiements à la demande**
- **Réduit le cout, le temps, les risques pour livrer de manière incrémentale.**

La livraison continue n'est pas le même que le déploiement continu !

CI and CD Are Built on Platforms and Ecosystems

- Jenkins

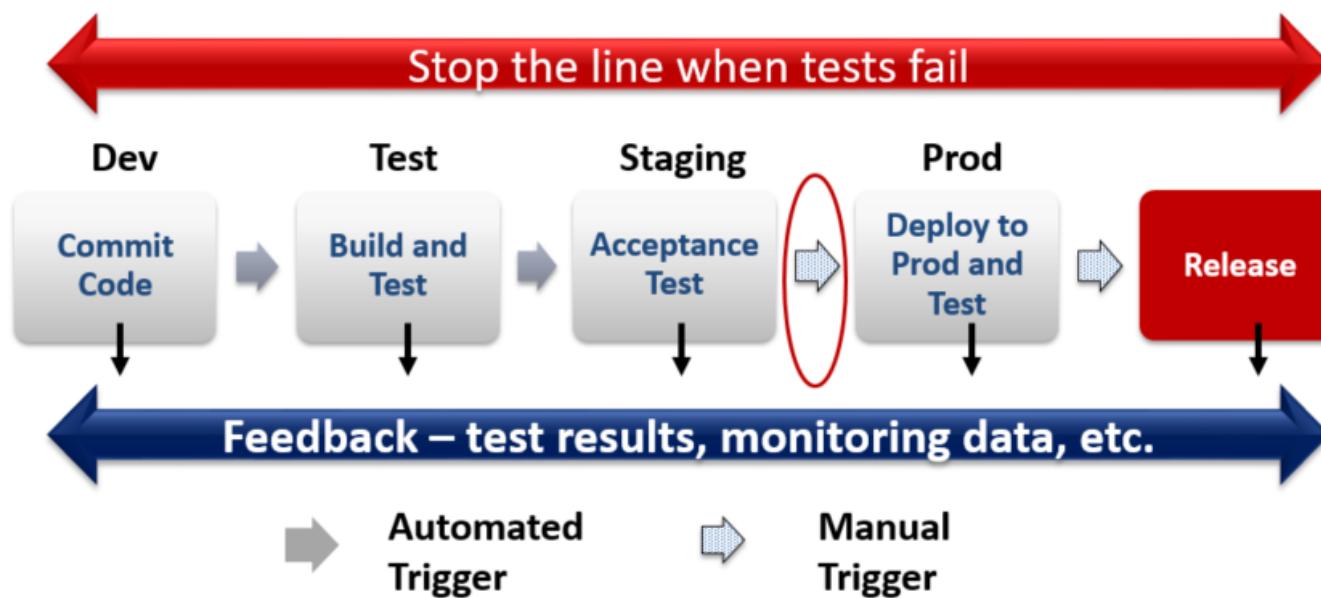


Et bien plus...

- Bien que beaucoup de ces outils soient open source, la façon dont ils sont adaptés et intégrés dans une chaîne d'outils (Toolchains) DevOps déterminera leur valeur.

Livraison et test continue

- Les tests automatisés dans des environnements de type production assurent le code et l'environnement fonctionnent comme prévu et sont toujours déployables.



Le déploiement (release) est l'installation d'une version spécifique de logiciel dans un environnement donné (par exemple, une nouvelle version en production).

Le test continue

Le test continue est le fait d'exécuter des tests automatiques comme une partie intégrante du pipeline de déploiement pour obtenir un feedback immédiat sur les risques métiers associés à version que l'on veut mettre en production.

- **Fonctionnel**
 - Tests Unitaires, API, intégration, système
- **Non fonctionnel**
 - Performance, sécurité, conformité, capacité

Déploiement continu (1)

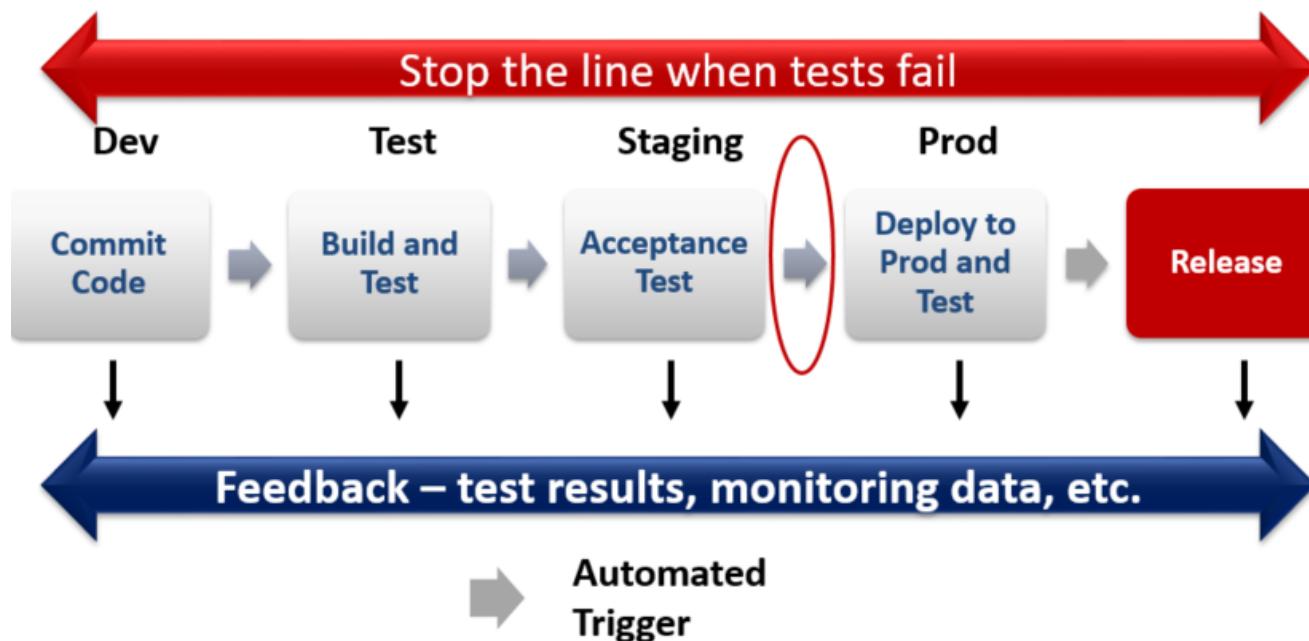
Le déploiement continu est un ensemble de pratiques qui permet à chaque changement qui passe des tests automatisés d'être automatiquement déployé en production.

- **Enlève l'étape manuelle du pipeline de la livraison continue;**
- **Permet plusieurs déploiements par jour.**



Le déploiement continu n'est peut être pas pratique ou possible pour toutes les entreprises (soumises à des contraintes réglementaires ou autres). Ce n'est pas forcément le but de toute organisation.

Le code déployé en production peut être invisible pour les clients mais les fonctionnalités peuvent être exécutées et testées par le personnel interne.



- «Release» c'est le processus ou l'événement de mettre à disposition des fonctionnalités à un segment de clients

CHAINE D'OUTILS DEVOPS

Tableau périodique des outils DEVOPS



PERIODIC TABLE OF DEVOPS TOOLS (V2)

EMBED DOWNLOAD ADD

XebiaLabs
Deliver Faster

 Follow @xebialabs

91	En	92	En	93	En	94	En	95	En	96	En	97	En	98	Pd	99	Fm	10	Pd	101	Fm	102	Fm	103	Fm	104	Pd	105	En	
Xlr	Ur	Bm	Hp	Au	Pl	Sr	Tfs	Tr	Jr	Rf	Sl	Fd	Pv	Sn	XL Release	UrbanCode Release	BMC Release Process	HP Cedar	Automic	Plutora Release	Serena Release	Team Foundation	Trello	Jira	HipChat	Slack	Flowdock	Pivotal Tracker	ServiceNow	
106	Os	107	Fm	108	Os	109	Os	110	En	111	Os	112	Os	113	En	114	Fm	115	Fm	116	Os	117	Os	118	Os	119	Os	120	En	
Ki	Nr	Ni	Zb	Dd	Ei	Ss	Sp	Le	Sl	Ls	Gr	Sn	Tr	Ff	Kibana	New Relic	Nagios	Zabbix	Datalog	Elasticsearch	StackState	Splunk	Logentries	Sumo Logic	Logstash	Graylog	Snort	Tripwire	Fortify	

Outils d'automatisation

- **Intégration continue et livraison continue**
- **Gestion de la configuration**
- **Processus de build, de test et de déploiement**
- **Création à la demande d'environnement de développement, test, production...**
- **Traiter l'infrastructure, le test, la sécurité comme du code**
- **La supervision et les tableaux de bords**
- **Expérimentation**
- **Support et exploitation**

L'automatisation seule ne fait pas le DevOps – Mais on ne peut pas réussir sans elle !

Bénéfices de l'automatisation

- **L'automation permet :**
 - Des délais livraison plus rapides
 - Des mises production plus fréquentes, moins turbulentes
 - Moins d'erreur
 - Plus de qualité
 - Une amélioration de la sécurité et l'atténuation des risques
 - Une récupération plus rapide après sinistre
 - Une satisfaction des entreprises et des clients
- **L'automatisation donne des tâches aux machines et permet aux personnes de:**
 - Analyser les erreurs, les données
 - Résoudre des problèmes
 - Prendre des décisions basées sur les retours, des données.
 - Utiliser leurs compétences, leur expérience et leur jugement

“Vos outils seuls ne feront pas votre succès.”
Patrick Debois

*Eviter les outils qui créeront
des silos !*

- **Tool chain : Chaines d'outils (vs. la solution d'un fournisseur unique)**
- **Outils partagés**
- **Self-service**
- **Penser à l'architecture du logiciel pour permettre**
 - L'automatisation des tests
 - La supervision
- **Expérimentation**

La philosophie d'une chaîne d'outils implique l'utilisation d'un ensemble d'outils intégrés, complémentaires et spécifiques à une tâche pour automatiser les processus de livraison et de déploiement.

La communication et la collaboration peuvent également être automatisées



Des outils et des plateformes novateurs facilitent et accélèrent la communication, la collaboration à travers le spectre des Dev et Ops.

Comment	Outils
<ul style="list-style-type: none">▪ Emettre des alertes et alarmes▪ Améliorer les réponses▪ Fournir la mise des statuts en un coup d'œil▪ Améliorer le workflow▪ Améliorer le flux de l'information▪ Permettre la collaboration virtuelle▪ Permettre le partage du travail, « cross-functional », cross-skilled	<ul style="list-style-type: none">▪ Plateforme de communication▪ Tableaux de bords▪ Tableau Kanban▪ Group chat rooms (chatops)▪ Outils de gestion de projet et de workflow▪ Partage de document▪ Wikis and outils de gestion des connaissances▪ Outils ITSM▪ Outils sociaux▪ Backlogs partagés

Premières étapes pour automatiser

- **Faire/Connaitre l'architecture avant d'automatiser**
- **Auditer ses outils et ses compétences d'automatisation**
- **Simplifier au préalable**
- **Ne pas automatiser un mauvais processus !**
- **Rechercher des vendeurs qui peuvent satisfaire vos besoins.**
 - **Vos besoins sont plus important que l'outil !**

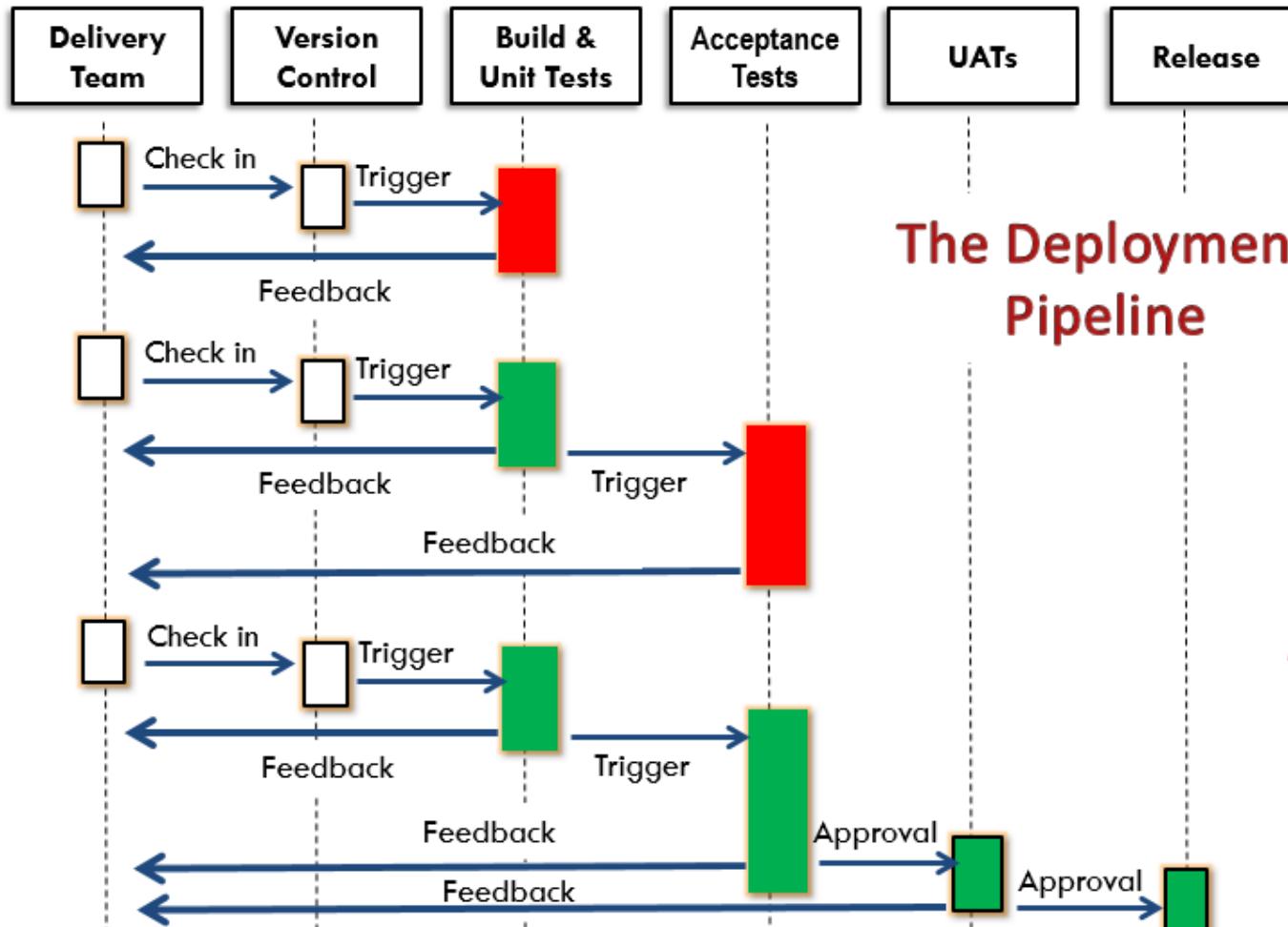


Premières étapes pour automatiser (suite)

- **Automatiser le travail à valeur ajoutée, répétitif et qui est sujet à des erreurs.**
- **Optimiser les goulets d'étranglement et la communication**
- **Améliorer les pratiques de surveillance/monitoring et de notification**
- **Attendez-vous à ce que ce soit un processus itératif - Votre chaîne d'outils évoluera avec le temps ...**

Ne sous-estimez pas l'effort et le coût de la construction de toolchains à partir d'applications open source. Open source n'est pas nécessairement gratuit. Cela signifie que vous pouvez modifier la source en fonction de vos besoins.

DEVOPS TOOLCHAINS INDUSTRIALISATION



The Deployment Pipeline

The deployment pipeline is an automated process for managing all changes, from check-in to release. Toolchains span silos and automate the deployment pipeline.

Source: Continuous Delivery:
Reliable Software Releases
through Build, Test, and
Deployment Automation

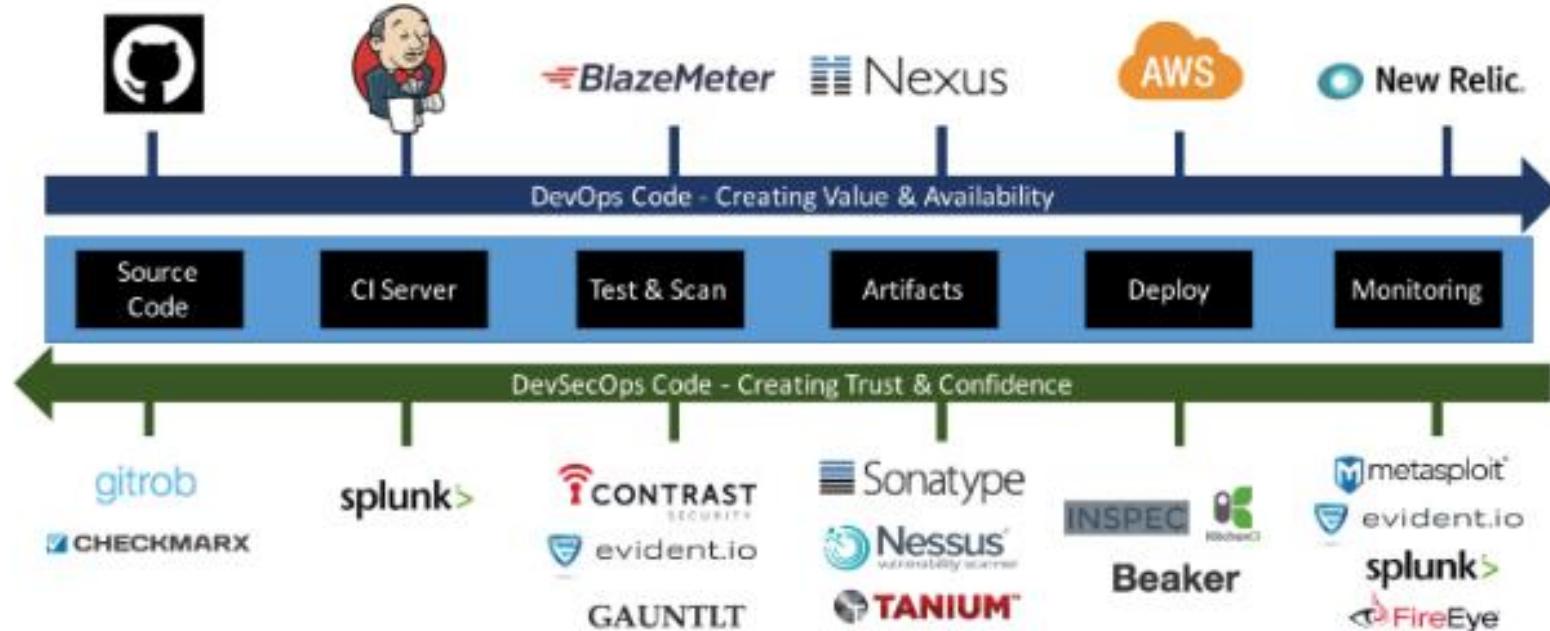
The DevOps toolchain is composed of the tools needed to support a DevOps continuous integration, continuous deployment, and continuous release and operations initiative. (Source: Gartner)

- **Une « Toolchains » Chaine d'outils automatise les tâches dans le pipeline de déploiement**
- **Chaque élément de la chaîne d'outils sert un but spécifique**
- **Les applications dans les chaînes d'outils sont connectées via des API**
- **Ils ne doivent pas être homogènes, ni d'un seul fournisseur**
- **Les chaînes d'outils sont généralement construites autour d'écosystèmes open source et fermés**
- **Nécessite une conception architecturale pour assurer l'interopérabilité et la cohérence**

Comment ces outils devraient-ils s'interfacer avec des outils opérationnels tels que des applications de surveillance ou de support?

DevOps Toolchains

■ Exemple:



La façon dont ces outils sont adaptés et intégrés à votre pipeline de déploiement déterminera leur valeur.

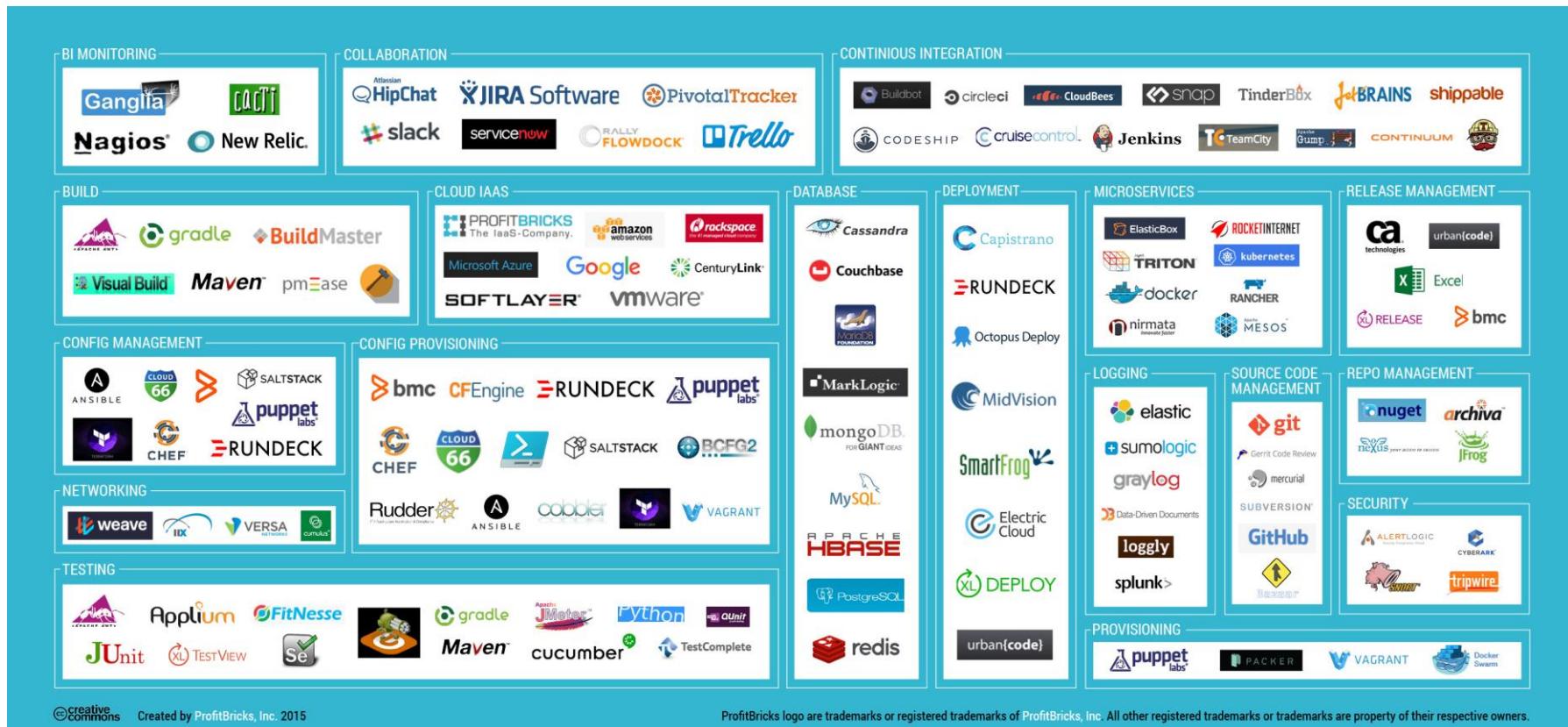
Les éléments d'une chaîne d'outils

- **Le pipeline de déploiement décompose la livraison logicielle en étapes logiques**
- **Chaque étapes fournit :**
 - La possibilité de vérifier la qualité des nouvelles fonctionnalités sous un angle différent
 - Un feedback rapide aux équipes
 - De la visibilité sur les changements
- **La chaîne d'outils DevOps rend possible l'automatisation et accélération de chaque étape**

ÉLÉMENTS TYPES

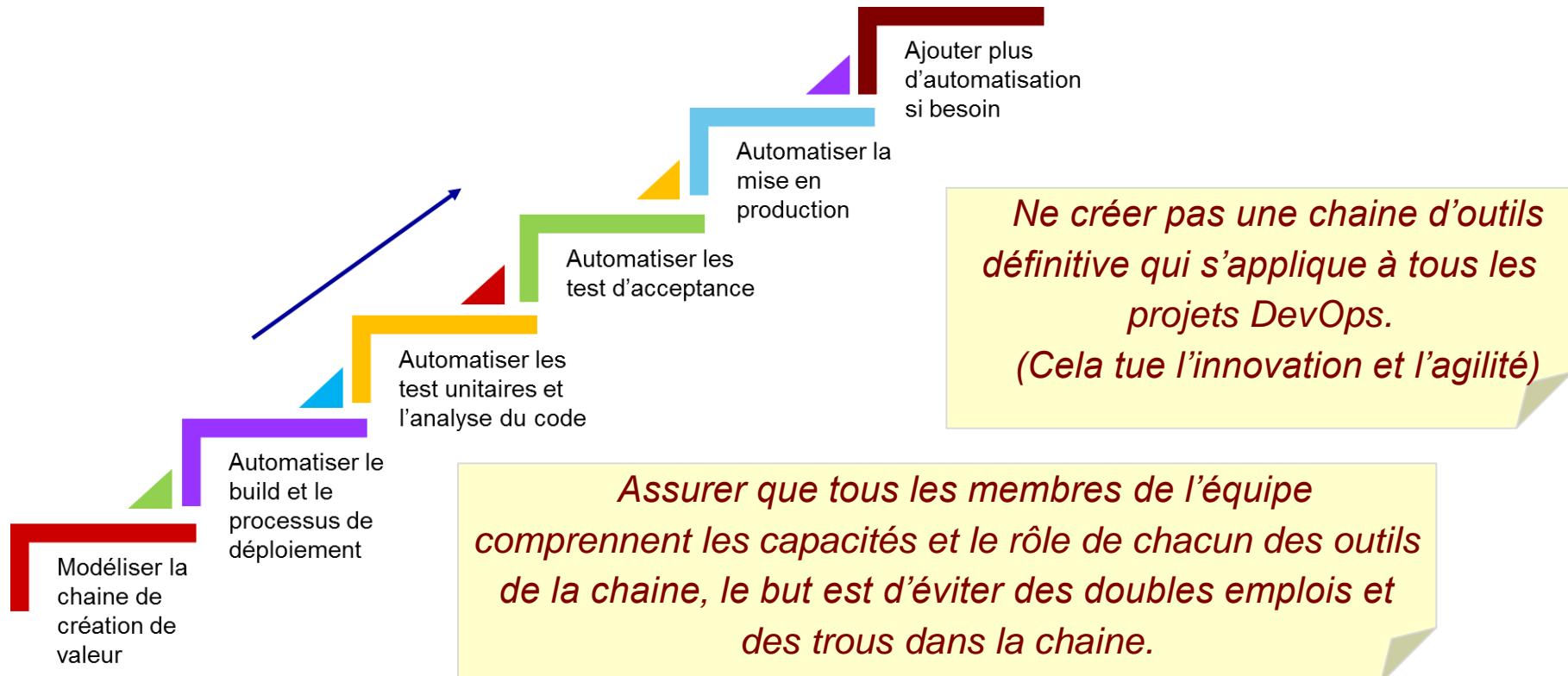
- Gestion des exigences
- Orchestration et visualisation
- Gestion des versions
- Intégration continue
- Gestion des artefacts
- Containers et virtualisation des systèmes d'exploitation
- Automatisation des tests et de l'environnement
- Configuration et déploiement du serveur
- Gestion de la configuration système
- Alertes et alarmes
- Surveillance

L'écosystème DevOps a de larges options d'outils



Construisez votre pipeline de déploiement itérativement

- La chaîne d'outils est une base qui nécessite une innovation continue et une personnalisation pour atteindre vos priorités spécifiques DevOps



Standardisation des livrables

- **Prise en compte des exigences d'exploitation, d'architecture dans la génération du livrable.**
- **Avoir des livrables standards même si les équipes et fournisseurs sont différents.**
- **Faciliter la formation et le passage de connaissances.**
- **Le déploiement sera plus facilement industrialisable.**
 - e.g. La réutilisation des procédures de déploiement)
- **Pousser les packages dans des dépôts (repository).**

Standardisation des livrables (suite)

- **Il faut intégrer les changements de données, d'environnement (DB, queue, ...) – par exemple :**
 - Règles, procédures pour gérer les jar, war et ear.
 - Comment est décomposé l'application WEB.
 - Création de package RPM pour les déploiements sur Red Hat
 - Création de Package APT pour les déploiements sur Debian
 - Git clone.
 - Ces packages ne sont pas toujours suffisant !
- **Des outils de déploiement peuvent être utilisés**
 - Fabric, Capistrano, ansible...

Quels sont vos besoins ?

Liquibase : Gestion de version des bases de données

- **Avec l'agilité on doit gérer des développement en parallèle, plusieurs branches... Les développeurs ne sont pas les meilleurs pour gérer les incrément et les changements sur les bases de données.**
- **Un outil comme Liquibase peut aider ; C'est une solution pour une synchronisation parfaite entre votre code et vos bases de données au sein d'une équipe :**
 - Suivi et gestion des versions et de tag
 - Application des changements de schéma
 - Rollback possible (à un tag, une version, à un temps donné, ...)
 - Permet de comparer des bases de données
 - Génère de la documentation
 - Supporte plusieurs SGBD
 - Plusieurs formats possibles: XML, SQL, YAML, JSON, SQL
 - Il s'intègre à Maven, Grails, Spring et Hibernate
- **Une meilleur gestion donnera plus de rapidité et une meilleur qualité.**

Versionner sa base de données

■ Bénéfices:

- Tester facilement l'état de la base par rapport à l'état du code
- Intégrer ces tests dans un processus d'intégration continue
- Suivre facilement l'état de la base et faciliter le refactoring
- Un schéma doit pouvoir être détruit et recréé de manière répétable
- Améliorer le travail en équipe car les modifications sont visibles et applicables facilement par tous
- Faciliter les mises en production puisqu'il n'est plus utile d'exécuter manuellement une suite d'instruction SQL
- Peut réduire le temps de passage en production en supprimant l'étape de passage de script par les DBA

■ Inconvénient de Liquibase :

- Une certaine complexité (XML, notion de changelog et de changeset, versionning sur id/auteur/chemin).
- La courbe d'apprentissage est assez élevée
- Orientation plutôt Dev que Ops (Il faut d'abord former et gagner la confiance des DBAs)

LE CLOUD ET LA VIRTUALISATION

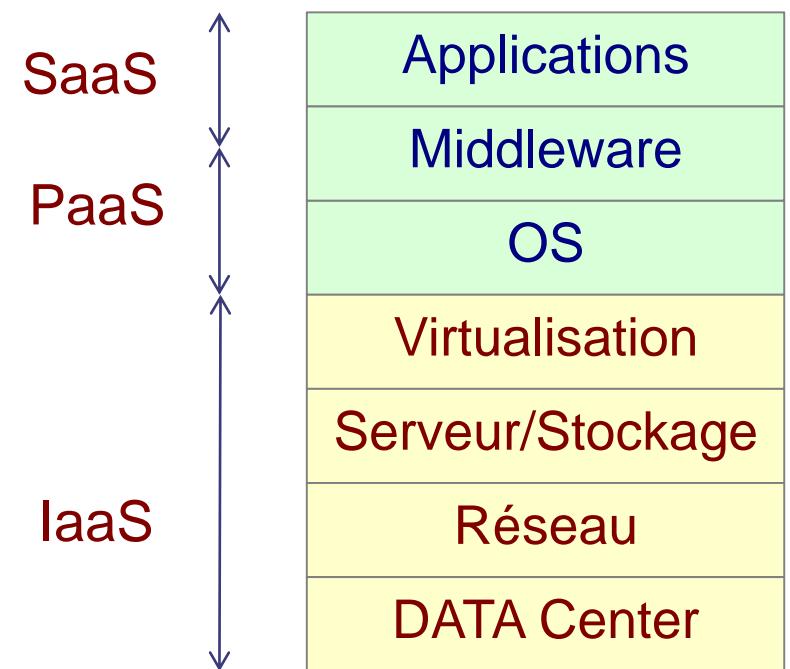
Les apports de la virtualisation hardware

- **Investissement dans le matériel réduit**
- **Optimisation et réduction des coûts**
- **Meilleure utilisation des ressources**
- **RéPLICATIONS très facile des machines**
- **Reprise après sinistre facilitée**
- **Regroupe plusieurs serveurs physiques sous-employés sur un seul hôte qui exécute des systèmes virtuels.**
- **Réduit la consommation électrique et le nombre d'administrateurs.**
- **Participe beaucoup à la réalisation des économies (locaux, consommation électrique).**
- **Création de machine à la demande.**

Cloud - Que sont l'IaaS, le PaaS et le SaaS ?

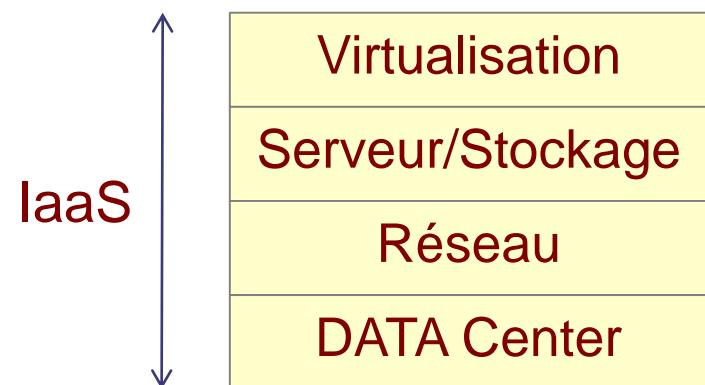
- Il existe 3 grands modèles qui ont chacun un rôle spécifique :

- IaaS : Infrastructure as a Service
- PaaS : Plateforme as a Service
- SaaS : Software as a service



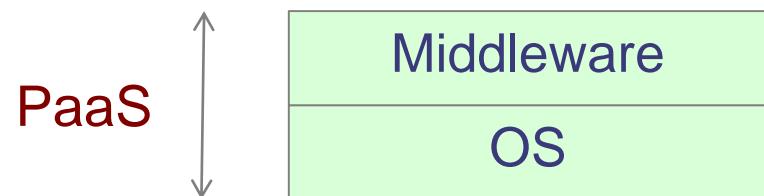
IaaS – Infrastructure as a Service

- **Extensible:** permet de faire évoluer l'infrastructure matérielle (serveurs, réseau et stockage)
- **Externalisation, dématérialisation de l'infrastructure matériel**
- **Permet de payer à la consommation**
- **Optimisation des couts**
- **Redondance, continuité de service**
- **Ex de fournisseur:**
 - OVH, Amazon EC2



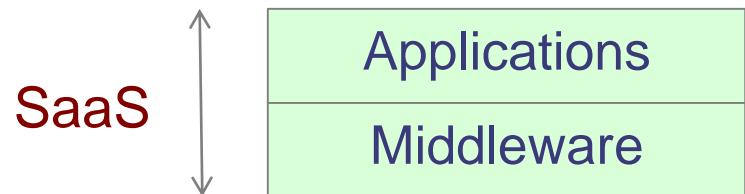
PaaS – Plateforme as a Service

- Repose sur l'IaaS
- Externalise les applications middleware : bases de données, couches d'intégration de données et environnements de développement des applications.
- Facilite l'accès aux non experts
- Flexible, adaptable.
- Continuité, back-up, sécurité des données
- Ex: Hébergement site web, serveur mail...

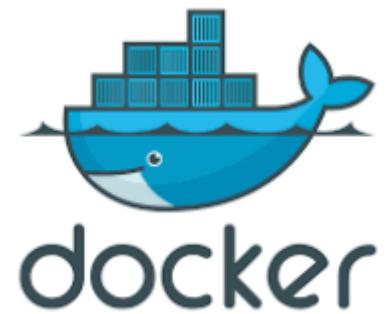


SaaS – Software as a Service

- **Accès à des applications via internet**
- **Pas de coûts d'équipements supplémentaires**
- **Pas de frais d'installation**
- **Paiement à l'utilisation**
- **TCO (cout total de possession) maîtrisable et plus accessible.**
 - Investissement initial faible
- **Utilisation évolutive**
- **Maintenance et mise à jour automatique**
- **Ex: CRM, Visioconférence, JIRA**



Quelques outils de virtualisation



Définir un processus commun et répétable

- **Plusieurs étapes sont nécessaire à la fourniture d'un environnement :**
 - Machine
 - Réseau
 - Configuration
 - Application
 - Middleware
 - Données (stockage et db)
 - Un environnement, c'est un ensemble de machine sur lesquelles tournent des applications/middleware/données ...

Une palette d'outils pour le déploiement, provisioning !



SALTSTACK



Lequel choisir ?
Pourquoi ?
Que faire des
nouveaux
produits?

etc...

Quels critères pour choisir son outils ?

- **Culture plutôt Dev ou Administrateur système ?**
- **Facilité de prise en main ?**
- **Utilisation du Ruby ou python ou fichier de configuration**
- **Mise à l'échelle**
- **Technologie**
- **Interopérabilité**
- **Gestion des configurations**
- **Maturité du produit**
- **Support et formation**

Puppet vs. Chef vs. Ansible vs. Salt

Puppet

Client master

Automatic push



- Cookbook (Livre de recette)
- Ruby
- Plutôt esprit Dev que Ops (Ruby)

Ansible

No client master

Action via SSH

Push method



- Python
- Simple command for simple tasks
- YAML configuration file : Playbooks.
- Ops Friendly

Chef

Client master, Workstation

No automatic push



- Cookbook (Livre de recette)
- Ruby
- Plutôt esprit Dev que Ops (Ruby)

Salt

No client master

Action via SSH

Push method



- Python
- Client sont appelés mignons
- YAML configuration file : states.
- Ops Friendly

Puppet vs. Chef vs. Ansible vs. Salt

- Alors que Puppet et Chef s'adressent aux développeurs et aux entreprises axées sur le développement, Salt et Ansible sont beaucoup plus à l'écoute des besoins des administrateurs systèmes ;
- Ansible a une interface simple et conviviale s'adapte parfaitement à la mentalité admin système ;
 - Ansible a beaucoup d'outils pour gérer des systèmes linux et unix. Ansible est rapide et simple à appréhender.
- Salt est le plus robuste des quatre ;
 - Comme Ansible, il résonnera avec les administrateurs de système.
 - Il est très évolutif et très performant,
 - Salt est bloqué uniquement par son interface Web.

Puppet vs. Chef vs. Ansible vs. Salt

- **Puppet est la plus mature et probablement la plus accessible des quatre du point de vue de l'utilisabilité, bien qu'une solide connaissance de Ruby soit fortement recommandée.**
 - Puppet n'est pas aussi simple qu'Ansible ou Salt, et sa configuration peut parfois devenir complexe.
 - Puppet est le pari le plus sûr pour les environnements hétérogènes, alors qu'Ansible ou Salt pourrait être une meilleur choix dans une infrastructure plus grande ou plus homogène.

Puppet vs. Chef vs. Ansible vs. Salt

- **Chef est stable et bien conçu;**

- Alors qu'il n'est pas tout à fait au niveau que Puppet en termes de caractéristiques brutes, c'est une solution très performante.
- Chef a une courbe d'apprentissage plus difficile aux administrateurs qui manquent d'expérience de programmation,
- Il pourrait par contre être plus naturel pour les développeurs et les administrateurs orientés développement.

DÉPLOIEMENT APPLICATIF

Capristano

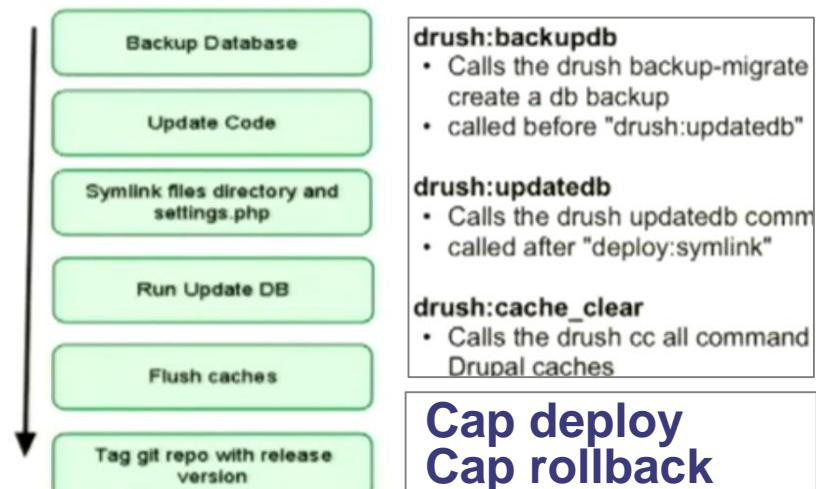


■ Capristano :

- Action via SSH
- Simple d'utilisation
- Déploiement automatiques avec quelques « recettes »
- Déploiement en plusieurs phase (workflow)
- Rollback intégré (à scripter)
- Se connecte à des outils de provisioning comme Ansible, chef.
 - Supporte AWS
 - Ruby (Plus orienté Développement)

■ Exemples :

- Current : lien vers version actuelle
- Release : 20170121, 20170120
- Shared file



Fabric

- **Fabric :**

- Action via SSH
- Simple d'utilisation
- Fichier de configuration unique
- Apprentissage rapide
- Maintenance facile
- Python
- Définition d'étapes à exécuter via la ligne de commande.
- Exemples (ci-contre) :

```
from fabric.api import *
env.use_ssh_config = True

def production():
    env.hosts = ['ec1']

def clone():
    with cd('~/webapps/fabric_demo'):
        run('git clone https://github.com/edgecaselabs/fabric-demo.git myproject')

def checkout(branch=None):
    with cd('~/webapps/fabric_demo/myproject'):
        run('git checkout %s' % branch)

def pull():
    with cd('~/webapps/fabric_demo/myproject'):
        run('git pull')
        #sudo('git pull', user='webapp')

def migrate():
    with cd('~/webapps/fabric_demo/myproject'):
        run('python3.4 manage.py migrate')
```

Fabricfile.py

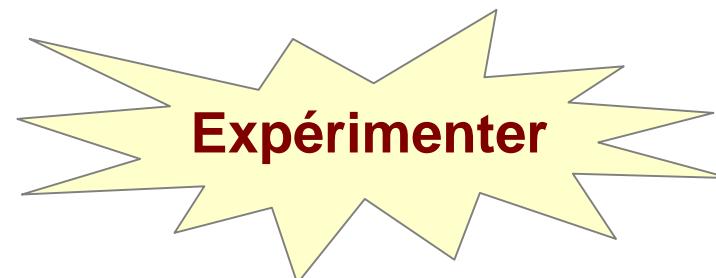
```
production pull migrate
Executing task 'pull'
run: git pull
```

Command line
Exécution et résultat

```
Executing task 'migrate'
run: python3.4 manage.py migrate
out: Operations to perform:
out:   Synchronize unmigrated apps: messages, staticfiles
out:   Apply all migrations: admin, contenttypes, sessions, auth, widgets
out:   Synchronizing apps without migrations:
out:     Creating tables...
out:       Running deferred SQL...
out:       Installing custom SQL...
out:     Running migrations:
out:       Rendering model states... DONE
```

Retours réels

- **Ne pas se cloisonner à un seul outil !**
- **Pourquoi ne pas prendre 1, 2 ou X outils différents ?**
 - Parfois un outil fait très bien une chose, mais est très complexe pour en réaliser une autre.
 - La maintenance de ces outils et des scripts doit être pris en compte.
 - N'oubliez pas le LEAN ! La valeur pour le client; C'est le résultat qui compte; Resté simple.
 - Ne pas transformer son pipeline en usine à gaz !
 - L'open source n'est pas gratuit.
 - Les options payantes ne font pas tout.



MONITORING APPLICATIF

Monitoring Applicatif 1/2

- **On ne fait pas que regarder si la base de données est accessible**
 - On doit détecter les erreurs, la disponibilité, mais pas seulement
 - On doit passer au niveau supérieur de la supervision !
- **La supervision doit être prévue dès la conception**
 - Disponibilité
 - Capacité (Performance, utilisation des ressources....)
 - Sécurité
 - Etc...
- **Le monitoring doit aider à améliorer le Temps moyen de rétablissement des services - Mean Time To Recover MTTR**

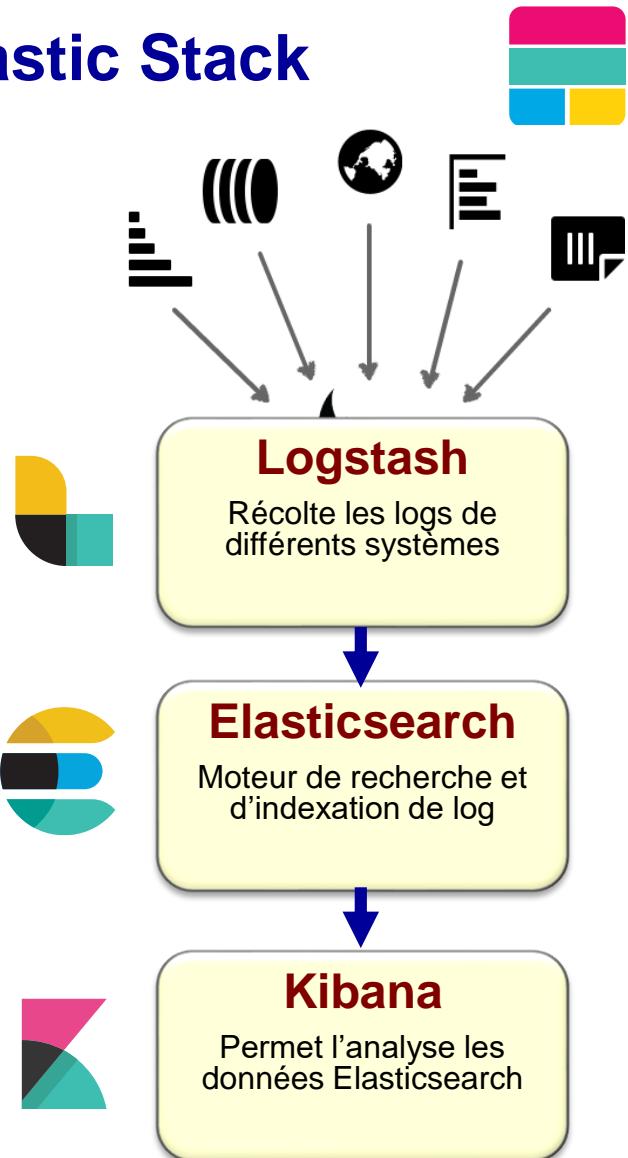
Monitoring Applicatif 2/2

- **Bonne Pratique : Application / Services qui s'auto-diagnostique ;**
 - Ex: Nagios détecte que la base de données est accessible; une application ne sait pas s'y connecter
 - *Pourquoi ne le dit-elle pas ?*
 - Cela aidera les opérationnels à détecter la source de l'incident. On peut demander au Dev de faire des pages de supervision qui indique si le service sait se connecter à son écosystème.
- **Maitriser et de suivre les performances et la consommation virtualisée ou cloud :**
 - Si on utilise le cloud et que l'on paye à la consommation, ou même si on utilise la virtualisation
- **Quelques outils:**
 - Nagios, Zabbix, AWS CloudWatch ,New Relic, etc...

Centralisation des logs applicatifs

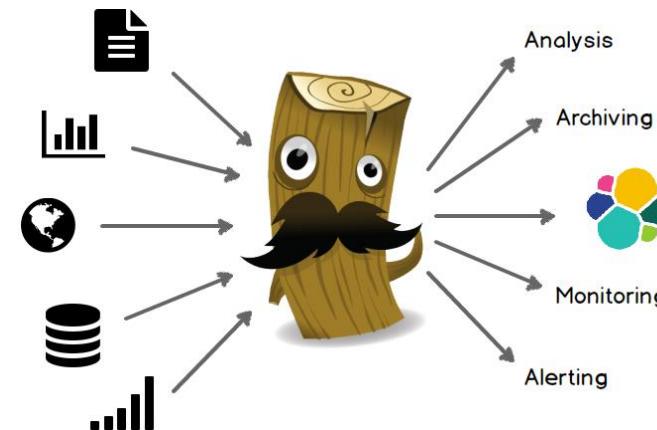
- **La multitude des serveurs**
- **Multitude type de serveurs/logs**
- **Découpage d'application en microservice**
- **Font que c'est très difficile de les suivre**
- **-> outils de centralisation des logs**
- **Exemples d'outils Splunk / ELK (ElasticSearch - Logstash - Kibana)**

- **Nouvelle dénomination du produit Elastic Stack**
- **Logstash :**
 - Récolte, filtre, traite et envoie les logs
- **Elasticsearch**
 - Index et permet de recherche dans les logs récoltés
- **Kibana**
 - Permet de visualiser, explorer les données
 - Génération de rapport, tableau de bord
 - Faire des relations entre les données



Logstash

- C'est un ETL : Extract Transform Load
- Logstash est un pipeline open source qui ingère et traite simultanément des données provenant d'une multitude de sources, puis les transforme et les envoie vers votre stockage préféré.



Logstash

■ Les entrées :

- Fichier texte, fichier de log (ex: log apache, log4j)
- Site WEB
- Base de données et flux de données
- Objets connectés (IoT)
- Git
- Etc...
 - (<https://www.elastic.co/guide/en/logstash/current/input-plugins.html>)

■ Les filtres :

- Analyse et transforme les données
- Enrichissement des données, structuration de données
- Uniformisation du format
- Obtenir une localisation à partir d'une adresse IP
- Bibliothèques de filtres très riches
 - (<https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>)

Logstash

■ Les sorties :

- ElasticSearch
- Email
- HipChat (ChatOps)
- JIRA (ex: création de ticket ou anomalie automatique)
- Nagios
- Etc...

- **Permet d'indexer les données et faire des recherches très rapides ;**
- **Mise à l'échelle facile, permettant de gérer des nœuds :**
 - 1 PC ou une multitude de serveurs pour une grande quantité de données ;
 - Permet une résilience des données.
- **Offre des fonctionnalités de recherche à facettes (affinage de la recherche) ;**
- **Utilise le moteur d'indexation LUCENE ;**
- **Interface accessible via JSON et java ;**
- **Outils d'aide à son administration.**

- **Mise en page de vos recherches, de vos rapports ;**
- **Exploration vos données ;**
- **Visualisation facilitée ;**
 - graphique, tableau, histogramme, pie chart, carte ;
- **Exploration grâce aux relations des données;**
- **Plus que jolie, c'est utile et puissant ;**
- **Vue alternative et moins standard ;**



DEVOPS DANS L'ENTREPRISE

RÔLES

Combler le manque de compétence DevOps

- La demande de ressources DevOps rend difficile, pour les organisations, d'attirer et de retenir les talents ;
- Le rythme vertigineux auquel les technologies évoluent rend difficile pour les individus de maintenir un ensemble de compétences à jour ;
- S'assurer que les individus ont des compétences de communication, collaboration (Soft Skills) et une culture adaptée pour relever le défi.

■ Strategies :

- Formation et certification ;
- Coaching et programme d'immersion ;
- Restructuration de la rémunération et de la culture d'entreprise ;
- Compléter les équipes internes avec des talents externalisés ;
- Bonus de recrutement ;

Les directeurs informatiques d'aujourd'hui cherchent des travailleurs capables de changer de vitesse et de s'adapter aux changements technologiques.

Compétences et caractéristiques DevOps

▪ Compétences

- **Métier** : Connaissance des priorités et des processus métier
- **Technique** : Spécialiste avec une large connaissance généraliste – Expérience ou avec au moins de l'intérêt pour écrire du code
- **Soft** : Communication, collaboration, travail d'équipe
- **Autogestion** : prise d'initiative, gestion du temps et du stress, auto-motivation, concentré

Les connaissances techniques du généraliste incluent une compréhension des pratiques de DevOps, des pratiques d'ingénierie de logiciel modernes et des architectures modernes.

Compétences et caractéristiques DevOps (suite)

■ Stratégies:

- Adaptable ;
- Orienté client ;
- Artisan ;
- Curieux ;
- Data-driven, orienté par les données ;
- Engagé ;
- Empathique;
- Transparent.

Rôles DevOps

▪ Les rôles suivants apparaissent comme essentiels pour le succès de DevOps :

- Leader, coach DevOps ;
- Ingénieurs logiciels, développeurs et testeurs ;
- Release manager ;
- Architecte d'automatisation de livraison continue ;
- Ingénieur d'intégration continue, de build ;
- Ingénieur sécurité ;
- Assurance Qualité (QA) ;
- Ingénieur d'opérations de DevOps ;
- Support Informatique ;
- Agile Service Manager® ;
- Agile Process Owner ® ;

Quels autres rôles pensez vous nécessaires à DevOps ?



Source: <http://techbeacon.com/7-devops-roles-you-need-succeed>

Qu'est-ce qu'un ingénieur DevOps ?

- Il n'existe actuellement aucune description de poste reconnue par l'industrie ou une piste de carrière formelle pour un ingénieur de DevOps
- Il y a des avantages et des inconvénients avec le concept d'une équipe de DevOps, il y a des pour et des contres...
- Les caractéristiques générales incluent quelqu'un qui:
 - Souhaite contribuer avec son talent technique aux initiatives d'amélioration du métier, des processus ;
 - Aime collaborer avec les autres ;
 - Veut être dans un milieu de travail qui favorise une culture du partage.

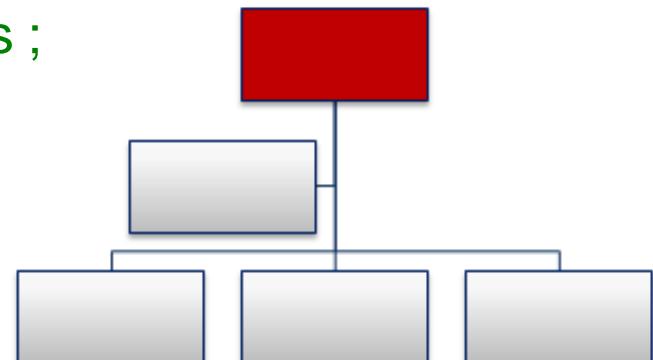


CONSIDERATIONS ORGANISATIONNELLES

Structure d'organisation DevOps

■ Voici des possibilités:

- Faire des liaisons entre des opérationnels et des équipes agiles/scrum ;
- Créer des équipes produits multi compétences (vs équipe projet) ;
- Créer des services opérationnels partagés ; qui donneront du support aux différentes équipes de développement.



There is debate about the pros and cons of DevOps teams.

Equipe DevOps (1)

■ Equipe DevOps :

- Etendre le concept d'une équipe Agile ou Scrum
- Intégrez les compétences de développement et opérationnelles dans un seul groupe générique
- Peut être temporaire ou dédié à un produit spécifique
- Peut être des «équipes de tigres»(« Tiger Team ») interprofessionnelles pour des projets à court terme
- Peut évoluer pour offrir des services partagés
- Ont des responsabilités partagées
- Doivent adhérer aux normes définies pour le développement, l'automatisation, le risque et la conformité qui s'applique à toutes les équipes de DevOps



Il n'y a pas de structure idéale pour une équipe DevOps !

The creation of DevOps departments or teams is a growing trend. 22% of 2016 State of DevOps survey respondents indicated they were part of a DevOps department.

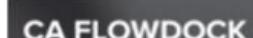
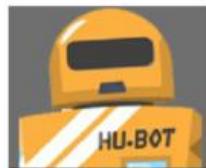
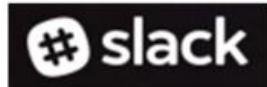


■ Inconvénients des équipes consacrées à DevOps :

- Moins d'engagement dans le flux de valeur IT ;
- Risque d'être un autre silo ;
- Dev et Ops se lavent les mains de leur responsabilité ;
- Les activités de DevOps deviennent le problème de quelqu'un d'autre.

Indépendamment de la structure, une équipe de DevOps doit avoir un engagement continu, avoir un bon équilibre entre les personnes, les pratiques et les compétences d'automatisation.

Chat client + chat bots = Développement basé sur les conversations, fourniture de support.

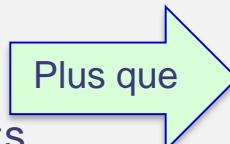


Eric Sigler	I think something's up with the latest changes to Apollo.
Eric Sigler	!lock apollo production
Officer URL	 apollo production locked
Eric Sigler	Here's a log snippet from the web server:
Eric Sigler	enhance "10.0.0.1", "-", "02/Dec/2014:06:34:32 +0000", "GET /someurl HTTP/1.0", "-", "-", "200", "3704", "-", "a-totally-fake-user-agent", "127.0.0.1"
Officer URL	"*a-fake-production-webserver-dns-name* (US-WEST-2B, environment: production)", "-", "02/Dec/2014:06:34:32 +0000", "GET /someurl HTTP/1.0", "-", "-", "200", "3704", "-", "a-totally-fake-user-agent", "127.0.0.1"
Eric Sigler	!status
Officer URL	Status: NORMAL
Eric Sigler	!define me apollo
Officer URL	apollo is a highly available service that creates and delivers On-call Handoff Notifications to Mercury.
Eric Sigler	All clear
Eric Sigler	!unlock apollo production
Officer URL	 apollo production unlocked

- La transparence de ChatOps raccourcit les boucles de feedback, améliore le partage d'informations, améliore la collaboration des équipes et permet une formation croisée.
- Il peut également être utilisé pour diminuer le MTTR.

AGILE

▪ Concepts du développement Agile

- Les individus et leurs interactions
 - Des logiciels opérationnels
 - La collaboration avec les clients
 - L'adaptation au changement
- 
- Les processus et les outils
 - Une documentation exhaustive
 - La négociation contractuelle
 - Le suivi d'un plan

*Nous reconnaissons la valeur des seconds éléments,
mais privilégiions les premiers.*

Qu'est-ce qu'être agile ?

- **Être orienté client ;**
- **Être économique (LEAN) ;**
- **Être collaboratif ;**
- **Être communicatif ;**
- **Être adaptable ;**
- **Être mesurable ;**
- **Être cohérent ;**
- **Être axé sur les résultats ;**
- **Être réfléchi.**



*Il est plus important «d'être agile»
que de «faire de l'agile».*

- **Scrum est un cadre simple permettant d'avoir une collaboration efficace d'équipe pour des projets complexes.**
- **Scrum fournit un ensemble de règles qui créent une structure «juste assez» pour que les équipes puissent se concentrer sur leur innovation et sur la résolution de ce qui peut être vu à la base comme un défi insurmontable... (Scrum.org)**
- **Scrum constitue LA pratique de développement logiciel Agile, la plus couramment utilisée;**
 - Extrêmement simple mais difficile à maîtriser
 - Absence de processus ou de technique pour fabriquer les produits applicatifs ;

Scrum augmente les possibilités de mettre en production plus fréquemment



▪ Roles :

- Product Owner
- Scrum Master
- Team

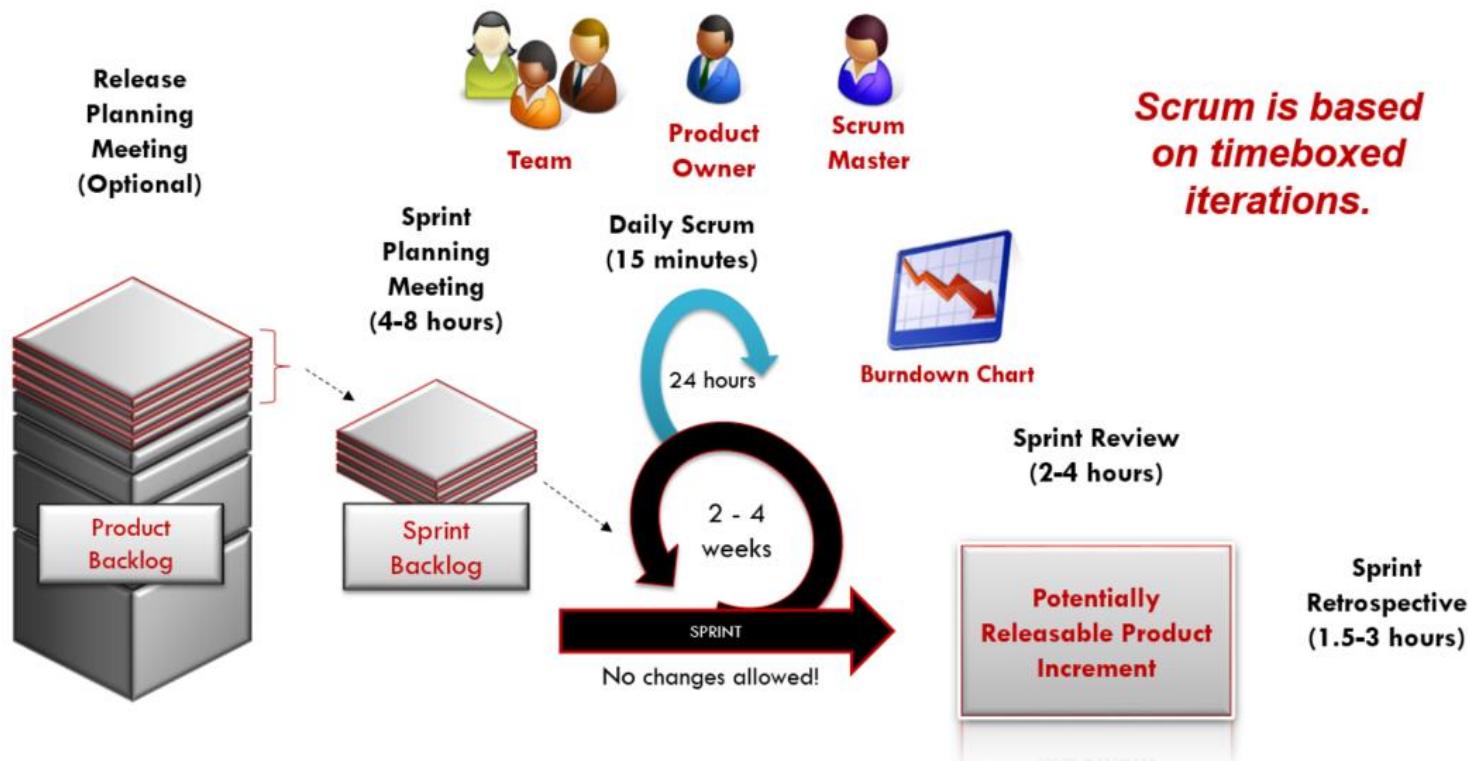
▪ Artifacts :

- Product Backlog
- Sprint Backlog
- Burndown Charts
- Potentially Shippable Increment

▪ Meetings :

- Release Planning
- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

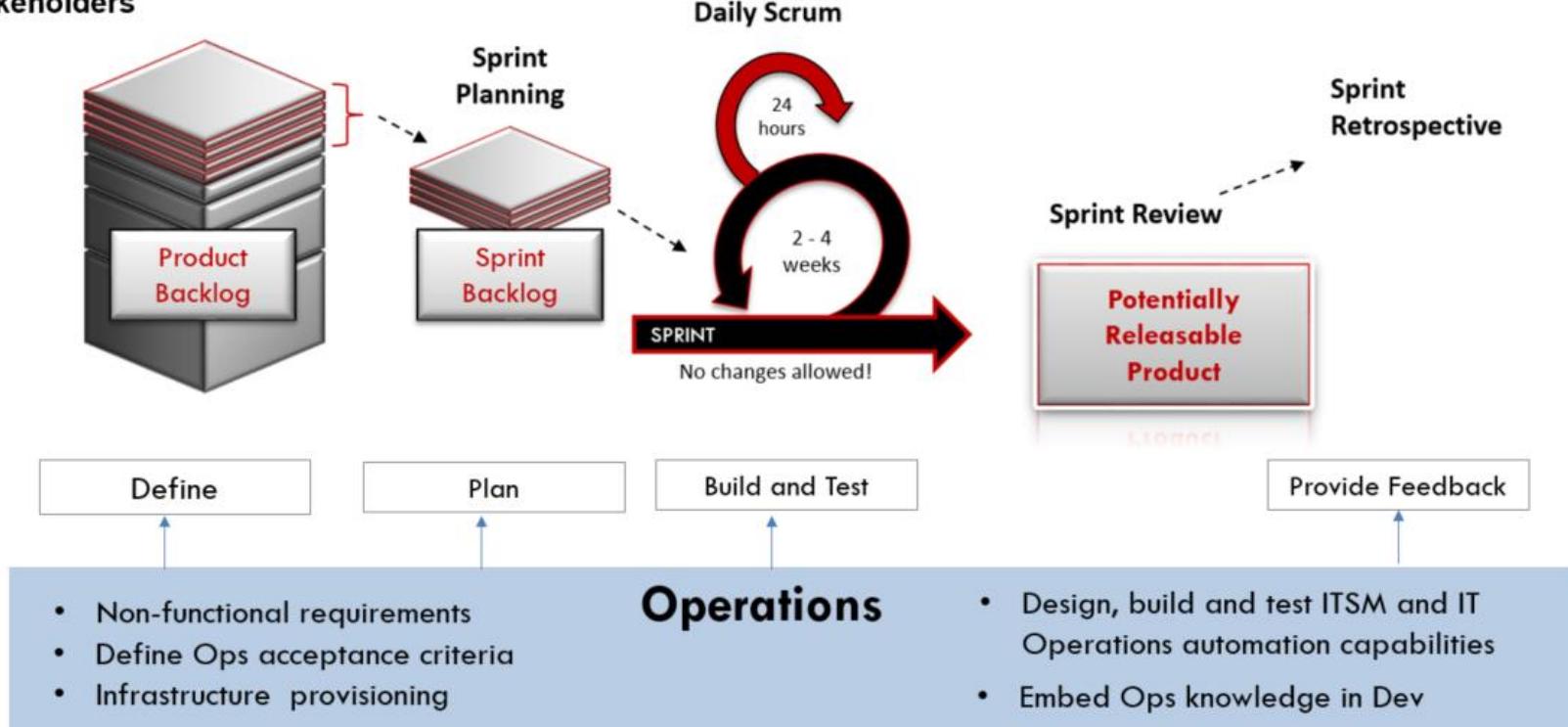
Scrum = 3 Rôles + 4 Artefacts + 5 Evènements



Implication Ops dans SCRUM

Ops are also involved in the
Definition of Done

Inputs from executives,
customers, users, team,
stakeholders



Augmenter l'agilité

■ DevOps augmente l'agilité en :

- Brisant les silos
- Améliorant les contraintes
- Adoptant une approche unifiée de l'ingénierie des systèmes
- Appliquant des principes agiles au Dev et aux opérations
- Partageant les connaissances, les compétences, l'expérience et les données
- Reconnaissant la criticité de l'automatisation
- Déployant plus rapidement avec moins d'erreurs



DevOps étend les principes agiles au-delà des frontières du logiciel à l'ensemble du service livré.

DEVOPS & ITSM

ITSM - IT Service Management

- **La gestion de services IT (ITSM) couvre la mise en œuvre et la gestion de services IT de qualité qui répondent aux besoins de l'entreprise:**
 - Fournit une structure cohérente à des processus tels que la gestion des changements, de la configuration, des mises en production, des incidents et des problèmes ;
 - Les processus ITSM reposent sur l'ensemble du cycle de vie des services, depuis la stratégie, la conception, la transition, les opérations et l'amélioration continue ;
 - Se concentre sur la création de valeur ;
 - DevOps a besoin de pratiques ITSM pour atteindre l'objectif de déployer des changements plus rapidement sans causer de perturbations.

Des processus de gestion de service répétables peuvent ouvrir la voie à une livraison continue et une production accrue.

- **ITIL est un ensemble de publications sur les meilleures pratiques en matière de gestion des services ;**
- **Le cœur d'ITIL se compose de cinq livres qui fournissent des conseils sur les meilleures pratiques pour une approche intégrée des processus de gestion des services :**
 - ITIL® Service Strategy
 - ITIL® Service Design
 - ITIL® Service Transition
 - ITIL® Service Operation
 - ITIL® Continual Service Improvement



DevOps et ITSM

- **DevOps ne s'arrête pas au déploiement et au support de début de vie ,**
- **ITSM ne démarre pas au moment du déploiement et au support de début de vie ;**
- **Les processus ITSM sont essentiels aux opérations en cours, à l'amélioration continue et à la création de valeur ;**
- **DevOps a besoin des meilleures pratiques ITSM pour atteindre l'objectif de déployer des changements plus rapidement sans causer de perturbations ;**
- **L'intégration de DevOps et ITSM permet d'identifier, de réduire et d'éliminer les risques et les contraintes en cours ;**

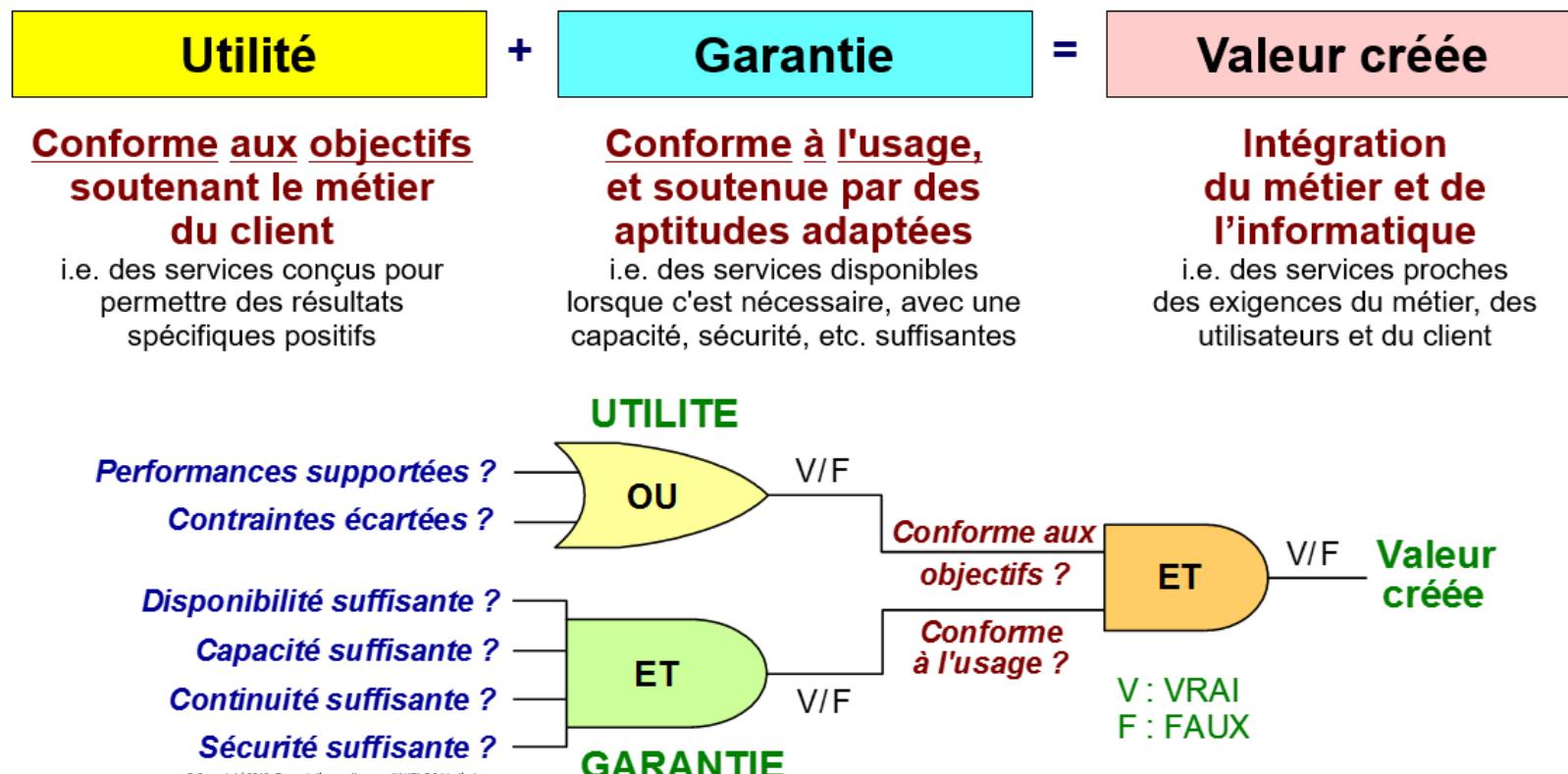
- Chaque organisation a des processus ITSM en place ;
- Beaucoup basés sur les meilleures pratiques d'ITIL ;
- ITIL encourage les organisations à adapter en permanence leur processus aux besoins actuels de l'entreprise ;
- Réalisez ceci en appliquant des principes agiles, lean et DevOps sur une base de bonnes pratiques d'ITIL ;
- Commencez par les processus ITIL clés



La création de valeur selon ITIL

■ La création de “valeur” d'un service

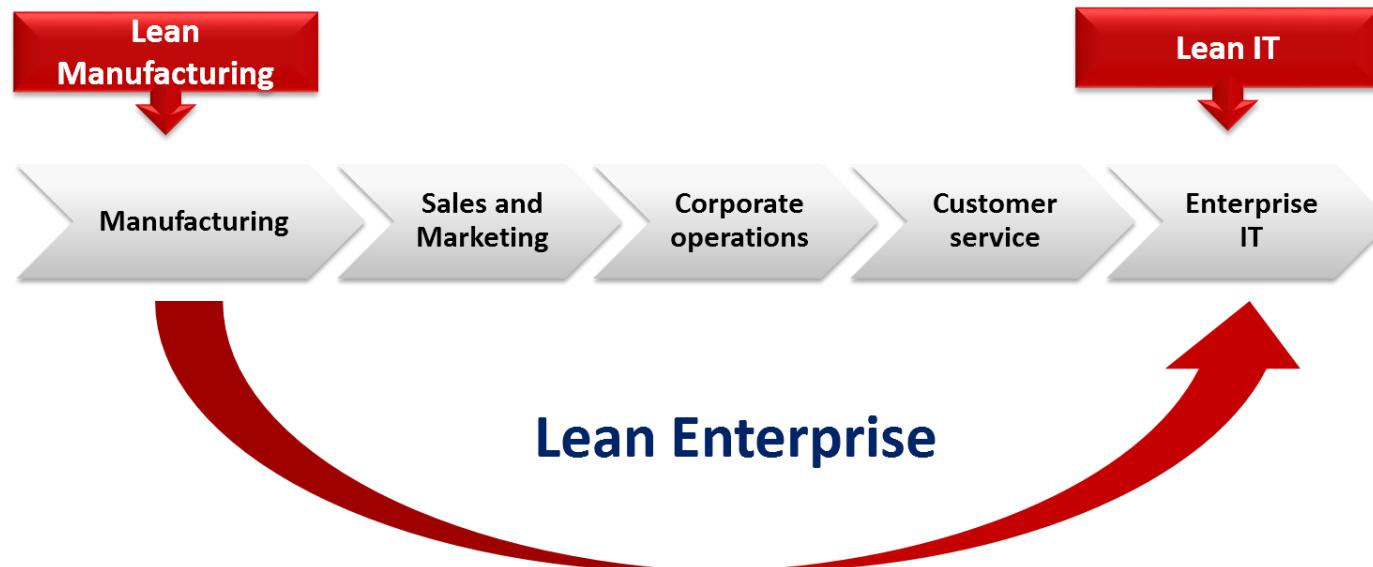
- l'utilité correspond à ce que le client obtient,
- la garantie décrit la manière avec laquelle elle est délivrée



LEAN

Perspectives du Lean

- Le Lean IT applique les idées clés de la gestion de production LEAN au développement et à la gestion de produits et services informatiques.



Termes Importants

- **Lean (adjective)**
 - Économique, dépourvu de richesse ou d'abondance
- **Lean Production**
 - Une philosophie de production axée sur la réduction des déchets et l'amélioration du flux des processus pour améliorer la valeur globale des clients
- **Flux**
 - Décrit la façon dont les personnes ou les produits passent par un processus

Termes Importants

- **Flux continue:**

- Transférer doucement des personnes ou des produits de la première étape d'un processus jusqu'au dernier avec un minimum (ou aucun) d'attente entre les étapes

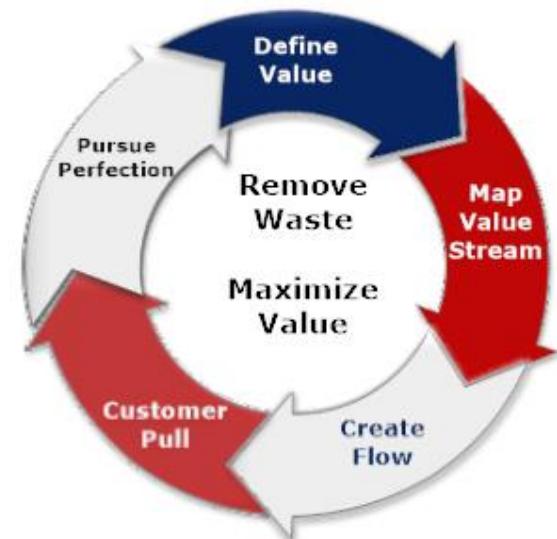
- **Chaine de création de valeur (Value Stream Mapping):**

- Lean outil qui représente le flux d'information, de matériaux et de travail à travers les silos fonctionnels avec un accent mis sur la quantification des déchets, y compris le temps et la qualité.

OUTILS	SUJET
A3 thinking :	Résolution des problèmes
Flux continue	
Kaizen :	Amélioration continue
Kanban :	Pull system (Flux tiré)
KPI :	Key Performance Indicator/ Indicateur clé de performance
Plan Do Check Act :	PDCA - Cycle de Deming (Préparer, Faire, Vérifier Agir)
Process mapping :	Visualiser le processus pour identifier les domaines d'amélioration
Root cause analysis :	Identification des causes d'origine des erreurs, des déchets
SMART goals :	Specific Measurable Achievable Relevant Time-bound
Value stream mapping :	Représenter le flux

Les 5 principes du LEAN

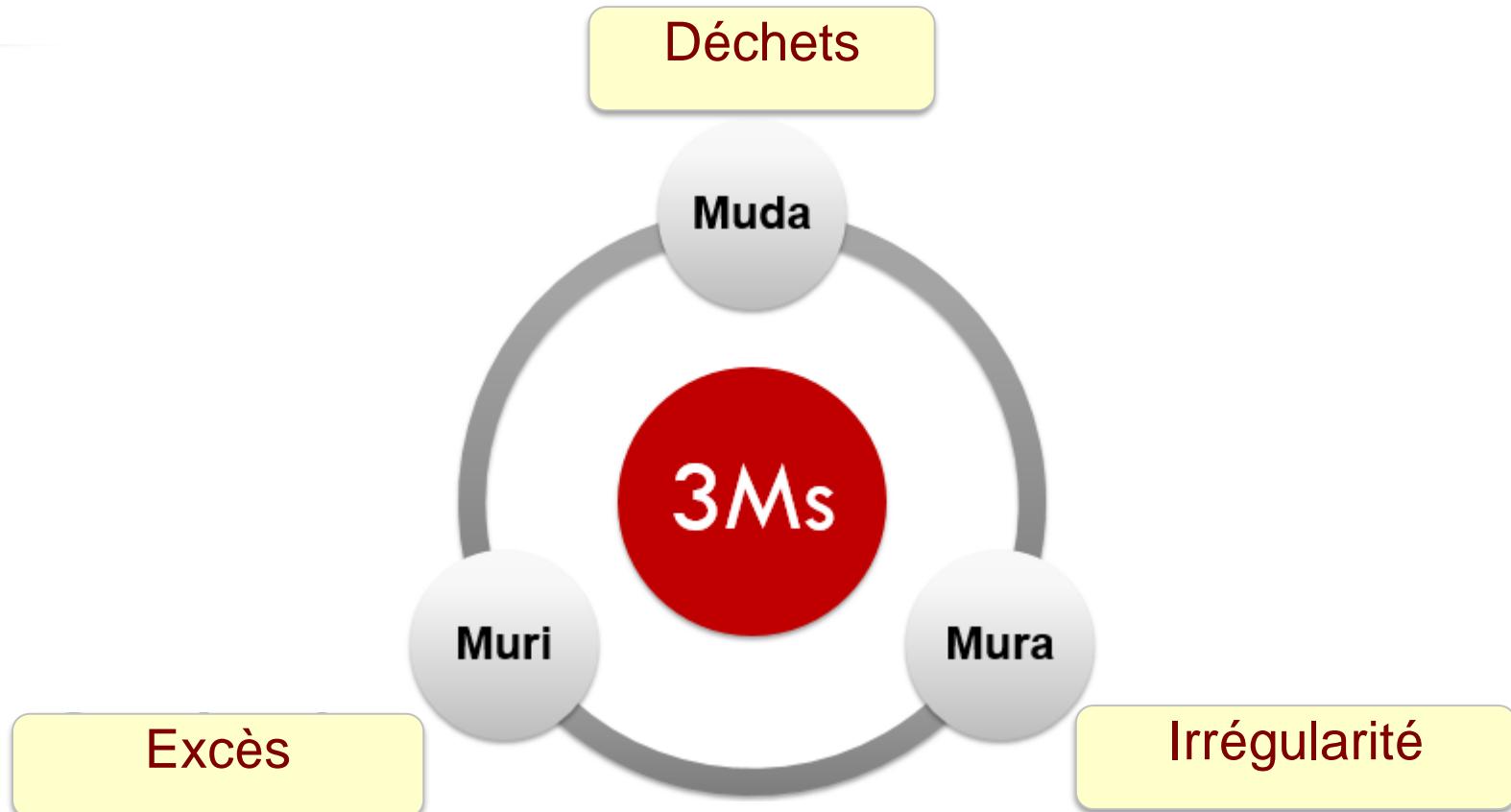
- Définir précisément la valeur du point de vue du client final ;
- Identifier l'ensemble de la chaîne de création de valeur pour chaque service, produit ou famille de produits et éliminer les déchets ;
- Faire passer les étapes de création de valeur restantes ;
- Lorsque le flux est initié, laissez le client tirer ce que le client veut quand le client le veut ;
- Poursuivre la perfection.



The voice of the customer (VOC) :

Le processus de la voix du client capture et analyse les exigences du client et ses retours pour comprendre ce que le client veut.

- Les Mura et Muri provoquent des Muda (déchets).



Sources de déchets = Temps d'arrêt

- **L'objectif de la pensée LEAN est de créer plus de valeur pour les clients, avec moins de ressources et moins de déchets;**
- **Les déchets sont des activités qui n'ajoutent pas de valeur au processus.**

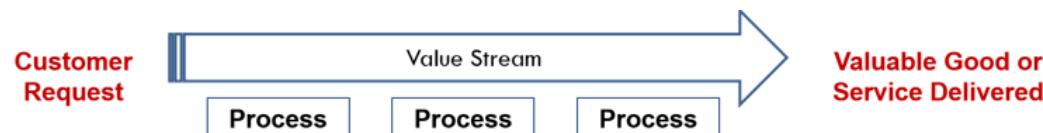
Source	Objectif	Exemples
Défaut	Écarts par rapport aux exigences; les erreurs	Anomalies, erreurs connues, désinformation
Surproduction	Produire trop et plus rapidement que nécessaire	Code ou documentation excéssifs
Attente	Retards du à l'attente d'une étape précédente	Retard de décision, d'approbation, de réponse.
Non utilisation	Compétence, créativité non utilisé	Compétences, innovation, communication inutilisées
Inventaire	Prendre plus de matériaux que nécessaire	Logiciels, infrastructure, arriérés ou non utilisés; courriels excessifs
Déplacement	Déplacer des personnes ou des biens plus souvent que nécessaire	Déplacer, changer trop souvent de code ou d'infrastructure
Traitement Excessif	Faire plus que nécessaire	Over-engineering, mission de faire des templates ou actifs réutilisables

PRATIQUES DU LEAN

Les pratiques du LEAN sont si pertinentes pour DevOps, que le Lean a été récemment ajouté comme l'une des valeurs principales de DevOps.

Value Stream Mapping

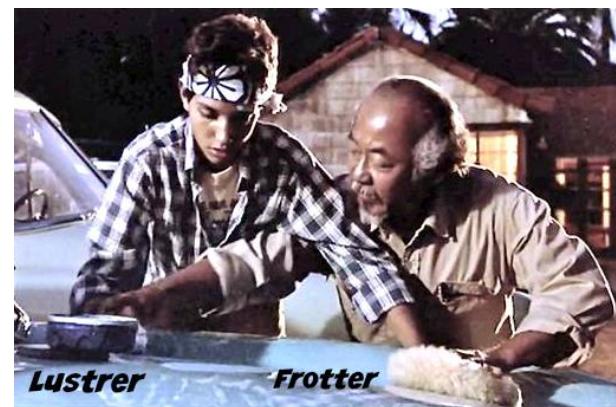
- La chaîne de création de valeur est un outil LEAN qui permet de représenter le flux d'informations, des matériaux et du travail à travers des silos avec une quantification des pertes, de la qualité et du temps :
 - C'est la séquence des activités requises pour concevoir, produire et livrer un produit ou un service spécifique ;
 - Les flux de valeur traverse généralement plusieurs processus
 - La chaîne de création de valeur permet aux équipes multi-compétences de voir le flux complet du point de vue travail et information ;
 - Identifie les zones de déchets, de non valorisation qui pourraient être éliminées dans le but d'améliorer le flux et d'offrir une plus grande valeur ;
 - Identifie, priorise et mesure les améliorations.



- “L’improvement Kata” est issu du système de production de Toyota comme un moyen structuré de créer une culture d’apprentissage continu et d’amélioration:

- Un kata est une manière structurée de penser et d’agir que vous pratiquez jusqu’à ce que le modèle devienne une habitude ;
- Par la pratique, un modèle de comportement devient une seconde nature ;

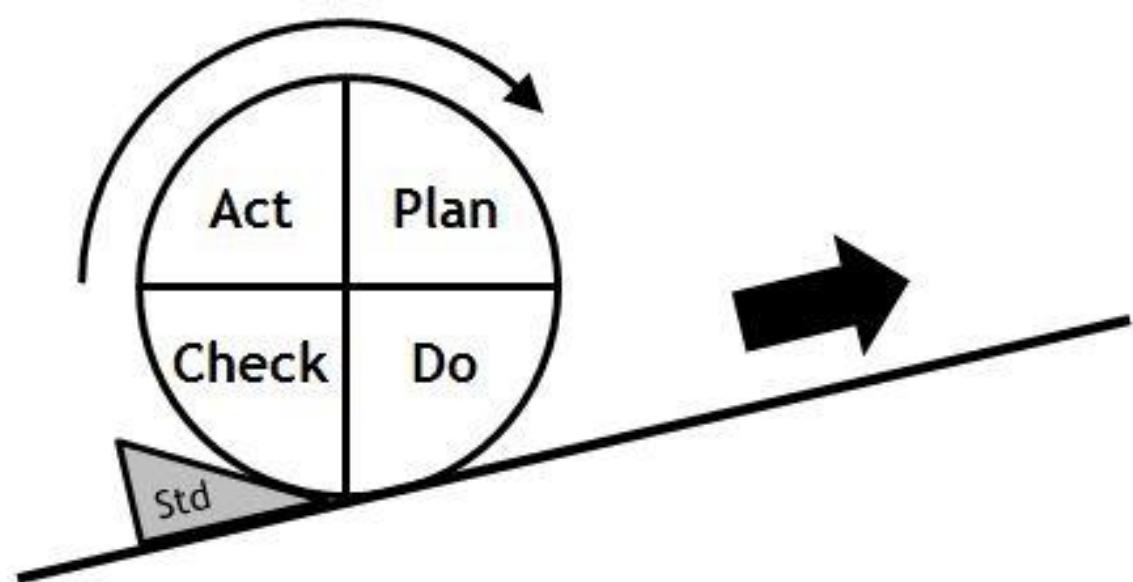
Pratiquer « l’improvement kata » devrait être quotidien, et pas seulement un projet complémentaire ou «en temps voulu».



Le Cycle de Deming (PDCA)

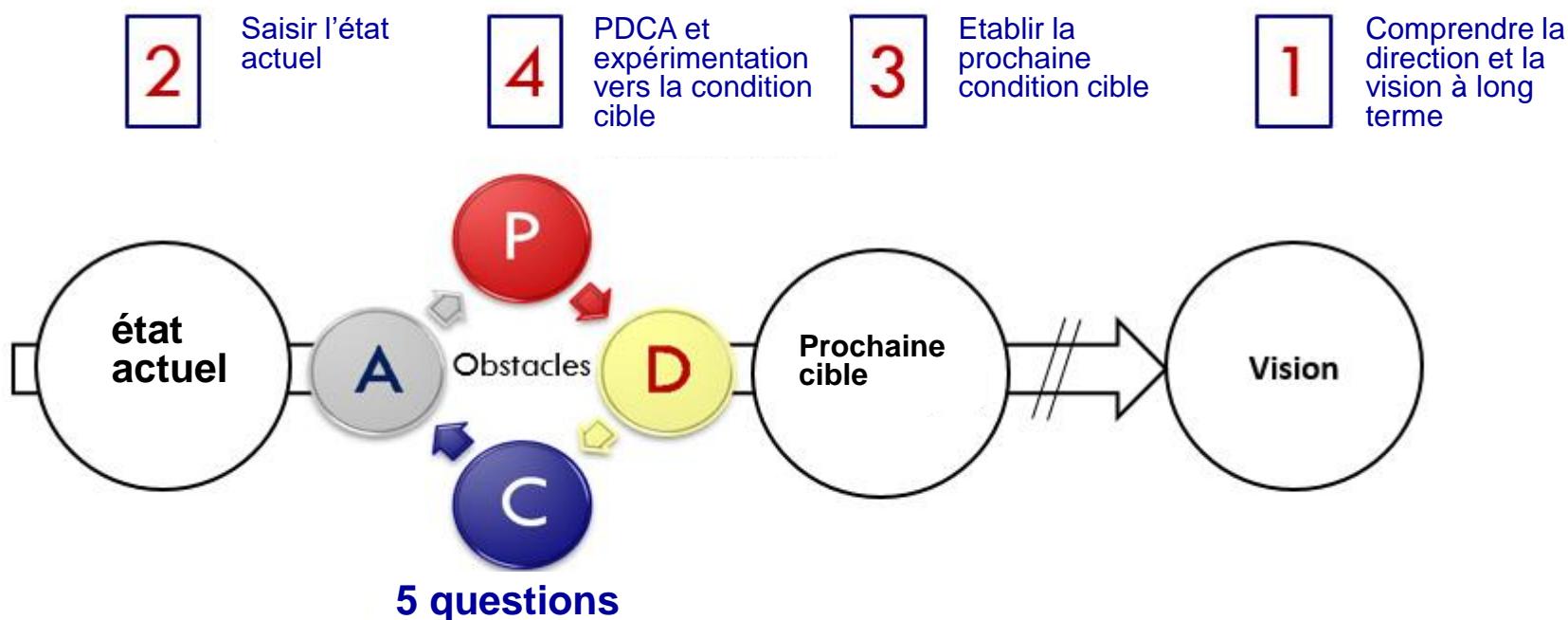
- **Le cycle de Deming permet une amélioration continue en exécutant ces 4 étapes simples:**

- Plan: Préparer
- Do: Faire
- Check : Vérifier
- Act: Agir



The Improvement Kata – 4 étapes

- L'Improvement Kata est un processus en quatre étapes qui se concentre sur l'apprentissage et l'amélioration du travail.
- Il tient compte de la vision ou de l'orientation à long terme de l'organisation :



Improvement Kata – 5 Questions

- 1. Quel est la prochaine cible?**
- 2. Quel est la situation actuelle?**
- 3. Quels obstacles vous empêchent d'atteindre votre prochaine cible ?**
- 4. Quelle est votre prochaine étapes ? (Prochain PDCA/ expérimentation)**
- 5. Quand pouvons voir ce que nous appris en faisant cette étape ?**

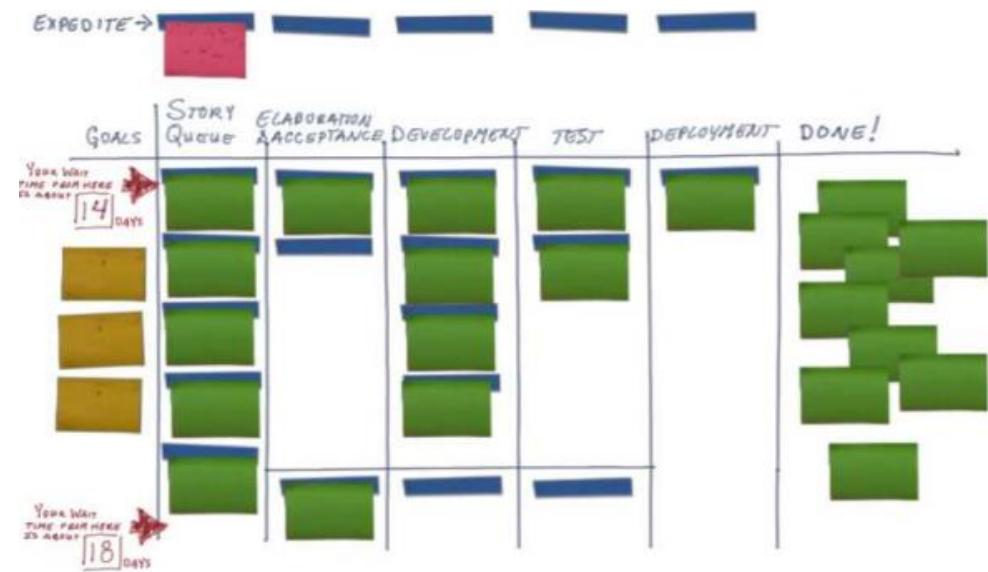
Les équipes utilisant l'improvement Kata apprennent à s'efforcer d'atteindre une condition cible et s'adaptent en fonction de ce qu'ils apprennent.



KANBAN

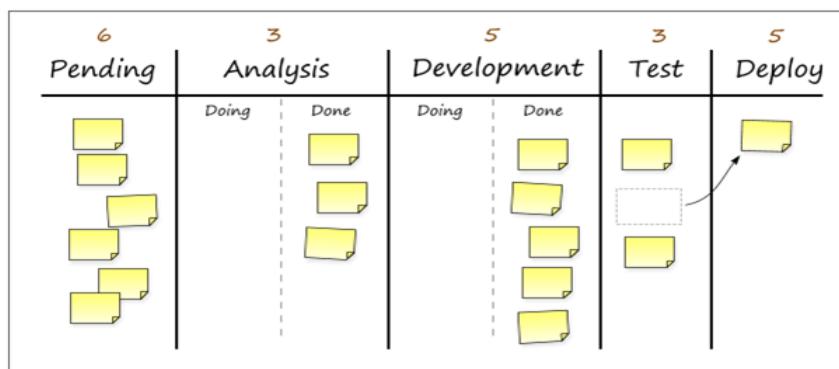
- Kanban propose d'augmenter la fluidité et de délivrer en continu (Throughput) :

- Travailler en flux tiré,
- Se focaliser sur le travail en cours (WIP),
- Se focaliser sur la qualité,
- Visualiser les flux.



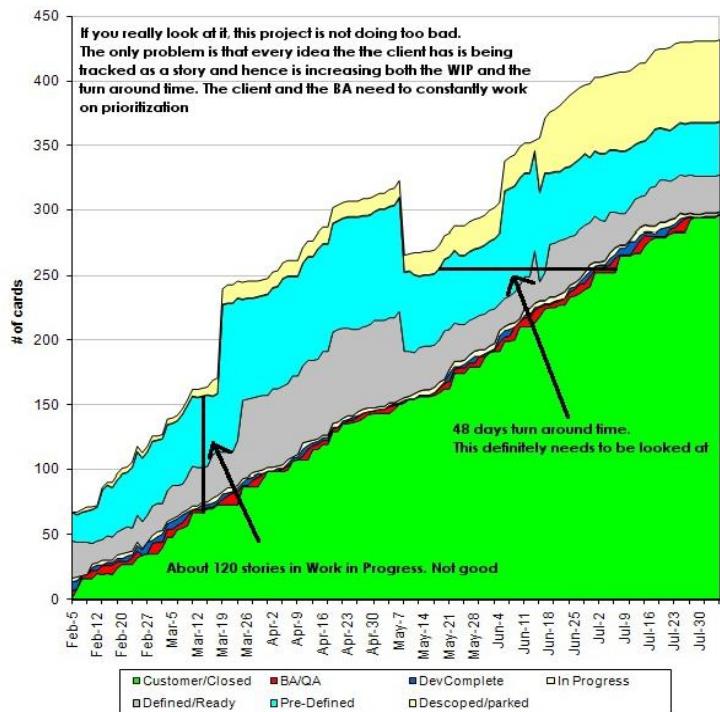
▪ Idées principales :

- Créer des items de petite taille et visualiser leur état,
- Optimiser le flux global. La seule chose vraiment importante est la quantité de travail délivrée,
- On limite le travail en cours (le moins de stock possible -> Méthode Pull).
 - Chaque colonne doit avoir un nombre maximum de fiches.



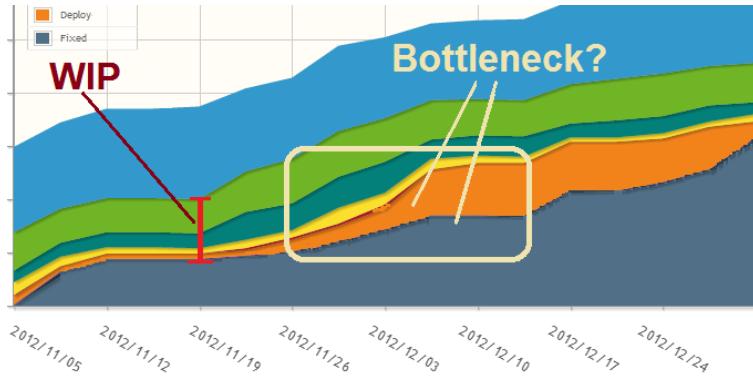
KANBAN : Visualiser pour limiter le WIP

- **WIP : Work In Progress / Travail en cours ;**
- **Lead Time : Temps entre la demande et la livraison ;**
- **Trop de travail à faire signifie probablement :**
 - Vous ne finissez pas ce qui est engagé,
 - Une étape ultérieure ne peut pas intégrer votre travail.



KANBAN : Optimiser

■ Trouver le « BottleNeck »



Inutile d'en rajouter !

Design	Development	QA Testing	UAT	Deployment	Done
As a user, I need to be able to request admin rights.	As a admin, I need to be able to invite a user to join	As a user, I need to be able to sort	As a admin, I need to be able to invite	As a admin, I need to be able to set permissions for a u...	As a user, I need to be able to request admin rights.
As a user, I need to be able to change my password.	As a admin, I need to be able to set permissions for a u...	As a user, I need to be able to sort	As a user, I need to be able to sort all of the users ...	As a user, I need to be able to sort items	As a admin, I need to be able to invite a user to join.
As a user, I need to be able to filter catalog items	As a user, I need to be able to sort	As a admin, I need to be able to sort	As a user, I need to be able to sort items	As a user, I need to be able to login to the application...	As a user, I need to be able to change my password.
		As a user, I need to be able to sort	As a admin, I need to be able to delete users.	As a admin, I need to be able to delete users.	
			As a user, I need to be able to change my password.		

- Soulager le goulot,
- Élargir le goulot,
- Trouver le goulot suivant.

Mise en œuvre de KANBAN

- **Apprenez à connaître votre système ;**
- **Identifiez et priorisez vos sources ;**
- **Définissez votre processus ;**
- **Concevez votre tableau de flux ;**
- **Définissez les limites ;**
- **Décidez des rôles ;**
- **Décidez des réunions.**

KANBAN, les réunions

- **On distingue 3 grands types de réunions :**

- Réunion de définition, d'expression du besoin ;
- Réunion quotidienne ;
- Réunion d'apprentissage, d'amélioration continue et d'analyse du feedback.

