

DAWAN Paris  
DAWAN Nantes  
DAWAN Lyon

11,rue Antoine Bourdelle, 75015 PARIS  
32, Bd Vincent Gâche, 5e étage - 44200 NANTES  
Bt Banque Rhône Alpes, 2ème étage - 235 cours Lafayette 69006 LYON



# Formation Usine Logicielle: Devops, Intégration et Déploiement Continus

Plus d'info sur <http://www.dawan.fr> ou **0810.001.917**

Formateur: Matthieu LAMAMRA

# Présentation de la formation

Qui suis-je? « mon parcours » « mes compétences »

Exprimer votre besoin:

- Le contexte **IT**
- Votre environnement de travail
- Vos **objectifs** pour cette formation.

# Objectifs

Comprendre le mouvement DevOps

Comprendre les notions d'intégration et de déploiement continus - CI/CD

Gérer les interconnexions entre solutions d'automatisation

Stabiliser et optimiser l'environnement de développement

Avec ces acquis vous serez en mesure de :

- ✓ Installer et configurer une solution de versioning
- ✓ Installer et configurer un serveur de CI/CD
- ✓ Associer des outils de build, de tests, d'assurance qualité



# Introduction DevOps



Que désigne pour vous **l'Agilité** ?

Que vous évoque le terme **DevOps** ?

Qu'entendez vous par **Intégration Continue (CI / CD)** ?

# Vers l'agilité

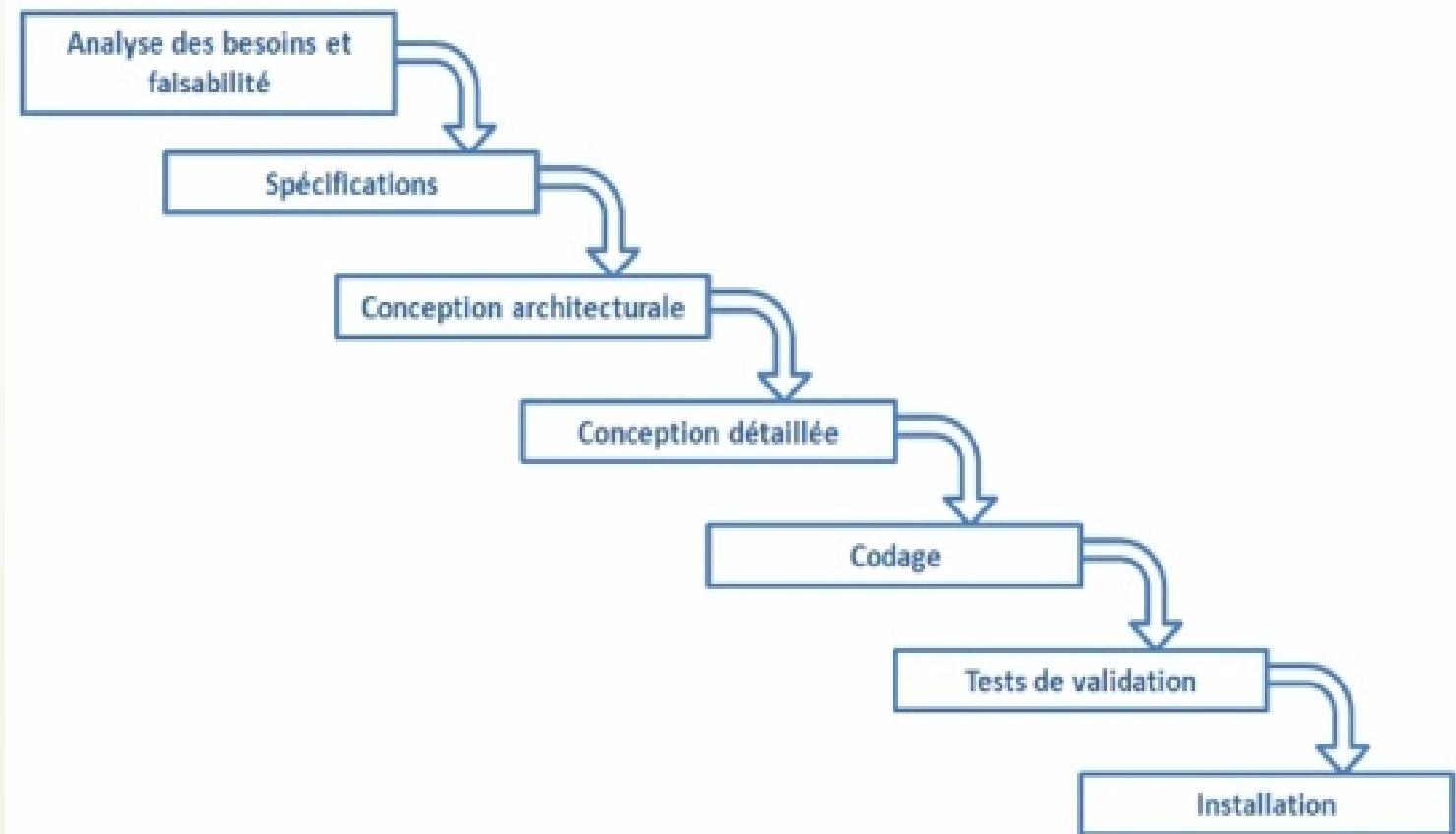
- Cycles prédictifs : Le cycle en cascade

## Failles :

- Activités en silos
- Mauvaise communication
- Documentation pléthorique

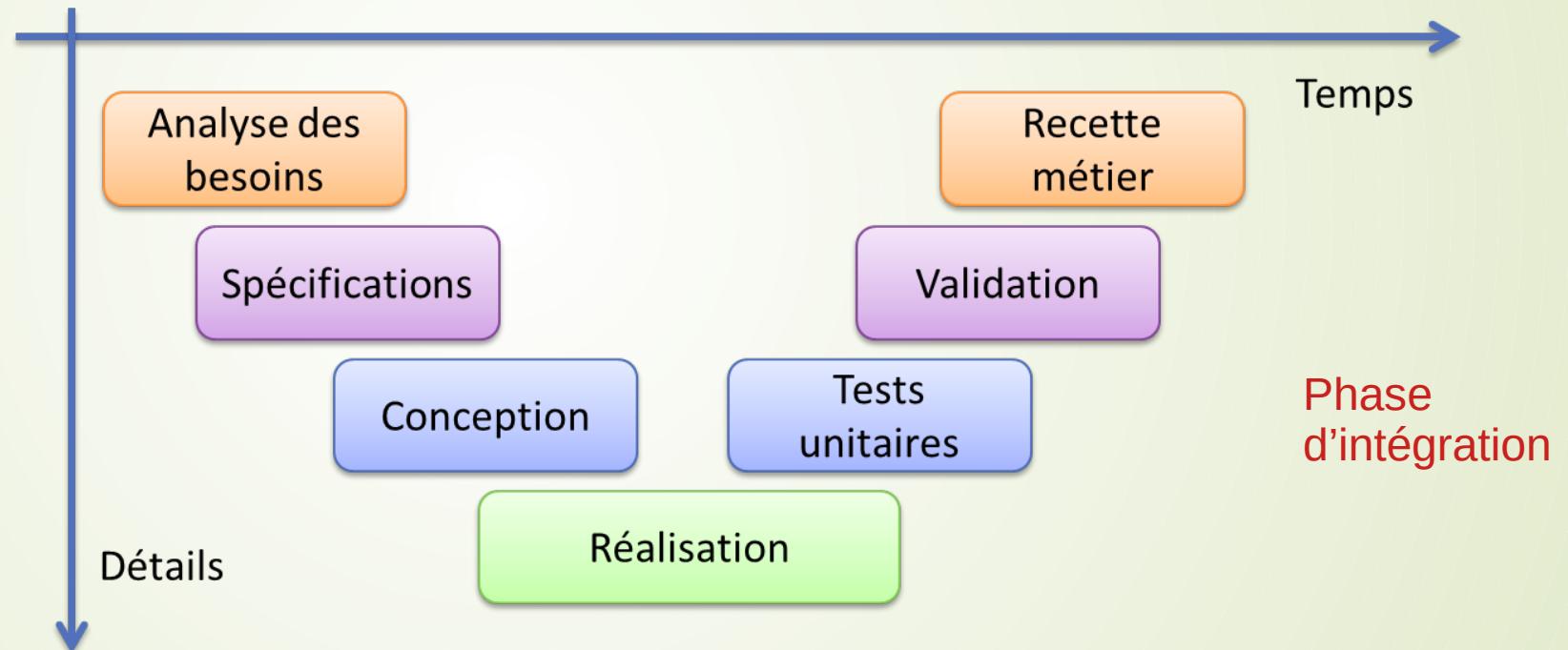
=> Effet tunnel

=> Levée tardive des facteurs de risques



# Vers l'agilité

- Cycles prédictifs : Le cycle en V





# Cycles prédictifs

- Budget et plan, **tout est défini**
- La spécification, en amont, doit pouvoir **tout prévoir** et décrire de façon précise et exhaustive
- **Isolement** des équipes et transmission de l'information **unidirectionnel** :
  - À chaque passage de section les erreurs/imprécisions/surcoûts se multiplient



# Cycles prédictifs

- Tunnel : recette métier tout à la fin, opacité de l'avancement
- Périmètre / délais contractuels : la qualité est la variable d'ajustement

# Agilité : Historique

- Années 30, 40 : réflexion sur le cycle de développement **itératif**
- 1976 : Méthode **EVO** « Evolutionary Value Delivery » => **cycles courts**
- 1977 : Jay Galbraith : « **organisational design** » défini comme « un processus de décision visant à aboutir à une cohérence entre les objectifs originels et structurant de l'organisation (la stratégie) et les modes d'organisation du travail et de coordination des unités et du personnel »
- 1986 : développement de **Scrum** : flexibilité, multidisciplinarité, collaboration
- 1991 : développement de **RAD** « Rapid Application Development » : cycle de développement **itératif, incrémental et adaptatif**

# Agilité : Historique

- 2000 : Réunion d'unification (17 fondateurs) des différentes méthodes dans le but de trouver un socle commun de valeurs et de bonnes pratiques.
- Résultats : écriture du **Manifeste pour le développement logiciel Agile** :  
**<http://agilemanifesto.org/iso/fr manifesto.html>**
- et création de l'**Agile Alliance** (association chargée de la promotion de l'agilité et du soutien aux équipes) :  
**<http://www.agilealliance.org/>**

# Manifeste Agile

- Les 4 Valeurs

Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

- **Les individus et leurs interactions** plus que les processus et les outils
- **Des logiciels opérationnels** plus qu'une documentation exhaustive
- **La collaboration avec les clients** plus que la négociation contractuelle
- **L'adaptation au changement** plus que le suivi d'un plan

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





# Les 12 Principes

#1 : Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.

# Les 12 Principes

- #2 : Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.

# Les 12 Principes

#3 : Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.



# Les 12 Principes

- **#4** : Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
- **#5** : Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
- **#6** : La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.



# Les 12 Principes

#7 : Un logiciel opérationnel est la principale mesure d'avancement.



# Les 12 Principes

- #8 : Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
- #9 : Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.



# Les 12 Principes

#10 : La simplicité - c'est-à-dire l'art de **minimiser la quantité de travail inutile** - est essentielle.

# Les 12 Principes

- #11 : Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
- #12 : À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.



# Attention !

- Appliquer un framework agile ne rend pas votre organisation agile
- Vous pouvez être agile sans utiliser de « méthodes agiles »

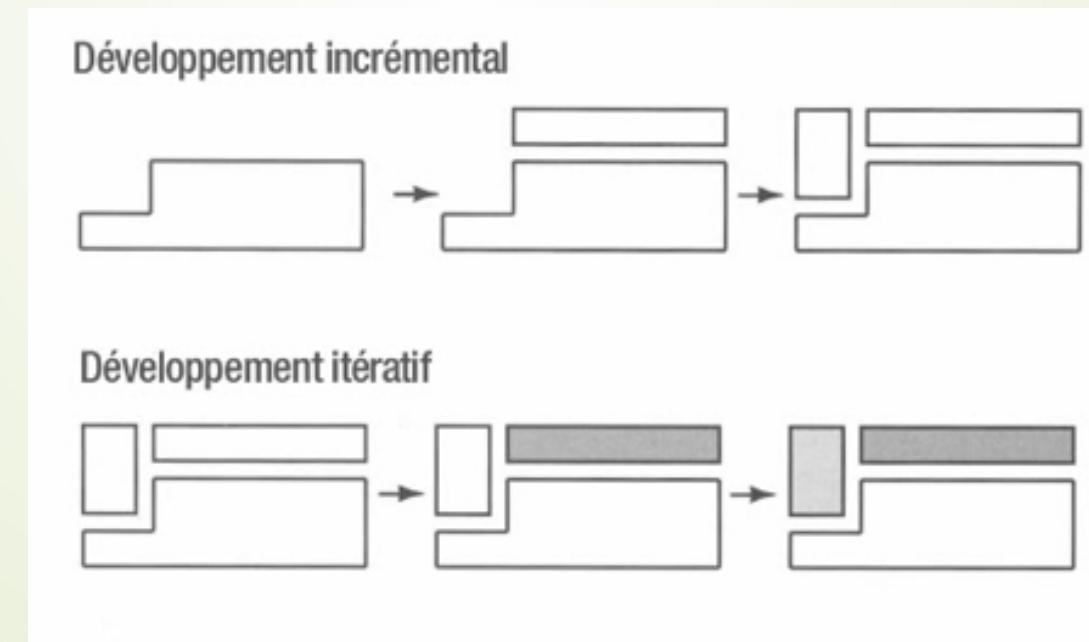


# En résumé

- Livrer en continu
- Des logiciels fonctionnels
- En s'adaptant au changement

# Méthodes Agiles

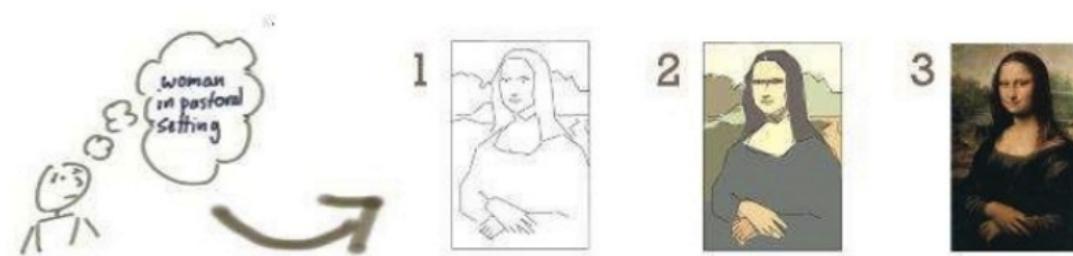
- Une méthode Agile est une **approche itérative, incrémentale et adaptative** qui est menée dans un **esprit collaboratif**, avec le minimum de **formalisme**.
- Elle génère un produit de **haute qualité** tout en prenant en compte l'**évolution des besoins du client**.



# Méthodes Agiles

- Itérative ET Incrémentale...

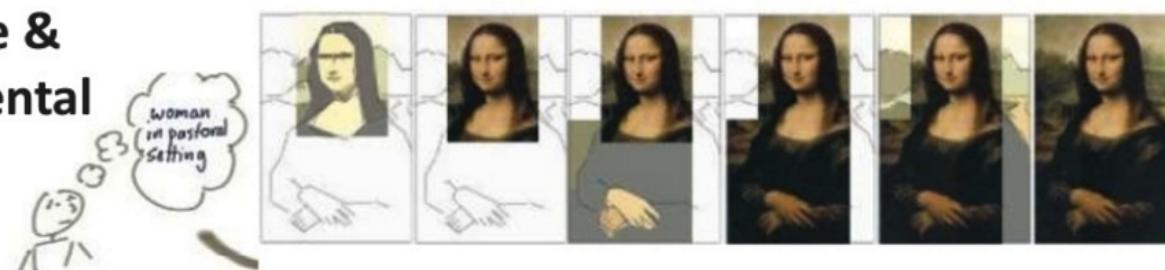
**Iterative**



**Incremental**

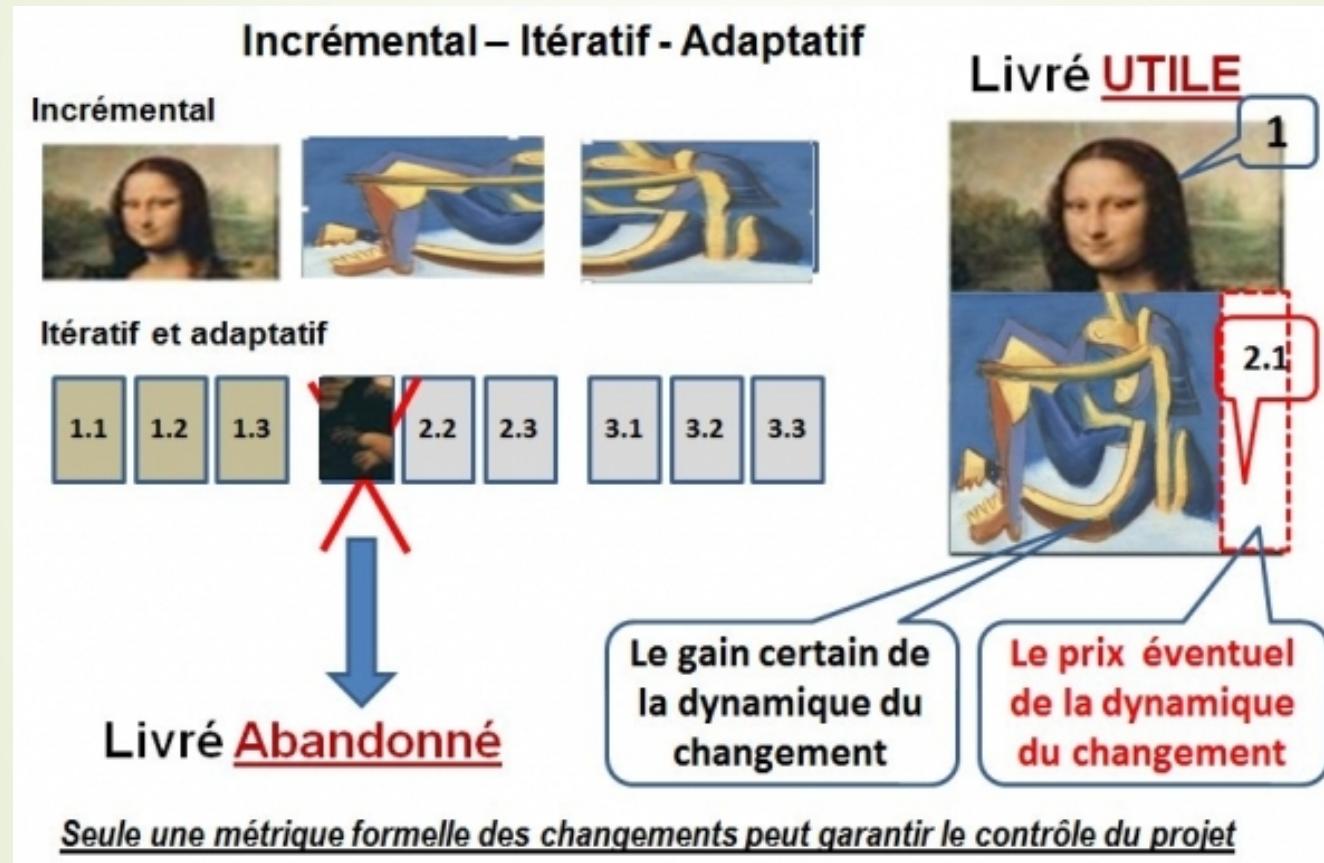


**Iterative &  
Incremental**



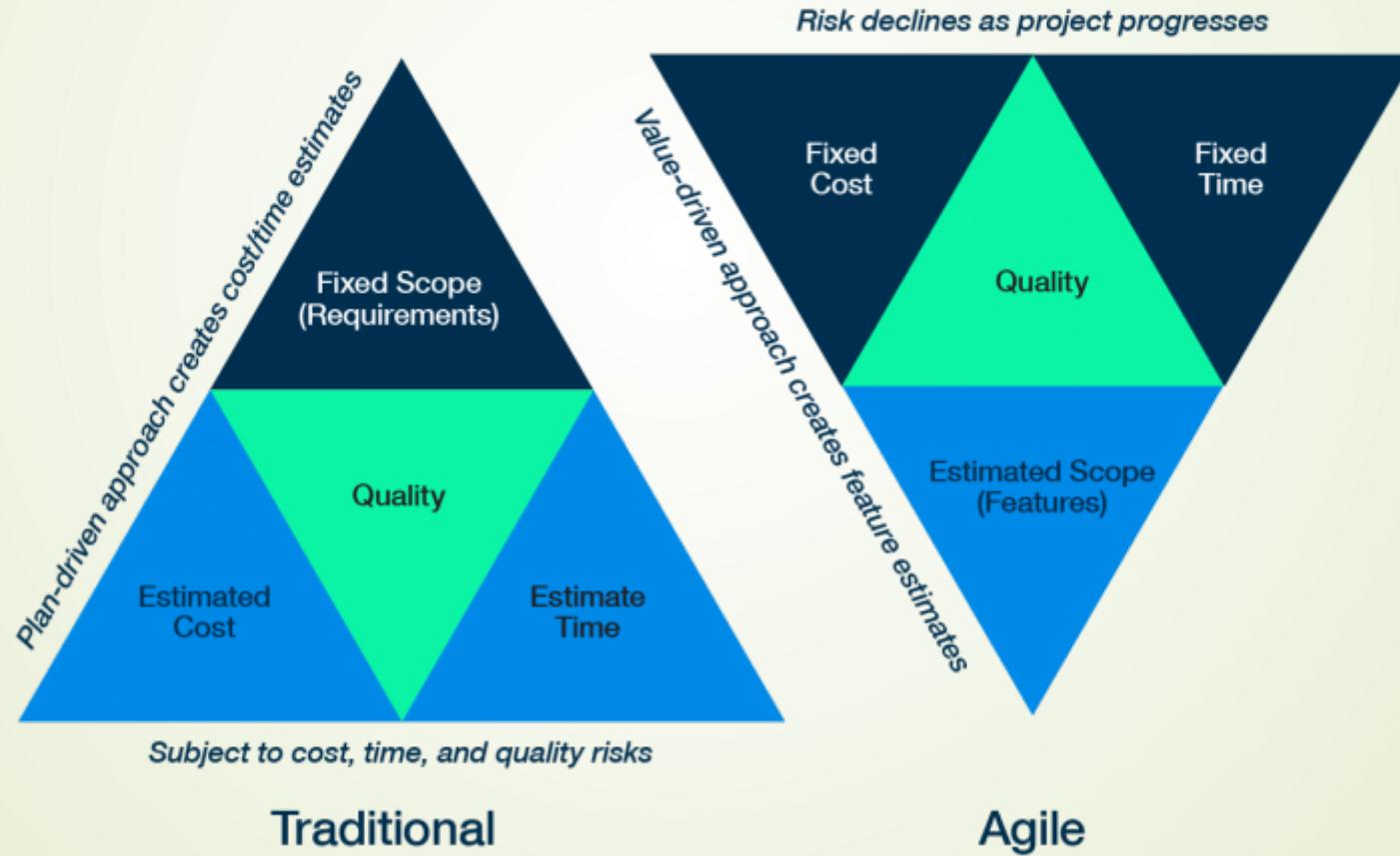
# Méthodes Agiles

- ... ET Adaptive



# Méthodes Agiles : intérêts

## Iron Triangle Paradigm Shift



# Méthodes Agiles : intérêts

- Réduire le temps de mise sur le marché (**Time to market**)
- Améliorer la qualité du produit
- Réduire les activités sans réelle valeur ajoutée  
=> Travailler sur des éléments de plus grande valeur (cf LEAN)
- Avoir une meilleure prévisibilité
- Améliorer les conditions de travail

# Méthodes Agiles : quand ?

## Favorisants :

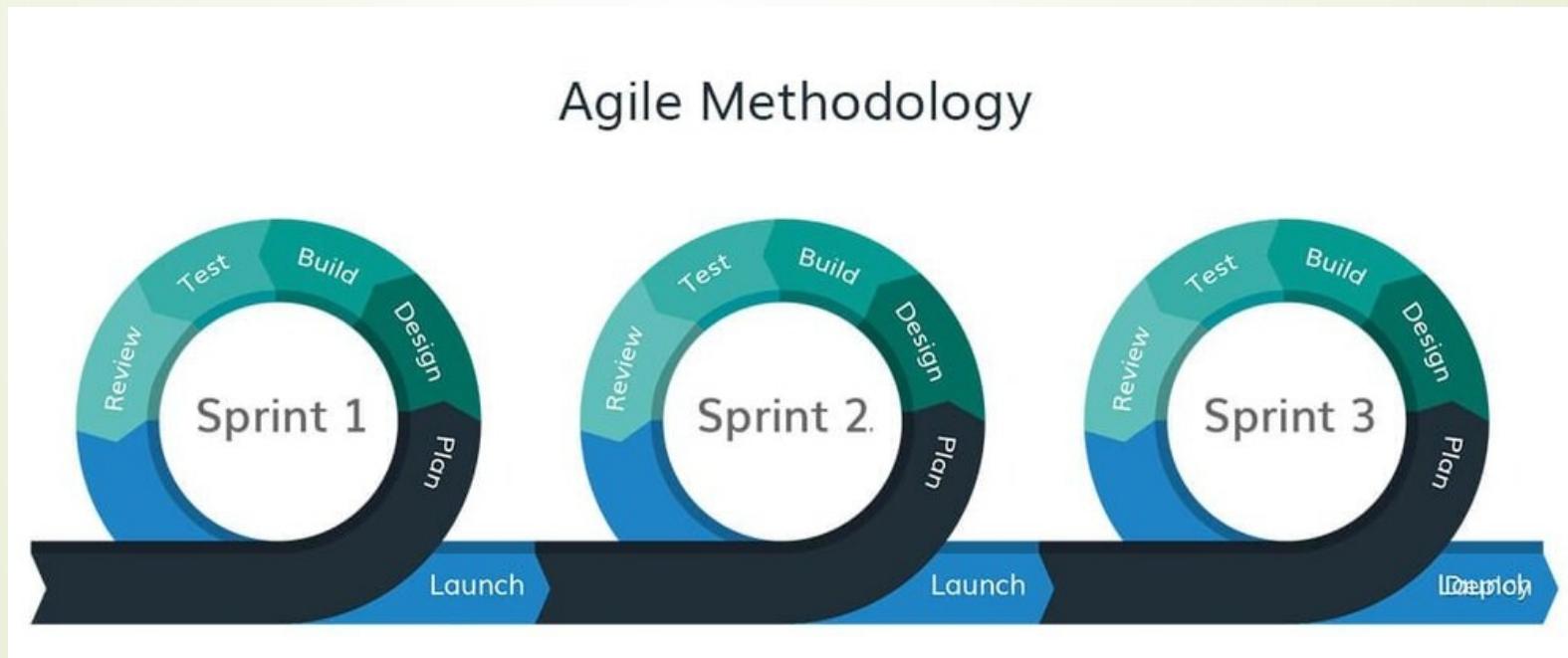
- Besoin rapide de mise à disposition du produit
- Imprévisibilité des besoins du client
- Nécessité de changements fréquents
- Besoin de visibilité du client sur l'avancement des développements
- Présence de l'utilisateur assurant un feedback immédiat

## Défavorisants :

- Indisponibilité du client ou de l'utilisateur
- Dispersion géographique des ressources humaines
- Inertie des acteurs du projet ou refus des changements
- Gouvernance complexe de la DSI

# Le cycle Agile

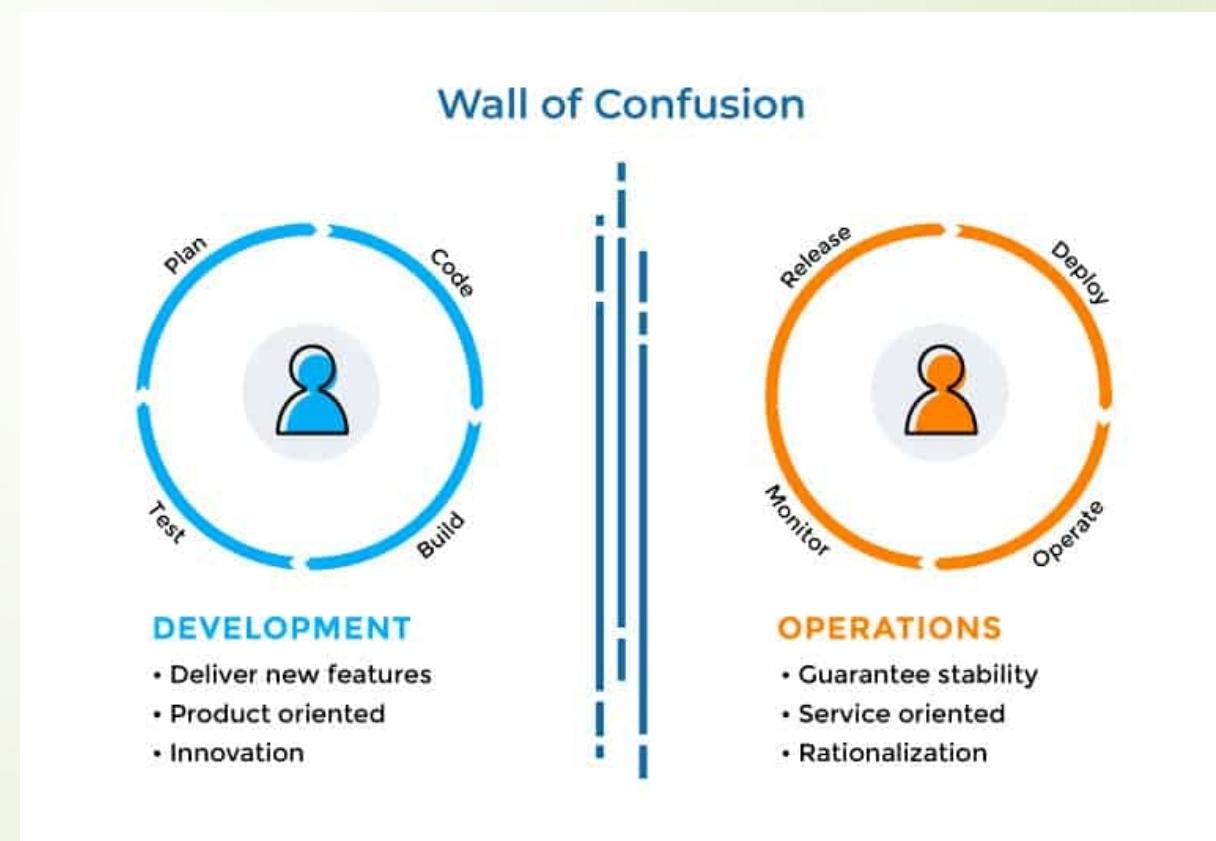
- Sprint : itérations sur un incrément
- Chaque sprint représente un cycle de développement
- Augmente la fréquence de livraison



# Introduction DevOps

- Besoin d'Agilité dans la relation Dev / Ops
  - Stéréotypes Dev    VS    Stéréotypes Ops

C'est pour résoudre ce problème, dans le contexte de la conférence Agile 2008 à Toronto sur l'infrastructure agile, que le mot « **DevOps** » a été employé pour la première fois par l'ingénieur système belge Patrick Debois



# Introduction DevOps

- Atteindre une organisation DevOps
  - Questionner d'abord la raison d'être de l'organisation « Pourquoi ? » avant son objet « Quoi? » et ses moyens « Comment ? » => **Modèle du Cercle d'or**
  - Se concentrer sur les relations humaines ...
    - construire une **culture de collaboration** entre Devs, Ops, Clients, Partenaires...
    - Les actions de chacun sont améliorées par la compréhension des besoins des autres
    - adopter un **vocabulaire et des nomenclatures communs** aux différentes parties
    - appliquer une **gouvernance collective et responsabilisante** au sein de l'équipe DevOps
    - agréger les compétences en formant des **squad DevOps multidisciplinaires**
  - ... grâce à l'automatisation de la production de valeur
    - automatiser les opérations transformant un code en **livrable** => **Intégration continue**
    - automatiser les déploiements => **livraison / déploiement continu**
    - automatiser les opérations sur les serveurs => **gestion de configuration, orchestration**
    - automatiser les remontées d'informations pour faciliter les décisions => **monitoring**

# Introduction DevOps

- Atteindre une organisation DevOps
- Rôles liés à l'**Agilité** et **Scrum** dans le cadre d'un **management horizontal**
  - Le « **Product Owner** », qui possède la vision du produit et qui exprime le besoin, doit pouvoir observer régulièrement l'avancée du projet au rythme de la diffusion de livrables ; il est soit client, soit en interface avec le client
  - L' « **Agile Process Owner** » utilise les principes et pratiques Agile et Scrum pour concevoir, gérer et mesurer les processus individuels
  - « **Le Scrum Master** » (Dev) et L' « **Agile Service Manager** » (Ops) sont les garants du cadre méthodologique de l'équipe. Ils doivent veiller à ce que les processus agiles mis en place soient respectés. Ils ne sont pas nécessairement chef de projet

# Introduction DevOps

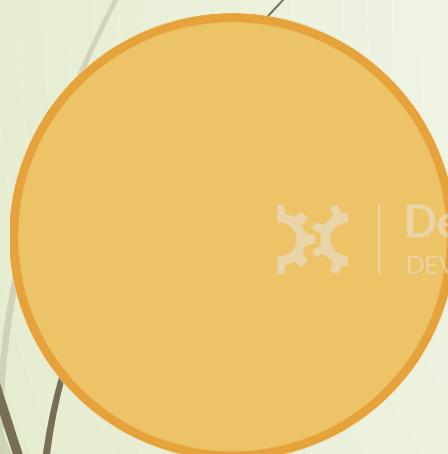
- Atteindre une organisation DevOps
  - Profils liés spécifiquement à DevOps
    - L' « **évangéliste** » : rompu aux pratiques DevOps, forme et conseille les membres de l'équipe
    - Le « **release manager** » : responsable principal des processus DevOps mis en oeuvre
    - L' « **automation architect** » : s'occupe de l'automatisation des processus DevOps
  - Rôles traditionnels dans une équipe DevOps
    - **Développeur** : en charge de l'évolution du produit
    - **Testeur** : vérifie le caractère opérationnel du code
    - **Ingénieur Qualité** : évalue la qualité du code, les performances, l'expérience utilisateur « **UX** »
    - **Ingénieur sécurité** : fournit aux Devs et Ops les bonnes pratiques de sécurités, et teste la sécurité du code et des environnements d'exécution
    - **Opérateur** : Administrateur système et réseau en charge de la stabilité des environnements
  - Ces rôles sont cumulatifs et distribués : ils forment un **squad**, divisés en « **tribus** » , dans les grandes organisations
  - Autres rôles : Client, Partenaires, Hiérarchie des décideurs...

# Introduction DevOps

- Topologies DevOps (idéal)

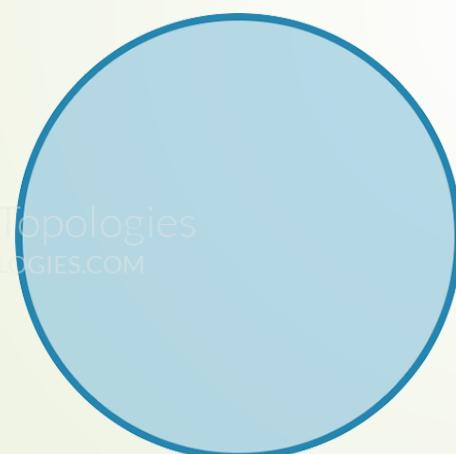
NOK

Silo



● Dev

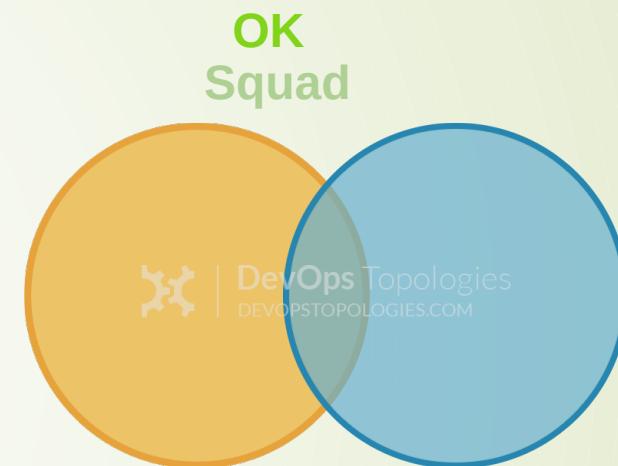
Silo



● Ops



OK  
Squad



● Dev

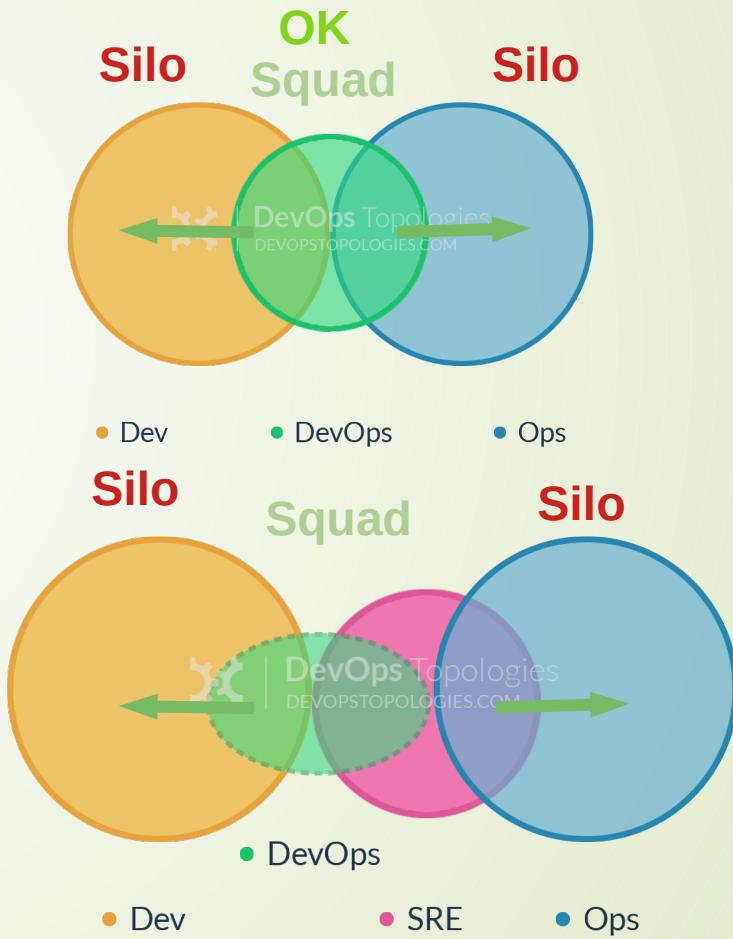
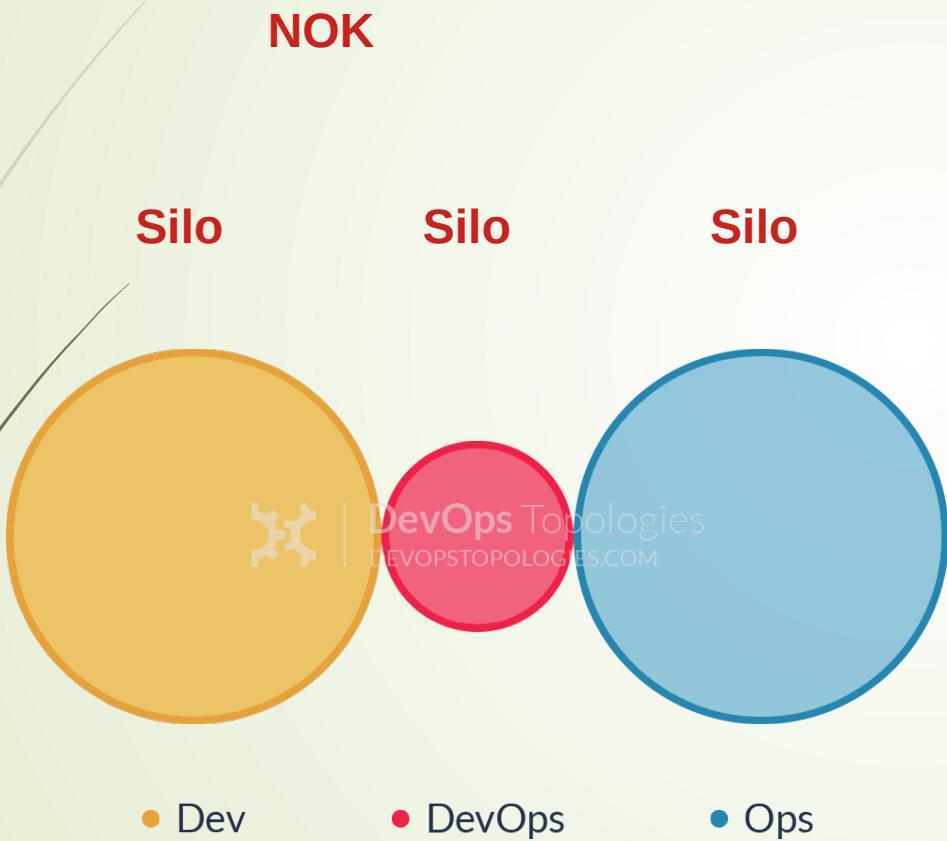
● Ops



● Dev   ● Ops

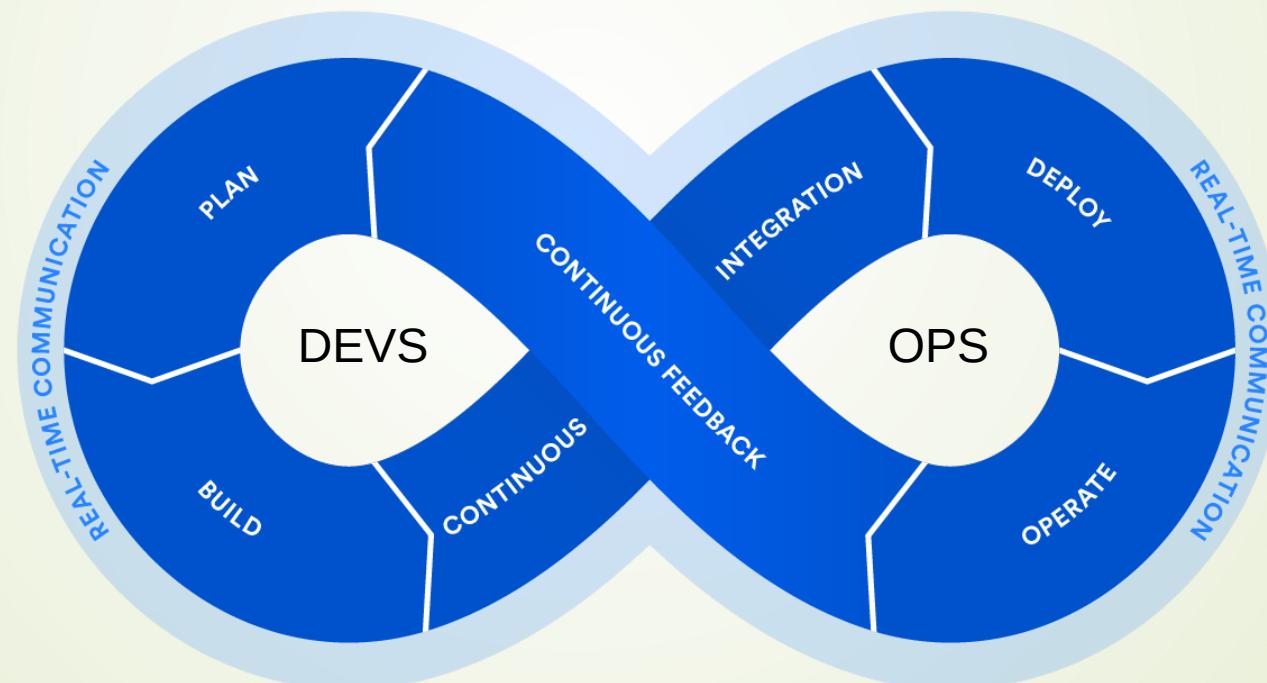
# Introduction DevOps

- Topologies DevOps (Réel)



# Introduction DevOps

- Le cycle DevOps



# Introduction DevOps

- Intégration Continue : Définition

Ensemble de pratiques et de solutions logicielles concourant à éliminer les erreurs - « régressions » consécutives à l'évolution d'un projet.

Exemples :

- Gestion automatisée du build / compilation
- Tests automatisés : unitaires, « End To End / E2E », couverture de tests
- Gestion de la qualité : conventions, lisibilités, performances
- Gestion de l'information projet : Contrôle de version (VCS), monitoring, dataviz

# Introduction DevOps

- Déploiement Continu

Pratiques et solutions logicielles permettant l'automatisation du déploiement d'une solution d'un environnement vers un autre en fonction du résultat d'un processus d'intégration continue

Exemples :

- Gestion de configurations : automatisation des installations / mises à jour
- Orchestration : télé-distribution et allocation automatique des ressources
- Gestion d'un cloud : Virtualisation / Isolation système

# Toolchain DevOps

## CONTINUOUS INTEGRATION



## CONTINUOUS DELIVERY



## CONTINUOUS DEPLOYMENT



# Introduction DevOps

- Panorama des outils de CI / CD

VCS : **Subversion** (SVN)

Centralisé



/ **Git**

Décentralisé



Gestion CI / CD : **Jenkins**

Open source (Java)

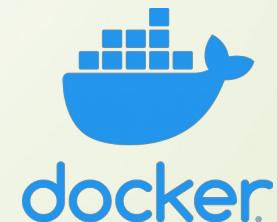


/ **Gitlab CI**

Open source (go /ruby)



Exploitation de conteneurs **docker**



# Introduction DevOps

- Panorama des outils de CI / CD

Build :



- xml

/ **Maven™** - java

Test :



+ déclinaisons / **Selenium**



e2e - java

Qualité :



Déploiement : via **Ansible** Python-YAML





# Bénéfices de DEVOPS

- Pour l'IT

- Fiabilisation du cycle de vie des services (**confiance**)
- Concentration des équipes sur les tâches à valeur ajoutées
- Réduction des **temps de réparation** (incident) et des **temps de mises en œuvres** (fonctionnalités)
- Meilleure Réactivité au changement
- Incontournable pour les projets big data : on ne peut tester qu'en environnement de prod

# Bénéfices de DEVOPS

- Pour l'entreprise

- Meilleur **time to market** par l'augmentation de la livrabilité => **avantage concurenciel**
- Augmentation de la production
- Baisse des coûts de production à moyen terme et de maintenance à court terme
- Meilleur qualité de vie au travail et meilleures collaborations entre services
- Meilleur prévisibilité des roadmap