1. DevOps

1.1. Présentation de devops (origine, concepts, ...)

DevOps

Devops est la concaténation des trois premières lettres du mot anglais development (*développement*) et de l'abréviation usuelle ops du mot anglais operations (*exploitation*), deux fonctions de la gestion des systèmes informatiques qui ont souvent des objectifs contradictoires *(ex : nouvelles fonctionnalités apportant de l'instabilité / fiabilité exigée en exploitation)*

Le **devops** est un mouvement en ingénierie informatique et une pratique technique visant à l'**unification** du développement logiciel (***dev***) et de l'administration des infrastructures informatiques (***ops***), notamment l'administration système. *(source wikipedia)*

Origine et principes de DevOps

Apparu autour de 2007 en Belgique avec Patrick Debois, le mouvement Devops se caractérise principalement par la promotion de l'automation et du suivi (monitoring) de toutes les étapes de la création d'un logiciel, depuis le développement, l'intégration, les tests, la livraison jusqu'au déploiement, l'exploitation et la maintenance des infrastructures.

Conférences "DevOpsDays" (la première en 2009 à Gand / Belgique)

Autre terminologie: "***Agile Infrastructure***"

Les principes Devops soutiennent des cycles de développement plus courts, une augmentation de la fréquence des déploiements et des livraisons continues, pour une meilleure atteinte des objectifs économiques de l'entreprise. *(source wikipedia)*

Avant "DevOps"

Univers "Dev" et "Ops" traditionnellement séparés dans les années 1995-2010 pour les raisons suivantes :
- beaucoup de complexités à gérer → besoin de se spécialiser .
- solutions à bases de serveurs (ex : ServApp JEE) à administrer et dans lesquels on déploie des applications (à développer).
- peu d'automatisation , contextes différents (O.S. , ...)

Objectifs "dev"

. apporter les changements nécessaires au moindre coût et le plus vite possible
souvent au détriment de la qualité lorsque des retards viennent mettre les délais du planning en péril

Objectifs "ops"

. garantir la stabilité du système.
. se concentrer sur contrainte qualité,
. besoin de temps et de moyens
. contrôler sévèrement les changements apportés au système

Vers "DevOps"

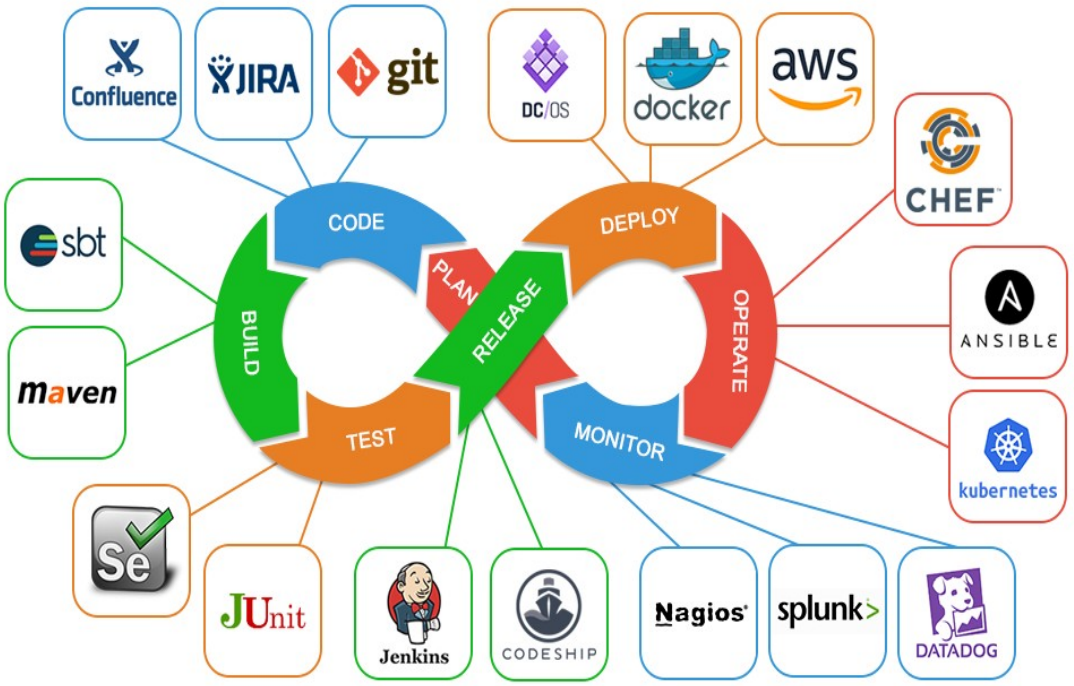
- changement de paradigme ("séparation" → "unification")
- les équipes "dev" et "ops" ne doivent pas se rejetter la faute en cas de problèmes mais doivent collaborer et s'entraîder de façon à ne forme qu'une seule "méga-équipe" .

- automatisations (intégration continue , déploiements, …)
- cycles courts (dev, tests, intégration , mise en prod, …)

- bonnes métriques / supervisions
- applications des méthodes agiles (tests , communications, …)
- alignement avec les technologies "cloud" et "web"

1.2. cycle de vie devops

Cycle de vie "DevOps"



Première phases du cycle "devops"

1. Planification (et modélisation , suivi)

2. Codage/implémentation (avec tests unitaires)

Phases suivantes du cycle "devops"

3. Build (de "dev" ou "release/prod")
(assemblage , pré-released)

maven

… éventuels profils de 'dev' / 'prod'
avec "bases" et ...
plus ou moins simplifié(e)s

4. Tests d'intégration (intégration continue)

Exécution automatique des tests dans le but d'avoir un feedback concernant la qualité du code produit en cours de déploiement

Phases suivantes du cycle "devops"

5. Release (de ou vers "prod")

maven profile de 'prod'
avec vraies "bases" / "serveurs" / ...

Environnement au sein duquel l'identification
des anomalies doit idéalement pouvoir se faire aisément.

6. Deploy (packaging & déploiement)

Dernieres phases du cycle "devops"

7. Operate (fonctionnement en "production")

fonctionnement en cluster ,
...

8. Monitoring (surveillance & métriques)

surveillance/exploitation ,
remontées/interprétations des mesures , gestion des logs,
Redimensionnement éventuel (élasticité cloud, ...)

1.3. Devops , concrètement ...

Concrètement "DevOps" c'est avant tout :

- De l'agilité au niveau de l'infrastructure (et pas que sur le papier en théorie)
- Ça se traduit par des automatisations efficaces (intégration continue et déploiement de conteneurs "docker") de façon à ce que les développeurs puissent assez facilement construire des "briques" prêtes à être déployées sans trop de d'ajustements/paramétrages .
- Et inversement , certains informaticiens (très techniques) peuvent packager des environnements sophistiqués (avec sécurité , …) correspondant aux contraintes réelles de production sous forme d'images "docker" de façon à ce que le développeur puisse facilement y intégrer et tester de nouvelles fonctionnalités.