Les 10 pratiques pour adopter une démarche DevOps efficace

William Gravier
RESPONSABLE D'ACTIVITE DEVOPS – SOCIETE POESI

Qu’est-ce que DevOps ?

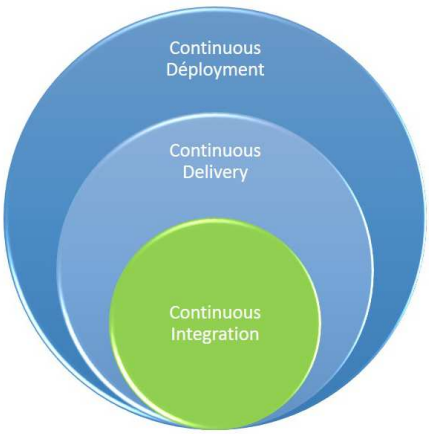
Le nom « **DevOps** » vient de la contraction du mot « **development** » (développement) et « **operations** » (exploitation).

DevOps, c’est une culture et une stratégie opérationnelle qui vise à améliorer la communication entre les études et l’exploitation afin de réduire le temps de mise sur le marché d’un produit.

C’est aussi un ensemble de bonnes pratiques destiné à répondre au besoin croissant d’industrialisation et de normalisation du système d’information.

Les trois processus DevOps

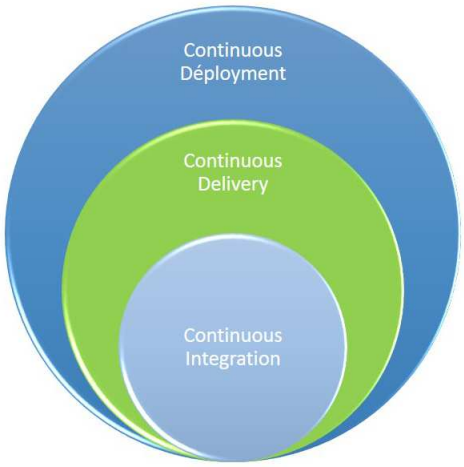
L'intégration continue



L'intégration continue ou Continuous Integration est un processus orienté études consistant à compiler, tester et déployer sur un environnement d'intégration. Le but est de tester aussi souvent et autant que possible les non-régressions du livrable pour détecter les bugs le plus tôt possible. La plupart du

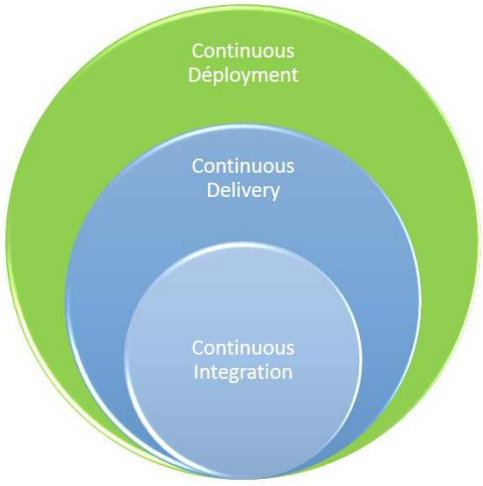
travail est réalisé par des outils de test. Le déploiement sur la plateforme d'intégration devient simple et peut être réalisé par les études sans faire intervenir l'exploitation.

La livraison continue



La livraison continue ou Continuous Delivery est un processus d'intégration et de production. Le but est de compiler, tester et livrer une application à chaque étape de son cycle de vie (recette, pré-production, répétition, production). Cette étape est réalisée après validation des tests effectués en intégration. La phase de test correspond aux tests fonctionnels du livrable. Le passage d'un état à l'autre est entièrement automatisé, c'est pourquoi le livrable doit être constitué de tel sorte qu'il soit déployable en production dès la mise en recette.

Le déploiement continu



Le déploiement continu ou Continuous Deployment est un processus de production. Le but est de compiler, tester et déployer une application en production. Le Continuous Deployment nécéssite que les processus de Continuous Integration et de Continous Delivery aient été réalisés avec succès. Le déploiement est réalisé par un simple « press button ». Après le déploiement, il doit être possible de mesurer les éventuels impacts à l'aide d'outils de

mesure de la performance et d’outils de supervision. En cas de problème, un processus automatisé de retour arrière peut être exécuté.

L’amélioration continue

Ce processus ne fait pas partie des processus DevOps mais il semble essentiel pour POESI.

Le processus d’amélioration continue consiste à améliorer continuellement les trois processus précédents. Ce processus est représenté comme un ensemble d’indicateurs capable de mesurer le « Time to market » et la qualité des processus précédents dans le but de faire ressortir des axes d’amélioration.

L’AGILITE des études et l’ITILISISATION de la production

Au fil des années, les méthodes de travail des études et de l’exploitation se sont éloignées et leurs cultures ont divergées.

- **Ils se sont d’abord éloignés physiquement.**
  Les équipes ont été cloisonnées, externalisées ou délocalisées.
- **Puis éloignés culturellement !**
  Les développeurs doivent répondre à un besoin métier, faire preuve de réactivité et déployer le plus souvent possible tandis que l’exploitation doit s’assurer du maintien en condition opérationnelle du système d’information.
- **Enfin éloigné méthodologiquement !**
  Les développeurs travaillent en mode Agile tandis que la production travaille en ITIL. Les deux méthodologies ont le même objectif mais divergent aux niveaux des pratiques.

En quoi DevOps est complémentaires des pratiques Agile ?

L'un des principes fondamentaux du processus Agile est de livrer plus souvent en plus petite quantité. Il est courant en Agile d'obtenir un potentiel livrable à la fin de chaque sprint (habituellement tous les 15 jours).

Le nombre élevé de déploiements a pour conséquence de les entasser sur les environnements d'exploitation. L'exploitation privilégie la sécurité, leur processus ne leur permettant pas de suivre la cadence des développements en mode Agile.

DevOps est particulièrement complémentaire au développement en mode Agile car il permet de déployer un livrable automatiquement depuis la phase d'intégration jusqu'à la production. Un livrable est packagé pour la production dès l'étape d'intégration.

DevOps est donc une extension d'Agile dans le déploiement d'application.

En quoi DevOps est complémentaire des pratiques ITIL ?

ITIL (IT Infrastructure Library) est devenu en quelques années une référence au niveau mondial autour des bonnes pratiques dédiées à l'industrie IT. C’est un ensemble de bonnes pratiques de planification, de documentation, de processus et de contractualisation qui semble être du côté opposé du spectre des DevOps et Agile.

ITIL a cependant sa place. Dans les grandes organisations, les outils mis en place pour gérer les processus sont devenus très lourds. Une simple tâche comme redémarrer un serveur Jboss est devenue complexe et nécessite des heures de procédures et d’étapes de validation.

Il est cependant tout à fait possible d’automatiser un certain nombre de tâches, comme par exemple la saisie automatique des données dans la gestion de configuration ou encore la remontée automatique d’incidents.

C’est ce que propose la démarche DevOps. Il ne s’agit pas de provoquer le chaos ni de perturber les services.

Les 10 pratiques pour adopter une démarche DevOps efficace

Pratique 1: Travailler en mode collaboratif

Une des philosophies fondamentale de DevOps est de faire travailler en étroite collaboration les études et l'exploitation.

Cette pratique est déjà utilisée dans la communauté Agile lors des « onsite customer ». Elle consiste à travailler en collaboration avec l'ensemble des acteurs responsables d'une application pour résoudre un problème.

Pour y arriver, l'exploitation et les études se réunissent régulièrement pour résoudre les problèmes, planifier les mises en production, communiquer sur les nouvelles fonctionnalités et vérifier la compatibilité de l'application avec l'architecture en place.

L'utilisation d'un outil commun permet de favoriser la communication entre les études et l'exploitation afin de gérer les déploiements de l'intégration à la production. Cet outil peut se présenter sous la forme d'une application web conviviale qui permettra de piloter les déploiements depuis la phase d'intégration jusqu'à la production.

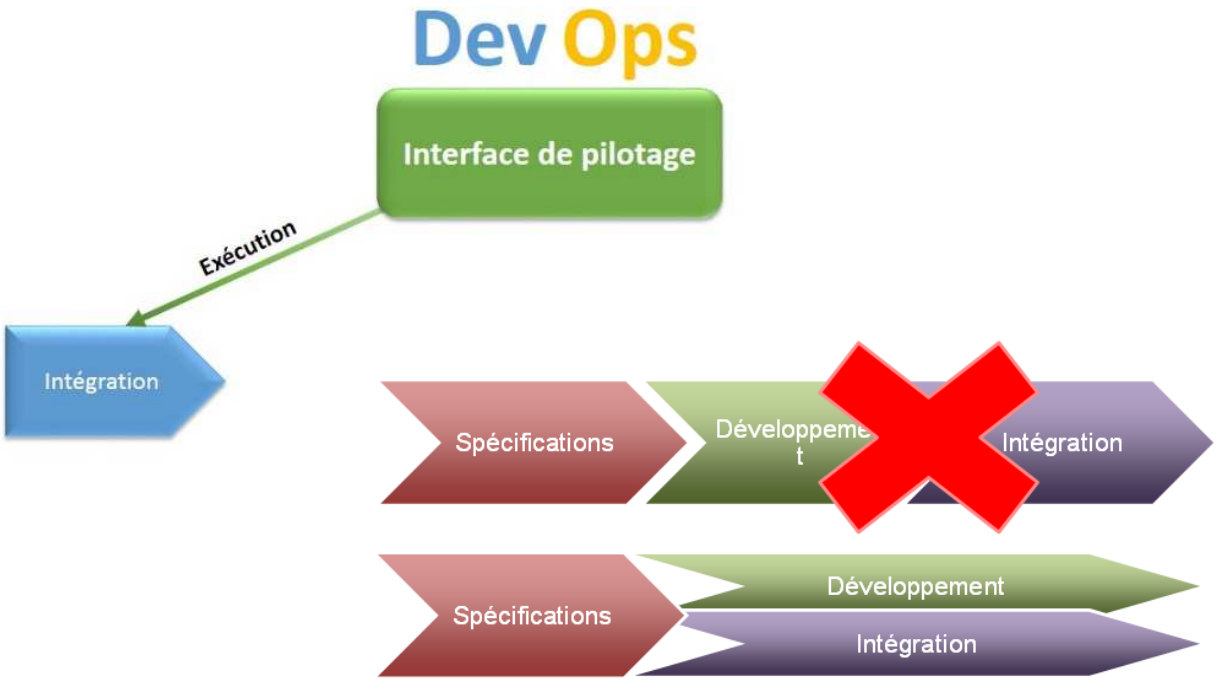
Pratique 2: L’intégration continue

La phase d'intégration est souvent source d'instabilité et de stress. Une étude a montré que 5% des bugs représentent 95% du temps de debug. Ces derniers sont le plus souvent rencontrés lors de la phase d'intégration. Hors, dans un environnement classique, les spécifications, le développement et l’intégration ont lieu séquentiellement, la correction de bug est donc réalisée à la fin du processus. Il faut revenir au développement pour les corriger.

Le but de l’intégration continue est d'effectuer en parallèle les étapes de développement et d'intégration du projet afin d'éliminer au fur et à mesure les bugs avant la phase finale du projet.

L'existence et la validation progressive des tests d'intégration responsabilisent les développeurs sur la bonne interaction de leurs modules de développement.

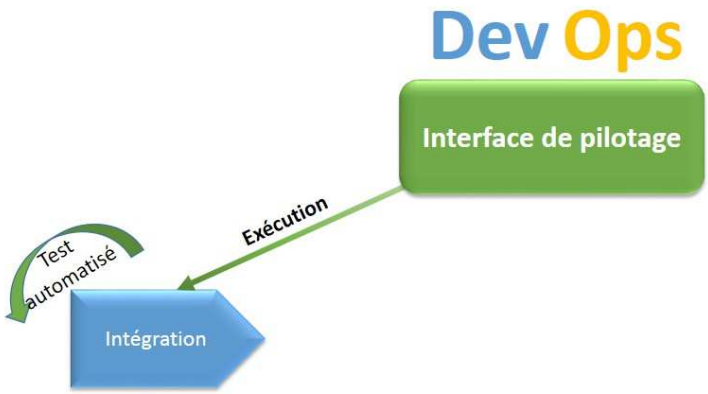
Aujourd’hui, il existe de nombreux produits Editeur ou Open Source pour mettre en place une plateforme d’intégration continue.



Pratique 3: L’automatisation des tests

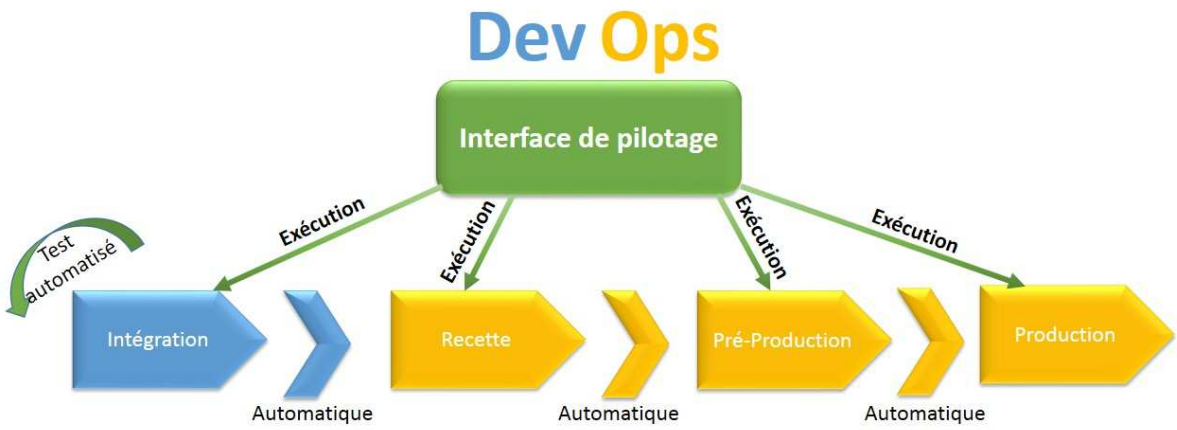
Les tests de non régression doivent être automatisés dans une plateforme d’intégration continue. C’est déjà une pratique courante adoptée par les équipes de développement Java.

Les équipes études jouent parfois plusieurs fois par jour leurs tests automatisés, ils corrigent ainsi les problèmes tout de suite et bénéficient d’un niveau élevé de qualité, ce qui ne peut que satisfaire l’exploitation.



Pratique 4: La livraison continue (Continuous Delivery)

La livraison continue étend la pratique de l'intégration continue. Avec la livraison continue, lorsque votre intégration est réussie dans un bac à sable, vos modifications sont automatiquement promues pour la recette, et le déploiement est automatiquement exécuté. Cette promotion peut être automatique jusqu'au moment où les modifications doivent être vérifiées par une personne tiers, généralement au point de transition entre le développement et les opérations.



La livraison continue permet aux équipes de développement de réduire le temps entre l’intégration et la production. Elle permet à l'entreprise d'être plus réactive. Le fait que les déploiements soient lancés automatiquement augmente très nettement la durée de déploiement et la sécurité.

Pratique 5: Gestion de la configuration automatisée

La gestion de la configuration permet :

- De stocker et de tracer les différentes versions ou révisions de toute information destinée à être utilisée par un système (matériel, logiciel, document, donnée unitaire, etc.).
- Pour déployer des configurations à travers un parc informatique sous formes de fichiers et données.

La mise en place de configurations automatisée répond à un besoin de simplification du parc, et d'industrialisation des déploiements et des configurations. L'uniformisation des configurations des postes de travail et des serveurs permet une meilleure gestion quotidienne et simplifie grandement l'administration du parc. Cela peut être utile pour valider facilement les prérequis d'une application ou encore faciliter le remplacement du matériel.

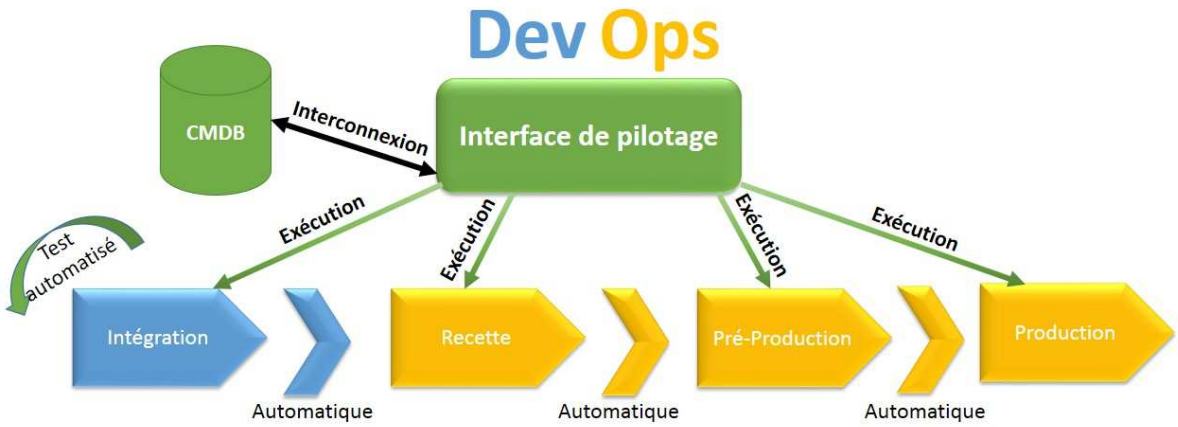
Les développeurs n'ont pas l'habitude de tenir compte de la gestion de la configuration dans leurs solutions. Ils considèrent également que les problèmes de configuration sont des problèmes de production. DevOps considère que la gestion de configuration doit être prise en compte dès l'intégration. Pour cela les études doivent avoir une vision globale de leur produit et de leurs dépendances.

Les données de gestion de la configuration sont stockées dans une CMDB. Nous constatons très souvent que cette base de données ne possède pas de données suffisamment intègres pour interconnecter les plateformes d'intégration continue et de livraison continue afin de récupérer la gestion de configuration automatiquement.

La saisie manuelle des données augmente le risque d'erreur et la forte volumétrie complique la gestion. Cela a pour effet de réduire la pertinence des données saisies dans la base.

On s’aperçoit lorsqu’on utilise des outils de scan d’intégrité que la CMDB (Configuration Management DataBase) dépasse rarement les 50% d’intégrité. Hors pour totalement automatiser le cycle de vie de mise en production, il faut disposer d’une CMDB proche des 100% d’intégrité.

Pour cela, nous préconisons d’automatiser autant que possible la mise à jour de la CMDB avec des outils d’inventaires, des outils de contrôle de cohérence et du monitoring.

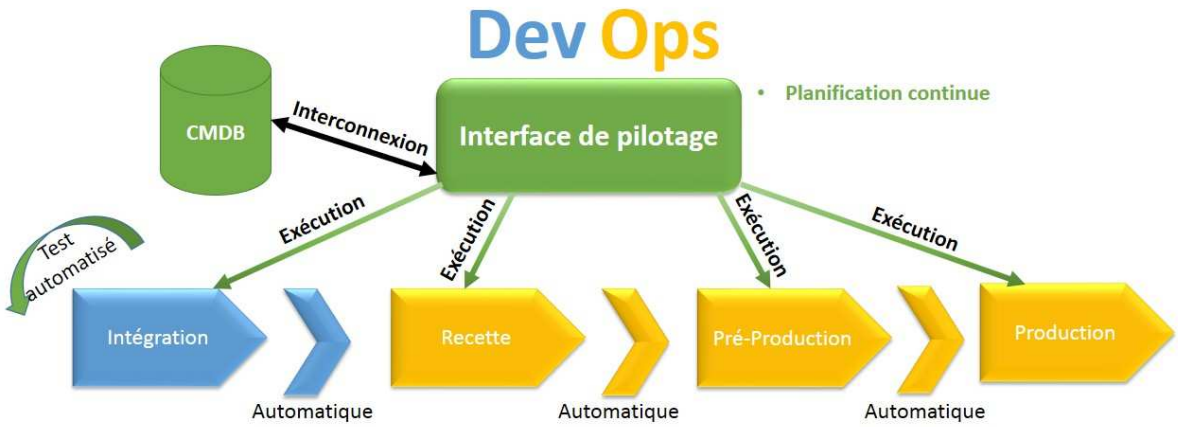


Pratique 6: La planification continue

Du point de vue des équipes de développement, la planification du déploiement a toujours exigé l'interaction avec l'exploitation, dans certains cas la planification est réalisée par des chefs de projet d'exploitation.

Dans la démarche DevOps, les équipes de développement font une planification continue tout au long du niveau d'avancement d'un projet en collaboration avec l'exploitation. L'adoption d'une approche transversale dans la démarche projet afin d'impliquer les études et l'exploitation est alors primordiale.

La planification continue est gérée dans l'interface de pilotage.



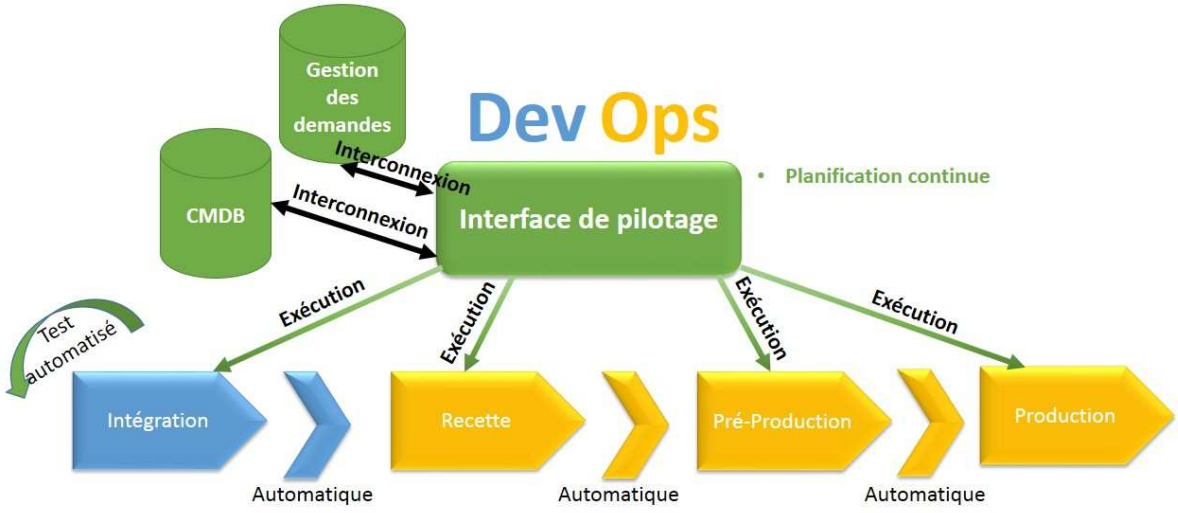
Pratique 7: La gestion des demandes et des RFC

Dans une organisation ITIL, toute modification du système d'information doit passer par une demande ou une RFC (Request For Change).

A chaque livraison, les études réalisent une demande de changement validée ensuite au CAB (Change Advisory Board) par les membres de l'exploitation.

C'est seulement lorsque la RFC est validée que la livraison peut être exécutée sur les différents environnements d'exploitation.

Dans la démarche DevOps, cette demande est suivie et interagie avec la livraison. Elle peut être saisie automatiquement et validée au fur et à mesure que le livrable avance dans le cycle de livraison.



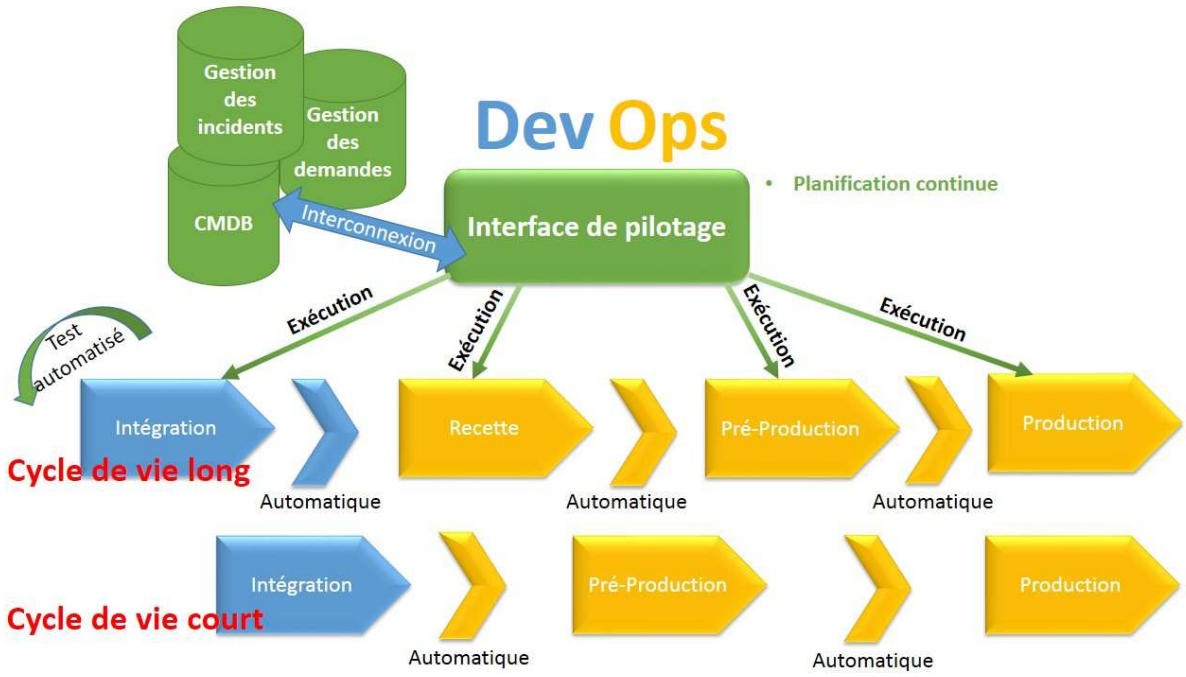
Pratique 8: Le cycle de vie d’une application

La démarche DevOps recommande de définir plusieurs cycles de vie d’une application.

Un cycle de vie représente toutes les étapes de validation d’une application avant sa mise en production.

On déterminera le passage par tel ou tel cycle de vie en fonction de release (mineur ou majeur), en fonction d’évènements (besoin de corriger un incident critique), etc…

Afin d’éviter certains abus, il est tout à fait possible de contrôler ou d’orienter vers un cycle de vie la livraison d’une application en fonction de règles déterminées dans un comité regroupant les études et l’exploitation.



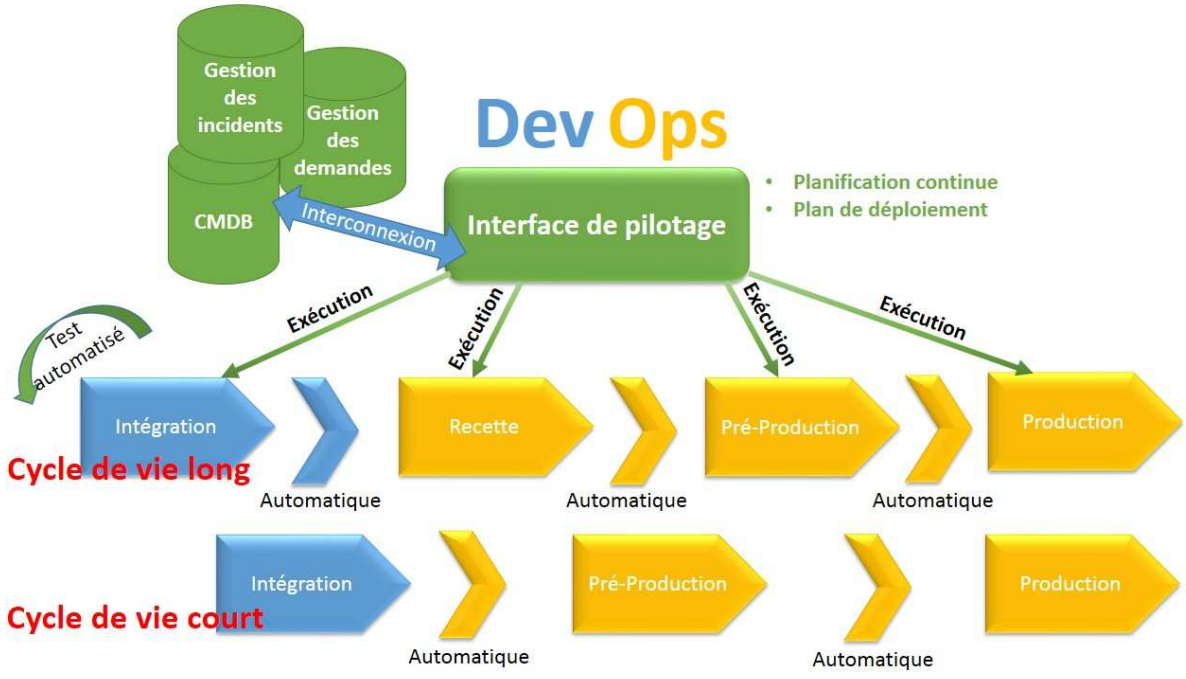
Pratique 9: La gestion des domaines applicatifs

Une application n'est pas seulement du code Java à déployer en production. Une application, c'est aussi un modèle de données, des batchs, des interconnexions avec d'autres applications, des planifications de scripts… Le développement de ces scripts peut-être réalisé par les études mais aussi par l'exploitation (pour ce qui concerne des scripts d'ordonnanceur par exemple).

Une livraison est donc un ensemble de domaine applicatif à orchestrer de manière logique selon des étapes de mise à jour.

Dans la démarche DevOps, les études et l'exploitation sont en contact permanent afin que chacun ait une vision globale de l'application à livrer.

L'interface de pilotage peut permettre aux études et à l'exploitation de construire un plan de déploiement et de l'exécuter automatiquement.

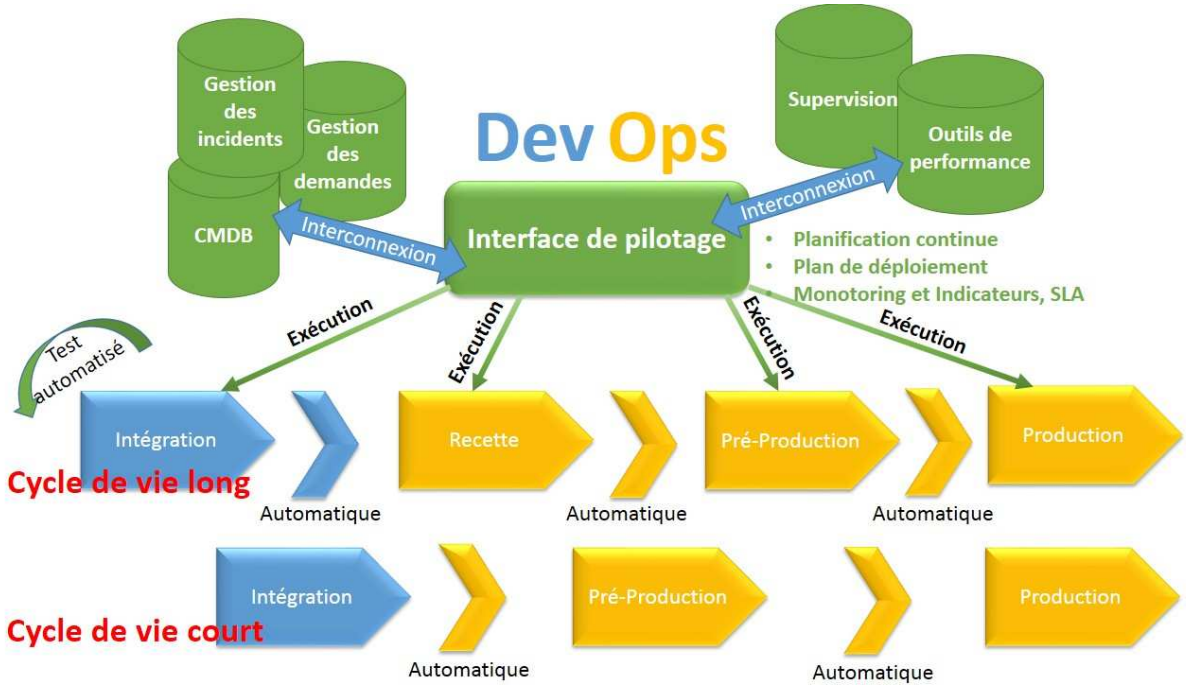


Pratique 10: Tableaux de bord automatisés

Les tableaux de bord sont essentiels pour mesurer l'efficacité des processus mis en place et pour définir des axes d'amélioration.

Du côté de l'intégration continue, il est possible de mesurer la qualité du code, la durée d'exécution, le nombre de commit journalier…
Du point de vue de la livraison continue, il est possible de mesurer la durée d'un cycle de déploiement, le taux d'échec, le nombre de retour arrière, le nombre de déploiement par domaine applicatif, etc…

Le plus important est de disposer d'un outil proposant des indicateurs personnalisables afin de répondre convenablement aux besoins de chacun dans le but d'améliorer la qualité de service.



Quels bénéfices pouvez-vous tirer de DevOps ?

- **Un processus de déploiement répétable et sécurisé.** Le processus de déploiement est unique et exécuté automatiquement, le risque d'erreur est donc très limité.
- **Une forte diminution de la durée de déploiement.** Le processus est normé et simplifié, la complexité est reportée sur les automates.
- **Une baisse des coûts de production.**
  Le nombre d'heures consacrées en astreinte est réduit, la fluidité de déploiement est améliorée.
- **Une réduction du nombre d'erreurs**

  Grâce à l'automatisation, on est assuré que chaque version contient la bonne configuration, les bonnes données, les bonnes librairies et qu'elle sera installée de la même manière à chaque fois.
- **Une meilleure maîtrise de vos déploiements**

  L'exploitant peut déployer une multitude de technologies sans pour autant avoir la connaissance des outils.
  Le déploiement devient un jeu d'enfant, un simple « press button » sur une interface web et le plan de déploiement s'exécute automatiquement. L'expertise technique intervient en support N2 en cas d'erreur.
- **Une amélioration de la qualité de service**

  Le temps gagné sur le déploiement est consacré à l'amélioration de la qualité (maintenance, supervision et performance)
- **Une meilleure entente entre les équipes**

  Réduction du stress, compréhension des problématiques de chacun deviennent le maître mot de votre organisation.

Présentation d'une solution DevOps

On peut imaginer une solution composée de trois outils qui permettent de suivre un déploiement applicatif tout au long de son cycle de vie.

Elle permet :

- D'automatiser le processus de **« Continuous Delivery »**
- De rendre **répétable** tout déploiement d'application et quel que soit la technologie
- De **faciliter la collaboration entre les développeurs et les opérationnels**
- **D'établir des standards, des normes et des protocoles** pour assurer la gouvernance et maitriser les coûts
- **D'afficher une série d'indicateurs et de rapports personnalisables** en fonction des besoins de chacun.

Présentation des outils

L'automate :

L'automate permet de construire un **processus technique générique de déploiement par technologie ou par application**. C'est lui qui va exécuter les lignes de commande sur les serveurs.

Le coffre-fort de release :

Le coffre-fort de release **garantit que le package qui a été déployé en recette est le même que celui qui va être déployé en pré-production et en production**. C'est un gestionnaire de source, il a pour rôle de versionner et de déposer les packages sur les serveurs distants.

L'interface de pilotage :

L'interface de pilotage représente le cœur de la solution de « **Continuous Delivery** » puisqu'elle est le **pool unique de communication entre les « Dev » et les « Ops »**. Dans cette

interface, les Devs réalisent une demande déploiement et conçoivent un plan de déploiement qui sera ensuite validé, complété et exécuté par les Ops. Le plan de déploiement représente l'ensemble des actions à réaliser pour vérifier, tester et déployer une application sur les différents environnements.

L'interface pilote le coffre-fort de release et l'automate par des appels de webservices. L'utilisateur final n'a qu'un seul outil de déploiement, ce qui permet de s'exonérer de tout coût de formation sur des outils spécifiques.

Les interconnexions possibles :

- L'interface de pilotage est interconnectée avec la CMDB et les RFC **pour une meilleure cohérence des données avec le SI**. Elle permet également de faciliter la saisie de la demande de déploiement des Devs.
- L'automate est **interconnecté avec les outils de supervision** afin de les désactiver au moment du déploiement et de les réactiver une fois le déploiement terminé. Il est également interconnecté avec des outils de performances afin de réaliser les premières mesures et tester l'accessibilité de l'application.

Le plan de déploiement :

Chaque application, chaque technologie nécessitent des phases de backup, de test, de préparation, de déploiement et de retour arrière tout au long de leur cycle de vie. Ce plan peut également varier d'une application à l'autre, d'une technologie à l'autre.

Prenons un exemple :

Je souhaite déployer l'application « Pets » en production. Pour cela, je dois suivre le cycle de mise en production normal, c'est-à-dire « Recette », « Mise en production », « Production ». La

release est composée d'un war Jboss, d'un script SQL et d'un batch DataStage. Les processus de déploiement seraient :

1. Le développeur réalise une demande de déploiement et planifie ces dates de mise en recette, pré-production, production
2. Il dépose le package dans un répertoire partagé qui est automatiquement inséré dans le coffre-fort de release
3. Il réalise le plan de déploiement suivant :
    - 1 – Sauvegarde l'application Jboss et vérification du package
    - 2 – Sauvegarde de la base de données et vérification du package
    - 3- Sauvegarde partiel du projet DataStage et vérification du package
    - 4 – Stop des intances Jboss
    - 5 – Exécution du script SQL
    - 6 – Déploiement du War
    - 7 – Démarrage des instances Jboss
    - 8 – Déploiement du projet Datastage
4. Les intégrateurs valident le plan
5. Les intégrateurs exécutent par un simple clic le déploiement en recette, puis les études et l'intégration valident le déploiement et la release
6. La même chose pour la pré-production
7. La même chose pour la production

Le retour arrière :

En cas d'anomalie, le retour arrière est exécuté par un simple clic dans l'interface de pilotage.

Le plan de retour arrière est défini automatiquement par l'outil, dans ce cas il serait :

1. Stop des intances Jboss
2. Exécution du script SQL de retour arrière
3. Déploiement du War de backup
4. Démarrage des instances Jboss

5. Déploiement du projet Datastage de backup

Une action = un processus technique

Derrière chaque action de ce plan se trouve un processus complet de déploiement dans l'automate.

Par exemple, l'action « Déploiement du projet DataStage » est en réalité un ensemble d'actions techniques habituellement réalisées manuellement par les opérationnels comme :

- Vérification de l'espace disque, disponibilité des instances
- Transfert du package
- Import du projet
- Compilation des batchs

Les avantages d'une telle architecture

La complexité de déploiement est supporté pas l'outil, un déploiement applicatif complexe peut être exécuté par un simple clic :

- Il y a un seul outil de déploiement commun aux développeurs et opérationnels quelle que soit la technologie
- La sécurité est améliorée, l'outil n'oubliera jamais de réaliser des actions, de tester le contenu du package ou de vérifier le socle
- Le plan de retour arrière est systématique
- L'action de déploiement devient simple, l'opérationnel peut se concentrer sur des tâches à plus hautes valeurs ajoutées comme la performance, la supervision …
- Il y a nullement besoin d'être expert pour déployer une application. Une seule personne peut déployer une application sur plusieurs technologies. L'expertise technique intervient uniquement en support niveau 2 en cas d'éventuel problème

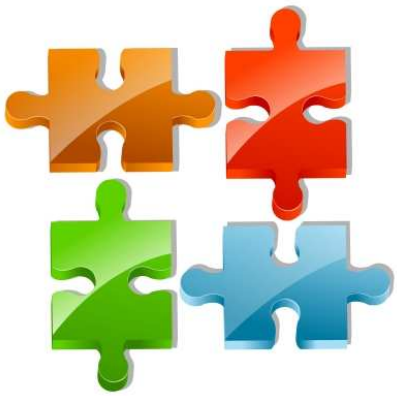
L’offre POESI

Conseil & Audit

Notre métier de consultant consiste à comprendre les problématiques de nos clients et les aider à relever leurs défis au travers de nos expériences.

La démarche conseil de POESI consiste à dresser un **bilan de maturité** des différentes pratiques DevOps afin de définir les **axes d’amélioration** apportant **le plus de valeurs**.

Intégration des outils d'automatisation



Un projet d’intégration d’automatisation des déploiements applicatifs demande de nombreuses compétences et savoir-faire. Il ne s’agit pas de simplement intégrer un produit mais bien de concevoir **une solution qui correspond entièrement à votre besoin**.

Dès le début du projet nous **constituons une équipe composée de membres des études et de la production** afin de définir un processus de livraison commun et qui répond aux exigences de chacun.

Amélioration continue

Parce que l’entreprise et les technologies évoluent, la réussite d’un projet d’intégration n’est pas une fin en soi. Au contraire, le projet n’est que la toute première étape sur le chemin permettant de faire entrer la DSI dans une démarche

Continuous Delivery pérenne. La démarche POESI consiste à **mesurer, monitorer, interviewer pour déterminer des axes d'amélioration**.

Pour nous contacter

Votre contact DevOps :

William GRAVIER

wgravier@poesi.fr

Tél: 06.61.39.07.12

5, face Quai Marcel Dassault 92150 Suresnes

Le détail de nos offres DevOps sur http://poesi.fr