DevOps

Chapitre 1 - Découverte de la méthodologie Devops

Concepteur et développeur d'applications

Objectifs pédagogiques

- ➢ Garantir la stabilité du système
- ➢ Être capable de maintenir une application et de la faire évoluer sans interruption des services
- ➢ Préparer et exécuter le déploiement d'une application
- ➢ Connaissance du processus d'intégration continue
- ➢ Connaissance des règles de mise en production

Découverte de la méthodologie DevOps

1. Introduction

De nos jours, il est difficile de ne pas avoir entendu parler de la méthodologie DevOps lorsqu'on est développeur ou administrateur système. Dans ce cours nous allons découvrir les fondements de ce mouvement, ses avantages et inconvénients.

Nous commencerons par découvrir d'où vient cette méthodologie et en quoi elle consiste, quelles sont ses caractéristiques. Nous essayerons de comprendre la culture DevOps puis nous verrons les piliers indispensables.

2. Origine et définition

Avec l'arrivée du cloud, les métiers de l'IT ont fortement évolué, concernant les compétences métier, la maîtrise d'outils mais aussi dans l'approche des besoins du secteur.

Alors qu'avant, le but était de produire des applications sans bug qui étaient livrées une fois par an, aujourd'hui elles doivent être produites et déployées en continu, avec des micro cycles. En effet, les applications doivent désormais être évolutives, performantes et hautement disponibles, au même titre que les infrastructures sur lesquelles elles sont déployées.

En 2009, le terme DevOps apparaît. **La démarche, issue de l'agile**, avait pour objectif d'établir une collaboration plus efficace entre les équipes de développements (les "Dev") et les équipes de production (les "Ops"), deux univers longtemps divisés par leur approches et objectifs.

La démarche DevOps a pour **but de faciliter les relations entre les différents corps de métiers** en se reposant sur le déploiement des applications en continu (**Continuous Integration**), sur les tests automatisés, et sur les livraisons en continu (**Continuous Delivery**). L'objectif de qualité est divisé entre les développeurs (dev) et les personnes en charge de la production (ops), en s'appuyant sur des outils d'automatisation.

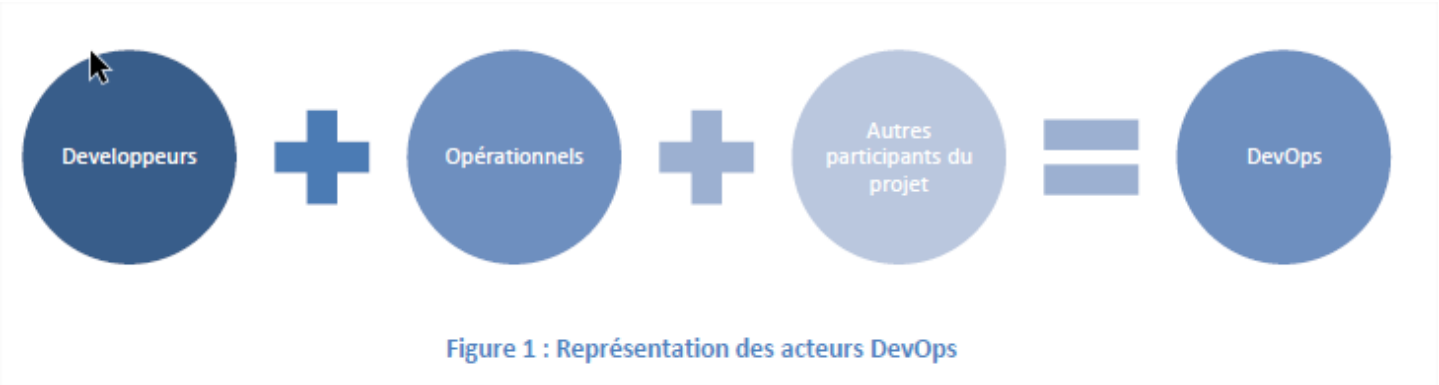

Figure 1 : Représentation des acteurs DevOps

L'expert DevOps va donc **faire évoluer les architectures systèmes** et mettre en place des outils et des services pour les développeurs : environnements de développement à l'image de la production, outils d'orchestration des

tests, déploiements automatisés. Ces nouveaux outils étant nombreux et nouveaux, il y a de forts besoins de profils d'Ops spécialisés en industrialisation du code applicatif.

De manière globale, un DevOps est chargé :

1. automatiser le cycle de vie d'une application ;
2. automatiser les installations "Infrastructure as a code" ;
3. concevoir et mettre en œuvre des architectures systèmes ;
4. superviser l'infrastructure et les services ;
5. assurer la haute disponibilité de l'infrastructure et les performances des applications ;
6. accompagner à la transformation des équipes vers l'intégration continue et la démarche DevOps.

3. Comprendre la culture DevOps

3.1. La culture

DevOps est plus qu'un mouvement, en 10 ans elle a bien évolué et apporte maintenant sa culture à l'ensemble du système d'information (SI) d'une organisation. Ainsi, DevOps cultive les bonnes pratiques IT au travers de ses valeurs. Néanmoins, pour obtenir des résultats, cela passe par l'adhésion complète des acteurs de la DSI, les "Dev", les "Ops" mais aussi toute la ligne managériale et décisionnelle. Devops touche ainsi la plupart des acteurs d'un projet IT.

Figure 2 : Les acteurs d'un projet IT complexe

3.2. Les valeurs

DevOps ne se limite pas à la mise en place d'outils et de processus dédiés, elle apporte aussi de nombreuses valeurs indispensables pour travailler collectivement au service d'un objectif commun dépassant les objectifs personnels et d'équipe.

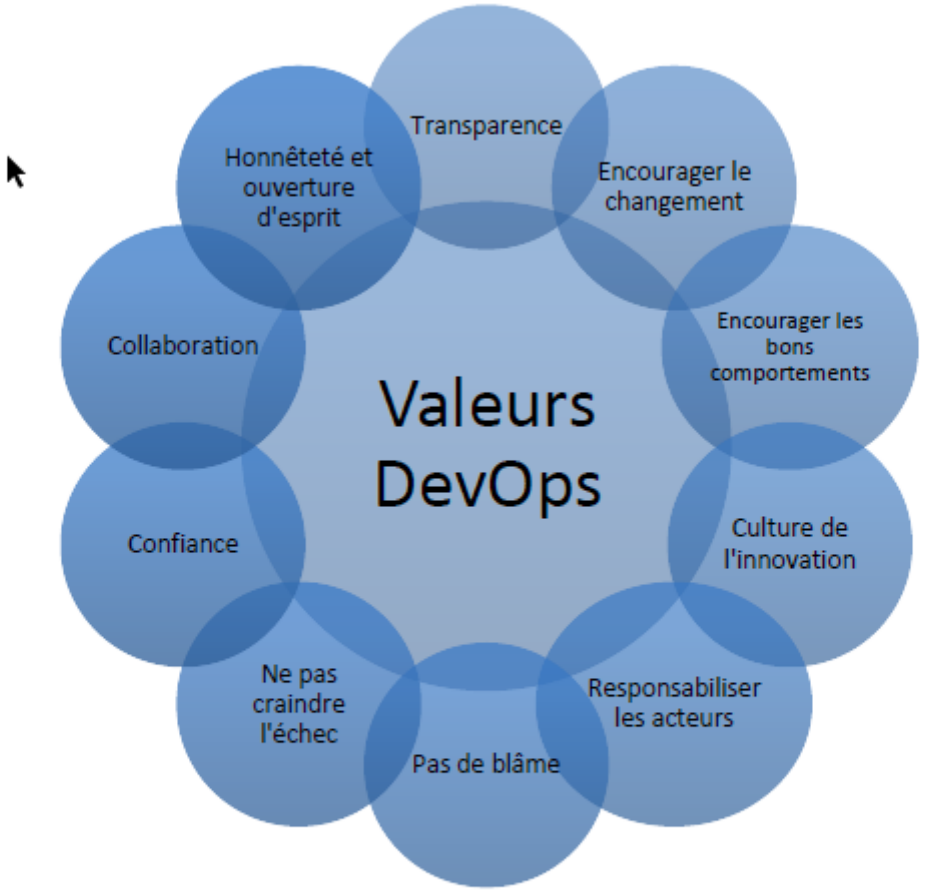

Figure 3 : Les valeurs de la culture DevOps

Transparence : l'avancement des tâches de chaque acteur du projet doit être clairement visible et ne doit pas être masqué ou susceptible d'une mauvaise interprétation. Cet aspect permet de disposer d'une vision concise sur l'état global d'un projet. C'est la seule façon de pouvoir anticiper les risques de retards de livraison et la transparence encourage la solidarité entre les acteurs.

Encourager le changement : le changement est une composante inévitable dans le monde du numérique. L'être humain a souvent tendance à être réfractaire à l'évolution car l'adaptation génère un stress inévitable pouvant être interprété par un risque d'échec possible. Or, le changement permet d'améliorer et faire évoluer les services IT proposés. Une fois la période d'adaptation écoulée, l'utilisateur final en retire des bénéfices certains.

Encourager les bons comportements : étape cruciale pour améliorer les pratiques dispensées dans l'organisation, les attitudes positives doivent procurer un sentiment de gratification pour l'acteur en question.

Culture de l'innovation : l'innovation est l'essence même de l'informatique, c'est une valeur clé destinée à encourager l'inventivité et l'ingéniosité dans la DSI. Il faut ainsi encourager les nouvelles idées à aboutir afin de faire évoluer le SI avec l'entreprise dans son environnement.

Responsabiliser les acteurs : chaque collaborateur doit prendre conscience de ses responsabilités, sans chercher à excuser ses erreurs. Une prise de responsabilité permet un gain de maturité dans la fonction exercée et met en avant les éventuelles faiblesses de travail à corriger.

Pas de blâme : il est nécessaire d'éviter l'apparition de règlements de comptes au sein d'une DSI. Le fait de blâmer autrui pour ses erreurs créer un sentiment d'animosité provoquant un climat accusateur et impropre à une bonne collaboration. De plus, la multiplication des blâmes entraîne le collaborateur dans une spirale émotionnelle négative impactant fortement son implication dans le projet et endommageant ses performances sur le long terme. Plus le collaborateur se sentira incompétent, plus ses résultats tendront à la baisse.

Ne pas craindre l'échec : la peur de l'échec constitue un frein à l'évolution. Elle positionne le collaborateur dans une zone de confort. Elle constitue par conséquent un frein à l'évolution qui n'est pas acceptable dans une DSI puisque seule la prise de risques permet d'améliorer les structures actuelles.

Confiance : un climat de confiance fluidifie l'interaction interservices au sein d'un projet. C'est une étape indispensable pour obtenir une coopération efficace entre les participants. De plus, elle permet de se dispenser de vérifications inutiles qui peuvent générer d'importantes pertes de temps. Le temps passé à vérifier le travail d'un autre acteur est du temps perdu sur l'avancement de ses propres tâches.

Collaboration : il est important que chaque partie prenante dans l'organisation de la DSI apprenne à collaborer avec les autres acteurs de façon sereine. Alors que les développeurs souhaitent à tout prix diffuser de nouvelles fonctionnalités, les services de production essayent de freiner les multiplications des mises en production afin de maintenir une infrastructure stable. Une collaboration efficiente consiste à réunir ces deux parties pour trouver l'équilibre tout en assurant le développement continu des applications.

Honnêteté et ouverture d'esprit : l'honnêteté fonctionne de pair avec la transparence. Elle permet d'obtenir une justesse dans l'équilibrage des rôles. C'est une façon de se concentrer sur le cœur d'activité avec facilité. L'ouverture d'esprit offre à l'organisation une meilleure écoute de l'environnement au service des opportunités d'évolution.

3.3. Un complément des méthodes Agiles

On décrit parfois DevOps comme le « chaînon manquant des méthodes Agiles ». Les méthodes Agiles sont des groupes de pratiques portées sur les projets en conception logicielle. Ces pratiques sont référencées au travers de l'Agile manifesto, un recueil co-rédigé en 2001 par 17 grands acteurs du domaine informatique qui a popularisé le terme « Agile ». Les méthodes Agile impliquent au maximum le client et permettent de lui offrir une grande rapidité de traitement de ses demandes.

Toutefois le nom de méthodes Agiles est considéré comme fortement réducteur puisqu'il ne s'agit pas uniquement d'une simple méthode de développement car elle apporte également une approche sur la gestion de produits, la communication et la satisfaction cliente. On parle alors d'esprit Agile, de culture Agile ou encore de courant Agile.

Concrètement, la vision de la gestion de projet est remplacée par la gestion de produits, car l'objectif final d'un projet est de donner naissance à un produit. Cet aspect a été défini en réaction des approches traditionnelles devenues inadaptées qui consistaient à répondre techniquement à un cahier des charges défini par le client, à développer le produit sans mesure temporelle puis à livrer le produit pour la phase de recettage. On parle ici d'effet tunnel qui peut générer des conflits avec le client et créer un déphasage entre le besoin initial du demandeur et l'application réalisée. En effet, le client pense le produit, l'exprime dans ses mots d'une façon qu'il estime suffisamment précise et la société de développement réalise la demande en interprétant le cahier des charges. Toutefois, l'expérience a démontré que le client exprime la quasi-totalité du temps de la frustration et de la

déception lors de la première utilisation de son produit car le résultat est différent de ce qu'il avait imaginé, et cela même si tous les points du cahier des charges ont été scrupuleusement respectés. Entre autres, la méthode Agile définit des jalons préalablement déterminés avec le client pour valider les développements en cours.

Pour se faire, les méthodes Agiles sont répertoriées en 4 valeurs et 12 principes généraux. Les valeurs définissent les axes fondamentaux sur lesquels la méthode repose tandis que les principes détaillent de façon plus précise les règles à respecter. Pour qu'une méthode soit considérée en tant qu'agile, elle doit respecter tous les principes et donc par extension se conformer aux 4 valeurs également.

3.3.1. Les 4 valeurs

L'équipe : « Les individus et leurs interactions, plus que les processus et les outils ». Dans l'optique agile, l'équipe est bien plus importante que les outils (structurants ou de contrôle) ou les procédures de fonctionnement. Il est préférable d'avoir une équipe soudée et qui communique, composée de développeurs (éventuellement à niveaux variables), plutôt qu'une équipe composée d'experts fonctionnant chacun de manière isolée. La communication est une notion fondamentale.

L'application : « Des logiciels opérationnels, plus qu'une documentation exhaustive ». Il est vital que l'application fonctionne. Le reste, et notamment la documentation technique, est une aide précieuse mais non un but en soi. Une documentation précise est utile comme moyen de communication. La documentation représente une charge de travail importante, mais peut pourtant être néfaste si elle n'est pas à jour. Il est préférable de commenter abondamment le code lui-même, et surtout de transférer les compétences au sein de l'équipe (on en revient à l'importance de la communication).

La collaboration : « La collaboration avec les clients, plus que la négociation contractuelle ». Le client doit être impliqué dans le développement. On ne peut se contenter de négocier un contrat au début du projet, puis de négliger les demandes du client. Le client doit collaborer avec l'équipe et fournir un compte rendu continu sur l'adéquation du logiciel avec ses attentes.

L'acceptation du changement : « L'adaptation au changement, plus que le suivi d'un plan ». Lla planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client tout au long du projet. Les premières livraisons du logiciel vont souvent provoquer des demandes d'évolution.

3.3.2. Les 12 principes

- Principe n°1 : La plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à forte valeur ajoutée.
- Principe n°2 : Le changement est accepté, même tardivement dans le développement, car les processus agiles exploitent le changement comme avantage concurrentiel pour le client.
- Principe n°3 : La livraison s'applique à une application fonctionnelle, toutes les deux semaines à deux mois, avec une préférence pour la période la plus courte.
- Principe n°4 : Le métier et les développeurs doivent collaborer régulièrement et de préférence quotidiennement au projet.
- Principe n°5 : Le projet doit impliquer des personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin et faites leur confiance quant au respect des objectifs.
- Principe n°6 : La méthode la plus efficace pour transmettre l'information est une conversation en face à face.
- Principe n°7 : L'unité de mesure de la progression du projet est un logiciel fonctionnel (ce qui exclut de comptabiliser les fonctions non formellement achevées).
- Principe n°8 : Les processus agiles promeuvent un rythme de développement soutenable (afin d'éviter la non qualité découlant de la fatigue).
- Principe n°9 : Les processus agiles recommandent une attention continue à l'excellence technique et à la qualité de la conception.
- Principe n°10 : La simplicité et l'art de minimiser les tâches parasites, sont appliqués comme principes essentiels.
- Principe n°11 : Les équipes s'auto-organisent afin de faire émerger les meilleures architectures, spécifications et conceptions.
- Principe n°12 : À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son processus de travail en conséquence.

3.4. Les méthodes dites Agiles

Depuis la publication de l'Agile manifesto en 1991, plusieurs méthodes se sont conformées aux valeurs et principes Agile. Nous pouvons en dénombrer 3 principales qui sont très largement répandues dans le monde du développement logiciel.

3.4.1. RAD (développement rapide d'applications)

Décrite pour la première fois en 1991 par James Martin, cette méthode présente une méthodologie idéale pour développer rapidement des applications. D'après son auteur, la méthode apporte 3 avantages compétitifs à l'entreprise : une rapidité de développement, un faible coût de production et une amélioration de la qualité de l'application. La méthode se fonde sur quatre principes fondamentaux :

- Les outils utilisés pour le développement : conception, prototypage, génération de code. Ils sont détaillés dans un référentiel.
- Les personnes intégrées aux projets doivent être compétentes, expérimentées et motivées.
- Le management de la gestion de projet doit être mis en avant tout en évitant d'être confronté à un mode bureaucratique trop exacerbé.
- La méthodologie utilisée se doit d'être à la fois bien formalisée et souple.

De nombreux principes sont également abordés tels que la spécialisation des rôles, les formalisations de processus ou encore l'organisation des divers types de réunions.

Cette méthode a été très largement utilisée dans les années 90, c'est également la première méthode qualifiée en d'Agile. Toutefois la tendance montre qu'elle est parfois délaissée au profit de Scrum et XP qui sont plus récentes.

3.4.2. Scrum

Il s'agit d'une pratique, créée par Jeff Sutherland et Ken Schwaber en 2004 dans l'Agile Software Management With Scrum, par laquelle le développement est échelonné par une fréquence de jalons très soutenue. On retrouve aujourd'hui la méthode Scrum dans plus de 50% des projets dits Agiles. Tout comme la précédente, cette méthode repose sur quatre principes fondamentaux (citations de Jeff Shutherland extraites de son manifeste pour le développement agile de logiciels) :

- **La planification est utile** : "Il est tellement tentant de dresser des plans sans fin, de détailler à l'envi tout le travail qui doit être accompli sur un projet massif... mais lorsque les plans détaillés ne correspondent pas à la réalité, ils s'effondrent. Incluez dans votre méthode de travail l'hypothèse du changement, de la découverte et de nouvelles idées."
- **Inspecter et adapter** : "De temps en temps, interrompez ce que vous êtes en train de faire, revoyez ce que vous avez fait et déterminez si c'est toujours ce que vous devez faire et si vous pouvez le faire mieux."
- **Changer ou disparaître** : "S'accrocher à l'ancienne manière de procéder et à l'ancien système de commandement et de contrôle et de prévisibilité rigide ne fera que précipiter l'échec. Pendant ce temps, les concurrents qui sont prêts à changer vous laisseront mordre la poussière."
- **Echouer rapidement pour pouvoir corriger au plus tôt** : "La culture d'entreprise accorde souvent plus de poids aux formulaires, procédures et réunions qu'à la création de valeur visible qui peut être inspectée à intervalles courts par les utilisateurs. Tout travail qui ne produit pas une valeur réelle est une folie. En travaillant sur les produits par cycles courts, vous pouvez obtenir au plus tôt les réactions des utilisateurs et éliminer immédiatement les efforts qui ne servent manifestement à rien."

De plus, Scrum repose sur trois piliers :

- **La transparence** : Scrum met en avant la communication entre les managers et leurs équipes afin de pouvoir accéder rapidement à une bonne compréhension de l'état global du projet.
- **L'inspection** : Un point d'inspection régulier doit être instauré et réalisé avec un inspecteur compétent.
- **L'adaptation** : Si un défaut est constaté lors de l'inspection, le développement en cours déjà réalisé doit être revu puis adapté.

Cependant la méthode est difficile à appliquer et les projets ont tout de même tendance à être rendus en retard et dans une situation de dépassement budgétaire. A la fin d'un sprint de développement, plus de 80% des équipes ne testent pas si les logiciels fonctionnent ou ne réalisent pas des tests complets sur les nouvelles fonctionnalités produites.

3.4.3. XP (eXtreme Programming)

En génie logiciel, cette méthode détaille des bonnes pratiques sur la réalisation d'une application tout en incluant une composante de gestion de projet. Elle s'adapte tout particulièrement aux équipes à effectifs réduits dotées de besoins variables. Comme son nom l'indique, elle pousse à l'extrême des principes de programmation simples. La méthode a été inventée par Ken Beck, Ward Cunningham et Ron Jeffries en 1999 au travers du livre Extreme Programming Explained.

La méthode repose sur cinq valeurs fondamentales :

- **La communication** : elle oblige les développeurs, les décideurs et les clients à communiquer au travers des tests, de la programmation en binôme (travail commun de deux développeurs exécuté sur un seul poste de travail) et par la tenue du planning.
- **La simplicité** : Ken Beck part du principe que la façon la plus simple d'arriver au résultat est la meilleure. Une trop forte anticipation est selon lui une perte de temps délibérée.
- **Le feedback** : le retour d'information sur un développement est primordial pour le programmeur et le client. Le feedback est ainsi porté sur les tests unitaires et les tests fonctionnels.
- **Le courage** : il concerne la motivation de changer l'existant en cours de phase de développement. Par extension, il permet de sortir d'une situation inadaptée.
- **Le respect** : ajouté dans le référentiel dans sa deuxième version, cette valeur traite du respect envers le client et les proches collaborateurs.

Ces valeurs se déclinent également en treize pratiques complémentaires :

- **Client sur site** : dans une situation idéale, un représentant du client doit se rendre physiquement sur site durant toute la durée du projet pour répondre aux questions des développeurs.
- **Jeu du Planning** ou (Planning Poker) : le client doit fournir des scénarios d'utilisation sur les fonctionnalités dont il souhaite disposer. Une fois la tâche accomplie, le temps de développement estimé pour chaque tâche est proposé au client qui priorise ensuite les développements en fonction du capital temps dont il dispose.
- **Intégration continue** : elle consiste à intégrer au corps du programme une fonctionnalité dès lors qu'elle a été développée en vue d'éviter toute surcharge de travail d'intégration.
- **Petites livraisons** : la multiplication des livraisons doit se dérouler le plus fréquemment possible dans l'optique de réduire les coûts relatifs à la livraison.
- **Rythme soutenable** : l'équipe de développement ne doit pas faire d'heures supplémentaires car un développeur fatigué est moins performant. Il est ainsi impensable de fournir un logiciel de moindre qualité au client.
- **Tests de recette** : en fonction des scénarios d'utilisation fournis par le client, des procédures de tests sont rédigées puis automatisées dans la mesure du possible. Cette tâche est parfois confiée à un collaborateur dont cette partie en constitue la mission principale.
- **Tests unitaires** : au minimum un test doit être réalisé pour chaque fonctionnalité et conservé durant toute la durée du projet. Cette partie apporte de l'exhaustivité dans la réalisation des tests unitaires avant les rendus planifiés et la livraison finale.
- **Conception simple** : imaginer les prochaines évolutions est une perte de temps sans garantie de retirer un bénéfice ultérieur. Par conséquent, concevoir une application très simple facilitera inévitablement les futures itérations
- **Utilisation de métaphores** : l'utilisation d'images verbales, à la place de termes techniques pointus, pour décrire le fonctionnement du système est préconisée pour faciliter la compréhension.
- **Refactoring** : le factoring est un remaniement du code destiné à améliorer régulièrement le code sans en modifier le comportement. Cela permet de restructurer les fondations d'une application en vue de l'ajout de nouvelles fonctionnalités. Ce principe fonctionne conjointement avec le principe de conception simple dans un souci de gain de temps. Il faut donc toujours développer l'application simplement sans anticiper les évolutions éventuelles et lorsque le besoin se présente adapter le code applicatif en fonction des nouvelles demandes.
- **Appropriation collective du code** : l'équipe est collectivement responsable de l'application. Cela signifie qu'un développeur pourra apporter une modification à une fonctionnalité qu'il n'aura pas développée personnellement.

- **Convention de nommage** : selon le principe précédent, il est impératif de définir une nomenclature commune et de la respecter attentivement.
- **Programmation en binôme** : les sessions de développement sont réalisées en binôme composés d'un « driver », qui utilise le clavier pour programmer, et d'un « partner » qui lui suggère de nouvelles fonctionnalités et qui détecte en temps réel les possibles bugs. Un driver doit changer régulièrement de partenaire pour améliorer la connaissance du produit et la communication au sein de l'équipe.

4. Les piliers

Les ressources pour tirer le meilleur parti du DevOps

Pour tirer le meilleur parti du DevOps, tous les employés doivent impérativement adopter une nouvelle façon de penser et d'agir. L'une des possibilités pour y parvenir consiste à définir un nouveau modèle d'exploitation où les employés responsables des services et fournitures informatiques et des processus métier travaillent en étroite collaboration et en interdépendance les uns par rapport aux autres. Les collaborateurs deviennent ainsi des professionnels « T-Shaped » : des experts qualifiés dans différents domaines capables de travailler de manière flexible avec toutes les équipes de l'entreprise. Dans la plupart des cas, il faut prévoir des formations ou augmenter le nombre d'employés.

Même si ces étapes sont souhaitables, les changements nécessaires peuvent être lents. Dans ce cas, il peut être judicieux de travailler en réseau avec des partenaires de son écosystème pour avoir accès à des qualifications et compétences supplémentaires.

Les processus

L'époque où le cycle de vie de l'approvisionnement produit durait de 18 à 24 mois est révolue. De nos jours, les entreprises doivent fournir de nouveaux produits, des mises à jour, des fonctionnalités et des correctifs sur une base trimestrielle, parfois même hebdomadaire. Le DevOps contribue à raccourcir les cycles de lancement des nouvelles versions pour les ramener à quelques semaines voire quelques jours. Malheureusement, dans les grandes entreprises, les processus agiles DevOps sont souvent ralentis voire freinés par les exigences de gouvernance, de conformité et de sécurité.

Pour surmonter ces difficultés, plusieurs approches alternatives sont possibles.

Des processus visant à raccourcir le délai entre l'idée, le développement du prototype et le lancement du produit peuvent être mis en place. Une autre option consiste à attribuer des objectifs de gouvernance en fonction du type de développement ou d'automatiser les processus de test et de validation. De plus, il faudra appliquer des modèles permettant de casser les silos individuels et de tous les connecter au sein de l'entreprise. Via ces étapes, les nouvelles versions seront disponibles plus rapidement.

La technologie

Le progrès est une épée à double tranchant. Les entreprises souhaitant tirer profit du DevOps doivent avoir recours à de nouvelles technologies, comme les solutions d'automatisation et le Cloud, et intégrer des innovations de manière continue, tout en protégeant leur infrastructure existante, les processus qui ont fait leurs preuves et leur propriété intellectuelle. Le grand défi est de poursuivre une approche associant la mise en œuvre d'une architecture multi-niveaux à une coopération continue entre les équipes internes et les partenaires, ainsi qu'à l'utilisation de plateformes de *crowdsourcing*.

Le crowdsourcing, la production participative, ou l'externalisation ouverte, est l'utilisation de la créativité, de l'intelligence et du savoir-faire d'un grand nombre de personnes, en sous-traitance, pour réaliser certaines tâches traditionnellement effectuées par un employé ou un entrepreneur.

L'information

En automatisant les procédures de développement et d'exploitation, les entreprises peuvent rationaliser et regrouper leurs processus, tout en identifiant les données pertinentes. Cela influe à son tour sur la manière dont les décisions sont prises et évaluées au moyen de chiffres clés directement liés aux objectifs commerciaux de l'entreprise. Les sociétés les plus progressistes ajoutent également des plateformes d'informatique cognitive (Cognitive Computing), afin de garantir un certain niveau de transparence tout en gardant le contrôle total de tous les processus.

Voilà pour la théorie. Dans la pratique, aucune entreprise ne possède aujourd'hui d'« équipe DevOps » exemplaire, car changer la façon de penser et de travailler exige plus que de simples processus de restructuration. Néanmoins, de plus en plus de sociétés se lancent dans l'aventure DevOps et obtiennent des réussites durables en matière d'innovation.

5. Les trois processus DevOps

5.1. L'intégration continue

L'intégration continue ou **Continuous Integration** est un processus orienté études consistant à *compiler*, *tester* et *déployer* sur un 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.

5.2. 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 telle sorte qu'il soit déployable en production dès la mise en recette.

5.3. Le déploiement continu

Le déploiement continu ou **Continuous Deployment** est un processus de production. Le but est de *compiler*, *tester* et *déployer* une application dans un environnement de production. Le Continuous Deployment nécessite que les processus de Continous Integration et de Continuous 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 en arrière peut être exécuté.

5.4. L'amélioration continue

Un dernier processus essentiel mais ne faisant pas partie de la méthodologie DevOps existe. Il s'agit de l'amélioration continue. Ce processus consiste à améliorer les trois processus précédents. Il est représenté comme un ensemble d'indicateurs capables de mesurer le "Time to market" et la qualité des processus précédents dans le but de faire ressortir des axes d'amélioration.

Time to market : c'est le temps que met une idée pour se transformer en fonctionnalité utilisée. Il est difficile d'être plus précis, car cette idée peut être un produit, un projet, une fonctionnalité ou tout simplement une user story. Il y a également plusieurs interprétations possibles de la date de fin : elle peut correspondre à la livraison (une équipe opérationnelle se charge de vérifier que la fonctionnalité est utilisable en production), à un test de la fonctionnalité par l'utilisateur final en production (sans que celle-ci soit encore officiellement proposée) ou une première utilisation de la fonctionnalité par un utilisateur lambda.

6. Les outils DevOps

6.1. Gestionnaires de versions

Les gestionnaires de versions, ou outils de versioning, sont des logiciels utilisés pour organiser le code source d'un programme informatique. Ils permettent de centraliser le stockage des fichiers sources dans un centre de dépôt, de conserver un historique des versions ou de chaque modifications apportées à un projet. Ils permettent en autre de conserver la possibilité de revenir en arrière facilement en cas de détection de bugs relatifs à un déploiement passé. Ces outils offrent donc une grande réactivité et une bonne organisation. De plus, ils sont indispensables en cas de travail collaboratif sur un projet. Chaque participant possède un compte attitré permettant d'identifier l'apport d'un collaborateur sur le programme. Un gestionnaire de version peut également alerter les collaborateurs lors de l'ajout de nouvelles fonctionnalités au projet en demandant une validation manuelle pour chaque conflit détecté.

Les utilisateurs des outils de versioning sont exclusivement des développeurs. Ils constituent un de leur outil de travail principal. Il peut également arriver qu'un administrateur accède au centre de dépôt pour récupérer et déployer une version en production. Cependant ces accès externes sont exceptionnels et seuls les développeurs doivent posséder un accès en écriture.

6.2. Outils d'intégration continue

Les outils d'intégration continue permettent d'agir sur les aspects de « continuous development » et de « continuous deployment ». Ils permettent de pousser l'automatisation des processus au maximum afin de gagner en productivité. Pour cela, ils articulent les déploiements et les tests des applications lors de la phase de

développement. Il s'agit en quelque sorte d'un assistant de développement traitant les tâches annexes de développement.

Ce type d'outils concerne principalement les développeurs qui en sont les principaux utilisateurs. Grâce à ces outils, les développeurs peuvent se concentrer pleinement sur leur cœur de métier, le développement d'applications.

Par ailleurs, la mise en production des logiciels est généralement chronophage et nécessite parfois de collaborer avec un opérationnel. Désormais, avec des outils d'intégrations continues adéquats, le développeur pourra maîtriser cette partie et multiplier les déploiements applicatifs. Il gagne donc en confort de travail et en autonomie.

Les outils de cette catégorie les plus connus sont GitLab CI, Travis CI, Circle CI, Jenkins et Docker.

6.3. Gestionnaires de configurations de serveurs

DevOps est certes une idéologie avant tout, mais son intégration dans une société passe aussi par l'utilisation de logiciels adaptés. Les gestionnaires de configurations sont essentiels à la pratique de DevOps. Ils permettent de déployer des fichiers de configuration sur un ou plusieurs serveurs simultanément. Ils permettent ainsi de réduire considérablement le temps dépensé par les administrateurs systèmes et réseaux sur les installations et les évolutions du parc de serveurs.

Par ailleurs, plus le nombre de serveurs utilisés dans une société est élevé, plus l'intégration de ces outils est judicieuse. En effet, l'utilisation de ce type d'outil permet de gagner en efficacité et en homogénéité. Le temps utilisé pour mettre à jour deux cents serveurs deviendra donc le même que pour un seul. L'administrateur sera certain que la configuration sera absolument identique sur toutes les machines. Le risque d'erreur humaine devient par conséquent quasiment nul. Certains outils sont également utilisables pour configurer des équipements réseaux tels que des commutateurs ou des routeurs.

Il existe de nombreux outils sur le marché pour répondre à ce besoin. Les plus connus sont Chef, CFEngine, Puppet et Ansible. Ils s'inscrivent directement dans le portefeuille de compétences des ingénieurs DevOps et des administrateurs systèmes.

6.4. Outils de supervision

Les outils de supervision, également appelés outils de monitoring en anglais francisé, sont des plateformes de contrôle du SI. Ils permettent de surveiller le bon fonctionnement d'un système d'exploitation, d'un équipement matériel, de services Windows ou Linux, de processus, de requêtes, de bases de données, de sauvegardes, etc. En cas de dysfonctionnement, une alerte est automatiquement mise en évidence dans le panneau de contrôle et peut également être envoyée par mail, sms, notification ou appel téléphonique.

Ce type d'outil est indispensable. S'il ne permet pas de gagner en productivité, il offre une réactivité incomparable sur le temps de détection des erreurs et donc par extension sur leur résolution. Une solution de supervision doit être entretenue avec rigueur et exhaustivité pour permettre au système d'être réellement performant. Une supervision partielle pourrait compromettre la détection d'erreurs potentielles dans le SI et donc provoquer des pannes critiques indésirables et non localisées.

Dans les grandes organisations, ces outils se destinent principalement aux services d'exploitations, ils constituent l'outil de travail principal des exploitants et des services de support. Dans les structures IT de taille plus modeste, ils peuvent être utilisés par les administrateurs systèmes et réseaux ou par les développeurs directement.

Il existe de nombreux outils de supervision sur le marché, offrant chacun plus ou moins de possibilités de configuration. La solution la plus utilisée sur le marché est de loin le couple NAGIOS/Cacti.

6.5. Outils de gestion de logs

Les gestionnaires de logs sont des outils destinés à collecter les logs, parfois appelés fichiers journaux pour les langages conservateurs français, de tout ou partie d'un SI. En plus de cette collecte centralisée, les outils sont désormais suffisamment évolués pour apporter une valeur ajoutée aux utilisateurs par une meilleure lisibilité. Il est désormais possible de recevoir des alertes sur des logs suspicieux. Par ailleurs des moteurs de recherche puissants et des filtres sont intégrés à ces outils pour faciliter l'accès aux informations désirées.

Par ailleurs, la centralisation des logs offre parfois une sécurité d'accès à l'information qui serait nullement possible autrement, par exemple en cas d'arrêt total d'un serveur ne pouvant plus redémarrer.

Ces outils sont principalement utilisés par les administrateurs, toutes spécialités confondues, pour comprendre les raisons d'une panne ou surveiller le bon déploiement de certaines applications sensibles.

Il existe de nombreux outils sur le marché.

6.6. Outils d'analyse de la performance

Les outils d'analyse de performances sont des logiciels permettant de mesurer de façon précise des points de fonctionnement d'un SI. Ils sont basés sur des mesures préalablement définies par les utilisateurs et peuvent être répétées si besoin. Ces outils sont le produit d'une volonté d'amélioration de la part des DSI qui cherchent à améliorer toujours plus les temps de réponse des applications. Elles permettent également de détecter les goulots d'étranglement des réseaux et donc d'améliorer les performances globales du SI.

Les utilisateurs de ces outils sont essentiellement les administrateurs réseaux et systèmes. Ils sont généralement utilisés de façon ponctuelle mais ils doivent néanmoins être mis en place lors de la mise en production de nouveaux équipements pour conserver un historique de fonctionnement et ainsi fournir une comparaison temporelle sur un point donné.

Le marché compte plusieurs solutions dont les plus connues sont gratuites. Iperf et Graphite sont très souvent utilisés dans les entreprises.

6.7. Solutions propriétaires DevOps

L'essor du mouvement DevOps a incité les principales sociétés de développement à produire des solutions complètes intervenant sur de multiples aspects du mouvement. Ils peuvent ainsi inclure dans leurs suites de nombreux modules de suivi de projet, d'état d'avancement, de déploiement automatisé, de versionning et de gestion de configuration.

Ces solutions ont pour avantages d'offrir de nombreux services complémentaires. Aucune interconnexion de produits n'est alors à mettre en œuvre. Cependant elles sont généralement très onéreuses et inflexibles. Il faut donc adapter l'organisation de la DSI à l'outil et non l'inverse. Ces solutions sont généralement peu adaptées aux réels besoins des organisations.

Il existe de nombreux produits de cette catégorie sur le marché. Les principaux sont Urban Code d'IBM, OpenShift de Redhat et Microsoft Azure.

6.8. Liste non-exhaustive d’outils DevOps

Outils	Description	Catégorie
Agile Manager	Agile Manager est une suite complète développée par HP permettant de créer des liens entre les équipes, de mettre en place automatiquement des indicateurs de performance et de suivre les états d'avancement d'un projet	Solutions propriétaires DevOps
Ansible	Ansible est un outil de développement continu permettant de déployer et gérer la configuration des applications	Outils d’intégration continue
Ant	Ant est un outil en lignes de commandes permettant de gérer les processus Apache2 et d'automatiser les opérations répétitives de développement	Outils d'intégration continue
Aws Amazon		
BCFG2	BCFG2 est un outil de gestion de serveurs permettant de déployer massivement des configurations	Gestionnaire de configuration serveur
Bluepill	Bluepill est un outil de supervision permettant de contrôler des processus ou des groupes de processus	Outils de supervision
Cacti	Cacti est un outil de supervision permettant de réaliser des graphique depuis des sources de données	Outils de supervision
Capistrano	Capistrano est un outil d'automatisation d'administration de serveurs permettant d'ordonnancer des scripts entre les différents noeuds	Gestionnaire de configuration serveur
CFEngine	CFEngine est un outil de management des configurations permettant d'automatiser des déploiements de configurations serveurs et de synchroniser des données entre des serveurs hétérogènes	Gestionnaire de configuration serveur
Chef	Chef est un outil de gestion de serveurs permettant de déployer automatiquement et massivement des configurations	Gestionnaire de configuration serveur

Circle CI	Saas	Outils d'intégration continue
Docker	Docker est un outil d'intégration continue permettant de construire, d'intégrer et exécuter des applications sur des plateformes distribuées. Il automatise le déploiement dans des conteneurs logiciels	Outils d'intégration continue
Fabric	Fabric est un outil d'intégration continue permettant de déployer automatiquement et massivement des applications sur des systèmes Linux par SSh	Outils d'intégration continue
Ganglia	Ganglia est un outil de supervision permettant d'assurer le contrôle de clusters	Outils de supervision
Git	Git est un gestionnaire de version permettant d'organiser et de gérer le versionning d'un logiciel	Gestionnaire de versions
GitLab		
God	God est un outil de supervision simple d'utilisation permettant de superviser le fonctionnement d'applications et de serveurs	Outils de supervision
Gradle	Gradle est un outil d'intégration continue qui permet d'automatiser le construction, les tests et le déploiement d'applications	Outils d'intégration continue
Graphite	Graphite est un outil de supervision enregistrant automatiquement des mesures prédéfinies et générant des graphiques statistiques	Outil d'analyse de la performance
Graylog	Graylog est un outil de centralisation de logs permettant de centraliser, stocker, rechercher et analyser des logs depuis n'importe quelle source prédéfinie	Outil de gestion des logs
Icinga	Icinga et Icinga2 sont des outils de supervision permettant de superviser tout type de ressources matérielles et de réaliser un reporting graphique ou mesuré	Outil de gestion des logs
Iperf	Iperf est un outil d'analyse de la performance permettant de mesure la performance des réseaux	Outil d'analyse de la performance
Jenkins	Jenkins est un outil open sources d'intégration continue permettant	Outils d'intégration continue

	d'automatiser et superviser l'intégration de tâches répétitives	
Kibana	Kibana est un outil de gestion des logs permettant de stocker et agencer des logs collectés en vue d'une analyse ultérieure	Outil de gestion des logs
Loggly	Loggly est un outil de management des logs permettant de détecter rapidement des causes problématiques au bon fonctionnement d'un système	Outil de gestion des logs
Logtash	Logstash est un outil de gestion des logs permettant de collecter, parser et stocker les logs après utilisation	Outil de gestion des logs
Maven	Maven est un outil d'intégration continue complémentaire à Apache2 permettant de gérer et automatiser la production de logiciel JAVA et JEE	Outils d'intégration continue
Mcollective	Marionnette Collective est un framework intégrable dans Puppet destiné à exécuter des tâches sur des groupes de serveurs	Gestionnaire de configuration serveur
Microsoft Azure	Microsoft Azure est une plateforme cloud permettant d'héberger des serveurs virtuels et de disposer de différents services tels que des workflow, des espaces de stockage, des synchronisations de données. Un ensemble d'API permet de faciliter l'accès à la plateforme	Solutions propriétaires DevOps
Monit	Monit est un outil de supervision des systèmes Unix-like permettant de gérer les problèmes pouvant apparaître lors du cycle de vie d'un serveur	Outils de supervision
Multihost SSH Wrapper	Multihost SSH Wrapper est un outil permettant d'exécuter simultanément une commande linux vers plusieurs systèmes hôtes	Gestionnaire de configuration serveur
Nagios	NAGIOS est l'un des outils de supervision les plus complet du marché, il permet de superviser toutes les ressources d'un SI	Outils de supervision
New Relic APM	New Relic APM est un outil de supervision permettant de mettre en place la supervision des applications et des bases de données de façon automatique	Outils de supervision

OpenShift	OpenShift est une PaaS produit par RedHat permettant de déployer automatiquement des applications, établir des workflow et suivre visuellement les étapes d'avancement d'un projet de développement	Solutions propriétaires DevOps
Pager Duty	Pager Duty est un outil d'agrégation pour les outils de supervision systèmes permettant d'améliorer la visibilité globale sur le SI et de fournir des alertes planifiées	Outils de supervision
PaperTrail	PaperTrail est un outil de gestion de logs permettant de centraliser, stocker et organiser la gestion des logs et fournit une interface web pour les consulter	Outil de gestion des logs
Perforce	Perforce est un outil de gestion de configuration et de version permettant d'organiser et archiver les modifications appliquées sur un développement tout au long de son cycle de vie	Gestionnaire de versions
Puppet	Puppet est un outil de management d'infrastructure permettant de superviser l'état d'un SI. Sa finalité est d'automatiser les livraisons logicielles sur des noeuds de serveurs	Outils d'intégration continue
Rancid	Rancid est un outil de gestion et déploiement de configuration réseau permettant de déployer automatiquement et massivement des configurations. Il maintient un historique des changements effectués	Gestionnaire de configuration équipements réseaux
Runit	Runit est un outil de supervision permettant de lancer des démons sur des plateformes Unix-like	Outils de supervision
SaltStack	SaltStack est un outil de gestion de configurations serveurs permettant de déployer des infrastructures et de les manager	Gestionnaire de configuration serveur
Sensu	Sensu est un outil de supervision permettant de détecter les problèmes de fonctionnement de services ou démon	Outils de supervision
Snorby Threat Stack	Snorby Threat Stack est un outil de supervision permettant d’organiser	Outils de supervision

	simplement le contrôle de points de sécurité	
SonarQube		Outil de qualité du code
Splunk	Splunk est un outil de collecte de logs permettant de faciliter l'accès aux informations	Outil de gestion des logs
Squid	Squid est un outil d'optimisation web permettant d'améliorer la performance des infrastructures web client-serveur	Outil d'analyse de la performance
Subversion	Subversion est un gestionnaire de version permettant d'organiser et de gérer le versionning d'un logiciel	Gestionnaire de versions
Supervisor	Supervisor est un outil de supervision visant à contrôler des processus systèmes sur des groupes de serveurs Linux	Outils de supervision
Travis CI	Saas	Outil d'intégration continue
Upstart	Upstart est un outil de supervision et de gestion des démons linux permettant d'agir sur les tâches de démarrage et les processus courants des serveurs	Outils de supervision
Urban Code	Urban Code est une suite complète portée sur le développement d'applications, le déploiement continue, la gestion des configurations et le versionning. La suite est	Solutions propriétaires DevOps
Vagrant	Vagrant est un outil de gestion de configurations serveurs utilisé pour créer et configurer des environnements de développement virtuels	Gestionnaire de configuration serveur