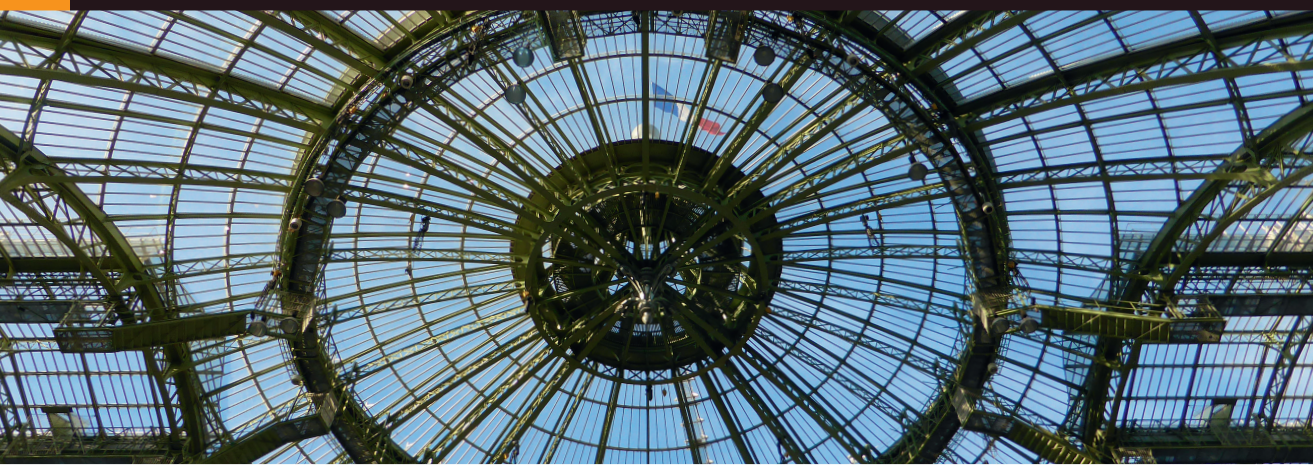
ÉTUDES, DÉVELOPPEMENT & INTÉGRATION

DÉCOUVRIR DEVOPS

L'essentiel pour tous les métiers



Stéphane Goudeau
Samuel Metias

Préface de Patrick Debois

DUNOD

Illustration de couverture :
Verrières du Grand Palais, Paris, 2016
© Christine Goudeau

5 rue Laromiguière, 75005 Paris
www.dunod.com

ISBN 978-2-10-075045-0

Avant-propos

Toutes les entreprises sont désormais confrontées au formidable enjeu que constitue leur transformation numérique, portées qu'elles sont par le maelström que constitue l'irruption du logiciel qui, pour citer Marc Andreesen, « dévore le monde »...

Cette transformation, portée par l'élan de la « destruction créatrice » chère à Schumpeter et par l'accélération que lui confère le numérique, leur impose de créer de la valeur de plus en plus vite pour leurs clients. Afin, tout simplement, de s'adapter pour survivre... Ce sont ces raisons qui ont poussé les entreprises à adopter – bon gré mal gré – des approches agiles afin de faire évoluer leur système d'information et de répondre aux besoins de leurs clients. Cette transformation vers l'agile ne se fait généralement pas sans mal car elle remet en cause bien des pratiques managériales et nécessite un changement de paradigme culturel au sein de l'entreprise ; en effet, l'agile se fonde sur des valeurs telles que la collaboration et la valorisation des équipes plutôt que des individus, l'auto-organisation, la communication, l'adaptation permanente, le droit à l'échec... Autant de valeurs qui peuvent remettre en cause certains principes culturels et managériaux des entreprises historiquement liées à la mise en place d'organisations souvent rigides, bureaucratiques, hiérarchiques et bien peu transparentes.

Mais passer à l'agile n'est pas suffisant et ne constitue généralement que la première étape d'un voyage dont l'objectif n'est autre que de pouvoir créer en continu de la valeur pour les utilisateurs et donc pour ses clients. En effet, il ne suffit pas pour les équipes de développement de pouvoir délivrer de nouvelles fonctionnalités applicatives à l'issue de chacune des itérations du processus de développement ; il faut aussi que ces nouvelles fonctionnalités puissent être mises en production immédiatement et, si possible, de manière entièrement automatisée afin de minimiser au maximum les risques sur la production. Or, dans beaucoup d'entreprises, les équipes en charge du développement (« Dev ») et de la production (« Ops ») sont généralement situées dans des silos organisationnels distincts avec des objectifs pour le moins antagonistes : les premiers ont généralement pour mission de délivrer de nouvelles fonctionnalités le plus vite possible afin de répondre aux besoins des métiers, les seconds ayant la responsabilité opérationnelle du bon fonctionnement du système d'information, de sa stabilité, de sa continuité de services, etc. L'objectif du mouvement « DevOps », objet

de cet ouvrage, n'est autre que d'abattre les murs entre ces deux organisations en créant une véritable synergie entre les développeurs et les opérationnels de la production. Avec la volonté commune de délivrer en continu de la valeur aux utilisateurs.

Mettre en production en continu n'est pas chose aisée et nécessite la mise en œuvre d'une automatisation des processus de déploiement et d'approvisionnement et de celle de la chaîne de fabrication logicielle, la prise en compte de ces problématiques au sein même de l'architecture logicielle afin de pouvoir dé-corréler le déploiement des fonctionnalités du déploiement du code lui-même, la mise en place d'une véritable culture de la mesure du fonctionnement du logiciel et du système sous-jacent, etc. Autant de questions de nature technique abordées au fur et à mesure, de manière très didactique, au sein de cet ouvrage. Mais il ne faut pas s'y tromper, l'essentiel de cette transformation repose sur une évolution fondamentale de la culture de l'entreprise et de son organisation.

Ce voyage transformationnel a été entrepris au sein d'une entreprise comme Microsoft depuis plusieurs années et – je peux personnellement en témoigner – ce voyage n'a pas toujours été facile… Car il ne suffit pas pour chacune des organisations « Dev » et « Ops » d'atteindre, chacune de son côté, un optimum local mais bien à l'organisation toute entière d'atteindre un optimum global. Ce qui nécessite d'avoir une approche systémique et donc de repenser complètement l'organisation, son mode de fonctionnement, les méthodes d'évaluation des performances des collaborateurs, etc. Ceci a nécessité notamment une implication personnelle du plus haut niveau managérial de l'entreprise et une évolution très significative de chacun de ses groupes produits mais, désormais, des résultats tangibles sont visibles pour chacun de nos clients, avec la mise en place d'une logique « *as a service* » pour l'ensemble de nos logiciels, non seulement quand ils sont mis en œuvre sous forme de services dans le cloud mais aussi ceux qui sont mis en œuvre chez nos clients et partenaires, y compris par exemple Windows 10 qui évolue désormais en continu.

« *Ceux qui comprennent ne comprennent pas qu'on ne comprenne pas* » nous disait Paul Valéry. C'est parce que Stéphane et Samuel qui ont choisi d'écrire cet ouvrage ne croyaient pas que cette citation leur était destinée qu'ils ont chaussé leurs bottes de pédagogues et mis tout leur talent au service de la mise à la portée du plus grand nombre de concepts quelquefois bien peu évidents.

Bernard OURGHANLIAN
Directeur Technique et Sécurité
Microsoft France

Préface

Le succès d'une fête ne se juge pas au nombre d'interactions entre ses participants. De même une collaboration efficace ne se mesure pas avec de simples métriques, telles que le nombre de réunions, de tickets, etc. Nous savons tous que plus nous sommes ouverts au dialogue et à la compréhension de l'autre, meilleure sera la qualité des interactions. L'empathie devient alors un état d'esprit.

Ce livre présente DevOps sous différents angles, ceux de tous les participants à la chaîne de production d'un logiciel selon leur place dans cette chaîne. Réexaminer les problèmes communs à travers le regard des autres améliorera votre compréhension et élargira votre vision de ces problèmes.

Le long chemin qui va de la décision initiale à la mise en production est ici décrit comme dans un guide de voyage qui expliquerait les étapes d'un développement DevOps dans une entreprise.

Parfaitement équilibré entre les aspects « technos » et les aspects « processus », il explique les interactions entre les deux. Les uns n'existent pas sans les autres et ils s'influencent mutuellement. Faire tomber les silos organisationnels et techniques est essentiel pour changer les mentalités.

Bonne lecture !

Patrick DEBOIS
Pionnier du mouvement DevOps

Table des matières

Introduction

Lorsque Patrick Debois créé le terme DevOps en 2009, il est certainement loin de se douter qu'il est l'un des pionniers d'un mouvement dont l'influence ne cessera de s'accroître dans le monde des technologies de l'information. Comment imaginer qu'aujourd'hui une démarche dont l'un des objectifs est d'établir une collaboration plus efficace entre les équipes de développement et d'infrastructure ait pu susciter un tel intérêt ?

En appliquant une démarche DevOps, les services informatiques sont aujourd'hui à même de poursuivre leurs activités existantes en toute efficience. Ils peuvent notamment réaliser des tâches qui leur étaient jusqu'alors impossibles. Pour ce faire, ils doivent se renouveler, se transformer et s'adapter. Cette évolution est d'autant plus nécessaire que la nature même des métiers de l'IT a changé. Jadis, il s'agissait *juste* d'écrire du code exempt de bug, de livrer une nouvelle version tous les ans, puis de recommencer. Aujourd'hui, les applications doivent être produites et déployées en continu. À l'ère du cloud, les solutions logicielles doivent être évolutives, disponibles, hyperperformantes avec une latence plus faible et, bien entendu, à moindre coût. DevOps permet aux équipes de développement et d'infrastructure d'être plus réactives face à ces nouvelles exigences.

DevOps est par conséquent le moyen de concrétiser cette évolution, avec comme philosophie, l'idée d'un monde dans lequel chaque composante de l'organisation d'une entreprise collaborerait efficacement pour l'atteinte de mêmes objectifs. Il s'agit tout d'abord de pallier les conséquences négatives issues de la séparation des développeurs et des responsables opérationnels d'une organisation : le fameux *wall of confusion*. En effet, le développeur cible avant tout la production de code qui répond aux exigences fonctionnelles. Il est donc fort possible qu'il ne s'intéresse guère à la maintenance de la solution en fonctionnement opérationnel. À l'inverse, les responsables système ne seront guère enclins à favoriser des changements qu'ils considèrent comme autant de risques pour la stabilité de l'application.

DevOps apporte des réponses à cette problématique par la mise en application de différents concepts d'ordre culturel et technologique. De plus, avec l'avènement du cloud computing, DevOps est devenu un passage obligé : le succès de la mise en œuvre d'une démarche DevOps et la réussite d'une évolution vers le cloud sont intimement

liés. Toutefois, le champ d'application de DevOps va bien au-delà du périmètre du cloud.

Samuel et moi sommes donc convaincus de l'intérêt de DevOps. Malgré tous les bienfaits que l'on attend de cette philosophie, nombreux sont ceux qui s'interrogent encore sur la nature exacte de cette démarche et qui hésitent encore à l'adopter, et c'est la raison pour laquelle nous avons souhaité rédiger cet ouvrage.

Nous nous proposons de vous faire découvrir notre vision de DevOps et de son rôle dans la transformation digitale des organisations. Nous présenterons les principes de sa mise en application et les illustrerons avec différents exemples de mise en œuvre, au sein de grandes entreprises pour lesquelles cette démarche constitue un élément clé du processus de *continuous delivery*. Enfin, nous étudierons comment les évolutions technologiques les plus avancées peuvent influer les outils et les processus DevOps de demain.

À qui s'adresse ce livre ?

Ce livre s'adresse à toutes les personnes intéressées par les systèmes d'informations modernes et innovants, à tous les passionnés d'informatique qui pensent que l'organisation est aussi importante que la technique pour réussir, ainsi qu'aux familiers de la notion d'agilité dans le monde de l'informatique.

La structure de ce livre se veut facile d'accès. Quel que soit votre rôle au sein de l'entreprise ou votre usage de l'outil informatique, ce livre se veut donc pédagogique avant tout.

Aussi, comme l'illustre la figure suivante, en fonction de votre profil de lecteur, certains chapitres vous seront sans doute plus directement bénéfiques que d'autres bien que tous présentent un intérêt certain.

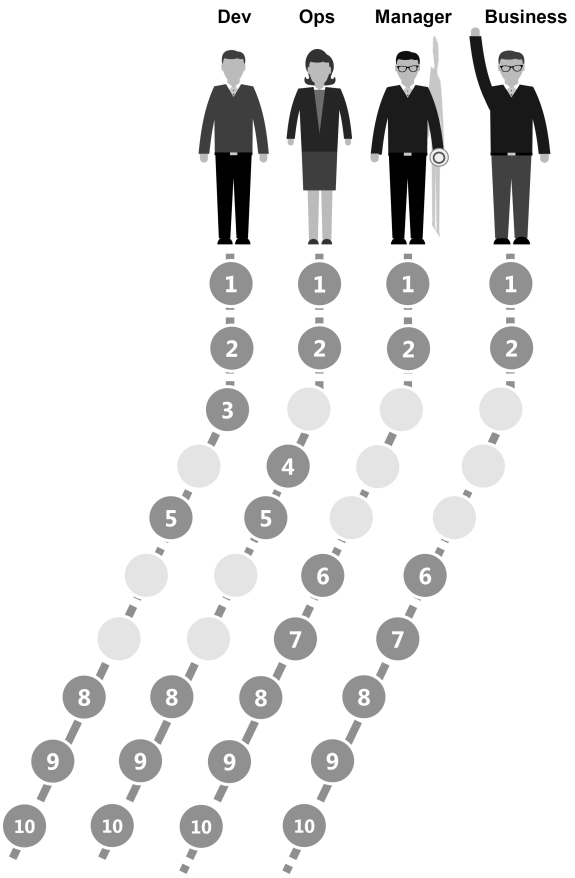
Nous avons fait le choix d'aborder le sujet sous différents points de vue, notre but étant de répondre au mieux aux interrogations et problématiques pratiques des différents métiers impactés par DevOps. Le fait même d'étudier une démarche qui se veut collaborative du point de vue des différents acteurs nous a parfois amenés à revenir parfois sur le même sujet dans différents chapitres. C'est un choix volontaire, le regard de différents acteurs posés sur ce même sujet peut parfois amener à une perception différente…

Pédagogique, accessible, pragmatique, voilà les maîtres mots qui ont guidé la rédaction de cet ouvrage…

Qui sommes-nous ?

Stéphane Goudeau est architecte dans la division Développeurs Expérience (DX) de Microsoft France.

Il travaille dans l'industrie des systèmes d'information depuis près de 25 ans. Après un début de carrière chez Bull Intégration Services, il a rejoint Microsoft comme consultant en 1996. Depuis, ses multiples rôles et responsabilités, exercés en tant que consultant principal, architecte, ou directeur technique du Microsoft Technology



Center, lui ont permis d'acquérir une expertise de l'offre Microsoft sur les technologies de développement et d'infrastructure, ainsi qu'une une expérience approfondie dans la conception et à la mise en œuvre des systèmes d'information à destination de start-up, sociétés d'édition logicielle ou grands groupes français.

Engagé sur le cloud dès les premières heures de la plateforme Microsoft Azure, il s'est totalement investi dans l'exploration de ses modèles d'architecture, de développement et de gestion opérationnelle. C'est ainsi qu'il est devenu adepte de la philosophie DevOps depuis maintenant plusieurs années.

Passionné par l'agilité, l'innovation et le management, Samuel Metias a une expérience importante en conseil autour des méthodes agiles et de l'innovation. Il a également une expérience en tant qu'adjoint au maire de Colombes avec environ 200 agents sous sa responsabilité.

Concernant l'agilité, il a coaché de nombreux projets, participé à la construction de méthodologies agiles et formé des équipes sur les pratiques.

Plus spécifiquement, il est leader au sein de la division Service de Microsoft France de l'offre Agile & DevOps. Il anime la communauté DevOps de l'ensemble de la filiale

française de Microsoft et depuis novembre 2015, il est responsable de l'alignement des offres DevOps de Microsoft Services à travers le monde.

Dans ses postes précédents, il a également une expérience d'architecte d'entreprise. Il a participé à la définition des méta-modèles d'architecture d'entreprise, à la conduite du changement suite aux architectures retenues ainsi qu'au pilotage des projets notamment autour des acteurs métiers.

Chez Microsoft, il s'appuie sur l'expérience importante des groupes produits de Microsoft pour en tirer les meilleures pratiques. Dans la continuité de l'agilité, il aide clients et partenaires à en tirer profit selon leur contexte.

Remerciements

Les auteurs tiennent tout d'abord à remercier leurs familles respectives qui ont patiemment supporté les longues heures de travail passées à la rédaction de ce livre au détriment du temps passé ensemble. Merci tout d'abord à nos épouses et nos enfants.

Nous tenons à remercier chaleureusement M. Debois qui a accepté de nous relire et de signer cette préface en toute amitié.

Nous tenons également à remercier notre employeur commun qui nous permet de baigner quotidiennement dans l'univers DevOps en interne comme en externe.

Stéphane tient à remercier tout spécialement ses amis visionnaires et ex-collègues Blaise Vignon et Jakob Harttung, qui l'ont invité, il y a quelques années, à s'intéresser à DevOps, ainsi que son manager Guillaume Renaud et son directeur Nicolas Gaume, qui l'ont encouragé dans cette voie : un voyage qui lui aura permis d'apprendre, en continu, sur bien des sujets... Et pour ce voyage, la couverture de notre ouvrage nous offre une nef un peu particulière, celle du Grand Palais : merci à Christine Goudeau pour ses talents de photographe.

Samuel tient à remercier tout particulièrement Monsieur Jean-Patrick Ascenci qui a été son premier mentor et manager dans sa carrière professionnelle et qui lui a donné le gout de son métier ainsi que Messieurs Mario Moreno et Charles Zaoui qui ont été des modèles d'expertise et de bienveillance et l'ont fait plonger dans le monde de l'agilité qu'il n'a plus jamais quitté.

Samuel a également une pensée particulière pour Antoine Durand et Laurent Le Guyader ses accolytes au quotidien autour de l'Agile et de DevOps ainsi que pour Randa Debbi au soutien toujours indéfectible et si précieux.

Enfin, nous remercions chaleureusement nos relecteurs qui nous ont inondés de nombreux et bons conseils. Nous tenions à les citer ici : Jean-Luc Blanc, notre éditeur, Charlotte Dando, Gaelle Cottenceau, Blaise Vignon, Franck Halmaert, Hervé Leclerc, Marc Gardette, Stéphane Vincent, Jean-Marc Prieur, Julien Corioland, Guillaume Davion, Jivane Rajabaly, Laurent Le Guyader, Benjamin Guinebertière...

1

La démarche DevOps enfin expliquée

Patrick Debois raconte qu'il a inventé le terme DevOps, après une série de conférences, peu fréquentées d'ailleurs, tandis qu'il cherchait à qualifier simplement la notion de gestion agile de l'infrastructure.

L'important n'est donc pas tant de savoir comment ni pourquoi Patrick Debois a utilisé pour la première fois ce terme, mais plutôt de comprendre ce qu'il cherchait à expliquer et que l'on pourrait résumer en : les infrastructures informatiques et les *opérations* peuvent être agiles. Plus encore, elles devront, elles auront l'obligation de l'être demain pour réussir.

Il faut remarquer aussi que l'usage du mot DevOps n'est pas anodin. Sa simple concaténation marque l'impérieuse obligation de collaborer étroitement entre les *développements* et les *opérations*. La collaboration est une nécessité, mais collaborer étroitement en partageant largement jusque dans la responsabilité de l'échec ou du succès est la philosophie DevOps. Un véritable choc culturel.

1.1 CULTURE

Comme nous l'avons dit, même si elle peut être facilitée par l'adoption de nouveaux outils et technologies, la démarche DevOps est avant tout une philosophie. Sa dimension culturelle est donc fondamentale.

Réduction des cycles de livraison	Optimisation des ressources	Amélioration de la qualité	Une nouvelle culture
Par l'industrialisation de la chaîne complète de production logicielle.	Par leur gestion unifiée, par l'autoscaling natif et par la possibilité d'automatiser par code les déploiements et la configuration.	Et de la disponibilité par l'instrumentation, la supervision et les tests.	Fondée sur la collaboration et sur une permanente recherche de l'amélioration sur l'apprentissage en continu.

Figure 1.1 — Les fondamentaux d'une démarche DevOps

1.1.1 Confiance réciproque et compréhension globale du système

Au cœur de cette culture, il y a la volonté de changer le mode d'interaction entre les équipes de développement et les opérations. L'objectif est non seulement de partager l'information, mais aussi les responsabilités, ce qui suppose une évolution des mentalités pour parvenir à établir la relation de confiance requise et l'implication de tous les acteurs.

Au-delà de cette confiance, il faut que chacun puisse acquérir une compréhension globale du système, de sorte que nul ne puisse ignorer les besoins ou les contraintes de l'ensemble des acteurs du système d'information. Cela se traduit par une évolution de l'organisation, de ses processus, du rôle et des périmètres de responsabilités de chacun.

1.1.2 Kaizen : la recherche de l'amélioration continue

Pour parvenir à étendre la portée de l'ensemble des acteurs du système, la culture DevOps favorise le développement des compétences dans une recherche perpétuelle d'amélioration. Cette approche est similaire au processus industriel Kaizen, qui s'est développé au Japon, dans la reconstruction qui a fait suite à la seconde guerre mondiale. Le mot *Kaizen* est issu de la fusion des deux mots japonais *kai* et *zen* qui signifient respectivement *changement* et *bon*.

Figure 1.2 — Le Kaizen en caractères japonais

La pratique quotidienne de cette démarche d'amélioration continue est la condition sine qua non de la réussite de sa mise en œuvre. Elle permet une évolution progressive, sans *à-coups*, et incite chaque acteur du système à proposer des optimisations simples et concrètes sur l'ensemble de la chaîne de production. Appliquée au cycle de production logicielle, cette réorientation culturelle impacte l'ensemble

de l'organisation, qui devient ainsi plus prompte à s'adapter et à rechercher le changement plutôt que de le fuir.

1.1.3 Lean Startup : construire, mesurer, apprendre

Autre source d'inspiration pour la culture DevOps, la philosophie *Lean Startup* proposée en 2008 par un américain, Eric Ries, initialement à destination des start-up. Depuis, la cible de cette démarche s'est étendue à l'ensemble des entreprises qui souhaitent mettre à disposition un produit sur le marché. Reprenant certains des principes du *Lean Management* (là encore une méthode de rationalisation de la production qui nous vient de l'industrie japonaise, en l'occurrence Toyota), Lean Startup vise à réduire les gaspillages et augmenter la valeur en continue pendant la phase de développement du produit.

Cette approche se traduit par une volonté d'amélioration continue de la performance (en termes de productivité et de qualité), par une réduction des délais, des coûts et la définition d'un *produit minimum viable* que l'on peut soumettre à l'évaluation des consommateurs. Cela suppose la mise en place de systèmes de mesure et des processus de remontée d'information systématique et le développement d'une culture fondée sur l'apprentissage en continu.

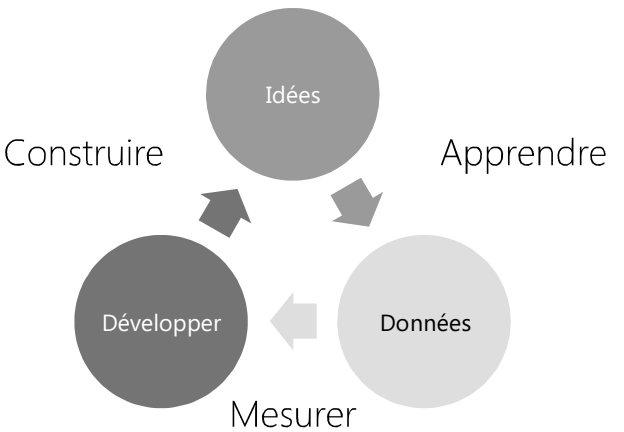


Figure 1.3 — Le triptyque fondamental du Lean Startup

Au-delà de *construire*, il y a les hypothèses et la décision de construire. Cette décision est prise en fonction de ce qui est appris du comportement et des attentes des utilisateurs. Cet apprentissage se fait à partir des données mesurées. Ces données sont mesurées par des instrumentations mises en place en fonction des hypothèses à valider. L'objectif est de permettre à l'entreprise de se doter des moyens permettant de contrôler en permanence l'adéquation entre la vision du produit, son implémentation et les attentes du marché.

Ces mesures permettront de valider les hypothèses ayant conduit à la définition du cahier des charges du produit. Mais elles permettront aussi d'optimiser l'intégralité

des chaînes de valeur métier dépendant de services informatiques, et de s'assurer de sa fiabilité en résolvant les problématiques au plus tôt afin de limiter leur impact.

Elles permettront de réagir au plus tôt pour procéder aux changements requis et permettront d'étalonner la performance commune, puisque dans la démarche DevOps, les résultats, comme la livraison d'un service en production, sont désormais partagés. Elles nécessiteront la mise en place de processus communs de déploiement, de supervision (détection et prévention d'incidents de performance, de sécurité, de disponibilité), de support et de remédiation.

1.1.4 Une vision positive de l'échec

Enfin, cette culture qui recherche le changement, va encourager l'expérimentation en proposant une vision positive de l'échec. Pour définir de nouveaux besoins, il faut accepter de prendre des risques et cela ne doit pas être un frein. L'échec, au même titre que le succès, devient une source d'apprentissage (*Fail fast*). Anticiper une situation où le système ne répond plus et y remédier dans les plus brefs délais suppose l'implication de l'ensemble des acteurs. La capacité du système à se remettre en service après un dysfonctionnement sera augmentée si les plans d'escalade et processus internes sont partagés entre les équipes.

Une bonne pratique permettant de valider l'efficacité de ces processus collaboratifs consiste à volontairement introduire des erreurs dans le système (*Fault Injection Testing*) et à gérer la situation qui en résulte en maximisant la collaboration entre les équipes. Un exemple de cette approche est le service *Chaos Monkey* que Netflix a développé sur la plateforme Amazon Web Services. Nous reviendrons d'ailleurs plus en détail sur cette implémentation dans la suite de cet ouvrage.

Autre exemple, la démarche *Resilience Modeling and Analysis* (RMA), issue du standard de l'industrie *Failure Mode and Effects Analysis* (FMEA), qui consiste à identifier en amont les éléments risquant une panne au sein d'une solution et les modes de défaillance qui en résultent. L'objectif est de définir les stratégies de résilience et de disponibilité dès la phase de conception de l'application en bâtissant un modèle qui détaille les possibilités de dysfonctionnement et qui documente l'ensemble des actions destinées à en atténuer l'impact.

Les contre-mesures ainsi définies pour s'assurer que l'incident ne puisse pas se produire ou pour limiter ses effets concernent les équipes de développement ainsi que les opérations. C'est l'un des objectifs de la modélisation des menaces dont nous reparlerons dans le chapitre 5 *DevOps vu par la qualité*.

1.1.5 Une culture aux multiples facettes...

Comme on peut le constater, la culture DevOps s'inspire donc d'une pluralité de disciplines déjà mises à l'épreuve avec succès dans le monde de l'industrie. Elle est fondée sur le respect mutuel des équipes, la confiance réciproque et le partage des responsabilités. Elle s'appuie sur une organisation et un management adapté. Enfin,

elle s'inscrit dans une démarche d'amélioration, d'expérimentation et d'apprentissage en continu.

1.2 COLLABORATION

1.2.1 Une implication du métier

La structure même du mot DevOps nous indique que la démarche s'appuie largement sur la collaboration entre le monde des développements et celui de l'infrastructure. La structure du mot est moins explicite sur le troisième acteur essentiel de cette collaboration : le métier.

En effet, la collaboration qu'insuffle cette démarche n'a qu'un seul objectif : réussir à impliquer les acteurs métiers de manière pertinente et continue. Cet objectif n'est pas nouveau en soi, mais les expériences précédentes n'ont pas toujours été concluantes, même avec des méthodes agiles.

Comment pouvons-nous espérer qu'un acteur métier s'implique durablement dans les projets informatiques si au moindre déploiement, il est le témoin privilégié des mésententes entre ces deux acteurs essentiels de l'IT ?

Réussir à reconstruire la confiance et à la conserver dans le temps est donc le premier objectif de toute démarche DevOps. Pour reconstruire cette confiance, il faut avoir conscience des *clés de confiance* de chacune des parties et travailler prioritairement sur ces sujets. En général, la confiance n'est jamais très loin. Il manque souvent juste un peu de volonté… et de méthode !

1.2.2 Identifier la clé de confiance des opérations

Pour réussir à construire la confiance entre deux mondes qui cohabitent, il faut identifier les leviers sur lesquels elle peut se construire. Chaque rôle dans cette collaboration n'espère pas bâtir cette confiance pour les mêmes raisons. Leurs objectifs premiers sont différents mais complémentaires.

Les équipes opérations tiennent avant tout à maintenir la production fonctionnelle et disponible. Certes, pour y parvenir sans recourir au paradigme de la stabilité à tout prix et gagner en fiabilité, ils doivent maîtriser les impacts de tous les changements, mais pas seulement. En réalité, l'une de leurs principales attentes vis-à-vis des équipes de développement est la qualité du produit qui leur est livré. De leur point de vue, si le produit est de qualité suffisante, il n'aura pas d'impact néfaste sur la production.

Et un produit de qualité du point de vue de des équipes de production n'est pas un concept particulièrement complexe : c'est un produit sans bug majeur sur un environnement *iso-production*. En théorie, pour y parvenir, il suffit de tester. Mais la théorie n'est pas si simple à appliquer pour de multiples raisons, parmi lesquelles certaines sont très courantes :

- **La stratégie de recette et de tests est rarement partagée**. Puisque ce sont les développements qui assument la responsabilité de la grande majorité des tests,

la typologie de tests à réaliser comme les résultats de ces derniers ne sont pas partagés.

- **La stratégie de recette et de tests n'est pas toujours définie et/ou rigoureusement respectée**. Lorsqu'une typologie et un planning de tests existent, ils ne sont pas toujours respectés rigoureusement : cela finit toujours par se savoir et par éroder la crédibilité de tout le plan qualité.
- **La stratégie de recette et de tests n'est pas appliquée par les bonnes personnes**. Il arrive trop souvent que le code soit implémenté, les tests écrits et le *sign-off* réalisé par la même personne. Il est évident que cela nuit à la crédibilité du résultat.

Pour que les équipes en charge des opérations souhaitent rechercher de nouvelles formes de collaboration, tous ces obstacles doivent être levés en définissant ensemble une stratégie de tests et de recette, garantissant un niveau de qualité satisfaisant pour tous et appliquée avec rigueur.

Vous l'avez compris la stratégie de tests et de recette est l'une des clés de confiance des équipes opérations pour réussir une démarche DevOps.

1.2.3 Identifier la clé de confiance des développements

Démontrer la qualité du produit développé n'est pas suffisant en soi. Il ne faut pas croire d'ailleurs que les développeurs rechignent à réaliser des produits sans bugs et sans dysfonctionnements. Au contraire, la véritable difficulté qui constitue leur clé de confiance est leur capacité à disposer des bons éléments pour réaliser les tests qui les rassurent sur la qualité de leur produit.

Dit autrement, il s'agit pour les équipes de développement de disposer d'environnements *iso-production* à la demande dans des temps raisonnables et d'outils permettant de réaliser un maximum de tests avec le moins d'effort supplémentaire possible.

Il ne faut pas négliger non plus l'importance du cadre défini et généralement accepté qui détermine ce qu'est un produit de qualité du point de vue de l'organisation.

Lorsque se pose la question du niveau de test attendu pour s'assurer d'une qualité produit suffisante, il y a assez rarement une réponse. Et lorsque cette réponse est celle de l'un des développeurs, elle est souvent différente de celle d'un autre membre de la même équipe. Il est pourtant fondamental de connaître les tests minimums requis et les résultats attendus de ces tests.

Il faut néanmoins relativiser : la grande majorité des équipes de développement partagent des stratégies de tests relativement respectées et tous les développeurs ont conscience de la nécessité de faire des tests. La grande souffrance de ces équipes est de ne pas avoir les moyens de réaliser correctement ces tests.

Pour caricaturer, un développeur réalise un code de grande qualité, définit des algorithmes complexes, mais son code, particulièrement optimisé ne fonctionne pas en production parce qu'il n'a jamais pu le tester dans des conditions similaires à cet environnement de production.

En règle générale, la plus grande difficulté des développements est d'obtenir des environnements et des machines de tests dans des délais raisonnables. Au-delà de la difficulté à parfois obtenir un environnement dont les caractéristiques sont le plus proches possible de l'environnement final, ce sont les délais de mise à disposition de ces environnements qui sont problématiques.

Il n'est pas rare de devoir compter les délais en semaines et de devoir faire intervenir sa hiérarchie pour obtenir les outils nécessaires pour faire son travail. Pourtant, la plus grande partie des développeurs se soumettra volontiers à un choix plus restreint d'outils et d'environnements pour peu qu'on leur garantisse l'efficacité des tests et des délais d'approvisionnement raisonnables.

La qualité des environnements provisionnés et les délais de provisioning font donc partie des clés de confiance des développements.

1.2.4 Identifier la clé de confiance du métier

La confiance des métiers est le moteur de leur implication dans le monde de l'IT. A contrario leur défiance tire son origine des décalages permanents de perception entre les deux mondes des équipes informatiques. Il n'est pas rare d'observer des entités métiers préférant travailler avec des sociétés extérieures plutôt qu'avec le service informatique interne.

Pourtant, les métiers sont demandeurs d'un service informatique à la hauteur des meilleurs concurrents du marché. Après tout, qui mieux que les informaticiens de leur société peuvent comprendre les contraintes de leur métier ? Aujourd'hui l'informatique est partout, ils ont donc toute latitude pour voir leur métier dans sa transversalité.

Dès lors que les équipes informatiques font valoir leur expertise, leur réactivité et la qualité de leur collaboration, au moyen d'une démarche DevOps par exemple, les équipes métiers s'impliqueront naturellement. Il est certain que personne ne souhaite s'impliquer dans une coopération avec des services en conflit.

La qualité de la collaboration et l'expertise des équipes informatiques sont donc les clés de confiance des métiers.

1.3 AUTOMATISATION

1.3.1 Une automatisation pour pérenniser la confiance

Pour retrouver de la confiance dans une collaboration parfois difficile, il est important de commencer par travailler ensemble sur les clés de confiance identifiées pour chacune des parties prenantes. En trouvant ensemble un mode de fonctionnement qui vous convient collégialement, vous bâtirez les fondations d'une nouvelle culture de collaboration.

Une bonne solution pour démontrer que cette collaboration retrouvée est efficace et pérenne est d'introduire de l'automatisation. Une collaboration qui fonctionne et

qui est automatisée (au moins en partie dans un premier temps) démontre une volonté et un investissement qui rassurent les métiers.

En effet, une collaboration qui fonctionne ne doit pas être mise en péril si l'équipe s'agrandit ou les personnes sont remplacées. Pourtant, et c'est normal, toute collaboration repose avant tout sur des femmes et des hommes qui s'entendent sur une manière de fonctionner.

La structuration du processus autour de l'automatisation permet donc de créer un cadre dans lequel de nouveaux collaborateurs entrent naturellement. La pérennité de ce cadre de collaboration n'empêche pas de le faire évoluer.

En effet, le temps aidant et la maturité grandissant, il est plus que probable que le cadre de collaboration soit dans l'obligation de s'adapter au changement de son environnement technique comme humain. Le processus doit intégrer un dispositif d'amélioration continue et la chaîne d'automatisation doit le prendre en compte. Le changement doit se faire de manière pragmatique et structurée avec une rigueur de tous les instants.

Pour que la confiance reste à la fois dans les hommes et dans le cadre procédural automatisé, il est important d'éviter l'instabilité et l'incertitude. Ces notions ne sont pas contradictoires avec la prise en compte de l'amélioration continue, si ces principes sont respectés rigoureusement et évitent ainsi le changement à tout va.

C'est à ce prix que la confiance peut naître, s'accroître et se consolider entre toutes les parties prenantes d'une démarche DevOps.

Néanmoins, la consolidation de la confiance n'est pas le seul avantage à introduire une démarche automatisée, elle permet également de gagner du temps et de la qualité.

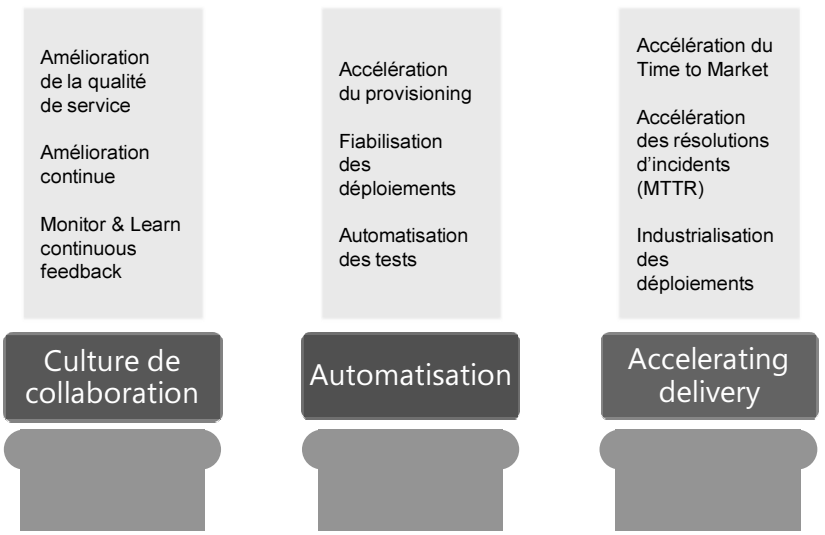


Figure 1.4 — L'automatisation au centre des piliers d'une démarche DevOps

1.3.2 Une automatisation pour gagner en qualité

Automatiser, c'est effectivement gagner du temps : d'abord pour les tâches les plus répétitives à faible valeur ajoutée, mais aussi pour les tâches plus complexes techniquement. Ces mêmes tâches sont également celles qui sont les plus susceptibles d'introduire des erreurs d'inattention mettant à mal la collaboration. Automatiser, c'est donc également assurer un niveau de qualité constant et optimal.

Cette notion peut paraître évidente, mais dans les faits, tout n'est pas si simple. Il n'est pas toujours facile d'identifier les tâches à faible valeur ajoutée, sans oublier que personne n'aime voir son travail dévalorisé. La notion même de valeur ajoutée peut faire l'objet d'un débat.

En réalité, les processus qui devraient naturellement porter la collaboration sont rarement définis d'un commun accord, chacun gardant plutôt jalousement son bout de processus dans son coin pour le faire évoluer à sa guise. Rester dans cet état d'esprit avant d'automatiser ne fera pas gagner en qualité. En réalité, il n'y aurait même aucun bénéfice à automatiser.

Pour gagner en qualité il faut donc partager le processus qui porte la collaboration et établir un consensus sur les tâches les plus critiques candidates à l'automatisation. Le consensus ainsi établi évite le débat sur la valeur de ces tâches, on se concentre sur la valeur apportée par l'automatisation.

En général ce sont les tâches les plus pénibles pour les équipes, soit parce qu'elles sont particulièrement chronophages sans intérêt intellectuel particulier, soit parce qu'elles sont particulièrement complexes et éprouvantes. Dans les deux cas, ce sont les tâches les plus susceptibles de générer des erreurs, et du fait de ces erreurs de générer de la tension entre les partenaires.

Lorsque ces tâches sont identifiées, les automatiser apporte un gain évident et souvent immédiat. L'automatisation peut d'ailleurs se faire à moindre coût si on ne cherche pas à optimiser en même temps que l'on automatise, car, automatiser et optimiser sont deux activités bien distinctes.

1.3.3 Automatiser n'est pas optimiser

Automatiser consiste à rendre systématique à l'aide d'outils technologiques un enchaînement d'activités spécifiques qui n'est pas forcément optimisé ou industrialisé.

Il est pourtant vrai que l'inconscient collectif associe facilement l'automatisation et les automates à l'industrie et donc à une certaine forme d'optimisation. Une démarche DevOps évite ce type de préjugés. Lorsque nous automatisons une collaboration, nous ne nous embarrassons pas de savoir si elle est optimale et industrialisable, nous nous contentons de nous assurer qu'elle fonctionne.

Il est possible et même recommandé pour accélérer et tendre vers du *continuous delivery* de chercher ensuite à supprimer les étapes inutiles et à optimiser la chaîne d'activités.

1.4 CONTINUOUS DELIVERY

Lorsqu'une collaboration fonctionne et qu'elle est au moins en partie automatisée et ainsi pérennisée, il faut entrer dans une phase d'amélioration continue avec pour objectif l'accélération continue des cycles de livraison.

Le *continuous delivery* consiste en une automatisation complète de la chaîne de mise en production autant au niveau de la plateforme technologique que du processus de collaboration associé. Sa mise en œuvre implique donc de ne plus avoir aucune interaction humaine entre la réalisation du service ou du produit et sa mise à disposition finale. Il s'agit d'un objectif ambitieux qui peut volontairement n'être que partiellement atteint. On préfère alors parler *d'accélération du delivery* plutôt que de *continuous delivery*.

Pour y parvenir, la recherche de gaspillages et d'une industrialisation des processus est le premier axe de travail important qui est au cœur de la culture DevOps.

L'autre axe de travail fondamental concerne alors l'extension du périmètre d'automatisation. Le degré de confiance dans les processus automatisés augmentant avec le temps et la maturité des équipes sur le sujet, il devient nécessaire d'envisager en permanence d'automatiser un peu plus ce qui fonctionne déjà afin de tendre vers le *continuous delivery*.

Pour y parvenir de manière agile, ou autrement dit de manière itérative et incrémentale, il faut avoir conscience des différents processus et des différentes problématiques que soulève une démarche DevOps.

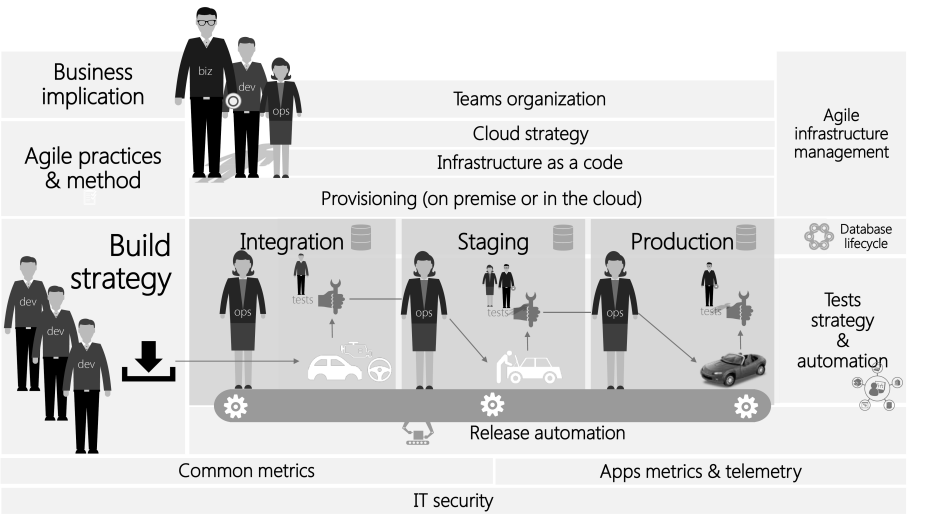


Figure 1.5 — Les principales pratiques d'une démarche DevOps

Savoir traiter et mettre en œuvre chacune de ces problématiques séparément pour en démontrer la valeur rapidement est primordial. Il faut avoir constamment conscience que tous ces processus sont liés entre eux. Il s'agit donc de savoir construire

sa feuille de route agile autour des sujets DevOps en fonction du contexte pour créer un cercle vertueux d'amélioration continue.

1.5 PERSPECTIVES TECHNOLOGIQUES

Même si DevOps est avant tout une transformation de la culture des acteurs et des processus du système d'information, sa dimension technologique n'est pas neutre.

1.5.1 L'intégration de multiples outils

Le succès de sa mise en application ne repose pas sur un seul outil ou sur une seule technologie : le principe est de s'appuyer sur ce que chaque outil fait de mieux. Mais c'est aussi du niveau d'intégration de ces outils que dépend l'optimisation des processus, la coordination de la gestion des livraisons, l'efficacité de la collaboration, la durée des cycles de déploiement ou la capacité à gérer de multiples environnements. En effet, ces outils se fondent sur la manipulation d'objets communs, idéalement, tous issus de la même source, et partagés entre les équipes de développement et d'infrastructure. Ainsi, il ne peut y avoir de divergence de vue vis-à-vis de ces objets entre les développeurs et les opérations.

1.5.2 Le partage de modèles d'infrastructure

Cette démarche est facilitée par la virtualisation des systèmes et par la prise en charge par de multiples outils d'automatisation de provisioning et de configuration. Aujourd'hui, il est possible de réserver et configurer les ressources processeur, réseau et stockage d'une plate-forme complète à partir d'un environnement *bare-metal* ou d'un hyperviseur disposant d'un référentiel d'images virtuelles.

Ce provisioning est réalisé sur la base de modèles d'infrastructure qui vont pouvoir être partagés et répliqués sur les différentes plates-formes (développement, intégration, recette, pré-production, production) qui pourront varier en termes de dimensionnement mais seront identiques en termes de configuration. Ainsi le comportement observé sur la plate-forme d'intégration sera le même que celui constaté sur la production.

1.5.3 Infrastructure as code

L'environnement de déploiement d'une application est donc devenu un livrable du projet. À ce titre, il doit être archivé et lui aussi géré en versions, en tirant parti des pratiques issues du monde du développement (tels que l'utilisation de référentiels partagés avec contrôle de version, l'automatisation des tests...).

Cette nouvelle approche suppose la capacité à gérer directement par des lignes de code le provisioning et la configuration de l'infrastructure dans un langage permettant de les créer et les faire évoluer. Les services d'infrastructure, qu'ils s'exécutent à

demeure ou dans le cloud, doivent donc être dynamiquement modifiables *via* des interfaces de programmation. Cette composante *infrastructure as code* est un élément clé de la dimension technologique de DevOps.

1.5.4 L'instrumentation au cœur du système

Enfin, comme nous l'avons vu précédemment, l'instrumentation d'une solution est essentielle. Il faut pouvoir, en continu, observer le comportement du système sur le plan technique. Ainsi développeurs et équipes de gestion des opérations pourront, au plus tôt, identifier les limites en termes de performance et corriger les dysfonctionnements.

Il faut également, sur le plan fonctionnel, obtenir des informations permettant de répondre aux besoins fluctuants des utilisateurs et valider les hypothèses ayant conduit à définir le cahier des charges du produit.

En résumé

Pour résumer, nous pourrions dire que DevOps va au-delà de ce que laisse supposer son nom, car c'est une démarche de collaboration agile entre le monde des études et du développement (Dev), le monde des opérations, de la production et des infrastructures (Ops) et les représentants des métiers, des utilisateurs et des clients (Business) pour l'ensemble du cycle de vie du service, de sa conception initiale jusqu'à son support en production.

La mise en œuvre s'appuie donc sur trois piliers, à commencer par la collaboration pour construire une relation de confiance, puis sur l'automatisation pour pérenniser la confiance et enfin l'industrialisation pour accélérer et tendre vers le *continuous delivery*.

Enfin, et surtout, DevOps est une transformation culturelle. La dimension technologique n'est là que pour faciliter l'évolution qu'elle sous-tend.

2

DevOps dans la transformation digitale

La transformation digitale est la réponse de chaque entreprise face au défi que représente l'essor des technologies numériques. Concrètement cela suppose la dématérialisation de ses processus, évolution centrée sur une innovation à laquelle participent des concepts et des solutions technologiques tels que l'agile, le cloud, le big data, le *machine learning* et la mobilité.

Cette transformation est efficace si elle s'inscrit dans une démarche globale dont DevOps peut être la clé...

DevOps est agile, c'est un fait, mais il ne suffit pas de l'affirmer, il faut le comprendre.

2.1 DEVOPS EST AGILE

Pour entendre que DevOps est agile, il faut comprendre ce qui se cache derrière le mot *agile*. Aujourd'hui il a été utilisé dans tellement de contextes différents que sa véritable signification s'est diluée dans la conscience générale.

2.1.1 Une définition de l'agile

La dilution et l'incompréhension de l'agilité ont pour conséquence d'affecter son image au sein de l'entreprise et de complexifier son adoption.

Prenons l'exemple d'une direction générale qui décrète que le groupe doit désormais être agile sans plus de précisions. En réalité, cette direction ne sait pas exactement

quelle signification mettre derrière le mot agile. Imaginez maintenant l'écart de compréhension que ce flou va engendrer parmi les directions, les équipes et les salariés.

Finalement, le retour sur investissement semble inexistant pour les principaux intéressés. L'agilité a été perçue comme une notion marketing apparentée à une coquille vide.

Il est vrai que le mot *agile* peut avoir plusieurs sens selon le contexte où il est utilisé :

- Une **organisation** peut être agile quand elle est prête à se remettre en cause en cycles d'amélioration continue et à changer sa hiérarchie et ses règles de management périmètre après périmètre. Le changement n'est plus subi mais perçu comme générateur d'efficacité dans chaque département de l'entreprise. Cette vision de l'agilité est relativement neuve, centrée sur les comportements, les ressources humaines et l'organisation hiérarchique d'une entreprise.
- Le **processus de R&D, l'innovation et le marketing** produit peuvent être agiles. C'est le cas notamment de la conception de nouveaux produits innovants ou des méthodes marketing de recherches de signaux faibles annonciateurs de changement pour l'entreprise. L'iPhone ou l'iPad sont souvent cités en exemple de ces produits conçus de manière agile. Le Lean Startup est une méthode agile qui formalise un certain nombre de concepts qui se prêtent à cette thématique.
- L'**outil informatique** lui-même peut être agile dans sa conception et son architecture. L'apport des démarches d'architecture orientée services (voire microservices) avec la décorrélation complète du moteur d'exécution applicatif et des interfaces utilisateurs contribuent à l'atteinte de cet objectif. La mobilité, les nouveaux devices que ce soient les smartphones, les tablettes ou les objets connectés, tirent directement parti de cette approche.
- Enfin le **processus de gestion** des projets et des hommes autant que le *run* peuvent être agiles. C'est dans ce cadre que l'on exploite pleinement les méthodes de projet agiles ou la démarche DevOps, en s'appuyant sur un processus agile *de bout en bout*, autrement dit *du besoin client à la livraison client*.

On voit bien que la signification du mot *agile* dépend du contexte dans lequel on l'utilise. Mais on comprend également que l'agilité s'appuie sur des valeurs communes qui s'appliquent totalement à une démarche DevOps. Et ces valeurs ne se résument pas au manifeste agile qui est le socle commun de ces méthodes, ce serait bien trop limité. Pour pouvoir s'appliquer à la fois aux développements, à l'infrastructure, à la conception de produits ou encore au marketing, ces valeurs se doivent d'être plus étendues.

Nous sommes dans une démarche itérative, incrémentale et collaborative, nous sommes dans une démarche agile, quels que soient le contexte ou le domaine d'application.

2.1.2 DevOps dans la continuité des méthodes agiles

DevOps respecte de manière tout à fait évidente les valeurs fondamentales de l'agilité, et s'inscrit dans la filiation des *méthodes agiles* traditionnelles.

Lorsque les méthodes agiles traditionnelles sont correctement mises en œuvre, elles génèrent souvent une forme d'enthousiasme auprès des utilisateurs et des populations métiers.

Malheureusement, l'enthousiasme initial de l'utilisateur se transforme vite en frustration quand la coopération des services de développement et de gestion des opérations n'est pas agile et que les différents services informatiques ne collaborent pas ensemble dans la direction attendue par les métiers.

L'idée de DevOps est de pousser les démarches agiles jusqu'aux équipes de production en passant par une collaboration agile basée sur la confiance. DevOps hérite pleinement de la culture agile des méthodologies projets pratiquées depuis des années.

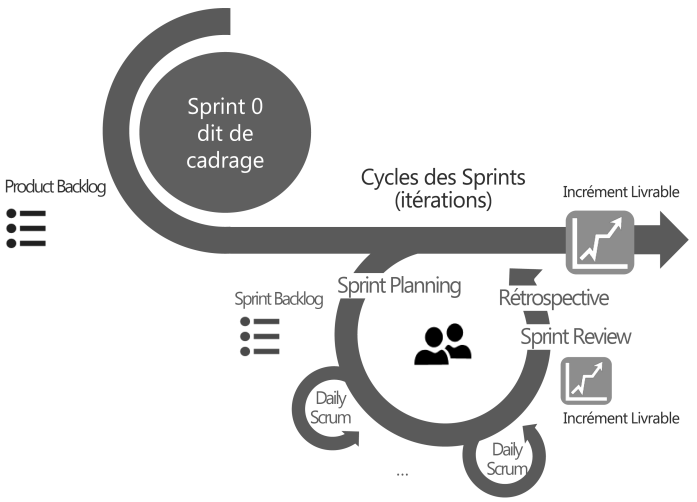


Figure 2.1 — Le cycle de développement agile

DevOps n'est pas agile seulement parce qu'elle respecte les valeurs fondamentales de l'agilité, elle est agile parce que, comme les autres méthodes à succès, elle apporte sa pierre dans l'édifice des bonnes pratiques liées à l'agilité :

- **Scrum est la méthode agile la plus populaire aujourd'hui.** C'est effectivement la démarche la plus séduisante car elle se concentre sur l'aspect managérial des hommes. Sa similitude avec le rugby qui en a inspiré le nom est significative : on parle d'équipe et d'esprit d'équipe avant tout. Les hommes, l'équipe et la responsabilisation des membres de l'équipe sont les maîtres mots.
- **XP Programming est la méthode agile la plus en vogue auprès des ingénieurs.** En effet, elle introduit les pratiques d'ingénierie agile les plus abouties et

bénéficie d'une réputation sans faille concernant la qualité des produits logiciels développés.

- **Agile UP est la méthode agile partageant le plus de concepts avec les méthodes classiques.** Cette méthode est en effet une évolution du *Rational Unified Process* d'IBM puis du *Unified Process* avec Ivar Jacobson notamment pour y introduire les principes de l'agilité. En conséquence, c'est aussi la méthode agile la plus facile à introduire auprès d'équipes évoluant depuis des années avec des méthodes classiques. En d'autres termes, agile UP est la méthode agile la mieux adaptée à la conduite du changement aujourd'hui.
- **Lean est une méthode issue du monde industriel** dont les principes se sont rapidement imposés au sein du monde agile. La recherche de l'industrialisation et de l'optimisation toujours dans une optique de processus orienté client est fondamentale. Les principes de cette méthode sont, comme nous l'avons vu, au cœur de la culture DevOps.
- **Lean Startup est une méthode dérivée du Lean orientée pour les entrepreneurs** autant internes qu'externes à une entreprise**.** Elle vise à rationaliser l'état d'esprit innovant des start-up tout en supprimant les causes d'échecs et de gâchis. Cette approche scientifique orientée produit se combine également très bien avec une démarche DevOps orientée IT.
- **Kanban est une méthode assimilée aux méthodes agiles, bien que non itérative de prime abord.** En réalité, cette méthode combine plusieurs *itérations de cérémonies* parallèles couplées à un *backlog* à flux tendu. Kanban se concentre sur l'industrialisation du processus agile au moins sur son aspect développement.

Il existe d'autres méthodes agiles, mais nous nous sommes limités ici aux plus populaires. Dans toutes ces approches, de nombreux aspects différents de l'agilité sont couverts : le management des hommes (Scrum), la qualité du logiciel (XP), l'adoption du changement (agile UP), l'industrialisation (Kanban) et l'optimisation (Lean) du processus agile.

Cette énumération ne couvre pas un aspect essentiel : la collaboration. C'est le point fort de DevOps qui se concentre sur les aspects de collaboration du processus agile, du besoin client à la livraison au client. DevOps reprend donc les principes de l'agile et les complète.

2.2 DEVOPS ET LE CLOUD

L'attraction qu'exerce le *cloud computing* sur l'industrie des technologies de l'information rend plus impérieuse encore la nécessité pour les développeurs de logiciels et les responsables des opérations de collaborer, pour la création ainsi que l'exploitation d'applications et de services innovants et massivement *scalables*.

2.2.1 Une évolution globale du processus de développement

Le cloud offre aujourd'hui la possibilité de créer et déployer de nouvelles applications et leur infrastructure sous-jacente en quelques minutes. Toutefois de multiples changements sont requis afin d'en tirer le meilleur parti : définition des modèles de nouveaux services, automatisation de leur déploiement, gestion des configurations et des mises à jour, monitoring des performances, remontée d'alertes, et enfin et surtout intégration des processus de développement.

2.2.2 Une évolution du rôle des équipes de gestion des opérations

Tout cela n'est pas sans incidence sur le métier du responsable opérationnel. À l'ère du cloud, celui-ci devra savoir exploiter au mieux les capacités de virtualisation mises à disposition dans les data centers en déléguant totalement les opérations d'infrastructure qui jadis étaient de sa responsabilité.

Dans un premier temps, l'évolution vers le cloud s'est traduite par une séparation des responsabilités entre la gestion du logiciel et des données, de celle de l'infrastructure physique. Grâce à la virtualisation des serveurs, du stockage et du réseau, les systèmes physiques sont devenus totalement découplés des éléments numériques qu'ils hébergent.

Mais aujourd'hui, la machine virtuelle qui, après le serveur, était devenue peu à peu l'élément de référence pour le déploiement, est en train de céder la place à l'application elle-même.

2.2.3 Une évolution du modèle centrée sur l'application

En effet, avec le cloud, il faut revoir la façon dont les applications sont conçues, produites, déployées et mises à jour. Mais comment gérer le cycle de vie complet d'une application, avec la fourniture de ses ressources, la gestion de la configuration, le déploiement, les mises à jour logicielles, la supervision et le contrôle d'accès ?

Cette évolution vers un modèle centré sur l'application a entraîné une accélération de la mise à disposition de mécanismes d'automatisation de provisioning, de configuration, voire de l'orchestration des services proposés par les différents acteurs du cloud. Les plateformes cloud ont donc elles aussi été adaptées pour faciliter l'application de ce nouveau modèle.

Les services génériques proposés répondent aux multiples scénarios applicatifs cibles. Aussi doivent-ils offrir le niveau de configuration et d'extensibilité requis, mais surtout, ils doivent être exposés par une API. En conséquence, l'infrastructure du cloud, à l'instar de l'application, devient dépendante du code. Les descriptions de configuration peuvent donc être fournies aux services cloud *via* des API bien définies. Elles sont directement corrélées aux profils de services IaaS et PaaS que propose le cloud afin de provisionner et configurer les services cibles et leurs différentes ressources connexes. En conséquence, la composante *infrastructure as code* de DevOps, que nous avons déjà évoquée, devient incontournable lorsqu'il s'agit du cloud.

2.2.4 Une évolution des architectures

Avec le cloud, l'infrastructure est de moins en moins sous le contrôle des directions informatiques et la probabilité d'interdépendances de services liés à des SLA différents devient de plus en plus forte. Toute solution d'échelle significative étant sujette au dysfonctionnement, les architectures doivent évoluer vers un modèle *FailSafe* afin de garantir l'évolutivité, la disponibilité et la fiabilité du système. Cette évolution se traduit par de multiples activités au cours de la phase de conception :

- **Découper l'application** en composants fonctionnels en établissant des niveaux de priorité vis-à-vis de la criticité métier et du coût.
- **Définir un modèle** décrivant le comportement de l'application attendu en production (variations de charge planifiées et facilités par les modèles d'autoscaling proposés par le cloud...).
- **Définir un modèle de disponibilité** pour chaque composante de l'application et hiérarchiser les étapes de remédiation selon une classification de modes de défaillance établie en amont.
- **Identifier les points et les modes de défaillance** en adoptant, comme nous l'avons vu précédemment, une démarche d'analyse et de modélisation de la résilience des services de l'application.

Comme on peut le constater, ces éléments sont susceptibles d'impacter aussi bien les développeurs que les équipes de gestion opérationnelle.

2.3 DEVOPS, BIG DATA ET MACHINE LEARNING

Big data et machine learning peuvent également contribuer à la mise en œuvre d'une démarche DevOps. Ces deux notions ne sont pas nécessairement liées mais sont complémentaires.

2.3.1 Les complémentarités entre big data et machine learning

Le big data vise à collecter un grand volume de données (individuellement potentiellement petites) ou des données volumineuses (de l'ordre du péta-octet), ayant un fort potentiel de variabilité ou de vélocité, à les stocker et à les rendre exploitables. Le big data ne cible pas nécessairement l'analyse prédictive des données. Rendre la donnée compréhensible est un but suffisant dans un premier temps.

Le *machine learning* vise à exploiter des données pour les analyser avec une approche scientifique et statistique. Il est important de noter que le *machine learning* est une discipline d'intelligence artificielle et en ce sens le but de l'analyse est clairement de faire des prévisions pour pouvoir prendre des décisions. Pour faire ces prévisions, il n'est pas toujours nécessaire de faire du big data, autrement dit de disposer d'une quantité astronomique de données.

En 2012, un statisticien nommé Nate Silver a pu prédire avec succès les résultats de l'élection américaine dans tous les états en partant de données pesant à peine 200 ko. Autre exemple, en 2014 la société française Data Publica propose un découpage pour la réforme des régions basé sur un algorithme intelligent utilisant uniquement des données fournies par l'INSEE pesant à peine 1,7 Mo. Ces exemples mettent en lumière une notion fondamentale que tout praticien de business intelligence se doit de connaître : la pertinence des données.

2.3.2 L'apport du big data et du machine learning à DevOps

Une démarche DevOps est tributaire de la collecte de données, aussi bien concernant l'usage par l'utilisateur du produit (télémétrie) que dans l'aspect opérationnel de la démarche (monitoring). En ce sens et de par son aspect agile, l'application rigoureuse des principes DevOps permet de limiter la collecte uniquement aux données pertinentes. L'application des notions d'expérimentation et d'apprentissage permet de continuellement remettre en cause ce choix des données collectées, la pertinence pouvant évoluer au cours du temps.

La collecte de données pertinentes ne préjuge en rien de la quantité de données à collecter et en cela les principes du big data s'appliquent pleinement pour réussir à exploiter ces données. Un exemple courant est l'exploitation issue des logs des différents outils de la chaîne de déploiement automatisée DevOps.

Mais l'aspect le plus intéressant reste le lien possible entre les données collectées sur l'usage fait par l'utilisateur du produit (télémétrie) et les modèles de machine learning. En effet, les modèles prédictifs de plus en plus évolués liés à cette discipline pourraient anticiper demain les futurs besoins des utilisateurs en se basant sur leurs usages actuels. Un outil puissant pour inventer les produits de demain…

La mise en œuvre de solutions liées au big data et la définition des modèles de machine learning adaptés au cycle de développement logiciel viennent enrichir DevOps qui bénéficie ainsi de l'expérimentation et du traitement des données les plus pertinentes.

2.4 DEVOPS ET MOBILITÉ

2.4.1 Un monde en constante évolution

Si les grandes tendances de la transformation digitale s'apparentaient à une famille, DevOps et la mobilité seraient certainement frère et sœur tant cette démarche est adaptée au monde de la mobilité. En effet, aujourd'hui le monde de la mobilité est constitué d'un modèle applicatif distribué *via* des magasins applicatifs : de l'iPhone à Windows en passant par Android et Mac OS X, tous les systèmes possèdent désormais leurs *stores* qui deviennent de plus en plus le lieu privilégié par les utilisateurs pour s'approvisionner en applications.

Ce modèle a cela de particulier qu'il est réellement hyperconcurrentiel. Les éditeurs rivalisent de créativité pour répondre aux besoins des utilisateurs et il n'est pas rare de trouver une dizaine d'applications différentes pour répondre à une de nos attentes.

Face à cette concurrence exacerbée, on pourrait croire que les applications font la course aux fonctionnalités. C'est peut-être parfois le cas, mais la clé de la réussite est en réalité l'adéquation entre la fonctionnalité offerte et notre besoin réel. Et si une chose est bien constante, c'est le changement : nos besoins évoluent régulièrement et rapidement. L'application qui évoluera avec nous a le plus de chance de rencontrer le succès car elle aura un avantage concurrentiel constant sur son marché.

Il n'existe pas aujourd'hui des centaines de façon de faire pour réussir ce challenge imposé par les utilisateurs : seule l'agilité de bout en bout mise en œuvre avec une démarche DevOps permet de tenir ce rythme effréné.

Mais la mise à jour et l'enrichissement continus des applications ne sont pas le seul atout d'une démarche DevOps dans le monde la mobilité. Il existe un autre paradoxe dont nous nous accommodons sans problème aujourd'hui et qui paraîtra bien incompréhensible demain : les applications mono-plateforme.

2.4.2 Un monde multiplate-forme

Très concrètement aujourd'hui, lorsque vous voulez utiliser votre application préférée, vous devez la télécharger trois fois en moyenne : une fois sur votre ordinateur de bureau, une fois sur votre tablette et une dernière fois sur votre smartphone. Quand cette application est bien conçue, il est possible de continuer une tâche commencée sur un autre *device* depuis n'importe lequel d'entre eux mais le plus souvent ce n'est pas le cas.

Les *applications universelles* et les *environnements de travail multistations* vont devenir la norme du monde informatique. L'agilité des architectures orientées services, voire microservices, vont s'imposer par nécessité. L'avènement de ce parc applicatif moderne ne fera que renforcer nos attentes en fonctionnalités utiles et évolutives. Nous entrerons alors réellement dans l'ère de la mobilité avec toutes ses caractéristiques espérées depuis longtemps.

Les méthodes traditionnelles de gestion de projet issues de l'analogie avec le monde du BTP seront complètement obsolètes. La démarche DevOps apporte une grande partie de la solution dans ses principes mais il est absolument certain que sa mise en œuvre évoluera grandement en même temps que la maturité du marché.

2.4.3 Un monde sans frontière traditionnelle

À cela on peut objecter que cette analyse s'applique seulement au monde du grand public et que le monde professionnel sera globalement préservé de ces bouleversements à venir. S'il est vrai que le rythme d'adaptation du monde professionnel, notamment en ce qui concerne le parc applicatif interne et les progiciels, est souvent plus lent, ce serait une erreur de penser que les impacts de ces changements majeurs seront minimes pour l'entreprise.

Il y a au moins deux raisons de penser que les professionnels seront en première ligne du changement.

Tout d'abord, il est déraisonnable de croire que les murs de l'entreprise sont imperméables à une culture grand public. De la même manière que nous avons vu le phénomène BYOD (*Bring Your Own Device*) en entreprise, les utilisateurs vont devenir de plus en plus exigeants vis-à-vis des applications professionnelles. Celles-ci, y compris les progiciels des grands éditeurs, vont devoir répondre à des attentes à la fois d'ergonomie et de réactivité qui deviendront la norme dans le grand public.

Ensuite, il ne faut pas oublier que refuser ostensiblement ces avancées laisse une image d'archaïsme au sein de sa force de travail mais également par viralité en dehors. Au-delà de l'impact sur les ressources humaines dont les talents sont de plus en plus disputés aujourd'hui, cela n'est pas de nature à créer un avantage concurrentiel sur son marché. Or, l'ergonomie et le design alliés à la réactivité sont les principaux prérequis à développer pour espérer acquérir un avantage concurrentiel sur son marché.

Pour conclure, il est certain que les exigences d'un monde mobile poussent tous les marchés dans tous les secteurs à adopter un jour une démarche DevOps. Il faut néanmoins être pragmatique et réaliste : toutes les démarches DevOps ne se ressembleront pas, elles dépendront du marché, du contexte, de la taille de l'entreprise, de sa culture et de ses enjeux. Il paraît évident que certains secteurs mettront plus de temps que d'autres à percevoir la nécessité de cette transformation.

Pour certaines entreprises, cette adoption se fera à marche forcée ; pour d'autres ce sera plus en douceur. Mais il paraît évident que pour survivre toutes passeront par là. On peut faire l'analogie avec l'agilité ou d'autres sujets de transformation digitale. On l'a vu, DevOps est complémentaire mais également indispensable au succès de ces démarches.

2.5 DEVOPS, INNOVATION ET DESIGN THINKING

2.5.1 Les principes du *design thinking* et des méthodes d'innovation

Initialement et principalement destiné aux designers, le *design thinking* est une méthodologie de créativité inventée au milieu du siècle dernier. Cette méthode préconise de situer en amont des projets pour identifier les besoins des utilisateurs, voire les anticiper, en s'appuyant sur trois à sept étapes selon les versions autour desquelles il faut itérer, sans forcément suivre un ordre préétabli.

Les principales étapes de cette méthodologie sont :

- **Empathy.** Au cours de cette étape, l'objectif est de ressentir l'audience cible du service, pressentir ses plaisirs et ses frustrations pour mieux percevoir ces désirs. Dans les faits, une rencontre et des interviews sont très utilisées.
- **Define.** Lors de cette étape, il faut définir et affiner ses objectifs en fonction de la perception empathique que l'on a acquis. Les enjeux sont liés aux objectifs et groupés par audiences pour toujours être au plus proche à la fois du besoin et du désir de son audience cible.

- **Ideate**. C'est la phase de divergence de la méthode. Elle cherche à trouver des idées créatives, simples et intelligentes pour répondre aux besoins et aux désirs de l'audience cible. Cette phase itère de manière très régulière avec la suivante pour également diverger sur les contraintes éventuelles qui peuvent être détectées dans les phases suivantes.
- **Prototype.** C'est la phase de convergence de la méthode. L'objectif est ici de construire des solutions démontrables à partir des idées retenues lors de la phase précédente, de relever les difficultés ou contraintes qu'elles peuvent représenter et de définir les solutions appropriées pour pouvoir ensuite les tester.
- **Test.** Cette étape est celle de l'expérimentation auprès de l'audience cible, de l'apprentissage et de la rétrospective. Elle permet de confronter à la réalité l'ensemble des hypothèses retenues et éventuellement de *pivoter* pour reprendre un terme cher au Lean Startup dont la philosophie est très proche.

Si l'ensemble des méthodes d'innovation ne suivent pas nécessairement ces étapes, elles se retrouvent toutes autour des principes de divergence et de convergence centrées sur l'utilisateur. Nous pouvons facilement transposer les principes du *design thinking* sur la grande majorité des démarches d'innovation connues à ce jour.

2.5.2 L'innovation est agile

Les méthodes d'innovation sont itératives et collaboratives par nature. Il paraîtrait absolument contre-productif, pour ne pas dire impossible, de procéder différemment pour réussir à aboutir à des solutions viables. Or les principes d'itération et de collaboration sont deux des trois principes fondamentaux de l'agilité.

Pour être complètement dans les valeurs de l'agilité, il est important d'avancer par incrément. Les méthodes d'innovation ne sont pas réfractaires à l'incrémentation, d'autant plus que la complexité des produits et services inventés de nos jours rend quasiment indispensable une avancée par petit pas.

En dehors des innovations avec un périmètre relativement modeste, la grande majorité des inventions résultantes des méthodes d'innovation sont itératives, incrémentales et collaboratives, et sont donc des méthodes agiles à part entière.

Ainsi, il n'est pas surprenant de voir par exemple le Lean Startup reprendre une grande partie des principes des méthodes d'innovation étant elle-même à la croisée des chemins avec les méthodes agiles de gestion de projet plus classique.

Et si les méthodes d'innovation sont agiles, elles sont dès lors d'autant plus complémentaires avec DevOps.

2.5.3 Complémentarité avec DevOps

Si les méthodes d'innovation et DevOps sont agiles, elles n'ont pas les mêmes objectifs. Les premières visent à inventer des usages, des services et des produits. DevOps aide à les réaliser et à les mettre en œuvre rapidement et efficacement. Cependant, l'innovation n'étant jamais statique et le *design thinking* préconisant

d'itérer en permanence, les solutions mises en œuvre ne peuvent également rester statiques et doivent en permanence s'adapter. Mieux encore, DevOps doit permettre de mieux entendre l'utilisateur pour alimenter de manière efficace le processus de création et d'innovation continue.

La complémentarité devient évidente. Les méthodes d'innovation alimentent les solutions à réaliser selon une démarche agile, et DevOps optimise sa concrétisation en réduisant au maximum les temps de mise en production et en maximisant ainsi les résultats de l'expérimentation.

Le feedback continu, inhérent aux méthodes DevOps et à ses mécanismes de collaboration, permet d'alimenter en permanence le processus de création des méthodes d'innovation afin de l'optimiser.

En résumé

Le digital est aujourd'hui au cœur de la production de valeur de toutes les entreprises. En ce sens, nous pouvons dire que toutes les entreprises aujourd'hui sont des entreprises digitales quel que soit le secteur d'activité ou la chaîne de valeur.

La transformation digitale vise à rendre plus agile et plus collaboratif leur informatique et leur technologie tout en étant en capacité permanente d'innover rapidement. C'est exactement l'objet d'une approche DevOps au travers d'une démarche de collaboration optimisée.

Selon une étude IDC datant de novembre 2015 et portant sur plus de 200 entreprises françaises, dont plus de la moitié a plus de 1 000 salariés, 52 % des entreprises considèrent DevOps comme le principal levier de leur transformation numérique.

Selon la même étude, les entreprises ayant mis en place une démarche DevOps bénéficient d'une nette amélioration de la collaboration entre les équipes, y compris les équipes métiers, d'une amélioration de la satisfaction client, d'un gain en qualité important, d'une vraie accélération des déploiements logiciels et d'une baisse du ratio dépenses/valeur.

DevOps est un élément clé de la transformation numérique des entreprises car complémentaire de concepts et de solutions qui accélèrent leur capacité de réponse à l'essor des technologies numériques.

DevOps respecte et étend les pratiques des méthodes agiles en s'appuyant sur ces trois grands principes : collaboration, itération et incrémentation. DevOps les met en application jusqu'au monde du support et de l'infrastructure et de la production.

Enfin, nous pouvons compléter en nous rappelant que DevOps est un accélérateur d'adoption d'un certain nombre de grands concepts d'aujourd'hui : le cloud, la mobilité, le machine learning ou encore l'innovation.

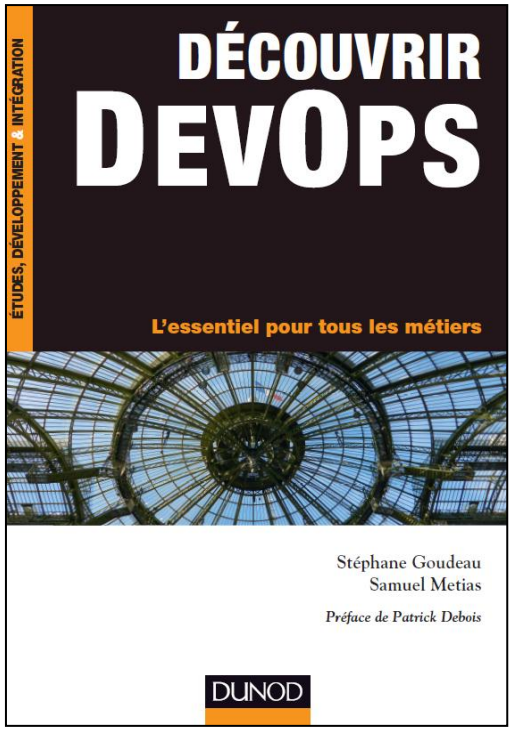
L’indispensable DevOps en français !

Disponible à partir du 16 mars en librairie et sur Dunod.com

Commandez-le sur dunod.com

Acheter le livre

24.90€



Samuel Metias
Stéphane Goudeau

2 experts Microsoft au service de votre entreprise

INFOPRO

Stéphane Goudeau, Samuel Metias

Préface de Patrick Debois

DÉCOUVRIR DEVOPS

L'essentiel pour tous les métiers

MANAGEMENT DES SYSTÈMES D'INFORMATION

APPLICATIONS MÉTIERS

ÉTUDES, DÉVELOPPEMENT, INTÉGRATION

EXPLOITATION ET ADMINISTRATION

RÉSEAUX & TÉLÉCOMS

Ce livre s'adresse à tous ceux qui s'intéressent aux systèmes d'informations innovants, à tous les passionnés d'informatique qui pensent que l'organisation est aussi importante que la technique pour réussir, ainsi qu'aux familiers de la notion d'agilité dans le monde de l'informatique.

DevOps est une démarche qui permet aux équipes de développement et d'infrastructure de collaborer plus efficacement face à ces nouvelles exigences du mode logiciel. À l'ère du **continuous delivery** et du **cloud** DevOps s'inscrit dans le prolongement des **méthodes agiles** et s'inspire d'autres expériences telles que **Lean Startup**, **Scrum**…

L'originalité de ce livre est d'aborder le sujet sous différents points de vue pour répondre au mieux aux interrogations et problématiques pratiques de tous les métiers concernés qu'il s'agisse des développeurs, des opérationnels, mais aussi du management de la DSI et des acteurs métiers.

Cet ouvrage offre ainsi une **vision à 360°** de la démarche DevOps. Il a été rédigé de manière pédagogique et concrète pour vous donner toutes les informations dont vous avez besoin pour entreprendre une démarche DevOps dans votre organisation.

STÉPHANE GOUDEAU

est architecte dans la division Développeurs Expérience DX de Microsoft France. Précurseur sur la plateforme Microsoft Azure, il est adepte de la philosophie DevOps depuis maintenant plusieurs années.

SAMUEL METIAS

anime la communauté DevOps de l'ensemble de la filiale française de Microsoft et, depuis novembre 2015, il est responsable de l'alignement des offres DevOps de Microsoft Services à travers le monde.

9 782100 745876

6782837

ISBN 978-2-10-074587-6

Les actus du savoir

DUNOD

dunod.com