L'avenir de DevOps :
ce que vous devez savoir

ca technologies | Automic™

Résumé

Cet e-book contient cinq articles qui expliquent la formation du DevOps et proposent une vision de l'avenir de cette méthodologie de construction d'applications, désormais devenue indispensable.

L'histoire du DevOps

Comment le DevOps a évolué d'un simple hashtag pour devenir une filière complexe, représentant plusieurs milliards de dollars, pour les développeurs de logiciels. Découvrez les principaux acteurs à l'origine de DevOps, ainsi que l'accent mis sur l'importance de Git dans les premières phases de développement de la philosophie DevOps.

Release Automation: les tenants et les aboutissants de votre transformation numérique

Il existe de nombreuses théories divergentes concernant les conditions nécessaires au démarrage de votre projet DevOps ; toutefois, très peu proposent des plans exploitables. Ce chapitre explique de quelle manière la nature d'Application Release Automation promeut les principes fondamentaux du DevOps et définit une voie pouvant stimuler un changement radical.

Y a-t-il vraiment une composante opérationnelle dans l'équation DevOps?

Saviez-vous que lorsqu'Instagram a été vendue à Facebook pour un milliard de dollars, l'entreprise n'employait aucun spécialiste des opérations informatiques ? Les compétences opérationnelles seront toujours indispensables dans le cadre du cycle de vie des applications, et ce chapitre étudie le rôle changeant des opérations dans DevOps, ainsi que son incidence sur les équipes.

Ressentiment envers les machines : le problème des outils DevOps

De nouveaux outils DevOps sont lancés à un rythme effréné, et chaque équipe DevOps aspire à satisfaire des exigences différentes. Le résultat est que chaque système DevOps, même au sein d'une même organisation, offre une combinaison d'outils unique. En l'absence de « système » global pour contrôler et coordonner les outils, DevOps manque de structure et d'uniformité.

Faire du DevOps une réalité au sein des organisations « historiques »

Ce chapitre dissipe la mauvaise réputation des applications « historiques » et propose une manière de les intégrer aux technologies de pointe sur lesquelles repose DevOps. Par exemple, comment intégrer Siebel à un processus de Continuous Delivery ? Et comment LinkedIn a-t-il introduit la technologie des conteneurs dans ses systèmes historiques ?

L'impact des microservices sur les systèmes informatiques des entreprises

Au lieu de construire intégralement chaque application, les microservices permettent de concevoir des applications composites à partir de composants prêts à l'emploi, fournissant certaines fonctions spécialisées. Cette approche pourrait se traduire par un nouveau bond en avant en termes de vitesse de déploiement, et pourrait bien incarner l'avenir du DevOps.

TABLE DES MATIÈRES

1re partie : Historique de DevOps

La naissance du concept **6**

Le mouvement Agile **8**

L'importance de Git **9**

2e partie : Application Release Automation: la solution et la finalité de votre transformation numérique

La Grande Charte de la transformation numérique **10**

Où commencer **11**

Un modus operandi **12**

3e partie : Y a-t-il vraiment une composante opérationnelle dans l'équation DevOps?

Pas d'Ops? **15**

Le DevOps en pratique **16**

L'opportunité de la croissance **17**

4e partie : La peste soit des machines : le problème des outils DevOps

Un terrain de jeu inégal **18**

La guerre des machines **19**

Le calme après la tempête **20**

5e partie : Faire du DevOps une réalité au sein des organisations « historiques »

LinkedIn: liée à ses systèmes historiques **23**

Modernisation réaliste **24**

6e partie : L'impact des microservices sur les systèmes informatiques des entreprises

L'industrialisation de l'informatique d'entreprise **27**

L'avantage de la spécialisation **28**

Un complément d'agilité **28**

Devez-vous opter pour les microservices? **29**

1re partie : Historique de DevOps

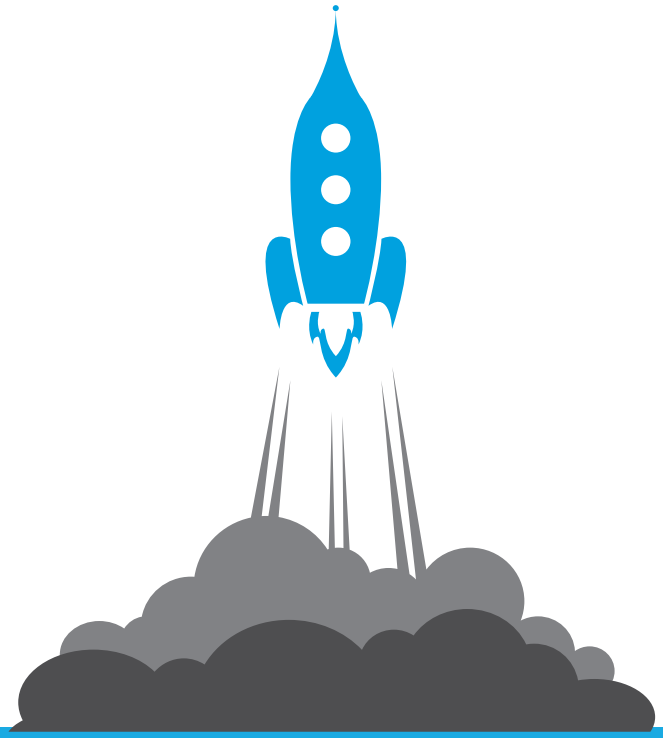
Debois assistait à la conférence dans le but de discuter de l'introduction de pratiques Agiles, telles que les mêlées, dans l'approche Ops.

Il est très pertinent que le terme « DevOps » soit né et documenté sur Twitter: sa nature collaborative, qui réunit des individus issus de différents horizons et se comporte comme un référentiel de connaissances combinées, la vérification automatique de toutes les interactions, etc. Tout comme une mise à jour itérative et rapide du logiciel Twitter n'est pas réalisable sans le DevOps, le mouvement DevOps n'aurait pas pu progresser à une telle vitesse sans une plate-forme telle que Twitter.

La naissance du concept

Patrick Debois était consultant informatique ; il aimait étudier les organisations informatiques sous toutes leurs coutures, et a donc choisi de travailler à cette fin. Alors qu'il testait un projet, il a directement pu constater les différences de culture entre le Développement et les Opérations : le développement Agile et le déploiement ITIL.

En 2008, lors de la conférence Agile Development au Canada, Debois a été le seul à se présenter à une session intitulée « Infrastructure Agile ». Même le présentateur Andrew Shafer ne s'est pas rendu à cette session, supposant qu'elle n'intéresserait personne. Debois a retrouvé Shafer, et les deux professionnels ont parlé de leurs réflexions communes ; Debois assistait à la conférence dans le but de discuter de l'introduction de pratiques Agiles, telles que les réunions Scrum, dans l'approche Ops.



En juin 2009, Flickr a organisé une conférence intitulée « **10+ deploys per day: Dev and Ops Cooperation at Flickr** » (10 déploiements et plus chaque jour : la coopération entre Dev et Ops chez Flickr). Pendant la diffusion de la conférence en Belgique, sur Twitter, Debois a déploré de ne pouvoir y assister ; sur le ton de la plaisanterie, quelqu'un a répondu qu'il devrait lancer sa propre conférence en Belgique. C'est exactement ce qu'a fait Debois, suscitant une demande sur Twitter; il a opté pour le nom « devopsdays ». Le hashtag **#DevOps** est resté.

Debois n'a pas donné naissance au DevOps. Il n'a fait que l'adopter et en étudier le comportement. L'histoire de la naissance de ce nom en Belgique est simplement celle de la mise en lumière d'une philosophie en développement depuis plusieurs années.

Le tournant du 21e siècle a marqué le début d'une révolution dans le domaine du génie logiciel. Sous une forme ou une autre, des entreprises telles que Flickr, Google et Amazon ont mis en œuvre DevOps au début des années 2000. Ce n'est pas un hasard si ces sociétés figurent aujourd'hui parmi les entreprises les plus célèbres et performantes ; un groupe de leaders devançant constamment leur époque et leurs concurrents. Plus d'une décennie plus tard, des entreprises moins célèbres essaient encore de reproduire leurs modèles DevOps.

Avant la naissance du surnom « DevOps », il y avait le Manifeste agile (« Manifesto for Agile Software Development »), rédigé en 2001 – un postulat court et précis :

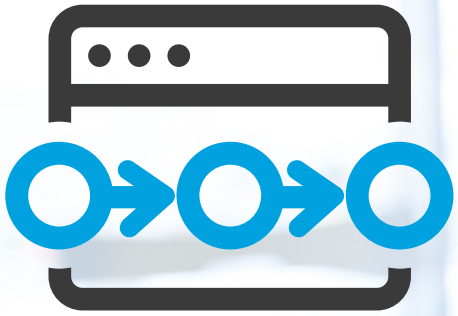
Le mouvement Agile

« Nous découvrons de meilleures approches pour faire du développement logiciel, en en faisant nous-mêmes et en aidant les autres à en faire. Grâce à ce travail, nous en sommes arrivés à favoriser :

Les individus et leurs interactions plus que les processus et les outils ; un logiciel qui fonctionne plus qu'une documentation exhaustive; la collaboration avec les clients plus que la négociation contractuelle ; l'adaptation au changement plus que le suivi d'un plan.

Cela signifie que bien qu'il y ait de la valeur dans les éléments situés à droite, notre préférence se porte sur les éléments qui se trouvent sur la gauche. »

L'importance de Git



Dès lors, ces valeurs ont commencé à se manifester dans la pratique et dans la conception d'outils. Un excellent exemple est celui des systèmes décentralisés de contrôle des versions, tels que Git. Au lieu d'imposer aux développeurs de travailler avec un référentiel centralisé, en déléguant le contrôle central de l'ensemble d'un projet, Git a démocratisé le développement de logiciels en permettant à des équipes de développement et à des développeurs individuels d'avoir accès à des copies de travail complètes de leur logiciel, qu'ils peuvent intégralement tester et déployer.

Cette distribution leur permet de travailler sans être connectés en réseau, lorsqu'ils sont en déplacement, etc. Ils peuvent également réaliser des modifications directement sur leur disque dur, sans être tributaires d'un créneau horaire d'accès à la version centrale. Avec Git, les développeurs peuvent simplement intégrer des ensembles complets de modifications à un référentiel à leur convenance, ou à la fin d'un mini-projet.

Git satisfait à de nombreux principes de la méthode Agile, tels que la possibilité de travailler n'importe où, d'intégrer des modifications à la convenance des développeurs et de travailler de manière autonome ; mais il satisfait surtout aux principes « Les individus et leurs interactions plus que les processus et les outils » et « L'adaptation au changement plus que le suivi d'un plan ». Un référentiel central peut être mis en œuvre, si le projet s'y prête et les membres de l'équipe souhaitent le faire, mais il n'est pas nécessaire.

Ces principes sont désormais devenus la norme et aujourd'hui, le DevOps est l'aboutissement de ces principes. Les **capacités des équipes DevOps par rapport aux équipes qui n'ont pas franchi le cap** sont incroyables ; cependant, le travail requis pour s'affranchir des méthodes établies, ainsi que l'investissement en termes de temps et d'argent, peuvent encore être dissuasifs pour certaines entreprises. **L'organisation en cascade du génie logiciel reste appropriée pour certaines applications**, mais les entreprises numériques désireuses d'accélérer le rythme de leurs publications n'auront bientôt plus que deux choix : adopter le DevOps ou disparaître.

Ron Gidron

Ron Gidron est directeur marketing produit de Release Automation chez Automic Software. Au cours des 14 dernières années, il a occupé différentes fonctions dans la vente, le marketing produit, l'avant-vente aussi bien dans des startups qu'au sein de grandes entreprises. Ron est passionné par l'interaction des logiciels, des utilisateurs et des tendances du marché.

2e partie : Application Release Automation : les tenants et les aboutissants de votre transformation numérique

C'est l'automatisation qui est l'aboutissement des efforts collaboratifs de la méthodologie DevOps, et non l'inverse.

Si le DevOps repose sur une évolution culturelle vers la collaboration, l'Application Release Automation (ARA) est l'unique manière de mettre en œuvre cette approche aux cadences imposées par le marché numérique actuel. Le choix est maintenant simple : adopter DevOps ou disparaître. Le Release Automation et DevOps forment une recette parfaite pour améliorer l'agilité, mais au sein des entreprises, la hiérarchie est souvent perplexe.

Les augmentations phénoménales de la rapidité de déploiement constatées chaque année dans le rapport « State of DevOps » par rapport aux méthodes traditionnelles de génie logiciel ne sont pas simplement le fruit d'un apéritif et de quelques accolades confraternelles entre les équipes de développement et de production . Seuls les ordinateurs peuvent atteindre de telles vitesses, et DevOps s'appuie sur le Release Automation pour réaliser cette ambition.

La Grande Charte de la transformation numérique

Il s'agit de notre Grande Charte. Des traitements automatisés, courts et reproductibles, autorisant la réalisation de modifications incrémentielles, génèrent l'agilité professionnelle qui a permis au DevOps de s'imposer ; toutefois, ils sont également indispensables à la survie des entreprises sur le marché numérique qu'est aujourd'hui devenue cette filière.

Il existe deux façons d'atteindre cet objectif : la première consiste à essayer d'imposer tous les idéaux dont on entend parler dans les conférences et les blogs consacrés au DevOps ; généralement, il s'agit de messages flous, décrivant ce que le DevOps n'est pas (très utile !) ou comment cultiver l'insaisissable culture DevOps qui fait de nous tous les meilleurs amis du monde, du jour au lendemain.

Où commencer

La deuxième manière est plus réaliste. La plupart des entreprises doivent lancer une initiative DevOps en plus de gérer leur immense charge de travail habituelle. Elles doivent disposer d'un plan d'action pour tirer profit de leur investissement dans l'initiative, mais également mettre en place des indicateurs de mesure pour justifier cet investissement vis-à-vis de l'entreprise.

De nombreux concepts DevOps semblent s'appliquer à des entreprises disposant d'une quantité infinie de temps et de ressources, qui peuvent mettre toutes leurs autres tâches en attente. Ces idées sont viables en théorie, mais à quel plan peut se fier une entreprise normale, dans la pratique ? Ou comme me l'a un jour demandé un dirigeant informatique, « Que puis-je faire demain ? ».

L'automatisation peut être le point de départ, ainsi que l'objectif. Les outils d'automatisation du déploiement applicatif nécessitent de faire évoluer les méthodes de travail vers la collaboration, grâce à un outil tangible : un plan. Celui-ci définit les rôles des individus, ainsi que leurs relations.

La première étape consiste à décomposer le cycle de vie de l'application et à en éliminer les processus manuels et de création de scripts, puis à l'automatiser autant que possible. L'étape suivante consiste à automatiser les déploiements, en créant un processus de déploiement automatisé.

Ce processus est réutilisable et constitue le point central du mouvement DevOps au sein de l'entreprise. À mesure que chaque équipe optimise son rôle, le « conduit » se resserre, et sa rapidité et son débit augmentent. Le Release Automation incite les membres de l'équipe à s'informer sur les opérations qui ont lieu avant et après leur section du conduit, ce qui promeut la diffusion du savoir au sein du service.

Le Release Automation fournit un plan d'action, ainsi qu'une plate-forme permettant de promouvoir l'évolution de cette philosophie.

Un modus operandi

Les connaissances approfondies des meilleurs cerveaux de votre service informatique se développent à l'intérieur du processus, qui devient un référentiel capable de survivre même si ces experts quittent l'entreprise. Ceci crée un modus operandi pour l'entreprise – une identité qui se propage et peut être développée à mesure que les mentalités et les méthodologies plus subtiles du DevOps évoluent dans le temps. Le Release Automation fournit un plan d'action, ainsi qu'une plate-forme permettant de promouvoir l'évolution de cette philosophie.

Bien qu'ARA promeuve la collaboration, un produit dédié, indépendant des outils utilisés, permet à chaque membre de l'équipe DevOps de continuer à utiliser ses outils préférés – et surtout, ceux avec lesquels il travaille le plus rapidement. L'enfermement propriétaire peut être un problème avec les fournisseurs qui contraignent les consommateurs à utiliser leurs outils pour travailler avec leur propre solution ARA. Les meilleurs fournisseurs proposent une plate-forme universelle et des intégrations à une vaste gamme d'outils DevOps, laissant ainsi à leurs clients la liberté de choisir, tout en minimisant les perturbations liées à l'adoption de la technologie ARA.

Dire qu'ARA et DevOps vont bien ensemble est un euphémisme. ARA n'est pas seulement indispensable au bon fonctionnement du DevOps ; elle stimule également considérablement sa fertilité. ARA est un plan d'action que vous pouvez exploiter dès demain pour promouvoir rapidement l'agilité au sein de votre entreprise. Et sans elle, le DevOps n'est qu'un club social pour vos équipes informatiques.

Scott Willson

Scott Willson est le directeur marketing produits pour Release Automation chez Automic Software, avec à son actif plus de 20 années d'expérience des technologies couvrant le développement de logiciels, l'avant-vente, l'après-vente et le marketing. Scott est passionné par la technologie et la manière dont elle peut aider les entreprises à créer de la valeur ; il dirigeait des organisations DevOps avant même que l'expression DevOps ne soit inventée.

3e partie : Y a-t-il vraiment une composante opérationnelle dans l'équation DevOps?

Bien que la créativité ne soit rien sans la fiabilité, la conviction d'avoir besoin d'une équipe de production spécialisée est souvent erronée.

On aura toujours besoin d'individus ayant appris à déployer du code tout en privilégiant la simplicité. Lorsqu'Instagram a été vendue à Facebook pour un milliard de dollars, l'entreprise ne comptait pas le moindre spécialiste de production.

Les déploiements peuvent souvent être réalisés par des développeurs passionnés par le comportement de leur code en production. En théorie, cela devrait concerner tous les développeurs, de la même manière que les équipes de production doivent pouvoir comprendre le processus de création de code. Si cette théorie n'incarne pas le DevOps en soi, elle est représentative de l'intérêt porté par les professionnels au vaste écosystème auquel s'intègrent leurs tâches spécialisées ; en soi, rien de nouveau.

Pas d'Ops?

La spécialité d'Instagram consiste principalement à publier des photos en ligne. La composante opérationnelle n'est donc pas aussi hétérogène que dans de nombreuses autres entreprises. Instagram était une jeune entreprise native du Web, dénuée de la complexité des interactions entre les systèmes historiques et les applications numériques. Souvent, une jeune entreprise débute avec quelques développeurs ouverts d'esprits, prêts à « se salir les mains » en réalisant le travail des équipes de production. Il est acceptable de tâtonner de cette manière, à condition de n'avoir que quelques clients et de pouvoir se permettre d'être brièvement hors ligne sans risque de voir plonger sa rentabilité.

Certains secteurs d'activité sont différents. Par exemple, les entreprises financières ne peuvent commettre aucune erreur, et exigent donc une fiabilité et une sécurité extrêmes. La présence d'équipes de production spécialisées est alors indispensable, car des réclamations ou des temps d'arrêt pourraient terrasser ces organisations. Toutefois, la plupart des entreprises ne sont pas aussi exposées et peuvent adopter une approche plus agile des opérations.

Pendant la phase de croissance de l'entreprise, les développeurs ayant à cœur le déploiement sont progressivement submergés par des tâches de production, au risque de gagner en frustation et de perdre en motivation. Dans ce cas précis, peut-être qu'un expert en production est nécessaire, mais ses compétences et sa manière de s'intégrer seront essentielles. Dans un contexte de forte croissance, les équipes de production doivent garantir la fiabilité dans des environnements en mouvement.

Le DevOps en pratique

Si un spécialiste de la production est mobilisé et les autres employés se contentent de lui confier toutes les tâches de déploiement, cela crée exactement le type de silos que le DevOps aspire à éliminer. Il est beaucoup plus simple d'éviter la création de silos en premier lieu. Ce sont les fondamentaux du DevOps – pas une révolution indistincte, initiée par la direction de l'entreprise, mais la démonstration concrète d'un changement de mentalité. La contribution essentielle du spécialiste de la production ne réside pas dans sa capacité à soulager les développeurs de toutes les tâches de déploiement, mais dans sa connaissance de la gestion de systèmes à grande échelle – une compétence que l'équipe actuelle ne possède pas.

Idéalement, le spécialiste de la production doit être passionné par la simplicité du déploiement. En effet, la simplicité est toujours préférable, mais elle aide également les développeurs à mieux appréhender le processus de déploiement. Par ailleurs, le spécialiste de la production doit être impliqué dès l'étape du développement, afin de conseiller les développeurs sur la mise en œuvre de la simplicité – par exemple, en évitant d'introduire une nouvelle base de données ou de développer une nouvelle solution lorsqu'il est possible de réutiliser les ressources existantes. Considérez-le comme un consultant en complexité : plus les opérations restent simples, plus le service reste agile.

L'opportunité de la croissance

Par ailleurs, bien que les équipes de production fournissent l'infrastructure, les développeurs doivent rester propriétaires de tous les services qu'ils ont créés. Le spécialiste de la production doit permettre aux développeurs d'acquérir de nouvelles compétences, afin qu'ils puissent suivre leur code jusqu'à la phase de production, au lieu de confier tout ce travail aux équipes de production. Bien que cela puisse sembler être une solution idéale pour les développeurs, en pratique, le développement personnel rend les employés heureux. Avant l'arrivée du spécialiste de la production, les développeurs acquéraient de nouvelles compétences ; intégrer un spécialiste de la production dans l'équipe doit donc accélérer cette tendance, et non l'étouffer.

S'il est nécessaire de mobiliser d'autres spécialistes de la production, il est recommandé d'appliquer le même modèle d'intégration que pour le premier membre ; familiarisez les nouveaux employés avec tous les aspects du cycle de vie du génie logiciel, et plus particulièrement avec le côté développement, et invitez-les à vous conseiller. S'ils ne voient leur code que lors de la phase de production, vous ne leur permettez pas de faire leur travail correctement.

Ainsi, il est possible de déployer des opérations fiables, même en l'absence d'une équipe dédiée pour effectuer tout le travail après l'écriture du code. Le vrai DevOps repose sur le partage des connaissances, qui permet aux développeurs de gérer leur code et d'en assumer la responsabilité, tout au long du cycle de vie.

Ron Gidron

Ron Gidron est directeur marketing produit de Release Automation chez Automic Software. Au cours des 14 dernières années, il a occupé différentes fonctions dans la vente, le marketing produit, l'avant-vente aussi bien dans des startups qu'au sein de grandes entreprises. Ron est passionné par l'interaction des logiciels, des utilisateurs et des tendances du marché.

4e partie : La peste soit des machines : le problème des outils DevOps

La quantité pléthorique d'outils et de méthodes de développement de la culture DevOps, associée à la diversité des compétences du personnel, crée une toile vierge pour les services informatiques. Il existe un million de combinaisons permettant de mettre en œuvre DevOps ; toutefois, votre approche est parfaitement inefficace si elle ne correspond pas aux besoins spécifiques de votre entreprise. DevOps est donc une forme d'art, qui laisse la chance d'être un Monet de l'informatique. Comme dans l'art, il n'existe pas deux œuvres semblables.

Un terrain de jeu inégal

Une large part de la diversité et des possibilités de différenciation dans DevOps concerne la chaîne d'outillage. Le prpblème des outils DevOps, c'est que bien souvent, chaque outil est conçu pour un silo – les outils de gestion de serveurs pour les serveurs, les gestionnaires de configuration pour la configuration, et ainsi de suite. Chaque outil est spécialisé. Si vous essayez d'utiliser un gestionnaire de configuration pour un réseau, cela ne fonctionnera pas. Trop souvent, les personnes tentent de repousser les limites de logiciels cloisonnés et se heurtent à un mur.

Nous avons désormais une très riche chaîne d'outillage DevOps, qui compte des centaines d'outils, mais aucun d'eux ne résout vraiment le problème de la chaîne d'outillage dans son ensemble – la conception, le déploiement et la gestion d'une application dans son ensemble. Les évolutions culturelles ont souvent lieu au niveau « micro ». Les équipes commencent progressivement, en utilisant un nouvel outil pour une tâche particulière et, au fil du temps, une vaste chaîne d'outils s'amoncelle.

La guerre des machines

Ironiquement, bien des fois, nous nous retrouvons avec des configurations rappelant une machine de Rube Goldberg : des amoncellements de bric et de broc, qui donnent l'impression d'avoir été délibérément réfléchis à l'excès. Plus la production approche, s'il subsiste des inquiétudes concernant la sécurité, le partage des ressources, etc., plus cette machine devient complexe.

Le problème prend de l'ampleur lorsque l'on prend en compte le fait que chaque ordinateur est différent. Contrairement à un ordinateur normalisé, personne ne peut rédiger un guide d'utilisation lorsqu'il existe autant de combinaisons d'outils et de préférences.

Il existe même plusieurs ordinateurs hétérogènes au sein d'une même entreprise, puisque les membres de l'équipe DevOps sont encouragés (et à raison, d'ailleurs) à utiliser les outils avec lesquels ils se sentent à l'aise et travaillent rapidement. Il n'existe aucune cohérence entre les ordinateurs et, bien souvent, même au sein des ordinateurs individuels eux-mêmes.

Les nouvelles entreprises peuvent construire des modèles DevOps natifs du Web, simples et homogènes, mais les entreprises établies ne peuvent espérer reproduire leur simplicité. Au sein des grandes entreprises utilisant des applications historiques, il existe tellement de composants qu'une défaillance d'une partie de l'application affecte l'ensemble de l'organisation. Ces entreprises ne peuvent pas se permettre d'être hors ligne, quelle que soit la durée de la défaillance. Les systèmes historiques, énormes et encombrants, associés aux outils natifs du Web plus rapides, créent un maillage technologique complexe qu'il est impératif de gérer efficacement.

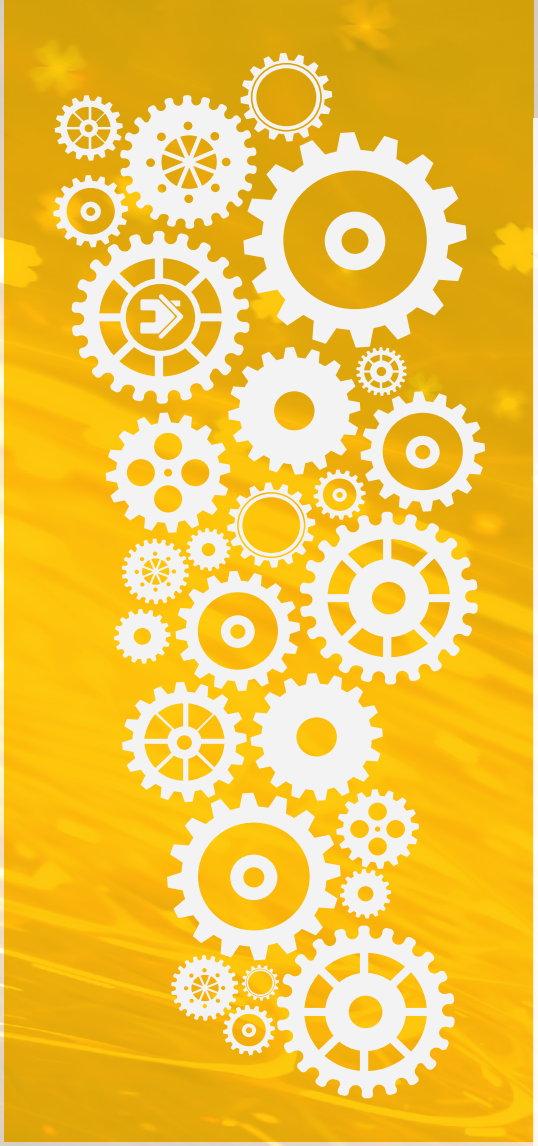
Les meilleurs outils d'automatisation ARA peuvent résoudre ce problème.

Le calme après la tempête

La solution à ce problème viendra d'une nouvelle approche du problème ; il s'agit d'élever le statut des services et applications en les plaçant au sommet de la chaîne d'outils DevOps, au lieu de les considérer comme un dernier composant hâtivement assemblé.

Les meilleurs outils ARA peuvent résoudre ce problème. Ils fournissent une capacité de modélisation d'applications flexible, mais étroitement intégrée, qui facilite le « regroupement » des exigences sous-jacentes (telles que la configuration) ou des exigences relatives à l'exécution (telles que les conteneurs, les machines virtuelles, les logiciels de base et les états de configuration).

Au lieu d'assembler les composants de la chaîne d'outillage en les assemblant à l'aide de scripts et de processus d'intégration complexes, l'intégration des différentes composantes de la chaîne d'outillage au modèle d'application fournit un point de commande et de contrôle central, du point de vue applicatif. Vous demandez à accéder à un logiciel particulier, en spécifiant une version particulière et peut-être même une configuration d'environnement spécifique ; l'automatisation déduit les exigences du modèle et se met au travail à votre place.



Cette approche est révolutionnaire, car elle vous exonère de la conception et de la gestion d'environnements et vous permet de gérer efficacement votre conduit. Elle offre également à votre organisation une souplesse accrue dans la gestion des coûts et une amélioration du coût total de possession en intégrant, dans la couche d'orchestration, des règles de placement de logiciels qui gèrent automatiquement le placement de chaque composante en fonction de la stratégie définie ; et cela, sans vous obliger à redévelopper vos applications ou créer votre propre ordinateur à la Rube Goldberg.

Ce que vous obtenez alors est un processus de déploiement automatisé, qui fournit une vue agrégée de l'ensemble de la chaîne d'outils et facilite la gestion globale de ses différents composants. Les membres de l'équipe DevOps impliqués à chaque étape de la chaîne d'outils savent à quoi s'attendre et à quel moment l'attendre ; cette visibilité étendue rend plus logique et plus naturelle la diffusion réciproque des compétences entre les filières Dev et opérationnelle.

Enfin, elle dévoile une méthodologie permettant d'ordonner le chaos et insuffler de l'ordre à la licence artistique.

Ron Gidron

Ron Gidron est directeur marketing produit de Release Automation chez Automic Software. Au cours des 14 dernières années, il a occupé différentes fonctions dans la vente, le marketing produit, l'avant-vente aussi bien dans des startups qu'au sein de grandes entreprises. Ron est passionné par l'interaction des logiciels, des utilisateurs et des tendances du marché.

5e partie : Faire du DevOps une réalité au sein des organisations « historiques »

En réalité, le terme « historique » désigne n'importe quelle application ou processus de gestion informatique actuellement déployés.

Lorsque nous entendons les termes « DevOps » et « historique », nous avons tendance à imaginer des scénarios extrêmes. DevOps est synonyme d'entreprises numériques, nées dans le Cloud, comme Facebook, Uber et Netflix, qui se déploient même pendant que leurs équipes dorment. À l'inverse, le terme « historique » a tendance à avoir une connotation négative. Lorsque nous entendons ce mot, nous pensons presque instinctivement à des systèmes monolithiques et encombrants, difficiles à exploiter.

En réalité, le terme « historique » désigne n'importe quel processus ou application de gestion informatique actuellement déployés. Le fait qu'il ait été déployé au cours des cinq dernières minutes ou des cinq dernières années ne le rend pas plus ou moins « historique ». Vous devez toujours vous préoccuper des mêmes problèmes – par exemple, l'introduction d'une nouvelle technologie perturbera-t-elle ou compliquera-t-elle l'exécution des systèmes actuellement opérationnels ?

En définitive, peu importe que vous utilisiez le Continuous Delivery moderne ou que vous débrouilliez avec un système historique vieux de trente ans : l'objectif de chacun est de trouver des solutions permettant de limiter les mises à niveau à grande échelle, tout en réutilisant et en améliorant les processus d'infrastructure existants ; d'automatiser ce que vous faites déjà sans rajouter une complexité excessive.

En réalité, comprendre comment les jeunes entreprises numériques ont abordé leurs problématiques peut fournir de précieux enseignements pour la modernisation de l'informatique traditionnelle.

LinkedIn: liée à ses systèmes historiques

Personne n'imaginerait que LinkedIn repose sur des applications historiques. Après tout, l'entreprise fait figure de chef de file du mouvement du mouvement des microservices, et elle est à l'origine de nombreuses contributions innovantes aux architectures applicatives modernes. Toutefois, LinkedIn reste contrainte de gérer et d'assurer la maintenance de systèmes historiques, dans lesquels elle a lourdement investi.

Lorsque LinkedIn a décidé de suivre une nouvelle tendance avec Linux Containers, on aurait pu croire qu'elle serait l'une des premières entreprises à opter pour l'une des grandes plates-formes de gestion de conteneurs, telles que Docker, Cloud Foundry, OpenShift, Kubernetes, Mesosphere ou autre.

Cependant, ça n'est pas le cas. L'entreprise a décidé de concevoir intégralement sa propre plate-forme de gestion Linux Containers, appelée LPS. On pourrait croire que cette décision découlait du célèbre syndrome « Not invented here » (Pas inventé ici).

Cependant, une fois encore, ça n'est pas le cas. La raison pour laquelle LinkedIn a décidé de construire LPS est certes intéressante pour les entreprises nées en ligne, natives du Cloud… mais pas seulement elles. Je pense en effet que les motivations de LinkedIn sont très importantes pour toute entreprise gérant des systèmes historiques.

LinkedIn ne souhaitait pas modifier les processus de gestion informatique qu'elle avait déployés, et qui répondaient parfaitement à ses besoins. Par exemple, lorsqu'un bug de sécurité apparaît dans une bibliothèque de base telle qu'OpenSSL, LinkedIn dispose déjà de processus capables de mettre à jour des dizaines de milliers de serveurs physiques avec un correctif, rapidement et efficacement.

Si l'entreprise avait opté pour une solution de gestion de conteneurs prête à l'emploi, elle aurait potentiellement été obligée de reconstruire et redéployer des centaines de milliers de conteneurs sur l'ensemble de son système, ainsi que sur la structure physique sous-jacente. Cela impliquerait un nouvel ensemble gigantesque de processus informatiques. Or, l'entreprise disposait déjà d'un système efficace. Lorsqu'elle a envisagé l'intégration d'une solution de gestion de conteneurs, elle a longuement réfléchi aux implications à long terme de l'exécution et la gestion de cette nouvelle technologie.

En fin de compte, sa solution de gestion de conteneurs LPS s'est parfaitement intégrée à ses processus opérationnels existants. Elle intégrait Linux Containers de telle manière que lorsque la bibliothèque OpenSSL sous-jacente recevait un correctif sur le serveur physique, tous les conteneurs en cours d'exécution sur le serveur recevraient la nouvelle bibliothèque, sans qu'il soit nécessaire de reconstruire ou redémarrer les conteneurs. L'entreprise pouvait avoir le beurre et l'argent du beurre . Et surtout, elle n'avait pas besoin de réinventer ou de compléter ses processus de gestion informatique pour adopter cette nouvelle technologie.

Je ne dis pas que toutes les entreprises doivent se lancer dans la construction de leurs propres outils, comme l'a fait LinkedIn.

Modernisation réaliste

La plupart des entreprises ne possèdent pas l'expertise, le temps ou la capacité de construire leurs propres plates-formes de gestion des conteneurs. Ce que j'entends par là, c'est que ces organisations dites « historiques » doivent bien réfléchir aux conséquences de l'adoption de nouvelles technologies et à la manière dont celles-ci pourraient affecter leurs processus opérationnels.

Si la révolution numérique a entraîné le déploiement de nouveaux outils à un rythme terrifiant, elle a également eu pour conséquence de moderniser rapidement nos interactions avec les systèmes historiques. Prenons l'exemple de Siebel. Autrefois considéré comme un inhibiteur de la transformation numérique, Siebel peut désormais être intégré au processus de Continuous Delivery, avec le Release Automation. Avant que cela ne soit possible, la tentation aurait été de remplacer intégralement Siebel, à grands frais.

Toutefois, tout ne repose pas toujours sur le choix de l'outil autonome le plus performant ou le plus récent. Le plus important, dans la transformation numérique et l'adaptation au changement, c'est la manière dont les différents ajouts s'intègrent à votre écosystème et s'affectent mutuellement. Le temps pris pour comprendre cet aspect vous permettra d'économiser des ressources à long terme, lorsque l'outil que vous aurez choisi résistera à l'épreuve du temps. Si vous accomplissez cette démarche avec soin, en tenant compte de tous les paramètres, les applications numériques et les pratiques DevOps peuvent soutenir votre activité, plutôt que vous donner l'impression de livrer une bataille au quotidien.

Lucas Carlson

Lucas Carlson est vice-président directeur de la stratégie chez Automic Software ; sa mission est d'aider à dynamiser la stratégie de l'entreprise et le succès des clients, tout en soutenant l'expansion mondiale de l'entreprise.

Dernièrement, Carlson a endossé le rôle de directeur de l'information chez CenturyLink, où il a fondé et dirigé le laboratoire d'innovation de l'entreprise, responsable de recherche et du développement de produits. En outre, Carlson a été fondateur et président-directeur général d'AppFog, une brillante entreprise PaaS (Platform-as-a-Service) spécialiste de la connexion des développeurs aux infrastructures et services (tels que MySQL, Mongo et Redis). Carlson a dirigé AppFog lors de son acquisition par CenturyLink en 2013.

Carlson est également un auteur à succès.

6e partie : L'impact des microservices sur les systèmes informatiques des entreprises

Elles n'ont pas besoin d'être esthétiques ou conçues sur mesure ; elles doivent juste fonctionner et être simples et spécifiques.

L'influence des microservices sur les systèmes informatiques des entreprises pourrait être colossale : si DevOps est le moyen idéal pour permettre aux hommes de collaborer à vitesse maximale, les microservices nous permettent également de fédérer les technologies à une vitesse beaucoup plus élevée. L'incroyable écart entre DevOps et le développement en cascade, en termes de rapidité, pourrait bein se creuser davantage alors que nous atteignons une vitesse record en matière de génie logiciel – du moins, jusqu'à ce qu'une prochaine technologie révolutionnaire, capable de changer les règles du jeu, bouleverse à nouveau le monde de l'informatique.

Si vous ne le saviez pas déjà, les microservices sont des solutions spécialisées, prêtes à l'emploi et intégrées, qui ont chacune le comportement spécifique d'une application ; il s'agit de fonctionnalités fréquemment utilisées, telles qu'un formulaire de saisie d'identifiant/de mot de passe, une passerelle de paiement, une application de panier d'achat, une application d'identification/de session, une fonction de recherche, etc. Elles n'ont pas besoin d'être esthétiques ou conçues sur mesure ; elles doivent juste fonctionner et être simples et spécifiques.

L'industrialisation de l'informatique d'entreprise

À l'heure actuelle, nous créons intégralement des applications ; le génie logiciel n'est donc pas encore industrialisé. Peut-être n'est-ce pas DevOps qui nous permettra pas d'être aussi rapides que possible ; peut-être est-ce plutôt une évolution dans l'architecture des applications qui permettra d'industrialiser entièrement le génie logiciel – ou du moins, c'est ce dont nous nous rendrons compte, avec du recul, dans quelques années.

Les meilleures équipes DevOps utilisent ARA comme un processus répétable, permettant de conserver le même nombre de traitements, même lorsque le nombre d'éléments utilisés peut varier à chaque déploiement. Si les microservices sont au génie logiciel ce que les engrenages, les actionneurs et d'autres pièces sont à l'ingénierie mécanique, ce qui compte vraiment, dans la construction d'une application, c'est l'étincelle d'innovation, qui associe de manière créative tous les microservices fonctionnels dans de nouveaux produits et services innovants – les applications.

Uber et Airbnb utilisent des composants applicatifs communs, tels que les passerelles de paiement, les services GPS mobiles, etc. ; ils auraient pu les acheter ou les télécharger gratuitement, plutôt que de construire intégralement. Une vue d'ensemble et de haut niveau sur de nombreuses applications permet de constater qu'elles utilisent les mêmes composants fonctionnels. L'inspiration, à l'origine de ces applications, est ce qui permet d'associer ces services fonctionnels pour fonder des entreprises valant des milliards de dollars ; c'est également ce qui les différencie de toutes les autres applications, qu'elles dominent sans conteste.

Au lieu d'émuler la création d'une robe sur mesure, comme le fait aujourd'hui l'industrie informatique (ce qui nécessite d'acheter, de mesurer, de couper et de coudre la matière), les microservices peuvent accomplir pour l'informatique ce qu'IKEA a apporté au mobilier, ou ce qu'Henry Ford a apporté à la construction automobile. Si nous parvenons simplement à intégrer des microservices testés et éprouvés dans un ensemble composite, la majeure partie d'une application peut être construite beaucoup plus rapidement.

L'avantage de la spécialisation

Chaque homme sur la ligne de production de Ford, la première dans son genre, était spécialisé dans son travail. La spécialisation est un sous-produit de l'évolution ; nous avons des spécialistes qui préparent nos repas, fournissent de l'eau à notre domicile et nous chantent même des chansons. Chaque microservice peut être créé et testé par une entreprise spécialisée. Par conséquent, au lieu de demander à un développeur touche-à-tout de créer chaque composant conformément à une norme médiane (ceci concerne particulièrement les services informatiques des petites entreprises, qui comptent des effectifs réduits et des salaires modestes), nous pouvons acheter ou télécharger des microservices en open source auprès d'entreprises ou d'individus spécialistes. Les interactions avec les autres microservices peuvent également être testées avec soin, afin d'identifier d'éventuels bugs et de fournir un ensemble cohérent d'API, constitué de composants avec lesquels des composants différenciateurs peuvent ensuite interagir efficacement.

Un complément d'agilité

Puisque les architectures applicatives actuelles réunissent tous les composants sur une même plate-forme, l'ensemble doit être redémarré si un incident se produit au niveau d'un composant (eh oui, le vieux remède « éteignez-le, puis rallumez-le » perdure dans le génie logiciel !). La possibilité de gérer les microservices sur une plate-forme légère et dédiée permet d'identifier plus rapidement les problèmes ; ces derniers devraient de toute façon être plus rares, en raison des composants spécialisés précédemment évoqués. Cette architecture de microservices distribuée et abstraite permet de réinitialiser un composant en moins d'une seconde – un délai qui devrait passer inaperçu auprès des utilisateurs.

Une architecture de microservices permet d'effectuer la mise à jour des composants innovants des applications, qui doivent évoluer en raison du marché et des retours des utilisateurs. Entre-temps, les composants fondamentaux peuvent être laissés de côté, ce qui permet d'éviter les complications liées à la mise à jour de l'ensemble d'une application. Ce dont nous disposons maintenant, c'est l'agilité requise pour mettre à niveau une application à une vitesse largement supérieure, sans affecter l'expérience de l'utilisateur. Au lieu de fabriquer et remplacer une voiture entière plusieurs millions de fois, nous pouvons concrètement améliorer la voiture pendant la conduite, sans même que l'utilisateur ne s'en aperçoive.

Devez-vous opter pour les microservices?

La règle d'or du génie logiciel est la suivante : si quelque chose améliore votre agilité et que vous pouvez vous l'offrir, vous devriez l'utiliser. Le problème de tout nouveau logiciel est la perturbation initiale des pratiques qu'il provoque. Bien que les microservices s'exécutent sur des plates-formes abstraites et plus petites, ils représentent une technologie supplémentaire, que vous devez gérer. De nombreuses considérations et mises en garde doivent être prises en compte lors de l'adoption d'une application reposant sur une architecture de microservices.

Incroyablement, la perturbation initiale est la raison même pour laquelle certains services informatiques n'ont pas encore lancé leur initiative DevOps. Toutefois, à l'instar du DevOps, je pense que la transition vers les microservices permettrait d'accroître considérablement la rapidité du génie logiciel au sein des services informatiques des entreprises. Si nous ne passons plus notre temps à concevoir chaque partie d'une application, le concept Continuous Delivery devient un objectif réaliste.

Scott Willson

Scott Willson est le directeur du marketing de produit pour Release Automation chez Automic Software. Il possède plus de 20 années d'expérience des technologies couvrant le développement de logiciels, l'avant-vente, l'après-vente et le marketing. Scott est passionné par la technologie et son utilisation pour aider les entreprises à créer de la valeur ; il dirigeait des organisations DevOps avant même que l'expression DevOps ne soit inventée.

Pour plus d'informations ou pour obtenir une démonstration du produit, veuillez consulter le site Web www.automic.com.

ca technologies | Automic™

Automic, leader de l'Automatisation, acquis par CA Technologies, aide les entreprises à gagner en compétitivité en automatisant leur processus du Cloud aux environnements Hybride, en passant par le Big Data et l'Internet des Objets. Avec ses bureaux dans le monde entier, Automic est au service de plus de 2 700 clients, dans tous les secteurs tels que la Finance, l'Industrie, la Distribution, les Transports et les Télécommunications. Pour plus d'informations, consultez le sitewww.automic.com