LIVRE BLANC | MARS 2016

DevOps et les testeurs

Dans une série d'articles, Paul Gerrard, consultant et spécialiste des tests, aborde diverses questions sur ces derniers. Il explique l'adoption du concept DevOps du point de vue des tests et des testeurs. DevOps fait partie d'une approche globale à laquelle les organisations ont recours pour livrer fréquemment des applications de qualité. Le résultat le plus probant d'une implémentation DevOps réussie est la réduction du laps de temps écoulé entre le moment où un changement logiciel passe du stade d'idée au stade de la production.

Paul Gerrard
Gerrard Consulting

Sponsorisé par

ca technologies

Les conséquences de DevOps pour les testeurs

Contexte

Dans cet article, j'aborderai la question de l'adoption d'une stratégie DevOps et des conséquences que cela entraîne au niveau de la phase de test et pour les testeurs. Le mouvement DevOps (à défaut d'appellation plus appropriée) gagne rapidement du terrain. À l'instar de nombreux autres mouvements déjà survenus dans le secteur, le rythme d'adoption progresse plus vite que la définition du mouvement elle-même. La notion de DevOps n'est pas encore clairement définie. Des divergences culturelles, l'émergence de nouvelles technologies et une longue liste d'études de cas (des réussites pour la plupart) alimentent de nombreux débats.[1]

Selon votre interlocuteur, DevOps peut être une solution à un problème ou un objectif en soi. Dans certaines entreprises, l'objectif est le « passage au numérique » et l'adoption de DevOps s'inscrit dans une approche globale visant à livrer fréquemment des applications de grande qualité. C'est précisément cette perspective que je soutiendrai dans cet article. Toutefois, dans le marketing des technologies et services DevOps, cet objectif peut devenir confus. La difficulté posée par le changement culturel (ou plus concrètement, par les changements de comportement) indispensable à la réussite est fréquemment sous-estimée.

Par ailleurs, j'appliquerai dans cet article un autre postulat selon lequel les testeurs concernés par DevOps ignorent totalement ce concept. Cet article sera donc rédigé comme une sorte d'introduction à DevOps et comme un argumentaire sur l'impact de cette méthode sur les pratiques de test. Si vous êtes déjà rompu à cette méthodologie, j'espère que cet article vous sera malgré tout utile. Si vous n'êtes pas un testeur, cet article sera pour vous l'occasion de connaître le point de vue de ce type d'utilisateur.

Pour les débutants : que signifie le concept DevOps ?

DevOps est tout simplement l'appellation donnée à la notion de rapprochement entre les équipes chargées du développement (Dev) et de la production IT (Ops). Dans le flux (« pipeline ») de livraison, qui va de la validation du code source à la mise en production, une partie des activités opérationnelles est automatisée par les développeurs, tandis que les équipes de production IT disposent d'une meilleure visibilité, et d'une certaine influence, sur les activités de développement. Le but est, avant tout, d'accélérer le déploiement et l'implémentation des logiciels. Le rapprochement des équipes de développement et de production IT, au sein d'une même équipe agile, donne lieu à ce qui pourrait être qualifié d'« opérations agiles ».

Le résultat le plus probant d'une implémentation DevOps réussie est la réduction du laps de temps écoulé entre le moment où des modifications logicielles passent du stade d'idée à la phase de production. Lorsqu'un développeur déclare qu'une modification logicielle est « terminée », la transition est effectuée avec l'aide d'une automatisation globale. Des outils et processus automatisés interviennent à tous les niveaux : configuration système, processus de création, phase de test, déploiement dans les environnements de test, de stockage intermédiaire et de production, supervision post-déploiement, évaluation et fonctionnement.

DevOps n'est-il donc qu'un ensemble d'outils ?

D'une certaine manière, l'objectif de DevOps consiste à éliminer les engorgements qui perturbent le flux de livraison par le biais de l'automatisation. Cependant, l'automatisation de processus en plusieurs étapes nécessite malgré tout une gouvernance. La plupart des processus automatisés ne sont pas réellement autonomes ; il leur est impossible de mener une tâche à terme sans intervention humaine, que ce soit pour une opération de maintenance ou pour le traitement des exceptions. Un processus DevOps entièrement automatisé ne vaut rien sans une dimension humaine. Si les outils assurent une grande partie du travail, c'est par l'intervention des personnes qui l'exécutent que le processus fonctionne (ou échoue).

DevOps n'est donc rien d'autre que le rapprochement des équipes de développement et de production IT avec l'aide d'outils ?

Non, il ne se résume pas à cela non plus. Les transitions entre les processus automatisés impliquent souvent d'autres processus, généralement des tests, sous une forme ou une autre. La création de tests automatisés est le travail des développeurs et des testeurs. Ces tests ont pour but de fournir suffisamment d'informations à d'autres processus, ou d'autres personnes, pour assurer la transition entre les différentes étapes du flux. Les testeurs et développeurs qui réalisent ces tests garantissent l'efficacité et la fiabilité du processus DevOps.

« Tout cela me donne la migraine. Au final, que représente DevOps ? » C'est une discipline émergente en plein essor. La question est posée et largement débattue dans une publication très pertinente.[1] Le débat a émergé quelques semaines seulement après la rédaction du présent article. Vous remarquerez donc que la définition de DevOps n'est pas encore fixée. Elle ne le sera peut-être jamais.

Mais qu'est-ce que cela implique pour les testeurs ? Cela signifie qu'il n'existe pas de « vérité unique » et que votre rôle dans un régime DevOps en constante évolution (c'est le propre de chaque régime) n'est pas encore fixé. Il y a cependant deux grandes contributions que vous pouvez d'ores et déjà apporter :

1. Vous devez prêter attention aux éléments qui bloquent et travailler à leur amélioration.
2. Vous devez identifier les opportunités et les interventions qui apporteront de la valeur au processus DevOps.

S'il existe un mantra qui décrit le mieux le levier DevOps, c'est « si c'est douloureux, recommencez ». C'est peut-être un cliché, mais c'est aussi le point de départ de l'implémentation et de l'amélioration des pratiques de test DevOps.

Si c'est douloureux, recommencez (souvent)

La difficulté ou la douleur que nous ressentons lorsque nous effectuons une tâche spécifique nous affecte négativement. Si une tâche nous rebute, nous avons tendance à la mettre de côté. Et lorsque nous nous y attaquons enfin, cela nous coûte encore plus. Une visite chez le dentiste, nettoyer le garage, intégrer un logiciel, réaliser des tests, les exemples sont légion. Moins nous effectuons ces tâches, plus l'expérience est traumatique lorsque nous sommes au pied du mur. Martin Fowler suggère trois raisons pour lesquelles la pratique courante, voire continue de certaines tâches atténue la douleur.[2]

La première vient du fait que de lourdes tâches complexes sont difficiles à planifier, gérer et contrôler. En les scindant, ces tâches sont plus gérables, moins risquées et, en cas de problème, plus faciles à analyser. La seconde est que de nombreuses tâches (et le test en est l'illustration parfaite) génèrent des retours. Si ces retours sont reçus suffisamment tôt et régulièrement, les problèmes peuvent être résolus rapidement, probablement avant qu'ils n'entraînent d'autres pertes de temps. Troisièmement, si nous pratiquons une activité plus régulièrement, nous la réalisons de mieux en mieux. Nous apprenons à procéder efficacement. Nous pouvons également déceler des opportunités pour introduire une forme d'automatisation.

Du point de vue du testeur, ce mantra nous oblige à aborder plus sérieusement la notion d'automatisation dans le processus de test. Si des interventions manuelles sont nécessaires (habituellement entre des étapes automatisées du processus DevOps), celles-ci sont généralement perçues comme des points d'achoppement (engorgements, origines des retards et aspects du processus potentiellement les moins fiables et les plus sujets aux erreurs). Les tests manuels sont une étape pénible. Vous adorez peut-être les tests exploratoires ; vous pensez peut-être que seul un être humain est en mesure de déceler ces fichus bogues à côté desquels l'automatisation pourrait passer et que seul un testeur est suffisamment fiable pour prévenir un désastre.

En tant que testeur, il vous est probablement difficile de faire confiance aux développeurs et à l'automatisation pour mener à bien les tests. Si cela est douloureux, vous devez le faire plus souvent.

Test, automatisation et confiance

Les débats font rage autour de certaines notions comme, par exemple, la phase de test et vérification[3], et la crédibilité que nous accordons aux testeurs, aux vérifications et à l'automatisation.[4, 5]

Loin de moi l'idée d'accorder une confiance aveugle aux contrôles automatisés. Nous devons faire preuve de plus de discernement. Nous pouvons toutefois, pour le besoin de cet article, dissocier les tests et l'exécution des tests en quatre composants.

1. Vérifications pouvant être automatisées par les développeurs, dans le cadre des processus d'intégration continue et de vérification au niveau des composants.
2. Vérifications pouvant être automatisées (généralement par les testeurs système) pour exercer les transactions de bout en bout ou les liens au niveau des API.
3. Tests pouvant effectuer des vérifications de compatibilité pour démontrer la compatibilité entre différents navigateurs, systèmes d'exploitation et plates-formes.
4. Tests que seul l'homme peut réaliser.

De toute évidence, chaque environnement étant différent, je ne peux, dans cet article, que formuler quelques suggestions sur la manière d'élaborer ces distinctions. La question la plus pertinente de cet article est la suivante : « Comment le testeur peut-il se « débarrasser » des vérifications manuelles tardives ? » J'ai déjà évoqué l'idée d'une élimination des vérifications manuelles tardives[6] qui nécessitent une confiance et un effort proactif.

C'est là que tous vos efforts devront se concentrer :

1. À chaque fois que c'est possible, toute vérification manuelle pouvant être effectuée au niveau des composants devrait être transférée aux développeurs. En tant que testeur, vous pouvez suggérer ces tests lors d'une réunion entre pairs ou d'une session de tableau blanc. Il est possible que vous deviez écrire vous-même ces tests et les inclure dans le régime d'intégration continue.
2. Les tests de bout en bout ou de l'interface utilisateur peuvent nécessiter une automatisation. Ces tests étant généralement chronophages, fragiles et sujets à maintenance, ils devront être réduits au strict minimum. Posez-vous la question s'ils doivent être exécutés à chaque ajout de code ou s'ils peuvent être réservés aux versions plus importantes, dont la publication est moins fréquente.
3. Quels sont les tests uniquement manuels pouvant être exécutés au niveau des composants qui ne sont pas encore intégrés à une version finale (RC) ? Ces tests manuels peuvent-ils être effectués lors de sessions en binôme avec des développeurs ? Existe-t-il des alternatives à ces tests ? La création de scénarios-maquettes (story-boards) ou de prototypes de style BDD est-elle utile ? Les vérifications de l'interface utilisateur peuvent-elles être effectuées sur des maquettes de type « mock-ups » ou « wire-frames » ?
4. Quels sont les vérifications à effectuer une seule fois, manuellement, par opposition à celles à retenir à des fins de régression et qui sont candidates à l'automatisation ?

J'ai évoqué précédemment la notion de confiance. Une autre façon d'aborder la question consisterait à se demander comment un système peut être testé de façon fiable en l'absence totale de tests manuels tardifs. Imaginez un environnement dans lequel tous les tests seraient effectués par des outils. Vos préoccupations seraient-elles dominées par le fait que, tout simplement, vous ne faites pas confiance aux développeurs pour réaliser convenablement les tests ? Le déplacement des tests vers la gauche (autrement dit, à un stade plus précoce, comme suggéré dans mon précédent article) devrait réduire les doutes. Si, en tant que testeur, vous agissez en facilitateur pour identifier les risques et les évaluer, sélectionner des tests et vous assurer qu'ils soient incorporés aux processus de développement et d'automatisation, vos préoccupations pourraient être atténuées.

De toute évidence, vous devez cesser de vous considérer comme le garant d'une certaine qualité, le dernier rempart, la seule personne sérieuse. Vous devez réfléchir davantage comme un visionnaire, un identificateur et gestionnaire des risques, un meneur, un facilitateur et un coach/mentor.

Pratique, supervision et amélioration

En dépit de toutes les bonnes intentions pour réduire, voire éliminer, les vérifications manuelles tardives, des bogues continueront de passer au travers des mailles du filet. Lorsqu'un logiciel passe en production, les problèmes apparaissent. L'une des disciplines clés de DevOps du point de vue de la production IT est une supervision détaillée.

Une supervision de chaque couche : des composants aux simples transactions effectuées dans les applications, en passant par les intégrations et les messages, et bien évidemment l'infrastructure elle-même. L'un des objectifs de la supervision consiste à tirer la sonnette d'alarme en cas de défaillance, avant que l'expérience utilisateur en pâtisse. C'est plutôt ambitieux, mais c'est le but ultime.

Lorsque des problèmes apparaissent en production, l'analyse tirée de la supervision permet non seulement d'identifier la cause du problème et de le résoudre, mais aussi d'affiner le processus de test, manuel ou automatisé, et de réduire la probabilité que des problèmes similaires se produisent par la suite. J'ai d'ailleurs consacré un autre article au rôle des tests et des analyses sur l'ensemble du flux de livraison.[7]

Certaines personnes assimilent les tests automatisés du processus DevOps à une forme de « supervision ». Cette supervision élargie, depuis le processus DevOps jusqu'en production étend le champ d'application des tests. DevOps ne minimise donc pas le rôle des testeurs.

Conclusion

Récemment, quelqu'un m'a demandé s'il existait des cas de figure dans lesquels une organisation ne devrait pas tenter d'adopter une approche DevOps. C'est une bonne question, mais je pense que la véritable préoccupation est de savoir si le mouvement DevOps est là pour perdurer et si les testeurs devraient en tenir compte. Et je répondrai tout simplement par une autre question.

Pourquoi ne voudriez-vous pas que vos développeurs et vos équipes de production IT collaborent ? Pourquoi ne voudriez-vous pas envoyer des versions et des déploiements plus fiables en phase de test et de production ? Pourquoi ne voudriez-vous pas que la meilleure technologie vienne soutenir des pipelines plus informatifs, plus efficaces et plus précis ? DevOps est assurément une avancée, mais il n'est pas toujours facile d'y parvenir. Inutile de préciser qu'un changement culturel s'impose et que ce n'est pas toujours évident à réaliser.

DevOps permet aux testeurs d'avoir une plus grande influence sur les étapes en amont des projets et les oblige à réfléchir plus sérieusement à l'automatisation des processus de test, de provisioning d'informations et décisionnels. Les testeurs doivent adopter DevOps, car cette approche leur permet d'être plus proactifs et de jouir d'une plus grande autorité et de plus de respect au sein des équipes de projet.

À propos de l'auteur

Paul Gerrard est consultant, enseignant, auteur, webmaster, développeur, testeur, conférencier, entraîneur d'aviron et éditeur. Il a assumé de nombreuses missions de conseils concernant tous les aspects des phases de test et d'assurance qualité logiciels, et s'est spécialisé dans l'assurance qualité des tests. Il a animé des présentations et des didacticiels lors de nombreuses conférences sur le thème des tests, en Europe, aux États-Unis, en Australie et en Afrique du Sud, et a été récompensé à plusieurs reprises.

Diplômé des universités d'Oxford et Imperial College London, Paul a été lauréat en 2010 du prix d'excellence Eurostar European Testing, et en 2013 du prix The European Software Testing Awards (TESTA) Lifetime Achievement.

En 2002, Paul a coécrit l'ouvrage « Risk-Based E-Business Testing » avec Neil Thompson. En 2009, il a écrit « The Tester's Pocketbook ». En 2011, il a coécrit « The Business Story Pocketbook » avec Susan Windsor et a rédigé « Lean Python » en 2014.

Cette année-là, Paul a été également président de programme à la conférence EuroSTAR de Dublin.

Il est directeur de Gerrard Consulting Limited, directeur de TestOpera Limited et hôte du Test Management Forum.

Courriel : paul@gerrardconsulting.com
Twitter : @paul_gerrard
Site Internet : gerrardconsulting.com

Pour plus d'informations, rendez-vous sur la page **Développement et test** de CA Technologies.

Restez connecté à CA Technologies sur ca.com/fr

CA Technologies (NASDAQ : CA) fournit les logiciels qui aident les entreprises à opérer leur transformation numérique. Dans tous les secteurs, les modèles économiques des entreprises sont redéfinis par les applications. Partout, une application sert d'interface entre une entreprise et un utilisateur. CA Technologies aide ces entreprises à saisir les opportunités créées par cette révolution numérique et à naviguer dans « l'Économie des applications ». Grâce à ses logiciels pour planifier, développer, gérer la performance et la sécurité des applications, CA Technologies aide ainsi ces entreprises à devenir plus productives, à offrir une meilleure qualité d'expérience à leurs utilisateurs, et leur ouvre de nouveaux relais de croissance et de compétitivité sur tous les environnements : mobile, Cloud, distribué ou mainframe. Pour en savoir plus, rendez-vous sur **ca.com/fr**.

Références

1. « What is DevOps », The Agile Admin, http://theagileadmin.com/what-is-devops/
2. « Frequency Reduces Difficulty », Martin Fowler, http://martinfowler.com/bliki/FrequencyReducesDifficulty.html
3. « Testing and Checking Refined », James Bach, Michael Bolton, http://www.satisfice.com/blog/archives/856
4. « A New Model for Testing », Paul Gerrard, http://dev.sp.qa/download/newModel
5. « The New Model and Testing v Checking », Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/659
6. « How to Eliminate Manual Feature Checking », webinaire de Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/622
7. « Thinking Big: Introducing Test Analytics », Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/630

CS200-181159_0316