ca technologies

Guide de référence des tests en continu

2018

Les clés pour la réussite de votre parcours de test en continu

CHAPITRE 1

Plus le rythme des affaires s'accélère, plus les entreprises prennent conscience de la nécessité de revoir leurs processus de développement et de mise en production des logiciels pour préserver leur compétitivité. Dans un monde où la cadence des mises en production s'est considérablement accélérée, il n'est plus envisageable de faire accepter des délais de résolution des problèmes de six à dix-huit mois aux clients. Les projets doivent progresser et produire des résultats parfaits plus rapidement.

Nous assistons à une véritable révolution en matière de responsabilité des tests : celle-ci passe en amont du cycle de développement logiciel, pour finalement le couvrir de bout en bout. Ainsi, les cloisonnements traditionnels entre développeurs et testeurs sont abattus, et la qualité devient la responsabilité de chacun. Les tests sont devenus essentiels, car les conséquences de tests médiocres sont désormais plus visibles.

Cette évolution technique appelle aussi une évolution culturelle : au fur et à mesure que les tests évoluent, pour s'étendre tout au long du cycle de développement logiciel, les entreprises et leurs équipes IT doivent revoir leur culture de développement et de tests logiciels.

En termes plus simples, nous devons cesser de considérer les tests comme une phase dans le cycle de développement logiciel, une tâche autonome devant être exécutée à un moment spécifique. Désormais, chacun doit participer à cette tâche, à tout moment et avant même le début du développement, pour éliminer les défauts et problèmes à la source plutôt que de prendre le risque de les manquer avant la livraison aux clients.

L'objectif de la démarche DevOps consiste à éliminer les cloisonnements entre développement et production, afin de s'assurer que les logiciels ainsi produits, selon des méthodologies agiles de préférence, puissent être déployés aussi rapidement que possible sans nuire à leur qualité. Cependant, même les organisations qui mettent en œuvre la démarche DevOps se voient souvent obligées de faire des compromis entre qualité et rapidité.

Dans la culture de tests actuelle, 63 % des retards de développement sont dus aux pratiques d'assurance qualité en place dans le cycle de développement logiciel, et 70 % des tests sont encore manuels. Cela n'est plus acceptable : les organisations ne peuvent plus se contenter d'exécuter des tests manuellement. En outre, il convient de prendre en compte d'autres problèmes :

56 % ***des dépendances critiques sont indisponibles***

50 % ***du temps est perdu à rechercher des données de test***

64 % ***des défauts émergent pendant la phase de définition des exigences***

Les échecs, en particulier ceux qui atteignent les utilisateurs, ont un impact négatif sur les données, les processus métier, l'adoption client, la fidélisation et les marques elles-mêmes. Il n'est pas possible de produire des logiciels de qualité rapidement en continuant d'utiliser des processus en cascade (ou même agiles) dépassés.

Il convient de développer un cycle plus complet, dynamique et fluide afin de garantir un développement et un déploiement de logiciels de qualité ; un cycle dans lequel les erreurs seront détectées dès les premiers stades, plutôt que lors de contrôles intervenant seulement en fin de cycle. Nous devons nous engager dans un parcours de test en continu.

QUE REPRÉSENTE UN PARCOURS DE TEST EN CONTINU ?

La pratique des tests en continu vise à intégrer des tests dans chaque phase du cycle de développement logiciel, afin de détecter et de corriger les comportements inattendus dès qu'ils apparaissent.

Les tests deviennent alors un composant fondamental et continu de chaque étape du cycle, depuis la phase de définition des exigences jusqu'à la production, pour garantir la réalisation de la valeur métier attendue.

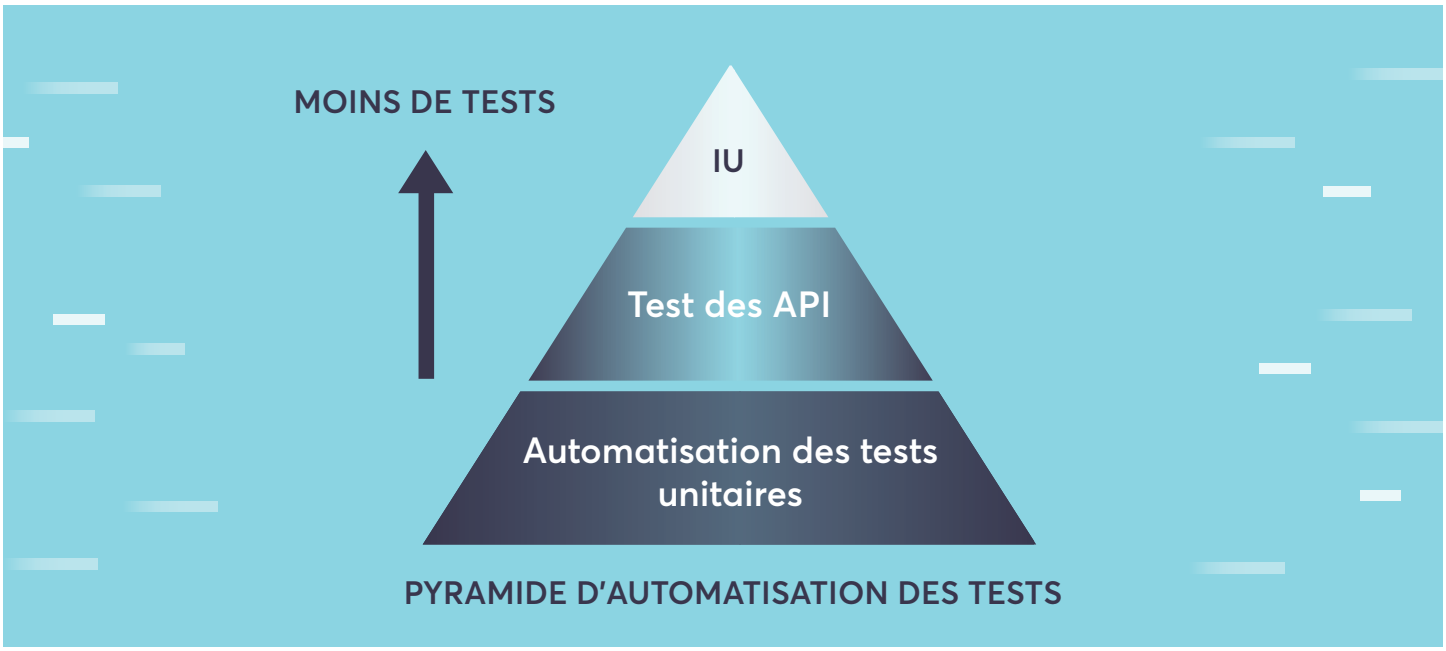
CHAPITRE 2 DÉFIS LIÉS À LA QUALITÉ

La vie n'a jamais été facile pour les organisations IT. Imaginez un service rempli d'artisans et d'ingénieurs dédiés cherchant à développer et concevoir un produit qui fonctionne conformément à des spécifications, lesquelles peuvent s'avérer ne pas être à la hauteur des attentes et limites des utilisateurs finaux une fois le produit livré. Le rôle des logiciels a évolué : de nos jours, les produits doivent être disponibles sur un large éventail de plates-formes et fournir une série de services en évolution constante.

La qualité ne peut qu'en pâtir lorsque les personnes en charge de tester, développer et créer les exigences des logiciels travaillent séparément les unes des autres. C'est pourquoi l'IT doit sortir de son image de fonction de back-office, pour disposer de son propre siège à la table des décisions.

La qualité pâtit aussi lorsque les délais sont irréalistes et que les capacités sont surexploitées en vue de produire un logiciel sans respecter des métriques de qualité appropriées.

Dans le cycle de développement traditionnel, les tests étaient réalisés en silos. En conséquence, la plupart de ces questions restaient sans réponse : « La qualité des tests est-elle bonne ? Quelle est la couverture fonctionnelle ? Quelle est la couverture fonctionnelle au-delà du code ? Quels sont les critères d'acceptation ? Est-ce qu'ils sont testés efficacement ? » Beaucoup de testeurs étaient dans l'impossibilité de répondre.



PYRAMIDE D'AUTOMATISATION DES TESTS

Pour garantir qualité et rapidité, les testeurs doivent réaliser un nombre important de tests unitaires et réduire les tests au niveau des API et (encore plus) de l'interface utilisateur. Les tests (fonctionnels, de performance et de sécurité) devraient être effectués en parallèle et plus en amont du cycle (« shift left ») pour garantir un processus de test en continu dès le début du cycle.

Il arrive trop souvent que les équipes consacrent du temps à des activités qui ne génèrent pas de valeur, testant les mauvaises choses ou le mauvais ensemble de données. Imaginons que l'équipe de testeurs compte huit personnes : si elle n'intervient qu'en fin de cycle et détecte que 40 % du travail doit être retesté et corrigé (une statistique courante), cela signifie qu'en ayant testé le travail plus tôt, elle aurait pu travailler en effectif réduit et prendre moins de retard. Il convient de réaliser une analyse des flux de valeur, pour vérifier que les tests sont réalisés de manière appropriée. L'analyse des flux de valeur est une méthode qui permet d'analyser les processus actuels afin d'identifier un processus plus efficace, en utilisant une idéologie de « lean management ».

La plupart des entreprises continuent d'utiliser des outils « hérités » : si ces outils conviennent à des méthodologies de travail en cascade, ils ne fonctionnent pas bien en mode « shift left ». Ils constituent même un obstacle à des tests agiles. Ils ne sont pas compatibles avec une automatisation nécessaire à un environnement d'intégration continue, en plus d'être rejetés par les développeurs. L'organisation doit adopter un nouvel ensemble d'outils qui ne soit pas un obstacle à la vitesse et à la qualité, et qui permette d'incorporer la qualité dans le processus, pas « uniquement » de tester l'application comme avant.

Au-delà de l'automatisation des tests, c'est aussi le processus de définition des exigences qui doit être amélioré, pour créer des exigences claires, complètes et qui puissent être testées.

Actuellement 64 % des défauts trouvent leur origine dans la phase de définition des exigences.

Très souvent, les exigences sont spécifiées d'une telle façon que seuls un développeur et un testeur disposant de l'expertise appropriée sont en mesure de les interpréter et éventuellement de les compléter. Le risque est alors que les autres membres de l'équipe ne les interprètent pas de la même manière et introduisent des défauts dans le cycle de vie avant même qu'une seule ligne de code n'ait été écrite. D'autres fois, les exigences sont définies de façon extrêmement détaillée, sur des dizaines et des dizaines de pages : cette longueur et ce niveau de détail rendront leur maintenance difficile dans le temps. Sans oublier que développeurs comme testeurs trouveront difficile d'appréhender correctement un tel volume d'informations. Il convient d'utiliser un meilleur moyen de définir les exigences, un moyen qui convienne à des équipes agiles qui adoptent la démarche DevOps.

Une autre approche est également nécessaire en ce qui concerne l'assurance qualité. À commencer par la direction, qui doit se concentrer sur la qualité et pas uniquement sur la livraison. Désormais, la livraison ne se résume plus au déploiement ou à la mise en production d'un logiciel, elle inclut également l'expérience utilisateur/client. Cela doit être pris en compte dans la définition des exigences sur lesquelles les développeurs et testeurs s'appuient pour comprendre ce qu'ils sont en train de construire. Alors que les mises en production se produisent sur un rythme mensuel désormais, il est essentiel de mettre en place des boucles de rétroaction pertinentes et efficaces. L'assurance qualité ne doit plus être vue comme une activité secondaire parce qu'elle a été, jusqu'à présent, identifiée comme une propriété partagée par les services de l'entreprise.

Pour cela, des changements d'ordre culturel et organisationnel doivent survenir. C'est un problème plus humain que technique, mais il n'en est pas moins essentiel à un parcours de tests en continu performant. Quel que soit le changement, certaines personnes seront en mesure de le gérer et d'autres pas. Ce changement représente aussi une menace pour les centres d'excellence, dans la mesure où il suppose de confier la mission de tests en partie à des personnes qui ne sont pas formées.

Les « centres d'excellence » doivent se transformer en « centres de formation » : les testeurs les plus compétents et expérimentés doivent aider les personnes amenées à effectuer des tests tout au long du cycle de développement logiciel à connaître les processus appropriés et leur fournir les bons outils. Une prise en charge de la gestion du changement ascendante et descendante est nécessaire.

Au moment où les tests remontent dans le cycle de développement, les performances et la sécurité doivent opérer le même mouvement. Elles ne doivent plus être considérées comme des tâches « non fonctionnelles », mais être traitées sur un pied d'égalité avec les tests fonctionnels. Les tests de sécurité actuels avant le déploiement en production sont

inadéquats : il est temps de renforcer la sécurité, en intégrant des pratiques de test de sécurité spécialisées dans le processus de test en continu.

Une fois qu'une application est en production, elle peut ne pas offrir un niveau de performance correct dans des conditions de trafic réel. Si l'ajout de puissance de traitement peut sembler une solution appropriée, il ne sera possible qu'en recourant à une infrastructure basée dans le Cloud. Par conséquent, il est nécessaire de tester les performances des applications bien en amont de la production, au niveau unitaire (développement), avant d'accroître la portée des tests de performance au fur et à mesure que les unités de code commencent à former des composants plus importants pour créer des stories, des fonctionnalités et des epics, en exploitant les tests de performance réalisés précédemment. Pour cela, les entreprises doivent faire appel à des outils et processus appropriés.

Un changement de culture implique de combiner des initiatives de changement ascendantes et descendantes avec un leadership performant. Dans l'ensemble, il est nécessaire de mettre en place une culture de tests en continu. Les dirigeants et responsables IT doivent échanger avec les testeurs et les développeurs, et se concentrer ensemble sur la création d'un plan. Ils doivent les faire devenir une partie de la solution en les impliquant dans l'initiative et leur donner l'occasion d'échouer et d'en tirer des leçons.

CHAPITRE 3 COMPOSITION DES TESTS EN CONTINU

Dans un processus de tests en continu, tout le monde code et teste. L'équipe est responsable individuellement et collectivement de l'ingénierie et de la livraison de logiciels de qualité. En quelques années, nous sommes passés de l'approche en cascade au scrum, puis à la méthodologie Agile et à la fusion entre les démarches Agile et DevOps, pour atteindre une agilité métier alimentée par des tests en continu.

Dans ce système où la responsabilité des tests est collective, les rétrospectives permettent d'inspecter et d'adapter le processus et ses résultats en continu, redéfinissant la définition de « travail fait ». La mise en place d'un état d'esprit de tests en continu implique de fournir aux développeurs des retours rapides afin de garantir la qualité de leur code et la circulation rapide de ce dernier à travers le pipeline de mise en production. Cela implique également de créer ou de localiser les données qui doivent alimenter les tests fonctionnels et non fonctionnels à exécuter avant et pendant la production.

Les trois composants clés des tests en continu sont les suivants :

RESSOURCES

Fournissez aux développeurs les outils de test dont ils ont besoin. Les développeurs et équipes agiles doivent prendre conscience que les tests font partie de leurs responsabilités.

PROCESSUS

Automatisez tout.

TECHNOLOGIES

Fournissez des solutions de test que les développeurs utiliseront.
Cela implique de l'Open Source « en tant que code » dans le Cloud.

L'entreprise doit s'assurer qu'elle dispose d'une définition des tests en continu comprise universellement. Comme c'est une démarche encore assez nouvelle, les définitions et pratiques peuvent varier, comme d'autres domaines du développement logiciel.

La pratique des tests en continu vise à intégrer des tests dans chaque phase du cycle de développement logiciel, afin de détecter et de corriger les comportements inattendus dès qu'ils apparaissent.

Imaginez une ligne d'assemblage logiciel automatisée et incluant des tests à chaque étape (au niveau de la collecte des exigences, et depuis l'intégration du premier code jusqu'en production). Cette ligne inclut tous les aspects des tests jusqu'aux tests d'intégration et de performances ainsi que la supervision en production. Il s'agit de mettre en place un régime de test efficace et efficient, centré sur les applications et de bout en bout (comprenant l'intégration continue, la livraison continue et la production). L'objectif est d'être aussi efficace qu'humainement possible pour faire passer les changements en production dans le délai le plus court possible, afin d'atteindre un niveau de qualité et une expérience client optimaux.

CHAPITRE 4 LES DISCIPLINES DES TESTS EN CONTINU

Accélérer et améliorer un processus de mise en production de code à travers des tests en continu nécessite de faire appel à différentes disciplines pour s'assurer que la qualité est préservée. Pour incorporer de la qualité dans les applications, les équipes doivent ajouter des pratiques de tests plus modernes sous forme de petits contrôles de qualité réalisés tout au long du pipeline des applications. Cela permet de garantir que de petites sections de code sont testées en continu.

L'automatisation des tests n'est pas toujours la première tâche à laquelle les équipes doivent s'attaquer. La raison ? Même lorsque les tests des API et de l'interface utilisateur sont automatisés et intégrés dans un agent d'intégration continue, ces tests ont des dépendances (données de test, interfaces et environnements) qui doivent être satisfaites avant l'exécution. Vous devez réaliser les tâches suivantes :

- Éliminer les contraintes environnementales et libérer les développeurs et testeurs pour leur permettre de faire leur travail, même manuellement. Virtualisez toutes les interfaces que vous ne possédez ou ne contrôlez pas, ou que vous ne souhaitez pas tester.
- Utiliser des environnements de test éphémères.
- Automatiser le provisioning et la gestion des données de test pour alimenter vos tests, manuels ou automatisés, à la demande.
- Automatiser vos tests de façon à ce qu'ils puissent s'exécuter dans vos interfaces virtualisées sur la base de données de test appropriées.
- Configurer le moteur d'orchestration de votre pipeline de façon à ce qu'il n'y ait aucune intervention manuelle sur les étapes précitées.
- Vous concentrer (ensuite) sur les disciplines complémentaires.

11 DISCIPLINES DE TESTS EN CONTINU

1. **ENVIRONNEMENTS VIRTUALISÉS**
   Les tests en continu sont synonymes de tests plus fréquents. Cela signifie que vous serez confronté à de multiples environnements plus fréquemment… et à des goulets d'étranglement si ces environnements ne sont pas disponibles en continu. Certains environnements sont accessibles via des API, d'autres via des interfaces de messagerie. Ils peuvent présenter des architectures modernes, s'appuyer sur des systèmes mainframe ou encore des systèmes client/serveur monolithiques. Comment faire pour coordonner vos tests dans des environnements qui ne sont pas forcément à jour ni opérationnels ? En les virtualisant. En les rendant éphémères, disponibles à la demande, vous pourrez éliminer cette contrainte de votre cycle de développement et contrôler et configurer ces environnements selon vos besoins.

2. **GESTION DES DONNÉES DE TEST**
   Vous devez disposer de données appropriées et suffisamment diverses pour alimenter vos scénarios positifs et négatifs. Or, vous ne trouverez pas toute cette diversité en production. La génération de données synthétiques vous permet de réaliser des tests en continu tout en protégeant les informations d'identification personnelle. Par ailleurs, elle permet de disposer rapidement des données requises, ce qui n'est pas possible si vous utilisez des données de production que vous devez d'abord conditionner et provisionner.

3. **AUTOMATISATION DES TESTS**
   Dans un pipeline d'applications intégralement orchestré, les scripts actuels échoueront pour des raisons qui ne sont pas liées au code des applications. En conséquence, personne n'aura confiance dans les résultats. Vous avez donc besoin d'une automatisation fiable pour mettre en place des tests en continu.

4. **ORCHESTRATION DU PIPELINE**
   C'est le pilier de votre cycle de développement logiciel. Tout y est lié. Elle doit être intégrée dans votre suite d'automatisation. Vous devez comprendre comment elle fonctionne, comment interpréter les résultats et comment la rendre évolutive. C'est la façon dont vous intégrez vos attributs de test en continu dans votre pipeline. Elle doit être transparente : tout le monde doit avoir une visibilité complète sur ce qui est exécuté dans le pipeline. C'est un outil de workflow automatisé qui va exécuter tous vos tests automatisés et s'intégrer avec les activités de déploiement de code au fur et à mesure que celui-ci progresse dans le pipeline. Comme dans toute initiative d'adoption DevOps, vous ne pourrez pas mettre en place des tests en continu si vous ne disposez pas de la fiabilité et de la vitesse d'un pipeline standardisé et automatisé.

5. **TESTS DES API**
   Cette étape permet l'alignement de la pyramide de test. Il est important de tester autant que possible au niveau unitaire et des API, afin de limiter la dépendance vis-à-vis des tests d'interface utilisateur. Pour mettre en œuvre un processus de tests en continu, vous devez appliquer correctement le concept de la pyramide de tests, en renforçant les tests unitaires et des API et en réduisant la dépendance vis-à-vis des tests d'IU, en particulier pour les tests de la logique métier.

6. **TESTS DE PERFORMANCE/CHARGE**
   Même si vos applications sont fonctionnellement bonnes, vous devez radicalement changer la façon dont vous envisagez les tests de performance. Comme vous testez plus en amont dans le cycle de développement (« shift left »), dans des phases où les composants d'application et d'infrastructure sont moins nombreux, vos tests sont par nature plus petits, mais bien plus importants en nombre. Vous devez rendre cette capacité accessible à tout le monde au sein des équipes agiles : tous les développeurs et testeurs doivent être en mesure de créer un test de performances et de l'exécuter directement, sans avoir à envoyer des requêtes à qui que ce soit.

7. **TESTS DE SÉCURITÉ**
   Les développeurs doivent clarifier les attentes autour de l'état du code, afin de faciliter l'autorisation ou le refus des mises en production. Ils doivent recevoir une formation et des outils appropriés pour mesurer le niveau de sécurité de leur code. La sécurité des applications étant un domaine en constante évolution, la formation des développeurs en la matière doit elle aussi être adaptée en continu. Le pipeline des applications doit être configuré de manière à exécuter automatiquement des analyses statiques, dynamiques et de composition logicielle, afin que les problèmes de sécurité soient détectés et signalés, et aussi utilisés comme une opportunité d'amélioration pour les développeurs. Les tests de sécurité doivent survenir à chaque étape du cycle de développement.

8. **DÉVELOPPEMENT PILOTÉ PAR LES TESTS ET DÉVELOPPEMENT PILOTÉ PAR LE COMPORTEMENT**
   Appuyez-vous sur les critères d'acceptation de chaque user story pour créer les tests visant à vérifier qu'ils sont respectés. Cela vous permettra de centrer les tests sur les sprints et de garantir que les développeurs développent ce que le métier attend. Avec le temps, les équipes finiront par définir des critères d'acceptation bien plus détaillés, lesquels exigeront des tests également bien plus exhaustifs.

9. **GÉNÉRATION DE TESTS AUTOMATISÉE**
   L'activité de conception et de rédaction manuelle de tests ou de scripts de tests automatisés est un goulet d'étranglement. Nous vous recommandons de générer automatiquement vos nouveaux tests d'acceptation manuels et automatisés au début du sprint. Ainsi, les tests pourront se déclencher automatiquement dès que les développeurs commenceront à travailler sur leur code. Ils pourront ainsi obtenir des retours immédiats sur la qualité de leur code, directement à la source.

10. **INGÉNIERIE DES EXIGENCES**
    Vous devez inclure l'ensemble des parties prenantes au cycle de développement logiciel dans ce parcours de tests en continu. Cela vous oblige à trouver un meilleur moyen de communiquer et de collaborer concernant les exigences : plus tôt vous inclurez l'ingénierie des exigences dans votre initiative de tests en continu, plus votre parcours sera fluide, car les membres de l'équipe travailleront avec les mêmes informations dès le départ.

11. **BOUCLES DE RÉTROACTION**
    Ces boucles de rétroaction sont essentielles à la réussite des tests en continu, à la manière de tableaux de bord en temps réel, automatiques et accessibles par l'équipe entière. Présentes à travers l'ensemble du cycle de développement logiciel, pas uniquement dans la phase de production, ces boucles doivent servir de compas pour vous aider à gérer le passage vers des tests en continu.

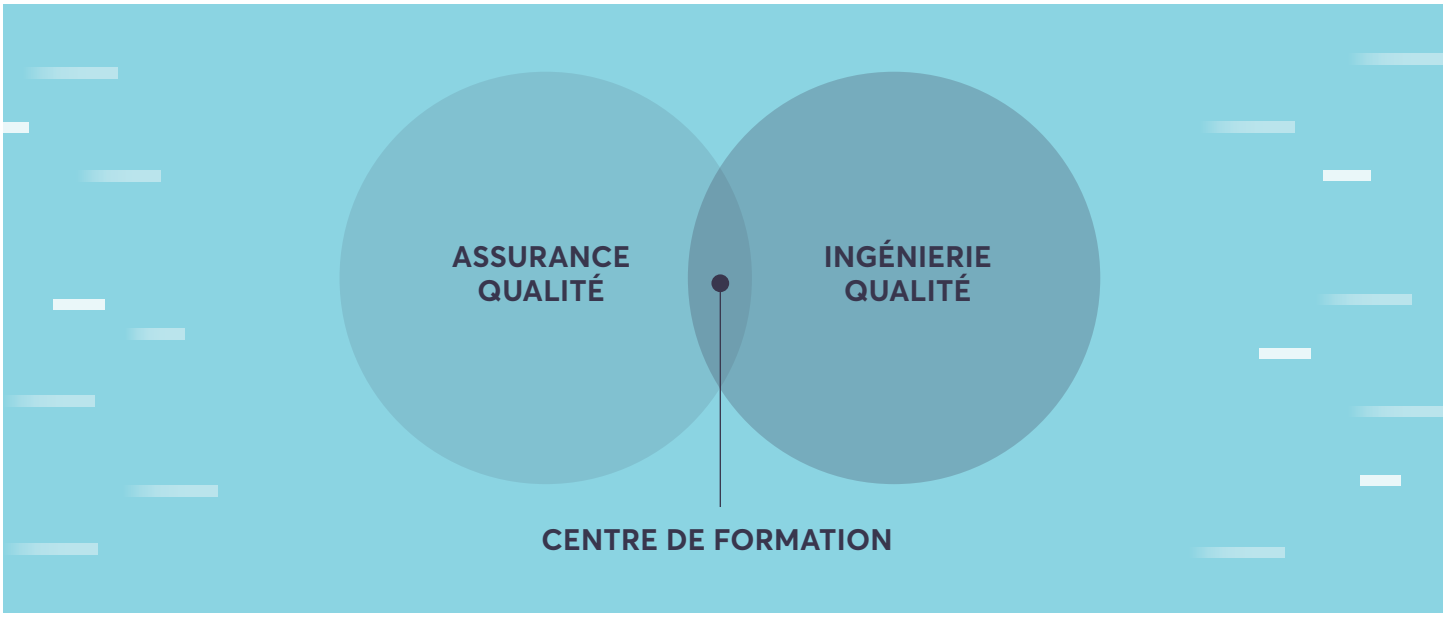
CHAPITRE 5 ACCÉLÉRER DANS LES VIRAGES

Composant essentiel de l'approche DevOps, le passage à une culture de tests en continu implique de s'engager dans une nouvelle façon d'apprendre et de pratiquer. Elle exige aussi de prendre de nouvelles habitudes, soutenues par la science et la connaissance pratique. Face à des clients, professionnels comme particuliers, très peu tolérants en cas d'expérience de mauvaise qualité, ce type d'initiatives est appelé à prendre beaucoup d'importance dans la réussite d'une entreprise.

Si certaines des nouvelles pratiques peuvent sembler aller à contre-courant de la culture existante, c'est là la marque du changement et de l'innovation. Les équipes doivent être familiarisées à ces nouvelles pratiques et accompagnées dans la gestion de la transition.

Il peut, par exemple, être utile d'incorporer des notes de versions dans le code d'une nouvelle fonctionnalité, afin que tout le monde puisse les lire. Comme dans le monde de la gestion de projet, vous devez créer un plan qui permette à n'importe qui de vous suppléer en cas de besoin.

L'assurance qualité doit évoluer pour devenir un centre de formation à la qualité : un endroit où les équipes intègrent leurs connaissances en matière de test, de déploiement, de supervision et même d'autocorrection, de façon à ce que les équipes d'applications puissent les exploiter dans leurs scrums sans dépendre d'équipes externes.



Tout doit être traité comme du code afin de pouvoir être contrôlé de façon standard et évolutive, à travers l'intégration continue. Les équipes doivent réfléchir à la façon dont leur code sera testé, afin de s'assurer que, s'il doit être intégré dans le référentiel de codes source, il fonctionnera comme il est censé le faire, sans provoquer d'effets négatifs dans d'autres parties de l'application.

Les personnes impliquées, développeurs et testeurs, doivent devenir des acteurs à part entière de cette nouvelle culture. Une organisation DevOps doit être mise en place afin de recruter ou de former à nouveau les personnes, pour leur permettre d'endosser différentes casquettes. Pourquoi ne pas mettre en place une organisation matricielle, qui permettrait à l'équipe de développement d'applications de recruter ou d'utiliser les compétences dont elle a besoin quand elle en a besoin ? La gestion du changement au niveau humain est aussi vitale que la transformation physique pour les tests en continu.

Du point de vue des plates-formes, le Cloud est un facteur de succès de plus en plus important. Il est impossible de déployer un parcours de tests en continu avec seulement les systèmes sur site et hérités.

CHAPITRE 6 OÙ EN SUIS-JE DANS MON PARCOURS DE TESTS EN CONTINU ?

Il arrive un moment, tôt dans le parcours de test en continu, où les responsables et les parties prenantes doivent se poser cette question. Il est aussi essentiel de comprendre l'impératif des tests en continu, et de les définir de manière adéquate, que de s'interroger, par la suite, sur l'état d'avancement de l'équipe.

DEUX POINTS ESSENTIELS À GARDER EN TÊTE À CE STADE

- Toujours placer le client au centre du processus de changement et des activités de tests en continu en résultant
- Déterminer comment mesurer la qualité de façon appropriée.

Il convient d'adopter de nouvelles techniques, mais aussi d'identifier et éliminer les barrières présentes.

TECHNIQUES À ADOPTER

1. **AMÉLIOREZ LA RELATION ENTRE TESTEURS ET DÉVELOPPEURS**
   Conservez des petites équipes, encouragez la collaboration entre les équipes et créez des rapports facilement accessibles que vous partagerez en ligne.

2. **FAITES DE L'AUTOMATISATION UNE PRIORITÉ**
   Plutôt que d'éliminer toutes les tâches de test manuelles, cherchez plutôt les opportunités d'automatisation. Concentrez-vous sur les tâches appelées à être exécutées de manière répétée. Il s'agit d'installer la plomberie d'abord, afin de pouvoir faire couler l'eau plus tard sans craindre d'éventuelles fuites. Mappez votre cycle de développement logiciel afin d'identifier les opportunités d'automatisation.

3. **PENSEZ PETIT**
   Décomposez le travail en petits incréments, plus faciles à tester après le codage et la conception. Cela facilitera l'automatisation des tests, et aussi le déploiement.

4. **SUIVEZ TOUT**
   Utilisez des métriques et des critères de réussite/échec. L'objectif des tests en continu consiste à identifier immédiatement si les choses fonctionnent ou non ; veillez donc à ce que cela soit possible.

5. **ÉQUIPEZ-VOUS DES OUTILS DE TEST EN CONTINU APPROPRIÉS**
   Identifiez les outils qui vous aideront à développer, tester et analyser en continu. Sélectionnez les meilleurs outils compatibles entre eux et intégrez-les dans votre environnement de travail. Choisissez des outils dont l'assistance est pilotée par une communauté au sein de laquelle chacun peut partager ses défis, ses solutions et des scénarios d'utilisation intéressants.

6. **DÉVELOPPEZ UN SYSTÈME POUR AFFICHER VOS RÉSULTATS**
   Analysez en profondeur vos résultats : c'est ainsi que vous pourrez déterminer si votre code fonctionne et identifier les lacunes. Définissez vos indicateurs clés de performance (KPI) et vos critères d'acceptation, et rendez-les quantifiables. Créez des tableaux de bord pour suivre vos KPI, en y intégrant une ligne de référence et les changements ultérieurs.

Le passage à des tests en continu peut paraître cher. La meilleure méthode consiste à commencer par un petit projet, que vous amènerez à la réussite, avant d'enchaîner avec d'autres petits projets. C'est un processus itératif dans lequel vous devez viser de petites victoires pour apprendre d'elles et générer un élan.

Des échecs ne manqueront pas de se produire, mais à ce moment-là, les équipes et la direction seront préparés et pourront apprendre de ces petits échecs pour gagner en expérience.

Il est important dans ce contexte que la direction de l'entreprise assume ses responsabilités. Un ambassadeur seul ne pourra jamais fédérer plus de 20 à 30 % des personnes. Pour le reste de l'organisation, il appartiendra à la direction de définir les attentes et d'engager la transition.

Les obstacles au passage à des tests en continu doivent être identifiés individuellement. Il peut notamment être question de l'impossibilité à accéder et à utiliser des éléments Open Source. La culture liée au travail (les personnes, mais aussi les pratiques) pourrait ne pas être suffisamment préparée à cette transition. Vous pourriez manquer de temps pour lancer l'implémentation, la formation et la transition vers des tests en continu. Votre entreprise pourrait également être limitée par une équipe de direction inefficace et des applications héritées.

L'objectif ultime à ce stade est de développer et de déployer un modèle de maturité des tests en continu, ce qui signifie faciliter un bon état d'esprit en termes d'attitude et d'aptitudes. Renforcez l'automatisation des tests unitaires et des API dans les domaines fonctionnel, de performances et de sécurité ; limitez l'automatisation des tests au niveau de l'interface utilisateur, les scripts étant très sensibles aux changements de l'interface utilisateur. Ajoutez une boucle de rétroaction qui inclut des outils de supervision des performances et des utilisateurs dans les environnements de préproduction du pipeline. Une télémétrie complète permettra d'accélérer la résolution des défauts.

Toutes ces fonctionnalités doivent être entièrement intégrées, collaboratives et ne pas fonctionner en silos. L'entreprise doit être capable d'accepter et de comprendre l'Open Source comme étant une approche permettant aux personnes d'expérimenter et d'échouer rapidement sans avoir à justifier les dépenses engagées. Elle doit aussi garantir un accès adéquat aux données de test, car il s'agit là de l'une des barrières les plus importantes.

CHAPITRE 7 MÉTRIQUES

Tout projet nécessite des métriques afin de mesurer la progression et la qualité, et d'identifier la valeur métier produite. Il en va de même pour les tests en continu. En réalité, il serait plus juste de parler de validation continue, dans la mesure où il s'agit de constamment vérifier que votre application ou service produira (ou est en train de produire) la valeur métier attendue. Parmi les métriques de référence les plus importantes, citons l'adoption client, l'engagement et les revenus. Les équipes doivent comprendre et quantifier le ressenti des utilisateurs finaux, et les ajouter à une boucle de rétroaction continue à destination des développeurs et du métier.

Elles doivent également suivre le nombre de problèmes existants (le nombre de défauts en préproduction et en production) et les évaluer à l'aune de la vitesse de déploiement. Comme la qualité du code augmente progressivement, il peut être utile de suivre le ratio entre le nombre de défauts et la cadence de mise en production, constante ou, mieux encore, croissante.

Combiner la couverture du code et la couverture demandée (couverture de fonctionnalités) permet de suivre la couverture de fonctionnalité globale.

Assurez-vous de comprendre ce que la qualité signifie pour votre entreprise et créez votre propre ligne de référence. Si vous ne savez pas d'où vous partez, vous serez incapable de mesurer votre progression. Utilisez un indicateur clé de performance correspondant à la qualité de la production.

Apprenez le type de couverture nécessaire au niveau des tests unitaires et fonctionnels, de performances et de sécurité. Il ne s'agit plus de penser en termes de tests fonctionnels et non fonctionnels.

Mesurez les temps globaux de production. L'objectif des tests en continu est de réduire les délais de mise en production de composants de meilleure qualité et offrant une valeur métier. Concentrez-vous sur l'augmentation des mises en production tout en réduisant le nombre de défauts de production et en limitant les délais de commercialisation à travers l'automatisation.

Faites évoluer les modes de fonctionnement des personnes, pour qu'elles regardent au-delà des tests réussite/échec en développant une définition alignée de la notion de « travail terminé ». Les défauts doivent être suivis afin que l'analyse permette des améliorations.

Vous pouvez mesurer la réussite de tests en continu en réalisant des tests dans des environnements de test inférieurs, dans lesquels les tests seront plus petits et donc plus rapides. Reconnaissez que les tests peuvent suivre le rythme du développement et que vous pouvez intégrer du code à n'importe quel point du cycle de développement, pas uniquement à la fin du sprint.

Pour évaluer et mesurer la qualité par rapport à la quantité, prenez en compte les éléments suivants :

DÉLAI DE MISE SUR LE MARCHÉ

ANALYSE DES FLUX DE VALEURS

CALENDRIER ET COÛTS D'EXÉCUTION DES TESTS MANUELS

TESTS DE RÉGRESSION

Commencez à déplacer les tests plus en amont vers les développeurs en démocratisant les outils de test et en faisant passer les testeurs au rang d'ingénieurs.

Chaque responsable doit connaître les métriques de tests en continu les plus pertinentes. Les métriques idéales pour faire l'objet d'un suivi sont les suivantes :

Au fur et à mesure que vous passez d'une approche en cascade à une approche agile, vous devrez disposer de tests plus efficaces, ce que permettront l'automatisation et le « shift left ». Cependant, au-delà de l'automatisation et du shift left, le passage à des tests en continu implique de repenser la façon dont vous testez.

Il ne s'agit pas de suivre une méthode intervenant entre intégration continue et livraison continue, mais de développer une compétence en faisant appel à un ensemble d'outils qui rendent possible des manières différentes de tester et de penser les tests.

Vous avez besoin d'outils pour planifier, créer les scripts de test, les automatiser et les implémenter dans les chaînes d'outils d'implémentation d'intégration et de livraison continues.

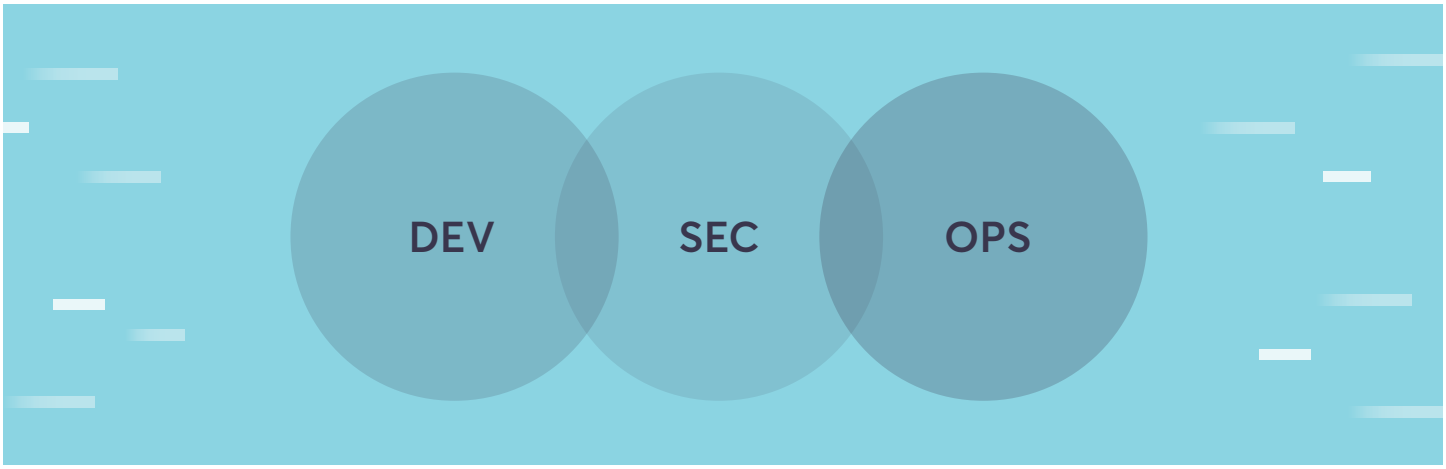
CHAPITRE 8 SÉCURITÉ

Au moment de définir la stratégie d'une transition vers des tests en continu, il est essentiel de pleinement prendre en compte la sécurité. Les spécialistes de la sécurité feront valoir, à raison, que la sécurité a toujours été traitée comme un problème de « fin de chaîne », à gérer en opposition aux développeurs. Alors que les développeurs souhaitent que leurs produits passent rapidement en exploitation, il y a toujours des équipes de sécurité pour leur mettre des bâtons dans les roues.

La sécurité mérite sa place dans le flux de test en continu dès le départ. Il s'agit d'un sujet essentiel dans l'identification et la confirmation de l'ensemble des aspects de la chaîne de valeur, du lancement du projet à la livraison au client. Les boucles de rétroaction vont aider à confirmer si le produit atteint le niveau de qualité attendu et permettre à l'équipe de localiser et de corriger les problèmes de sécurité au fur et à mesure qu'ils émergent.

Les développeurs créent souvent, sans le savoir, des vulnérabilités. Tester en continu la sécurité ne permet pas seulement de corriger ces vulnérabilités ; il s'agit également de déterminer les futures formations à proposer aux développeurs, afin qu'ils soient mieux sensibilisés à la sécurité de leur code et ne créent plus ces vulnérabilités. L'équipe de sécurité doit devenir un partenaire, un conseiller et un expert de confiance, qui donne aux développeurs la formation et les outils permettant une évaluation de leur travail.

Les tests de sécurité doivent survenir à chaque étape du cycle ; les développeurs doivent exécuter des contrôles de sécurité afin de détecter les problèmes dès qu'ils surviennent. Actuellement, la sécurité et les tests se trouvent du mauvais côté du cycle par rapport aux développeurs. Il s'agit de ne plus voir le conflit entre sécurité et qualité, mais d'envisager un partenariat entre les équipes de sécurité et de développement.



Les exigences permettant de définir un niveau de sécurité adéquat doivent être clarifiées. La sécurité est le travail de chacun, mais il convient de déterminer quand ce travail est accompli. L'équipe de développement doit bénéficier de la formation et des outils nécessaires pour comprendre et évaluer ses performances par rapport aux exigences. Les tests doivent être incorporés dans nos pipelines de façon à ce que nous puissions nous évaluer en continu. Comme dans tout processus DevOps, les analyses de sécurité permettent de mesurer l'état actuel pour organiser les formations en amont afin d'améliorer les compétences des équipes. Apprendre aux développeurs à écrire du code correctement dès le début contribuera non seulement à obtenir des résultats plus sûrs, mais aussi à accélérer l'ensemble du processus.

La responsabilité de la sécurité doit également être assumée par les niveaux les plus hauts de la hiérarchie de l'entreprise, qui doivent diffuser les questions et solutions vers les niveaux inférieurs. Pour obtenir une adoption réelle à l'échelle de l'entreprise, vous devez avoir l'adhésion de la direction.

Enfin, l'équipe de sécurité elle-même doit changer de manière de penser afin de garantir que ses membres se considèrent comme des membres à part entière de l'organisation IT de l'entreprise.

CHAPITRE 9 PARÉ POUR L'AVENIR

Un plan de tests en continu doit être évolutif, pour rester pertinent dans une ère de changement et de rapidité incomparable. Pour être paré pour l'avenir, vous devez avoir une stratégie pour tous les aspects du développement, du codage aux tests, qu'ils soient fonctionnels, de performance ou de sécurité. Cela comprend également la gestion de l'environnement, de la génération des données de test à la virtualisation des services.

Il est impératif de maintenir une ligne de vue claire afin de s'assurer que les outils utilisés pour les tests en continu suivent ce même schéma. L'approche de tests en continu doit s'aligner avec la stratégie métier de l'entreprise, et l'organisation IT doit sortir de son rôle de simple soutien pour devenir acteur de cette stratégie métier.

Une entreprise a besoin de ressources dédiées pour accomplir cela. Sans déléguer le tout à des tiers, elle ne doit pas non plus les exclure totalement : il est acceptable de faire appel à des intégrateurs et des fournisseurs de services globaux, mais l'entreprise doit impérativement impliquer ses propres équipes dans le processus. Construisez votre propre expertise, tout en examinant et en renforçant vos relations avec les tiers qui pourraient avoir une solution susceptible d'être intégrée dans votre pipeline, aujourd'hui ou demain.

Identifiez et assemblez des équipes dédiées pour concevoir les plates-formes qui soutiennent votre pipeline d'applications et vos métriques de tests en continu. Vous passerez ainsi d'une organisation employant quelques testeurs manuels et quelques ingénieurs à une organisation qui a davantage de capacité de création.

En passant à des tests en continu, votre organisation verra le nombre de tests par boîte noire baisser et le nombre et la pertinence des tests par boîte blanche augmenter : vos testeurs devront être prêts à opérer cette transition. Celle-ci entraînera la transformation de votre centre d'excellence en un centre de formation, et de la composition de vos équipes et testeurs.

Le Big Data sera essentiel, notamment pour affiner la couverture du code pour tous les types de tests. Le marché est en train d'exploser en ce qui concerne la couverture des tests de régression, notamment au niveau de la mise au point d'une méthode scientifique pour déterminer si ces tests atteignent les niveaux appropriés de couverture du code.

Les entreprises doivent être prêtes et disposées à suivre ce changement. Voici quelques-unes des étapes clés :

L'IA DANS UN RÔLE PLUS IMPORTANT

- Automatisation de l'automatisation, génération automatique de tests à la volée en remplacement de tests de régression impossibles à maintenir.
- Observation en temps réel des utilisateurs de plus en plus importante ; tests requis pour valider les observations.
- Enregistrement et relecture désormais révolus.
- Focus renforcé sur la qualité de la définition des exigences et les nouvelles façons de les analyser pour identifier ce qui manque.

L'équipe de direction doit reconnaître la rapidité avec laquelle le monde autour d'elle change. Les collaborateurs effectuent désormais leur travail avec des technologies mobiles et des applications basées dans le Cloud ; la vitesse est devenue déterminante. Nous évoluons vers une expérience client transparente de bout en bout. Cloud, DevOps, Agile, Big Data : quelles que soient les technologies utilisées, le produit final doit être une expérience client transparente. Vous devez décider de la manière d'appliquer cela au test. C'est là que les deux mondes se rencontrent.

L'utilisateur/consommateur doit percevoir la valeur visée par le métier. C'est ainsi que les tests en continu deviennent des activateurs d'innovation.

Avec l'Internet des objets, les appareils sont connectés de toutes les manières possibles. Pour l'instant, les tests en continu sont des défis pour les navigateurs et les API, mais du fait du développement des appareils mobiles et physiques et de l'Internet des objets, les hubs d'intégration qui permettront de tester en continu deviendront les futurs moteurs de changement. Et là où ils feront défaut, les utilisateurs ne manqueront pas de s'en rendre compte.

Vous devez toujours penser en amont, en faisant notamment appel à l'analyse prédictive.

Les spécialistes de l'assurance qualité et de l'évaluation qualité vont évoluer rapidement. Leurs niveaux de compétence vont s'accroître et les futurs ensembles de compétences nécessaires pour des tests en continu seront différents de ceux d'aujourd'hui. Les entreprises doivent se concentrer sur le mobile et l'omnicanal en priorité.

Celles qui seront les plus audacieuses pourront non seulement gagner des parts de marché en adoptant une culture de test en continu, mais également attirer les meilleurs talents pour les aider.

RÉSUMÉ

Transformer son entreprise pour passer à des tests en continu est un parcours, tout comme le passage à une démarche agile ou DevOps. Les entreprises et leurs équipes IT essuieront des revers qu'elles doivent anticiper et accepter. C'est pourquoi la planification est essentielle à chaque étape du parcours.

Les tests en continu sont autant une culture qu'une technique. L'ensemble des équipes peuvent apprendre des échecs, dans une approche « fail-forward », en éliminant les silos du passé. Tous les grands développements et les victoires personnelles sont précieux dans la mesure où ils s'appuient sur des fondations solides, mêlant connaissances et expérience.

Bien que les tests en continu s'appuient sur les technologies, les êtres humains sont au cœur du processus. Une fois qu'ils seront à l'aise et en confiance, ils pourront fournir l'énergie, la vision et les capacités pratiques nécessaires pour les déployer.

Grâce à cela, l'entreprise bénéficiera de suffisamment de soutien pour naviguer avec réussite dans cette nouvelle économie globale.

CS200-383435_0818