Aller plus vite avec la distribution continue

Mark Mansour

Amélioration continue et automatisation logicielle

Il y a plus de 10 ans, nous avons entrepris chez Amazon un projet visant à comprendre la rapidité avec laquelle nos équipes transformaient leurs idées en systèmes de production de haute qualité. Cela nous a amenés à mesurer le rendement de nos logiciels afin d'améliorer la vitesse d'exécution. Nous avons découvert qu'il fallait en moyenne 16 jours entre l'enregistrement du code et la mise en production. Chez Amazon, les équipes commençait avec une idée, puis mettaient généralement un jour et demi à écrire du code pour la concrétiser. Il nous fallait moins d'une heure pour créer et déployer le nouveau code. Le reste du temps, presque 14 jours, nous le passions à attendre que les membres de l'équipe commencent une génération, effectuent des déploiements et exécutent des tests. À la fin du projet, nous avons recommandé d'automatiser nos processus de post-vérification afin d'améliorer notre vitesse d'exécution. L'objectif visait à éliminer les retards tout en maintenant, voire améliorant la qualité.

Au cœur de cette recommandation figurait un programme d'amélioration continue pour accélérer notre exécution. Nous avons fondé notre décision d'améliorer notre vitesse d'exécution sur le principe de leadership Insistance sur les normes les plus élevées. Ce principe consiste à avoir des normes élevées sans relâche, à relever constamment la barre et à fournir des produits, des services et des processus de haute qualité. Nos principes de leadership décrivent comment Amazon gère les affaires, comment les dirigeants dirigent et comment nous plaçons le client au centre de nos décisions.

Amazon avait déjà créé des outils de développement logiciel pour améliorer la productivité de ses ingénieurs logiciels. Nous avons créé notre propre système de compilation centralisé et hébergé, Brazil, qui exécute une série de commandes sur un serveur dans le but de générer un artefact qui puisse être déployé. À ce stade, Brazil ne tenait pas compte des modifications du code source. Une personne devait lancer une génération. Nous disposions également notre propre système de déploiement, Apollo, qui nécessitait de charger un artefact de construction lors du lancement d'un déploiement. Inspirés par l'intérêt suscité par la distribution continue dans le secteur, nous avons construit notre propre système, Pipelines, pour automatiser le processus de distribution de logiciels entre le Brazil et Apollo.

Pipelines : notre outil de déploiement continu

Nous avons lancé un programme pilote visant à automatiser le processus de distribution de logiciels pour un petit nombre d'équipes. Au moment où nous avions terminé, l'équipe pilote principale avait réussi à réduire de 90 % le temps total nécessaire pour passer de la vérification à la production.

Le projet a validé le concept de pipeline en tant que moyen permettant à nos équipes de définir toutes les étapes nécessaires à la publication de logiciels pour les clients. La première étape d'un pipeline consiste à créer un artefact. Le pipeline exécute ensuite cet artefact de génération à travers une série d'étapes jusqu'à ce qu'il soit publié pour tous les clients. Nous utilisons des pipelines pour réduire le risque qu'un nouveau changement de code ait un impact négatif sur nos clients. Chaque étape dans le pipeline doit nous assurer que l'artefact de génération ne

contient pas de défauts. Si un défaut atteint la production, nous voulons remettre la production dans un état sain le plus rapidement possible.

Lorsque nous avons lancé Pipelines, il ne pouvait modéliser qu'un seul processus de publication par application. Cette limitation a favorisé la cohérence, la standardisation et la simplification des processus de publication d'une équipe. Cela a eu pour effet de réduire le nombre de défauts. Avant d'utiliser des pipelines, il était courant que les équipes utilisent des processus de publication différents pour les corrections de bogues et les mises à jour majeures. Tandis que les autres équipes voyaient le succès des équipes qui avaient piloté la distribution automatisée, elles ont commencé à migrer leurs processus de publication gérés manuellement vers des pipelines afin d'améliorer également la cohérence. Les équipes qui utilisaient différents processus de publication utilisaient désormais un processus normalisé utilisé par tout le monde. En outre, lorsqu'ils ont transféré leurs processus de publication dans un outil, les membres de l'équipe ont souvent revu leur approche et trouvé des moyens de simplifier leur processus.

L'équipe Pipelines avait pour objectif annuel d'augmenter l'utilisation par le biais de l'« adoption séduisante ». En d'autres termes, ils devaient rendre le produit tellement bon que les gens l'exigeraient. Nous avons mesuré le nombre d'équipes utilisant un pipeline pour déployer leur logiciel en production, et nous avons classé les pipelines en fonction de leur niveau d'automatisation. Nous avons constaté que des équipes cherchaient à utiliser un pipeline pour publier les logiciels et à passer à des versions entièrement automatisées. Cependant, nous avons remarqué que dans certaines entreprises, la façon dont nous mesurions la qualité pouvait amener les équipes à automatiser leur processus de publication sans prévoir aucun test.

La réponse à la question « combien de tests sont suffisants ? » est jugement personnel. Il faut une équipe pour comprendre le contexte dans lequel ils opèrent. Pour faire face à cette situation, nous avons utilisé un autre principe de leadership, la propriété. Ce principe consiste à penser à long terme et à ne pas sacrifier la valeur à long terme pour des résultats à court terme. Les équipes logicielles d'Amazon ont des exigences élevées en matière de tests et y consacrent beaucoup d'efforts, car posséder un produit revient également à assumer les conséquences de tout défaut de ce produit. Si un problème devait avoir un impact sur les clients, ce sont les membres de la petite équipe logicielle responsable qui gèrent le problème et le résolvent en temps réel. La tension entre l'accélération de la vitesse d'exécution et la résolution des problèmes de production signifie que les équipes sont motivées pour effectuer des tests adéquats. Cependant, si nous investissons trop dans les tests, nous risquons de ne pas réussir, car d'autres ont progressé plus rapidement que nous. Nous cherchons toujours à améliorer nos processus de publication de logiciels sans devenir un obstacle pour l'entreprise.

Un autre problème auquel nous avons été confrontés est que les équipes n'apprenaient pas entre elles les meilleures pratiques en matière de publication de logiciels. Les équipes responsables sont encouragées à travailler de manière autonome, ce qui signifie que les ingénieurs résolvent leurs problèmes de déploiement de manière indépendante. Lorsqu'ils ont trouvé une solution répondant à leurs besoins en termes de publication de logiciels, ils ont fait connaître la technique à d'autres ingénieurs via des listes de diffusion, des réunions d'exploitation et d'autres canaux de communication. Ce style de communication posait deux problèmes. Premièrement, ces canaux constituent un canal de communication au mieux, ce qui

signifie que tout le monde n’a pas appris les nouvelles techniques. Deuxièmement, les dirigeants encourageant leurs équipes à adopter les nouvelles meilleures pratiques n’avaient aucun moyen de savoir si leurs équipes avaient fait le travail nécessaire pour adopter réellement les meilleures pratiques. Nous avons réalisé que nous devions aider tous les ingénieurs à accéder aux meilleures pratiques que nous avions apprises et donner aux dirigeants la possibilité d’identifier les pipelines qui nécessitaient une attention.

Notre solution consistait à mécaniser notre apprentissage en ajoutant des vérifications des meilleures pratiques aux outils que nous utilisons pour créer et publier les logiciels. Nous étions sensibles au fait qu'une bonne pratique pour une organisation pouvait ne pas l'être pour une autre. Nous avons donc permis à ces vérifications d'être configurées pour chaque organisation. Les vérifications des meilleures pratiques au niveau organisationnel ont donné aux responsables la possibilité d'adapter leurs processus de publication pour répondre aux besoins de leur entreprise. Les responsables souhaitant encourager ou appliquer une nouvelle meilleure pratique ont pu commencer par émettre un avertissement à partir des outils utilisés quotidiennement par les ingénieurs. En plaçant des messages dans les outils, il était presque certain que les membres de l’équipe seraient informés de la meilleure pratique et de la date à laquelle elle entrerait en vigueur. Nous avons constaté qu'en donnant aux équipes le temps d'apprendre et de débattre d'une nouvelle meilleure pratique, une organisation avait la possibilité d'itérer et d'améliorer ses vérifications. Cela a finalement permis d'améliorer la qualité des meilleures pratiques et une meilleure adhésion de la communauté des ingénieurs.

Nous avons systématiquement identifié les meilleures pratiques à appliquer. Un groupe de nos ingénieurs les plus expérimentés a répertorié les raisons courantes pour lesquelles une version ne fonctionne pas. Ils ont identifié les étapes qui auraient facilité la publication. Nous avons ensuite utilisé cette liste pour élaborer notre ensemble de contrôles des meilleures pratiques. Au cours de ce processus, nous nous sommes rendu compte que, même si nous souhaitions que les nouvelles révisions logicielles atteignent les clients instantanément, sans effort et sans dégrader la disponibilité, nous accordons la priorité à la disponibilité, puis à la rapidité et enfin à la simplification de la tâche pour nos ingénieurs.

Réduire le risque qu'un défaut atteigne les clients

Il est probable que tous les ingénieurs introduisent, à un moment donné, un défaut dans l’un de nos systèmes. Nos processus de publication, y compris nos pipelines et nos systèmes de déploiement, doivent être conçus pour identifier ces défauts le plus rapidement possible et éviter qu’ils affectent négativement nos clients. Nous devons nous assurer que nos processus de publication sont configurés correctement et que notre artefact de génération fonctionne comme prévu.

Hygiène de déploiement : la forme la plus élémentaire de test de déploiement garantit que l'artefact nouvellement déployé peut être démarré et peut répondre aux tâches. Dans le cadre du flux de travail post-déploiement, nous effectuons des vérifications rapides pour nous assurer que l'artefact nouvellement déployé a démarré et traite le trafic. Par exemple, nous utilisons des points d'ancrage Lifecycle dans le fichier AWS CodeDeploy AppSpec pour déclencher des scripts simples permettant d'arrêter, de démarrer et de valider le déploiement. Nous vérifions également que nous avons suffisamment de capacité pour traiter le trafic client. Nous avons

développé des techniques telles que des hôtes minimaux sains dans CodeDeploy pour vérifier que nous avons toujours une capacité suffisante pour servir nos clients. Enfin, si le moteur de déploiement peut détecter une défaillance, il doit annuler la modification afin de minimiser le temps pendant lequel les clients voient un défaut.

Tests avant la production : l'une des meilleures pratiques d'Amazon consiste à automatiser les tests unitaires, d'intégration et de pré-production, puis à les ajouter à notre pipeline. Nous insistons pour que nous effectuions des tests de charge et de sécurité, en privilégiant l'ajout de ces essais dans nos pipelines. Par tests unitaires, nous entendons tous les tests que vous pourriez vouloir effectuer sur votre machine de génération, notamment les vérifications de style, la couverture du code, la complexité du code, etc. Les tests d'intégration sont pour nous des tests incluant tous les tests prêts à l'emploi, tels que l'injection de défaillances, les tests de navigation automatisés et similaires. Il existe de nombreux articles intéressants sur les tests unitaires et d'intégration, et je ne vais donc pas entrer plus dans les détails ici.

Nos tests unitaires et d'intégration visent à vérifier que le comportement de notre artefact de construction est correct. Plus nous effectuons de validations, plus le risque qu'un défaut soit exposé à nos clients est faible. Afin de réduire le temps nécessaire pour remettre un produit à nos clients, nous essayons de détecter un défaut le plus tôt possible dans le processus de validation. En général, cela signifie que si vos tests sont plus petits et plus rapides, vous recevez rapidement un retour d'information sur tous les problèmes liés à vos modifications.

Chez Amazon, nous utilisons également une technique appelée *test de pré-production*. Un environnement de pré-production est le dernier endroit où les tests ont lieu avant de déployer nos modifications en production. Un test dans l'environnement de pré-production utilise la configuration de production du système pour qu'il fonctionne exactement comme un système de production. Cette approche offre deux avantages. Tout d'abord, les environnements de pré-production testent la configuration de production pour s'assurer que le service peut se connecter correctement à toutes les ressources de production, y compris les magasins de données de production. Deuxièmement, cela garantit que le système interagit correctement avec les API des services de production dont il dépend. Les environnements de pré-production ne sont utilisés que par l'équipe propriétaire de ce service, et le trafic des clients ne leur est jamais envoyé. L'exécution de tests de pré-production augmente notre confiance dans le fait que le même code et la même configuration fonctionneront en production.

Validation en production : lorsque nous fournissons du code à nos clients, nous ne le faisons pas d'un seul coup. L'ampleur de l'impact de la diffusion d'un défaut vers tous les clients à la fois est trop grande. Au lieu de cela, nous déployons dans des cellules, une instance complètement indépendante d'un service. Lorsque nous déployons des modifications vers notre premier groupe de clients dans notre première cellule, nous sommes extrêmement prudents. Nous laissons seulement un petit nombre de clients voir le nouveau changement, et nous collectons des informations pour savoir si le nouveau code fonctionne. Nous surveillons le nombre d'erreurs émises par nos services après un déploiement de version Canary. Si le taux d'erreur augmente, nous annulons automatiquement la modification. Par exemple, nous pouvons attendre 3 000 points de données positifs, sans aucun point de données négatif, avant de poursuivre un déploiement.

Une complication peut survenir si vos tests automatisés manquent un cas d'utilisation. Nous nous efforçons de détecter toutes les erreurs avec nos tests structurés et répétables, qu'ils soient automatisés ou manuels. Cependant, même lorsque nous faisons de notre mieux, un défaut peut passer à travers. Pour tester nos tests, nous laissons le nouveau changement en production pendant une période donnée pour voir si un non-membre de l'équipe trouve un problème. Nous avons passé beaucoup de temps à débattre de la question de savoir si nous devions laisser les modifications en production ou combien de temps nous devions attendre après le déploiement d'une version Canary avant de déployer vers le reste du groupe de déploiement. La plupart de nos équipes ont décidé d'attendre un certain temps en plus de recueillir des points de données positifs avant de passer à notre routine de déploiement. La durée d'attente d'un pipeline dépend fortement de l'équipe. Certaines équipes attendent pendant des heures et d'autres attendent quelques minutes. Plus l'impact est important et plus le temps de résolution du problème est long, plus le processus de publication est lent.

Une fois que les résultats dans la première cellule sont convaincants, nous exposons le nouveau changement de code à un plus grand nombre de clients jusqu'à ce qu'il soit complètement publié. Comme nous l'avons fait avec le déploiement de la version Canary, nous attendons de savoir que le déploiement dans la première nouvelle cellule est convaincant avant de passer à la cellule suivante. À mesure que notre confiance en l'artefact de construction augmente, nous réduisons le temps de vérification du changement de code. Il en résulte un modèle dans lequel nous visons à passer de la vérification à notre premier client de production le plus rapidement possible. Cependant, une fois que nous sommes en production, nous publions lentement le nouveau code à l'intention des clients, dans le but d'accroître la confiance à mesure que nous accélérons progressivement le reste de nos déploiements.

Pour vérifier que nos systèmes de production continuent de répondre aux besoins de nos clients, nous générons un trafic synthétique sur nos systèmes. Nous voulons des retours rapides si notre service ne fonctionne pas correctement. Nous effectuons donc nos tests synthétiques au moins toutes les minutes. Nous concevons des tests synthétiques pour nous assurer que nos processus en cours d'exécution sont sains et que toutes les dépendances sont testées, ce qui implique souvent de tester toutes les API destinées au public.

Contrôle du moment de publication du logiciel : pour contrôler la sécurité de nos versions logicielles, nous avons mis au point des mécanismes nous permettant de contrôler la vitesse à laquelle les modifications se produisent dans notre pipeline. Nous utilisons des métriques, des fenêtres de temps et des contrôles de sécurité pour contrôler la publication de notre logiciel.

Les pipelines peuvent être configurés pour empêcher un déploiement lorsqu'une alarme est déclenchée en fonction d'un changement de métrique. Nous utilisons systématiquement des métriques et appliquons des alarmes sur l'état de nos systèmes, des cellules, des zones de disponibilité, des régions et tout ce que vous pouvez imaginer. Nous configurons nos pipelines pour arrêter le déploiement de code lorsqu'une métrique importante déclenche une alarme. Cependant, une équipe doit parfois déployer un correctif pour que l’alarme du système soit traitée. Pour ce scénario, nous permettons aux équipes de remplacer les alarmes afin d’empêcher les modifications de passer par un pipeline.

Nos pipelines peuvent spécifier une fenêtre de temps dans laquelle les modifications sont autorisées à progresser dans un pipeline. Les équipes peuvent trouver leurs propres fenêtres de temps pour limiter le moment où les modifications atteignent les clients. Les équipes AWS préfèrent publier des logiciels lorsque de nombreuses personnes sont en mesure de réagir rapidement et de réduire les problèmes causés par un déploiement. Pour que cela devienne une réalité, les équipes définissent généralement leurs créneaux horaires de manière à ne déployer que pendant les heures ouvrables. D'autres équipes d'Amazon préfèrent publier des logiciels lorsque le trafic client est faible. Ces fenêtres de temps peuvent être remplacées si nécessaire.

Nous pouvons également arrêter un pipeline en fonction du contenu de l'artefact de construction. Par exemple, nous pouvons bloquer un artefact de construction contenant un empaquetage ou une référence Git incorrects. Nous avons utilisé cette fonctionnalité lorsque nous avons découvert qu’une modification apportée à un empaquetage contenait une régression de performance. Si nous ne supprimions que l'empaquetage de notre référentiel d'empaquetages, les pipelines contenant déjà l'empaquetage défectueux déploieraient néanmoins cette modification incorrecte chez les clients.

Notre approche de la rapidité de notre exécution

Nous avons constaté que les équipes étaient impatientes d'adopter l'automatisation. Nous sommes tous extrêmement motivés pour créer et proposer à nos clients des fonctionnalités qui améliorent leur vie. La distribution continue en fait une solution durable. Nous savons que l’automatisation permet aux ingénieurs de gagner du temps en supprimant les tâches manuelles frustrantes, propices aux erreurs et fastidieuses. Nous avons vu que le déploiement continu avait un impact positif sur la qualité. Nous avons constaté que l'automatisation permet aux équipes de publier fréquemment, un changement à la fois, pour faciliter l'identification des régressions.

Lorsqu'il s'agit de nouveaux systèmes, la plupart des membres de l’équipe comprennent bien la surface à tester, ce qui rend certains tests manuels réalisables. Cependant, à mesure que les systèmes deviennent plus complexes et que les membres de l'équipe changent, la valeur de l'automatisation augmente. Nous aimons automatiser nos systèmes afin de pouvoir nous concentrer sur l'ajout de valeur client plutôt que de gérer manuellement le processus de transmission de ces modifications aux clients.

Pendant de nombreuses années, Amazon a mis en place des programmes d’amélioration continue axés sur la rapidité avec laquelle nous publions des logiciels aux clients et sur la sécurité de ces versions. Nous n’avons pas commencé avec tous les tests de contrôle des risques et les tests dont j’ai parlé dans cet article. Au fil du temps, nous avons inventé des moyens d’identifier et de réduire les risques.

Les programmes d'amélioration continue sont gérés par les responsables de l'entreprise à différents niveaux. Ainsi, chacun d'entre eux peut adapter son processus de publication de logiciels en fonction des risques et de l'impact sur son activité. Certains de nos programmes d’amélioration continue s’appliquent à de grands secteurs d’Amazon, et parfois les responsables dans les petites entreprises gèrent leurs propres programmes. Nous savons qu'il

existe toujours des exceptions à la règle. Nos systèmes disposent de mécanismes d'exclusion afin de ne pas ralentir les équipes qui ont besoin d'une exemption permanente ou temporaire. En fin de compte, nos équipes sont propriétaires du comportement de leurs logiciels et sont responsables d'investir de manière appropriée dans leur processus de publication.

Nous avons commencé par identifier notre problème, y remédier et itérer. Pour rendre ce travail durable, nous devions le faire progressivement et saluer les améliorations apportées au fil du temps. Lorsqu'Amazon a commencé à utiliser des pipelines, de nombreuses équipes n'étaient pas certaines que le déploiement continu fonctionnerait. Pour aider les équipes à démarrer, nous les avons encouragées à coder leur processus de publication actuel, étapes manuelles comprises, dans un pipeline. Pour la plupart des équipes, leur pipeline a servi d'interface graphique avec leur processus de publication sans promouvoir automatiquement les artefacts de construction via un processus de publication. À mesure que la confiance s'installait, ils ont progressivement automatisé les différentes étapes de leur pipeline jusqu'à ce qu'ils n'aient plus besoin de déclencher manuellement une étape de leur pipeline.

Des avancées rapides jusqu'à aujourd'hui. Amazon est arrivé au point où les équipes visent une automatisation complète lorsqu'elles écrivent un nouveau code. L'automatisation est pour nous le seul moyen de continuer à développer notre activité.