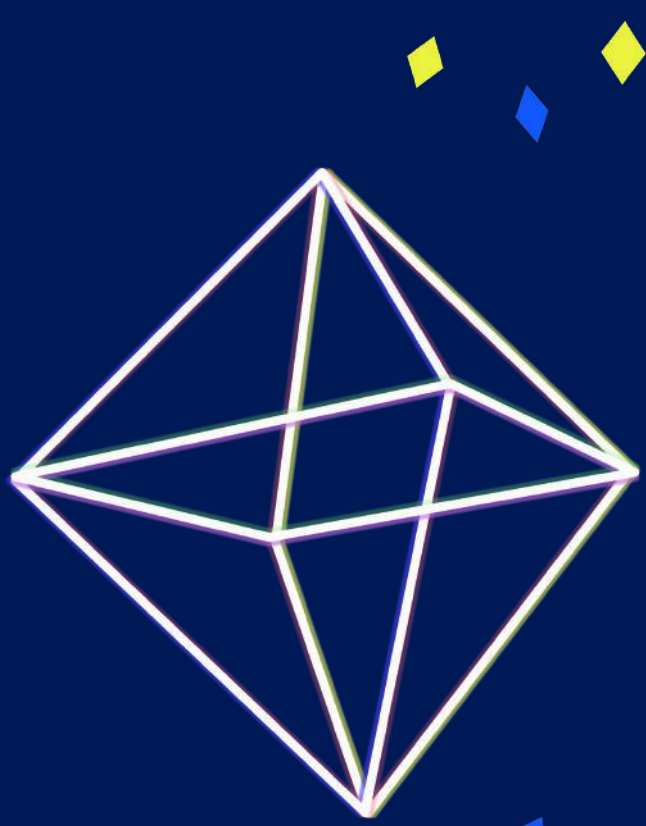
THE STATE OF

Software Engineering Excellence 2025



harness

THE STATE OF
Software
Engineering
Excellence
2025

harness

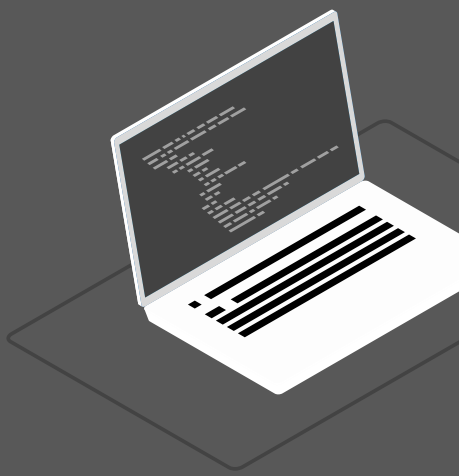
Le fossé de l'excellence engineering est bien réel. Et il vous coûte des millions.

La panne mondiale de CrowdStrike. La faille de la chaîne logistique de SolarWinds. L'attaque par ransomware de Colonial Pipeline. Ce ne sont pas des incidents isolés, mais les symptômes d'une crise systémique dans la manière dont nous concevons, déployons et sécurisons les logiciels. Alors que les gros titres s'emballent sur la transformation par l'IA, la vérité est plus dérangeante : la plupart des équipes d'ingénierie sont submergées par la dette technique, les processus manuels et les vulnérabilités de sécurité qui auraient dû être résolus il y a une décennie.

Mais voici ce qui est fascinant : les entreprises qui excellent ne se contentent pas d'éviter les catastrophes, elles libèrent aussi des millions de valeur cachée. Elles livrent des fonctionnalités 60 % plus rapidement, réduisent leurs coûts de cloud de 15 % et font chuter les incidents de sécurité à un niveau quasi nul. La différence ne réside ni dans le talent ni dans le budget. Elle vient du fait de savoir précisément où se trouvent les lacunes et de les combler de manière systématique

Ce rapport révèle la réalité de la maturité en ingénierie en 2025, à partir des réponses franches de plus de 650 leaders du secteur qui ont complété l'« Engineering Excellence Maturity Assessment », un outil développé par EngineeringX, une communauté de centaines de CTO et de VP Engineering. Ce que nous avons découvert va vous surprendre : même les organisations qui se considèrent comme "DevOps matures" manquent à des pratiques fondamentales qui séparent les leaders de l'industrie du reste du marché.

Les cinq dimensions que nous avons mesurées, l'expérience développeur, la modernisation DevOps, l'optimisation, la qualité et la résilience, et le développement logiciel sécurisé, révèlent où se trouvent les plus grandes opportunités. En examinant ces données complètes, ce rapport vise à mettre en lumière l'adoption des meilleures pratiques en matière d'excellence en ingénierie, à identifier les défis courants et à souligner des domaines d'amélioration concrets alors que les entreprises poursuivent leur chemin crucial vers l'excellence en ingénierie.



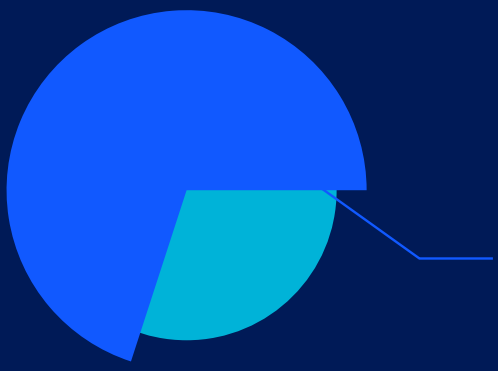
01: Developer Experience

L’expérience développeur est un élément clé de performance pour les organisations digital-first. C’est aussi un levier essentiel pour garder les équipes tech engagées et efficaces. Mettre en place une culture engineering mature passe avant tout par une excellente expérience développeur, ce qui implique de leur fournir les bons outils, les bonnes pratiques et un environnement de travail qui leur permet d’avancer vite et bien. Concrètement, cela signifie optimiser chaque point de contact dans leur quotidien : des environnements faciles à provisionner, une documentation claire et accessible, une planification sans friction, et des process de qualité bien intégrés.

Processus de planification et de définition des besoins

Une planification efficace et des besoins clairs sont les piliers d'une bonne expérience développeur. Lorsque les spécifications sont ambiguës ou qu'elles changent constamment, les ingénieurs sont confrontés à de la frustration, du travail en double et un sentiment de perte d’accomplissement. À l'inverse, un processus de planification bien défini, incluant des critères d'acceptation précis, réduit les frictions, renforce la confiance et permet aux équipes de développement de se concentrer sur la création de valeur plutôt que de déchiffrer des objectifs mal définis.

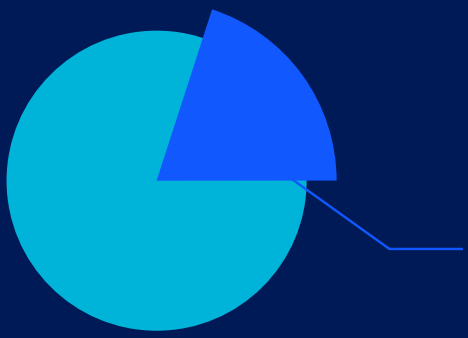
Pourtant, de nombreuses organisations rencontrent des difficultés :



70 %

1 leader de l'ingénierie sur 4 indique que

plus de 70 % des besoins n'ont pas de critères d'acceptation clairement définis.



20 %

Plus de la moitié (54 %) des leaders

de l'ingénierie ont constaté un délai moyen de plus de 20 % lié à un “scope creep” au cours des 12 derniers cycles de sprint.

Pour améliorer la maturité de votre processus de planification, donnez la priorité à la définition de critères d’acceptation clairs et cohérents pour toutes les fonctionnalités. Mettez en œuvre des outils et des pratiques qui favorisent la communication entre les équipes produit et ingénierie, réduisant ainsi l'ambiguïté et garantissant un périmètre prévisible.

Découvrabilité et Documentation

Les équipes d'ingénierie ont besoin d'un accès immédiat à des informations précises sur les logiciels et les services sur lesquels elles travaillent. Cela leur permet de prendre des décisions éclairées et de suivre une méthodologie de développement cohérente. Cependant, elles sont aussi sous pression pour livrer rapidement, et les leaders de l'ingénierie doivent s'assurer que ces processus ne créent pas de travail supplémentaire pour les développeurs. Un catalogue logiciel complet, documentant efficacement les métadonnées, les propriétaires et le statut des services, est crucial pour réduire les frictions et permettre aux ingénieurs de se concentrer sur l'innovation.

Ce point représente un défi de taille dans l'ensemble du secteur :

Seulement 21 % des équipes d'ingénierie disposent d'un catalogue logiciel qui est mis à jour automatiquement en cas de changement.

Près d'un tiers (29 %) des équipes n'ont aucun catalogue logiciel.

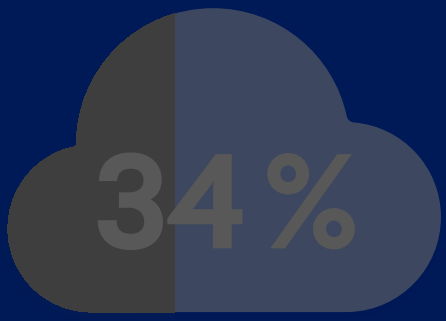
Pour améliorer la maturité en matière de découvrabilité et de documentation, mettez en place un catalogue logiciel automatisé, idéalement via un Portail Développeur Interne (IDP). Cette approche centralise les métadonnées, les propriétaires et le statut des services, garantissant une précision en temps réel sans effort manuel de la part des développeurs.

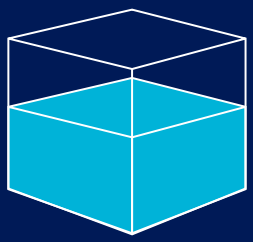
Environnements de développement

Les entreprises qui visent une livraison logicielle optimale savent à quel point des environnements de développement efficaces sont cruciaux. Les retards dans leur mise en place ou les incohérences entre les configurations freinent directement la productivité, ralentissent les cycles d'itération et ajoutent des frictions inutiles au flux de travail de développement. Simplifier la manière dont les équipes accèdent à des environnements standardisés et prêts à l'emploi est essentiel pour accélérer la livraison des fonctionnalités et maintenir la vélocité des développeurs.

Malgré leur importance, la mise à disposition des environnements reste un défi persistant :

Seulement un tiers (34 %) des équipes d'ingénierie peuvent créer rapidement des environnements de développement préconfigurés dans le cloud.





67 % des leaders de l'ingénierie affirment que leurs développeurs ne peuvent pas créer et tester un environnement de développement en moins de 15 minutes.

Pour améliorer la maturité de votre processus de développement, concentrez-vous sur la mise à disposition d'outils intuitifs et en libre-service pour que les développeurs puissent créer rapidement des environnements aux configurations cohérentes. Cela les libère de la surcharge du paramétrage manuel et leur permet de se concentrer sur les tâches à forte valeur ajoutée. Une approche efficace consiste à intégrer ce provisionnement automatisé des environnements dans un Portail Développeur Interne (IDP).

Processus de développement et hygiène du code

Des processus de développement solides et une hygiène de code rigoureuse sont fondamentaux pour garantir une expérience développeur productive et satisfaisante. Lorsque les revues de code deviennent des goulots d'étranglement et que la taille des commits devient ingérable, c'est tout le cycle de développement qui ralentit, affectant à la fois l'efficacité et la capacité à maintenir des logiciels de haute qualité. S'attaquer à ces éléments de base est crucial pour accélérer l'innovation et assurer des livraisons fiables.

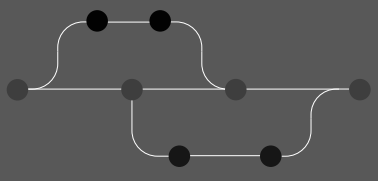
Les données du secteur révèlent des défis courants dans ce domaine :

61 %

des responsables de l'ingénierie déclarent que les revues de code prennent plus d'une journée dans leur organisation.

35 %

des équipes d'ingénierie ne suivent pas systématiquement la stratégie de branchement pour leurs dépôts de QA, de développement et d'infrastructure.



14 %

des responsables de l'ingénierie indiquent que leur commit de code moyen est inférieur à 30 lignes.

Pour améliorer la maturité de vos processus de développement et l'hygiène de votre code, donnez la priorité à l'efficacité des workflows de revue de code et encouragez des commits plus petits et plus fréquents. Mettez en œuvre des vérifications automatisées et des stratégies de branchement claires pour maintenir la qualité du code et accélérer l'intégration sans surcharger les développeurs.

Apprentissage et développement

Pour offrir une excellente expérience développeur, les leaders de l'ingénierie doivent s'assurer que leurs équipes disposent d'opportunités de développement personnel et d'acquisition de nouvelles compétences. Investir dans la formation continue permet aux équipes de rester à la pointe de la technologie, prévient la stagnation des compétences et a un impact significatif sur le moral et la rétention des talents.

Pourtant, de nombreuses organisations font face à un manque dans ce domaine :

19 %

Seulement 19 % des leaders de l'ingénierie déclarent avoir un programme structuré pour l'amélioration et la reconversion des compétences de leurs ingénieurs.

Pour renforcer vos initiatives d'apprentissage et de développement, les leaders devraient se concentrer sur l'établissement de programmes clairs et de parcours d'apprentissage prédéfinis. Ces programmes, qui incluent des formations internes et des certifications externes, sont cruciaux pour une montée en compétence efficace et pour s'assurer que les ingénieurs développent constamment leurs capacités.

Ce qu'il faut retenir

Améliorer l'expérience développeur grâce à une meilleure documentation, une planification rigoureuse et un provisionnement des environnements plus fluide accélère l'intégration des nouvelles recrues. En équipant les ingénieurs d'outils et de flux de travail efficaces, le temps nécessaire aux nouvelles recrues pour atteindre leur pleine productivité est considérablement réduit.

Pour une entreprise de 1 000 développeurs, cela représente des économies significatives :

Formula:

2.4 M$	par an de productivité perdue à cause d'un processus d'onboarding trop long.
40-60 %	Un onboarding 40 à 60 % plus rapide grâce aux bonnes pratiques en matière d'expérience développeur.
1 M$ à 1,4 M$	d'économies annuelles, rien que sur les coûts liés à l'onboarding.

02: Modernisation DevOps

La modernisation DevOps ne se résume pas à l'adoption de nouveaux outils ; c'est un impératif stratégique pour transformer la manière dont les logiciels sont conçus, livrés et sécurisés. C'est le socle qui permet d'établir une pratique d'ingénierie véritablement mature.

Cette transformation englobe la simplification des processus critiques, des pipelines de build et de l'automatisation des déploiements jusqu'aux stratégies avancées et à la sécurité renforcée. L'objectif est de gagner en efficacité, de réduire les risques et d'accélérer le rythme de l'innovation. En s'attaquant à ces fondations, les organisations peuvent améliorer de manière significative leur stabilité opérationnelle et leur productivité, posant ainsi les bases d'une pratique d'ingénierie mature.

Processus de build

Des processus de build standardisés sont essentiels pour maintenir la qualité et la sécurité des logiciels tout en permettant aux développeurs d'accélérer la livraison grâce à des pratiques DevOps modernes. Ces pipelines doivent inclure des contrôles qualité automatisés pour empêcher le code défectueux d'atteindre la production et pour réduire les changements de contexte pour les développeurs. Cela contribue en bout de ligne à un cycle de livraison logicielle plus efficace et plus fiable.

Malgré cette importance, de nombreuses organisations font face à des obstacles significatifs :

Plus d'un tiers (36 %) des organisations ne disposent pas de pipelines de build standardisés.

Alors que 56 % des organisations exécutent tous leurs cas de test dans le cadre du pipeline de build, seulement 17 % d'entre elles sélectionnent intelligemment les tests en fonction des modifications de code applicables.

55 % des pipelines de build ne sont pas sécurisés par des contrôles de qualité, ou ces derniers ne sont pas appliqués de manière stricte.

Pour gagner en maturité dans votre processus de build, concentrez-vous sur l'établissement de pipelines standardisés et basés sur des modèles, qui intègrent des contrôles qualité automatisés. Ces "portes de qualité" doivent lancer intelligemment les tests pertinents et appliquer des règles strictes pour empêcher la progression de code non conforme ou défectueux, améliorant ainsi la qualité et l'efficacité des développeurs.

Processus de déploiement

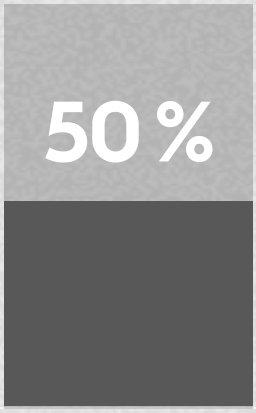
Le processus de déploiement est le point de jonction essentiel entre le développement et les opérations. Il définit de manière fondamentale la vitesse et la fiabilité d'une organisation dans sa démarche de modernisation DevOps. Un pipeline de déploiement optimisé est indispensable pour livrer des logiciels de manière rapide et cohérente, en s'assurant que les nouvelles fonctionnalités et les correctifs atteignent la production de manière sécurisée et efficace. Cette section explore les différentes facettes du déploiement moderne, de l'automatisation aux stratégies de rollback et à la sécurité.

Automatisation du déploiement

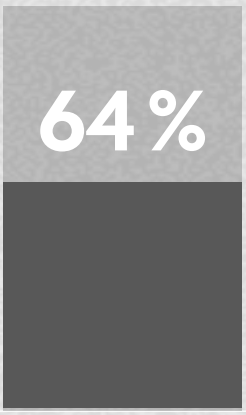
Des processus de déploiement standardisés et automatisés sont la pierre angulaire d'une modernisation DevOps efficace, optimisant directement le cycle de livraison. Lorsque les étapes manuelles sont réduites au minimum, les organisations peuvent accélérer les mises en production, diminuer les erreurs humaines et libérer un temps précieux pour l'innovation. L'automatisation des étapes de déploiement permet une livraison de logiciels cohérente, reproductible et rapide, ce qui est crucial pour la vélocité du développement moderne.

Malgré cet avantage évident, une part importante d'intervention manuelle persiste :

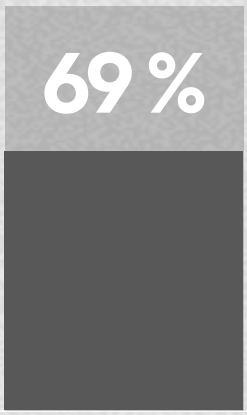
Moins d'une équipe d'ingénierie sur cinq dispose de pipelines de déploiement standardisés ou basés sur des modèles.



des déploiements de code applicatif dépendent encore d'étapes manuelles.



des changements de base de données reposent encore sur des étapes de déploiement manuelles.



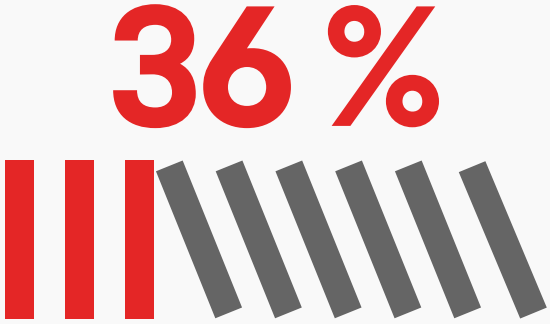
des déploiements de code d'infrastructure dépendent toujours d'étapes manuelles.

Pour améliorer la maturité de votre automatisation du déploiement, concentrez-vous sur la standardisation et la mise en place de modèles pour tous les pipelines. Donnez la priorité à l'automatisation de chaque étape possible, du code à l'infrastructure et aux changements de base de données, afin d'éliminer l'intervention manuelle et de réaliser des mises en production cohérentes, rapides et fiables.

Deployment strategies

Les stratégies de déploiement modernes comme les rolling updates, le déploiement canary, les approches blue/green, GitOps ou encore les feature flags sont fondamentales pour atteindre une maturité d'ingénierie. Mettre en œuvre ces pratiques permet aux organisations de mieux contrôler les risques liés aux livraisons, d'effectuer des déploiements par étapes et de garantir la stabilité en production. Tous ces éléments sont cruciaux pour une modernisation DevOps efficace.

Pourtant, l'adoption de ces stratégies reste un défi :



des équipes d'ingénierie n'utilisent pas de stratégies de déploiement telles que les rolling updates, canary, blue/green ou GitOps.

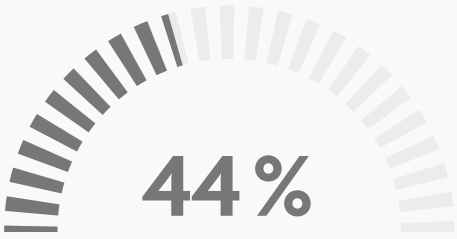
des équipes d'ingénierie n'utilisent pas de feature flags ou n'ont pas de processus défini pour leur déploiement et la gestion de leur cycle de vie.

Pour améliorer vos stratégies de déploiement, adoptez systématiquement ces techniques de livraison progressive et moderne. Établissez des processus clairs pour leur mise en œuvre et leur gestion afin de réduire les risques de déploiement, d'accélérer les boucles de rétroaction et de dissocier le déploiement du code de l'activation des fonctionnalités.

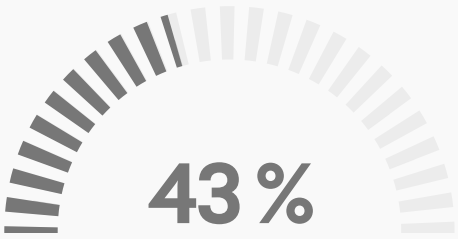
Rollbacks automatisés

De nombreuses organisations gèrent les rollbacks manuellement ou s'appuient sur des données subjectives après l'échec d'un déploiement. Ces méthodes introduisent des retards et de la variabilité dans le processus de récupération, ce qui a un impact direct sur le temps moyen de rétablissement (MTTR) et entrave les efforts de modernisation DevOps. La mise en place de capacités de rollback automatisées offre une approche plus cohérente et efficace pour atténuer l'impact des déploiements échoués, permettant une résolution plus rapide et une stabilité opérationnelle améliorée.

Les défis sont clairs :



des organisations s'appuient sur des rollbacks manuels pour les déploiements échoués.



des équipes d'ingénierie basent leurs décisions de rollback sur des données subjectives plutôt que sur des métriques objectives.

Pour aller plus loin, il est recommandé d’automatiser les rollbacks sur les déploiements critiques. En s’appuyant sur des métriques fiables pour les déclencher et les valider, on réduit les temps de rétablissement et on améliore la stabilité en production.

Sécurité du déploiement

Les leaders de l'ingénierie doivent également s'assurer que leurs processus de déploiement respectent les meilleures pratiques de sécurité et protègent les données sensibles de l'entreprise et des clients. Le DevOps moderne inclut un contrôle rigoureux de l'accès aux secrets et aux informations personnelles identifiables (PII) tout au long du pipeline de livraison logicielle, ce qui permet de prévenir les vulnérabilités pouvant mener à des violations de données ou à des problèmes de conformité.

Cependant, de nombreuses équipes ne parviennent pas à respecter ces exigences critiques :

43 % des équipes d'ingénierie ne stockent pas les secrets dans un gestionnaire de secrets centralisé ou un module de sécurité matériel (HSM), ce qui constitue une bonne pratique.

1 sur 10 1 équipe d'ingénierie sur 10 stocke des secrets et des informations personnelles identifiables en clair dans son système de déploiement.

Pour améliorer la maturité de la sécurité de votre déploiement, priorisez l'adoption d'une solution de gestion centralisée des secrets, comme un Module de Sécurité Matériel (HSM) ou un gestionnaire de secrets dédié, à toutes les étapes du déploiement. Appliquez des politiques strictes pour empêcher le stockage de données sensibles en clair et intégrez des contrôles de sécurité dans les pipelines pour maintenir la conformité et protéger l'intégrité des données.

Ce qu'il faut retenir

Moderniser le DevOps permet de fluidifier les processus de build et de déploiement, en réduisant fortement les tâches manuelles. Aujourd'hui, 23 % du temps des développeurs est encore consacré à ces actions répétitives. Automatiser ces étapes représente un levier d'efficacité considérable.

À l'échelle de 1 000 développeurs, les gains potentiels sont significatifs.

Formula:

2 M d'heures de travail par an

2M d'heures de travail par an x **23 %** consacrées à des activités manuelles de build et de déploiement = **460 k** heures/an

460k heures x 50 $/heure = **23 M$** perdus chaque année à cause de ces tâches

25-40 % Une réduction de 25 à 40 % de ce « toil »

6 M$-9 M$ d'économies annuelles

03: Optimisation

Une organisation tech mature sait tirer le meilleur parti de ses ressources, qu'il s'agisse d'infrastructure, de budget ou de temps humain. L'enjeu : gagner en efficacité sans compromettre la qualité.

En affinant en continu leurs usages, les équipes éliminent les gaspillages, accélèrent les livraisons et posent les bases d'une croissance durable.

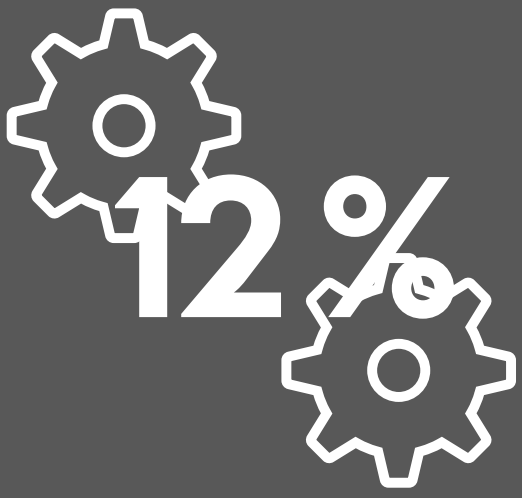
Coûts cloud

À mesure que l'usage du cloud s'intensifie, il devient essentiel de suivre les dépenses de près et d'optimiser leur impact. Une gestion fine des coûts cloud permet d'utiliser les ressources à bon escient et de garder la maîtrise des budgets.

Mais dans la pratique, ce sujet reste souvent mal adressé :

Seuls 2 sur 5

Seuls 2 responsables techniques sur 5 disposent d'un suivi des coûts par projet, équipe ou département.



seulement ont automatisé leurs efforts d'optimisation.

Afin d'améliorer la situation, il est essentiel de disposer d'outils offrant une visibilité fine et en temps réel, tout en automatisant la gestion des ressources afin d'optimiser l'efficacité globale

Indicateurs et analyses

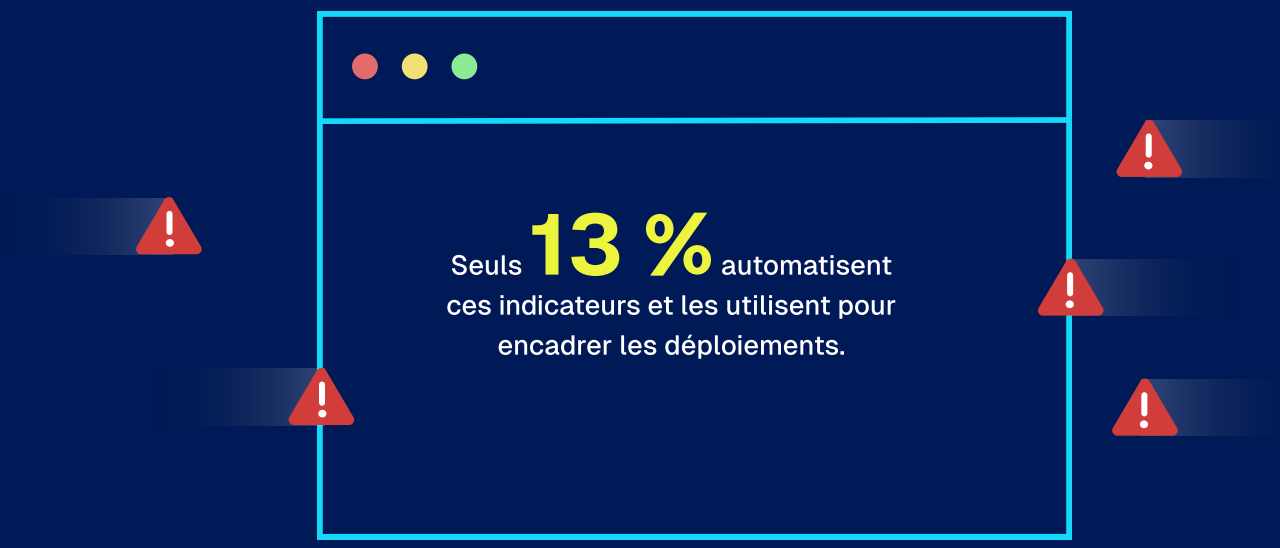
L'optimisation passe par une bonne lecture des données. Sans indicateurs clairs, difficile d'identifier les points de friction, de mesurer les progrès ou d'améliorer l'usage des ressources. Une approche pilotée par les données est essentielle pour faire évoluer les pratiques DevOps.

À ce propos, de nombreuses équipes doivent encore progresser :

17%

des responsables tech indiquent que les métriques ne sont pas analysées chaque semaine.

Près de la moitié ne suivent aucun SLO ni budget d'erreur sur leurs services.



Pour améliorer la performance, il est essentiel de définir des KPI actionnables couvrant tout le cycle de livraison, de sécuriser la collecte de données et de les intégrer dans les décisions quotidiennes.

Ce qu'il faut retenir

Optimiser les dépenses cloud grâce à une meilleure visibilité et à des actions automatisées peut générer un réel impact financier. Les organisations les plus matures enregistrent jusqu'à 10 à 15 % d'économies.

Les entreprises qui exploitent le cloud à grande échelle ont donc un vrai levier d'optimisation à portée de main.

Formula:

35 M$ de dépenses cloud au total - 5 M$ de dépenses via le marketplace cloud = **30 M$** de dépenses cloud optimisables

10-15% d'économies grâce aux meilleures pratiques d'optimisation cloud

3 à 4.5 M$ de réduction des coûts cloud

04: Fiabilité et résilience

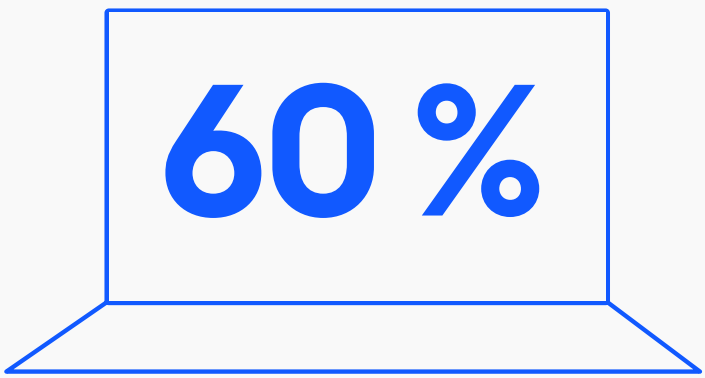
Une pratique engineering mature repose avant tout sur la qualité du logiciel et la résilience opérationnelle. Cela implique de tester rigoureusement les applications, de concevoir des systèmes capables de résister à des conditions extrêmes et de mettre en place des processus solides pour gérer chaque incident et en tirer des enseignements.

En faisant de ces fondamentaux une priorité, les organisations garantissent la fiabilité de leurs systèmes, renforcent la confiance des utilisateurs et assurent la continuité de leur activité.

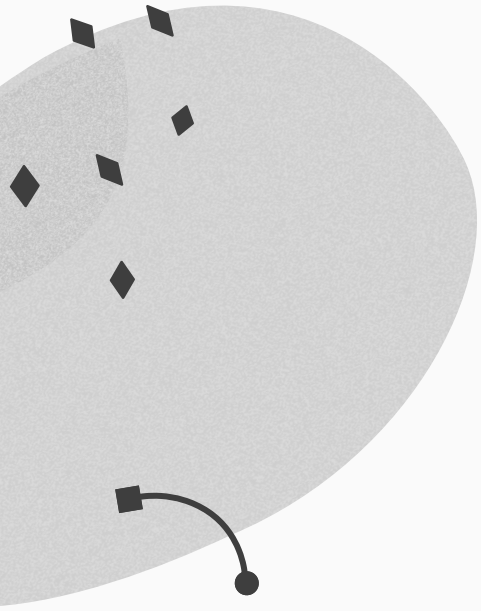
Tests de qualité

La qualité logicielle ne se résume pas à l'absence de bugs. Elle se construit à chaque étape du développement, grâce à des tests bien pensés, automatisés et intégrés aux workflows. Lorsqu'ils sont correctement mis en place, ils permettent non seulement de détecter les erreurs plus tôt, mais aussi de gagner en confiance, de limiter les retours arrière et de livrer plus souvent, avec moins de risques. C'est un socle indispensable pour des systèmes fiables et résilients.

Mais dans la pratique, les fondamentaux sont encore trop souvent négligés :



des environnements de test ou de staging
ne reflètent pas fidèlement la production.



25 %

des fonctionnalités sont livrées sans plan de test clair.

Seule 1 équipe sur 5 dépasse les 90 % de couverture en tests unitaires.

Et seuls 10% ont automatisé l'ensemble de leurs tests fonctionnels.

Pour élever le niveau, il faut aligner les environnements, structurer les tests dès la phase de spec, automatiser au maximum et intégrer des contrôles qualité tout au long du pipeline. C'est ce qui permet d'itérer vite sans sacrifier la fiabilité.

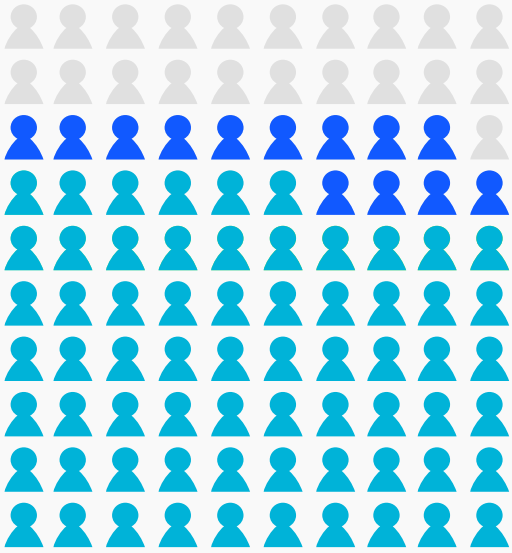
Tests de résilience

Les équipes les plus avancées intègrent désormais des tests de résilience, comme le chaos testing, directement dans leur cycle de développement. L'objectif : simuler des conditions extrêmes (pics de charge, pannes réseau, incidents inattendus) pour tester la robustesse des systèmes avant qu'ils ne soient confrontés à la réalité.

C'est un levier clé pour construire des applications capables de tenir sous pression et de récupérer rapidement en cas d'incident. Pourtant, cette pratique reste encore peu répandue :

66 % des équipes n'utilisent pas de tests de résilience de type chaos testing

13 % l'ont intégré à leur cycle de développement logiciel



Pour renforcer la résilience, il est conseillé d'intégrer des pratiques de chaos testing tout au long du SDLC. En reproduisant régulièrement des pannes dans un environnement contrôlé, les équipes peuvent mettre en évidence les points faibles, vérifier la réaction du système en conditions dégradées et accroître sa capacité à faire face aux imprévus.

Gestion des incidents

Même les équipes d'ingénierie les plus avancées doivent toujours se préparer à toutes les éventualités, car tôt ou tard, un incident finira par survenir. Disposer des bons outils et adopter les bons réflexes pour réagir rapidement et en tirer des enseignements est essentiel pour limiter les impacts, assurer la résilience des systèmes et améliorer continuellement les pratique.

Mais dans les faits, il y a encore une marge de progression :

Plus de la moitié des équipes (52 %) ne disposent pas des outils nécessaires pour gérer efficacement les incidents.

1 équipe sur 10 n’a pas de processus formalisé de gestion d’incident.

Seuls 27 % des responsables techniques déclarent organiser des post-mortems sans blâme après chaque incident.

Pour aller plus loin, il est indispensable de structurer un processus clair, avec des rôles bien définis et des outils adaptés pour détecter, communiquer et résoudre rapidement les incidents. Au-delà des outils, c’est aussi une culture du post-mortem constructif qu’il faut ancrer, pour apprendre de chaque incident et renforcer durablement la résilience des systèmes.

Ce qu'il faut retenir

Investir dans la qualité et la résilience permet de prévenir les interruptions et de renforcer la stabilité des systèmes. Des tests rigoureux, des architectures solides et une gestion d'incident bien rodée réduisent considérablement les impacts en production.

Pour 1 000 développeurs, les économies potentielles sont majeures.

Formula:

Pannes critiques = 3 pannes par an × 6 h par incident x 200 k$ par heure = 3.6 M$ de pertes/an

Pannes mineures = 12 pannes par an × 2 h par incident x 75 k$ par heure = 1.8 M$ de pertes/an

Coût total annuel des interruptions de production = **5.4 M$**

40 to 60 % Réduction grâce aux mesures de qualité et de résilience

2 M$ - 3 M$ d'économies annuelles

05: Sécurité du développement logiciel

Une organisation vraiment mature intègre la sécurité dès les premières étapes du développement, et non en fin de parcours. Cela passe par des politiques de sécurité claires, appliquées dès le départ, une visibilité complète sur les composants logiciels via des outils comme les SBOM (Software Bill of Materials), et des équipes formées aux bonnes pratiques.

Cette approche globale permet de limiter les vulnérabilités, de renforcer la résilience face aux menaces et de mieux protéger les actifs critiques tout au long du cycle de vie logiciel.

Sécurité et gouvernance intégrées

Les organisations les plus matures savent que le “shift-left” ne consiste pas à transférer la responsabilité de la sécurité vers les développeurs, mais à leur donner les moyens de construire des logiciels sécurisés par défaut. Pour y parvenir, il faut intégrer la sécurité et la gouvernance de manière fluide à chaque étape du cycle de développement, sans ajouter de complexité inutile.

Cette approche proactive réduit les vulnérabilités dès la conception, facilite la conformité et fiabilise les livraisons. Mais dans la pratique, les écarts restent importants :

des responsables tech déclarent que plus des deux tiers de leurs pipelines de build ne sont pas protégés par des scans de sécurité.

mettent plus de 7 jours à corriger une faille de sécurité critique.

Près d’1 sur 10 admet publier du code sans corriger les failles critiques

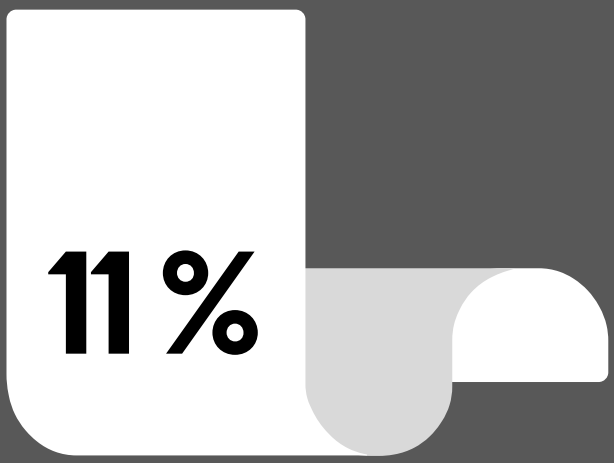
Pour renforcer la sécurité, il est essentiel de définir des standards clairs, d’automatiser les contrôles de sécurité dans les pipelines CI/CD, et de s’assurer que les failles critiques soient traitées rapidement, bien avant la mise en production.

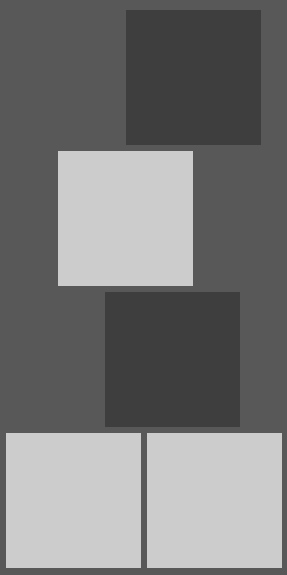
Software Bill of Materials (SBOMs)

Avoir une vision claire de ce qui compose un logiciel n'est plus une option. Le SBOM (Software Bill of Materials) apporte une visibilité totale sur les composants, dépendances et éventuelles vulnérabilités d'une application. C'est un outil clé pour mieux gérer les risques et renforcer la sécurité des systèmes.

Pourtant, cette pratique reste encore peu répandue

Seuls 11 % des responsables techniques déclarent générer un SBOM pour l'ensemble de leurs artefacts





Et plus de la moitié des équipes (57 %) en génèrent pour

moins de 30 %

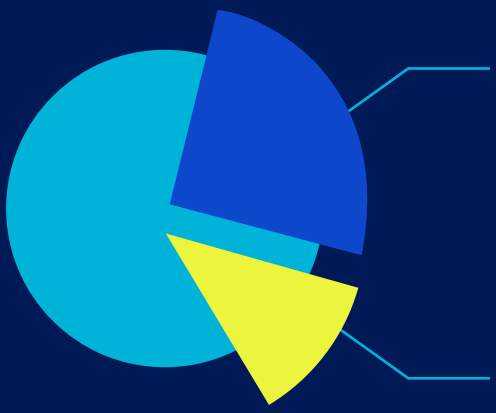
de leurs livrables, voire pas du tout.

Pour renforcer la sécurité, il est essentiel d'automatiser la génération de SBOMs à chaque étape du cycle de développement. Ces inventaires doivent être reliés à des outils de gestion des vulnérabilités pour identifier les risques en amont et répondre aux exigences croissantes du secteur en matière de conformité.

Formation à la sécurité

Même les meilleurs outils ne remplacent pas la vigilance des développeurs. Une pratique engineering véritablement mature repose sur des équipes formées aux bons réflexes de sécurité. Maîtriser les vulnérabilités courantes, les principes du code sécurisé et les différents types de menaces permet de limiter les risques dès la phase de développement.

Mais dans les faits, la niveau de formation reste très variable :



56% des développeurs suivent une formation annuelle ou semestrielle

23% ne sont jamais formés à la sécurité

Pour contrer ces tendances, il faut instaurer un programme clair, continu et accessible à tous. Formations régulières, ateliers pratiques et ressources en libre accès contribuent à renforcer la posture de sécurité des équipes sur le long terme.

Ce qu'il faut retenir

Intégrer la sécurité dès le début du cycle et automatiser les contrôles permet de limiter les tâches chronophages pour les développeurs. Ce "toil" réduit n'est qu'une partie du sujet, mais son impact sur la productivité est réel. Cette estimation se concentre uniquement sur ces gains de temps (sans même compter les économies liées à la prévention des failles ou à la protection de la marque).

À l'échelle de 1 000 développeurs, le potentiel de gain est significatif.

Formula:

1 k Devs x 2000 heures = **2 M** d'heures de travail annuelles

5 % du temps consacré à des tâches de sécurité manuelles = 100 k par heures x 50 $ par heures = **5 M$** par an

20-25% Réduction grâce à l'automatisation

1 M$ à 1.25 M$ d'économies annuelles de productivité

Pour aller plus loin

Entre rapidité d'exécution, sécurité et résilience, les équipes engineering doivent répondre à des exigences toujours plus élevées. Pour y parvenir, les pratiques les plus matures s'appuient sur une automatisation complète du cycle de développement, de la compilation au déploiement. C'est cette base qui permet de progresser rapidement sans compromettre la qualité.

Ce rapport révèle que la plupart des organisations se situent encore à un stade intermédiaire : les bonnes pratiques existent, mais elles sont appliquées de manière isolée ou partielle. Ce manque de cohérence freine l'efficacité globale, que ce soit pour l'expérience développeur, le DevOps, l'optimisation des ressources, la qualité logicielle ou la sécurité.

Pour passer à l'échelle, il est nécessaire de dépasser les approches par silos et d'unifier outils, workflows et standards au sein d'une plateforme centralisée. Cette vision intégrée est la clé pour atteindre une véritable excellence opérationnelle.

Évaluez votre maturité en répondant à l'Engineering Excellence Assessment.