Bien utiliser DevOps :

Les meilleures pratiques pour surmonter les obstacles

Table des matières

Introduction

Votre équipe a adopté DevOps. Vous avez mis en place de nouveaux processus, adopté de nouveaux outils et façonné une culture qui met l'accent sur la collaboration interfonctionnelle. Mais vous n'avez pas encore atteint votre vitesse maximale. Il vous manque quelque chose, quelque chose qui empêche votre organisation de devenir une véritable machine DevOps performante.

Cette pièce manquante, c'est souvent la mesure des données. Bien que la mesure soit l'un des cinq piliers du concept CALMS (culture, automatisation, lean, mesure, partage) créé par l'expert de DevOps Jez Humble, elle est souvent négligée par les équipes DevOps dans leur quête d'une plus grande rapidité et de plus d'autonomie. Cela peut toutefois créer des problèmes importants, car la précision des données est essentielle au bon fonctionnement d'une équipe DevOps, qu'il s'agisse de la réponse sur incident ou de la bonne compréhension de la complexité des microservices, entre autres.

Cet ebook est destiné à toutes les équipes et organisations qui ont goûté à DevOps et qui sont maintenant prêtes à s'y immerger totalement. Il est également conçu pour ceux qui font du sur-place sans profiter de DevOps pour réussir pleinement leur transformation numérique.

En partageant des expériences réelles, en particulier les leçons apprises par New Relic, nous voulons vous aider à surmonter les derniers obstacles en travers de votre réussite DevOps. Pour mieux comprendre comment définir des objectifs de fiabilité, démêler les fils de communications uniques et les exigences du développement de votre approche des microservices, nous avons réuni les meilleures pratiques éprouvées qui vous montrent comment aller plus vite, plus efficacement.

CHAPITRE 1

Équilibrer les SLO avec une livraison rapide des applications

Équilibrer les SLO avec une livraison rapide des applications

Vos cycles de développement sont plus rapides et vous déployez plus souvent du code, mais avec quel niveau de fiabilité ? La qualité et la fiabilité sont deux résultats également importants d'une approche réussie de DevOps.

C'est là qu'intervient le SRE. Le SRE (Site reliability engineering/Ingénierie de fiabilité de site) est un rôle interfonctionnel, prenant en charge des responsabilités traditionnellement spécialisées et indépendantes des équipes développement, opérations et autres groupes informatiques. Comme le SRE repose sur la collaboration entre le développement et les opérations, il va de pair avec la culture DevOps. DevOps et le SRE ont beaucoup en commun, mais le SRE est particulièrement dédié à l'amélioration continue et à la gestion de résultats mesurables, en particulier par le recours aux SLO (Service Level Objectives/objectifs de niveau de service).

> *« À la base, [SRE] correspond à ce qu'il se passe lorsque vous demandez à un ingénieur logiciel de concevoir une fonction opérationnelle. »*
>
> ***Ben Treynor Sloss, Vice-président Engineering, Google***

Commençons par quelques définitions importantes :

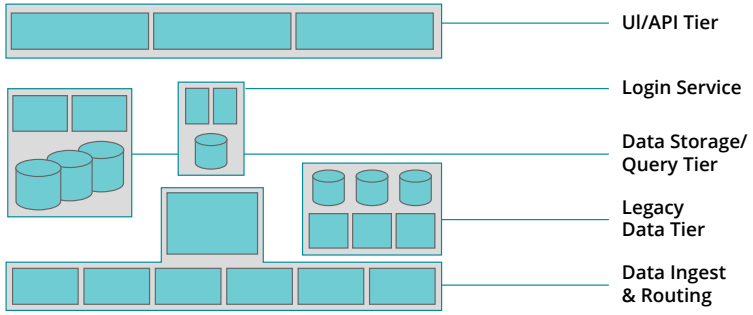
TERME	DÉFINITION	EXEMPLE
SLI (Indicateur de niveau de service)	Le SLI est la mesure clé de vos performances.	« Les clients peuvent se connecter et voir leurs données… »
SLO (Objectif de niveau de service)	Les SLO sont les valeurs cibles ou les objectifs de performances de votre système. Ils représentent un engagement permanent.	« 99,9 % du temps… »
SLA (Accord de niveau de service)	Le SLA définit les conséquences du non-respect de vos engagements SLI/SLO.	«… ils peuvent demander un remboursement pour les pertes induites par la non disponibilité du service. »

En savoir plus sur le SRE dans notre ebook, Site Reliability Engineering : Philosophies, Habits, and Tools for SRE Success (philosophies, pratiques et outils pour la réussite SRE)

Définir des SLI et SLO appropriés

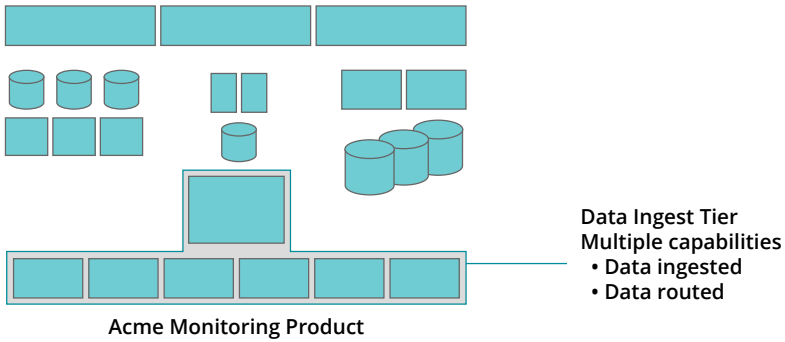
Bien que les meilleures pratiques du SRE dans l'industrie imposent des SLI et des SLO pour chaque service que vous fournissez, il peut s'avérer compliqué de les définir et les déployer si vous ne l'avez jamais fait. Voici sept étapes que nous utilisons chez New Relic pour définir les SLO et les SLI :

1. **Identifiez les limites du système :** la limite d'un système est définie par un ou plusieurs composants présentant une ou plusieurs fonctionnalités à des clients externes. En interne, votre plateforme peut avoir plusieurs éléments mobiles (nœuds de service, base de données, équilibreur de charge, etc.) mais ces éléments individuels ne sont pas considérés comme des limites du système car ils ne présentent pas directement de fonctionnalités aux clients. Pensez plutôt à des éléments multiples fonctionnant ensemble pour présenter des fonctionnalités. Par exemple, un service de connexion qui présente une API capable d'authentifier les informations de connexion des utilisateurs constitue un groupe logique de composants fonctionnant collectivement en tant que système. Avant de définir vos SLI, commencez par regrouper les éléments de votre plateforme en systèmes et définissez leurs limites. Concentrez-vous sur cet aspect lors des étapes suivantes car les SLI et les SLO de limites sont très utiles.



Systèmes et limites au sein d'une plateforme

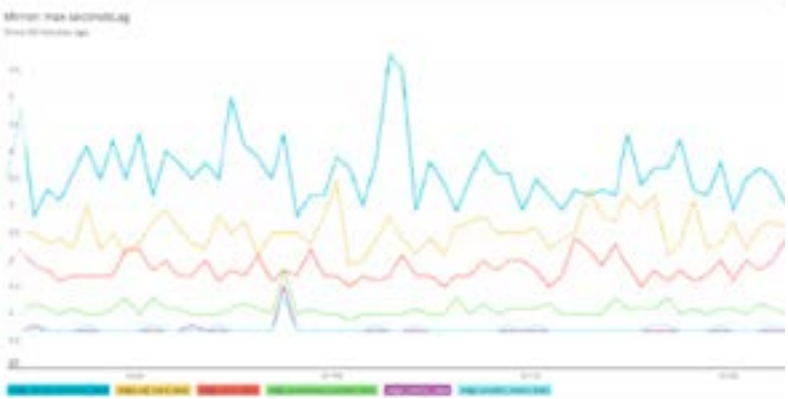
2. **Définissez les fonctionnalités présentées par chaque système :** regroupez à présent les composants de la plateforme en unités logiques (niveau interface utilisateur/API, service de connexion, niveau stockage de données/requêtes, niveau données héritées, ingestion et routage de données). Chez New Relic, les limites de notre système sont alignées sur les limites de notre équipe d'ingénierie. À l'aide de ces regroupements, articulez l'ensemble des fonctionnalités exposées par chaque limite du système.



Fonctionnalités définies sur une limite du système

3. **Créez une définition claire de ce qui est « disponible » pour chaque fonctionnalité :** par exemple, « envoyer des messages à la bonne destination » est une façon de décrire les attentes de disponibilité d'une fonctionnalité de routage de données. Privilégiez un langage clair pour expliquer la disponibilité plutôt que des termes techniques que tout le monde ne connaît pas forcément.
4. **Définissez les SLI techniques correspondants :** il est maintenant temps de définir un ou plusieurs SLI par fonctionnalité en utilisant votre définition de la disponibilité de chaque fonctionnalité. En partant de notre exemple ci-dessus, un SLI de fonctionnalité de routage de données pourrait être « temps nécessaire pour transmettre un message à la bonne destination ».

5. **Mesurez pour obtenir une référence :** il est évident que la surveillance vous permettra de savoir si vous atteignez ou pas vos objectifs de disponibilité. À l'aide de votre outil de surveillance, recueillez des données de référence pour chaque SLI avant de définir vos SLO.
6. **Appliquez des cibles SLO (par SLI/fonctionnalité) :** une fois que vous avez recueilli les données, mais avant de définir vos SLO, posez à vos clients des questions qui vous aideront à identifier leurs attentes et à adapter vos SLO pour les satisfaire. Choisissez ensuite des cibles SLO sur la base de vos références, de l'opinion du client, de ce que votre équipe peut s'engager à prendre en charge et de ce qui est réalisable en fonction de vos moyens techniques actuels. En suivant notre exemple de SLI pour le routage des données, le SLO pourrait être « 99,5 % 9des messages transmis en moins de 5 secondes ». N'oubliez pas de configurer un déclencheur d'alerte dans votre application de surveillance, avec un seuil d'avertissement pour les SLO que vous définissez.



Exemple de SLI pour la fonctionnalité de routage des données

7. **Procédez par itérations et ajustez :** n'utilisez pas d'approche de définition définitive pour vos SLO et SLI. Vous devez considérer qu'ils vont (et qu'ils devraient) évoluer dans le temps à mesure qu'évoluent vos services et les besoins des clients.

Conseils supplémentaires concernant les SLO et les SLI

- **Veillez à ce que chaque instance logique d'un système ait son propre SLO :** par exemple, pour les systèmes fragmentés (contrairement aux systèmes horizontaux), mesurez les SLI et les SLO séparément pour chaque fragment.
- **Faites la différence entre les SLI et les alertes :** le processus SRE ne remplace pas un véritable système d'alertes.
- **Utilisez des SLO composés le cas échéant :** vous pouvez créer un SLI composé pour capturer plusieurs conditions SLI et faciliter sa compréhension par les clients.
- **Créez des SLO spécifiques à vos clients le cas échéant :** il n'est pas rare que les clients importants reçoivent des SLA qui leur procurent une meilleure disponibilité de services que ceux proposés à d'autres clients.

> *« Pour atteindre l'excellence opérationnelle, nous mesurons tout. C'est la seule façon de gérer et améliorer l'ensemble. »*
>
> *Craig Vandeputte, directeur de DevOps, CarRentals.com*

CHAPITRE 2

Créer une politique d’astreinte juste et efficace

Créer une politique d'astreinte juste et efficace

La prochaine étape pour améliorer la fiabilité tout en accélérant les déploiements consiste à s'assurer que votre organisation est capable de traiter tout problème logiciel (de jour comme de nuit), rapidement et efficacement. Pour ce faire, vous avez besoin d'une politique d'astreinte.

Attendez… ne passez pas directement au prochain chapitre. Nous sommes conscients que le terme « astreinte » peut provoquer des réactions désagréables chez de nombreuses personnes. Mais c'est parce que la plupart des organisations ne comprennent pas bien le concept de rotation. Cette erreur entraîne non seulement stress et regrets dus à l'échec de vos SLA avec les clients, mais installe également une ambiance non productive et désagréable pour l'équipe d'ingénieurs épuisés et frustrés.

Commencez par les fondamentaux

Une politique d'astreinte efficace et juste nécessite deux conditions préalables importantes :

1. **Organisation et système structurés :** résoudre les problèmes efficacement est beaucoup plus facile lorsque vos systèmes (services ou applications) et vos équipes produits sont tous bien organisés et structurés en unités logiques. Par exemple, chez New Relic, nos 57 équipes d'ingénieurs prennent en charge 200 services individuels, chaque équipe agissant de manière autonome pour gérer entièrement au moins trois services pendant le cycle de vie, de la conception au déploiement en passant par la maintenance.
2. **Une culture de responsabilité :** avec DevOps, chaque équipe est responsable du code qu'elle déploie en production. Les équipes prennent tout naturellement des décisions différentes sur les modifications et les déploiements lorsqu'elles sont responsables et astreintes au service, par comparaison avec les environnements traditionnels où quelqu'un d'autre est responsable du code une fois celui-ci utilisé en production.

Appliquez ces meilleures pratiques pour améliorer vos pratiques d'astreinte

Structurez de manière juste votre équipe et votre organisation

Chez New Relic, chaque ingénieur et responsable d'équipe de l'organisation produit prend la responsabilité des services de l'équipe avec une astreinte par rotation. Les équipes sont responsables d'au moins trois services, mais le nombre de services pris en charge dépend de leur complexité et de la taille de l'équipe. En ce qui concerne votre organisation, regardez la taille de l'ensemble de l'entité d'ingénierie et des équipes individuelles avant de choisir une approche de rotation d'astreinte. Par exemple, si l'équipe comprend six ingénieurs, chaque ingénieur pourrait être le contact principal d'astreinte toutes les six semaines.

Soyez souple et créatif lorsque vous mettez en place les rotations

Vous pourriez laisser à chaque équipe le soin de concevoir et mettre en places a propre politique de rotation d'astreinte. Donnez de la liberté et de l'autonomie aux équipes pour qu'elles réfléchissent à la meilleure façon d'organiser des rotations adaptées à leurs besoins spécifiques.

Chez New Relic, chaque équipe dispose de l'autonomie nécessaire pour créer et mettre en place son propre système d'astreinte. Par exemple, une équipe se sert d'un script qui effectue une rotation aléatoire de l'ordre d'astreinte de la personne secondaire.

Suivez les mesures et surveillez les incidents

Pour veiller à ce que la rotation d'astreinte soit juste et efficace, il est très important de la surveiller et de suivre les mesures d'incidents. Chez New Relic, nous suivons le nombre d'appels, le nombre d'heures d'appels et le nombre d'appels hors horaire. Nous regardons ces mesures aux niveaux ingénieur, équipe et groupe. Le suivi des mesures nous aide à attirer l'attention sur les équipes ayant à faire face à des charges d'appels ingérables (si une équipe a en moyenne plus d'un appel hors horaire par semaine, cette équipe est considérée comme ayant une charge d'astreinte élevée). En maîtrisant ces mesures, nous pouvons déplacer les priorités en réduisant la charge technique d'une équipe ou en fournissant plus d'assistance pour améliorer les services.

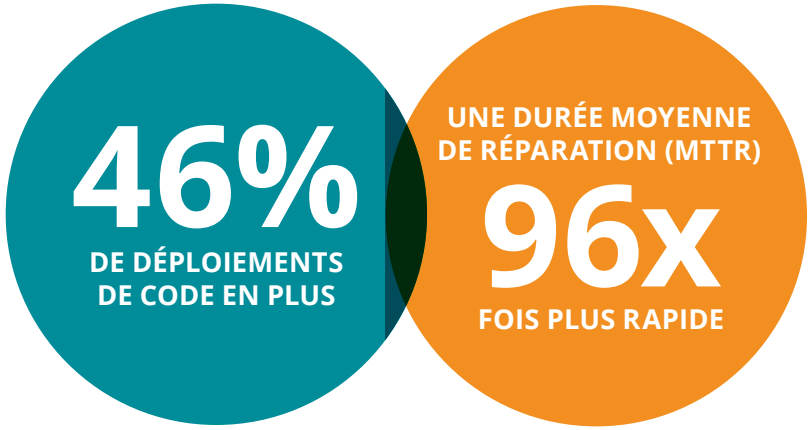
Adaptez votre politique pour l'aligner sur la situation de votre entreprise

Une politique d'astreinte adaptée à une équipe chez New Relic peut s'avérer inapplicable dans votre entreprise. Pour créer une rotation d'astreinte à la fois juste et efficace, prenez les éléments ci-dessous en considération :

- **Croissance :** quel est le rythme de croissance de votre entreprise et de votre groupe d'ingénierie ? Avez-vous un roulement important du personnel ?
- **Localisation :** votre organisation d'ingénierie est-elle centralisée ou répartie sur plusieurs sites ? Disposez-vous des ressources adéquates pour assurer des rotations continues ?
- **Complexité :** quel est le niveau de complexité de vos applications et comment sont-elles structurées ? Quel est le niveau de complexité des dépendances entre les services ?
- **Outils :** disposez-vous d'outils de résolution d'incidents qui envoient aux ingénieurs des notifications de problèmes automatisées et actionnables ?
- **Culture :** l'astreinte fait-elle partie intégrante de votre culture d'ingénierie ? Avez-vous une culture non culpabilisante, centrée sur la recherche des causes et leur résolution plutôt que sur la recherche des coupables ?

> *« Les meilleurs performeurs [DevOps] sont deux fois plus susceptibles de dépasser leurs propres objectifs dans les domaines de la rentabilité, des parts de marché et la productivité. »*

PAR RAPPORT AUX MAUVAIS ÉLÈVES, LES MEILLEURS PERFORMEURS DE DEVOPS ONT[1]



1 Source : « 2017 State of DevOps Report », Puppet and DORA

CHAPITRE 3

Répondre efficacement aux incidents

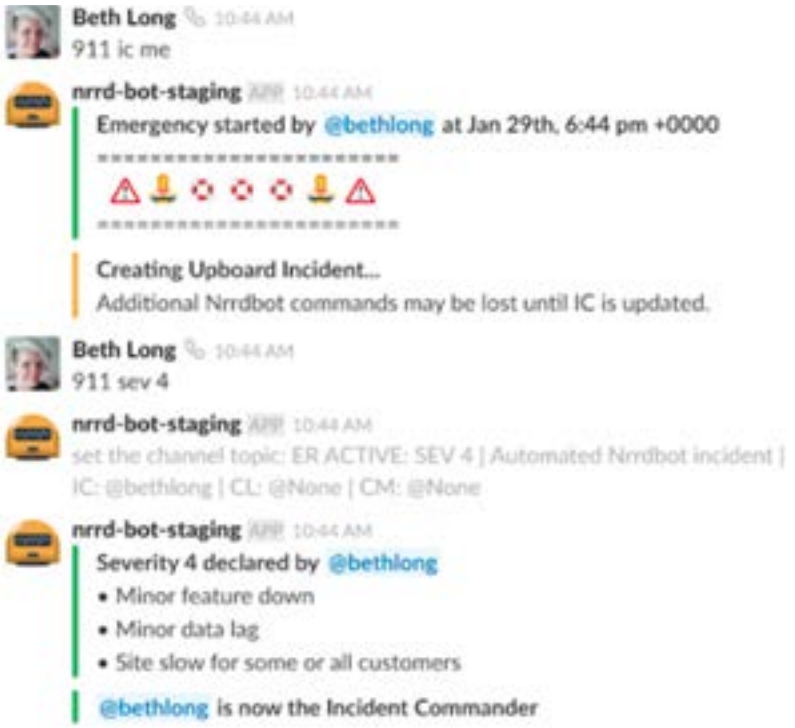
Répondre efficacement aux incidents

Les rotations d'astreinte vont de pair avec la gestion des incidents. Qu'est-ce qu'un incident ? Un incident survient lorsqu'un système se comporte de manière inattendue, avec un impact potentiel sur les clients (ou les partenaires et employés). Compétence au cœur de l'approche DevOps « vous le développez, vous en avez la propriété », la gestion des incidents est souvent dévalorisée, les équipes s'en détournant une fois le problème résolu. Souvent, les organisations dénuées de gestion des incidents efficaces improvisent, parent au plus pressé, en utilisant l'organisation, les méthodes et les communications disponibles. En cas de gros problème, tout le monde se bouscule pour trouver un plan pour le résoudre. Il existe une bien meilleure approche, qui non seulement réduit la durée et la fréquence des pannes, mais apporte aux ingénieurs responsables le soutien nécessaire pour répondre efficacement.

Création d'un processus de gestion des incidents efficace

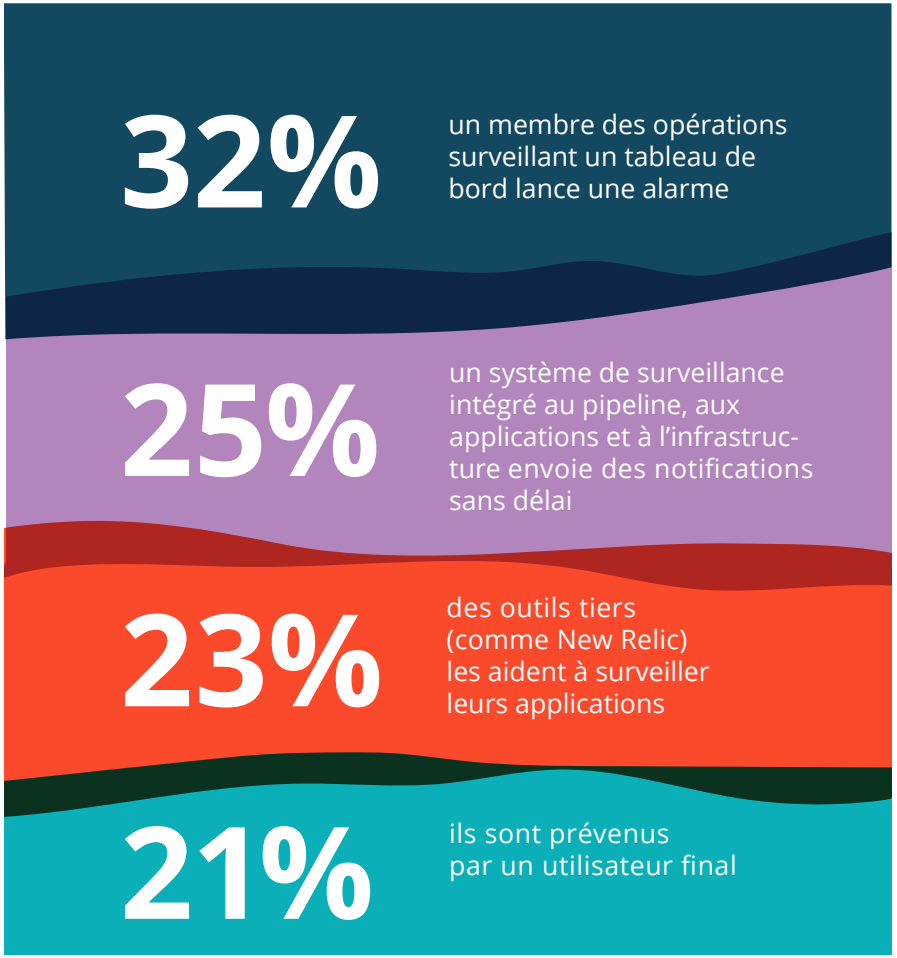
1. **Définissez les niveaux de gravité :** les niveaux de gravité déterminent le niveau de support nécessaire et l'impact potentiel sur les clients. Par exemple, New Relic utilise une échelle de gravité de 1 à 5.
    - Niveau 5 : pas d'impact sur les clients, peut servir à prendre conscience d'un problème.
    - Niveau 4 : bogues mineurs ou faible latence des données qui impactent, mais ne gênent pas les clients.
    - Niveau 3 : latence élevée des données ou fonctionnalités indisponibles.
    - Niveaux 2 et 1 : incidents graves entraînant des pannes.
2. **Renforcez vos services :** chaque service doit être doté d'un système de surveillance et d'alerte pour un signalement proactif des incidents. Le but est de détecter les incidents avant les clients pour éviter les pires scénarios où des clients en colère appellent l'assistance technique ou publient des commentaires sur les réseaux sociaux. Avec le signalement proactif des incidents, vous pouvez y réagir et les résoudre aussi vite que possible.
3. **Définissez les rôles de réponse :** chez New Relic, les membres des équipes d'ingénierie et d'assistance jouent les rôles suivants pendant un incident : responsable d'incident (dirige les résolutions), chef technique (diagnostique et corrige), chef des communications (informe tout le monde), responsable de la communication (coordonne la stratégie de communication des urgences), interface d'incident (collabore avec l'assistance technique et les dirigeants pour le niveau de gravité 1), responsable des urgences (facultatif pour le niveau 1) et responsable d'ingénierie (gère le processus post-incident).
4. **Créez un plan d'action :** il s'agit de la série de tâches par rôle, qui couvre tout ce qu'il se passe lors du cycle de vie d'un incident, notamment son signalement, la définition de sa gravité, le choix des chefs techniques appropriés à contacter, le débogage et la résolution du problème, la gestion du flux de communications, le transfert des responsabilités, la clôture de l'incident et la conduite d'une analyse rétrospective.
5. **Mettez en place des outils et automatismes appropriés en soutien de l'intégralité du processus :** de la surveillance aux alertes, en passant par les tableaux de bord et le suivi des incidents, l'automatisation du processus est essentielle pour maintenir les collaborateurs concernés informés et garantir l'exécution efficace du plan.

6. **Menez des analyses rétrospectives :** après l'incident, demandez à vos équipes de mener une analyse rétrospective dans les deux jours suivant l'incident. Mettez l'accent sur le fait que le but de l'analyse n'est pas de rechercher des coupables, mais de comprendre les véritables causes du problème.
7. **Mettez en place une politique de non-répétition d'incidents :** si un problème de service a un impact sur vos clients, il est alors temps d'identifier et de payer la dette. Une politique de non-répétition d'incidents signifie que votre équipe interrompt toute activité sur ce service tant que la cause première du problème n'a pas été corrigée ou contenue.



Exemple d'incident déclaré dans Slack

COMMENT LES ÉQUIPES DEVOPS DÉCOUVRENT LES PROBLÈMES[2]



2: Quelle: « DevOps Survey Results », 2nd Watch, 2018.

CHAPITRE 4

Surmonter la complexité des microservices

Surmonter la complexité des microservices

Les microservices et DevOps fonctionnent mieux ensemble. Les entreprises ont compris que décomposer des applications monolithiques en services distincts peut leur apporter des gains spectaculaires de productivité, rapidité, agilité, extensibilité et fiabilité.

Mais bien que les équipes reconnaissent les changements à apporter pour le développement, le test et le déploiement des microservices, elles surestiment souvent les changements substantiels nécessaires dans les domaines de la collaboration et des communications. Les ingénieurs de New Relic ont développé les meilleures pratiques suivantes pour favoriser un environnement collaboratif qui simplifie la complexité et les défis de communication inhérents au monde des microservices.

Bonnes pratiques de communication pour un environnement de microservices

- **Faites en sorte que les dépendances en amont et en aval soient au courant des changements majeurs :** avant de déployer des changements importants au niveau de votre microservice, informez les équipes qui en dépendent en amont et en aval afin que, en cas de problème, elles ne perdent pas de temps à en rechercher la cause profonde.
- **Communiquez vite et souvent :** cela est particulièrement important pour la gestion des versions, les mises en obsolescence et les situations où vous avez besoin de fournir temporairement une compatibilité ascendante.
- **Traitez les API internes comme des API externes :** rendez votre API plus facile à appréhender par les développeurs avec de la documentation, des messages d'erreur détaillés et un processus pour envoyer les données de test.
- **Traitez les équipes en aval comme des clients :** créez un fichier LISEZ-MOI avec un schéma de l'infrastructure, une description et des instructions pour une exécution locale, ainsi que des informations sur la manière de participer.
- **Créez un canal réservé aux annonces :** il est essentiel d'avoir une seule source crédible pour les annonces importantes plutôt que de se reposer sur les équipes pour glaner des informations importantes auprès de plusieurs panneaux, discussions et e-mails.
- **Centrez-vous sur votre « voisinage» de service :** vos équipes en amont, en aval, d'infrastructure et de sécurité sont toutes « voisines » de votre service. En tant que bon voisin, vous devez participer à leurs démos et annonces, donner à vos voisins l'accès à la carte de votre service et maintenir une liste de contacts.

Utilisation des données pour mieux comprendre comment fonctionnent les microservices

Décomposer vos applications monolithiques en microservices est très ardu. Vous devez d'abord comprendre l'intégralité d'un système avant de le partitionner en limites de services. Ensuite, le partitionner avec précision pour créer de véritables microservices (chacun avec une seule

fonction, détaillé et peu associé avec des applications et d'autres services) peut s'avérer compliqué. Voici quelques conseils de mesure et de surveillance que vous pouvez appliquer pour repérer les microservices de votre environnement qui ont encore trop d'interdépendances avec d'autres services et applications :

1. **Déploiements :** votre équipe synchronise-t-elle des déploiements avec des équipes en amont et en aval ? Si vous voyez des marqueurs de déploiement dans votre solution de surveillance qui sont synchronisés sur plusieurs services, cela signifie que vous n'avez pas vraiment découplé vos services.
2. **Communications :** un microservice ne devrait nécessiter que peu de communication avec d'autres services pour exécuter sa fonction. Si vous voyez un service avec de nombreux aller-retour de demandes vers les mêmes services en aval, c'est un signe clair qu'il n'est pas découplé. Le débit est un autre marqueur à surveiller. Si le nombre d'appels par minute pour un microservice donné est beaucoup plus élevé que celui de l'application en général, cela signifie clairement que le service n'est pas découplé.
3. **Magasins de données :** chaque microservice doit avoir son propre magasin de données pour éviter les problèmes de déploiement, les problèmes d'immobilisation de la base de données et des changements de schéma qui créent des problèmes pour les autres services partageant le magasin de données. Une bonne surveillance peut vous montrer si chaque microservice utilise son propre magasin de données.
4. **Extensibilité :** dans un véritable environnement de microservices, les pics des services placés sur des hôtes doivent correspondre aux pics de débit des services individuels. Cela indique une extension dynamique, un des avantages les plus importants des microservices. D'autre part, si vous voyez des pics correspondants sur tous les services et hôtes, il y a de bonnes raisons de croire que vos services ne sont pas découplés.
5. **Developers per application:** si vous avez une communication efficace entre vos équipes de microservices, vous n'aurez pas besoin de « gurus d'architecture » qui s'occupent de tous les microservices pour garantir leur bon fonctionnement. Si vous avez 100 ingénieurs pour 10 services, et que deux de vos ingénieurs font du développement sur les 10 services, c'est probablement le signe qu'ils ne sont pas réellement découplés et que le développement de tous vos services repose sur une connaissance trop exclusive et les compétences en communication de ces deux développeurs.

IL EST COMMUNÉMENT ACQUIS QUE LES MICROSERVICES ET LES CONTENEURS SONT ESSENTIELS[3]

3: Source: « Enterprise Priorities for Hybrid Cloud Management», Ponemon Institute, June 2018.

CHAPITRE 5

Utiliser les données pour atteindre la vélocité numérique

Utiliser les données pour atteindre la vélocité numérique

L'élément fondamental de DevOps est la vitesse : livraison plus rapide des logiciels, résolution plus rapide des problèmes, innovation plus rapide. Lorsque vous avez atteint la vitesse dont a besoin votre activité pour saisir les nouvelles opportunités du marché, battre la concurrence sur le terrain de l'innovation et satisfaire vos clients, vous avez atteint la vélocité numérique.

Chez New Relic, nous savons que les données sont le carburant de la réussite de DevOps car elles vous aident à réaliser ce qui suit :

- Mesurer et suivre les performances de DevOps
- Fournir des commentaires instantanés qui permettent à chacun de se consacrer à ses tâches
- Optimiser la livraison des logiciels, les performances et les résultats commerciaux

L'une des questions les plus fréquentes que nous posent les clients concernant DevOps est « Par où commencer ? » Voici quelques principes fondamentaux qui devraient guider votre expérience globale de DevOps :

Éliminez les silos de données

Même si les mesures sont au cœur de la doctrine de DevOps, beaucoup d'équipes ne comprennent pas que les silos de données signifient que chacun voit midi à sa porte dans le domaine des performances. Avec les silos de données de performances, il n'y a pas de langage commun et cohérent entre les applications, l'infrastructure et ce que vivent les utilisateurs. Déployer des instruments sur tous vos systèmes pour tout présenter sur une seule plateforme met tous les intervenants sur la même longueur d'onde, avec des données et des mesures communes pour regrouper les différentes fonctions en une seule équipe. Chez New Relic, nos équipes d'ingénierie se battaient contre une prolifération d'outils et de priorités conflictuelles. En faisant en sorte que tous les intervenants soient d'accord sur les SLO les plus importants, puis en plaçant ces SLO dans des tableaux de bord partagés, les équipes ont commencé à agir dans le même sens.

Éliminez la complexité

Bien que les architectures applicatives modernes permettent de simplifier et d'accélérer le développement de différentes façons, la qualité dynamique de l'architecture modulaire d'aujourd'hui crée une complexité d'un nouveau type, avec de nombreux composants individuels formant un tout cohérent. Bénéficier d'une vision complète de votre architecture, même si elle est éphémère, vous permet de vous accommoder de cette complexité. Posséder les bonnes données vous permet de comprendre ce qui fonctionne et de mettre votre équipe dans la bonne direction.

Comprenez l'impact des changements

Avec DevOps, les déploiements et les modifications de code sont plus fréquents ; votre équipe a besoin de les maîtriser pour éviter les problèmes potentiels. Assurez-vous de disposer d'une fonctionnalité de reporting montrant vos déploiements récents et leur impact avant/après sur les performances des applications et l'expérience des clients, notamment les erreurs éventuelles qui se sont produites. Vous pouvez ainsi rapidement mettre en corrélation les modifications et leurs conséquences potentielles, et permettre à votre équipe de répondre rapidement, de rappeler une version, ou de fournir une résolution rapide des incidents éventuellement apparus.

Conclusion

La popularité de DevOps s'est largement répandue, profitant à des entreprises de nombreux secteurs différents engagés dans leur quête de rapidité, productivité, qualité et innovation grâce à l'adoption des principes de DevOps. En atteignant une vélocité numérique maximale, les organisations DevOps performantes réussissent plus de déploiements, avec un temps de récupération après panne bien plus court que leurs concurrents moins performants.

Les données sont le carburant d'un moteur DevOps performant. Les meilleures pratiques, comme celles décrites dans cet ebook, portent essentiellement sur la bonne utilisation des données pour alimenter la réussite. New Relic vous procure la visibilité complète dont vous avez besoin pour surveiller vos efforts DevOps et améliorer en permanence vos résultats à chaque étape.

C'est le point du départ de la réussite DevOps.
Visitez newrelic.com/devops