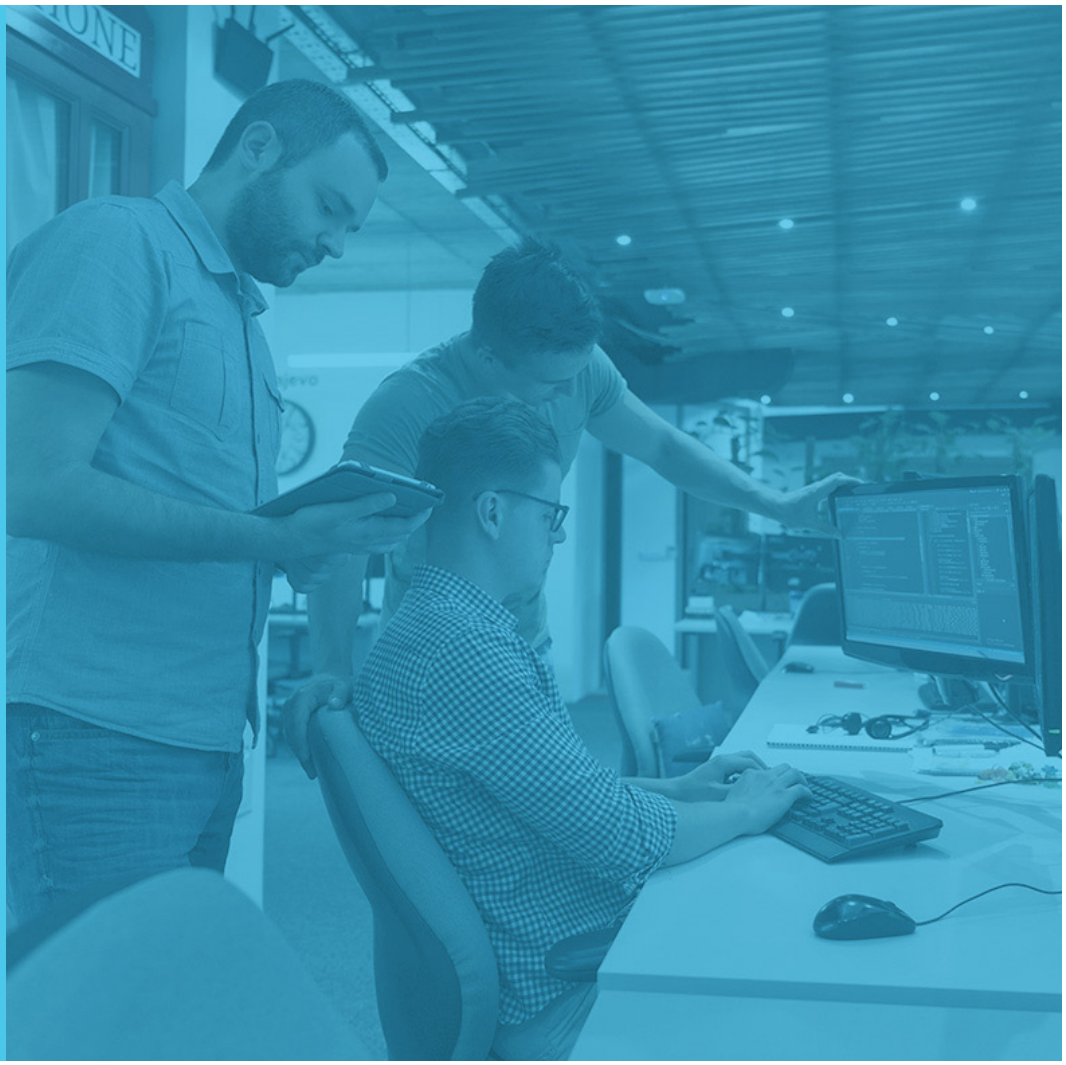
EMYODE

SecDevOps :
Personnes, Processus et Produits

Une partie intégrante de votre transformation numérique



Livre blanc

Remerciements

Créé et publié par l'équipe DevOps Emyode, ce document a été développé avec le soutien de l'ensemble de l'organisation et en collaboration directe avec :

Contributeurs clés

Jean-Paul Lizotte (Emyode)
Eduardo Mazza, PhD (Emyode)

Réviseur

Maxime Plante (Emyode)

Remarques

Veuillez envoyer vos commentaires ou suggestions concernant ce document à l'alias de retour Emyode (info@emyode.com).

Table des matières

Préface

Le Centre d'excellence Emyode : la courbe SecDevOps au DevOps

Pour effectuer ses recherches sur le sujet, le Centre d'excellence Emyode a travaillé intensivement en étroite collaboration avec de nombreux techniciens et représentants agissant sur le terrain ou occupant un rôle de soutien dans le domaine.

Ce document est basé sur les résultats d'une analyse détaillée des différentes façons d'améliorer la livraison de logiciels grâce aux compétences humaines, à l'amélioration des processus et à l'utilisation des outils appropriés.

— Jean-Paul Lizotte

1. Résumé

Le DevOps est un domaine d'expertise récent qui concilie l'art des valeurs transmises à la précision requise pour la livraison efficace de logiciels. À la base, il s'agit d'un mouvement de culture d'entreprise destiné à révolutionner les méthodes de travail en adoptant la transparence, des mesures objectives et une approche de l'automatisation à effet rapide et « à faible risque ».

L'utilisation du terme SecDevOps (aka DevSecOps) est selon nous inadéquate. En effet, la sécurité fait partie intégrante de la qualité et pour être considéré comme valeur ajoutée, le DevOps — qui contrairement à SecDevOps exclurait les processus de sécurité dans sa structure — se doit d'offrir une assurance qualité lors de la livraison de logiciels.

En d'autres termes, pour nous, et, espérons-le, pour la communauté DevOps émergente, toute pratique DevOps devrait envisager la sécurité comme élément constitutif de ses processus. Si nous utilisons le terme SecDevOps (aka DevSecOps) dans ce document, c'est uniquement pour démontrer, voire identifier, les contradictions existant entre les DevOps traditionnels et ceux qui ont intégré une assurance de sécurité dans leurs processus.

2. Un aperçu de DevOps

Le diagramme suivant illustre le principe de base d'un processus DevOps comparativement à un processus SecDevOps (aka DevSecOps). Dans sa plus simple expression, le SecDevOps inclut une préoccupation en matière de sécurité. Le DevOps, quant à lui, considère uniquement l'amélioration du développement, des opérations et de la livraison.

DevOps VS. DevSecOps

Éléments clés

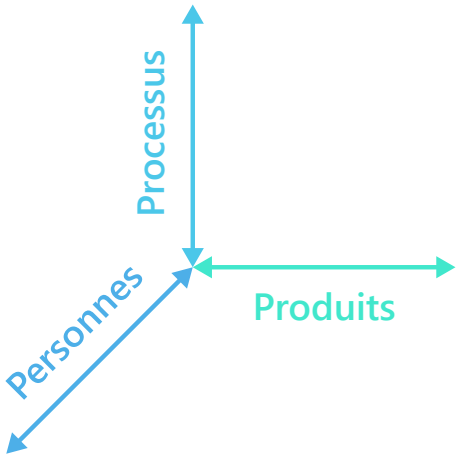
Avant d'aller plus loin dans cette approche de SecDevOps, quelques concepts importants qui s'y rattachent se doivent d'être révisés :

2.1 Trois dimensions

Pour traiter des domaines requis par SecDevOps, il faut d'abord tenir compte des éléments clés où interviennent habituellement les processus DevOps. Si l'on envisage DevOps comme un espace de croissance, cet espace est formé de trois axes principaux, X, Y, et Z, qui correspondent aux éléments suivants : Personnes, Processus et Outils (alias : produit). Chacun des axes a un rôle essentiel à jouer pour augmenter l'efficacité en matière de livraison de logiciels.

Le DevOps est communément décrit comme « ... l'union de personnes, processus et produits pour permettre la remise continue de valeur à nos utilisateurs finaux »

— Brown, 2015v (Darwish, 2018)



2.2 Le cycle de vie ou « le cycle PDVA (planifier-développer-vérifier-agir) »

Toutes ces dimensions peuvent croître ou s'améliorer selon les progrès réalisés. Mais pour réaliser des progrès, des actions doivent être exécutées. Par conséquent, un cycle de base peut être séparé en 4 parties : Planifier, Développer, Vérifier et Agir. Ce processus devient le levier naturel sur lequel agissent les personnes, force motrice de la croissance.

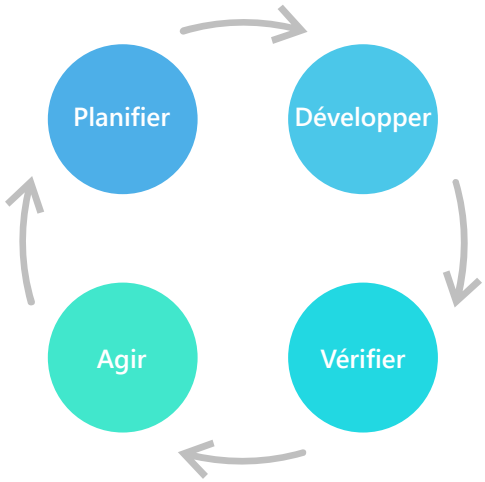
Comme vous pouvez le constater, un cycle est constitué de quatre éléments. Deux sont des points de contrôle pour la réflexion et deux pour l'action. D'où un équilibre entre agir et penser.

« Le cycle PDVA influe sur la fonctionnalité du logiciel, les plans de versions, les exigences, les tests, les environnements et la phase de livraison du logiciel. » — (Sharma, 2014)

La planification est certainement la phase la plus abstraite. Bien que certaines méthodologies telles que Agile, Scrum, CMMI et Kanban permettent de planifier et d'améliorer le développement de logiciels, elles ne traitent pas de l'amélioration continue lors de la phase de livraison.

Ainsi, DevOps peut être considéré comme une pratique, un changement de culture, où à chaque élément planifié devrait correspondre une action, une mesure du succès (le coefficient delta entre ce qui est livré et ce qui était attendu), un ajustement et un enseignement à tirer des modifications nécessaires à l'amélioration.

Des mesures objectives de différents éléments, telles que les défauts détectés, les délais de livraison et la satisfaction du client, permettent de constater si la planification ou les contrôles ont été efficaces en regard de ce qui a été fait.



Remarque : parfois, cette boucle peut inclure des étapes supplémentaires, comme la recherche, la démonstration, la reconnaissance et l'élimination de problèmes. Il n'existe pas qu'une seule « bonne façon de faire ».

C'est cette mutabilité, intrinsèque au DevOps, qui sera exploitée et réutilisée dans SecDevOps.

2.3 Changement dans les communications

« DevOps nécessite que chaque projet ait des canaux de communication clairs entre les équipes de développement et les personnes. » — (Vickrey, 2017)

Un autre changement proposé par DevOps est l'abolition des silos de travail, ce que nous appelons de manière plus conventionnelle « l'accroissement de la visibilité ». À proprement parler, le partage d'informations entre les différents acteurs liés au développement de logiciels de même qu'entre ceux de différents domaines peut aisément être considéré comme un outil d'amélioration.

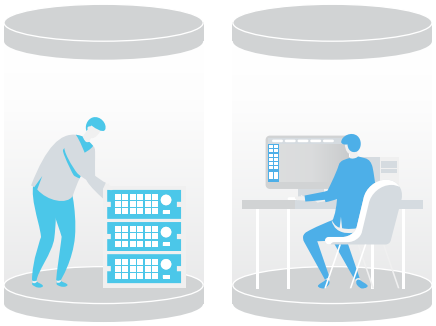
Voici deux des principaux arguments soutenus par les partisans de DevOps :

1. À l'étape de réalisation, la possibilité de voir ce qui est bien fait et de participer aux améliorations a un effet positif sur le moral de l'équipe. Les intervenants voient toute l'évolution de leur travail jusqu'à la phase de livraison.

2. En ayant les données sous la main, les intervenants sont autonomes et améliorent leur travail, d'où le besoin de rompre les silos. Les données circulent librement entre tous lors de la réalisation des logiciels, ce qui supprime, selon la situation, toute dépendance à la bureaucratie (ou bloqueurs artificiels).

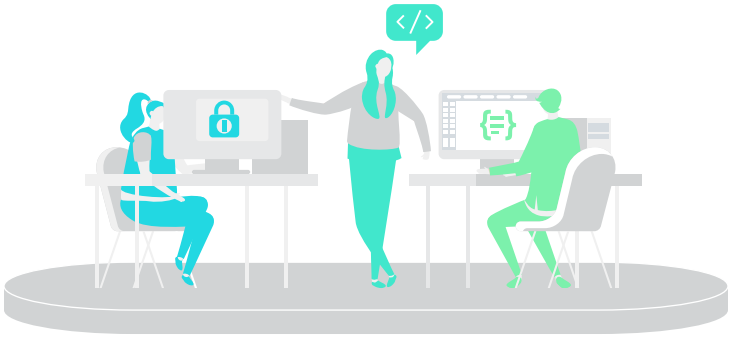
Des équipes de travail uniques sont une autre façon de supprimer les silos. Lors de points stratégiques déterminants, les intervenants peuvent communiquer verbalement entre eux et assurer la libre circulation des données.

Si nous envisageons Agile comme un processus de planification pendant la phase de « planification » et lors des rectifications, la proposition sous-jacente de DevOps consiste à ajouter, par exemple, des intervenants à l'infrastructure ou aux « opérations » lors des essais. La proximité physique, ou une simulation de celle-ci est le moyen structuré permettant de réduire les délais de communication lors de ces phases.



Méthode traditionnelle – Travail en silos

VS.



DevOps – Communication ouverte

2.4 Persistance des données

Un autre élément d'importance s'ajoute à l'élimination des bloqueurs de communication : « l'accès aux données en cas de besoin ». Cela peut se faire à partir d'une plateforme commune donnant accès à toutes les données partagées. Ce qui nous ramène à la question des « outils ». L'utilisation d'outils adéquats rend plus facile et plus sûre la phase de livraison. Complémentaires à la communication, ces outils permettent de donner une pertinence à toutes les données de communication et de livraison.

Une plateforme de travail collaborative avec partage de données est donc fortement recommandée. Autant il faut chercher à éliminer les silos susceptibles d'affecter l'efficacité d'une équipe, autant il faut veiller à ce que les connaissances soient partagées entre tous ses membres afin d'éviter toute responsabilité liée à une mauvaise communication.

Une base de données de gestion de configuration (alias : CMDB) est un outil puissant pour optimiser la persistance des exigences en amélioration continue.

Un trio pour maintenir l'unité

Des mesures visant les personnes, les processus et les outils pour assurer l'amélioration continue ont déjà été discutées, mais voici certaines des principales techniques utilisées pour les appliquer. Si ces éléments sont ici mis en évidence, c'est que, dans DevOps et SecDevOps, des points communs existent entre eux.

2.5 Personnes

Il a déjà été expliqué en quoi la dégradation des communications devient un facteur de collaboration humaine. Comme il a été décrit précédemment, encourager la proximité est l'un des moyens qui permettent d'améliorer la communication et l'efficacité.

La technique spécifique souvent utilisée pour franchir un obstacle majeur est de faire participer les équipes des « ops » dès le démarrage du projet et lors des différentes phases de planification à chaque étape de développement du logiciel.

Une autre technique consiste à donner de la latitude et de l'autonomie aux développeurs lors de la construction de l'infrastructure (parfois appelée « infrastructure en tant que code »).

La répartition du mérite et de la responsabilité entre tous les membres de l'équipe entraîne la motivation et sensibilise aux problèmes liés aux tâches de chacun. La chaîne est vue comme un ensemble dont chaque lien est essentiel. Cela est aussi vrai pour la sécurité que pour les « ops ».

2.6 Processus

Faire preuve d'une certaine rigueur, que ce soit dans le flux de travail ou à l'aide d'un processus documenté (et évolutif), est également considéré comme un moyen efficace d'améliorer l'efficacité. Puisque ce processus est traité comme un artéfact — tout comme lors d'une modification de code — il est possible de surveiller et de vérifier quels changements ont influé sur la performance.

« Un processus est une série d'étapes et de décisions impliquées dans la façon dont le travail est accompli. » — (Baird)

Il est difficile d'exprimer en termes simples ce que ceci a de crucial dans le succès ou l'échec d'un processus de livraison de logiciels. Disons que, si les personnes constituent la puissance brute derrière un projet, le processus en est le contrat ou la recette pour sa réussite.

Il est essentiel de comprendre qu'en lui-même, le processus de développement logiciel (DevOps) représente une méthode pour diriger un projet de livraison de logiciel. S'il existe, c'est pour créer l'ordonnancement et la structure du travail au quotidien et s'assurer que rien de connu ou de visible n'est laissé au hasard. Tous ces contrôles visent à remettre de la valeur à l'utilisateur final.

« Lorsque vous devez optimiser la rapidité de votre modèle de livraison, il est essentiel d'examiner vos processus et de "couper dans le gras". Pour une organisation, cela équivaut

à établir autant de processus reproductibles que possible en éliminant autant de tâches manuelles que possible, ceci afin d'accélérer le processus dès l'enregistrement du code et jusqu'à la production. » — (Gene Kim, 2013)

2.7 Produits (outils)

Ce segment suscite généralement beaucoup d'enthousiasme dans l'industrie. L'automatisation et le contrôle des artéfacts sont souvent cités comme principales raisons de commencer une entreprise DevOps. L'idée de posséder une plateforme logicielle en entreprise est d'accélérer les processus et d'imposer une certaine rigueur dans l'application des flux de travail. En adhérant à cet argument, il semble alors nécessaire d'utiliser ce type d'outils logiciels lors du développement de logiciels. D'ailleurs, beaucoup de produits existent déjà pour répondre à la demande et en posséder lors des phases de livraison reste tout à fait pertinent.

On utilise également ces « outils » pour suivre la progression ou toute régression dans le cycle de livraison d'un logiciel.

Dans chaque entreprise, on s'efforce constamment d'améliorer l'efficacité des processus en se servant d'outils, dont des solutions logicielles. Il est donc plus que raisonnable de penser qu'en amont, les créateurs de logiciels en utilisent eux-mêmes.

(Les particularités de ces techniques sont hors du domaine visé par ce document. Nous n'allons donc pas les approfondir. Il nous faut simplement les garder à l'esprit pour comprendre de quelle façon SecDevOps y est lié).

3. Parallèle avec SecDevOps

Un certain temps a été consacré à comprendre de quelle façon les principes DevOps sont censés influer sur le développement et la livraison de logiciels. Afin de mieux saisir en quoi DevOps et SecDevOps sont en réalité très similaires, nous avons tenté de définir quelques nuances et approches de base censées améliorer le processus de développement logiciel.

« DevSecOps est comme DevOps, mais avec des principes de sécurité intégrés. » — (Matteson, 2017)

Ce qui a changé

Les changements sont, heureusement assez subtils pour être facilement implémentés ou, malheureusement assez subtils pour se glisser au travers des mêmes considérations techniques, ce qui devrait [dans les deux cas] simplifier leur implémentation.

Toutefois, simple ne signifie pas nécessairement facile. Tout comme l'implémentation des pratiques DevOps doit

s'effectuer à un rythme croissant, les changements doivent s'effectuer avec soin pour ne pas biaiser la production d'un système de livraison de logiciels qui s'est montré efficace. Il s'agit d'une très bonne leçon à tirer, décrite dans l'ouvrage " The Phoenix Project ".

Ainsi, la valeur ajoutée suite à toute modification envisagée lors de l'implémentation de même que ses liens avec les priorités de l'entreprise nécessitent d'être bien mesurés.

De façon empirique, il est évident que la sécurité des technologies de l'information est importante. Intégrer (intrinsèquement) les exigences de sécurité dans le processus de développement logiciel est certainement un objectif louable, mais cela reste plus facile à dire qu'à faire.

C'est ici que le côtoiement entre DevOps et SecDevOps devient plus évident. Si DevOps constitue une mesure pratique pouvant être prise, ne pourrions-nous pas simplement utiliser les mêmes techniques de base pour implémenter SecDevOps ? Cette approche est celle préconisée par Emyode.

Revenir en arrière

Cette section explique comment certaines des propositions d'origine pour l'implémentation d'une culture DevOps s'appliquent également à une proposition SecDevOps, tout en soulignant certaines différences et certains défis potentiels qui s'y ajoutent.

Ironiquement, la mentalité DevOps crée une unité d'activation parfaite pour insérer la sécurité dans les trois axes décrits précédemment. De simples petits changements incrémentiels rendent la chose très réalisable.

3.1 SecDevOps et les personnes

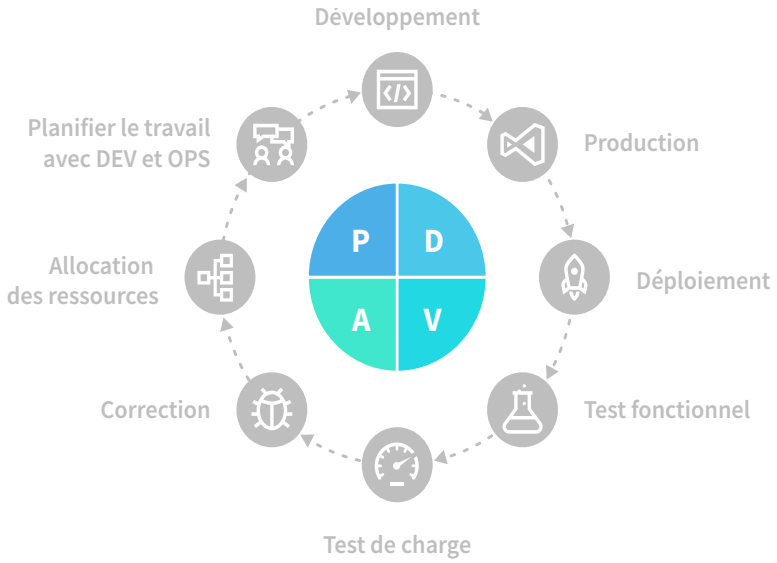
Quelles sont les nouvelles informations concernant les personnes qui peuvent s'appliquer à SecDevOps ? Une proximité devrait exister entre les gens de sécurité informatique et les équipes de développement de logiciels. Cette déclaration parait simple et évidente, mais les enjeux qu'elle sous-tend viennent démentir cette apparente simplicité.

En effet, lorsque l'on considère qu'il devrait y avoir d'un à six administrateurs de sécurité pour 100 développeurs (Aubuchon, 2010), il y a difficulté évidente à ce que ces deux équipes passent du temps ensemble.

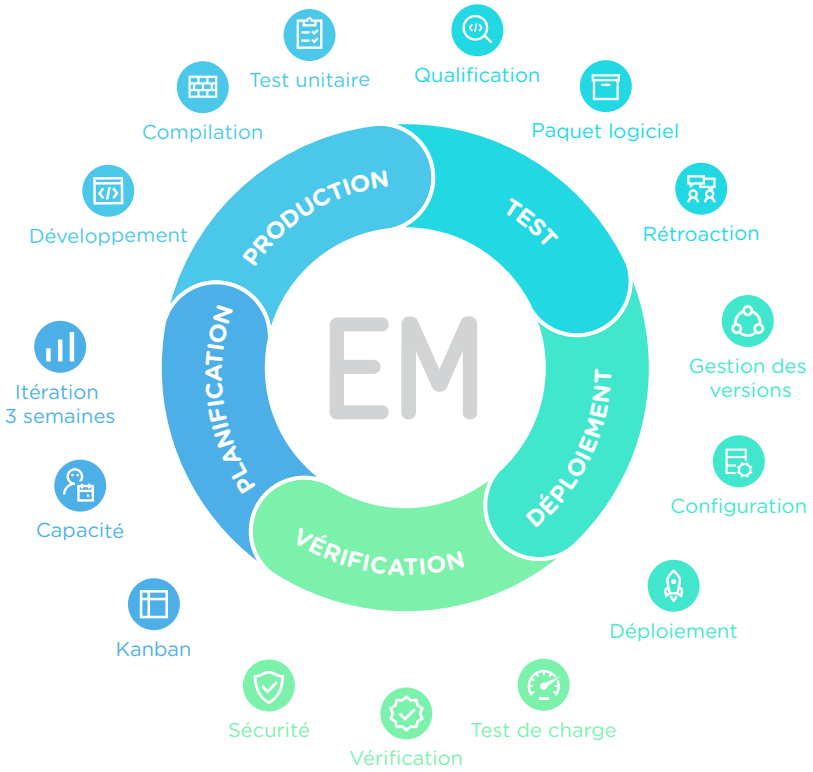
Une campagne de sensibilisation est une des solutions proposées pour résoudre ce problème. Lorsque la sécurité devient une priorité pour tous les membres d'une équipe de développement logiciel, en particulier s'ils sont spécifiquement formés, la probabilité qu'ils interrogent ou collaborent avec un OPS de sécurité de manière préventive et transparente avant de commencer leur travail augmente de beaucoup. La proximité physique mentionnée plus tôt appuie cette méthode aussi.

3.2 SecDevOps et les processus

Pour les processus, il est plus simple de définir ce qui peut être fait. À partir d'un cycle PDVA (Planifier, Développer, Vérifier, Ajuster) de base, il est possible de démontrer comment insérer la sécurité dans un processus existant.



Formulons l'hypothèse que la figure ci-dessous représente un processus DevOps typique.



On retient des leçons du DevOps traditionnel, mais on prend conscience des intérêts de sécurité. Et pour inclure la sécurité, il faut tout d'abord la planifier comme ressources dans l'équipe dès le début du projet, et ce, jusqu'à l'étape post-déploiement:

- Inclure les gens de la sécurité dans la planification
- Former et sensibiliser le développement
- Mettre des balises d'acceptabilité
- Isoler les étapes critiques des accès génériques
- Qualifier le déploiement
- Inclure la validation « post production »

Remarque: la phase « sécurité » représente tous les ajustements, bogues fonctionnels et problèmes de sécurité/ assurance qualité.

Ce modèle pourrait se nommer PDCDCDA, mais le principe reste le même. Tout d'abord la planification, et ensuite, pour chaque « exécution », un « contrôle » et un « ajustement » sont effectués juste avant le début du cycle suivant.

Considérez à quel point ces changements ne sont pas excessifs et, s'ils font déjà partie d'un processus éprouvé, assez faciles à assimiler. Si vous avez utilisé un flux de travail (ou workflow) avec champs personnalisés, il suffit de cocher plus de cases. Conceptuellement, l'ajout d'étapes de sécurité dans un processus est facile à comprendre. Le choix des tâches et des outils réels peut être un peu plus rébarbatif, mais dans les faits, l'insertion de tâches pertinentes pour la sécurité est tout à fait envisageable.

Il est possible par exemple d'imaginer une procédure détaillée de ce qui doit être contrôlé en cas de problème afin de garantir la sécurité de l'information (par exemple, la modification de tous les mots de passe en cas de faille de sécurité).

3.3 SecDevOps et produits (outils)

Dans toute chaîne de production, le maillon le plus faible est le facteur limitant l'efficacité de l'ensemble de l'entreprise. Le choix des outils est donc un point essentiel à traiter.

Il n'existe pas de recette magique pour garantir le succès, mais quelques recommandations dans le domaine DevOps vous aideront à améliorer vos chances de succès. Voici certains des éléments les plus pertinents.

- S'assurer que les outils correspondent aux besoins.
- Ne pas imposer d'outils douteux ou susceptibles d'être mal utilisés.
- Sélectionner des outils en fonction de la maintenabilité et de la fiabilité des mises à jour.
- Sélectionner les outils qui ont fait leurs preuves en matière de fiabilité.
- S'assurer que les outils sont bien compris et maitrisés, bien utilisés et répartis entre plusieurs personnes de l'équipe.
- S'assurer que les données mesurant les indicateurs KPI qui soutiendront l'entreprise DevOps sont bien recueillies et clairement affichées par les outils.
- S'assurer que les différents cycles de la chaîne sont facilement intégrés : c.-à-d. les outils de planification, outils de contrôle du code source, outils de développement, outils de test, outils de déploiement, etc.
- Sélectionner des outils qui encouragent une expérimentation à risque négligeable.
- Sélectionner des outils qui encouragent l'autonomie.

Si l'on convient que ces points valent la peine d'être traités, voici comment SecDevOps va transformer les outils déjà utilisés.

- Des données sont déjà recueillies et la télémétrie des KPI d'amélioration continue est déjà pratiquée. Ces données seront exploitées pour déclencher des alertes de sécurité.
- Des outils de planification sont déjà utilisés pour gérer le travail. Des artéfacts y seront inclus pour répondre aux préoccupations en matière de sécurité.
- Des points de contrôle établis dans les processus permettent de laisser fonctionner ou de bloquer la promotion de produits vers la production finale. Ces points de contrôle seront exploités pour identifier et résoudre les problèmes de sécurité.
- Une chaîne d'outils multi-étapes permet d'intégrer la validation du paquet/contenu. Lors de ces validations, de la même manière que sont traités les autres critères, une inspection qualifiant ou rejetant le contenu selon des critères de sécurité définis peut être insérée.
- Dans le cas des outils de gestion des tests, l'ajout d'un nouveau critère de garantie de sécurité suffit.

Un exemple serait d'avoir des outils automatisant les audits de sécurité ou détectant automatiquement les failles de sécurité sur l'ordinateur où le produit est installé.

La liste pourrait s'allonger sans fin, mais il faut garder à l'esprit que toute initiative DevOps doit être mise en œuvre selon de courtes étapes progressives. L'essence même de DevOps est d'effectuer les changements à la chaîne d'outils le plus efficacement possible, par petite dose et au plus faible risque.

Conclusion

DevOps, c'est le changement. C'est l'amélioration continue. C'est un gain d'efficacité. C'est une meilleure visibilité et une plus grande collaboration.

SecDevOps ne doit pas modifier cela, il doit exploiter cela. Dans un projet DevOps, l'insertion d'impératifs de sécurité dans l'un ou l'autre des axes essentiels devrait conserver ces valeurs et SecDevOps devrait fonctionner aussi bien que DevOps.

Dans cette perspective, si vous n'oubliez pas d'inclure vos préoccupations en matière de personnes, de processus et de pipelines de livraison, votre entreprise devrait être un succès.

Ouvrages cités

Aubuchon, Kurt. 2010. How Many Information Security Staff Do We Need? Infosec Island. [Online] September 26, 2010. http://www.infosecisland.com/blogview/8327-How-Many-Information-Security-Staff-Do-We-Need.html.

Baird, Scott. What is a Process? Processmodel. [Online] https://www.processmodel.com/blog/what-is-a-process/.

Brown, Donovan. 2015. What is DevOps. donovanbrown.com. [Online] September 01, 2015. http://donovanbrown.com/post/what-is-devops.

Darwish, Wesam. 2018. Premier Developer. blogs.msdn.microsoft.com. [Online] May 20, 2018. https://blogs.msdn.microsoft.com/msind/2017/03/20/how-devops-are-changing-the-face-of-it/.

Gene Kim, Kevin Behr, George Spafford. 2013. The Phoenix Project. s.l.: IT Revolution Press, 2013.

Matteson, Scott. 2017. Riding the DevOps Revolution. ZDNET. [Online] April 3, 2017. https://www.zdnet.com/topic/riding-the-devops-revolution/.

Sharma, Sanjeev. 2014. Adopting DevOps for continuous innovation. IBM.com. [Online] June 02, 2014. https://www.ibm.com/developerworks/library/d-devops-continuous-innovation/index.html.

Vickrey, Nate. 2017. Best Practices for Internal Communication in DevOps. Devops.com. [Online] May 22, 2017. https://devops.com/best-practices-internal-communication-devops/.