1 - L’importance du monitoring

1. L’importance du monitoring

Lorsque vous **mesurez vos progrès**, vous pouvez constater votre amélioration, c'est également un **puissant motivateur** pour vous aider à tenir vos objectifs. Plus vous progresserez, plus vous serez confiant et motivé.
Le fait même que vous soyez en progression vous donnera l'élan nécessaire pour **atteindre vos objectifs**. Mesurer vos progrès vous donnera finalement une image très réaliste de la situation. Si vous ne réalisez pas les progrès souhaités, vous pouvez l'apercevoir avant de prendre trop de retard. Vous pouvez ensuite **identifier les changements à apporter** et les mettre en œuvre.

La **première étape** pour mesurer vos progrès est d'**établir** des **objectifs clairs et précis**.

Lorsque vous vous fixez un objectif, vous devez vous assurer qu’il puisse être **mesuré.**

Par **exemple**, plutôt que de dire **"je veux perdre du poids"**, vous devez déterminer **"combien de poids vous voulez perdre"**. L'objectif que vous vous êtes fixé déterminera la méthode utilisée pour mesurer vos progrès. Par exemple, si plutôt que de perdre du poids, vous souhaitiez maigrir, vous pourriez dire que vous vouliez perdre 3 pouces de votre taille. Dans ce cas, le meilleur moyen de mesure serait un ruban à mesurer, où, comme pour la perte de poids, une balance serait la meilleure forme de mesure.

1.1 Le modèle à trois voies

Le "**three way model**" ou "**modèle à trois voies**" en français est un principe dont tous les modèles DevOps doivent dériver.

Il en est fait référencé dans deux ouvrages de référence du DevOps : le "DevOps Handbook" et "The Phoenix Project".

Ce modèle décrit les **valeurs** encadrant les **processus**, les **procédures** et les **pratiques à implémenter** dans une **équipe DevOps**.

a. Première voie : penser globalement

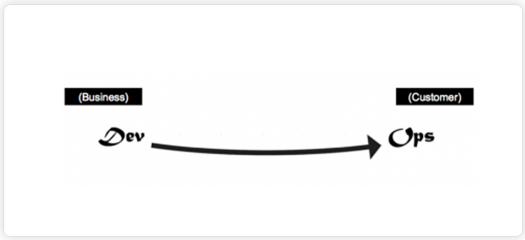


Figure 1 : Première voie

La **"Première Voie"** met l'accent sur la **performance de l'ensemble du système**, par opposition à la performance d'un silo de travail ou d'un département spécifique - cela peut être une division aussi grande (par exemple, Développement ou Opérations informatiques) ou aussi petite qu'un contributeur individuel (par exemple, un développeur, un administrateur système).

L'accent est mis sur **tous les flux de valeur métier** qui sont rendus possibles par les **technologies de l'information**.

En d'autres termes, elle commence lorsque les exigences sont **identifiées**, sont **intégrées dans le développement**, puis **transférées dans les opérations informatiques**, où la valeur est ensuite livrée au client sous la forme d'un service.

Les **résultats** de la mise en pratique de la **Première Voie** comprennent le fait de **ne jamais transmettre un défaut connu** aux centres de travail en aval, de **ne jamais permettre à l'optimisation locale de créer une dégradation globale**, de **toujours chercher à augmenter le débit** et de **toujours chercher à obtenir une compréhension profonde** du système (selon la roue de Deming).

b. Seconde voie : Amplifier les boucles de rétroaction

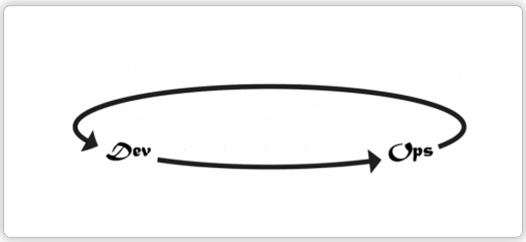
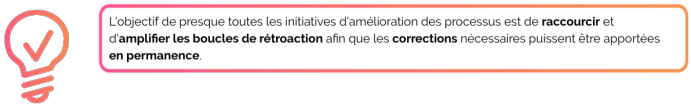


Figure 2 : Seconde voie

La **"Seconde voie"** consiste à créer des boucles de rétroaction de droite à gauche.



c. Troisième voie : Culture d'apprentissage et d'expérimentation continue

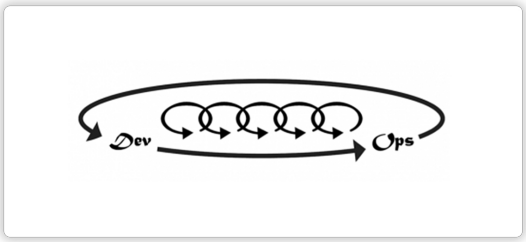
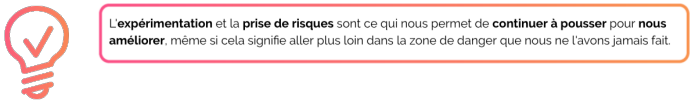


Figure 3 : Troisième voie

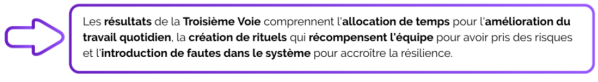
La "**Troisième Voie**" consiste à créer une culture qui favorise deux choses :

- L'expérimentation continue, la prise de risques et l'apprentissage de l'échec
- La compréhension du fait que la répétition et la pratique sont les conditions préalables à la maîtrise.

Nous avons besoin de ces deux éléments de la même façon.



Et nous devons maîtriser les compétences qui peuvent nous aider à sortir de la zone dangereuse lorsque nous sommes allés trop loin.

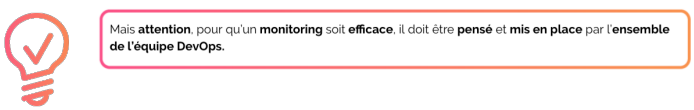


1.2 Un grand pouvoir implique de grandes responsabilités

De nombreuses organisations ont du mal à définir les objectifs commerciaux, les processus et les structures organisationnelles pour motiver et renforcer efficacement la collaboration au sein de l'équipe.

Avec des **mesures de performance** qui peuvent être revues et partagées universellement, vous pouvez suivre vos efforts de DevOps et prouver le **succès à chaque étape**.
Les **leaders** tirent profit de savoir que **tout le monde est aligné et progresse vers les mêmes objectifs**. Les membres de l'équipe peuvent ainsi collaborer plus facilement et plus rapidement, avec des **idées partagées.**

J'espère que maintenant vous n'imaginez plus concevoir un pipeline de production sans outils de monitoring intégré.



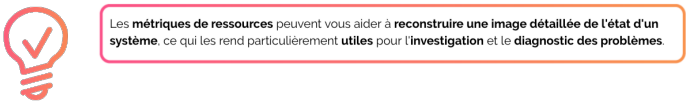
Si ce n'est pas le cas, vous allez vous heurter à des **problèmes** : au fur et à mesure de l'évolution de vos processus et de vos applications, les règles de monitoring risquent de ne plus être adaptées. Vous arriverez alors à un **état de monitoring incohérent** avec des générations d'incidents, des appels d'astreintes. Tout ça, car vous n'avez pas adapté votre monitoring à vos changements dus le plus souvent à un **manque de communication et de responsabilité** de chacun.

Ce genre de problème **ne devrait pas apparaitre dans une équipe DevOps**, car la communication n'y est plus un problème, mais un **atout**.

1.3 La collecte de métriques

Pour que votre monitoring soit **le plus efficace possible**, il doit être présent sur **l'intégralité de vos environnements, du développement à la production**. Pour chaque nouvelle version de vos applications, vous devrez vous poser la question des changements qui peuvent impacter votre monitoring.

La plupart des composants de votre infrastructure logicielle peuvent servir de **ressources à d'autres systèmes**, on parle alors de **ressources logicielles**. Certaines ressources sont de type hardware, c'est le cas des ressources d'un serveur tel que le CPU, la mémoire, les disques et les interfaces réseau. Mais une composante de niveau supérieur, comme une base de données ou un microservice de géolocalisation, peut aussi être considérée comme une **ressource** si un autre système nécessite cette composante pour produire du travail.



Pour **chaque ressource de votre système**, essayez de **collecter des métriques** qui couvrent **quatre domaines clés** :

- L'**utilisation** est le **pourcentage de temps pendant lequel la ressource est occupée**, ou le pourcentage de la capacité de la ressource qui est utilisée.
- La **saturation** est une mesure de la quantité de travail demandé que la ressource ne peut pas encore servir, souvent en file d'attente.
- Les **erreurs** représentent des erreurs internes qui peuvent ne pas être observables dans le travail produit par la ressource.
- La **disponibilité** représente le pourcentage de temps pendant lequel la ressource a répondu aux demandes. Cette mesure n'est bien définie que pour les ressources dont la disponibilité peut être vérifiée activement et régulièrement.

Voici des exemples de métriques pour les ressources les plus communes :

Ressource	Utilisation	Saturation	Erreurs	Disponibilité
Disk IO	% du temps d'occupation du disque	Longueur de la file d'attente	Nombre d'erreurs sur le disque	% du temps disponible p l'écriture
Memory	% d'utilisation de la mémoire totale disponible	Utilisation du swap	N/A	N/A
Microservice	Moyenne % du temps où les threads de service sont occupées	Nombre de requêtes mises en file d'attente	Nombre d'erreurs internes comme les levés d'exceptions	% du temps ou un service joignable
Base de Données	Moyenne % du temps où les connexions sont occupées	Nombre de requêtes mises en file d'attente	Nombre d'erreurs interne, comme la réplication	% du temps où la base données est joignable