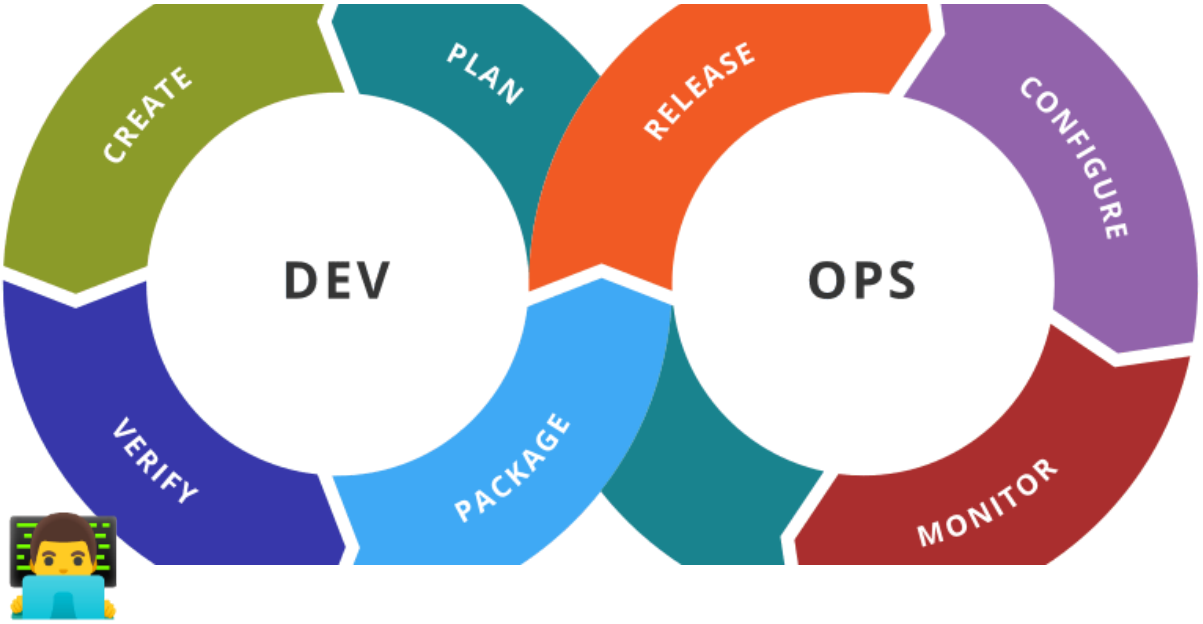

DevOps

Ingénieur DevOps

“Plus qu’un métier, une méthode, Au-delà de la méthode... une philosophie !”.

▼ Qu’est ce que le DevOps ?

DevOps signifie “Développement et Opérations”.

Le terme DevOps a été utilisé pour la première fois en 2007 en Belgique

Avant la création du métier d’ingénieur DevOps, quand on développait un projet les équipes étaient catégorisés en deux grandes familles : les **développeurs qui codent** et les **administrateurs systèmes et réseaux qui configurent et surveillent la santé des machines.**

Durant les études, il fallait choisir sa voie entre ces deux choix. Mais avec l’émergence de nouvelles technologies et de nouvelles demandes, les équipes jusque-là séparées devaient travailler de plus en plus ensemble et donc trouver de nouvelles façons d’échanger en modifiant leurs méthodes et leurs outils.

C’est ainsi que naquit les développeurs opérationnels (DevOps).

▼ Quel est son rôle ?

Le rôle des ingénieurs DevOps est d'utiliser un ensemble de pratiques et de méthodes qui ont pour objectif d'améliorer la stabilité et la vitesse de production des équipes en mettant l'accent sur la **collaboration et la communication** entre les développeurs et les équipes d'infrastructure.

Ils collaborent avec tous les développeurs pour automatiser les processus de déploiement, de test et de gestion de l'infrastructure et sont responsables de la conception et de la mise en œuvre des applications, des logiciels et des services pour leur organisation.

Un ingénieur DevOps doit aussi avoir une vaste étendue de connaissances/de culture liées à la tech (Os, langages, métiers, logiciels, sécurité, nouvelles technologies : Cloud, ia..) et une connaissance plus approfondie des **outils** liés à son métier.

Il est de nature sociable, patient, tolérant et a une communication claire pour gérer ses équipes. Il a un réseau conséquent (en réel ou virtuel). Il sait trouver des solutions de manière méthodique. Il peut travailler sous pression comme celle du temps, des demandes, des personnes et des **outils** imposés au lieu de travail.

▼ ~~DevOps~~ vs DevOps

L'équipe de développeurs (Dev) = collecte les exigences du projet pour développer le code
Elle va tester le code puis va livrer le projet à…
L'équipe de production et d'exploitation (Ops) = ils vont mettre cette application en production puis gérer sont exploitation
Ceci dit, ce modèle pose problème car les deux équipes ont des objectifs différents
D'un côté les Dev doivent faire évoluer les applications rapidement pour répondre au demande du marché (Time to market réduit = délai entre l'idée initial et la concrétisation du projet. Dans un contexte économique concurrentiel, un TTM rapide pour être décisif pour une entreprise ou un prestataire comme ESN. TTM rapide = permet au entreprise de s'adapter et de répondre plus rapidement à la demande du marché. Mais aussi pour proposer de l'innovation au client avant les autres)
De l'autre côté, les Ops se charge de garantir la stabilité du système. Cela implique un contrôle très sécère des changements apportés au système

d’information. Une panne de production pouvant couter très très cher à l’entreprise

C’est cet antagonisme qui crée des conflits d’intérêts entre ces deux branches
Les Dev blâment les Ops pour les retards et les problèmes de livraison
Et les Ops tiennent l’équipe Dev pour responsable des incidents en production lié à la mauvaise qualité du code

Et l’approche DevOps cherche donc à réconcilier ces deux mondes

▼ Méthodes DevOps

- Le DevOps suit un **cycle** que l’on peut résumer ainsi :

Discover : En vue du prochain sprint, les équipes doivent s’organiser et hiérarchiser leurs idées. Les idées doivent être alignées sur des objectifs stratégiques et avoir un impact sur les clients.

Plan : Les équipes DevOps doivent adopter des méthodes Agile pour améliorer la vitesse et la qualité. Les méthodes Agile aident les équipes à diviser le travail en tâches plus petites pour accélérer le temps de développement.

Build : On utilise un outil de versionning tel que Git. Git est un système de contrôle de version gratuit et open source. Il dispose d'un excellent support pour les branches, les merges et la réécriture de l'historique du dépôt, ce qui a entraîné l'apparition de nombreux workflows et outils innovants et utiles pour le processus de développement.

Test : L'intégration continue (CI) permet à plusieurs développeurs de contribuer dans un dépôt partagé unique. Lorsque des changements du code sont mergés, des tests automatisés s'exécutent afin d'en vérifier l'exactitude avant toute intégration. Les merges et les tests de code aident souvent les équipes de développement à s'assurer de la qualité et de la prévisibilité du code une fois le déploiement terminé.

Deploy : Le déploiement continu (CD) permet aux équipes de livrer des fonctionnalités en production fréquemment et de façon automatisée. Les équipes peuvent également effectuer le déploiement à l'aide de feature flags (Un feature flag (encore appelés « feature toggle », « feature switch » ou « feature flagging ») permet l'exécution conditionnelle du code d'un site ou d'une application pour

activer ou désactiver une fonctionnalité), afin de livrer du nouveau code aux utilisateurs régulièrement et méthodiquement, plutôt que d'un seul coup. Cette approche améliore la vélocité, la productivité et la durabilité des équipes de développement logiciel.

Operate : Gérez, de bout en bout, la livraison de services informatiques aux clients. Cela inclut les pratiques impliquées dans la conception, l'implémentation, la configuration, le déploiement et la maintenance de toute l'infrastructure informatique qui sous-tendent les services d'une organisation.

Observe : Identifiez et résolvez rapidement les tickets qui ont un impact sur le temps d'activité, la vitesse et les fonctionnalités des produits. Informez automatiquement votre équipe des changements, des actions à haut risque ou des pannes, afin que vous puissiez assurer la continuité des services.

Continuous feedback : Les équipes DevOps doivent évaluer chaque version et générer des rapports pour améliorer les livraisons futures. En recueillant un feedback continu, les équipes peuvent améliorer leurs processus et intégrer le feedback des clients pour améliorer la prochaine version.

▼ Les outils

Un DevOps doit maitriser plusieurs types d'outils comme des outils de versionning *(slide)*, des outils de script *(slide)*, des outils d'automatisations *(slide)*, des outils de gestion de configuration (slide), des outils de conteneurisation *(slide),* des outils de surveillance et de journalisation *(slide)*, des outils de collaboration *(slide)*, des outils de déploiement continu *(slide)*, des outils de test *(slide)*, des outils de virtualisation *(slide)*, des outils de gestion de base de données *(slide)*.

▼ Outils à slide : (preciser qu'ici ce n'est que des exemples)

Outils de gestion de versions :

- Git : Pour le contrôle de version et la collaboration dans le développement de logiciels.

Outil de script :

- Python.

Outils d'automatisation :

- Jenkins : Pour l'automatisation des builds, des tests et des déploiements.

- Travis CI, CircleCI : Des outils d'intégration continue pour automatiser les tests et les déploiements.

Outils de gestion de configuration :

- Terraform : Pour la gestion de l'infrastructure en tant que code.
- Ansible, Puppet, Chef : Pour automatiser la configuration et la gestion de l'infrastructure.

Outils de conteneurisation :

- Kubernetes : Pour orchestrer et automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées.
- Docker : Pour la création, le déploiement et l'exécution d'applications dans des conteneurs.

Outils de surveillance et de journalisation :

- Prometheus : Pour la surveillance des systèmes et des services.
- ELK Stack (Elasticsearch, Logstash, Kibana) : Pour la collecte, l'analyse et la visualisation des journaux.

Outils de collaboration :

- Jira, Trello : Pour la gestion des tâches, le suivi des projets et la planification agile.
- Slack, Microsoft Teams : Pour la communication et la collaboration d'équipe.

Outils de déploiement continu :

- Spinnaker : Pour l'automatisation du déploiement continu.
- GitLab CI, GitHub Actions : Intégration native avec des référentiels Git pour automatiser les pipelines CI/CD.

Outils de test :

- JUnit, PHPUnit : Pour les tests unitaires.
- Selenium : Pour les tests d'interface utilisateur.

Outils de virtualisation :

- Vagrant : Pour la création et la gestion d'environnements de développement virtualisés.

Outils de gestion de base de données :

- MySQL, PostgreSQL : Bases de données relationnelles populaires.
- MongoDB, Redis : Bases de données NoSQL souvent utilisées dans les applications Backend.

▼ Comment devenir DevOps ? Et quelles possibilités d’évolutions ?

▼ Pour devenir DevOps, il y a plusieurs voies possibles :

- La plupart suivent un cursus de développeur et se construisent une expérience au fur et à mesure.
- D’autres évoluent par le biais de formations tout au long de leur carrière (au sein de l’entreprise).

Dans tous les cas, sauf en ce qui concerne les autodidactes, le niveau de base est bac+3.

▼ Possibilités d’évolutions :

DevOps peut être un objectif en soi mais il permet aussi de progresser en tant que :

Lead DevOps : c’est-à-dire en étant responsable de projets et manager

VP of Engineering : pour assurer la bonne exécution technique d’une vision produit

Chief Technical Officer (CTO) : pour être en charge de l’innovation technique et le déploiement de technologies adaptées

Chef de projet infrastructure : pour mener à terme la réalisation d’un projet d’infrastructure, tout en trouvant un équilibre en termes de tarif, de délai et de qualité