PLATFORM ENGINEERING : FAITES PASSER LE DEVOPS À L'ÉCHELLE

PAUL BOISSON

LIVRE BLANC

ippon

SOMMAIRE

PAUL BOISSON

Formé en Génie des procédés industriels, j'ai basculé vers l'informatique en 2020. Je suis aujourd'hui Ingénieur Cloud & DevOps chez Ippon Technologies, formateur GitLab CI/CD en interne et chargé de monter une offre autour du Platform Engineering, depuis avril 2024. Au début de ma carrière, j'ai travaillé sur les sujets d'amélioration continue et j'ai toujours aimé optimiser les tâches : d'où mon intérêt pour le Platform Engineering. Mon premier objectif a été de définir les contours de cette discipline chez Ippon Technologies, en interrogeant les consultants qui la pratiquent.

Des tables rondes internes ont permis de mettre en évidence nos différentes expériences autour du Platform Engineering en tant que consultants en mission chez divers clients, tout en mettant en lumière le besoin de consolider nos connaissances communes. Depuis 2024, le Platform Engineering est une tendance très forte auprès des PME et grands groupes que nous accompagnons et qui fait partie de notre stratégie en 2025. Ce livre blanc incarne notre vision et est un premier pas vers la construction d'une offre commune à nos différentes practices.

INTRODUCTION

Les « plateformes » que tout le monde connaît sont celles des géants de la Tech : Uber, Deliveroo, Airbnb, Booking... Ces marques se sont positionnées en intermédiaires : avec une interface simple qui permet aux professionnels de se focaliser sur le service qu'ils doivent fournir et aux consommateurs d'accéder rapidement à un large choix de services.

Mais la **plateformisation** est un phénomène qui touche tous les domaines aujourd'hui.... et jusqu'à la DSI. En effet, vos équipes de développement peuvent désormais interagir avec les infrastructures, grâce à une plateforme qui permet de gérer l'ensemble du cycle de vie d'une application.
Côté Data, c'est pareil : on centralise et on met à disposition toutes les données de l'organisation de manière structurée et accessible pour tous les usages.

Sur Welcome to The Jungle, on comptait déjà, à la mi-mars 2025, 167 offres d'emploi pour la requête « Platform Engineer ». Sur LinkedIn, les posts consacrés au Platform Engineering ont pris le pas sur les posts autour du DevOps.

Cet ouvrage vous apporte un premier aperçu complet de l'approche « Platform », avec des témoignages d'experts, conseils de pionniers et cas clients.

REMERCIEMENTS

Mes collègues sont nombreux à avoir contribué à la rédaction de ce livre blanc, en me faisant part de leur point de vue, en m'aidant à rédiger certaines parties ou en apportant des corrections.

Je tiens d'abord à remercier Timothée AUFORT, Architecte Cloud et Practice Leader Cloud & DevOps, qui m'a permis de travailler sur le sujet du Platform Engineering au sein de notre practice, ainsi que Julide YILMAZ, architecte MLOps, qui a travaillé sur la partie Plateforme Data, Vivien MALEZE, Architecte Technique, Julie KOCIANOVA, Product Owner, et Rémy OLIVET, Practice Leader Data, qui m'ont apporté leur point de vue. Ils interviennent tous les quatre dans cet ouvrage sous forme d'interviews.

Merci également à tous ceux qui ont permis à cet ouvrage de voir le jour en corrigeant les textes, en s'occupant de la mise en page, en questionnant certains points. Je remercie à ce titre Jean-Rémy REVY, Alice DEROYE, Adrien PAYSANT, Nicolas LE GRIEL, Charlotte PHILIPPE, Arnaud FREISMUTH, Arnaud COSPAIN et Cédric PREZELIN.

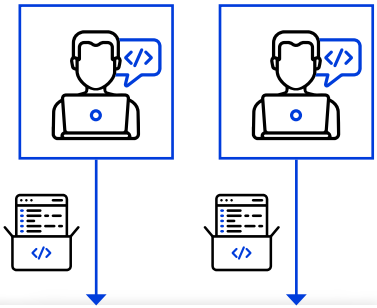
LE DÉVELOPPEMENT INFORMATIQUE, 30 ANS DE PROGRÈS

AVANT D'ABORDER LES ENJEUX ACTUELS, JE VOUS PROPOSE UN PAS DE RECUL POUR EXAMINER LA TRAJECTOIRE D'ÉVOLUTION DES ÉQUIPES DE DÉVELOPPEMENT APPLICATIF. C'EST IMPORTANT DE SAVOIR D'OÙ L'ON VIENT.

À partir des années 90, avec l'apparition d'Internet, le développement informatique encore réservé à certains usages comme la recherche, a tout simplement explosé. Le nombre de sites web a fortement augmenté. Même chose pour le nombre de professionnels travaillant dans l'informatique, qu'il a fallu répartir selon leurs compétences :

— **Les développeurs** créent le code de l'application puis en assurent le « Build » : ils utilisent un outil de compilation pour transformer le code en un langage compréhensible par les machines. Ils transmettent ensuite le projet aux équipes opérationnelles.

— **Les équipes opérationnelles** sont chargées de déployer l'application sur les environnements de production et d'en assurer le « Run » : la gestion quotidienne des applications et des infrastructures.



ÉQUIPE DEV

Développeur

Développe le code de l'application et le test.

Objectif :

Livrer rapidement des innovations.

Contraintes :

- La vélocité des Devs est limitée par le travail des Ops qui ont des procédures à respecter pour livrer en production.
- L'équipe des Devs doit fournir aux Ops, un processus de déploiement du code sans avoir accès aux environnements de production.

≠

ÉQUIPE OPS

Équipe de production

Met en place l'application en production puis gère son exploitation.

Objectif :

Assurer la stabilité du service.

Contraintes :

- L'équipe Dev fournit du code qui ne peut être pas stable en production.
- Les Ops ne connaissent pas toujours les applications qu'ils doivent déployer.

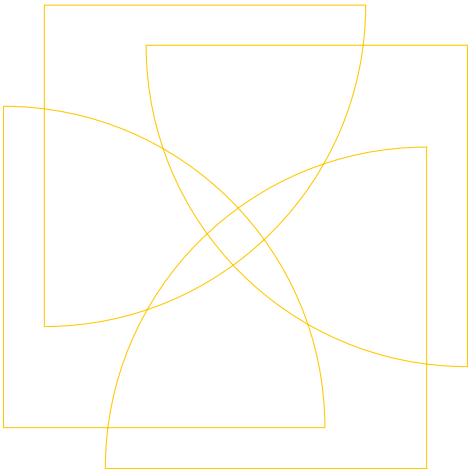
Organisation silotée : les Devs d'un côté, les Ops de l'autre

Ces deux équipes ont des objectifs et des contraintes différentes, voire contradictoires. Les développeurs doivent délivrer rapidement des fonctionnalités innovantes afin de suivre les nouvelles tendances et les nouveaux besoins. Les équipes opérationnelles doivent assurer la stabilité du système. Les « Ops » sont donc tenus de s'assurer de la qualité du code avant de le déployer en production, en s'appuyant sur des processus qui peuvent paraître plus lents et plus lourds que ceux des développeurs.

> LES ÉQUIPES DEV ET OPS AYANT ÉTÉ HISTORIQUEMENT CLOISONNÉES, IL EST COURANT QUE CES CONTRADICTIONS GÉNÈRENT DES FRUSTRATIONS ET DES TENSIONS ENTRE ELLES, LES AMENANT PETIT À PETIT À ROMPRE TOTALEMENT LEURS ÉCHANGES.

Cela donne lieu à ce qu'on appelle « mur de confusion », mur symbolique qui représente les difficultés de communication entre les équipes.

Pour résoudre ce problème de fond, Patrick Debois, consultant, chef de projet et praticien agile, a baptisé « DevOps » une philosophie de développement informatique qui fait disparaître la séparation entre les développeurs et les équipes opérationnelles.



LES CLEFS DU DEVOPS

01 UNE COLLABORATION PERMANENTE

Le « DevOps », ainsi baptisé depuis 2007, tend à réconcilier les Devs et les Ops en favorisant la collaboration des équipes, la communication et l'automatisation.
En termes de livrables, le DevOps fluidifie au maximum le passage d'une application (ou d'une fonctionnalité) de la phase de développement à la phase de livraison.

En opposition avec la ligne droite d'un projet classique qui a un début et une fin, cette philosophie est symbolisée par le signe de l'infini, qui représente la symbiose entre les Devs et les Ops, améliorant ensemble leur produit en permanence.

01 - Code

Ecriture des **lignes de code** (dans le langage choisi) qui permettront de construire l'application.

02 - Buid

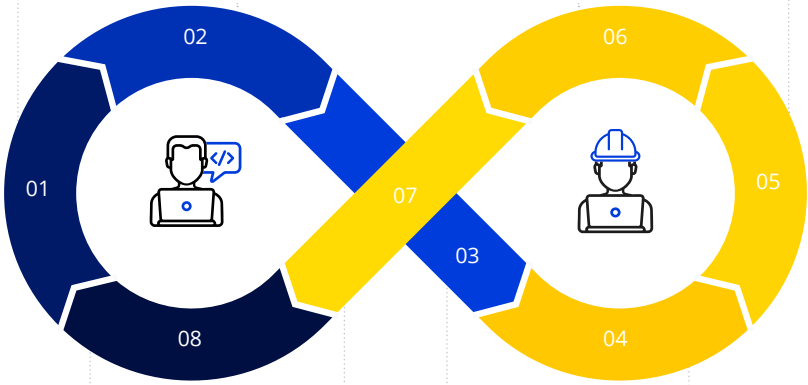
Transformation du code écrit par les développeurs en un **langage compréhensible par une machine** (en utilisant Maven par exemple).

06 - Operate

Gestion du Run, les équipes veillent au bon fonctionnement de l'application en production.

05 - Deploy

Déploiement de l'application sur les serveurs et sa mise à disposition pour les utilisateurs.



08 - Plan

Récupération des besoins des utilisateurs et **planification** des phases de développement.

07 - Monitor

Surveillance de l'application en production et récupération de données.

03 - Test

Test des fonctionnalités développées.

04 - Release

Dernière phase de test permettant de valider ou non le déploiement du code.

DevOps, l'amélioration continue

Si vous souhaitez en apprendre davantage sur le DevOps, je vous encourage à lire notre livre blanc dédié : **"Comprendre le DevOps pour le mettre en œuvre"**, par **Mehdi EL KOUHEN**.

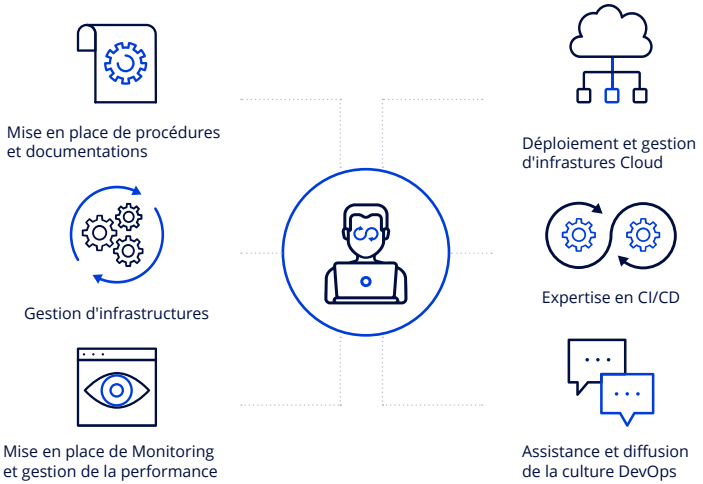
02 LES INGÉNIEURS DEVOPS, DES FACILITATEURS

SI LE DEVOPS EST UNE PHILOSOPHIE, POURQUOI AURAIT-ON BESOIN DE « PROFILS DEVOPS » OU « D'INGÉNIEURS DEVOPS » ?

Tout simplement pour faciliter la transition des équipes de développement, qui doivent se familiariser à la fois avec de nouveaux outils qui évoluent très rapidement, et des concepts hors de leur champ d'expertise initial.

Certains vous diront qu'il ne peut exister de « DevOps » en tant que personne, parce que cela signifierait que la démarche n'est portée que par cet individu, et non par toutes les équipes. Cependant, il ne me semble pas utile d'être aussi radical : le fait est que des profils DevOps sont apparus pour aider les organisations à adopter la philosophie DevOps !

Dans un monde idéal, ils devraient disparaître progressivement, une fois le DevOps complètement adopté par l'entreprise. Mais dans les faits, cet objectif n'est que très rarement atteint.



L'expert DevOps, un profil polyvalent

Les compétences de l'ingénieur DevOps couvrent une large gamme qui va de la maîtrise d'outils de CI/CD (Jenkins, GitLab CI/CD, GitHub Actions…) ou de surveillance des applications et des infrastructures, à la mise en place et à la gestion d'infrastructures (Kubernetes, AWS, Azure…).

Il s'appuie sur cinq grands principes qui forment l'acronyme CALMS : Culture, Automation, Lean, Measure, Sharing.

C	A	L	M	S
Culture	**Automation**	**Lean**	**Measure**	**Sharing**
Adopter la culture du DevOps. Se concentrer sur les personnes et pas seulement sur la technique. Accepter les évolutions et les changements.	Automatiser le plus de tâches possible pour ne se concentrer que sur celles qui apportent de la plus value aux projets.	Éliminer les tâches qui n'apportent pas de valeur. S'améliorer en continu, tester les choses, accepter l'échec.	Mettre en place des métriques qui permettent de mesurer son évolution et sa progression.	Adopter une culture de collaboration, échanger sur les avancées, partager les problèmes.

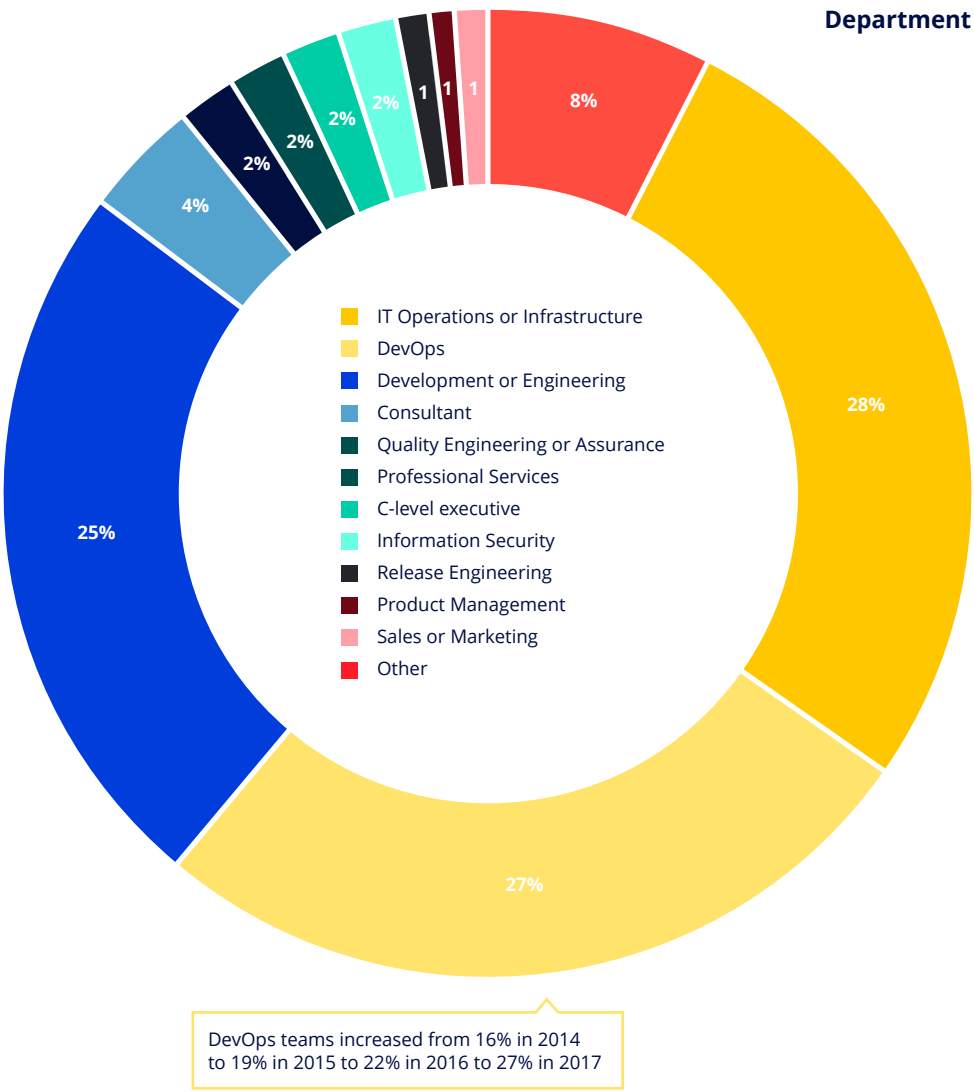
Les cinq piliers du DevOps

03 — DES BÉNÉFICES MESURABLES

EN ADOPTANT LA PHILOSOPHIE DEVOPS, LES ANCIENS « DEVS » ET « OPS » ONT RÉSOLU LEURS PROBLÈMES DE COMMUNICATION ET PROGRESSÉ SUR QUATRE MÉTRIQUES PRINCIPALES, IDENTIFIÉES PAR L'**INSTITUT DORA** *(DEVOPS RESEARCH AND ASSESSMENT).*

Les deux premières portent sur la vélocité, les deux secondes sur la stabilité :

- **La fréquence de déploiement** *(Deployment Frequency)* : c'est le rythme auquel une équipe est capable de livrer de nouvelles fonctionnalités.
- **Le délai de modification** *(Lead Time For Changes - LTFC)* : c'est la durée moyenne nécessaire pour qu'une fonctionnalité passe en production, à partir du moment où elle a fini d'être développée.
- **Le taux d'échec au déploiement** *(Change Failure Rate - CFR)* : c'est le taux de changement, sur du code applicatif ou même sur des modifications d'infrastructure, qui aboutit à des instabilités. Il est exprimé en ratio du nombre de déploiements présentant des défauts sur le nombre de déploiements au total.
- **Le temps moyen de récupération** *(Mean Time To Recovery - MTTR)* : c'est le temps moyen nécessaire pour remettre en état le service après l'apparition d'un défaut (bug, panne partielle ou totale).



Niveau d'adoption du DevOps, métier par métier, en 2017
(State of DevOps Report, by Puppet and DORA)

Le ***State Of DevOps report*** est un rapport annuel qui mesure l'adoption de la philosophie DevOps dans les entreprises et en chiffre les bénéfices en termes de productivité. Il s'appuie sur un sondage des entreprises et acteurs de l'IT.
Publié par l'entreprise Puppet chaque année, en collaboration avec DORA entre 2013 et 2017, il a montré depuis 2013 le lien de causalité directe entre l'adoption de la philosophie DevOps et la progression sur chacune de ces quatre métriques.

Depuis 2018, après son rachat par Google, l'institut DORA publie son propre rapport, intitulé ***Accelerate State of DevOps Report***. Je vous invite à suivre les publications de mon collègue **Jean-Rémy REVY**, Architecte Solution et Practice Leader chez Ippon Technologies, qui présente chaque année les nouveautés de ce rapport dans sa conférence « ***Accélérer avec accelerate.*** »

Ce nouveau rapport fondé lui aussi sur les réponses de milliers de professionnels permet de suivre les métriques listées en page 14 et de les classer en quatre niveaux de performances : Élite, Haut, Moyen, Faible.

PERFORMANCE LEVEL	CHANGE LEAD TIME	DEPLOYMENT FREQUENCY	CHANGE FAIL RATE	FAILES DEPLOYMENT RECOVERY TIME	PERCENTAGE OF REPONDENTS*
Elite	Less than one day	On demand (multiple deploys per day)	5%	Less than one hour	19% (18-20%)
Hight	Between one day and one week	Between once per day and once per week	20%	Less than one day	22% (21-23%)
Medium	Between one week and one month	Between once per week and once per month	10%	Less than one day	35% (33-36%)
Low	Between one month and six months	Between once per month and once every six months	40%	Between one week and one month	25% (23-26%)

* 84% uncertainly interval

2024 Accelerate State of DevOps, by DORA **: Une échelle pour vous situer par rapport au marché**

04 PAS DE FORMULE MAGIQUE

Le DevOps a permis à certaines entreprises de progresser, mais d'autres n'ont pas réussi à le faire adopter par leur équipe, ou n'en ont pas tiré les avantages qu'elles espéraient.

> LE DEVOPS EST UNE PHILOSOPHIE, CE N'EST PAS UNE NORME : CHACUN EST DONC LIBRE DE S'EN EMPARER COMME BON LUI SEMBLE, QUITTE À FAIRE TOUS LES MAUVAIS CHOIX !

CINQ ERREURS FRÉQUENTES

Dans le livre **Team Topologies: Organizing Business and Technology Teams for Fast Flow (2019)**, Manuel Païs et Matthew Skelton, deux consultants en informatique, décrivent justement les « anti-patterns » : les erreurs fréquentes d'implémentation du DevOps. En voici quelques exemples :

DevOps : les principaux pièges à éviter.

Erreur n°1 : L'équipe DevOps forme un nouveau silo

Le DevOps est né pour désiloter les équipes de développement et les équipes opérationnelles. Pour faciliter son adoption, les entreprises peuvent être tentées de construire une équipe DevOps dédiée, qui aide les Devs et les Ops à mieux travailler ensemble. Mais si cette nouvelle équipe perdure, on peut se retrouver dans une situation dans laquelle on a créé un troisième silo, qui défend ses propres intérêts et qui sépare encore plus les deux autres.

Erreur n°2 : On considère que les Ops ne sont plus nécessaires

Il arrive parfois que les développeurs considèrent qu'ils peuvent se passer des Ops, car ils disposent désormais de profils DevOps, qui peuvent mettre en place des infrastructures pour leurs applications. Mais ce modèle ne tient pas lorsque les applications prennent de l'ampleur : les équipes de développement se retrouvent à passer plus de temps sur la gestion des infrastructures qu'à développer.

Erreur n°3 : Une seule équipe resserrée

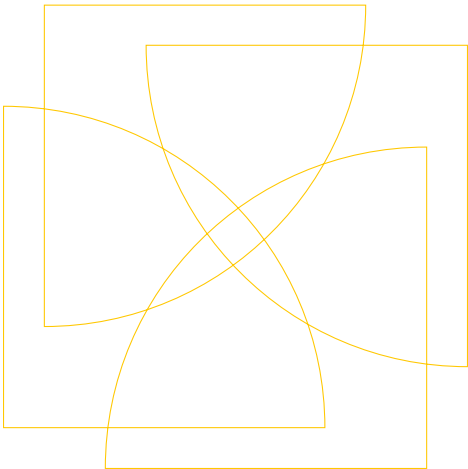
Si l'on pousse l'exercice à l'extrême, on peut avoir une équipe DevOps chargée de tout le cycle de vie des applications. Avec l'arrivée du Cloud, les décideurs peuvent penser qu'il n'y a plus lieu d'avoir une équipe d'Ops, car comme les équipes DevOps ont gagné en autonomie, elles seront en mesure de gérer à la fois leurs applications et l'infrastructure sous-jacente. On se retrouve alors avec des équipes de développement ayant une charge de travail très élevée et une pression immense.

Erreur n°4 : DevOps est le nouveau SysAdmin

Certaines entreprises ne voient dans la philosophie DevOps qu'un moyen d'automatiser des tâches sans prendre en compte l'aspect humain et plus spécifiquement le liant qu'on peut créer entre les équipes. Cela peut mener à un cas de figure dans lequel les DevOps sont simplement des SysAdmin, à qui on donne une casquette de DevOps pour surfer sur la tendance.

Erreur n°5 : DevOps est le nouvel Ops

Cet anti-pattern est proche du précédent, mais dans ce cas-là, les entreprises ne prennent pas la peine de faire appel à des SysAdmin et renomment simplement leurs Ops en DevOps, sans avoir forgé de compétences spécifiques et sans avoir adopté la philosophie DevOps.
Vous pouvez retrouver d'autres erreurs à éviter sur le site **DevOps Topologies**.



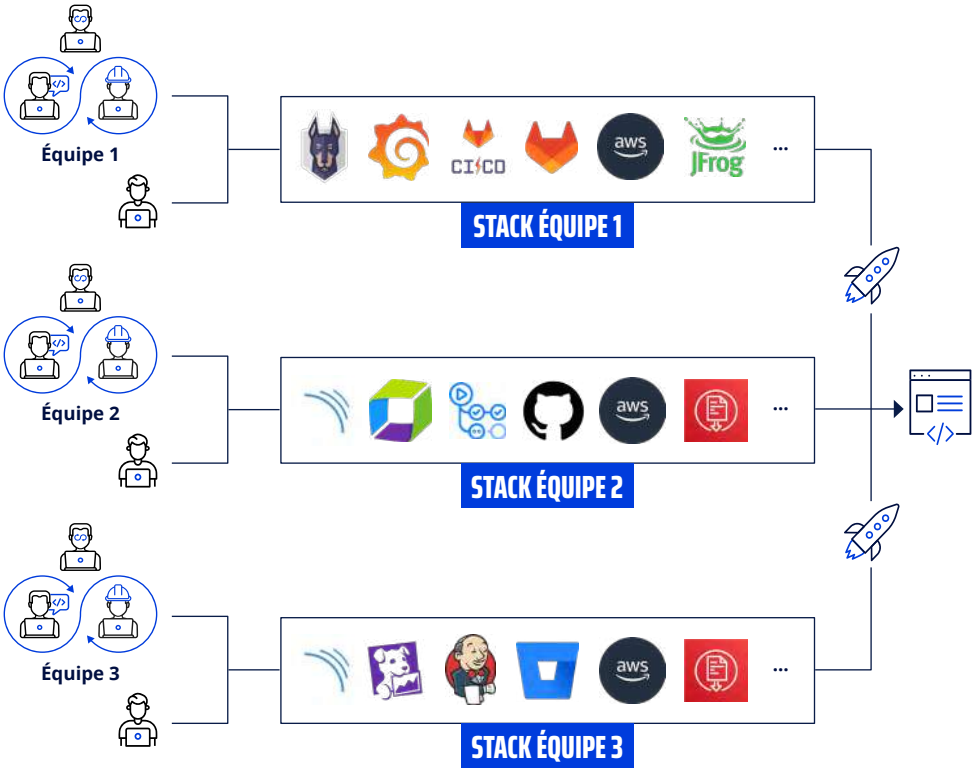
05 DES RESPONSABILITÉS TROP LARGES

Les équipes DevOps ont fortement progressé en efficacité et ont dû gérer le cycle de vie complet de leurs applications. Elles se sont donc retrouvées avec un champ de responsabilités beaucoup plus étendu qu'avant :

Trop de tâches à accomplir éloignent les équipes DevOps de leur cœur de mission.

Mettre en place une chaîne CI/CD, « conteneuriser » ses applications, construire une infrastructure Cloud, mettre en place une surveillance des applications… Les tâches se multiplient et les outils aussi. Je rappelle qu'une équipe DevOps, idéalement, est à la fois composée de développeurs et d'Ops, permettant d'avoir toutes les compétences nécessaires pour répondre à ces nouvelles responsabilités. Pourtant, c'est rarement le cas et les équipes de développement se retrouvent souvent avec un champ de responsabilités trop large, leur demandant beaucoup de temps et d'énergie, les empêchant de se focaliser sur un sujet en particulier et les amenant à délaisser certains aspects importants comme la sécurité ou l'optimisation.

Les équipes DevOps ont atteint leur limite lorsqu'il a fallu passer ce modèle à l'échelle des entreprises. En effet, ces équipes étant souvent autonomes dans leur approche et leurs choix technologiques, il est difficile de garder une cohésion dans les outils, dans les pratiques utilisées et surtout de se doter d'une vraie gouvernance.



Dans la même entreprise, trois équipes DevOps, ce sont souvent trois stacks techniques !

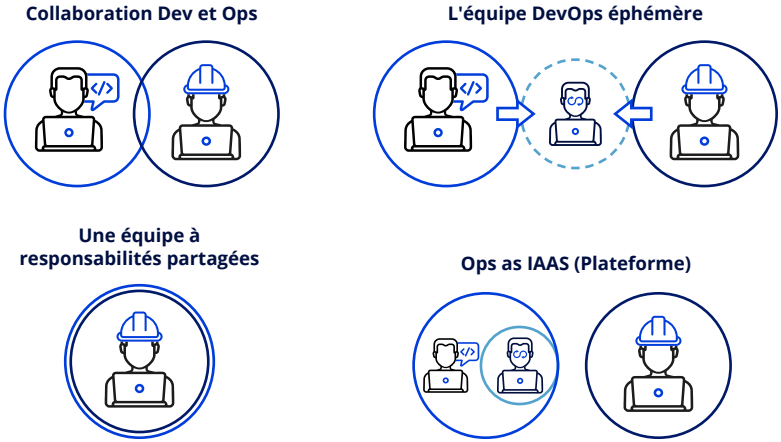
> MULTIPLIER LES ÉQUIPES DEVOPS, C'EST S'EXPOSER À UN DÉSÉQUILIBRE DE PERFORMANCE ENTRE ELLES ET À L'APPARITION DE SILOS TECHNOLOGIQUES.

Multiplier les équipes DevOps, c'est s'exposer à une fragmentation des pratiques et des compétences qui peut se manifester de plusieurs façons :

— Un déséquilibre de performance entre les équipes, certaines étant plus matures que d'autres dans leurs pratiques et leur expertise

— Des silos technologiques qui se créent naturellement quand chaque équipe fait ses propres choix d'outils et de technologies, rendant difficiles le partage d'expérience et la collaboration inter-équipes

Après avoir montré les anti-patterns du DevOps, les auteurs du livre **Team Topologies** décrivent des organisations d'équipes et des modes d'interaction qui fonctionnent :



Quatre patterns DevOps efficaces.

La collaboration Dev et Ops

C'est le « Graal » : une topologie dans laquelle les Devs et les Ops travaillent main dans la main, avec des échanges fréquents, même en se trouvant dans des équipes séparées. Cela nécessite que les Devs comprennent ce que font les Ops, et inversement. Tous partagent le même objectif : livrer fréquemment et de façon fiable en production.

L'équipe DevOps éphémère

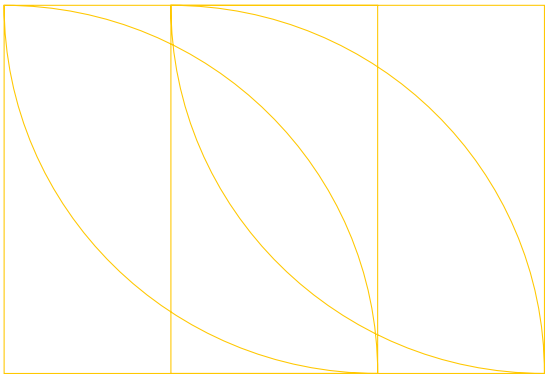
Une équipe DevOps éphémère met en place les moyens nécessaires pour amener les Devs et les Ops à collaborer. Si elle y parvient, cela signifie qu'elle peut… disparaître ! Mission accomplie.

Une équipe à responsabilités partagées

Ici, les Devs et les Ops sont si proches qu'on ne les distingue pas et qu'ils sont concentrés sur les mêmes objectifs. Cela peut s'appliquer seulement si le nombre d'applications à gérer est limité et si tous les efforts de l'entreprise y sont consacrés, comme chez Netflix.

Les Ops comme équipe-plateforme

Les Ops mettent en place l'infrastructure et le socle technique permettant aux équipes de Devs de devenir DevOps. Ainsi, ces derniers sont capables de gérer l'ensemble du cycle de vie de leurs applications, sans avoir à mettre en place d'outillage.



06 LA RÉPARTITION DES RÔLES

LES ÉQUIPES DEVOPS ONT PERMIS DE FAIRE DES PROGRÈS MAJEURS EN TERMES DE RÉACTIVITÉ PAR RAPPORT AUX DEMANDES DES UTILISATEURS FINAUX, MAIS LES ENTREPRISES ONT SOUVENT SOUS-ESTIMÉ LA CHARGE DE TRAVAIL ET LA DIFFICULTÉ DE LA TÂCHE.

La philosophie DevOps a fait ses preuves, mais elle a provoqué aussi des problèmes d'organisation au sein des entreprises qui ont adopté de mauvaises pratiques (les « anti-patterns » évoqués plus haut)

Il a donc fallu repenser l'organisation des équipes pour leur permettre de mieux travailler ensemble, tout en continuant à adopter la philosophie DevOps.
Le livre **Team Topologies** présente une organisation des équipes informatiques qui permet d'augmenter de 20 % la productivité des équipes, de réduire de 40 % les échecs de déploiement et de diminuer le Mean Time To Recovery (MTTR) de 35 %.

Cette organisation s'articule autour de quatre équipes :

STREAM-ALIGNED TEAMS	PLATFORM TEAMS
Les équipes « Stream-aligned » sont concentrées sur un objectif unique, par exemple le développement d'une application. Ce sont elles qui portent la valeur métier. Ici, les équipes de développement sont affectées à 100% aux nouvelles fonctionnalités. Mais pour rester DevOps, elles doivent continuer à utiliser des outils d'automatisation, sans pour autant en avoir la charge.	Les équipes « Platform » mettent en place les bases permettant aux équipes « Stream-aligned » de se concentrer sur leurs tâches. Elles prennent en charge tout ce qui n'apporte pas de plus-value au produit fourni par l'entreprise, notamment au niveau de la mise en place d'outils d'automatisation.
COMPLICATED-SUBSYSTEM TEAMS	**ENABLING TEAMS**
Les équipes sous-systèmes complexes sont là pour apporter une aide ponctuelle aux équipes « Stream-aligned » lorsque celles-ci n'ont pas les compétences nécessaires pour répondre à certains besoins.	Les équipes « Enabling » accompagnent les équipes « Stream-aligned » dans leur montée en compétences sur des sujets qui dépassent le cadre du développement.

07

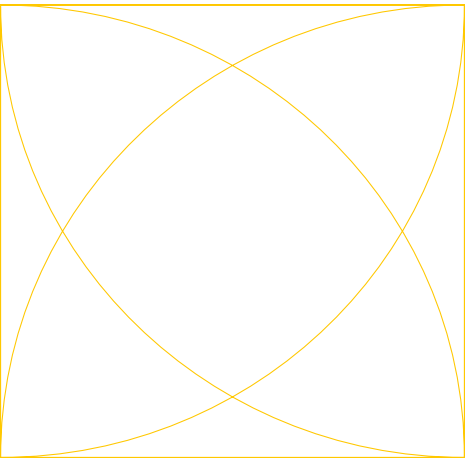
REPENSER LES INTERACTIONS ENTRE LES ÉQUIPES

EN DEHORS DE L'ORGANISATION DES DIFFÉRENTES ÉQUIPES AU SEIN D'UNE ENTREPRISE TECH, CE SONT LEURS INTERACTIONS ENTRE ELLES QU'IL FAUT REPENSER.

Dans l'ouvrage **Team Topologies**, trois modes d'interaction sont décrits :

- La **collaboration** : les équipes travaillent sur un même objectif.
- La **coordination** : une équipe soutient une ou plusieurs autres équipes pour les aider à remplir leur objectif.
- **X-as-a-service** : une équipe met en place un service qui permet aux autres de remplir leurs objectifs.

Attention, ces trois types d'interactions doivent rester les seuls !
Quelle est la conclusion concrète de cette approche organisationnelle ? En 2020, le State Of DevOps Report montrait que les entreprises qui avaient le plus progressé les années précédentes étaient celles qui ont mis en place des Platform Teams et faisaient du Platform Engineering.



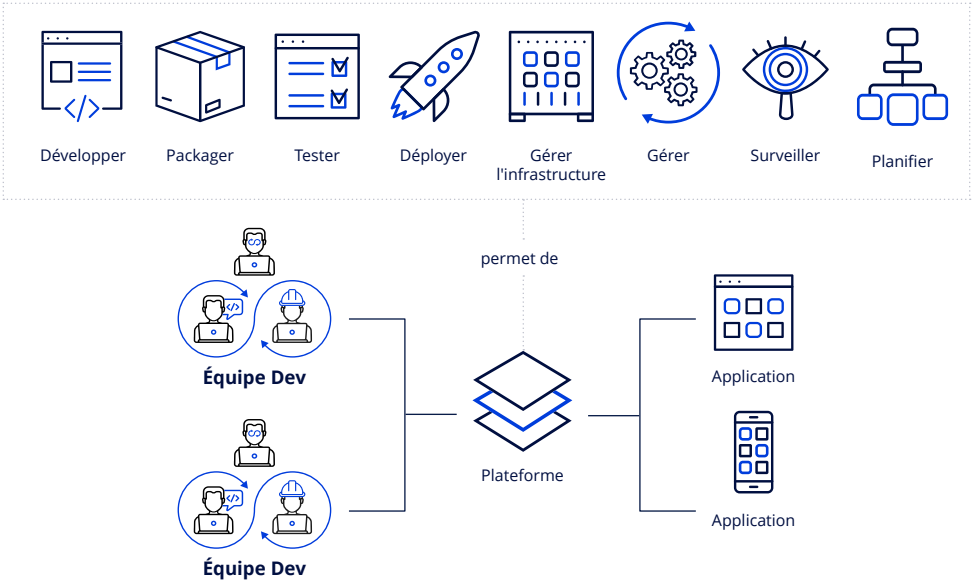
L'ÈRE DU PLATFORM ENGINEERING

01 UNE DÉFINITION SIMPLE

LE **« PLATFORM ENGINEERING »** OU INGÉNIERIE DE PLATEFORME, EST UN MOUVEMENT QUI VISE À IMPLÉMENTER LA PHILOSOPHIE DEVOPS EN METTANT À DISPOSITION DES ÉQUIPES DE DÉVELOPPEMENT UNE « PLATEFORME » PERMETTANT DE GÉRER L'ENSEMBLE DU CYCLE DE VIE D'UNE APPLICATION.

Cette plateforme peut être simplement un ensemble de ressources, mais elle peut aussi prendre la forme d'une application : la **Plateforme Interne de Développement (IDP)**.

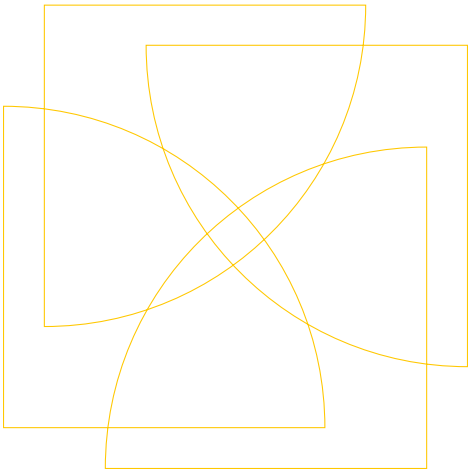
L'IDP comprend un ensemble d'outils (pipeline CI/CD, Terraform, Helm, ArgoCD...) et de services pré-configurés permettant aux développeurs de mettre en place et de gérer eux-mêmes leurs environnements, en faisant **abstraction de toute la difficulté qu'il y a derrière** : déploiement d'infrastructure, paramétrage de pipeline CI/CD, construction d'images Docker, mise en place d'outils de surveillance... Il leur devient donc plus facile de respecter le fameux principe « You build it, you run it » tiré d'une intervention de Werner Vogels, CTO d'Amazon, en 2006.

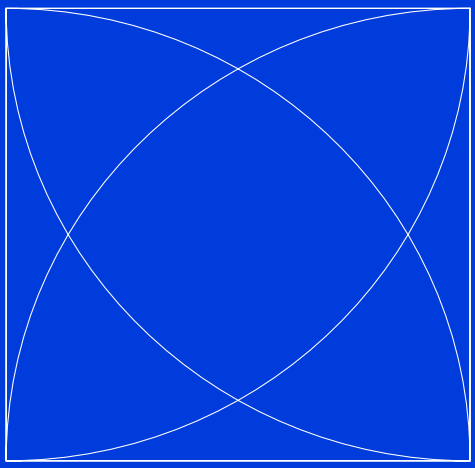


Le Platform Engineering, c'est aussi plus d'autonomie pour les équipes de développement.

> PLATFORM ENGINEERING CAN ACT AS A BARRIER AGAINST THE CHAOS OF TOOLS, TASKS, AND INFORMATION. BY STANDARDIZING TOOLS AND PROCESSES, IT CAN LIBERATE DEVELOPERS FROM THE BURDEN OF BECOMING TOOL EXPERTS SO THAT THEY CAN FOCUS ON THEIR CORE STRENGTHS: WRITING GREAT CODE AND MAKING EXCEPTIONAL PRODUCTS."
>
> *2024 Puppet State of DevOps Report: The Evolution of Platform Engineering*

Nous le verrons par la suite, mais le Platform Engineering peut être décliné dans d'autres domaines que le développement, la Data notamment (lire page 56).





LE POINT DE VUE DE NOS EXPERTS

« Selon vous, qu'est-ce que le Platform Engineering, et comment diffère-t-il d'autres approches comme DevOps ou SRE (*Site Reliability Engineering*) ? Quels sont ses principaux bénéfices pour les entreprises modernes ? »

TIMOTHÉE AUFORT

Practice Leader de la Practice Cloud & DevOps chez Ippon Technologies

« DEVOPS, SRE ET PLATFORM ENGINEERING : TROIS APPROCHES QUI TENDENT VERS UN BUT COMMUN »

Le Platform Engineering (PE) s'appuie fortement sur les approches DevOps ou SRE et n'existerait pas sans elles. Il ne faut absolument pas les opposer, mais plutôt les associer. La méthodologie DevOps tend à rapprocher les Devs (profils développeurs) et les Ops (profils opérationnels), elle permet de fournir de l'automatisation dans tout le cycle de vie d'un logiciel : développements, tests, déploiement et maintenance des infrastructures. L'approche SRE tend à rendre un logiciel plus évolutif et fiable dans le but d'assurer sa disponibilité une fois en production. Le Platform Engineering utilise les approches DevOps/SRE pour les passer à l'échelle.

Il permet de fournir à des équipes de développement une plateforme évolutive pour les aider à se concentrer sur leur produit, plutôt que sur les outils qui gravitent autour. Les trois approches tendent vers un but commun, auquel de nombreux ingénieurs font souvent référence : le principe du « You build it, you run it » énoncé par Werner Vogels, CTO d'Amazon.

LE PLATFORM ENGINEERING S'ADAPTE PARTICULIÈREMENT BIEN À UNE ENTREPRISE DE TAILLE MOYENNE OU GRANDE, CAR IL VA STANDARDISER LES OUTILS ET EXPOSER DES SERVICES COMMUNS À TOUTES LES ÉQUIPES PRODUITS, COMME UN SERVICE DE VERSIONING DU CODE, UN GESTIONNAIRE D'ARTEFACTS OU ENCORE UN GESTIONNAIRE DE SECRETS.

Par ailleurs, le Platform Engineering accélère grandement la mise en place des fondations d'un nouveau projet logiciel, permettant aux équipes produits de se concentrer sur la valeur métier pour leurs utilisateurs finaux.

VIVEN MALEZE

Architecte Technique chez Ippon Technologies

« AMÉLIORER L'EXPÉRIENCE DES DÉVELOPPEURS »

Pour moi, le Platform Engineering consiste à fournir aux équipes de développement les outils et solutions adaptés pour qu'elles puissent se concentrer uniquement sur leur travail de développement, sans avoir à gérer d'autres aspects techniques complexes. Contrairement au DevOps, qui s'occupe principalement de l'installation, de la gestion et du maintien des outils, le Platform Engineer se concentre sur l'utilisation des outils et leur configuration, pour répondre aux besoins spécifiques des utilisateurs. Un exemple concret : là où une équipe DevOps va déployer et maintenir un outil comme GitLab, un Platform Engineer va développer et adapter les pipelines CI/CD pour les équipes de développement.

Quant au Site Reliability Engineering (SRE), il intervient à un stade plus avancé, en s'assurant que les services déployés fonctionnent correctement. Son rôle est axé sur la disponibilité, la fiabilité et la performance des systèmes, souvent en collaboration avec les équipes DevOps ou de développement.
En termes de bénéfices pour les entreprises modernes,

LE PLATFORM ENGINEERING CRÉE UNE ÉQUIPE DÉDIÉE À L'AMÉLIORATION DE L'EXPÉRIENCE DES DÉVELOPPEURS.

Ces derniers peuvent ainsi se concentrer sur leur véritable valeur ajoutée, à savoir le développement de fonctionnalités, tandis que l'ensemble du système gagne en efficacité et en robustesse.

02 PLATFORM ENGINEERING : VOS PREMIERS PAS

TROIS DIFFICULTÉS À ANTICIPER

SI VOUS SOUHAITEZ METTRE EN PLACE UNE DÉMARCHE DE PLATFORM ENGINEERING AU SEIN DE VOTRE ENTREPRISE, VOICI QUELQUES FREINS QUE VOUS ALLEZ RENCONTRER.

Nous aborderons ensuite les piliers du Platform Engineering, qui permettent de surmonter ces difficultés.

Convaincre les décideurs

La plateforme n'est pas un élément qui apporte directement de la valeur aux produits finaux que sont les applications. La première difficulté qui me vient à l'esprit est donc celle de convaincre les décideurs d'investir du temps et des moyens pour une migration vers le Platform Engineering.

Obtenir l'adhésion des développeurs

La plateforme peut être perçue comme une contrainte qui vient s'ajouter au travail des développeurs : ils sont parfois réticents à changer leurs habitudes.

Gérer correctement un existant ultra dense

L'hétérogénéité des technologies utilisées par les équipes de développement est une grande difficulté dans la mise en place de la plateforme, et notamment la gestion du Legacy ou héritage technique.

DES PRINCIPES PORTEURS DE SOLUTIONS

Un des problèmes avec DevOps, on l'a vu, venait du fait qu'étant une philosophie, elle pouvait être interprétée librement et n'apportait pas de solution toute faite sur les outils. Lucas Galante, chez Humanitec, recense sur le blog platformengineering.org les piliers du Platform Engineering, qui permettent d'éviter ces zones de flou.

Se doter d'une mission claire et d'un rôle parfaitement défini

Les anti-patterns du DevOps sont apparus car le rôle des équipes DevOps a parfois été mal compris. Pour ne pas reproduire ce travers,

> L'ÉQUIPE PLATEFORME DOIT AVOIR UN RÔLE CLAIR DANS L'ESPRIT DES AUTRES ÉQUIPES : ELLE N'EST PAS UNE ÉQUIPE SUPPORT, ELLE DÉLIVRE UN PRODUIT À DESTINATION DES ÉQUIPES DE DÉVELOPPEMENT.

Considérer la plateforme comme un produit

Il faut absolument garder en tête que les utilisateurs de la plateforme sont les développeurs eux-mêmes (ou des profils Data dans le cas d'une plateforme Data) :

COMME POUR TOUT PRODUIT, IL FAUT D'ABORD TRAVAILLER SUR L'EXPÉRIENCE UTILISATEUR.

L'équipe plateforme doit adopter une démarche « produit » pour répondre aux besoins de ses utilisateurs, en prenant en compte attentivement leurs retours.

Rester concentré sur les problématiques communes

L'objectif de l'équipe plateforme est de capitaliser au maximum sur les besoins du groupe :

IL VAUT DONC MIEUX RÉPONDRE AUX BESOINS LES PLUS SOUVENT RENCONTRÉS PAR LES ÉQUIPES DE DÉVELOPPEMENT, QU'À DES DEMANDES SPÉCIFIQUES.

Bien comprendre la valeur de la plateforme

Ce point est principalement destiné aux décideurs que je mentionnais plus haut et qui peuvent sous-estimer l'intérêt d'une équipe plateforme et la considérer comme une dépense superflue. La plateforme est « *le* » produit qui permet aux développeurs de disposer d'une chaîne d'outils performants et d'assurer un flux de développement fluide.

Ne pas réinventer la roue

Les équipes DevOps se perdaient parfois parmi la myriade d'outils à leur disposition. L'équipe plateforme peut être tentée à son tour d'utiliser de nouveaux outils pour répondre à un même besoin : parce qu'ils sont plus performants… ou simplement par envie de les tester ! Mais changer d'outil représente toujours un effort et il faut systématiquement peser soigneusement le pour et le contre.

En dehors de ces piliers « *officiels* », nous avons extrait d'autres bonnes pratiques de nos expériences cumulées chez Ippon Technologies :

Former les développeurs

La plateforme répond à leurs besoins, mais il peut être frustrant de ne pas comprendre ce qu'elle fait exactement. Commencer par les former sur son utilisation et expliquer son fonctionnement interne peut être un moyen de les embarquer plus facilement.

Communiquer sur les évolutions de la plateforme

La plateforme ne peut pas répondre aux besoins de tous dès le départ, elle doit vivre et intégrer de nouvelles fonctionnalités au fur et à mesure, en fonction des retours des développeurs. Il est essentiel de communiquer sur ces évolutions pour montrer aux utilisateurs que leurs besoins sont pris en compte et que la plateforme est à leur service.

Penser « Services » plutôt « qu'Outils »

La force du Platform Engineering n'est pas tant de mettre des outils à disposition des développeurs que des services.

TYPIQUEMENT, ON NE CHERCHE PAS À LEUR FOURNIR UN BUCKET S3 PRÉCONFIGURÉ (UN SERVICE DE STOCKAGE D'OBJETS SUR AWS), MAIS UN SERVICE DE STOCKAGE TOUT COURT, PEU IMPORTE LES OUTILS QUI SONT DERRIÈRE. CELA CONSTITUE, DE FAIT, UN PAS IMPORTANT VERS LA DIGITAL FACTORY
(LIRE PAGE 63).

ENVISAGER LE PASSAGE VERS LE PLATFORM ENGINEERING COMME UNE MIGRATION

Lorsque l'on parle de « migration » dans l'IT, on évoque un projet **avec des moyens humains et financiers dédiés**, qui permettent de passer des applicatifs d'un système à un autre. C'est une opération qui est inscrite au planning de l'entreprise et qui mobilise de nombreuses équipes différentes. Il doit en aller de même pour le Platform Engineering.

CAS D'USAGE

JULIE KOCIANOVA

est Product Owner chez Ippon Technologies. Elle revient sur la mise en place d'une plateforme Data chez un client. La migration a eu lieu en 2024.
Le MVP de la plateforme a été développé entre les mois de février et août, pour une adoption en trois temps (trois équipes).

La première équipe à adopter la nouvelle plateforme Data l'a fait entre mi-août et fin septembre 2024, la deuxième équipe entre mi-novembre 2024 et fin janvier 2025. Quant à la dernière équipe, elle a commencé à migrer mi-février 2025 et devrait finir mi-avril 2025.

Julie, tu es Product Owner de cette plateforme Data, peux-tu revenir sur sa mise en place, sans omettre les difficultés rencontrées ?

J'ai une formation d'ingénieur généraliste, mais cela fait quelques années que je construis plutôt des produits B-to-B ou B-to-B-to-C.
Il faut bien comprendre qu'une plateforme est un produit tech pour des techs ! Elle doit simplifier la vie des développeurs qui l'utilisent.
Les développeurs sont des utilisateurs de mon produit, pas des techs à qui je fournis des outils. En créant une plateforme, on crée donc bel et bien un produit composé de services qu'on peut voir comme des features ou même comme des mini-produits.

De ce fait, il ne faut pas oublier de faire émerger le « vrai » besoin métier, parfois dissimulé sous une demande très détaillée de la part des utilisateurs. Par exemple, on m'a récemment demandé d'ajouter une fonctionnalité sur la plateforme afin de permettre le partage de données vers un bucket S3, le service de stockage d'objets sur AWS. La requête était claire et précise, mais au final elle ne correspondait pas au besoin primaire de l'équipe, qui était de pouvoir s'interfacer facilement avec des logiciels tiers, dans un temps imparti.

De manière générale, la plateforme est en constante évolution en fonction des besoins de nos utilisateurs. Et réciproquement, l'équipe Data est en complète réorganisation pour tirer le meilleur parti de la plateforme avec un budget serré !

Quelles bonnes pratiques peux-tu nous partager pour réussir la mise en place de sa plateforme : moments-clés, outils, cérémonies, organisation… ?

La plateforme doit être visible et lancée officiellement au sein de l'organisation qui l'adopte. Elle n'est pas là pour faire entrer de l'argent directement - il faut le dire et l'expliquer ! Il s'agit plutôt d'optimisation des coûts et d'optimisation opérationnelle.
Mais il faut cependant faire la preuve de sa valeur le plus rapidement possible, pour modifier le regard de **l'entreprise,** qui **au départ verra la plateforme seulement comme un centre de coûts, alors qu'on peut en faire un centre de revenus.**

COMPOSEZ VOTRE ÉQUIPE PLATEFORME

Une fois que ces principes sont bien compris et que le rôle de l'équipe plateforme est clair dans la tête de chacun, l'équipe peut être montée. Même s'il n'y a pas de règles précises concernant sa composition, voici quelques profils qui me paraissent intéressants. Il est bon de les compter au sein de l'équipe. Leur nombre est à préciser en fonction de l'importance de la plateforme.

L'Ingénieur Plateforme

C'est le cœur de l'équipe plateforme, il est en quelque sorte le « nouveau DevOps », avec qui il a des compétences en commun. Il va mettre en place les composants de la plateforme et les templates, ainsi que les ressources réutilisables à destination des développeurs.

L'Ingénieur Data Platform

Si l'Ingénieur Plateforme maîtrise l'outillage DevOps, l'ingénieur Data, lui, maîtrise l'outillage DataOps. Il est nécessaire d'avoir ce type de profil lorsque l'on met en place une plateforme Data (lire page 56).

L'Architecte Plateforme

Comme dans toute équipe développant un produit, il peut y avoir un responsable technique pour construire une plateforme de développement. L'architecte plateforme est ce référent technique ; il travaille avec le Product Owner pour réaliser le design d'une plateforme qui réponde aux besoins des utilisateurs tout en respectant les bonnes pratiques du développement, du Cloud, de la Data, etc.

Le Product Owner

Le Product Owner est le garant de l'expérience utilisateur et du développement de la plateforme.
Il priorise les services à développer et, avec les membres de l'équipe, définit les objectifs de la plateforme qui se traduisent en Roadmap, en Epic (un ensemble de stories) et en User Stories (des exigences rédigées du point de vue d'un utilisateur). Tout cela alimente le backlog : l'ensemble de tâches à effectuer. Le Product Owner est également garant du succès de la plateforme auprès des stakeholders.
C'est un profil essentiel si l'on souhaite avoir une approche produit pour la plateforme… et pourtant il est souvent négligé.

Le Développeur Plateforme

Lorsque l'on souhaite mettre en place un portail de développement avec un outil existant comme Backstage par exemple, ou alors si l'on veut le développer entièrement, il me semble indispensable d'avoir un développeur au sein de l'équipe plateforme, car la plupart des outils disponibles sur le marché demandent de mettre les mains dans le code. Je cite Viktor de la chaîne Youtube DevOps Toolkit dans sa vidéo « Getting Started with Backstage: From Zero to Operational Dev Portal », à propos de la mise en place de Backstage comme portail développeur :

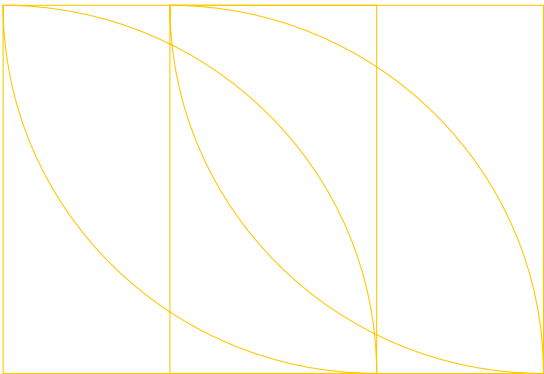
> BACKSTAGE IS A PROJECT THAT ASSUMES THAT THOSE USING IT ARE NODE NINJAS AND IF THEY ARE, THEY'RE SUPPOSED TO EXPERIENCE ONLY A MODERATE LEVEL OF PAIN. YOU'RE EITHER NOT SKILLED IN NODEJS AND TYPESCRIPT AND YOU WILL BE IN A LOT OF PAIN OR YOU ARE AND YOU WILL BE STILL IN MAIN ... BUT IN MORE MODUS QUANTITY. »
>
> *Viktor, DevOps Toolkit,*
> *Getting Started with Backstage*

Le message est clair : il faut des développeurs. Or, ce n'est pas toujours le cas.

L'agiliste

L'agiliste est le garant de la méthode agile au sein de l'équipe plateforme. Il peut s'agir d'un membre qui a une appétence particulière pour l'agilité, ou d'un profil dédié comme un Scrum Master ou un Coach Agile. Ce profil n'est pas toujours nécessaire : si tous les membres de l'équipe sont déjà à l'aise avec la méthode agile, il devient superflu.

Tous ces profils sont importants dans une équipe plateforme, mais cela ne veut pas dire qu'ils travaillent sans concertation. Au contraire !
À noter également : dans le cas où la plateforme déploie des composants sur du Cloud, il peut être utile d'avoir un Centre d'excellence Cloud (CCOE), constitué d'architectes et d'ingénieurs Cloud, pour mettre en place les bonnes pratiques. Une équipe d'experts en sécurité peut aussi présenter de l'intérêt. Ces deux équipes-là peuvent être dissociées de l'équipe plateforme, mais doivent collaborer étroitement avec elle.

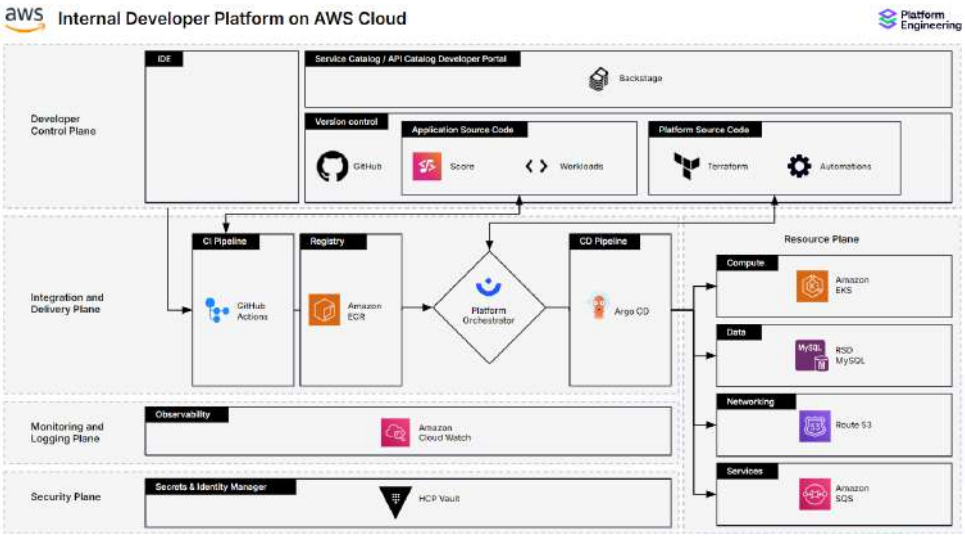


LES CINQ COUCHES D'UNE BONNE PLATEFORME

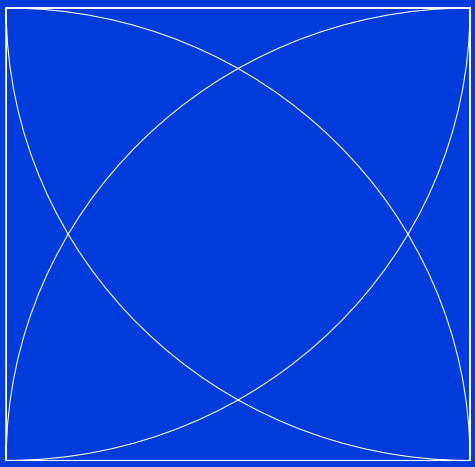
Le site **plateformengineering.org** présente des architectures de référence, pour aider les entreprises à monter leur plateforme de développement. Il liste cinq couches principales :

- **La couche développeur** : c'est celle avec laquelle les développeurs interagissent. On y retrouve les IDE (VsCode, IntelliJ), les gestionnaires de version de code (GitHub, GitLab, Bitbucket, etc.) et les outils d'infrastructure *as code* comme Terraform ou Puppet. Dans les plateformes les plus sophistiquées, on trouvera aussi un portail développeur pour faciliter les interactions des utilisateurs avec tous les outils sous-jacents, par exemple grâce à Backstage, Port, ou encore Cortex.

- **La couche d'intégration et de livraison** : c'est la couche dans laquelle les applications sont construites et déployées. On y retrouve les outils de CI/CD (GitHub Actions, Gitlab CI/CD), les *registry* d'images et d'artefacts (Amazon ECR, AWS CodeArtifact, JFrog Artifactory, etc.).
- **La couche ressource** : c'est la couche sur laquelle les ressources vont s'exécuter. Elle peut comprendre les services de calcul, de stockage ou de réseau d'un fournisseur Cloud.
- **La couche d'observabilité** : elle est consacrée à la récupération et à la lecture des logs, des métriques et des traces des environnements, ainsi qu'à la configuration des alertes. On peut y retrouver des outils comme Datadog, Dynatrace, Amazon CloudWatch...
- **La couche de sécurité** : elle est dédiée au stockage des données sensibles, grâce à des outils comme HCP Vault ou à des services comme AWS Secret Manager.



Architecture de référence d'un IDP sur AWS (source : platformengineering.org)



LE POINT DE VUE DE NOS EXPERTS

« Quels outils ou technologies considérez-vous comme essentiels pour construire une plateforme efficace ? Pouvez-vous nous donner un exemple concret d'automatisation ou d'intégration, qui a significativement amélioré les flux de travail des développeurs ? »

TIMOTHÉE AUFORT

est Practice Leader de la Practice Cloud & DevOps chez Ippon Technologies.

Les technologies pouvant constituer une plateforme efficace sont nombreuses et doivent, comme on l'a mentionné précédemment, venir avant tout répondre à de vrais besoins clients. Elles apportent à la fois :

— **Des services (liste non exhaustive) :**
- une plateforme d'intégration continue (GitLab CI, GitHub Actions…)
- des services de vérification de qualité (Checkmarx, Sonarqube…)
- un gestionnaire de secrets (Hashicorp Vault)
- un portail développeur pour créer des services applicatifs ou des workflows (Backstage)

— **Et des outils (idem) :**
- KICS pour scanner les vulnérabilités du code infrastructure
- Trivy pour scanner des images de containers ou bien des images machine des modules Terraform pour créer plus rapidement des briques d'infrastructure sur votre Cloud provider préféré
- des images docker de base standardisées pour faire du Build ou du Run

AU-DELÀ DU CHOIX DES TECHNOLOGIES, IL FAUT SURTOUT QU'ELLES INTERAGISSENT FACILEMENT POUR SIMPLIFIER LA VIE DES ÉQUIPES. UNE DOCUMENTATION UNIFIÉE ET VIVANTE EST DE MISE.

Un autre impératif pour favoriser l'engagement de vos clients est d'être transparent sur la façon dont ces technologies sont gérées ; il est primordial selon moi de favoriser au maximum « l'innersource » et d'encourager les clients à contribuer.

CAS D'USAGE

Chez notre client Bedrock Streaming, l'équipe plateforme a mis en place une toute nouvelle DevFactory fondée notamment sur GitHub Actions, en lieu et place d'un Jenkins vieillissant et ne répondant plus aux besoins de l'entreprise. De nombreux outils sont arrivés en sus de cette migration pour standardiser et mutualiser les usages des équipes : des charts Helm de base, SOPS pour chiffrer les secrets dans le code, Helmfile pour déployer un ensemble de charts Helm sur Kubernetes, Renovate pour gérer l'obsolescence logicielle… Tout cela s'est fait de façon automatisée avec des pipelines de CI/CD centralisées et réutilisables (bienvenue aux reusable workflows de GitHub Actions !). Cette migration de DevFactory a permis aux équipes de rembourser une grosse part de la dette technique (nombreuses duplications de code, secrets en clair…), d'automatiser un grand nombre d'actions qui étaient manuelles ou partiellement manuelles (le déploiement du code Terraform par exemple), d'améliorer la posture sécurité des projets ou encore de favoriser le déploiement continu jusqu'en production.

CAS D'USAGE

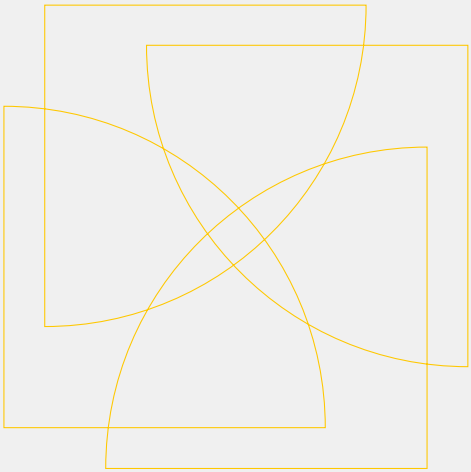
VIVEN MALEZE

est architecte Technique chez Ippon Technologies.

Voici la stack technologique que nous avons mise en place chez l'un de mes clients, pour une plateforme efficace :

- **Kubernetes** : pour orchestrer les conteneurs.
- **GitLab** : pour automatiser les déploiements via des pipelines CI/CD.
- **ArgoCD** : pour des déploiements GitOps simples et fiables.
- **OpenTelemetry** : pour l'observabilité, avec des backends comme :
  - **Grafana** : visualisation des métriques et création de dashboards.
  - **Prometheus** : collecte des métriques et gestion de l'alerting.
  - **Jaeger** : suivi des traces distribuées.
  - **Elastic/Kibana** : gestion et exploration des logs.
- **Backstage** : pour centraliser et simplifier l'expérience des développeurs.

Un bénéfice concret ?
Je mentionnerais notamment l'ajout de traces distribuées avec OpenTelemetry. Cela a permis de réduire de moitié le temps d'analyse des incidents. En cas de timeout, nous pouvons immédiatement identifier le service responsable grâce aux traces. Dans un cas spécifique, nous avons localisé un appel SQL inefficace, optimisé la requête et corrigé rapidement le problème, ce qui a considérablement amélioré les performances du système.

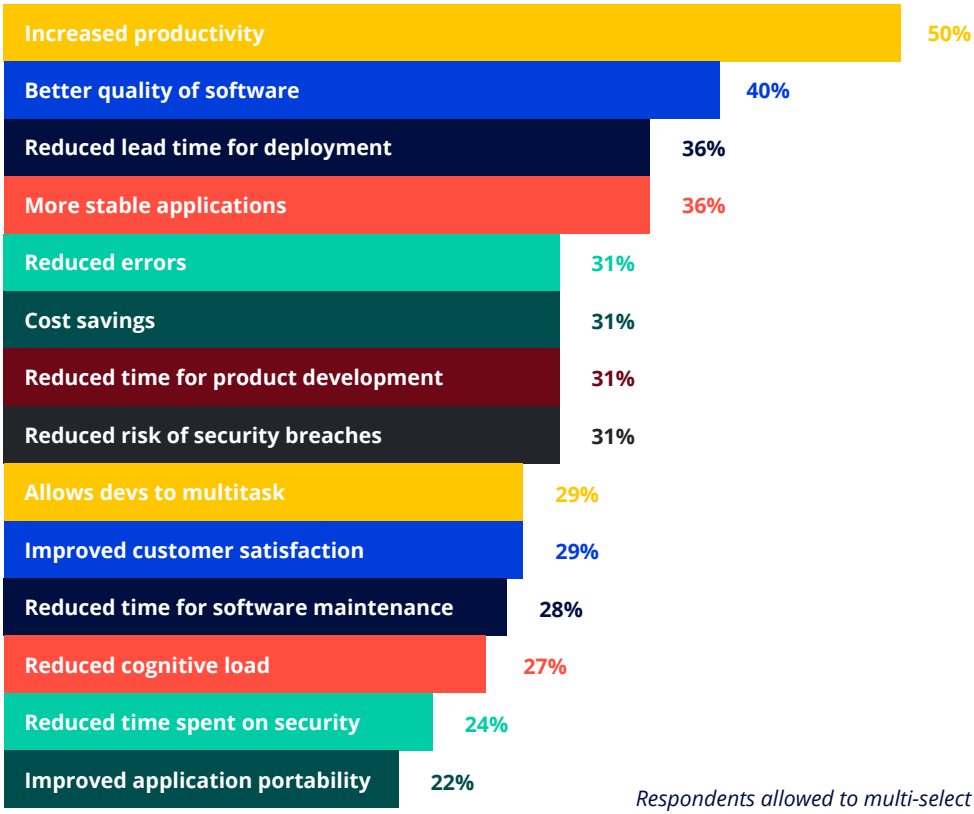


03 LES BÉNÉFICES DU PLATFORM ENGINEERING

L'approche Platform Engineering présente de grands avantages (source : *State Of DevOps Report 2020* by Puppet) :

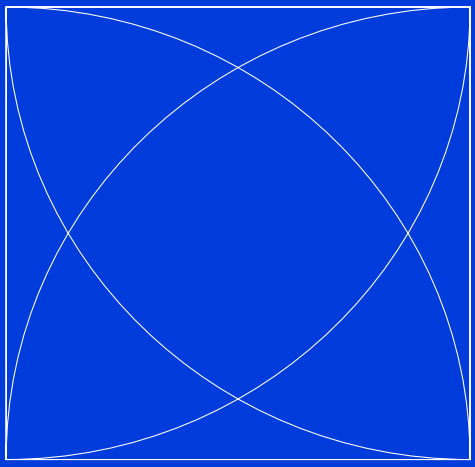
- **Les équipes de développement sont plus efficaces** : le principe du DevOps était de rendre les équipes autonomes sur leur infrastructure et leur outillage, malgré une montée en compétence qui restait très difficile à réaliser. Avec le Platform Engineering, elles peuvent atteindre plus facilement cet objectif, sans complexité technique. L'institut Dora, dans son **Accelerate State of DevOps Report 2024**, chiffre à 8 % l'amélioration des performances individuelles des développeurs qui ont accès à une plateforme de développement interne, et à 10 % celle des équipes entières.
- **La gouvernance est améliorée** : en standardisant les infrastructures et les outils, il est plus aisé d'en maîtriser la configuration, leur coût et leur sécurité.
- **Les développeurs sont apaisés** : le fait de devoir à la fois gérer la partie applicative et l'infrastructure leur demandait une grande polyvalence, mais les contraignait surtout à changer constamment de contexte, ce qui peut s'avérer fatiguant et stressant.
- **L'infrastructure progresse en continu** : l'infrastructure est gérée par l'équipe plateforme, qui se fonde sur les retours des utilisateurs pour améliorer le produit.

PLATFORM ENGINEERING'S KEY BENEFITS FOR DEVELOPERS :



Respondents allowed to multi-select

Les développeurs attribuent une quinzaine d'avantages au Platform Engineering, de la productivité à la portabilité, en passant par les réductions de coûts (source : *Puppet State of DevOps Report 2020*).



LE POINT DE VUE DE NOS EXPERTS

« En quoi le Platform Engineering peut-il influencer la stratégie business d'une organisation ? Comment les C-levels peuvent-ils mesurer son succès et en maximiser la valeur ? »

TIMOTHÉE AUFORT

Practice Leader de la Practice Cloud & DevOps chez Ippon Technologies

UN ACCÉLÉRATEUR DE BUSINESS

> LE PLATFORM ENGINEERING PEUT AVOIR UN IMPACT SIGNIFICATIF SUR LA STRATÉGIE BUSINESS, EN ACCÉLÉRANT LE TIME-TO-MARKET DES PRODUITS, EN AMÉLIORANT LA PRODUCTIVITÉ DES DÉVELOPPEURS ET EN RENFORÇANT LA FIABILITÉ DES SYSTÈMES.

Il permet aux équipes de se concentrer sur l'innovation et la création de valeur. Cela peut se traduire par une meilleure satisfaction client, une capacité accrue à répondre rapidement aux évolutions du marché, et une réduction des coûts opérationnels grâce à l'automatisation.

VIVEN MALEZE

Architecte Technique chez Ippon Technologies

ON N'AMÉLIORE QUE CE QUE L'ON MESURE

Pour que la plateforme soit un succès, il faut d'abord qu'elle soit adoptée. Et pour être adoptée, il faut qu'elle ait des impacts positifs au sein des équipes produits. Il est donc primordial pour les C-levels de mesurer ses incidences, notamment via les métriques « DORA » : fréquence de déploiement, délai moyen de modification, taux d'échec des modifications et délai de rétablissement d'un service.

Il est important d'utiliser aussi des métriques de satisfaction des équipes, à personnaliser car elles sont spécifiques au contexte. Enfin, n'oubliez pas de communiquer clairement sur les bénéfices obtenus pour maintenir l'engagement des parties prenantes.

04 TOUT N'EST PAS ROSE

LES « EFFETS SECONDAIRES » DU PLATFORM ENGINEERING

Les avantages du Platform Engineering ne valent que si la plateforme que vous proposez répond parfaitement aux besoins des développeurs. Lorsque ce n'est pas le cas, cela peut à l'inverse être très frustrant et les freiner dans leur travail.

Le *Accelerate State of DevOps Report 2024* souligne deux effets secondaires liés à la mise en place d'une plateforme de développement :

La **diminution du « throughput »**, à savoir la quantité de travail effectuée sur une période donnée (débit), est de l'ordre de - 8%. D'après le rapport, ce phénomène s'explique de la manière suivante : les développeurs ont généralement l'obligation de ne passer que par la nouvelle plateforme pour gérer l'entièreté du cycle de vie de leurs applications, ce qui les fait entrer dans un processus de déploiement plus complexe, dans lequel leur code est soumis à des batteries de tests automatisés et humains et des scans de sécurité qu'ils n'auraient pas forcément tous mis en place sans plateforme.

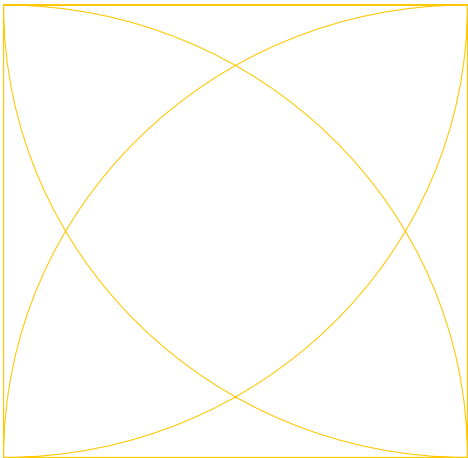
Voici la preuve d'un réel impact négatif si les besoins des utilisateurs ne sont pas pris en compte ! La plateforme devient rapidement une contrainte, alors qu'elle était censée les aider. En reprenant l'exemple des tests automatisés, on peut envisager de leur laisser une certaine liberté dans le choix des tests - voire la possibilité d'ajouter les leurs.

À prévoir également, une **augmentation du taux d'échec au déploiement** (*Change Failure Rate*). En effet, malgré l'apparente rigueur qu'apporte l'utilisation d'une plateforme, ce taux d'échec au déploiement augmente en moyenne de 14 %. Il serait dû à la fois à la hausse du nombre de travaux de correction, à une augmentation de la charge de travail et à un phénomène de « burnout » au sein des équipes de développement, comme l'explique le rapport :

— **Les développeurs expérimentent davantage**, ils prennent plus de risques car ils comptent sur les garde-fous de la plateforme. Mais cette confiance excessive peut produire du code moins stable, appelant des corrections qui alourdissent finalement la charge de travail.

— **La plateforme n'est pas parfaite** et malgré les efforts des équipes qui la gèrent, il est impossible de prévoir tous les scénarios et tous les tests de validation. Avec la montée en charge de la plateforme, certaines failles peuvent échapper aux contrôles et conduire au déploiement d'applications défectueuses.

— Les entreprises qui adoptent le Platform Engineering peuvent précisément être **celles qui connaissent déjà de fortes instabilités et du burnout**. La difficulté et la charge de travail que représente le lancement de la plateforme peuvent aggraver ces symptômes au lieu de les réduire.



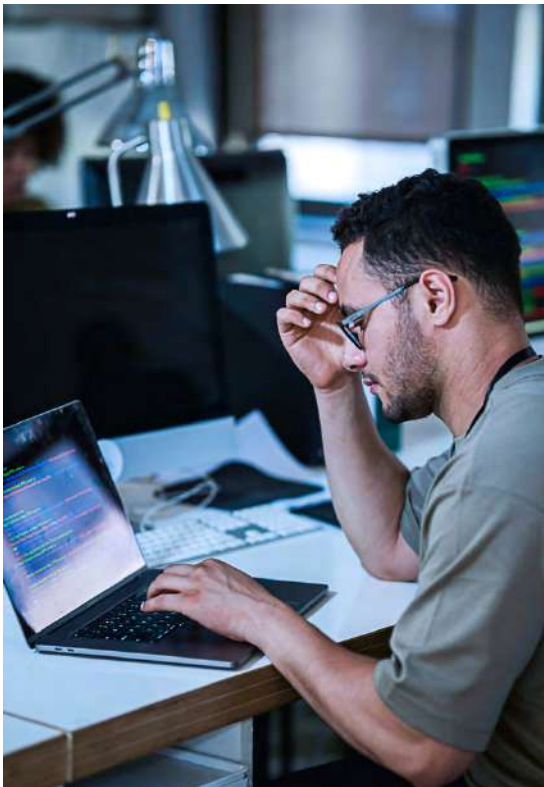
TIMOTHÉE AUFORT

Practice Leader de la Practice Cloud & DevOps chez Ippon Technologies

Je vois une autre raison expliquant l'augmentation du taux d'échec au déploiement : le manque de connaissance des développeurs sur les processus d'échanges avec l'équipe plateforme.

J'observe beaucoup d'allers-retours entre les équipes produits et l'équipe plateforme qui sont dus à la mauvaise utilisation d'un service ou d'un outil. Par exemple, la documentation n'était peut-être pas claire à la base ; dans ce cas-là, c'est l'équipe plateforme qui est responsable.
Autre exemple : un bug a été découvert par les équipes produits. C'est « bien », mais ça signifie surtout qu'elle ne font pas pas confiance à la plateforme, puisqu'elles ont pris le temps de tester un outil ou un service pour découvrir ce bug.

En plus de ces effets secondaires listés par l'Institut DORA, j'en vois d'autres qui pourraient affecter les développeurs. Nous partons du principe que nous les aidons en leur permettant de se concentrer seulement sur le développement de leurs produits, mais je connais bon nombre de développeurs chez Ippon Technologies qui aiment configurer, tester des outils par eux-mêmes et les mettre en place pour leur équipe. Le Platform Engineering ne risque-t-il pas de les brider, de les empêcher de s'épanouir ?
Trop de standardisation peut être frustrant. Mais lorsqu'une entreprise se développe, cela devient nécessaire pour éviter les dérives.



QUATRE CONSEILS POUR LIMITER LES RISQUES

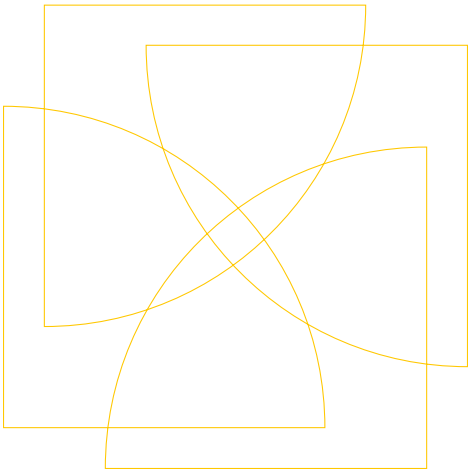
Bonne nouvelle, ces effets indésirables restent facilement évitables, à condition de :

- **Prioriser** les fonctionnalités à mettre en place au sein de la plateforme, en fonction des besoins du plus grand nombre ;
- Privilégier les fonctionnalités qui permettent aux développeurs d'être **autonomes** sur la gestion de leurs applications ;
- Laisser aux développeurs la **liberté** d'adapter certaines fonctionnalités à leurs besoins ;
- Prévoir une phase d'**évangélisation** à destination des développeurs, pour bien leur expliquer le fonctionnement de la plateforme et leur indiquer les bonnes pratiques.

> LA PLATEFORME EST AU SERVICE DES DÉVELOPPEURS - ELLE N'EST PAS LÀ POUR ÊTRE UN FREIN ! ÉVITEZ DONC LA POSTURE « ILS N'ONT QU'À S'Y CONFORMER » ET PRIVILÉGIEZ L'APPROCHE « COMMENT PUIS-JE RÉPONDRE À LEURS BESOINS ? »

Le Platform Engineering permet aux équipes de se concentrer sur l'innovation et la création de valeur. Cela peut se traduire par une meilleure satisfaction client, une capacité accrue à répondre rapidement aux évolutions du marché, et une réduction des coûts opérationnels grâce à l'automatisation.
Il est également nécessaire de faire preuve d'une certaine tolérance aux instabilités. Commencez par fixer avec les développeurs le seuil de tolérance à l'instabilité de leurs applications, pour leur indiquer dans quelle mesure ils peuvent (ou pas) expérimenter de nouvelles choses.

Un bon moyen de cadrer ce travail est de mettre en place des indicateurs empruntés au monde du *Site Reliability Engineering* (SRE) comme des SLOs (*Service Level Objectives*) et des SLAs (*Service Level Agreements*). Prévoyez également un « budget d'erreurs » en amont pour les instabilités et le manque à gagner.



LE POINT DE VUE DE NOS EXPERTS

Adopter le Platform Engineering et en tirer tous les avantages n'est pas si simple que cela. J'ai donc posé la question suivante à nos experts :

« Quels sont les principaux défis rencontrés lors de la mise en œuvre d'une stratégie de Platform Engineering ? Comment une organisation peut-elle surmonter ces obstacles et s'assurer de l'adoption par les équipes de développement ? »

VIVEN MALEZE

Architecte Technique chez Ippon Technologies

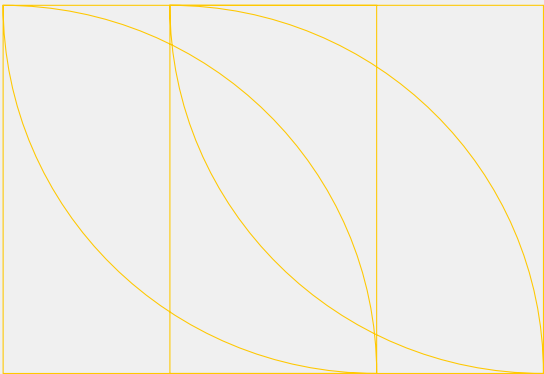
L'UTILISATEUR DOIT RESTER AU CENTRE

Il faut absolument aller discuter avec les développeurs pour comprendre leurs besoins et éviter la mise en place d'outils ou de workflows inadaptés.
Ensuite, la résistance au changement est fréquente : les équipes de développement peuvent hésiter à adopter de nouvelles pratiques ou rechigner à modifier leurs habitudes. On peut générer de la frustration si tout est standardisé et qu'il ne reste plus de marge de manœuvre pour explorer.
L'intégration des outils dans un environnement existant peut aussi se heurter à des contraintes techniques ou organisationnelles souvent problématiques.

Pour surmonter ces obstacles, adoptez une approche centrée sur les utilisateurs. Cela passe par une collaboration étroite avec les équipes de développement dès le début, des échanges réguliers, et l'itération sur les solutions que vous proposez.

COMMUNIQUEZ AVEC PRÉCISION SUR LES BÉNÉFICES, COMME LE GAIN DE TEMPS OU LA RÉDUCTION DES ERREURS. CELA FAVORISE L'ADOPTION.

Enfin, investissez dans la formation et le support pour accompagner les équipes dans la transition.



TIMOTHÉE AUFORT

Practice Leader de la Practice Cloud & DevOps chez Ippon Technologies

LA COMMUNICATION, ÇA S'ORGANISE !

Le principal défi est de livrer des fonctionnalités répondant aux besoins des équipes de développement consommateurs de la plateforme et d'éviter les dérives techniques. Trop souvent, j'ai constaté chez mes clients la mise en place de services sans qu'une récolte de besoins n'ait été faite au préalable. Il en résulte de nouveaux services à maintenir pour l'équipe plateforme, alors même que ces derniers sont peu, voire pas utilisés.

Pour éviter ce type de dérive, il faut vraiment adopter une démarche produit en allant au contact des utilisateurs finaux de la plateforme : via des interviews, des questionnaires ou plus généralement du support.
Attention, avec le Platform Engineering, on gère un produit vraiment technique, il sera donc très difficile pour un consultant Product Owner ne connaissant pas la tech d'effectuer une récolte de besoins efficace.

À mon sens, il est préférable que les développeurs de l'équipe plateforme prennent en charge le rôle de Product Owner. Il peut s'agir d'un rôle tournant, en fonction de la maturité de l'équipe.

Le tout, avec l'appui d'un Product Manager ayant une vision amont, pour identifier les macro-besoins des équipes produits.

Le second défi qui me vient en tête est de favoriser la collaboration inter-équipes. Multiplier les équipes peut faire sens quand une entreprise grossit, mais il faut être vigilant pour éviter les silos. Une mauvaise communication entre l'équipe plateforme et les équipes produits nuira à la confiance. L'équipe plateforme devra être attentive à cela, organiser des points de retour d'information avec ses clients ainsi que des démonstrations régulières.

CAS D'USAGE

Chez EDF, nous avons pris l'habitude de faire des démonstrations hebdomadaires à tous nos clients, soit une vingtaine d'équipes de développement en décembre 2024. Le format baptisé « Kiosque Clients » est totalement piloté par l'équipe plateforme en termes de contenu et permet à tous les membres de l'équipe de s'exprimer sur les nouveautés de leur zone de responsabilités. Chez Bedrock Streaming, l'approche était assez similaire à celle d'EDF avec des démonstrations récurrentes. En plus, **pour créer de la confiance et garder un lien fort avec les équipes de développement, chaque membre de l'équipe plateforme se voyait confier le support de plusieurs équipes-produits désignées ; cela permettait aux clients d'avoir un seul interlocuteur privilégié.**

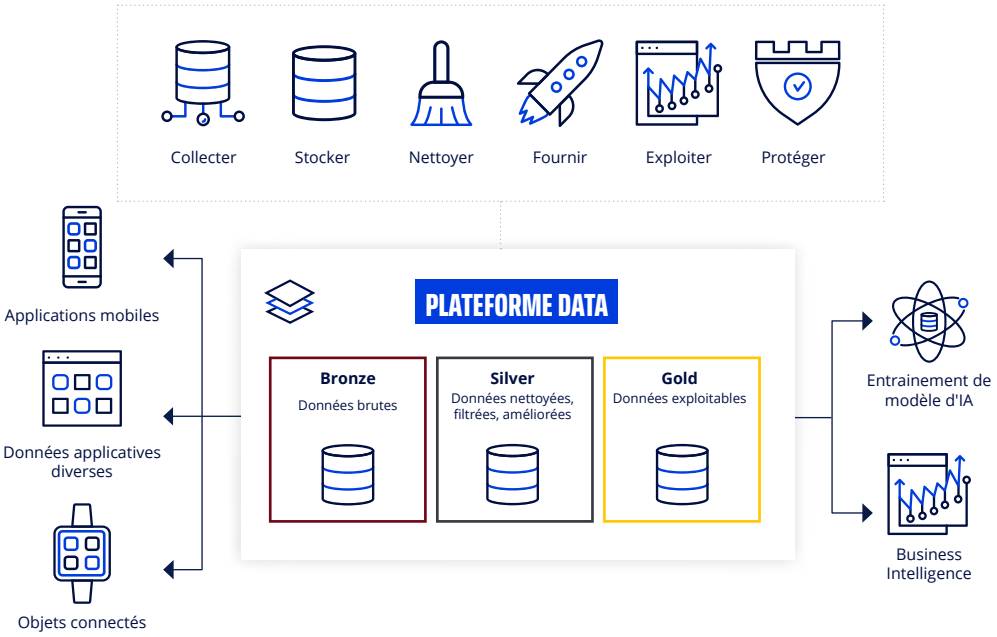
05 ZOOM SUR LA PLATEFORME DATA

Pour m'aider à rédiger cette section du livre blanc et aller plus loin sur le sujet de la Data Platform, j'ai fait appel à quelqu'un qui maîtrise mieux le sujet que moi, **Julide YILMAZ,** Architecte MLOps chez Ippon Technologies.

J'ai rédigé cet ouvrage avec mon regard d'ingénieur Cloud & DevOps et forcément je me suis concentré sur le Platform Engineering dans le cadre du développement informatique, mais l'approche vaut également dans le monde de la Data. La plateforme Data est même l'une des principales illustrations dans l'IT de la « plateformisation » des relations entre les métiers.

APRÈS L'APPARITION DU TERME DEVOPS, DE NOUVELLES TERMINOLOGIES SONT NÉES POUR APPLIQUER LA MÊME PHILOSOPHIE À D'AUTRES DOMAINES (SECOPS, MLOPS, DATAOPS, LLMOPS) OU SIMPLEMENT POUR EN INCLURE D'AUTRES (DEVSECOPS).

Cela montre que même si cette approche a été pensée initialement pour le développement informatique, ses principes peuvent très bien s'appliquer à d'autres métiers. Et c'est également le cas du Platform Engineering.



Le Platform Engineering adapté à la Data :
quand la donnée devient un produit à part entière

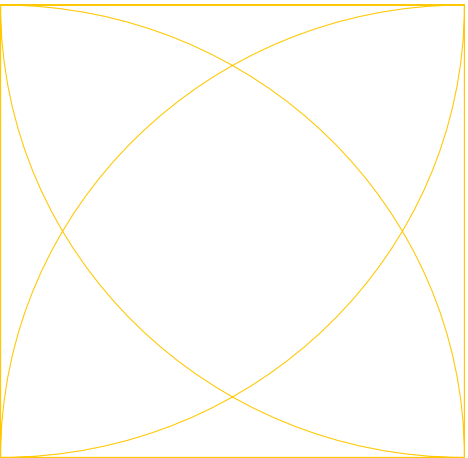
VERS UNE APPROCHE « DATA AS A SERVICE »

L'apparition des plateformes Data traduit un changement fondamental : le passage d'outils Data isolés (bases de données, ETL, BI) à une approche de service intégrée. C'est l'ère de la « Data as a Service » : plutôt que de fournir simplement un accès à une base de données, on propose un service complet comprenant APIs, documentation et garanties de service.

Plusieurs prérequis doivent être réunis. Il faut s'assurer de la qualité des données, avec une gouvernance efficace. Une plateforme unifiée doit être mise en place pour centraliser l'accès aux données. Des APIs doivent être développées pour automatiser les échanges. L'infrastructure sous-jacente doit être sécurisée et capable de monter en charge le moment venu.

Cette transformation est bien plus que technique : elle représente une nouvelle façon de considérer la donnée comme un véritable produit, avec ses propres cycles de vie et niveaux de service.

Une **plateforme Data** (ou plateforme de données) est une infrastructure technologique qui centralise, gère et met à disposition les données d'une organisation de manière structurée et accessible pour diverses utilisations, telles que l'analyse, la visualisation, le Machine Learning ou la prise de décision. Ses fondateurs sont, le plus souvent, les Data Engineers ou les Data Scientists et ses utilisateurs seront des équipes-produits venant consommer les données fournies par cette plateforme. Chez Ippon Technologies, nos ingénieurs de la practice Data travaillent régulièrement à la mise en place de ce type de systèmes.



LES SIX FONCTIONS D'UNE PLATEFORME DATA

Collecter/ Intégrer les données | Stocker les données | Nettoyer les données | Mettre à disposition les données | Exploiter les données | Assurer une gouvernance de données

Comme la plateforme de développement permet aux développeurs de gérer le cycle de vie complet de leurs applications, la plateforme Data permet aux Data Engineers ou Data Scientists de gérer le cycle de vie des données.

Collecter et ingérer des données

La première tâche est de centraliser les données, qui proviennent de sources multiples (bases de données, API, capteurs IoT, systèmes internes, réseaux sociaux, sites ou applications web et mobile…) et prennent des formats variés : données structurées, semi-structurées, non structurées. La plateforme peut utiliser des trackers et des connecteurs par exemple, pour récupérer toutes ces données.

Stocker des données

Avant de pouvoir transformer et utiliser ces données, la plateforme doit offrir un espace de stockage sécurisé et scalable. On parle de **data warehouse** lorsqu'elle permet de stocker des données structurées pour les analyses (exemples : Snowflake, Google BigQuery, Amazon Redshift) ou de **data lake** lorsqu'elle permet de stocker de grandes quantités de données non structurées ou semi-structurées (exemples : AWS S3, Azure Data Lake, Hadoop).
Il existe également les plateformes de type **data lakehouse** (exemples : Databricks, Delta Lake, Sage Maker) qui combinent les avantages des data lakes (flexibilité) et des data warehouses (performances analytiques). Un dernier type de data platform est le streaming data platform lorsqu'on gère les flux de données en temps réel, avec des technologies comme Apache Kafka ou Amazon Kinesis.

Au cours de leur évolution, les données vont être stockées dans différentes tables ou bases de données en fonction de leur niveau de maturité : bronze, silver et gold (lire page 57).

Transformer et traiter des données

Il faut que les données brutes issues de différentes sources soient nettoyées, enrichies et transformées pour les rendre exploitables. Cela s'opère via des pipelines automatisés.

Mettre à disposition des données

La plateforme Data propose des outils pour permettre aux utilisateurs (analystes, data scientists, métiers) d'accéder aux données via des API, des tableaux de bord ou des interfaces self-service.

Analyser et exploiter des données

La plateforme Data a pour objectif de fournir aux Métiers des données exploitables. Des outils d'analyse et de visualisation (comme Power BI, Tableau, ou Looker) peuvent être directement intégrés et supporter des cas d'usage avancés, IA et Machine Learning compris.

Assurer une gouvernance des données

Il est indispensable que la plateforme embarque des politiques de gestion des données pour assurer leur qualité, leur sécurité, mais aussi leur conformité aux réglementations en vigueur, RGPD et HIPAA notamment. Cela passe aussi par la gestion des autorisations d'accès aux données.

OUI, LES DONNÉES SONT UN PRODUIT

Dans une approche Platform Engineering, chaque jeu de données devient un produit à part entière :

- **Bronze** : données brutes ingérées, avec garantie d'immutabilité
- **Silver** : données nettoyées et validées, prêtes pour l'analyse
- **Gold** : données enrichies et agrégées pour des cas d'usage spécifiques

Cette approche produit s'accompagne d'exigences similaires à celles du développement applicatif, mais adaptées au monde de la donnée. Elle nécessite d'identifier clairement un propriétaire responsable du produit de données, tout comme un produit applicatif a son Product Owner.

> LES NIVEAUX DE SERVICE DOIVENT ÊTRE FORMELLEMENT DÉFINIS, AVEC UNE ATTENTION PARTICULIÈRE PORTÉE À LA FRAÎCHEUR ET LA DISPONIBILITÉ DES DONNÉES - LÀ OÙ UNE APPLICATION TRADITIONNELLE SE CONCENTRE PRINCIPALEMENT SUR SA DISPONIBILITÉ ET SES PERFORMANCES.

ORGANISATION ET GOUVERNANCE

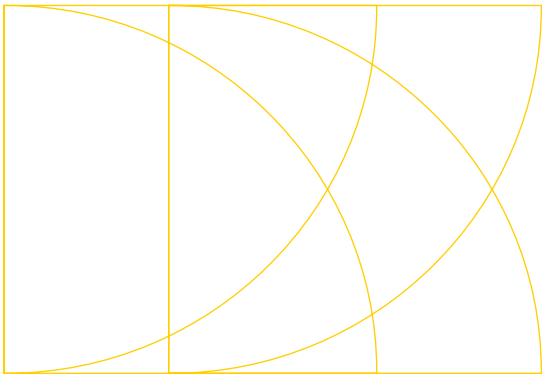
L'efficacité d'une plateforme Data repose sur un équilibre entre autonomie et contrôle :

Équipes et responsabilités

En complément des rôles Platform Engineering déjà évoqués, l'équipe s'enrichit de profils spécialisés dans la donnée. L'équipe plateforme développe et maintient les services-socles tandis que les Data Engineers créent et maintiennent les pipelines de données.

Gouvernance intégrée

La gouvernance intégrée transforme les contraintes traditionnelles en services natifs de la plateforme. Elle repose sur un catalogue de données unifié qui se met à jour automatiquement, assurant ainsi une vision claire du patrimoine de données. La qualité des données est constamment mesurée et garantie par des contrôles automatisés, tandis que la traçabilité des usages est intégrée nativement dans la plateforme. Enfin, la sécurité et la conformité ne sont plus des couches additionnelles, mais des composants « by design » de l'architecture, permettant une utilisation des données à la fois fluide et maîtrisée.



Trois spécificités

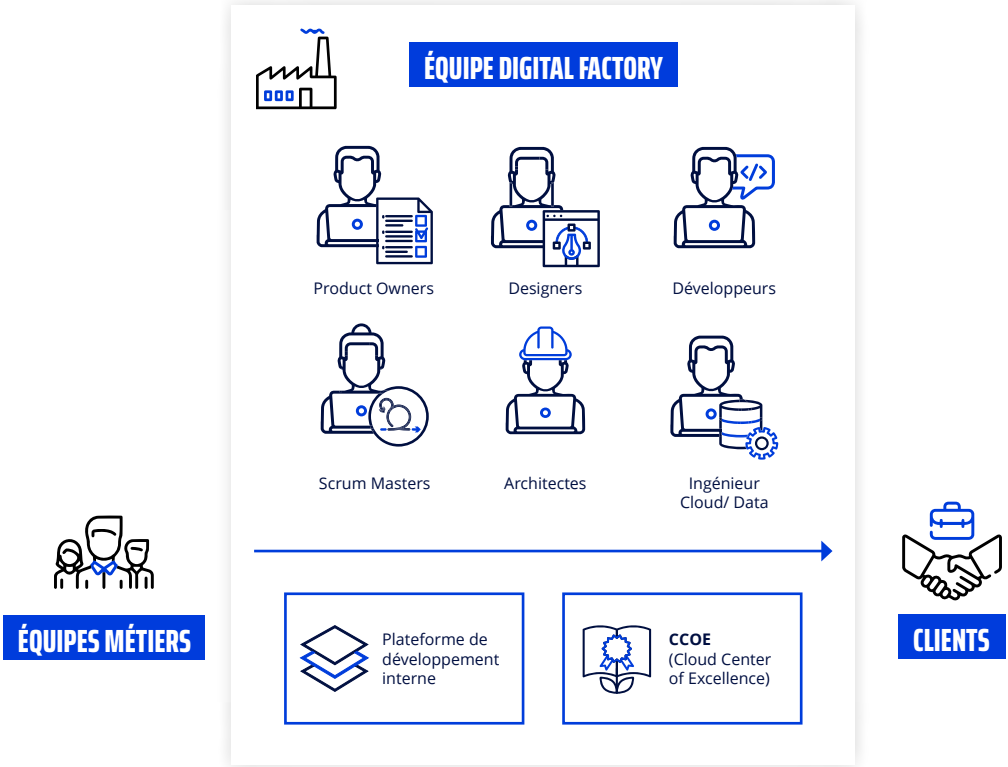
La plateforme Data partage de nombreux principes fondamentaux avec les plateformes DevOps, notamment l'automatisation poussée des processus, une approche privilégiant le self-service, l'Infrastructure *as Code* et une observabilité native. Elle se distingue cependant par trois caractéristiques : la gestion de volumes importants de données, la complexité des transformations à opérer sur ces données, et des enjeux de qualité et de gouvernance particulièrement critiques. Même chose dans l'approche produit, où les données sont traitées comme des produits à part entière, avec leurs propres cycles de vie, niveaux de service et indicateurs de qualité, tout en s'intégrant dans la chaîne d'outils DevOps existante.
Cette approche évite les silos traditionnels entre équipes Data et DevOps, créant un environnement où les données sont traitées comme des produits de première classe, avec le même niveau d'automatisation et de rigueur que le code applicatif.

06 VERS LA DIGITAL FACTORY ?

Le Platform Engineering se rapproche d'un autre concept plus poussé sur l'aspect produit : la Digital Factory.

> UNE **DIGITAL FACTORY** (OU USINE DIGITALE EN FRANÇAIS) EST UNE STRUCTURE ORGANISATIONNELLE CONÇUE POUR ACCÉLÉRER LA CRÉATION DE PRODUITS ET SERVICES NUMÉRIQUES ET EN MÊME TEMPS LA TRANSFORMATION NUMÉRIQUE D'UNE ENTREPRISE.

Elle regroupe des ressources humaines (des développeurs, des profils DevOps, des ingénieurs Cloud, des ingénieurs Data, des Scrum Masters, des Product Owners, etc.), des ressources technologiques (plateforme de développement interne par exemple) et méthodologiques (agile, DevOps), le tout pour développer rapidement des solutions numériques innovantes, telles que des applications, des plateformes ou des services digitaux, en réponse aux besoins des clients ou des métiers de l'entreprise.



La Digital Factory produit des solutions numériques « prêtes à consommer»

LES QUATRE CARACTÉRISTIQUES D'UNE DIGITAL FACTORY

Une équipe pluridisciplinaire

Une Digital Factory a la particularité de réunir des compétences variées autour d'un même objectif. Elle peut être composée de développeurs, de designers UX/UI, d'experts en data, de chefs de projet, de spécialistes en marketing digital, etc.

Méthodologies agiles et DevOps

Bien qu'elle soit généralement intégrée à l'entreprise, la Digital Factory fonctionne de manière autonome avec des cycles courts de développement (sprints), favorisant la rapidité et la prise de décisions.

Cette équipe peut s'appuyer sur le **design thinking** pour concevoir des produits qui répondent au besoin de ses clients, sur des frameworks **agiles** (Scrum, Kanban ou SAFe) pour gérer les projets de manière itérative et sur la philosophie **DevOps** pour livrer rapidement de nouveaux produits numériques.

Technologies avancées

Grâce aux différents profils qui la composent, la Digital Factory s'appuie sur des technologies modernes comme le cloud computing, l'intelligence artificielle, l'IoT (Internet des objets), ou des outils d'automatisation pour maximiser son efficacité.

Focus sur l'utilisateur final

La principale caractéristique de la Digital Factory est qu'elle se concentre sur la **création de produits numériques complets**, à destination des utilisateurs finaux (clients, collaborateurs, équipes métiers, marketing, vente ou autre). Elle est souvent proche des équipes métiers pour s'assurer que les solutions répondent aux besoins business.

QUELLE DIFFÉRENCE ENTRE DIGITAL FACTORY ET PLATFORM ENGINEERING ?

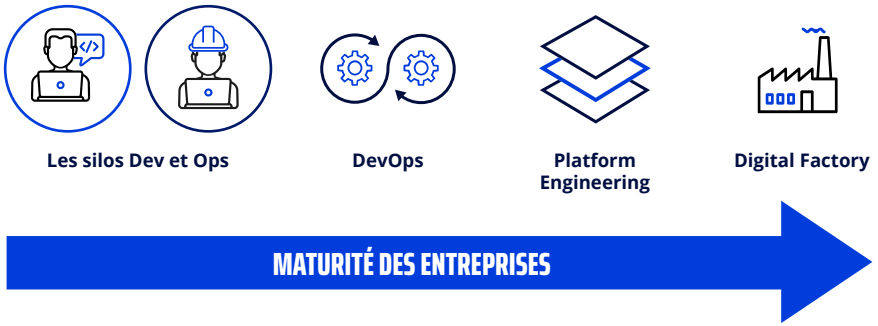
Ce sont deux concepts différents, qui ne s'adressent pas aux mêmes utilisateurs, mais qui sont tout aussi importants dans la transformation numérique des entreprises.

LA **DIGITAL FACTORY** EST ORIENTÉE VERS LA CRÉATION DE SOLUTIONS NUMÉRIQUES POUR RÉPONDRE AUX BESOINS DES UTILISATEURS FINAUX, TANDIS QUE LE PLATFORM ENGINEERING FOURNIT UNE BASE TECHNOLOGIQUE SOLIDE POUR QUE LES ÉQUIPES INTERNES (LES DÉVELOPPEURS, LES INGÉNIEURS DATA OU LES DATA SCIENTISTS) PUISSENT TRAVAILLER EFFICACEMENT ET DÉVELOPPER CES SOLUTIONS. LES DEUX APPROCHES SONT SOUVENT COMPLÉMENTAIRES

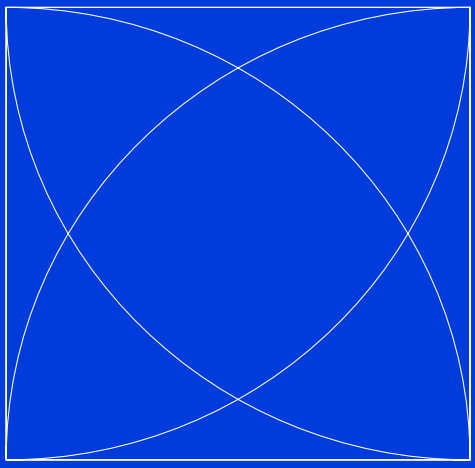
et coexistent dans une organisation moderne. La Digital Factory pourrait très bien utiliser une plateforme de développement interne pour développer ses produits.

	DIGITAL FACTORY	PLATFORM ENGINEERING
OBJECTIF PRINCIPAL	Produire des solutions numériques prêtes à l'emploi.	Standardiser et optimiser les flux techniques.
CIBLE	Utilisateurs finaux ou métiers.	Équipes techniques.
LIVRABLES	Produits numériques (applications, sites).	Plateformes, outils, pipelines.
FOCUS	Innovation produit et rapidité.	Efficacité technique et scalabilité.
APPROCHE	Axée sur les besoins utilisateurs.	Axée sur l'infrastructure et l'automatisation.

Même si les deux concepts ne sont pas directement liés, ce sont des marqueurs d'un certain niveau de maturité des équipes technologiques. Je considère que le Platform Engineering est l'une des briques permettant de mettre en place une Digital Factory, qui peut être perçue comme son évolution naturelle.



Les quatre âges de l'équipe IT



LE POINT DE VUE DE NOS EXPERTS

« Quel est l'intérêt pour une entreprise de basculer vers une organisation avec une Digital Factory ? »

RÉMY OLIVET

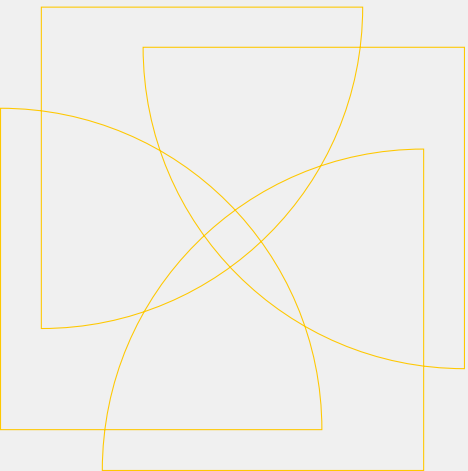
Practice Leader Data chez Ippon Technologies

La Digital Factory peut inclure le Platform Engineering, mais représente un concept plus large et plus centré sur l'accélération du développement et de l'innovation des produits numériques. Elle vise à intégrer une multitude de processus, d'outils et de méthodologies permettant de concevoir, de développer, de tester, de déployer et de maintenir des produits numériques plus rapidement, avec plus d'efficacité. La Digital Factory va au-delà de la gestion d'une plateforme et s'attaque directement à l'ensemble du cycle de vie des produits numériques. Le modèle est centré sur la création de valeur produit.

CAS D'USAGE

Je travaille aujourd'hui pour un organisme bancaire qui a opté pour une Digital Factory. Nous nous sommes organisés en trois couches :

- Une équipe de Platform Engineering ;
- Une Digital Factory qui construit, sur cette plateforme, des services orientés métiers, réutilisables, tels que la gestion documentaire ou le KYC (Know Your Customer) ;
- Les équipes de développement « stream-aligned » : elles mettent en place les solutions à destination des utilisateurs finaux (ici, les clients de la banque), en s'appuyant sur les services mis à disposition par la Digital Factory.

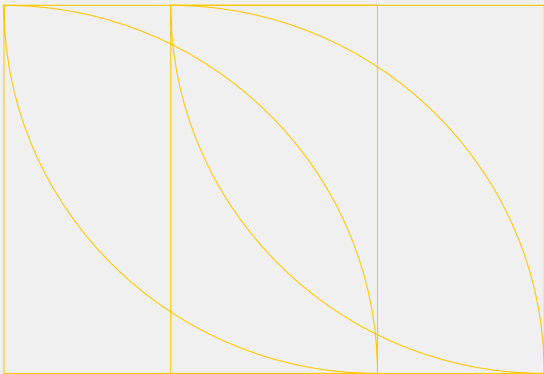


JULIE KOCIANOVA

Product Owner chez Ippon Technologies

> UNE DIGITAL FACTORY PRODUIT DES SOLUTIONS NUMÉRIQUES PRÊTES À L'EMPLOI, FACILEMENT ADOPTABLES PAR LEUR UTILISATEUR-CIBLE. SON TRAVAIL VA ÊTRE SIMPLIFIÉ PAR L'EXISTENCE D'UNE PLATEFORME, QUI PERMET AUX ÉQUIPES-PRODUITS D'ALLER PLUS LOIN ET PLUS VITE EN TERMES DE NOUVELLES FONCTIONNALITÉS ET D'EXPÉRIENCE UTILISATEUR.

Une entreprise moderne a donc tout intérêt à adopter une stratégie de plateformisation au sein de sa Digital Factory. La plateforme propose des services réutilisables et indépendants, chacun répondant à un point de douleur ou à une opportunité pour ses utilisateurs. Elle accélère la création de nouveaux produits digitaux et laisse de la bande passante aux équipes Produit pour se concentrer sur l'expérience utilisateur.



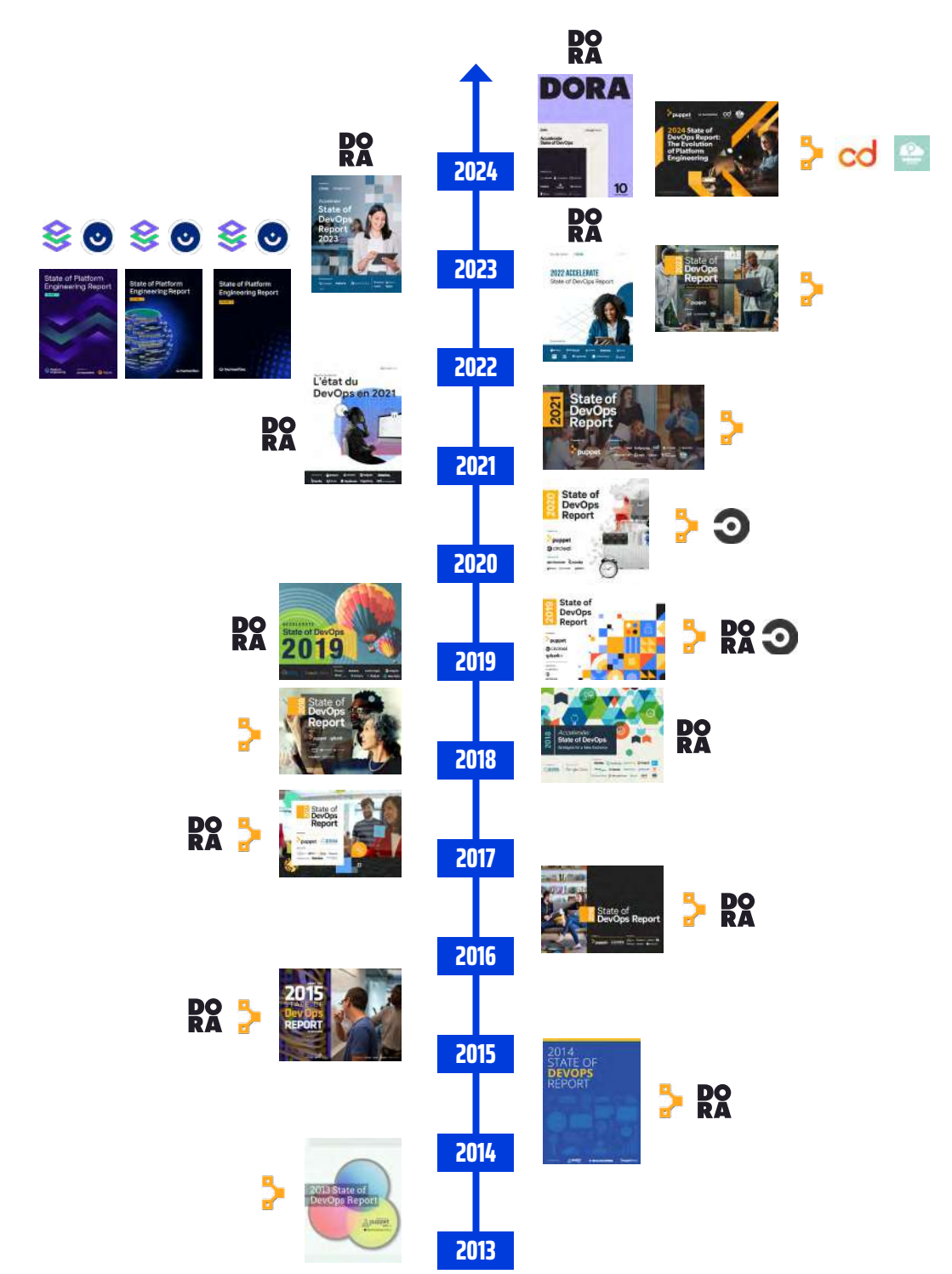
LE PLATFORM ENGINEERING S'ANCRE DANS LES HABITUDES

On aurait pu croire que le Platform Engineering allait devenir une tendance comme une autre, mais ses sponsors ont réussi à mettre en place des références qui se veulent stables ou pérennes :

— Le **site officiel de la communauté Platform Engineering** (**platformengineering.org**) peut être considéré comme une source fiable, même si de nombreux contributeurs travaillent chez Humanitec, entreprise qui vend des solutions de Platform Engineering. Ce site publie tous les ans un state of Platform Report (une déclaration de guerre à Puppet qui publie le State of DevOps Report chaque année ?)

— Les **PlatformCon** sont des événements qui ont lieu tous les ans pendant lesquels les entreprises partagent leur expérience. Ils sont portés par l'organisation Platform Engineering.

— Le ***Puppet State of DevOps Report*** et le ***Accelerate State of DevOps Report*** évoluent progressivement vers le Platform Engineering et seront encore un bon moyen de suivre ses progrès au cours du temps. Il y a également le State of Platform Engineering publié par Humanitec et l'organisation Platform Engineering depuis 2022.



Histoire des *States of DevOps Reports et State of Platform Engineering Reports*

CONCLUSION

Le Platform Engineering est un sujet d'autant plus passionnant qu'il dépasse les aspects techniques : il est nécessaire de bien connaître l'organisation des équipes IT pour comprendre pourquoi il est apparu et quels besoins il vient servir. Certains peuvent être tentés de dire que c'est le remplaçant de la philosophie DevOps, mais comme je l'ai montré dans cet ouvrage, il s'agit moins de remplacer que d'implémenter d'une façon plus simple, plus efficace et plus durable.

Quand le DevOps se limitait parfois à un outil marketing pour les décideurs, afin de vanter les performances de leurs équipes, le Platform Engineering, lui, est vraiment au service des développeurs. Il leur donne toutes les clés nécessaires pour déployer de l'infrastructure, en laissant la possibilité de placer des garde-fous sur la consommation et la configuration de ces ressources.

Je suis convaincu que cette approche va se répandre au sein des entreprises et que les Digital Factories constitueront le prochain jalon dans l'organisation des équipes IT. Chez Ippon Technologies, nous y croyons et nous avons la chance d'avoir plusieurs pôles d'expertise (Cloud & DevOps, Data Engineering, Software Engineering, Product) impliqués dans la construction de notre offre Platform Engineering et nous sommes prêt à accompagner nos clients dans cette transformation.

SOURCES

— **Are you an Elite DevOps performer? Find out with the Four Keys Project**
https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance?hl=en

— **State of DevOps Reports**
- 2024 State of DevOps Report: The Evolution of Platform Engineering
- 2023 State of DevOps Report: Platform Engineering Edition
- 2020 State of DevOps Report

— **DevOps metrics**
https://www.atlassian.com/devops/frameworks/devops-metrics

— **Team topologies**
https://lucidspark.com/fr/blog/team-topologies

— **Real Stories of Platform Leadership**
https://teamtopologies.com/platform-engineering

— **What Team Structure is Right for DevOps to Flourish?**
https://web.devopstopologies.com/

— **How to build your platform engineering team**
https://platformengineering.org/blog/how-to-build-your-platform-engineering-team

— **Create your own platform engineering reference architectures**
https://platformengineering.org/blog/create-your-own-platform-engineering-reference-architectures

— **How to Build a Platform Engineering Team [Focus & Roles]**
https://spacelift.io/blog/how-to-build-a-platform-engineering-team

— **Product Manager VS Product Owner : Quelles différences ?**
https://www.hetic.net/product-manager-vs-product-owner-quelles-differences

— **Getting Started with Backstage: From Zero to Operational Dev Portal - DevOps Toolkit**
https://www.youtube.com/watch?v=A-3Ai--Z-Gs

RÉÉCRITURE ET ADAPTATION

Florence BOULENGER

MISE EN PAGE

Alice DEROYE

CRÉDITS PHOTOS

Photo de Ron Lach - Couverture
Photos François Delauney - pages 7, 28, 45
Photo de Serpstat - page 14
Photo de Thirdman - page 17
Photo de Alena Darmel - page 24
Photo de Paulo gustavo Modesto - page 36
Photo de Christina Morillo - page 40
Photo de Kaboompics.com - page 49
Photo de Mizuno K - page 51
Photo de Google DeepMind - page 56
Photo de Serpstat - page 62
Photo de fauxels - page 63
Photo de Kerim Isazade - page 70

ippon

CONTACTEZ-NOUS !

fr.ippon.tech
blog.ippon.fr

contact@ippon.fr
+33 1 46 12 48 48
@ippontech