MEMOIRE DE FIN D'ETUDE MASTER 2

Campus Academy Titre de niveau I
CPCSI 2018-2020

Devops vs Legacy

« Comment des choix techniques, à court terme liés à des fortes contraintes, imposent-ils une réappropriation et une homogénéisation d'un SI pour migrer vers une vision long terme ? »

Entreprise : Financières des Paiement Electroniques

Tuteur : Olivier Chanteloup

Thibaut FONTAINE
2018- 2020

MEMOIRE DE FIN D'ETUDE MASTER 2

Campus Academy Titre de niveau I
CPCSI 2018-2020

Legacy & Devops

« Comment des choix techniques, à court terme liés à des fortes contraintes, imposent-ils une réappropriation et une homogénéisation d'un SI pour migrer vers une vision long terme ? »

Entreprise : Financières des Paiement Electroniques

Tuteur : Olivier Chanteloup

Thibaut FONTAINE
2018- 2020

REMERCIEMENTS

Je tiens tout d'abord à remercier particulièrement Yvan Karmouta en sa qualité de Devops, qui m'a soutenu durant ces deux années et m'a permis de monter en compétences sur les sujets liés à la culture devops.

Je remercie également mon tuteur, Olivier Chanteloup, Manager IT mais aussi Edgar Navarre Manager Infrastructure pour m'avoir encadré, de m'avoir permis de me responsabiliser dans mes différentes missions, tout en m'accordant leur confiance dans la réalisation de ces dernières.

Je remercie Jonathan Kulitza, Devops et Zurabi Goginashvili, en tant que chef de projet, pour leur collaboration et leurs conseils qui m'ont été très précieux.
Je suis reconnaissant envers l'équipe Sécurité, avec qui j'ai été amené à travailler, échanger et avec qui j'ai pu aborder des sujets liés à la sécurité.

Sans oublier mes remerciements à Nicolas et Kevin pour leur collaboration et leur soutien au cours de cette année.
Enfin, un grand merci aux équipes de développement Nickel, trop nombreux pour tous les citer, qui m'ont permis d'appréhender des notions de développement.

PLAN MEMOIRE

INTRODUCTION

PRESENTATION DE L'ENTREPRISE

Néo-banque en pleine croissance, Nickel est un modèle disruptif de comptes sans banque, diffusés dans des bureaux de tabac. Avec plus de 30 000 ouvertures de comptes par mois, Nickel se positionne comme une véritable alternative aux banques traditionnelles et une fintech incontournable depuis sa création en 2014. FPE a été racheté par la BNP en 2017. Ce rachat a pour but de consolider le développement de la marque Nickel tout en gardant notre indépendance vis-à-vis de la BNP. L'entreprise compte actuellement 400 salariés, avec un programme de recrutement de 100 personnes sur 2020 et 2021.En 2019 FPE a connu une croissance de +80% des effectifs en 2019 et 55 mobilités internes au 1er semestre 2020.

L'entreprise est répartie sur deux sites au niveau national, Nantes ainsi que Charenton (94) qui est le siège social de l'entreprise et également à l'international à Madrid en Espagne.

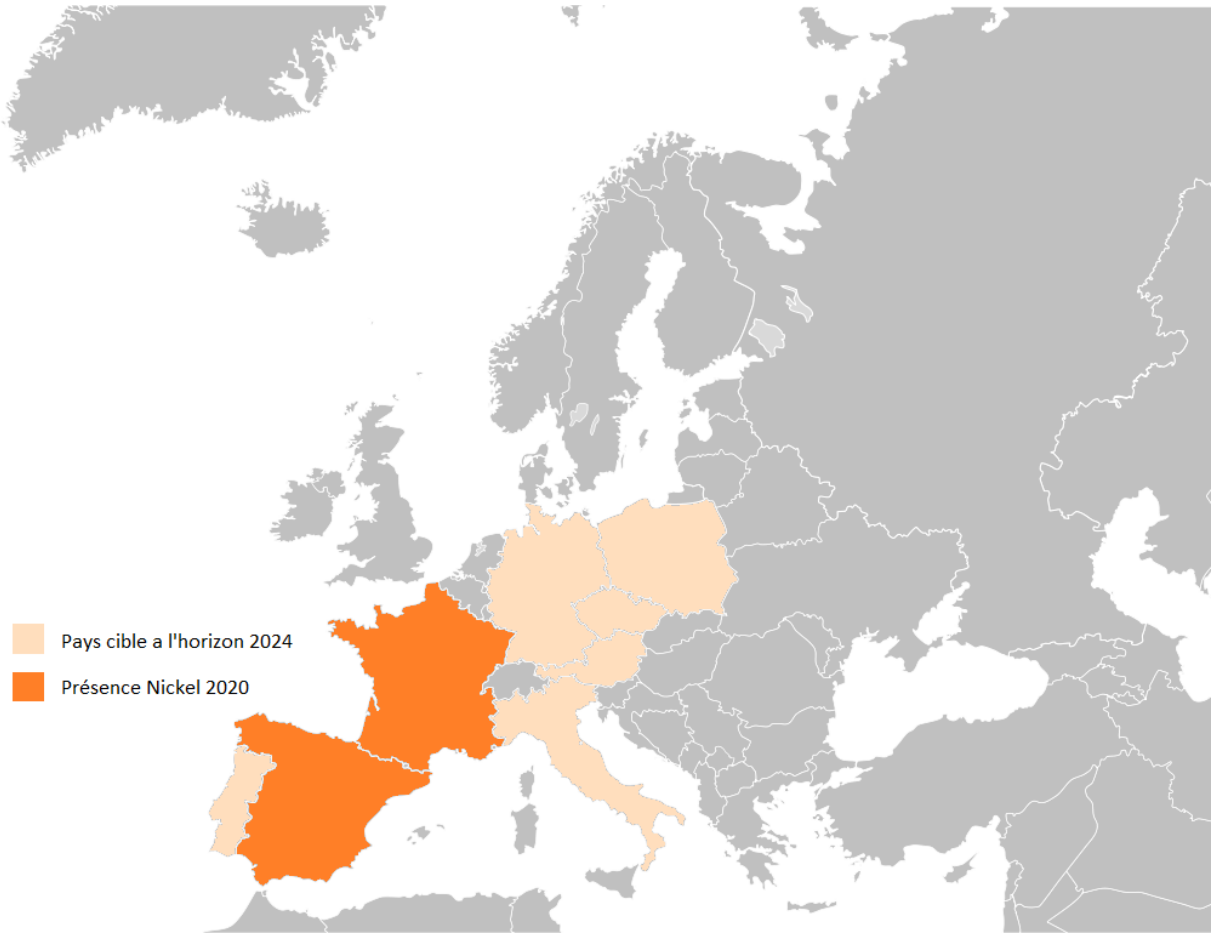


Figure 1 - Présence de Nickel en Europe

Présence de Nickel en Europe

Le site de Nantes représente environ 300 personnes réparties dans plusieurs services :

- Service client
- Conformité
- IT

Le site de Charenton compte environ 150 personnes réparties dans plusieurs services :

- Finance
- Ressources humaines
- Métier
- Risques
- Commercial
- IT[1]
- Administratif
- Marketing

Fin 2019, Nickel crée une équipe internationale, pour préparer le lancement dans de nouveaux pays à l'horizon de 2024. Dans un premier temps, c'est l'Espagne qui est choisi.

[1] Informtation Technology

MA PLACE DANS LA SOCIÉTÉ

J'ai intégré Nickel le 11 septembre 2017 au sein d'une équipe d'une équipe de 5 personnes réparties sur les sites de Charenton et Nantes. Celle-ci était composée de Yann Wallez, directeur de l'exploitation, Jan Yildiz, chargé des contrôles des opérations et support N2, Edgar Navarre, Administrateurs réseaux et systèmes et Matthieu Elazare technicien informatique. Ma première année d'alternance s'articule essentiellement sur des tâches d'assistance aux utilisateurs ainsi que l'administration de l'infrastructure interne. Ces tâches incluent la gestion du parc et éléments réseaux, mais également des sujets de sécurité. Mon travail au sein de Nickel m'a permis d'obtenir mon titre de niveau II d'administrateur réseaux et systèmes, en travaillant à l'époque sur le projet de redondance des firewalls internes (équipements réseaux).

À la fin de ma première année, j'ai décidé de m'orienter sur un master chef de projet informatique pour compléter mon cursus tout en restant chez Nickel. Pourquoi avoir fait ce choix ? Ayant commencé à apporter ma pierre à l'édifice, je souhaitais naturellement continuer ce que j'avais commencé à accomplir.

Mon année master 1 j'ai eu la chance de découvrir les concepts Devops grâce au recrutement d' Yvan Karmouta au sein de mon équipe. Il m'a énormément appris dans la manière de travailler dans un environnement IT complexe, tout en ayant une approche très pédagogique, ce qui m'a permis de monter rapidement en compétence.

Depuis, j'évolue sur des tâches d'administration systèmes que j'automatise et simplifie au quotidien, j'interviens sur le périmètre infrastructure Datacenter et Cloud.

INTRODUCTION

Nickel est fondé en 2014, à cette époque le SI était géré par dix personnes, essentiellement développeurs qui ont travaillé auparavant chez ABM technologies[2]. ABM technologies est devenu FPE [3]qui a développé le produit Compte-Nickel. Le système d'information historique de Nickel est hébergé chez Rackspace. Il est constitué de deux briques principales : Base de données (SQL[4]) et Web (IIS[5]) l'un gérant les opérations clients et l'autre permettant la gestion des comptes clients ainsi que la mise à disposition des données aux différents prestataires via des services web. La communication entre les terminaux de paiement (TPE) et le SI Nickel est gérée via la solution monétique STAP[6].

À ce moment-là, Nickel n'ayant pas les ressources internes nécessaires, la société fait le choix de sous-traiter les autres composants du SI dont elle n'avait pas l'expertise. Le core-banking, qui est chargé du "stockage" de l'argent, était porté par la société SAB. La gestion des flux bancaires (l'ensemble des transactions cartes bancaires) est détenue par la société Monext et en lien avec Mastercard.

Cela ne dura qu'un temps car suite au succès de Nickel, il devenait important de faire évoluer le SI afin de répondre à une demande croissante, d'apporter innovation et disruption dans le marché bancaire. Il est devenu évident de créer son propre "core banking[7]", avec la volonté de créer un système bancaire permettant des transactions en temps réel et ne plus utiliser le système de traitement fournis par SAB.

En 2017, une équipe de développeurs est formée : Cobalt (Core Banking Live Transactions). Elle se consacre uniquement au développement d'un core banking temps réel. Cette équipe de développeurs avait besoin d'un environnement technique souple et autonome afin de pouvoir développer rapidement un produit viable. Le choix s'est tourné vers Clever Cloud, un hébergeur Nantais, proposant une solution Paas [8]pour les développeurs et répondant aux contraintes technologiques modernes. Le core banking est basé sur des technologies innovantes: Scala(Langage), Cassandra(Bases de données), Kafka[9](Messaging) capable de tenir la charge de l'arrivée des nouveaux clients.

[2] Société de service informatique
[3] Financières des Paiements Electroniques
[4] Langage de programmation utilisé pour les bases de données
[5] Serveur Web des différents systèmes d'exploitation Windows
[6] Solution de développée par la société AFSOL
[7] Système informatique qui gère toutes les transactions financières des clients
[8] Plateform as a Service
[9] Technologies informatique

En parallèle, de la création de l'équipe Cobalt, est née l'équipe Spartacus (SPARTaCUS stands for PARTNERS And CUSTOMERS) afin de moderniser l'utilisation du produit Nickel et améliorer l'expérience utilisateur. Ils sont en charge de l'application mobile et de la brique frontend. Les briques applicatives reposent sur des technologies différentes open source comme Java, et propriétaires comme IIS et Microsoft SQL.

Le projet de modernisation de la brique frontend [10]est également hébergé chez Clever Cloud. Seules les briques propriétaires sont hébergées chez Rackspace.

En 2018, afin de moderniser l'infrastructure existante, il était nécessaire de monter des environnements de pré-production et de recette au plus proche de l'infrastructure historique sur lesquelles nous étions pleinement autonomes. Le choix s'est porté sur Google Cloud Platform pour répondre à ces besoins.

Sur GCP [11]différents environnements ont été mis en place:

Un environnement de production Frontend et Backend [12]pour répartir la charge liée à la forte sollicitation des systèmes.

Un environnement de pré-production avec les briques techniques MSSQL [13]et IIS.

Un environnement de recette pour les tests fonctionnels des applications d'une équipe de développeurs.

Avec l'arrivée de cet hébergeur, une équipe de développeurs est née, l'équipe Kitt[14], dont le projet est la modernisation de l'outil interne Back Office permettant la gestion des comptes clients pour les conseillers clients. Pour se faire, elle utilise la technologie cloud proposée par GCP et implémente des méthodes Devops.

Durant la même année, BNP Paribas rachetait Compte-Nickel, ce rachat amenait de nouvelles exigences de sécurité ainsi qu'un audit de la part de l'inspection générale du groupe (IG BNP) afin de mettre en évidence les points à améliorer.

Avoir 3 hébergeurs proposant des services de différents types, "Infogérance chez Rackspace, Cloud chez Google, PAAS chez clever-cloud", dans une entreprise en très forte croissance, mettait en évidence la difficulté d'avoir une "architecture cohérente" dans le contexte de croissance qu'a vécu FPE. Chaque hébergeur répond à des besoins spécifiques pour une équipe dédiée. L'équipe infrastructure était à cette époque composée de 3 personnes et l'hétérogénéité du SI rendait son administration beaucoup plus difficile.

[10] Brique d'une application, la partie émergée de l'iceberg (visible par le client)
[11] Google Cloud Platform
[12] Brique d'une application, la partie submergée de l'iceberg (non visible par le client)
[13] Microsoft Sql Server
[14] KITT is the Internal Tool Team

Chaque hébergeur avait sa propre manière d'être administré. L'infrastructure rackspace est en ½ infogérance, toutes les modifications, réseau, hardware, se faisaient via un ticket. Nous avions la main uniquement sur les systèmes d'exploitation des machines virtuelles Windows. Les applications hébergées chez Clever Cloud étaient gérées entièrement par les développeurs, du développement à la mise en production. L'environnement GCP était entièrement administré par l'équipe infrastructure, nous gérions toutes les couches (réseaux, firewalling,systèmes) sauf la partie applicative.
Le fait d'avoir un SI hétérogène, ne permettait pas à l'équipe infrastructure de la maîtriser de bout en bout, cela difficultés pour identifier de manière précise les origines d'incident technique.
Face à une telle croissance du SI et aux objectifs que FPE s'impose, il est intéressant d'avoir un point d'attention autour de l'aspect humain au sein de la société.

La santé financière de l'entreprise est bonne et en pleine croissance, mais la gestion financière des coûts informatiques n'était pas un sujet jugé prioritaire dans les projets informatiques. La gestion des coûts n'a pas été correctement conduite "faute de temps" et d'investissement sur le sujet.
La gestion des coûts d'infrastructure n'était pas un critère durant la croissance de l'entreprise.
Chaque hébergeur a son propre système de paiement.
Rackspace, abonnements mensuels et facturation supplémentaire à chaque opération de maintenance. (Extension de stockage sur VM, etc)
Il était impossible d'émettre des demandes de changement ou d'ajout sur l'environnement Rackspace car chaque opération technique était beaucoup trop onéreuse, sauf cas exceptionnel sur les serveurs de production SQL.
Clever Cloud étant un hébergeur cloud, la facturation se fait à la consommation de ressources.
Google Cloud Platform facture également à la consommation de ressources dans le cloud.
Au début, Google Cloud Platform était peu utilisé. Mais au fur et à mesure, les demandes des développeurs s'accumulaient pour avoir accès à des ressources sur l'environnement GCP...
Jusqu'au moment où la direction technique s'est rendue compte qu'en ajoutant toujours plus de nouvelles briques (machines virtuelles, liaison VPN[15],etc) sur GCP, la facture s'alourdissait chaque mois. C'est à ce moment-là que la direction a mis en place un nouveau process de validation pour chaque nouvelle demande émise par une équipe IT.
Chaque demande d'ajout ou de modification de l'infrastructure cloud était soumise à validation écrite et tracée dans un fichier, afin de pouvoir évaluer les coûts. Ce nouveau processus a permis à la direction une prise de conscience des coûts engendrés par l'utilisation du cloud en self-service.

[15] Virtual Private Network

Ce problème de mauvaise gestion des coûts a mis en évidence une problématique du cloud en général. Aujourd'hui, le système de facturation proposé par les hébergeurs cloud est basé, pour la plupart, sur un système de paiement à la consommation de ressources. Ce système comporte des avantages, une meilleure gestion des coûts sur l'infrastructure d'une application ou d'une infrastructure globale mais a contrario, nous pouvons vite se retrouver en surconsommation s'il n'y a pas une solution de cost management[16] mise en place. Il existe aujourd'hui des outils open source comme Komiser[17], permettant de monitorer les coûts des différentes ressources utilisées dans le cloud. L'outil est compatible avec les hébergeurs du GAFAM ainsi qu'OVH et Scaleway.
Certains hébergeurs proposent également leur propre outil, comme par exemple la plateforme Azure de microsoft. L'idée qu'avait la direction, visant à dire que le cloud est moins chère, était vraie sur le papier, mais à condition de consommer des services cloud sur un laps de temps et non pas pour monter un SI complet comme pour l'infrastructure interne.
En parallèle de la pleine croissance du produit Nickel, l'entreprise ne cesse de croître en terme d'effectifs : plus de 200 recrutements et 50 mobilités internes 2019. Cela a engendré d'énormes changements dans l'organisation et la restructuration au sein des différentes directions.
Ces différents changements ont amené de nouvelles problématiques, notamment dans la gestion des connaissances et la capacité à capitaliser les connaissances de chacun au sein de la société.
Une des grandes problématiques à laquelle Nickel est confronté à l'heure actuelle est la capitalisation des connaissances. En effet Nickel étant une startup qui est vite devenue une PME, le partage de connaissances est devenu un enjeu très important.
La société ayant connu une forte croissance en quelque mois, les équipes techniques n'ont pas eu le temps de partager les connaissances auprès des nouveaux collaborateurs fraîchement arrivés dans la société, ce qui pose problème à chaque départ en congés ou lors de l'absence d'une personne ayant un minimum d'ancienneté.
S'il y avait un problème technique grave dont la personne d'astreinte ne pouvait pas résoudre, elle était contrainte d'appeler la(les) personne(s) concerné(es).
Cela pouvait avoir un impact énorme si la personne ayant les connaissances n'était pas joignable à temps, car selon le problème technique, cela pouvait avoir un impact fort pour les clients comme l'impossibilité de retirer de l'argent ou bien d'accéder à son espace client.

[16] Management des couts
[17] voir webographie

PROBLEMATIQUE

Dans chaque PME, dont le produit ou service proposé repose sur l'IT, tout en ayant une croissance exponentielle, vient le moment où pour faire face à cette croissance, on rencontre des contraintes plus ou moins fortes, des difficultés à assurer le même niveau de qualité de service pour un million de clients que pour cent milles. Il est nécessaire de faire des choix importants pour ne pas accumuler de la dette technique et permettre de continuer à croître. En fonction des choix techniques qui vont être faits et de la prise en considération de l'ensemble des contraintes, les conséquences seront majeures.

Si le système d'information n'est plus capable de tenir la charge engendrée par la croissance du produit ou du service rendu aux utilisateurs, il devient urgent de remettre en question les moyens capacitaires que l'entreprise met en œuvre et de s'adapter pour l'avenir du produit.

Il y a deux choses importantes à souligner dans cet exemple.

Premièrement, la notion de capacité à tenir la charge. Pour avoir cette information, cela veut dire que l'on est capable de mesurer cette capacité et donc de l'anticiper.

Si l'information n'est pas anticipée, c'est pour deux raisons :

- Le choix hiérarchique de pousser le système jusqu'à ses limites car il n'y a pas les ressources nécessaires en interne pour intervenir.
- La non-écoute des alertes levées par les équipes techniques aux personnes décideuses.

Ce qui amène à la deuxième notion, celle de l'urgence. Cette notion est importante car elle va amener le personnel décideur à prendre une décision sur le court terme pour résoudre une problématique sans réfléchir au long terme.

Cela m'amène à la réflexion suivante, si chaque choix technique a été pensé pour répondre à une problématique sur le court terme, afin d'avoir une amélioration immédiate, il ne fait que reporter le problème à plus tard. La problématique suivante apparaît donc, **comment des choix techniques, à court terme liés à des fortes contraintes, imposent-ils une réappropriation et une homogénéisation d'un SI pour migrer vers une vision long terme ?**

L'objectif sera donc de définir les choix techniques, humains et les actions à mener, que ce soit immédiat ou sur une longue période, pour se réapproprier et uniformiser ce qui a été mis en place ces six dernières années.

PARTIE I

COMMENT PEUT-ON SE REAPPROPRIER SON SI ET POURQUOI HOMOGENEISER SON SYSTEME D'INFORMATION ?

I) – COMMENT PEUT-ON SE REAPPROPRIER SON SI ET POURQUOI HOMOGENEISER SON SYSTEME D'INFORMATION

LE SYSTEME D'INFORMATION NICKEL

LES DIFFERENTES COMPOSANTES

COMPOSANTE INTERNE

COMPOSANTE ORIENTE CLIENT

MOYEN ORGANISATIONNEL

COMMENT SAVOIR SI NOUS NE MAITRISONS PLUS NOTRE SI ?

QUELS SONT LES RISQUES AUXQUELS NOUS SOMMES EXPOSES ?

LES MOYENS MIS EN OEUVRE POUR UNIFORMISER LE SYSTEME D'INFORMATION

UNIFORMISATION DE L'INFRASTRUCTURE INTERNE

UNIFORMISATION DE L'INFRASTRUCTURE ORIENTE CLIENT

MAITRISER SA GESTION DE PROJET

LES ETAPES CLES POUR HOMOGENEISER

Aujourd'hui le système d'information est un élément vital d'une société. Il est d'autant plus important quand le business model est basé sur un service 100% numérique.

Le système d'information regroupe l'ensemble des moyens humains, matériels, logiciels, permettant de créer, traiter, stocker et délivrer une information grâce à des processus ou services.

Cet ensemble de moyens et de processus, permet à une entreprise de produire de la valeur ajoutée en proposant des services à ses clients. Chaque moyen mis en œuvre couvre un périmètre défini.

Le SI est composé de plusieurs catégories :

Les moyens organisationnels, regroupent les processus organisationnels des équipes techniques permettant de faire fonctionner le système d'information.

Les moyens techniques regroupent l'ensemble des outils informatiques permettant de traiter et stocker l'information ainsi que la structure hébergeant le système d'information, allant du choix de l'infrastructure au choix des technologies utilisées pour les briques logicielles. Ces moyens techniques sont mis en œuvre afin d'optimiser le fonctionnement du SI dans sa globalité.

LE SYSTEME D'INFORMATION NICKEL

Le SI chez Nickel est en constante évolution pour répondre au mieux aux besoins futurs. Nous pouvons découper les moyens techniques mis en œuvre au sein de Nickel en deux composantes :

- Une composante répondant au besoin interne qui a dû évoluer subitement pour être en capacité à suivre la croissance de l'entreprise, regroupant l'ensemble de l'infrastructure locale des différents sites.
- L'autre composante, quant à elle, répond au besoin des clients Nickel, qui évolue tout au long de la croissance de l'entreprise, brique par brique. Ces évolutions se sont faites au fur et à mesure que des nouveaux besoins se faisaient ressentir.

LA COMPOSANTE INTERNE

L'infrastructure locale a subi des changements drastiques durant les trois dernières années. Lors de mon arrivée dans l'entreprise, l'infrastructure n'était pas hautement disponible, mais adaptée au besoin initial que l'on retrouve souvent dans les TPE [18]ou dans de petites structures comme les

[18] Très petite entreprise

startup. Nous avions à ce moment-là, le strict nécessaire pour faire fonctionner l'entreprise et cela suffisait amplement.

Par la suite, l'entreprise a grossi et pour suivre cette croissance, des projets ont été mis en place pour sécuriser l'infra existante. Mon projet de licence faisait partie des projets de sécurisation de l'infrastructure réseaux, il portait sur la mise en place d'une redondance des équipements réseaux du site de Nantes.

Le site de Nantes est composé de quatre VLAN:

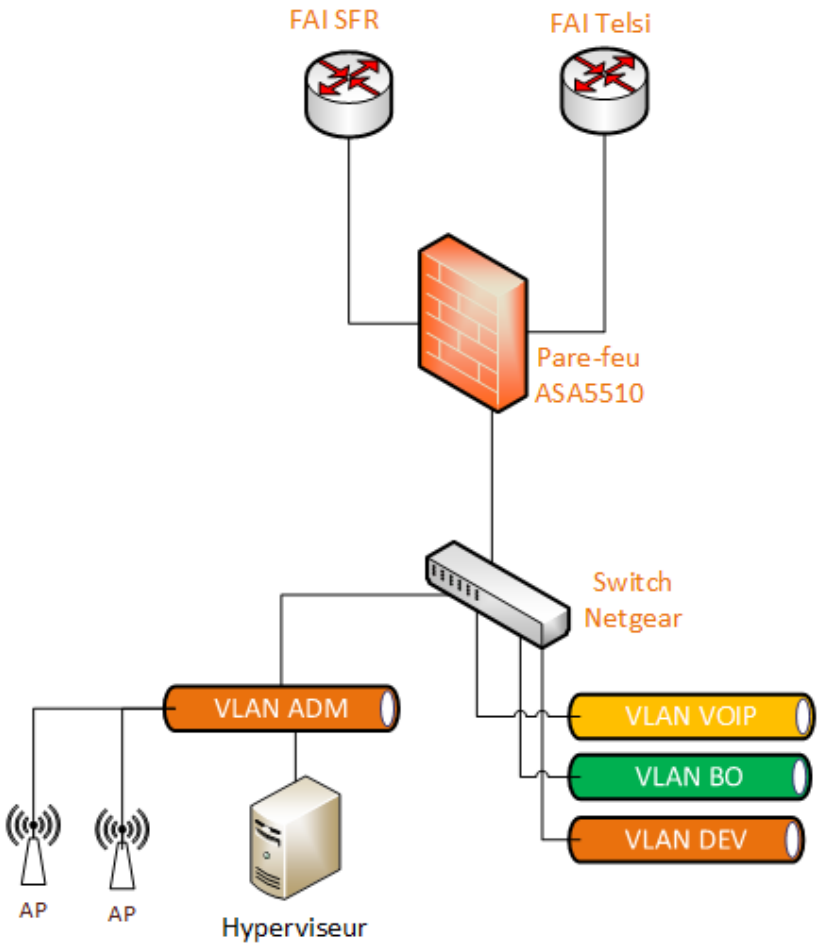


Figure 2 -Schéma réseau Nickel local

Un VLAN ADM, on l'on retrouve les divers équipements réseaux et serveurs internes.

Un VLAN VOIP, pour les téléphones physiques.

Un VLAN BO pour back office, ce sous-réseau étant dédié aux personnes du service client et conformité.

Un VLAN DEV, réservé pour les collaborateurs de l'IT avec des accès privilégiés pour accéder aux diverses ressources chez nos hébergeurs.

Ce type d'architecture commençait à montrer ses limites en terme d'évolution et de sécurité informatique, pour cela nous l'avons repensé dans son intégralité, afin de construire une architecture hautement disponible.

Nickel a fait le choix pour son service de messagerie d'utiliser la solution Google (G Suite[19]) proposant une suite de logiciels bureautiques, une messagerie personnelle et un espace de stockage dans le cloud pour chaque collaborateur. La solution G Suite répondait efficacement aux besoins de l'époque et encore aujourd'hui sur divers sujets sécurité, coûts, performances et facilités d'administration car a il n'y avait historiquement personne dans l'équipe infrastructure interne.

[19] Suite de logiciel bureautique Google

LA COMPOSANTE EXTERNE

L'infrastructure Datacenter est l'ensemble des composants permettant de fournir le service bancaire Nickel aux clients.

C'est l'infra de production comprenant tout le legacy[20], il était donc difficile d'apporter des améliorations permettant de faire évoluer celle-ci.

Elle est répartie au sein de plusieurs hébergeurs, Rackspace, Google Cloud Platform et Clever Cloud.

L'INFRASTRUCTURE RACKSPACE :

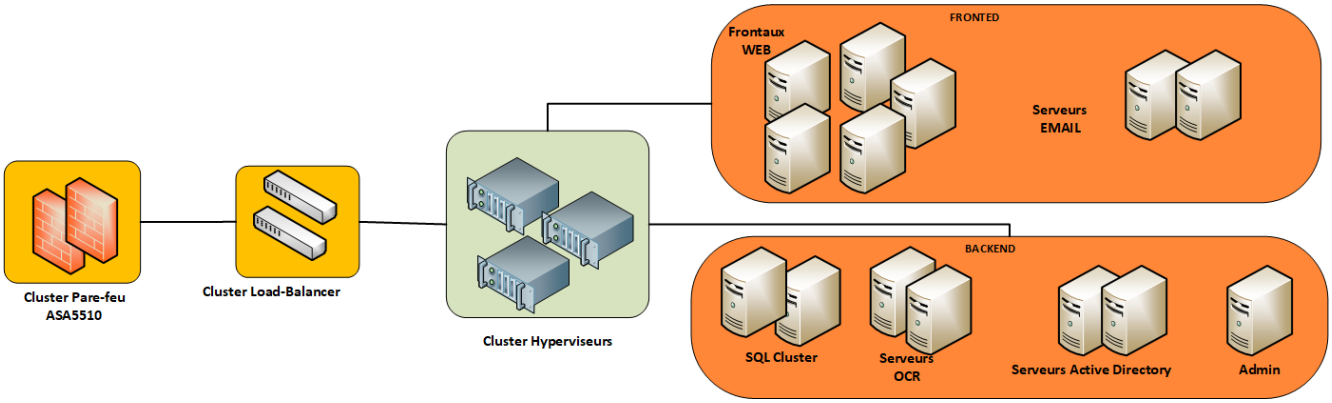


Figure 3 - Schéma infrastructure Rackspace

L'infrastructure Rackspace est un environnement exclusivement Microsoft à l'exception de la solution STAP qui elle, se trouve sous le système d'exploitation Linux. Cet environnement fait partie du legacy de l'entreprise.

Elle est composée de trois parties :

- une frontend
- une backend
- composante monétique.

- Le frontend :

Il est composé de cinq frontaux web IIS, hébergeant l'espace client, l'outil back office ainsi que divers web services permettant de se connecter et transférer des données avec nos partenaires externes.

Les frontaux web ont différentes versions du système d'exploitation, Windows server 2008 et Windows server 2012.

[20] Un système hérité continuant d'être utilisé dans une organisation, alors qu'il est supplanté par des systèmes plus modernes

Nous retrouvons également deux serveurs de mails sous Windows server 2008 pour contacter nos clients.

- Le backend :

Un cluster de serveurs physiques Windows server 2008 SQL eux-mêmes connectés à un SAN stockant l'ensemble des données clientes ainsi que les règles métier.
Deux serveurs OCR[21] permettant de récupérer les informations du client, par le scan des documents et la reconnaissance des pièces justificatives, lors de l'ouverture du compte bancaire.
Deux serveurs Active Directory (AD) pour la gestion des accès aux différents serveurs.
Ils hébergeaient également un logiciel de transfert de fichiers permettant l'envoi et la réception entre nos serveurs et ceux de nos prestataires notamment pour l'actualisation des soldes clients.
Un serveur administrateur, servant de rebond pour accéder aux serveurs hébergés chez Rackspace, il hébergeait également une solution de supervision.
La composante monétique est constituée uniquement du serveur linux hébergeant la solution monétique STAP.

Cette infrastructure Rackspace est vieillissante, 80% des serveurs sont sur la version Windows Server 2008 (la fin du support Microsoft a été annoncé le 14 janvier 2020). L'architecture a été construite courant 2012, pendant le développement du service bancaire compte-nickel. A cette époque, nous étions limités technologiquement (cf annexe date de sortie Windows Server). L'infrastructure avait été conçue pour gérer cent-mille clients mais avec le temps, elle a été capable de supporter un million de clients. Les différences de versions, que ce soit la version des framework .NET[22] ou la version SQL Server utilisée dans le développement de Nickel, sont contraignantes dans le temps car elle sont difficilement gérables et participent à engendrer un endettement technique qui sera difficile par la suite à payer.
L'infrastructure Rackspace était infogérée, nous n'avions pas la main sur les hyperviseurs, ce qui nous pénalisait lorsque nous voulions faire des modifications de configuration matérielle ou déployer de nouveaux serveurs. Il fallait alors passer par le support technique pour effectuer une manipulation. Nous perdions en autonomie, cela posait problème dans le cadre d'incidents techniques.

[21] Reconnaissance optique de caractères
[22] Langage de programmation propriétaire Microsoft

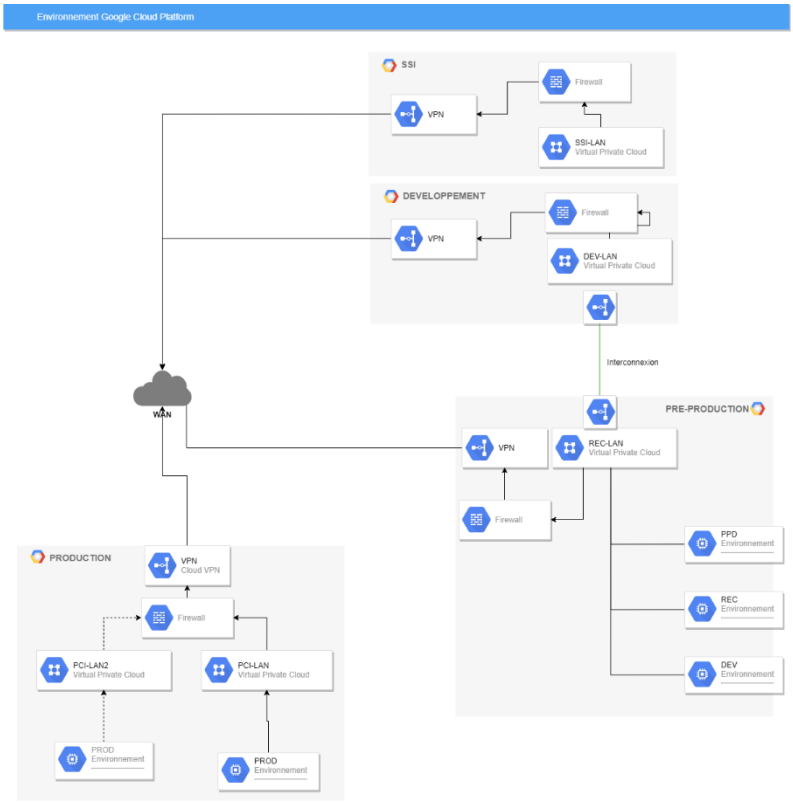


Figure 4 - Schéma Infrastructure Google Cloud Nickel

L'infrastructure Google Cloud Platform

L'infrastructure GCP a été construite début 2016 dont le but était de dupliquer l'environnement de production hébergé chez Rackspace et également d'héberger le site vitrine. Nous avions alors entièrement la main d'un point de vue technique et avions la possibilité de modifier la configuration comme bon nous semblait.

L'environnement GCP est découpé par projet, définissant chacun un environnement :

- Le projet PCI-DSS[23] dont le nom fait référence à la norme de sécurité de l'industrie des cartes de paiement. Il contient un environnement de production complet, SQL server, Serveur Web IIS, OCR, Serveur de transfert de fichiers.
- Le projet compte nickel-recette, qui a été construit pour l'environnement de recette, l'environnement de pré-production et par la suite pour l'environnement de développement.

[23] Payment Card Industry Data Security Standard

- Le projet de FPE-SSI, hébergeant des applications de sécurité informatique, et le projet fpe-dev regroupant l'ensemble des runners [24]gitlab pour les phases de Continuous Integration des équipes de développement.

Le choix de ce fournisseur cloud, n'a pas été fait avec une vision long terme mais une vision court terme. Il a été choisi à l'époque principalement pour palier à des questions de budget et de facilité d'utilisation. Google Cloud Platform était le moins coûteux du marché à cet instant. En soit l'approche était bonne, cela nous a permis de répondre au besoin initial, c'est à dire déployer une infrastructure de production équivalente à Rackspace, mais aussi de répondre aux différents besoins court termes. Nous avons pu faire évoluer l'infrastructure en construisant différents environnements de recette et pré-production afin d'améliorer le service bancaire et de gagner en autonomie.
Selon moi, ce choix n'a pas pris en compte certains paramètres importants. Nous n'avions pas une assez bonne connaissance du cloud à tous les niveaux, que ce soient les personnes décideuses ou les équipes techniques. Cela a engendré une mauvaise utilisation du cloud, car nous n'avions pas les bonnes pratiques en la matière. Une mauvaise utilisation du cloud se répercute donc sur la facture.

CLEVER CLOUD

L'infrastructure Clever Cloud était une boîte noire et entièrement gérée par les équipes de développement, pour la simple et bonne raison que nous n'avions ni le temps, ni les ressources, ni les compétences nécessaires pour nous occuper de cette infrastructure.

Les développeurs des équipes Spartacus et Cobalt étaient en relation direct avec des experts infrastructure chez Clever Cloud. Ce contact direct a eu pour effet de mettre de côté d'autres métiers de l'IT comme par exemple les équipes de gestion des risques et de sécurité qui n'étaient pas consultées dans les cycles de développement.

[24] C'est une machine qui prend des jobs et les éxécutes

MOYENS ORGANISATIONNELLES

L'organisation des équipes IT est répartie par produit/périmètre.

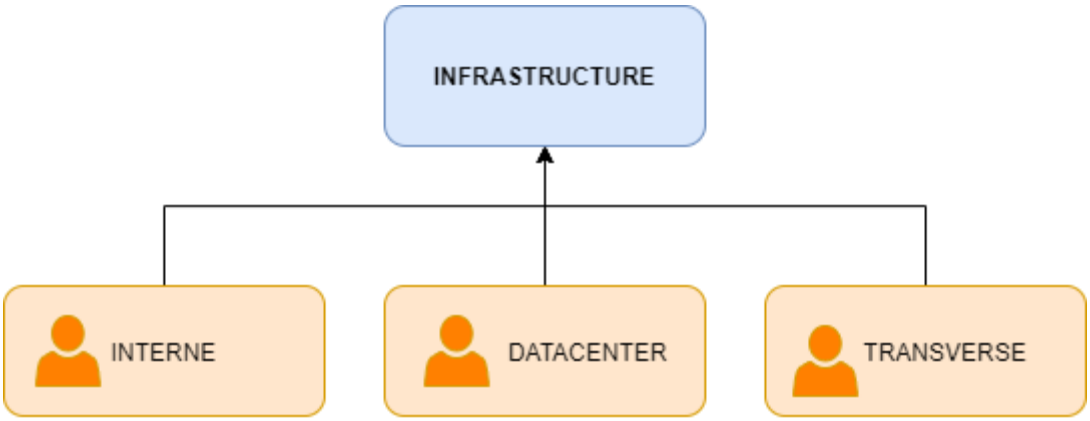


Figure 5 - Organisation Team Infrastructure

Les équipes infrastructure sont divisées en trois équipes :

Une interne

Une datacenter

Une transverse

L'équipe infra interne s'occupe du support N1, la gestion du parc interne comprenant les postes utilisateurs, les équipements réseaux et les serveurs.

L'équipe infra datacenter quant à elle, s'occupe de la gestion des serveurs et applicatifs chez les différents hébergeurs ainsi que les relations avec les différents partenaires et prestataires.

L'équipe transverse est composée d'un administrateur de bases de données, pour la gestion des bases de données des différents environnements, d'un devops et d'un chef de projet IT.

Les moyens organisationnels utilisés au sein de l'équipe infrastructure est la méthode KANBAN via l'outil Jira, un Wiki (Confluence) regroupant l'ensemble des connaissances SI et processus.

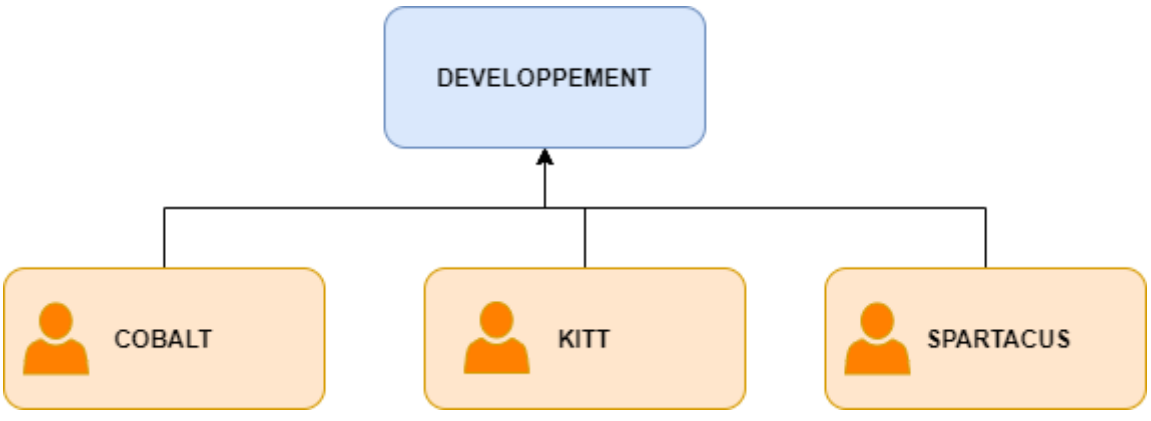


Figure 6 - Organisation Team Dev

Les équipes de développement sont divisées en 3 équipes gérant chacune leurs produits.
Cobalt pour la partie Core Banking, Kitt pour le développement des outils internes et Spartacus sur le périmètre des applications client (Espace client, parcours de souscription, site vitrine)
Chaque équipe de développeurs travaille en mode scrum sur des sprints allant de 2 à 3 semaines.
Les moyens humains engagés aujourd'hui dans le SI Nickel regroupent trois équipes de développeurs, deux équipes d'infrastructure, une équipe de sécurité.

COMMENT SAVOIR SI NOUS NE MAITRISONS PLUS NOTRE SI ?

Pour savoir si nous maîtrisons nôtre SI, il existe des indicateurs permettant de mesurer nos performances. MTTD [25]le temps de détection d'un incident et le MTTR [26]le temps de résolution d'un incident. Ce sont deux métriques importantes permettant de connaître notre état de maîtrise de l'infrastructure.
Mean time to detect (MTTD): temps de détection d'un incident
Le temps moyen de détection ou de découverte (MTTD) est un indicateur clé de performance (KPI[27]) pour la gestion des incidents informatiques et fait référence au temps moyen nécessaire pour détecter un problème. Il mesure la période entre le début d'une panne d'un système, d'un dysfonctionnement de service, et le temps qu'il faut à l'équipe d'exploitation pour identifier ce problème.
Le calcul est simple. Pour ce faire, il suffit d'examiner le nombre total d'incidents au cours d'une période et de calculer la différence entre le début de l'incident et le moment où une alerte a été remontée.
Par exemple, si l'incident a commencé à 8h00 et que l'équipe l'a détecté à 8h15, alors le temps de détection est de 15 minutes. À partir de là, il est facile de prendre la moyenne sur une période de temps (2 semaines, 1 mois, 1 trimestre, 1 an) pour calculer le temps moyen de détection (MTTD).

Mean time to Repair(MTTR) : temps de réparation d'un incident

[25] Mean time to detect
[26] Mean time to Repair
[27] Key Performance Indicator

La MTTR est une mesure précieuse car elle permet à une équipe de développement de trouver des moyens de réduire ou d'éliminer les temps d'arrêts. Aujourd'hui, plus l'arrêt de l'activité est important, plus cela a des conséquences sur la santé d'une entreprise (impact financier, image).
Le temps moyen de résolution (MTTR) fait référence au temps nécessaire pour réparer un système défaillant. Il est également connu sous le nom de temps moyen de résolution. Il est question de mesurer le temps moyen nécessaire à une équipe technique pour rendre un système de nouveau opérationnel, ou pour déployer un correctif à chaud(*hot fix*) en production.

Mean Time Between Failures (MTBF): temps entre deux incidents
Le temps moyen entre deux défaillances (MTBF) est une mesure de fiabilité et de disponibilité. Il est utilisé pour mesurer la capacité d'un système ou d'un composant à remplir les fonctions requises dans des conditions données et pendant une durée déterminée.

Ces différentes mesures sont essentielles, en particulier pour les équipes infra qui veulent donner le meilleur d'elles-mêmes. Pourtant, l'intégration des mesures MTTD, MTTF, MTBF, MTTR et autres mesures de niveau de service dans les opérations quotidiennes d'une entreprise peut s'avérer difficile. Elles servent énormément à l'équipe d'exploitation pour comprendre et anticiper les futurs incidents.

Pour illustrer mes propos, lors d'un incident survenu au cours de l'année qui a eu pour cause de bloquer le paiement sur internet pour nos clients pendant une durée de 4h. Nous avions rencontré des difficultés pour détecter l'origine de l'incident, le temps de détection de l'incident avait été supérieur à quatre heures. Toute la difficulté lors de cette incident a été d'isoler l'origine du problème qui a pris énormément de temps et explique le temps de détection.

QUELS SONT LES RISQUES AUXQUELS NOUS SOMMES EXPOSES ?

Dans chaque projet informatique il y a des risques, que ce soient des spécifications mal formulées, les choix technologiques, budgets et délais insuffisants... Nous ne démarrons, généralement, que très rarement un projet depuis une page blanche, nous partons souvent d'une base matérielle ou applicative. À partir de ce moment-là, nous devons faire des choix, qu'ils soient intentionnels ou non, mais nous devons les faire pour gagner du temps et/ou réduire les coûts. Parfois, cela se traduit par négliger l'implémentation des bonnes pratiques en terme de développement ou d'architecture. De plus la dette technique n'est pas uniquement liée à de mauvais choix d'implémentation. Comme Mike Rother le décrit dans son livre « *Toyota Kata »,* publié en 2009 en raison du chaos et de l'entropie, les processus qui ne sont pas continuellement améliorés se dégradent également. Les organisations qui ne veulent pas accepter cette vérité, ne continuent pas seulement à souffrir de leurs problèmes actuels, mais augmentent leurs difficultés avec le temps.

Pour nous aider à gérer cette dette technique, nous pouvons nous appuyer sur la règle des 20%. La règle des 20% décrite dans le livre *Devops Handbook nous* dit qu'en consacrant 20 % de nos cycles, pour que les services de développement et d'exploitation puissent mettre en place des solutions durables aux problèmes que nous rencontrons dans notre travail quotidien, nous nous assurons alors que la dette technique n'entrave pas notre capacité à développer et à exploiter rapidement, et en toute sécurité, nos services en production.

Marty Cagan, auteur de « *Inspired : How To Create Tech Products Customers Love »*, publié en 2008, l'ouvrage de référence sur la conception et la gestion des produits, a relaté la leçon suivante :

"L'accord entre les responsables produits et les équipes techniques se présente comme suit :

L'équipe projet prend 20 % de la capacité de l'équipe au départ et la donne aux équipes techniques pour qu'elles l'utilise comme bon leur semble. Ils peuvent l'utiliser pour réécrire, réorganiser ou modifier des parties problématiques du code. Tout ce qu'ils estiment nécessaire pour éviter d'avoir à venir dire à l'équipe "nous devons nous arrêter et réécrire tout notre code". Si vous êtes vraiment en mauvaise posture aujourd'hui, vous pourriez avoir besoin de 30 % ou plus des ressources. Cependant, je deviens nerveux quand je trouve des équipes qui pensent pouvoir s'en sortir avec beaucoup moins de 20% de temps." Pour absorber la dette technique au sein des équipes, il est alors primordial qu'elle soit prise en compte dès le début des projets informatiques.

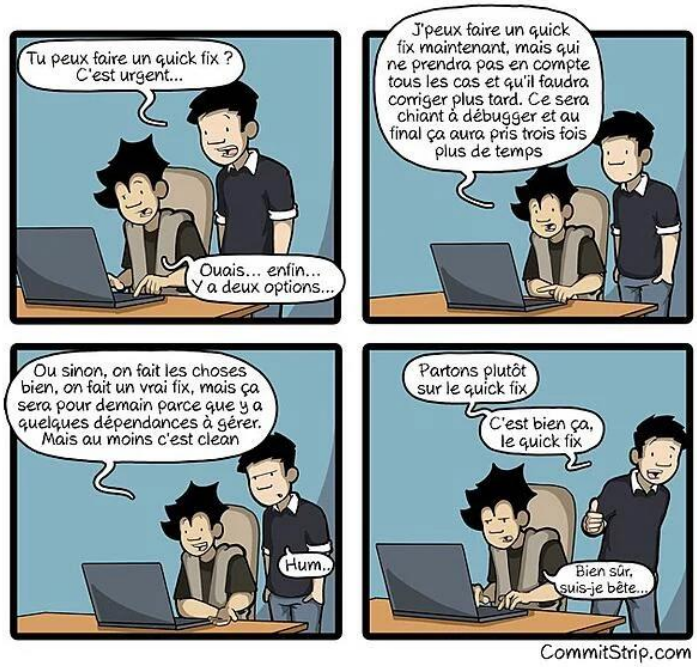


Figure 7 - Dette technique

"Nous créons parfois délibérément une dette technologique pour réduire le délai de livraison en production"

Avec le temps, qu'elle soit volontaire ou non, la dette technique grossie, d'une part par le fait que les technologies ne cessent d'évoluer d'année en année, d'autre part à chaque nouvelle livraison d'une fonctionnalité.

Sur le plan organisationnel, la perte de connaissances techniques constitue une source de l'augmentation de la complexité. Lorsqu'on veut faire évoluer un composant, on peut parfois être confronté à une méconnaissance des règles métier ou à l'empilage des différentes composantes informatiques.

Sur le plan technique, un composant applicatif doit évoluer, et son évolution est moins rapide que les sorties des nouvelles versions des frameworks ou versions des librairies développées par les éditeurs. Cela nous amène au constat suivant : un composant applicatif de qualité génère de la dette s'il n'est pas maintenu pour être en tout temps en phase avec les standards proposés par les éditeurs et les bonnes pratiques en vigueur.

La dette constitue un coût financier dans les projets pour une entreprise. Si elle n'est pas traitée, l'écart continuera de se creuser et le SI deviendra de plus en plus endetté. Il sera alors difficile d'en sortir.

Pour développer mes propos, je m'appuierai sur l'expérience d'une migration qui a eu lieu il y a quelques années.

Nous devions effectuer une montée de version des serveurs de base de données hébergés chez Rackspace qui étaient en version 2008, vers la version 2016. La migration comportait 2 parties, une montée de version du système d'exploitation et une montée de version du service MSSQL. La migration a eu lieu un week end et a mobilisé environs 15 personnes.

En avance de phase, il y a eu un découpage des tâches entre les différents acteurs et des tests des différentes briques techniques effectués sur des serveurs hébergés chez GCP pour tester la montée de version.

Les tests ont été concluants, la migration a été planifiée un week-end. Cependant lors de cette dernière durant le week-end, les choses ne se sont pas passées comme prévues (loi de Murphy), car des briques techniques qui fonctionnaient lors des tests ne fonctionnaient plus avec le reste du SI. Cela était dû aux versions des dépendances applicatives (Assembly et SSIS pour les connaisseurs) qui n'étaient pas entièrement compatibles. La migration a été avortée et reportée le mois d'après, afin de limiter l'impact utilisateurs.

La migration a eu des conséquences : elle a fortement impacté nos clients sur divers sujets. Il était impossible de faire un retour arrière et il n'était pas envisageable de finir la migration car l'impact client était trop grand. La situation a été stabilisée durant 1 mois. Le mois suivant la migration a été réalisée. Selon moi cela a montré une mauvaise gestion de notre dette technique et d'une mauvaise connaissance des dépendances associées.

LES MOYENS MIS EN OEUVRE POUR UNIFORMISER LE SI

UNIFORMISATION DE L'INFRASTRUCTURE INTERNE

Pour uniformiser le SI nous avons mis en œuvre diverses solutions. Nous avons conçu et standardisé nos infrastructures internes. D'un point de vu plus global, nous appliquons une politique de gouvernance IT suivant le plan stratégique de l'entreprise. À la suite de la croissance, nous avons profité du déménagement pour uniformiser notre infrastructure interne. Par conséquent l'architecture a été entièrement repensée par l'architecte réseaux de l'équipe infrastructure interne, Kevin Schlesser. Du fait des déménagements des sites, nous avions une plus grande marge de manœuvre pour structurer notre archi, ce qui nous a permis d'imaginer une infra modulable sur tous les niveaux pouvant s'adapter aux besoins actuels et prenant en compte nos contraintes futures, que ce soient des contraintes réglementaires, de sécurité informatique ou des processus de normalisation dans le but de faciliter des audits.

L'architecture a été pensée de façon à être modulable, c'est à dire que nous un avons un bloc d'équipements réseaux par usages spécifiques.
Un module se caractérise par une brique fonctionnelle de l'architecture du site, ce design permet d'ajouter facilement de nouvelles briques à l'architecture initiale. Cela permet d'assurer une forte croissance sans modifier l'architecture.

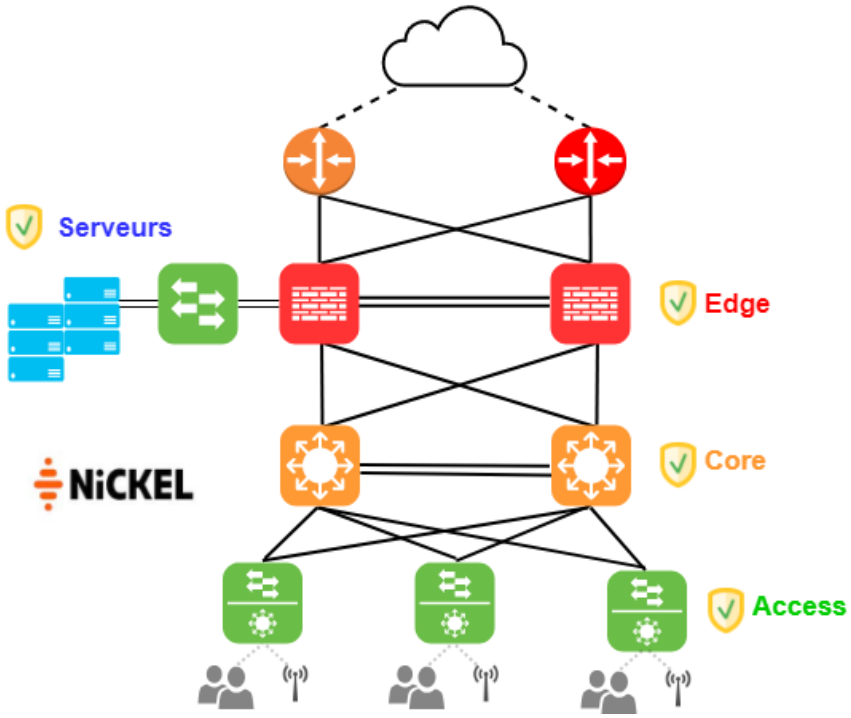


Figure 8 - Infrastructure cible Nickel

Un Cluster de pare-feu(*Edge*) Fortinet pour la gestion des règles réseaux ainsi que les liaisons VPN's avec nos différents partenaires.

Le cœur de réseau est composé de switch agregé de niveau 3 Aruba avec de l'agrégation de lien LACP entre les autres équipements réseaux.

Les switchs d'accès sont agrégés et redondés par agrégation de liens vers les switch de cœurs de réseaux.
Le WiFi est délivré par des bornes Ubiquiti et Aruba disposées à chaque étage des bâtiments.
Au niveau système, chaque site héberge des hyperviseurs[28] ESXi ainsi qu'un NAS[29] Windows Storage. Les ESX hébergent les différentes machines virtuelles hébergeant elles-mêmes différents services comme DHCP[30], Active Directory[31], Antivirus, Supervision[32], Contrôle d'accès, etc.
Les différents sites sont interconnectés entre eux via une liaison VPN ainsi qu'à Claranet notre infogéreur Cloud, hébergeant l'infrastructure datacenter.

[28] Plateforme de virtualisation
[29] Network Attached Storage
[30] Dynamic Host Configuration Protocol
[31] Service d'annuaire
[32] Permet de surveiller l'état des outils informatique

Chaque site possède les mêmes éléments techniques qu'ils soient réseaux, systèmes ou téléphoniques. Le fait d'avoir industrialisé l'infrastructure locale des différents sites nous donne la capacité aujourd'hui de pouvoir monter de nouveaux sites avec efficacité, en suivant des processus et documentations uniformisées.

Cela nous permettra de suivre les ambitions d'un développement à l'internationale à l'horizon de 2024.

UNIFORMISATION DE L'INFRASTRUCTURE ORIENTEE CLIENT

Après avoir vu les nombreuses limites de notre architecture sur trois offres datacenter trop hétérogènes, nous avons décidé de lancer un projet de migration en 2019 chez un unique hébergeur cloud (Claranet) vers une solution cloud dédié on premise[33]. Ce projet a pour objectif ambitieux d'homogénéiser l'ensemble du SI client afin de le sécuriser et d'en reprendre la maîtrise. Pour cela nous allons utiliser les technologies issues du cloud et des méthodes provenant de la culture devops. Dans cette démarche, il a été décidé de basculer notre infrastructure vers la technologie kubernetes[34]. Kubernetes va nous permettre d'homogénéiser les usines des équipes de développement et de basculer vers des technologies cloud. Je reviendrais dessus dans la parties deux

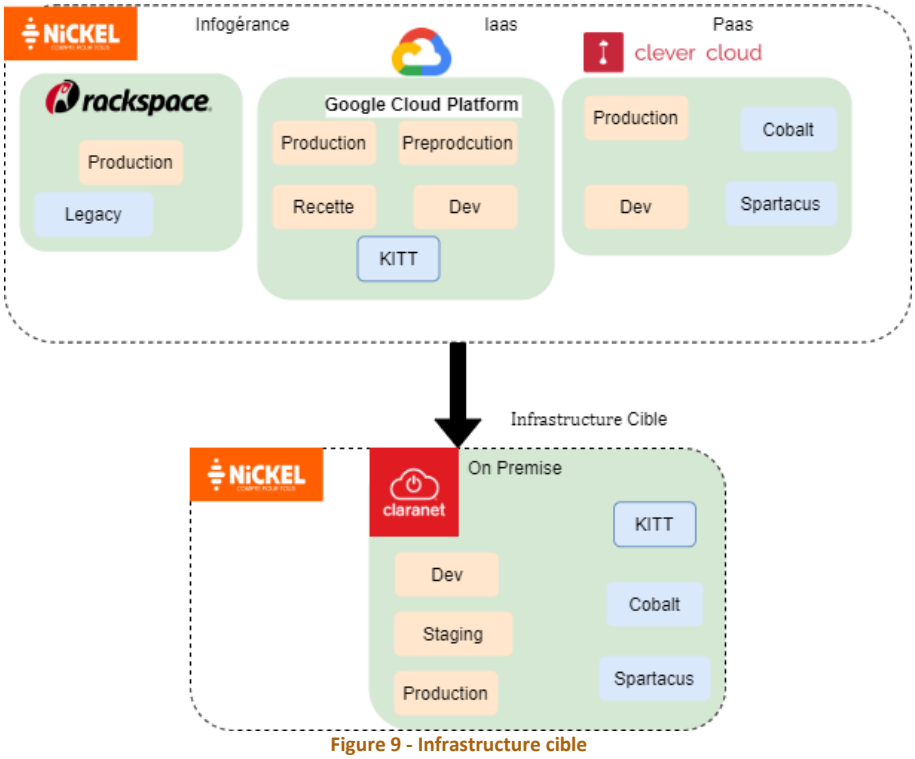


Figure 9 - Infrastructure cible

[33] Solution dédié et isolé des autres clients

[34] Système open source qui vise à fournir une plate-forme de déploiement automatisé

Le projet de migration est découpé en trois lots :

- Un premier lot englobant la partie Core Banking, l'ensemble des applications de l'équipe Cobalt
- Un lot deux regroupant les applications des équipes Spartacus
- Un lot trois réunissant l'ensemble du legacy et Kitt c'est à dire l'ensemble des serveurs hébergés chez Rackspace.

Ce projet de migration s'inscrit dans le plan stratégique de l'entreprise à l'horizon de 2024. De plus ce projet nous permet également de mettre en oeuvre la politique globale de gouvernance IT du groupe BNP Paribas qui est en cours de déploiement depuis l'audit de l'IG BNPP datant de 2019.

QUELS SONT LES ETAPES CLES POUR HOMOGENEISER ?

MAITRISER SA GESTION DE PROJET

Il est nécessaire d'avoir de la méthodologie, avoir une ouverture d'esprit, afin d'avoir une vision plus globale. Cela se traduit par de la gestion de projet bien maîtrisée qui se découpe en plusieurs phases. Nous nous appuierons sur la méthode de la Roue de Deming.
Nous avons une première phase(plan) de renseignement, être capable d'identifier chaque ressource de notre système d'information, cartographier l'ensemble des dépendances, ainsi que les actifs primordiaux à la vie de l'entreprise.
Une deuxième phase(do) de build avec des processus de développements communs, outils et technologies standardisés entre les différentes équipes de développement.
Une troisième phase de (check) validation, où nous allons vérifier (act) le bon fonctionnement des nouveaux processus de développement avant de passer à la dernière phase de livraison. Nous répétons ces différentes phases pour chaque lot projet que nous devons déployer.
Il est donc important d'avoir une bonne vision du SI, comprendre les enjeux et pousser la réflexion au-delà de ce qu'on nous demande que ce soit sur les processus de travail ou sur les choix technologiques. Pour nous aider à progresser vers une vision long terme, nous allons nous appuyer sur les méthodes issues de la culture devops et comprendre l'enjeux d'utiliser cette méthodologie de travail.

PARTIE II

AMELIORER LE SI A L'AIDE DES PROCESSUS DEVOPS COMMUN ENTRES TOUTES LES USINES DE DEVELOPPEMENT ?

II) – COMMENT PEUT-ON SE REAPPROPRIER LE SI A L'AIDE DES PROCESSUS DEVOPS COMMUN ENTRES TOUTES LES USINES DE DEVELOPPEMENT ?

COMPRENONS COMMENT TRAVAILLENT LES EQUIPES

QU'EST-CE QUE LA CULTURE DEVOPS ?

- INFRA AS CODE
- 12 FACTORS APP
- LES PIPELINE
- STRATEGIES DE DEPLOIEMENT

GOLDEN METRICS

AVANÇONS D'UN FRONT COMMUN

COMPRENONS COMMENT LES EQUIPES IT TRAVAILLENT ?

Les équipes de développement travaillent sur divers environnements techniques et n'utilisent pas les mêmes méthodes de développement.
Chaque équipe avait sa propre usine de développement hétérogène. Les équipes de développement, Cobalt (*Core Banking*) et Spartacus (*front*), étaient les premières du site de Nantes. Leurs applications étaient alors hébergées chez Clever Cloud, leurs codes étaient versionnés dans un gestionnaire de version (*Git*), cela leur garantissait une complète autonomie car au moment de la création du site de Nantes, l'équipe infrastructure n'était composée que d'une seule personne (Edgar Navarre). La méthode de livraison Clever Cloud est très simple, il suffisait au développeur de pousser le code dans une branche GIT pour que celui-ci soit automatiquement build, Release & deploy sur leur plateforme. Cette méthode de livraison a rapidement plu aux développeurs de part sa facilité d'apprentissage, mais ne permettait pas aux développeurs de s'approprier le processus de livraison du code, tout le savoir-faire restait chez Clever Cloud.
L'équipe Kitt (*outil interne*) est une équipe qui a été formée il y a 2 ans. Leurs applications étaient hébergées chez Google cloud, car ils utilisaient des technologies de déploiement automatique (*Ansible*[35]) pour leurs applications, mais également car ils travaillaient sur des technologies Microsoft. A l'inverse de la méthode de livraison Clever Cloud, livrer une application sur la plateforme GCP nécessite un engagement fort des développeurs pour comprendre la notion d'infrastructure présente dans les templates Ansible. Le background de l'équipe KITT sur les notions d'infrastructure a permis d'utiliser cette méthode, mais aurait difficilement été envisageable dans les autres équipes n'ayant pas une vision devops.

Quant aux équipes infrastructures, nous nous occupions des infrastructures Rackspace, GCP et interne. Le temps dont nous disposions était alloué pour des tâches d'administrations chronophages qui demandaient énormément de temps (Ex la création de serveurs, règles de pare-feu, création d'utilisateurs etc.) des tâches n'apportant aucune plus value.
Pour la partie Google Cloud, nous avons appris sur cette environnement des nouveaux concepts de la culture devops. J'ai pu pour la première fois mettre en place de l'infra as code en utilisant Terraform[36] et la gestion de configuration via Ansible, ce qui m'a permis de me rendre compte de la

[35] Outil de déploiement de logiciels et de gestion de configuration.

[36] Environnement logiciel d'infrastructure as code publié par HashiCorp

puissance de l'outil pour mettre en place des infrastructures ou les faires évoluer. J'ai pu faire évoluer notre infrastructure cloud d'une gestion manuelle à une infrastructure automatisée. L'ensemble du code produit était hébergé sur un répertoire git ce qui m'a permis d'en apprendre plus sur son utilisation et d'apprendre un peu plus sur de l'approche gitops.

QU'EST-CE QUE LA CULTURE DEVOPS ?

Mais qu'est ce qui défini la culture devops ? Nous pouvons définir le devops comme un mouvement visant à aligner le SI sur les besoins de l'entreprise, permettant de réduire le temps de commercialisation en ayant des cycles de livraison très courts et plus réguliers. Aujourd'hui, les entreprises proposant des services web sont très concurrentielles, il devient indispensable d'avoir la capacité de livrer rapidement des nouvelles fonctionnalités. La méthodologie devops consiste également à réunir les différents partis, en améliorant la communication entre les développeurs et les administrateurs systèmes afin de mieux collaborer ensemble. Ainsi en améliorant la communication et les outils techniques, le devops peut permettre aux équipes de réduire le temps de livraison des nouvelles fonctionnalités.
La culture devops regroupe un ensemble de concepts :
DevOps est un ensemble de pratiques, de guides et de culture.
CA.L.M.S

- Culture Automation
- Lean (Lean Management ou Continuous Delivery)
- Measurement
- Sharing

Objectifs :

- Briser les silos dans le Dev, Ops, le réseau, la sécurité, etc.

Dans une approche DevOps

- Améliorez quelque chose
- Mesurer les résultats
- Partagez les résultats avec l'ensemble de l'organisation

En annexe n°1 se trouve un extrait du livre "Accelerate: The Science of Lean Software and DevOps", de Nicole Forsgren, PhD, Jez Humble, and Gene Kim montrant les étapes permettant d'amener une culture DevOps au sein d'une entreprise.

INFRA AS CODE

Le concepts “d’infrastructure as code” (IAC[37]) peut s’expliquer de la manière suivante : C’est écrire la configuration de notre infrastructure sous forme de code déclaratif, déployable chez des fournisseurs de services cloud.
Pour les équipes techniques (dev ou ops), cela évite d’effectuer des opérations manuelles. On déclare les différents paramètres que nous avons besoin dans notre infrastructure, en déclarant notre architecture dans du code, nous pouvons le versionner pour n’avoir une seule source de vérité. En ayant une seule source de vérité de notre architecture, il est plus simple pour les équipes techniques de la faire évoluer et de connaître son état réel à un instant T. Cela nous permet pour les suivis d’infrastructure, inventaires, audits, de répondre avec exactitude, car notre infrastructure est “figée” dans le code. De plus, l’utilisation de l’IaC, permet de standardiser notre architecture dès sa conception et permet à l’ensemble des équipes IT d’avoir la même version de l'infrastructure.
Il y a plusieurs intérêts à mettre en place de l’IaC comme augmenter notre efficacité, en optimisant notre gestion des ressources ; la capacité de recyclage, une fois le code écrit, il peut être adapté et exécuté pour déployer un nouvel environnement par exemple; ou encore une réduction des coûts et de la charge de travail, le fait d’automatiser la gestion de l’infrastructure permet d’économiser beaucoup de temps et d’argent.
Il y a des contraintes à prendre en compte, il est difficile de la mettre en place sur une infrastructure mutable [38] existante, car l’IaC nous permet de faire de l’infrastructure immuable[39].

Pour les équipes techniques, cela demande de l’apprentissage pour appréhender ces nouveaux processus de travail, chose que j’ai pu constater lors que j’ai mis en place de l’infra as code sur l'environnement Google Cloud. Auparavant, les équipes d’administrateurs systèmes avait pour habitude de prendre soin de leurs serveurs, beaucoup d’action manuelles pour les entretenir. Aujourd'hui nous sommes dans l’air de l’industrialisation, les serveurs sont démarrés en quelques secondes puis supprimés lorsque l’on en a plus besoin. Il existe d’autres risques comme la réplication d’erreurs, l’écart de configuration ou encore la destruction accidentelle de l’infrastructure.
Le code initial étant développé par des humains, le risque zéro n’existe pas, et par conséquent, il se pourrait que des erreurs mineures se glissent dans le code.
Pendant ce temps, plusieurs machines pourraient avoir été créées automatiquement avec de telles erreurs de code. En effet les écarts de configuration peuvent arriver lorsqu’une action est faite de

[37] Infrastructure as code
[38] Les changements sont appliqués en plus de l'infrastructure existante avec un historique de changements.
[39] Chaque changement est en fait une nouvelle infrastructure.

manière manuelle sans respecter le workflow. Au fil du temps, les écarts peuvent produire des irrégularités ou provoquer des incidents. Enfin, pour le risque de destruction accidentelle, certains outils ont la capacité de détruire aussi vite qu'a été déployé un environnement. Il est donc nécessaire d'être prudent pour éviter les mauvaises surprises.

Figure 10 - Cloud Infrastructure - *Denise Yu (@deniseyu21).*

Pour configurer l'infra as code de notre plateforme Kubernetes, nous utilisons un outil basé sur le langage yaml appelé Helm. Helm nous permet de faciliter le déploiement d'une application ainsi que la gestion du cycle de vie des applications. Grâce à des "charts[40]" qui sont des packages kubernetes regroupant un ensemble de fichiers yaml contenant des variables d'environnements et d'infrastructures. Ces fichiers contiennent diverses informations sur la configuration en fonction des environnements pour exécuter une application dans un cluster kubernetes.

Nous avons mis à disposition des développeurs, un template Helm, qui leur permet de n'avoir qu'à récupérer ce template dans leur répertoire GIT, compléter ce template avec ses variables d'environnements. Le template Helm contient toutes les notions kubernetes nécessaires au déploiement d'applications sur cette plateforme. De cette manière, une personne non initiée à

[40] Package kubernetes

kubernetes peut déployer son application rapidement sans se préoccuper de l'infrastructure. Voici un exemple d'arborescence d'un package HELM.

```
├── Chart.yaml # fichier yaml contenant les informations a propos des applications (N° de versions)
├── README.md # OPTIONNELLE: Un fichier README lisible humainement
├── dev
│   └── values.yaml # Variable spécifique à l'environnement de développement
├── prod
│   └── values.yaml # Variable spécifique à l'environnement de production
├── staging
│   └── values.yaml # Variable spécifique à l'environnement de staging
├── templates # Template Helm commun a toute les usines de dev
│   ├── configmap.yaml
│   ├── cronjob.yaml
│   ├── deployment.yaml
│   └── service-prive.yaml
└── values.yaml #Variable commune et ou par défaut à tous les environnements
```

Une des notions introduite dans le devops est la méthodologie des 12 facteurs apps. Cette méthode relate des bonnes pratiques de développement, Elle a été écrite par Adam Wiggins (*co-fondateur d'Heroku*)

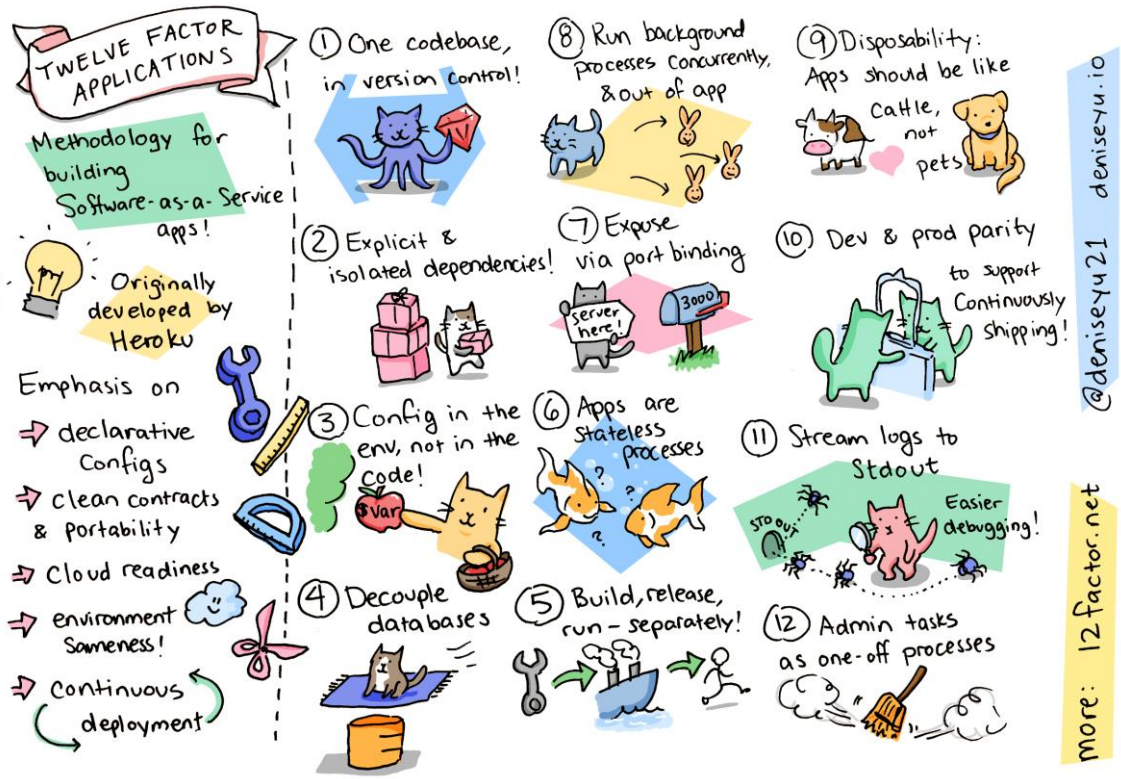


Figure 11 - 12 factors App - Denise Yu (@deniseyu21).

<u>1er facteur: Le code</u>

"Une base de code suivie avec un système de contrôle de version, plusieurs déploiements"[41]

Le premier facteur nous dit que tout le code doit se trouver à un seul emplacement centralisé et versionné. Nous apprenons également qu'une base de code est à l'origine de plusieurs déploiements et non pas l'inverse, qu'un déploiement ne doit pas être lié à plusieurs bases de code. Dans le cas où du code serait commun à plusieurs applications, alors celui-ci doit être factorisé puis extrait sous forme de librairie.

[41] Adam Wiggins, 12 factor app, 2017

Cette approche, permet d'avoir qu'une base de code ne contenant que le code de l'application, et implique l'utilisation d'un gestionnaire de dépendances permettant de récupérer les bibliothèques supplémentaires nécessaires au fonctionnement de l'application.
Un outil de gestion des versions tel que Git permet de répondre à ces premiers facteurs.

Application :
Les équipes de développement, utilisent la solution Gitlab repository pour leur versionning des applications. Dans le même répertoire que le code développeur, se trouve la configuration Helm contenant les informations sur l'infra as code Kubernetes

2ème facteur : Les dépendances

"Déclarez explicitement et isolez les dépendances"[42]

Les applications web modernes sont rarement conçues d'un seul bloc "dites monolithes" à usage unique. Elles sont, au contraire, conçues pour tirer partie des librairies (généralement externes). L'utilisation de librairies externes peut introduire certains problèmes. L'ajout d'une librairie dans le code peut rendre complexe sa mise à jour par la suite. Le fait de ne pas l'ajouter rend compliqué son déploiement, car on ne sait pas, lors du déploiement, quelle librairie ou version utiliser.
Pour palier à ces problèmes, il est possible d'utiliser un outil de gestion des dépendances de notre application. Il va récupérer et gérer les dépendances de notre application à notre place. Pour cela, il suffit de lister les dépendances ainsi que les versions que requiert notre application.
Application :
La gestion des dépendances et des librairies applicatives est portée par la solution Nexus chez Claranet. Nexus permet de stocker ce que l'on appelle des artéfacts applicatifs. Toutes les dépendances de nos applications sont stockées et versionnées dans Nexus.
3ème facteur: La configuration

"Stockez la configuration dans l'environnement"[43]

Les applications web nécessitent de la configuration. Que ce soit pour spécifier l'emplacement des ressources attachées (*base de données, API, …*), les paramètres de l'application, l'emplacement des environnements cible (*dev, staging,prod, …*).

[42] Adam Wiggins,2ème facteurs, 12 factor app, 2017
[43] Adam Wiggins,3ème facteurs, 12 factor app, 2017

Ce facteur consiste à utiliser des variables d'environnements configurées au niveau des applications ou des serveurs. Ces variables sont spécifiques à chaque environnement sur lequel l'application s'exécute.

Application :

Dans kubernetes, ces variables d'environnements se situent dans des objets configmap ou secrets qui vont permettre aux développeurs d'appliquer des paramètres différents (gestion des ressources, services managés, règles de pare-feu, etc) en fonction de l'environnement dans lequel il va déployer son application sans toucher au code applicatif. Ces configmaps sont gérés dans les templates Helm de nos applications

4ème facteur : Services externes

"Traitez les services externes comme des ressources attachées"[44]

L'application ne fait pas de distinction entre un service local et un service tier. Par exemple, le changement d'une base de données locale par une base de données managée par un hébergeur cloud tier doit pouvoir se faire sans modifier le code applicatif. Seule la configuration change. Cela nous permet d'attacher ou détacher une ressource externe à tout moment, quelque soit l'environnement (*staging, production*)

Dans kubernetes, tous les services qui ne font pas partie de l'application principale, tels que les bases de données ou le stockage externes, sont accessibles via des services externes. Ces appels aux services externes sont configurés dans les configmaps kubernetes de la même façon que le 3ème facteur : configurations.

5ème facteur: Assemblez, publiez, exécutez

"Séparez strictement les étapes d'assemblage et d'exécution"[45]

Le déploiement d'une application est divisé en trois phases :

- La phase d'assemblage (build), est une étape de transformation qui convertit du code en un ensemble exécutable.
- La phase de publication (release), contient à la fois la construction et la configuration, prête à être exécutée dans un environnement.
- La phase d'exécution (run), exécute l'application dans un environnement, dans une version donnée.

[44] Adam Wiggins, 4ème facteur, 12 factor app, 2017

[45] Adam Wiggins,5ème facteur, 12 factor app, 2017

L'idée clé est de séparer le processus de déploiement en deux phases distinctes. La première phase est de transformer le code en un livrable (build). La deuxième phase consiste à ajouter les variables d'environnement pour que le nouveau livrable soit utilisable (release + run).
Ces nouveaux processus de déploiement permettent de maximiser la capacité de livraison, simplifient les actions de retour arrière (rollback) en cas de problèmes mais permettent également d'implémenter ces processus dans des pipelines automatisés.
Les usines de développement utilisent l'outil gitlab CI pour la construction des applications.
Gitlab nous permet de lancer des pipelines où les taches Builds/Release/Run sont automatisées.

6ème facteur: Processus

"Exécutez l'application comme un ou plusieurs processus sans état"[46]

Une application doit être conçue pour que chaque requête puisse être traitée par des processus différents, qui ne sont pas liées et qui ne communiquent pas entre elles. Une application "sans état" (stateless) nous permet de faire de la scalabilité horizontale, sans modifier la configuration de l'application.
Par exemple, imaginons que nous voulions augmenter le nombre de requêtes sur notre site web, nous avons deux options : soit augmenter la puissance des serveurs (RAM,CPU), on parle alors de scalabilité verticale, soit on ajoute plus de serveurs, on parle alors de scalabilité horizontale.
Dans un cluster kubernetes scalabilité horizontale et stateless sont des fonctionnalités 'out of the box'. Ces fonctionnalités permettent à nos développeurs de facilement créer des applications stateless.

7ème facteur: Associations de ports

"Exportez les services via des associations de ports"[47]

Les applications 12 facteurs sont auto-suffisantes, c'est à dire qu'elles exposent elles-mêmes sur un nom de domaine et port préfini.
Exemple si on expose un service web (http) *https://thibautfontaine.fr:8080* maintenant que notre application est exposée en tant que service, elle peut être consommée par d'autres services ou applications via l'url définie.

[46] Adam Wiggins, 6ème, 12 factor app, 2017
[47] Adam Wiggins, 7ème, 12 factor app, 2017

Dans un environnement de production où l'on peut avoir plusieurs micro-services offrant différentes fonctionnalités, il faut que les micro-services communiquent sur des protocoles bien définis. Nous pouvons exposer sur le réseau nos micro-services sur des ports spécifiques. Toutes cette configuration est stockée de façon immuable dans les packages Helm.

8ème facteur: Concurrence

"Grossissez à l'aide du modèle de processus"[48]

Ce 8ème facteur aborde la notion de parallélisme. C'est à dire que pour suivre la croissance de l'application, il est nécessaire de créer des instances supplémentaires pour traiter un volume de traitements très courts dans le temps. Cela a pour avantages d'avoir un fort impact en terme de performances et d'avoir un coût très faible. Un autre avantage est la tolérance aux pannes. En effet, s'il y a une défaillance sur une des autres instances, cela ne paralysera pas l'application.
Kubernetes nous permet de faire évoluer l'application sans état pendant son exécution. Le nombre de répliques souhaitées est défini dans le modèle de déploiement helm et peut être modifié au moment de l'exécution.

9ème facteur: Jetable

"Maximisez la robustesse avec des démarrages rapides et des arrêts gracieux"[49]

Les instances(processus) faisant tourner les applications sont jetables. Elles peuvent démarrer ou être stoppées à n'importe quel moment, pour répondre à un besoin de montée en charge ou bien la montée de version de l'application (rolling update).

Il y a 3 points à respecter :

Minimiser le temps de démarrage des instances. Plus le temps de déploiement est court, plus cela rend agile les processus de release et de scalabilité horizontale.

Éteindre correctement les instances lorsqu'elles reçoivent un signal.

Prenons en exemple un service web en coupant le service (refusant, par la même occasion, toute nouvelle requête), tout en permettant à la requête courante de se terminer pour s'arrêter ensuite.

[48] Adam Wiggins, 8ème, 12 factor app, 2017

[49] Adam Wiggins, 9ème, 12 factor app, 2017

10ème facteurs: Parité dev/prod

"Gardez le développement, la validation et la production aussi proches que possible"[50]

Le principe qui se cache derrière est d'éviter au maximum le fossé entre les environnements de développement et de production. Ces disparités existent lorsqu'il n'y a pas assez de collaboration entre les différentes équipes techniques ou encore lorsque les outils sont différents entre les environnements de développement et de production.

Les applications 12-factor sont conçues pour du déploiement continu, d'où le besoin de réduire au maximum les écarts entre le développement et la production. C'est aussi un des principes lié au devops, que les outils utilisés dans les deux environnements soient identiques.

Dans un contexte kubernetes, la parité des environnements se paramètre dans les templates helm. Chaque environnement a des paramètres qui lui sont spécifiques pour exécuter l'application (ressources, services exposés, version d'application, etc)

11ème facteur : Logs

"Traitez les logs comme des flux d'événements"[51]

Les logs sont des flux d'événements, ordonnés dans le temps, collectés à travers les flux de sortie de toutes les applications et services externes qui tournent. Les logs, dans leur forme brute, sont au format texte avec un événement par ligne, n'ont pas de début ou de fin fixe, mais se remplissent en continu tant que l'application est en marche.

Une application 12 facteur ne s'occupe pas du stockage des logs, elle va les pousser vers un service tier qui s'occupera du stockage, du traitement et de l'affichage.

Lors du déploiement, les logs de chaque application seront liés par un environnement spécifique et routés vers des solutions de visualisation de logs et d'archivage.

Nous utilisons la stack Elastic (anciennement ELK[52]) pour la récupération des logs des différentes applications ainsi que Grafana pour la visualisation graphique de ces mêmes logs. Les logs applicatifs sont normés pour permettre une meilleure lisibilité lors qu'il est nécessaire de debug [53] le comportement d'une application.

[50] Adam Wiggins, 10ème, 12 factor app, 2017

[51] Adam Wiggins, 11ème, 12 factor app, 2017

[52] Elasticsearch Kibana Logstash

[53] Outil permettant d'analyser les bugs d'un programme

12ème facteur: Processus d'administration

"Lancez les processus d'administration et de maintenance comme des one-off-processes"[54]
Parfois, des opérations de maintenance telles que l'évolution des ressources systèmes allouées aux applications, des scripts de mises à jour, sont parfois nécessaires dans des environnements de production.
Les tâches d'administration sur la production doivent être ponctuelles, uniques, et surtout exécutées au préalable dans des environnements spécifiques. Par exemple, pour de petites tâches sur un conteneur spécifique, nous pouvons l'effectuer via les commandes d'administration kubernetes.
Pour les tâches plus complexes qui impliquent des changements, nous devons les réaliser via les templates helm afin que les modifications soient durables dans le temps. Dans le cadre des évolutions des ressources systèmes des applications, nous avons mis à jour les templates helm avec les nouvelles configurations de ressources afin qu'ils soient pris en compte lors du prochain déploiement de l'application.
Pour réaliser les tâches complexes nous les intégrons dans des pipelines automatisés.

LES DIFFERENTES PIPELINE

La mise en place de pipelines dans les différentes usines de développement, permettent aux développeurs d'automatiser certaines phases de construction (build,test,audit,release,)
C'est une méthode de développement faisant référence au 5ème facteur de la méthodologie 12 fact app. Les développeurs ajoutent régulièrement du code sur un dépôt (repository). Cela peut aller de quelques modifications par jour à quelques modifications par heure. Pour cela, ils utilisent une CI qui vient du mot anglais *Continuous Integration* qui veut dire intégration continue.

- Continuous Integration

L'intégration continue consiste à intégrer les changements apportés au code développé par les équipes de façon continue, afin de détecter et de corriger les éventuelles erreurs.
Pour détecter les éventuelles erreurs, la pipeline CI intègre différents types de tests (unitaires[55], fonctionnels[56], sécurité, etc) Après avoir passé ces différents tests, l'étape d'après est la livraison continue.

[54] Adam Wiggins, 12ème, 12 factor app, 2017
[55] Vérifie le bon fonctionnement d'une partie précise d'un code informatique
[56] Teste le fonctionnement du code avec l'ensemble du code de l'application

➔ Continuous Delivery

La CD pour *continuous delivery* en anglais signifiant livraison continue. La suite logique après l'étape d'intégration continue est la livraison continue pour s'assurer d'une livraison rapide des nouveaux changements de l'application aux clients. Cela signifie qu'après avoir automatisé les différents tests, nous automatisons le processus de publication de l'application ainsi que son déploiement sur une registry. Dans le cadre d'une livraison continue, nous pouvons décider de publier des versions de notre application de manière quotidienne, hebdomadaire, mais également à la volée.

Cependant la livraison continue comprend quelques contraintes , car il faut être assuré de la qualité ainsi que de la pipeline de livraison de l'application.

➔ Continuous Deployment

La dernière étape avec la livraison continue, c'est le déploiement continue, la CD pour *continuous deployment.* On pourrait certes confondre la livraison continue avec le déploiement continu mais cependant ce sont deux choses différentes. Le déploiement continue est le dernier maillon de la chaîne de livraison car c'est en effet lui qui va déployer de manière automatique la nouvelle version de l'application sur l'environnement souhaité.

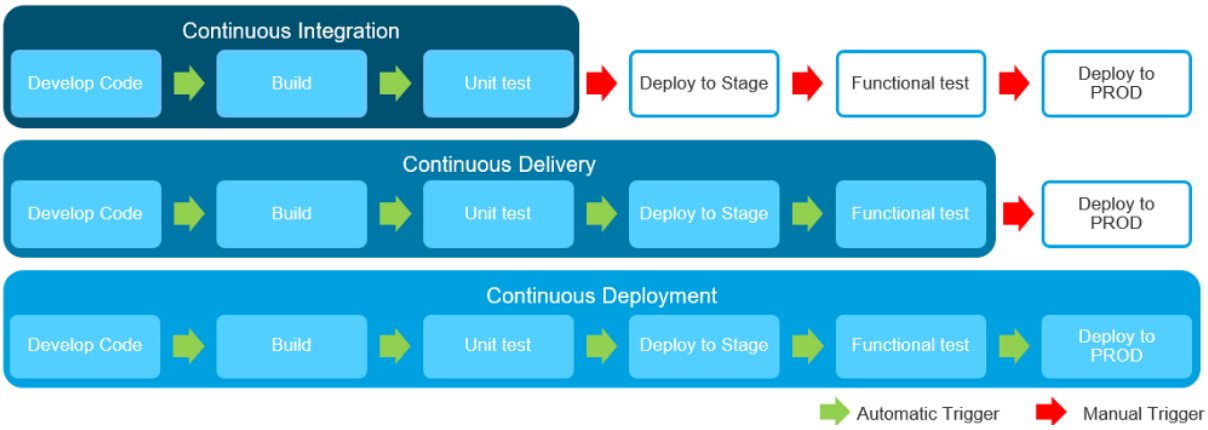


Figure 12 Les Pipelines – Devops Pipeline

Chez Nickel nous sommes arrivés au niveau de maturité continuous Delivery et commençons à challenger le continuous Deployment

Pour rappel, le continuous Delivery est une approche où nous pouvons déployer n'importe quelle version de l'application en appuyant sur un simple bouton. Pour cela nous utilisons des Artefacts.

https://medium.com/@jeehad.jebeile/devops-and-segregation-of-duties-9c1a1bea022e

ARTEFACT

Nous utilisons trois types d'artefact[57].

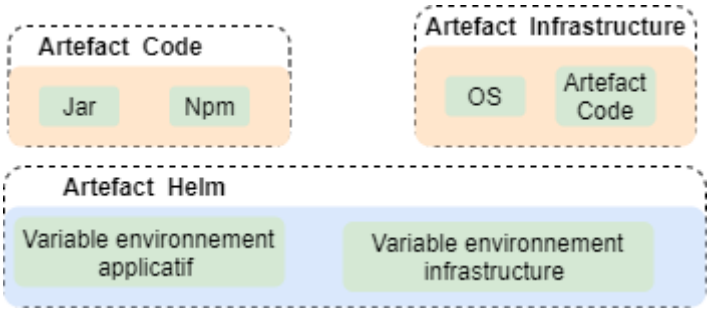


Figure 13 - Artefact Nickel

Un artefact stocké dans le repo nexus (artefact code), contenant les dépendances et librairies des applications (jar, npm), un artefact docker (artefact infrastructure) stocké et versionné dans la registry[58] Harbor contenant l'artefact code ainsi que le système (OS) pour faire fonctionner l'application. Le troisième artefact (artefact Helm) contient la configuration d'infra as code et la version de l'artefact Infrastructure à déployer.

LES ETAPES DEVOPS

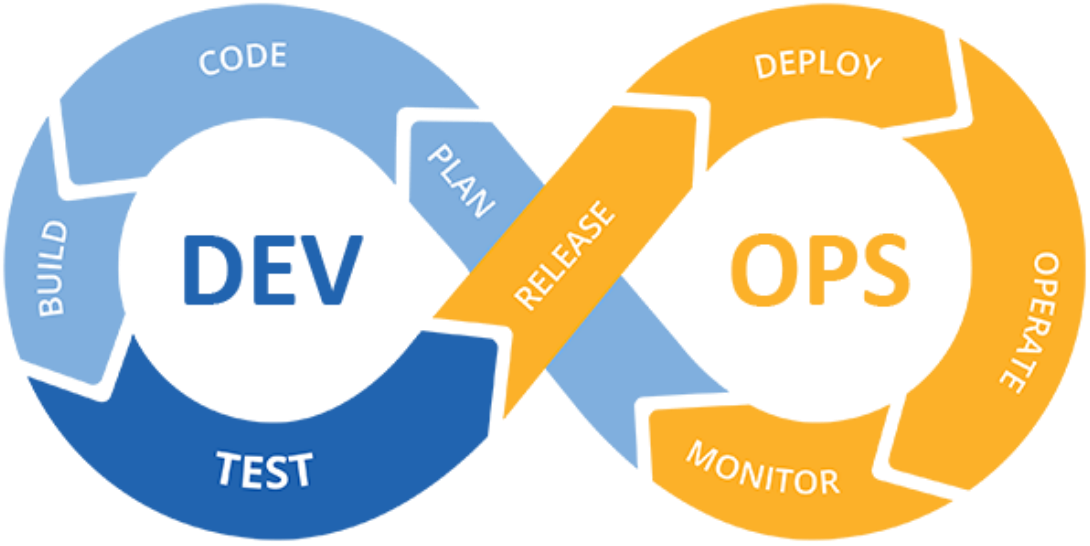


Figure 14 - Processus Devops

Plan:

Le Product Manager décrit dans un ticket Jira un besoin (fonctionnel, bug, feature etc…)

Code:

Le Développeur récupère le code depuis un répertoire GIT

[57] Tout objet produit par un développeur, stocké et versionné dans un repo et prêt à être déployé.

[58] Registre d'images

Il crée une nouvelle branche pour le ticket Jira

Il écrit le code répondant aux besoins et commit dans la branche

Build

À chaque commit du développeur dans sa branche GIT, une pipeline est automatiquement lancée. Cette pipeline contient une phase de build du code permettant au développeur d'avoir un rapide feedback sur l'état de son code.

Test

Une fois le développeur satisfait de son code, il propose une merge request avec ses changements à son équipe.

L'équipe fait une relecture du code et valide ou non le changement.

Une pipeline est alors déclenchée avec les étapes suivantes:

Build d'un artifact Nexus, test unitaire, build d'un artifact docker, build d'un artifact Helm contenant la configuration de l'environnement DEV, déploiement automatique dans l'environnement de DEV afin que l'équipe de développement valide l'intégration du code avec d'autres composants Nickel.

Release

La liste des changements demandés par le product owner est terminée, une nouvelle release de l'application peut être déployée dans l'environnement de staging afin de passer les tests fonctionnels.

Pour cela une pipeline Gitlab est lancée automatiquement pour récupérer les artefacts helm et docker et les déployer dans les environnements de staging.

Deploy

Une fois la release validée par le product owner, l'exploitation déploie les artifacts sur l'environnement de production.

Operate & Monitor

L'équipe exploitation surveille, grâce aux logs applicatifs et aux dashboards, le bon déploiement de l'application et son fonctionnement au quotidien.

LES STRATEGIES DE DEPLOIEMENT :

Pour déployer une application, il existe différentes stratégies de déploiement pour monter en version notre application :

Recreate: La version A est terminée, puis la version B est lancée.

La stratégie de re-création est un déploiement qui consiste à arrêter la version A puis à déployer la version B après que la version A ait été arrêtée. Cette technique implique un arrêt du service qui dépend à la fois de l'arrêt et de la durée de démarrage de l'application. Cette méthode est facile mettre en place, l'application est entièrement re-déployée. C'est également une contrainte car cela a un impact sur les clients. Prévoir un temps d'arrêt qui dépend à la fois de l'arrêt et de la durée de démarrage de l'application peut être un handicap en fonction de la criticité de l'application en question.

- rolling-update ou incrémental : Une version B est déployée lentement et remplace la version A.

La stratégie de déploiement progressif, consiste à déployer progressivement une nouvelle version une application en remplaçant les instances l'une après l'autre jusqu'à ce que toutes les instances soient déployées. Elle suit généralement le processus suivant : avec un pool d'application en version A derrière un load-balancer[59] , une instance de la version B est déployée. Lorsque le service est prêt à accepter du trafic, l'instance est ajoutée au pool. Ensuite, une instance de la version A est retirée du pool et arrêtée. En fonction du système qui s'occupe du déploiement progressif, nous pouvons modifier les paramètres suivants pour augmenter le temps de déploiement. On peut également faire du parallélisme afin d'augmenter le nombre d'instances simultanées à déployer en augmentant le nombre d'instances à ajouter en plus de la quantité actuelle et réduire les indisponibilités pendant la procédure de mise à jour.

Cette méthodologie est facile à mettre en place, le déploiement de la version se fait lentement instance après instance, mais comprend aussi des contraintes, la mise à jour peut prendre du temps et il n'y a pas de coupure "propre" des flux réseaux.

[59] Equilibreur de charge

Blue/Green : La version B est publiée en même temps que la version A, puis le trafic passe à la version B.

La stratégie de déploiement bleu/vert diffère d'un déploiement en rolling update, la version B (verte) est déployée aux côtés de la version A (bleue) avec exactement le même nombre d'instances. Après avoir testé que la nouvelle version répond à toutes les exigences, le trafic passe de la version A à la version B au niveau de l'équilibreur de charge.

Pour :

- Déploiement et retour en arrière instantanés.
- Évite les problèmes de versionnement, l'état de l'application entière est modifié en une seule fois.

Contre :

- Coûteux car il nécessite de doubler les ressources.
- Il convient de tester correctement l'ensemble de la plate-forme avant de la mettre en production.

Canary : La version B est publiée pour un sous-ensemble d'utilisateurs, puis le déploiement est complet.
Canary Release est une technique visant à réduire le risque d'introduire une nouvelle version de logiciel en production en déployant lentement le changement à un petit groupe d'utilisateurs avant de le déployer à l'ensemble de l'infrastructure et de le rendre disponible à tous.
La qualité de la version canary est évaluée en comparant les mesures clés qui décrivent le comportement de l'ancienne et de la nouvelle version. En cas de dégradation significative de ces paramètres, le canary est abandonné et tout le trafic est acheminé vers la version stable afin de minimiser l'impact d'un comportement inattendu. Un déploiement canary ne doit pas être utilisé pour remplacer des méthodes de tests tels que les tests unitaires ou d'intégration. Le but d'un canary est de minimiser le risque de comportements inattendus qui peuvent se produire sous la charge opérationnelle.

Cette stratégie permet d’avoir des informations importantes sur l'application, comme tester :

- Tester les performances de l’infrastructure.
- Effectuer des tests A/B sur la base de données démographiques et de profils d'utilisateurs, par exemple "les utilisateurs âgés de 20 à 25 ans vivant en Bretagne".

- Obtenir un retour d'informations de la part d'un sous-ensemble de bêta-testeurs ou d'utilisateurs de votre entreprise.

Pour :

- Publication d'une version destinée à un sous-ensemble d'utilisateurs.
- Pratique pour le contrôle du taux d'erreurs et des performances.
- Retour en arrière rapide.

Contre :

- Déploiement lent.

Test A/B : La version B est publiée pour un sous-ensemble d'utilisateurs sous certaines conditions. Les déploiements de tests A/B consistent à livrer à un sous-ensemble d'utilisateurs vers une nouvelle fonctionnalité dans des conditions spécifiques. Il s'agit généralement d'une technique permettant de prendre des décisions commerciales basées sur des statistiques, plutôt que d'une stratégie de déploiement.

Pour :

- Plusieurs versions fonctionnent en parallèle.
- Contrôle total de la répartition du trafic.

Contre :

- Nécessite un équilibreur de charge intelligent.
- Difficile de résoudre les erreurs pour une session donnée, le traçage distribué devient obligatoire.

Shadow: la version B reçoit le trafic client en même temps que la version A et n'a pas d'impact sur la réponse.

Un déploiement shadow consiste à publier la version B en même temps que la version A, puis transmettre les nouvelles demandes entrantes de la version A et les envoyer vers la version B également sans impact sur le trafic de production. Cela est particulièrement utile pour tester la charge de production sur une nouvelle fonctionnalité. Un déploiement de l'application est déclenché lorsque la stabilité et les performances répondent aux exigences. Cette technique est assez complexe à mettre en place et nécessite des exigences particulières, notamment en ce qui concerne le trafic sortant. Par exemple, dans le cas d'une plateforme de panier d'achat, si vous souhaitez tester le

service de paiement en mode "shadow", vous pouvez faire en sorte que les clients paient deux fois leur commande. Dans ce cas, vous pouvez résoudre le problème en créant un service de simulation qui reproduit la réponse du fournisseur.

Pour :

- Test de performance de l'application avec le trafic de production.
- Pas d'impact sur l'utilisateur.
- Pas de déploiement tant que la stabilité et les performances de l'application ne sont pas conformes aux exigences.

Contre :

- Coûteux car il nécessite le double de ressources.
- Pas un vrai test utilisateur et peut être trompeur.
- Complexe à mettre en place.
- Nécessite un simulateur d'API dans certains cas.

Il existe de multiples façons de déployer une nouvelle version d'une application et cela dépend des besoins, des capacités techniques et du budget. Lors de la publication dans des environnements de développement/staging, une recréation ou un déploiement accéléré est généralement un bon choix. En ce qui concerne la production, un déploiement progressif ou bleu/vert est généralement un bon choix, mais il est nécessaire de tester correctement la nouvelle plateforme.

Les stratégies blue/green et shadow ont un impact plus important sur le budget car elles nécessitent de doubler la capacité des ressources à un instant T. Si l'application manque des tests ou si l'impact/stabilité du logiciel n'est pas fiable, il est possible d'utiliser un canary release, un test a/b ou une livraison en rolling update. Si l'on souhaite tester une nouvelle fonctionnalité pour des utilisateurs spécifiques, nous pouvons filtrer en fonction de certains paramètres comme la géolocalisation, la langue, le système d'exploitation ou les fonctionnalités du navigateur, nous pouvons alors utiliser la technique de test A/B.

Enfin une livraison shadow est complexe et nécessite un travail supplémentaire pour simuler le trafic de sortant, ce qui est obligatoire lorsque l'on appelle des dépendances externes. Cependant, cette technique peut être utile lors de la migration vers une nouvelle technologie de base de données et utiliser le trafic fantôme pour surveiller les performances du système sous charge.

Au sein de chez Nickel nous utilisons pour la plupart des applications, une stratégie de déploiement rolling update et Blue/green pour les applications dont le trafic réseau ne pas être coupé. À terme, la cible est d'atteindre la stratégie de Canary pour minimiser les taux d'erreurs pour nos clients.

Afin de visualiser les erreurs de déploiement, nous mettons en place des outils de supervision.

SUPERVISER DES METRIQUES :

La supervision est un outil important dans une architecture informatique, il nous permet de mesurer, quantifier n'importe quel type de données. En règle générale, dans des infrastructures, nous ne mesurons que des données matérielles, consommation de la RAM, CPU, I/O disques, trafic réseaux.

L'arrivé du Cloud a modifié notre façon de monitorer notre infrastructure. Dans le cloud, on ne monitore presque plus les données matérielles citées précédemment mais nous monitorons des nouvelles métriques. Nous devons changer notre manière de superviser notre infrastructure, nous devons répondre à deux questions : qu'est-ce qui est cassé et pourquoi ? Dans le livre *Site Reliability Engineering* - publié par Oreilly en 2018 raconte dans le chapitre 6 des quatre golden signals. Ce sont quatre types de données qui nous permettent de superviser de manière efficace une infrastructure moderne et de répondre à nos deux questions. Les golden signals se basent sur quatre métriques; la latence, le trafic, les erreurs et la saturation.

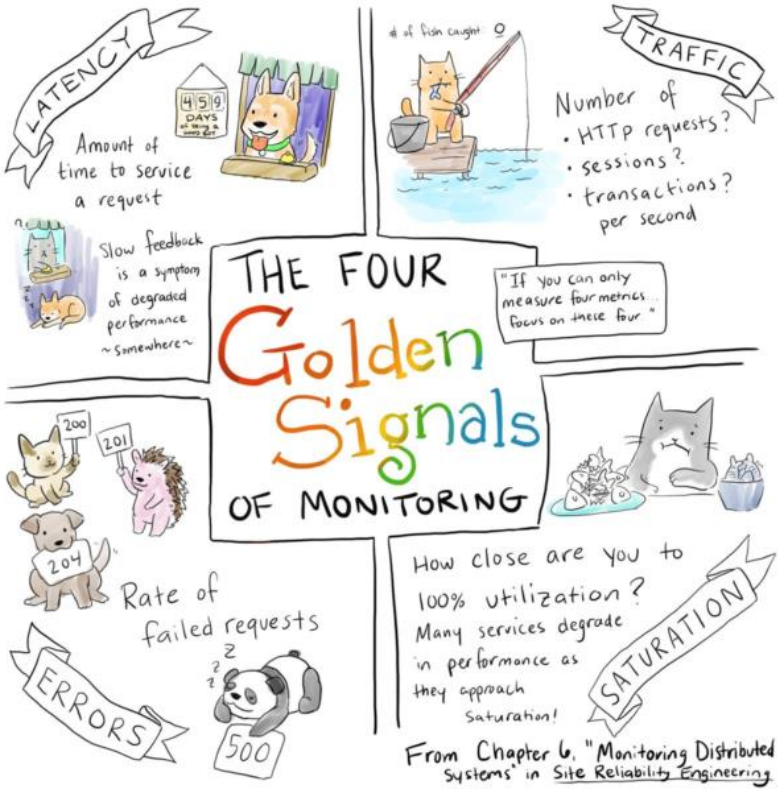


Figure 15 - Golden Signal - Denise Yu (@deniseyu21)

- La latence, c'est le temps de réponse que met notre application à envoyer un résultat à notre client. Par exemple, lorsque vous accédez à un site web, cela va relativement vite. Ce laps de temps entre le moment ou vous cliquez sur le site web et le moment ou le site web s'affiche s'appelle le temps de latence.
- Les erreurs, cette métrique mesure les types de code erreur web, il existe une multitude de codes erreur (*voir webographie, liste code erreur htpp*)
- Le trafic, nous permet de voir le nombre de transactions, le volume de requêtes pour accéder à notre service web.
- La saturation, va nous montrer le niveaux d'utilisation des ressources système (*RAM & CPU*) Ces ressources sont limitées par application, plus le niveau de saturation est élevé, plus le service rendu aux clients est dégradé.

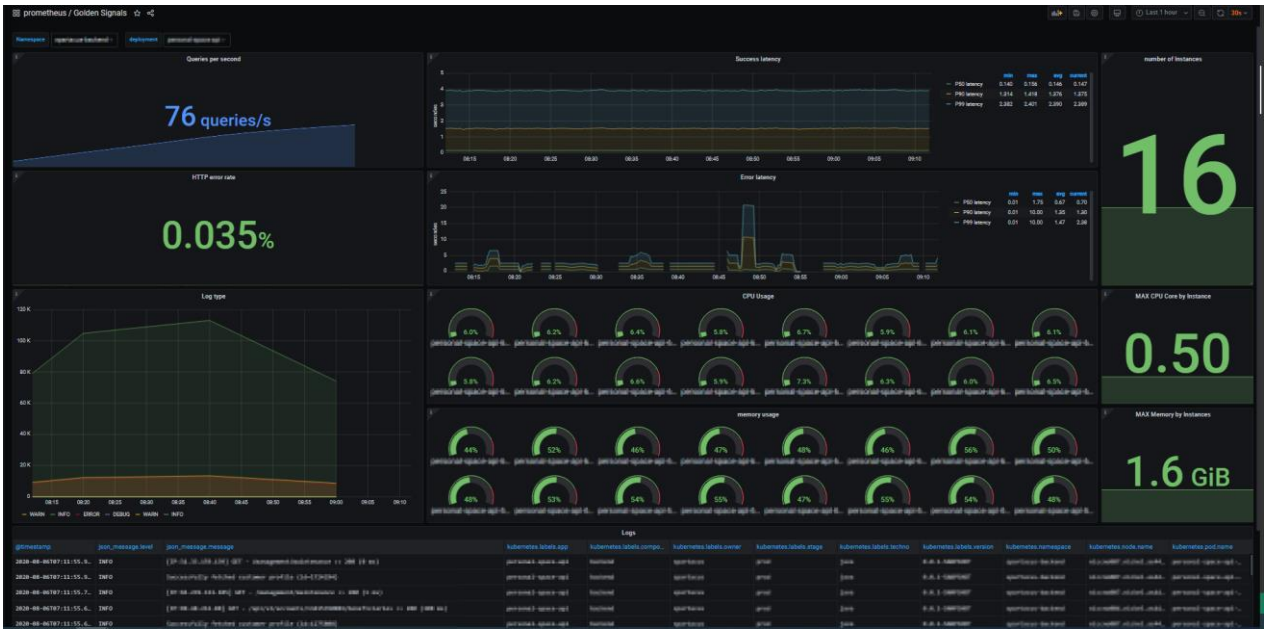


Figure 16 - Dashboard Grafana

Voici un exemple de tableau de supervision d'une application Kubernetes auquel j'ai pu participer à l'élaboration se basant sur les Golden Metrics.

En plus des golden signal, Google aborde un point de supervision, ce sont les *services-level metrics* en anglais, cette notion est abordée dans le livre de Google *The Site Reliability - Workbook*, écrit par Google Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara, and Stephen Thorne
Nous connaissons tous le SLA [60]ou Service Level Agreement en anglais qui est un indicateur contractuel par lequel un prestataire informatique s'engage à fournir un ensemble de services à un ou plusieurs clients. En cas de non respect des SLA cela entraîne des coûts financiers pour l'entreprise. Cependant, il existe d'autre indicateurs de niveaux de services type SLI [61]et SLO[62]
SLI ou indicateur de niveaux de services, mesure le temps de réponse d'un service qui est fourni à un client. Parmi les autres indicateurs de niveau de service figurent le taux d'erreur, mesuré en nombre de requêtes par seconde. Les mesures sont souvent agrégées, pour être transformées en moyenne.
Un autre type d'indicateur important est la disponibilité d'un service dans une période de temps. Bien que la disponibilité de 100 % soit impossible, une disponibilité proche de 100 % est envisageable. Exemple lorsqu'un hébergeur promet un taux de disponibilité à "4 neuf" (*99,99%*)
Une SLO est un objectif de niveau de service, c'est à dire une valeur cible ou une plage de valeurs pour un niveau de service qui est mesuré par un SLI.
Une SLO doit être la valeur cible d'une SLI comprise entre notre seuil le plus bas et notre seuil le plus haut.
Avoir une SLO adapté est compliqué, nous n'avons pas toujours la possibilité de choisir sa valeur. Mais il est possible de se fixer comme objectif une SLO à atteindre et travailler en ce sens pour atteindre des meilleures performances. J'ai pu mettre en pratique en optimisant la configuration du site vitrine de nickel, nous avons dépassé notre SLO initialement fixé dix milles requêtes seconde à trente milles requêtes seconde.
Pour les requêtes HTTP entrantes de l'extérieur vers notre service, la mesure des requêtes par seconde (RPS) est essentiellement déterminée par les désirs des utilisateurs, et nous ne pouvez pas vraiment définir un SLO pour cela.

[60] Service Level Agreement
[61] Service Level Indicators
[62] Service Level Objectives

Un autre point à surveiller lorsque l'on utilise des services cloud, est la supervision des coûts. Le système de facturation dans le cloud public fonctionne à la consommation. Il est nécessaire de ne pas prendre cela à la légère sinon la facture risquerait d'être salée. La culture devops a amené la notion de "FinOps" qui est de permettre à une entreprise de comprendre ses coûts liés à l'utilisation des services cloud. Cette compréhension va lui permettre à la fois de travailler à l'optimisation et à la réduction de ces coûts, ainsi l'aider dans ses prises de décisions qui impliquent un compromis entre les opérations et la finance.

Un des points fondamentaux est la compréhension et l'analyse des coûts.

L'information : comprendre et analyser ses coûts. La phase d'information vise 3 objectifs :

- Avoir de la visibilité sur notre consommation,
- Comment l'utilisation des ressources est répartie aux sein de l'entreprise,
- Analyser le prix chez les différents fournisseurs,

L'utilisation du cloud induit des notions d'élasticité, refléter par un moyen de de paiement à la demande ce qui n'est pas commun pour des services financiers "classiques "

La culture FinOps essaye d'y remédier. La gestion des ressources et la supervision précise via des outils permettant de visualiser les coûts des services en temps réel utilisés, permet d'avoir une vue réelle des coûts du cloud et de donner une meilleure visibilitée aux parties prenantes techniques et financières. L'analyse comparative entre les différentes équipes techniques permet de mettre en place des sondes et d'augmenter la performance de l'organisation dans son ensemble. Cela permet d'inclure les pratiques FinOps dans la logique business de votre organisation.

L'optimisation : prendre les bonnes décisions adaptées aux besoins.

Une fois en possession d'assez d'informations, nous pouvons commencer à définir un plan de réduction des coûts.

Pour mettre en place une réduction des coûts, il est nécessaire de comprendre l'utilisation du cloud dans l'entreprise pour mieux adapter la consommation au besoin. Une fois la demande bien comprise, nous pouvons envisager de réduire les coûts en s'engageant sur la réservation de ressources, plutôt que d'utiliser les ressources à la demande, plus onéreuses.

Le diagnostic des pratiques de consommation et l'analyse comparative permettent aussi de déceler les secteurs dans lesquels l'impact d'un redimensionnement de l'infrastructure nécessaire peut être le plus fort.

L'opérationnel : mettre en place une gouvernance

La mise en place d'une gouvernance cloud en rationalisant les ressources communes entre les différentes équipes. Également, l'utilisation d'indicateurs clés de performance précis permet de suivre les dépenses des équipes, afin d'éviter des dépenses inutiles car comme décrit l'article "2019 Plus 14 Milliards de dollars ont été dépensés inutilement"[63]. Comme décrit l'article *Cloud Waste To Hit Over $14 Billion in 2019* - de Jay Chapel du 4 février 2019.

AVANÇONS D'UN FRONT COMMUN

Nous avons vu que la culture devops au sein de Nickel nous apporte de nouvelles méthodes pour faire évoluer notre manière de travailler. Que l'on soit développeur ou administrateur systèmes, ces nouvelles méthodes nous permettent d'avancer plus vite tout en nous facilitant la vie. Il ne faut cependant pas foncer tête baissée et se précipiter dans la mise en place de la méthodologie devops. Il faut y aller par étape (*step by step*) en appliquant toujours une méthode KISS (Keep it Simple, Stupid) Cette méthode nous aide à avancer étape par étape en commençant par mettre en place des choses à réaliser puis de complexifier avec le temps. De plus, les nouveaux tableaux de supervision (golden metrics) a permis d'installer une certaine sérénité au sein de l'équipe exploitation. Cela leur permet de réduire considérablement le temps de détection des incident (MTTD) mais également d'augmenter leur temps de réparation (MTTR).
Nous avons réussi à uniformiser les outils technologiques entre les différentes usines de développement. Ces nouvelles méthodes de travail permettent aux développeurs de livrer des nouvelles fonctionnalités applicatives comprenant moins d'erreurs.
Cependant cette dernière soulève des interrogations, comment intégrer les enjeux cyber informatique liés aux domaines bancaires dans de la conception au déploiement d'information et comment peut-on garantir un niveau de sécurité suffisant dans nos processus d'automatisation. Nous allons voir comment le Cloud peut prendre en compte les aspects de sécurité et quels sont les risques liés à son utilisation dans le milieu bancaire.

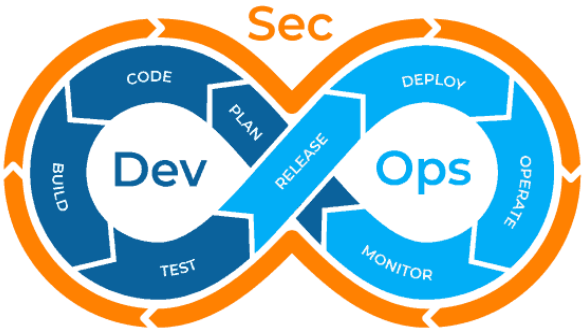


Figure 17 - Processus DevSecOps

[63] Jay Chapel, *Cloud Waste To Hit Over $14 Billion in 2019* - devops.com, 4 février 2019

PARTIE III

APPREHENDER LES ENJEUX CYBER ET REGLEMENTAIRE DANS NOTRE DEMARCHE D'UNIFORMISATION

III) - APPREHENDER LES ENJEUX CYBER ET REGLEMENTAIRE DANS NOTRE DEMARCHE D'UNIFORMISATION

COMMENT UTILISE-T-ON LA PLATEFORME TECHNOLOGIQUE D'UN HEBERGEUR CLOUD SANS ETRE DEPENDANT

LES CONTRAINTES DES TECHNOLOGIES CLOUD DANS LE MONDE BANCAIRE.

SECURISER LE COFFRE-FORT

UN COFFRE-FORT REGLEMENTE

III) - APPREHENDER LES ENJEUX CYBER ET REGLEMENTAIRE DANS NOTRE DEMARCHE D'UNIFORMISATION

Dans un milieu en constante évolution, il est important de comprendre les enjeux cyber liés au domaine informatique ainsi que les obligations légales que nous devons respecter. En particulier dans les organisations d'intérêts vitaux où ces obligations sont très strictes dont fait partie le secteur bancaire.

L'uniformisation d'un SI est une chose compliquée lorsque le SI a une lourde histoire. De plus, les géants du cloud (Amazon, Microsoft, Google) proposent énormément de services très complets et faciles à mettre en place pour les novices en la matière.

Prenons un exemple, Amazon est un site de E-commerce, fournissant également des services cloud. Si l'on prend l'histoire d'Amazon, ils ont révolutionné l'informatique que l'on connaît aujourd'hui. Ils ont commencé à innover un cloud privatif pour répondre à leur problématique de croissance, puis se sont mis à vendre ce service au grand public. Amazon étant une entreprise ayant n'a pas qu'un seul métier, et proposant différents services du numérique, rien nous dit que dans les années à venir ils ne proposeront pas un service de paiement en ligne et donc deviendront un des potentiels concurrents.

C'est le cas de Google qui va lancer Google Bank dès 2021 d'où l'importance pour les banques et autres entreprises proposant des services numériques de sortir du cloud public dans les prochaines années.

A contrario Microsoft est le seul fournisseur cloud dont le cœur de métier depuis la création de l'entreprise en 1975 à proposer des outils et des services informatiques aux entreprises.

COMMENT UTILISE-T-ON LA PLATEFORME TECHNOLOGIQUE D'UN HEBERGEUR CLOUD SANS ETRE DEPENDANT ?

Lorsqu'on utilise des services cloud, l'un des principaux pièges est de devenir dépendant aux services cloud. Les services proposés par les hébergeurs sont très performants et faciles d'utilisation afin qu'ils s'intègrent au mieux à notre cas d'usage. Qu'avons-nous comme solution pour éviter de s'enfermer technologiquement ? Nous avons une multitude d'opportunités technologiques pour homogénéiser notre SI tout en nous évitant de nous enfermer dans des technologies propriétaires proposées par les fournisseurs cloud.

Parmis ces possibilitées, la standardisation des applications ou de l'infrastructure. Nous pouvons adopter la méthode des 12 facteurs dans la conception de nos applications afin de les rendre

stateless afin d'avoir la possibilité de pouvoir la déployer sur des plateformes cloud. De plus l'utilisation des infra as code nous permet d'automatiser la gestion de notre infrastructure via du code, comme son nom nous l'indique.
Puis en utilisant des technologies multi-cloud, comme Terraform et Helm permet une plus grande flexibilité dans le choix d'un fournisseur cloud sans nécessité de grand changement dans le code. Sans compter que cela nous aide à répondre à des exigences de sécurité.

LES CONTRAINTES DES TECHNOLOGIES CLOUD DANS LE MONDE BANCAIRE.

Le secteur bancaire doit respecter plusieurs exigences par son caractère vital dans nos vies et celles des entreprises. Il est par conséquent soumis à différentes obligations réglementaires et sécuritaires. L'aspect réglementaire dans l'informatique est un élément important qui est nécessaire de prendre en compte dès le début d'un projet. En particulier lorsque cela touche à de la donnée personnelle, de ressortissant Français ou Européen. Car nous sommes soumis au RGPD[64] entré en vigueur depuis le 25 mai 2018 qui aide à la protection des données personnelles des utilisateurs d'un service internet. Outre la protection des données, les banques doivent respecter des normes, dont la norme PCI-DSS qui est imposée par les fournisseurs de carte de paiement (Visa, MasterCard, American Express, Discover Card et Japan Credit Bureau) visant à limiter la fraude sur les paiements en ligne,
ou par des législations comme la directive Européenne[65] datant du 25 novembre 2015 sur la norme de paiement DSP2 visant à renforcer la sécurité des comptes clients lorsqu'un client s'authentifie.

L'autre aspect dans le domaine bancaire qui est un élément tout aussi important, il s'agit de la sécurité. Ceci englobe un ensemble de règles et de bonnes pratiques à mettre en place au sein de la société à travers une politique de sécurité du système d'information (PSSI) afin de respecter la sécurité des clients et des salariés.

[64] Réglementation Générale à la Protection des Données
[65] Directive Européenne 2015/2366/UE

SECURISER LE COFFRE-FORT

Le milieu bancaire requiert un niveau de sécurité élevé dans divers domaines que ce soit l'accès ou le traitement des données, cycle de vie des ressources informatiques, etc. Les banques mettent en place des bonnes pratiques de sécurité, comme les recommandations proposées par l'ANSSI [66] Ces recommandations se déclinent sur plusieurs thématiques allant de la sécurité des accès aux locaux d'une entreprise aux méthodes utilisées pour chiffrer les communications.

Depuis l'intégration de Nickel au programme NIST en 2018. Le groupe BNP Paribas nous impose un ensemble de bonnes pratiques qui se traduit par un programme de Cyber sécurité.
Les recommandations du groupe sont basées sur un ensemble de recommandations issue de NIST Cyber Security[67] Elles se déclinent en plusieurs composantes, chacune couvrant un périmètre bien précis. L'équipe sécurité est chargée de faire appliquer ces différentes "réglementation" au sein des équipes techniques mais aussi au reste de l'entreprise.

La matrice NIST se décline en plusieurs volet :

La sécurité de la donnée couverte par le sujet **Data security**, comprenant la sécurité de la donnée dans son cycle de vie au sein de l'entreprise, c'est à dire du moment où elle est intégrée à nos systèmes au moment où elle est supprimée.

L'authentification des utilisateurs internes comme externes à l'entreprise sur nos systèmes informatiques et la gestion des habilitations sur nos applicatifs et ressources informatiques sont décrites au travers des thématiques **IAM**[68], **PAM**[69] et **Authentication Mechanisms**.

La gestion des risques informatiques, des incidents cyber et des plans de remédiation en cas d'incident (PUPA70) sont également couvertes par les sujets **Cyber Crisis Management Cybersecurity Management & Governance, Cyber Incidents Management.**

[66] Agence nationale de la sécurité des systèmes d'information
[67] Framework fournissant un cadre politique de conseils en matière de sécurité informatique pour les organisations du secteur privé aux États-Unis
[68] Identity and Access Management
[69] Privileged Account Management
[70] plan d'urgence et de poursuite de l'activité

L’intégration des enjeux cyber informatiques au sein des équipes est divisée en deux parties.
L’un, applicatif orienté dans la conception, utilisation des données et sécurisation des applications. Ainsi que de la protection contre des attaques des applications exposées sur internet couverte par les points **Application Security**, **DDoS Protection** et **Data sec**
Du côté infrastructure, cela couvre un périmètre plus large :
Il y a d’une part la gestion du parc, c’est à dire de pouvoir répertorier des actifs qui se connectent à notre réseau informatique donnant la capacité d’auditer l’état de notre parc informatique traité par le sujet **Asset Management.**
Le réseau informatique doit aussi être imperméable face aux attaques internes ou externes pour cela NIST préconise de mettre en place une segmentation au sein de notre infrastructure pour l’ensemble des cas d’usage d’accès aux environnements qu’ils soient en filaire, en wifi ou en accès à distance, qui est décrite par les thématiques **Network Sec Segmentation** et **Network Sec Wireless & remote.**
Dans la lutte contre l'obsolescence informatique et ne pas être vulnérable aux exploitations des failles de sécurité informatiques des outils, sont mis en place pour suivre ces vulnérabilités auxquelles nous pourrions être exposés. Ceci est présenté par le point **Vulnerability Management.**
Dans la lutte contre les virus informatiques, des outils de sécurité sont mis en place pour stopper des virus et déployer des correctifs.

Pour récolter, surveiller et analyser toutes les données qui sont générées et transitent au sein de l’entreprise, nous mettons en place des outils capable de gérer ce flux de données, de le stocker pour qu’il soit analysé par les équipes compétentes en cas de détection d’un événement inhabituel. Ce projet se retrouve dans la partie **Logging & Detection**

La recommandation NIST **Cyberculture** préconise de faire de la sensibilisation sur les enjeux et risques cyber au travers différentes activitées aussi bien pour les clients que pour les collaborateurs.

Customer Facing Security recommande de mettre des protections pour protéger nos clients face aux pirates informatiques (Prévention contre le phishing, authentification forte sur l’application mobile, etc)

Les recommandations NIST évoluent d’année en année pour s’adapter aux nouvelles évolutions technologiques, mais aussi en fonction des menaces qui sont en perpétuelle évolution. Chaque année des revues de la matrice sont effectuées par le groupe BNP ajoutant des nouvelles composantes.

Parmis les nouvelles composantes se trouvent celles du **Cloud Security** (gestion des données, chiffrement et contrôle d'accès) qui englobent plusieurs sujets vu précédemment.

UN COFFRE-FORT REGLEMENTE

RGPD et Privacy shield[71]

Dans le domaine bancaire, cette législation est un élément important à prendre en compte lorsqu'un projet débute. Le RGPD impose un certain nombre de règles à respecter dans l'utilisation des données utilisateurs, qu'ils soient collaborateurs ou clients. Dans le cadre bancaire, l'application du RGPD se traduit par différents aspects :

La mise en place d'un consentement clair et précis sur la collecte des données auprès de l'utilisateur lorsqu'il accepte les CGV [72]et CGU [73]

La mise en oeuvre d'un registre, pour cartographier les données, comprenant plusieurs éléments :

- Les finalités des données récoltées
- La durée de conservation des données
- Catégoriser les types de données
- Catégoriser qui y a accès
- Les mesures de sécurité appliquées
- Catégoriser les destinataires de données

Les sous-traitants ayant accès aux données d'une entreprise deviennent également coresponsable du traitement des données. Concernant les sous-traitants américains manipulant de la données de résident Européen, ils sont soumis au "Privacy shield" qui est reconnu par la Commission européenne offrant un niveau de protection adapté aux données à caractère personnel transférées par une entité européenne vers des entreprises américaines.

Depuis le 16 Juillet 2020, le Privacy shield a été invalidé par la cour de justice de l'Union Européenne. Par conséquent les entreprises européennes sont obligées de revoir contractuellement l'ensemble de leurs partenaires et sous-traitants qui pourraient manipuler et stocker de la donnée chez une entreprise basée au Etat-Unis. Chez Nickel cela s'applique sur nos prestataires et leurs sous-traitants. Actuellement le service juridique passe en revue chacun de nos partenaires, par exemple dans le cadre du projet de migration. Claranet collabore avec des entreprises américaines s'occupant de la

[71] Bouclier de Protection des Données
[72] Conditions générales de vente
[73] Conditions générales d'utilisation

supervision de leur infrastructure. Il a été demandé de prouver que les données envoyées à ces prestataires ne contiennent pas d'informations à caractères personnels.

CONCLUSION

CONCLUSION

En conclusion, ce travail avait pour ambition de comprendre comment assumer des décisions techniques à court terme, qui étaient liées elles-mêmes à des contraintes fortes, exigeant d'adapter ses choix sur le long terme. Tout d'abord, nous avons vu le contexte contraignant dans lequel Nickel a évolué au cours de ces dernières années, ce qui a permis de mettre en évidence des axes d'amélioration pour rationaliser notre système d'information, et au travers de la culture devops au sein des équipes de développement, nous avons pu automatiser les différentes étapes de développement. En automatisant ces étapes, cela a soulevé des points d'attention sur l'aspect sécurité.

En effet, Nickel a connu une croissance significative dans un laps de temps très court. Nous avons dû apprendre à nous adapter très rapidement et réfléchir à une approche long terme de notre SI. Cela nous permet d'avoir un système d'information le plus souple possible technologiquement et scalable nous dotant ainsi de la capacité d'appréhender au mieux les futures évolutions.

D'une part les concepts devops nous permettent d'aller de l'avant avec une certaine sérénité dans l'uniformisation de notre système d'information, même si réussir une transformation devops n'est pas une chose simple à mettre en place au sein d'une société, car cela demande beaucoup de changements que ce soit au niveau de l'organisation comme au niveau technique.

D'autre part, les nouvelles technologies qu'apportent la culture devops sont très appréciables, car elles contribuent à mieux concevoir nos applications tout en allégeant nos stratégies de déploiement grâce à l'automatisation.

Ensuite l'intégration des enjeux cyber et réglementaires, intégrés dès la conception, facilite la mise en conformité de nos applications dans leur cycle de vie.

Enfin, la segmentation des différentes étapes devops facilite l'intégration des enjeux cyber dans le cycle de vie de nos applications. Ce qui rend également l'application des réglementations bancaires plus simples à appréhender lorsqu'elles sont intégrées dès sa conception.

Finalement nous pouvons nous interroger sur le fait qu'avoir une vision long terme sur son système d'information nécessite d'avoir la capacité à monter en compétences sur les nouvelles technologies et concepts, et d'avoir assez de recul pour comprendre que la frontière qu'il y a entre les développeurs et administrateurs systèmes, s'efface de plus en plus et tend vers de nouveaux métiers. Ce qui m'amène à me poser la question suivante : Comment l'implémentation de choix long terme avec vision devops participe à la transformation de nos métiers ?

BILAN PERSONNEL

Ces deux ans au sein de Nickel m'ont permis de continuer de développer mes compétences tant dans le domaine scolaire que dans le monde de l'entreprise.
J'ai énormément appris sur le plan professionnel en découvrant des nouvelles méthodologies. Cela m'a permis de développer des nouvelles compétences encore inexplorées.
Cette expérience m'a grandi en me challengeant hors de ma zone de confort. J'ai su anticiper face aux contraintes des projets d'entreprise. J'ai appris à faire des choix pensés et pesés, parfois compliqués, en travaillant sur des nouvelles technologies que je ne maîtrisais pas.
Toutes ces expériences nouvelles ont éveillé ma curiosité tout en me donnant envie de m'investir d'avantage et me conforte dans mon objectif fixé de me spécialiser dans le futur en tant que DevSecOps.

Je terminerai en remerciant une nouvelle fois toute l'équipe Nickel pour leur aide et leur soutien sur le plan professionnel ainsi que sur le plan humain. Ces deux années d'alternance ont été très bénéfiques pour bien débuter ma carrière professionnelle. J'en sors grandi avec une grande fierté d'avoir pu ajouter ma pierre à l'édifice dans cette belle aventure. J'apprécie la confiance que Nickel a porté à mon égard, en particulier Benoit Beaulieu qui m'a recruté au sein de son équipe Sécurité du Systèmes d'Information en tant que Chef de projet technique sécurité.

LEXIQUE

IT : Information Technology
ABM technologies : Société de service informatique
FPE : Financières des Paiements Électroniques
SQL : Langage de programmation utilisé pour les bases de données
IIS : Serveur Web des différents systèmes d'exploitation Windows
STAP : Solution de paiement développée par la société AFSOL
Core Banking : Système informatique qui gère toutes les transactions financières des clients
Cassandra, Messaging Kafka : Technologies informatique
GCP : Google Cloud Platform
Fronted : Brique d'une application, la partie émergée de l'iceberg (visible par le client)
Backend : Brique d'une application, la partie submergée de l'iceberg (non visible par le client)
MSSQL: Microsoft sql server
VM: virtual machine
VPN Virtual Private Network
GAFAM: Google, Amazon, Facebook, Apple, Microsoft
OVH et Scaleway : Hébergeur Français
TPE : Très petite entreprise
G suite : Suite de logiciel bureautique Google
Legacy : Un système hérité continuant d'être utilisé dans une organisation, alors qu'il est supplanté par des systèmes plus modernes
OCR : Reconnaissance optique de caractères
.Net : Langage de programmation propriétaire Microsoft
PCI-DSS: Payment Card Industry Data Security Standard
MTTD: Mean time to detect
MTTR: Mean time to Repair
KPI : Key Performance Indicator
Cluster : Regroupement d'équipement informatique
Hyperviseurs : Plateforme de virtualisation
NAS: Network Attached Storage
DHCP: Dynamic Host Configuration Protocol
Active directory : service d'annuaire
Supervision : Permet de surveiller l'état des outils informatique
Kubernetes : Système open source qui vise à fournir une plate-forme de déploiement automatisé
IG : Inspection Générale
BNPP : BNP Paribas
Ansible : Outil de déploiement de logiciels et de gestion de configuration.
Terraform : Terraform est un environnement logiciel d'infrastructure as code publié par HashiCorp
IAC : Infrastructure as code
Mutable : Les changements sont appliqués en plus de l'infrastructure existante avec un historique de changements
Immuable : Chaque changement est en fait une nouvelle infrastructure
Charts : Package kubernetes

Test unitaires : Vérifie le bon fonctionnement d'une partie précise d'un code informatique
Test fonctionnel : Tester le fonctionnement du code avec l'ensemble du code de l'application
Artefact : Tout objet produit par un développeur, stocké et versionné dans un repo et prêt à être déployé.
Registry: Registre d'images
SLA: Service Level Agreement
SLI: Service Level Indicator
SLO: Service Level Objectives

BIBLIOGRAPHIE & WEBOGRAPHIE

BIBLIOGRAPHIE

- Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy, *Site Reliability Engineering* - publié par Oreilly, 2018
- Marty Cagan, *Inspired: How to Create Tech Products Customers Love* - publié par Wiley 2008

Gene Kim, Jez Humble, Patrick Debois, John Willis *The Devops Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations*, 2016

- Mike Rother, *Toyota Kata Managing People For Improvement, Adaptiveness, and Superior Results* - publié par McGraw-Hill Education, 2009

WEBOGRAPHIE

Listes des code http, https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP

mttd-vs-mttf-vs-mtbf-vs-mttr, https://alertops.com/mttd-vs-mttf-vs-mtbf-vs-mttr/

Saber Omrani le 11/09/201 la - https://blog.octo.com/la-dette-technique-dans-un-si/

Image - Art | deniseyu.io

Radio Devops - Comment faire sa transition devops in Podcast Spotify

Radio Devops - Podcast Spotify

Site Reliability Engineering Chapter 6 - Monitoring Distributed Systems - https://landing.google.com/sre/sre-book/chapters/monitoring-distributed-systems/

https://www.bizety.com/2018/08/21/stateful-vs-stateless-architecture-overview/

Sara Joudrey Stateful vs Stateless – What is the Difference, 15/02/2020 https://hedgetrade.com/stateful-vs-stateless-what-is-the-difference/

CNIL - Privacy Shield, https://www.cnil.fr/fr/le-privacy-shield

CNIL - Invalidation du privacy shield, https://www.cnil.fr/fr/invalidation-du-privacy-shield-la-cnil-et-ses-homologues-analysent-actuellement-ses-consequences

SLOs, SLIs, SLAs, oh my—CRE life lessons, https://cloud.google.com/blog/products/gcp/availability-part-deux-CRE-life-lessons

Lucas TERQUEM, « Le FinOps, ou la culture de l'optimisation des coûts liés au Cloud » , https://www.padok.fr/blog/finops-cloud

JAY CHAPEL, "Cloud Waste To Hit Over $14 Billion in 2019" , https://devops.com/cloud-waste-to-hit-over-14-billion-in-2019/

Romain Rouzaud, « Finops monde devops l'optimisation couts pratique, https://www.margo-group.com/fr/actualite/finops-monde-devops-loptimisation-couts-pratique/

Le FinOps, un architecte aux multiples facettes, https://www.lemondeinformatique.fr/actualites/lire-le-finops-un-architecte-aux-multiples-facettes-69900.html

zwindler, 29/07/2020 Le Métier d'ops va disparaître, https://blog.zwindler.fr/2020/07/29/au-secours-le-metier-dops-va-disparaitre/

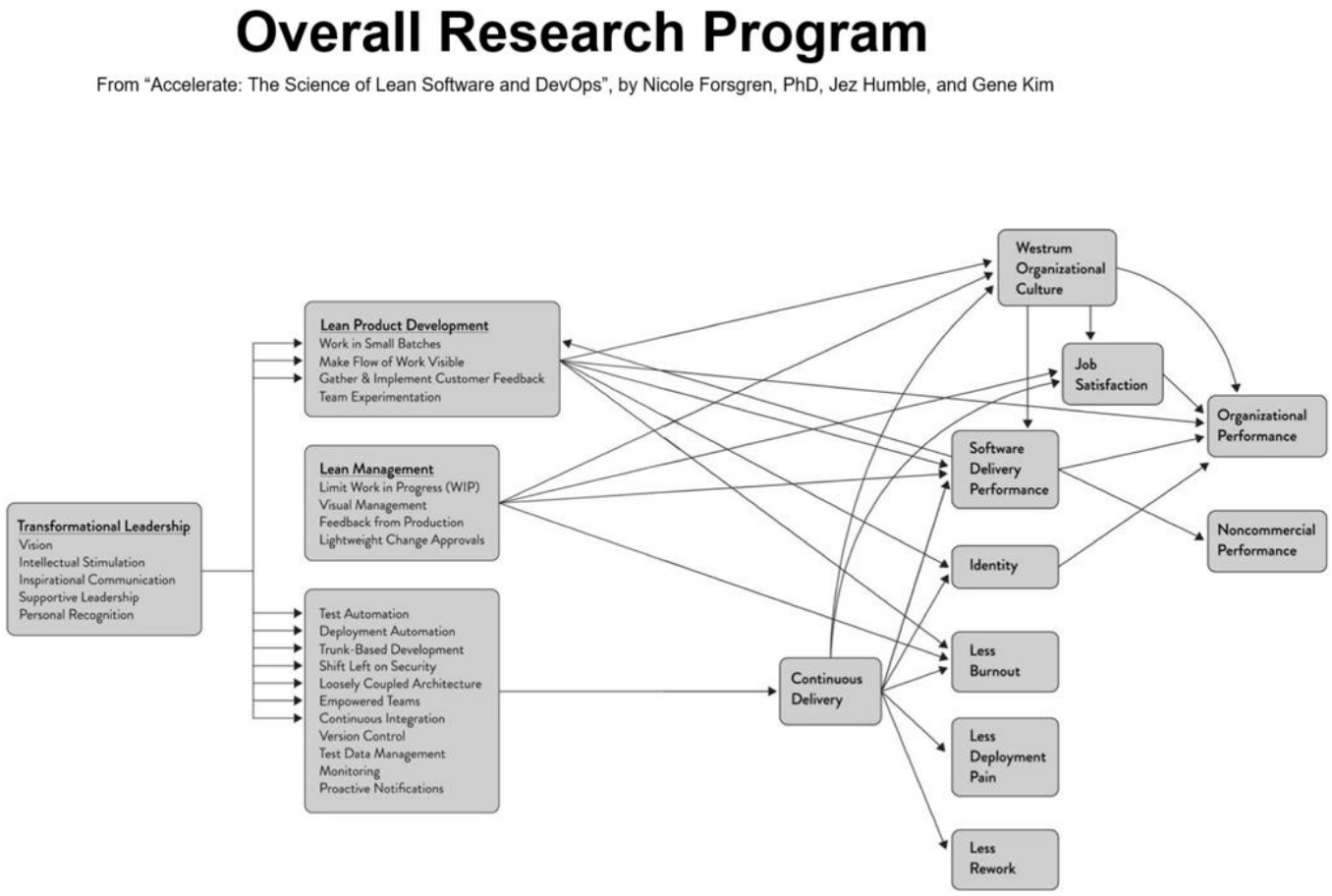
ANSSI Bonnes Pratiques, https://www.ssi.gouv.fr/administration/bonnes-pratiques/

TABLE DES ILLUSTRATIONS

TABLES DES ILLUSTRATIONS

ANNEXES

ANNEXE N°1 - EXTRAIT DU LIVRE “ACCELERATE: THE SCIENCE OF LEAN SOFTWARE AND DEVOPS”, BY NICOLE FORSGREN, PHD, JEZ HUMBLE, AND GENE KIM



High-Performance Team, Management, and Leadership Behaviors and Practices by Steve Bell and Karen Whitley Bell

	Team Practices	Management Practices	Leadership Practices
Culture	*Foster generative culture	*Foster generative culture	*Foster generative culture
	*Build quality in, continuously measure and monitor	*Focus on quality, protect teams to ensure quality	*Focus on quality, protect teams to ensure quality
	Focus on promoting organizational learning	Focus on promoting organizational learning	Focus on promoting organizational learning
		*Provide teams with time for improvement and innovation	*Provide teams with time for improvement and innovation
Organizational Structure			*Align, Measure and Manage to Flow (matrixed, cross-functional value stream organization structure)
		Establish small, cross-functional, multi-skilled teams; support bridging structures so teams can easily communicate and collaborate	Enable and support cross-skilling to reduce expert-dependent bottlenecks, and form communities of expertise
			Establish and support internal coaches and the appropriate infrastructure to scale and sustain them
Direct Learning and Alignment to Value	*Engage, learn from, and validate with customers (Gemba)	*Engage with and learn from customers and teams (Gemba)	*Engage with and learn from customers, teams, supply chain partners, and other stakeholders (Gemba)
	*Understand & visualize customer value, - identify measurable targets for quality	*Understand & visualize customer value, - identify measurable targets for quality	*Understand & visualize customer value, - identify measurable targets for quality
	* Practice creativity as part of overall work	*Practice creativity as part of overall work, encourage team members to utilize this time to learn and innovate	*Budget for and allocate time for creativity (i.e. Google's 20% target)
Strategy Deployment	*Visualize team goals and targets, understand how these targets advance enterprise strategy	Help teams to set and visualize goals and targets, understand and communicate how these advance strategy (catch ball)	Practice strategy deployment, visualize all goals, and near-term targets, communicate this clearly to managers and help them set appropriate targets and initiatives
	*Actively monitor and visualize performance to goals/targets	*Actively monitor and visualize performance to goals/targets	*Actively monitor and visualize performance to goals/targets
			Eliminate unnecessary controls, invest instead in process quality, and team autonomy and capability (*Teams that reported no approval process or used peer review achieved higher software delivery performance)

Improve flow through analysis and disciplined problem solving	Visualize & analyze work flow, identify obstacles to flow, (Process/Value stream mapping & analysis); *understand the connection between the work they do and its positive impact on customers	Visualize and analyze work flow, identify obstacles to flow, (Process/Value stream mapping & analysis) help teams understand how they support larger value stream	Visualize and analyze overall value stream flows (enterprise architecture), identify systemic obstacles to flow, prioritize and support mapping and analysis of lower level supporting flows
	Prioritize obstacles to customer value and experience, and team targets and goals	Prioritize obstacles to customer value and experience, and team targets and goals	Prioritize systemic obstacles to flow
	Apply disciplined problem solving to prioritized problems, analyze to identify root causes	Apply disciplined problem solving to prioritized problems, analyze to identify root causes	Apply disciplined problem solving to complex systemic issues to identify strategic improvement themes and targets (strategy deployment), apply learning to update standard work
	Escalate cross-functional and systemic problems	Coordinate cross-functional problem solving, solve or escalate systemic problems	Cascade prioritized problem solving targets to the appropriate stakeholders through catchball PDCA
	Form hypotheses about root causes, design and conduct controlled experiments, measure results, communicate learnings, repeat if needed, incorporate improvements into	Form hypotheses about root causes, design and conduct controlled experiments, measure results, communicate learnings, repeat if needed, incorporate improvements into	Learn from organization-wide PDCA cycles, and repeat learning/improvement cycles
Way-of-Work Rhythm & Routine	*Visualize, measure and monitor work flow, monitor for deviations, respond to deviations appropriately	*Visualize, measure and monitor work flow, monitor for deviations, respond to deviations appropriately	*Visualize, measure and monitor work flow, monitor for deviations, respond to deviations appropriately
	*Break demand into small elements (MVP's) and release regularly and often		
	*Visualize Demand, WIP, and "Done" (Kanban)	*Visualize Demand, WIP, and "Done" (Kanban)	*Visualize Demand, WIP, and "Done" (Kanban)
	*Minimize and visualize WIP	*Minimize and visualize WIP	*Minimize and visualize WIP
	Prioritize demand to goals and targets	Prioritize demand to goals and targets	Prioritize demand to goals and targets
	Develop & practice team standard work (rhythm & routine)	Develop & practice leader standard work (rhythm & routine)	Develop & practice leader standard work (rhythm & routine)
	Conduct daily stand-ups with standard routine, escalate obstacles as needed (catchball)	Conduct daily stand-ups with team leads, standard routine, resolve or bridge/escalate obstacles as needed (catchball)	Conduct stand-ups with direct reports with standard routine on a regular cadence, resolve escalated obstacles (catchball)
	Support team and peer learning	Coach team members; support team learning	Coach managers, have your own coach
	Conduct regular cadence of retrospectives (work, and way of work)	Conduct regular cadence of retrospectives (work, and way of work)	Conduct regular cadence of retrospectives (work, and way of work)

ANNEXE N°2 - TABLEAU DES CYCLES DE VIE WINDOWS MICROSOFT

Systèmes d'exploitation serveurs	Service Packs précédents	Service Pack actuel et date de disponibilité
Windows Server, version 1909 (Datacenter, Standard)	Non applicable	12 novembre 2019
Windows Server, version 1809 (Datacenter, Standard)	Non applicable	13 novembre 2018
Windows Server 2019 (Datacenter, Essentials, Standard)	Non applicable	13 novembre 2018
Windows Server, version 1803 (Datacenter, Standard)	Non applicable	30 avril 2018
Windows Server, version 1709 (Datacenter, Standard)	Non applicable	17 octobre 2017
Windows Server 2016 Datacenter, Essentials, Standard)	Non applicable	15 octobre 2016
Windows Storage Server 2016	Non applicable	15 octobre 2016
Windows Server 2012 R2	Non applicable	18 octobre 2013
Windows Server 2012	Non applicable	30 octobre 2012
Windows Server 2008 R2	Non applicable	SP1 22 février 2011
Windows Server 2008	SP1 (Windows Server 2008 a été commercialisé avec le SP1)	SP2 20 avril 2009

Source: https://support.microsoft.com/fr-fr/help/13853/windows-lifecycle-fact-sheet

ABSTRACT

How do technical choices, in the short term linked to strong constraints, impose a rationalization and standardization of an IS to migrate towards a long term vision?

In every SME, whose product or service offered is based on IT, while growing exponentially, comes the moment when, in order to cope with this growth, one encounters more or less strong constraints, difficulties in ensuring the same level of service quality for one million customers as for one hundred thousand.

Today the information system is a vital element of a society. It is all the more important when the business model is based on a 100% digital service.

The information system brings together all the human, hardware and software resources that enable the creation, processing, storage and delivery of information through processes or services.

Let's first understand how development teams work on different technical environments and do not use the same development methods.

In order to standardize their methods with devops technologies and methods

In an ever-changing environment, it is important to understand the cyber issues related to IT as well as the legal obligations we must respect. Especially in organizations of vital interest where these obligations are very strict, such as the banking sector.

In conclusion, this work aimed to understand how to take technical decisions in the short term, which was itself linked to strong constraints, requiring the adaptation of choices in the long term. First of all, we have seen the constraining context in which Nickel has evolved over the last few years. This enabled us to identify areas for improvement to rationalize our information systems, and thanks to the implementation of the devops culture within the development teams, we have been able to automate the various stages of development. By automating these steps, this has raised the question about the application of security policies.