Les Niveaux de Service en Environnement Hybride

VERSION ANONYMISEE

Table des matières

Introduction

L'informatique traditionnelle a pour vocation de supporter les besoins d'informatique du métier appelé *Business*. Là où anciennement les applications s'exécutaient sur des systèmes d'exploitation directement installés sur le matériel (*hardware*), l'arrivée de la virtualisation a elle permis de rajouter une couche d'abstraction entre le système d'exploitation (*Operating System* : *OS*) et le matériel. Un serveur ne signifie plus un seul système d'exploitation : il devient possible de compartimenter les applicatifs sur des systèmes d'exploitation différents, tous se partageant les ressources matérielles d'un même serveur. C'est cette mutualisation locale (sur une machine / serveur) qui permet de dimensionner plus efficacement les machines en utilisant la capacité moyenne plutôt que le scénario le plus pessimiste.

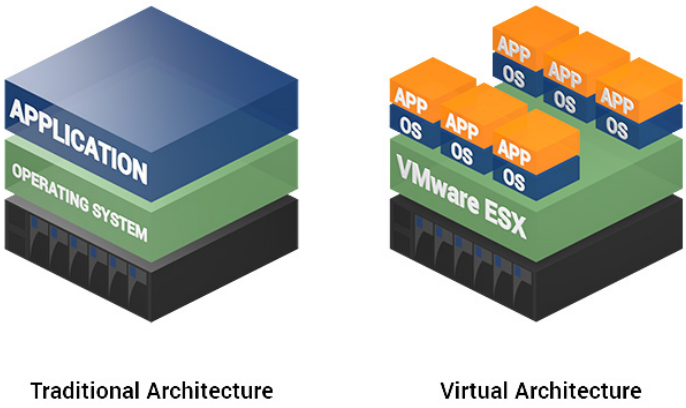


Figure 1 : architecture virtualisée et non virtualisée[1]

Cette technique n'est pas récente et n'a pas toujours eu exactement la même fonction. La virtualisation a un intérêt fort très tôt dans l'histoire de l'informatique pour les contextes de Système d'Information (SI) portés par les *mainframes*. Un *mainframe*, composant fort onéreux, capable de fournir une puissance de calcul phénoménale, a besoin d'être fragmenté pour profiter de multiples *OS*. Lorsque les systèmes centralisés (*mainframe*)ont perdu le vent de poupe au profit de la multiplicité de plus petits systèmes sous x86 (aussi appelés systèmes distribués), la virtualisation a changé de rôle et se transforme en une interface pour l'industrialisation de ferme de serveur via la mutualisation des machines.

[1] http://www.moderndata.com/wp-content/uploads/2014/09/virtual-architecture.jpg

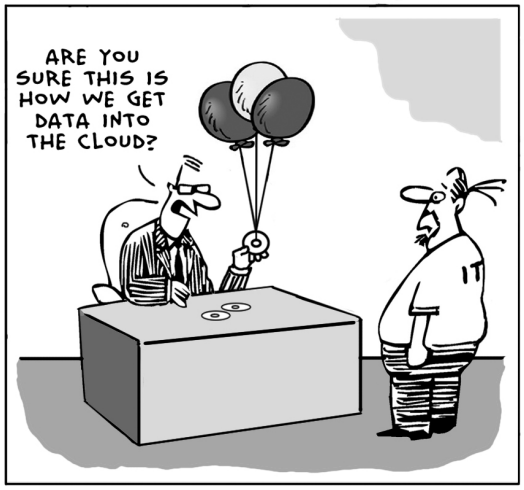


Figure 2 : gag [2]

Le *cloud* « est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement internet »[3].

Pourquoi faire chez vous, ce qu'on peut faire chez le voisin dont c'est le métier ? C'est ce que les fournisseurs de *cloud* nous vendent. Ils reprennent le principe de mutualisation qu'apporte la virtualisation et le pousse jusqu'au contexte d'industrialisation. La puissance de calcul est mutualisée puis décomposée et enfin vendue au client à travers différents types de services.

C'est un changement complet de paradigme. Anciennement, on ne tirait profit que de solutions locales par l'utilisation de puissance de calcul via le *hardware*. L'arrivée du *cloud* change radicalement la donne en transformant les acquisitions *softwares* et *hardwares* par des abonnements à des services.

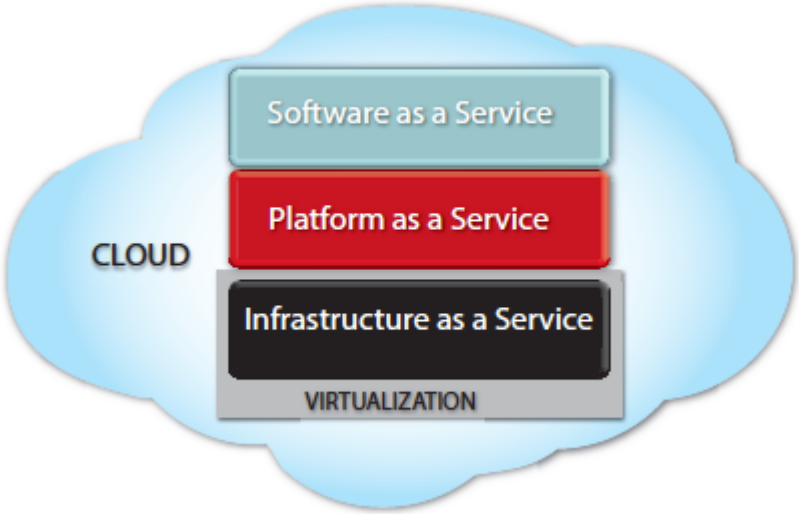


Figure 3 : types de service [4]

[2] https://fr.pinterest.com/pin/553168766700652428/
[3] https://fr.wikipedia.org/wiki/Cloud_computing
[4] https://support.rackspace.com/white-paper/virtualization-is-not-the-cloud/

Ces services, souvent dénotés aaS, peuvent concerner plusieurs briques de l'architecture : la mise à disposition d'un système d'exploitation, la mise à disposition d'une plateforme logicielle, ou uniquement l'utilisation d'un service spécifique rendu par un logiciel.

L'arrivée de ces services à travers le cloud plonge la Direction des Système d'Information (DSI) dans des dilemmes d'infrastructure : comment profiter des avantages du *cloud* (notamment la flexibilité) sans en assumer les inconvénients (notamment les problèmes liés à l'externalisation) ? L'alternative la plus commune est une combinaison de ces solutions appelée *cloud hybrid*.

Avant même de rentrer dans les détails techniques des solutions *clouds*, l'idée de transformer l'informatique traditionnelle vers des solutions de type *cloud* modifie nécessairement les manières de s'assurer des niveaux de services. La distanciation prise par l'utilisation de *IaaS*, *PaaS*, *SaaS* doit continuer à pouvoir rendre compte des services rendus. Ces services sont encadrés par les objectifs définis par le *Business*. Cette question m'amène directement à la problématique de mon sujet de stage : Comment est-il possible d'assurer les niveaux de service dans les environnements hybrides ?

Des solutions traditionnelles aux solutions Clouds

Dans cette partie, j'aborderai plus en détails les types d'architectures : des solutions traditionnelles à celles du *cloud* avec leurs interactions.

Informatique traditionnelle

L'informatique traditionnelle est déployée chez soi. Elle rassure les entreprises par le contrôle et les responsabilités qu'elle apporte. Que ce soit la partie *software* (applicatif) ou *hardware* (réseau, serveurs), avoir l'ensemble du SI sous sa responsabilité est gage de sécurité et de solutions fortement paramétrables. C'est historiquement la solution majoritairement adoptée par les entreprises avant les années 2000.

Détenir les clefs de toute son infrastructure oblige les sociétés à y investir temps, énergie et argent, tout cela à travers un service informatique dont l'importance sur le *Business* peut être crucial. Cependant, les entreprises ont, dans de rares cas, un cœur de métier tourné vers l'informatique. Même si fondamentale, l'*IT* (*Information Technology*) ne fait office généralement que de support métier. C'est en ce sens que le *Business* décide parfois de se tourner vers l'infogérance, c'est à dire vers un prestataire extérieur qui se charge de la gestion du SI. Ces solutions d'infogérance, dont les solutions *clouds* font parties, délèguent certaines fonctions du système d'information à des entreprises dont le cœur de métier est de gérer les problématiques IT et qui donc, par essence, peuvent prétendre à des solutions nouvelles, plus économiques et répondant plus rapidement à la demande d'utilisateurs chez le client. L'infogérance traditionnelle apporte avec elle de multiples inconvénients : lenteurs des exécutions entreprises, solution coûteuse à la fois en coût d'acquisition (CAPEX) mais surtout en investissement sur le temps (OPEX).

De manière traditionnelle l'utilisateur Alexandre effectuait une demande de création d'une machine virtuelle (*Virtual Machine* : *VM*), l'administrateur (infogéré ou non) devait ensuite valider et créer la *VM*. Le *Time to Market* (le temps moyen écoulé entre la génération d'une idée et sa

commercialisation[5]) long et largement améliorable est l'un des éléments qui a poussé l'émergence de solution *cloud*.

Définition du cloud

Quel que soit la solution *cloud* adoptée, celle-ci repose sur quatre piliers fondamentaux que les fournisseurs (*providers*) s'engagent de livrer au client :

- Les ressources matérielles (*hardware*) : serveurs, réseaux, etc. ;
- Une facturation à l'usage (*pay as you go*) : plus j'utilise les ressources, plus je paye ;
- Une autonomie (*self-service*) : à travers un portail, je suis client, je peux faire la demande la création d'une *VM* ;
- De l'agilité via l'automatisation (*provisionning*) : la *VM* que le client a demandée va être créée automatiquement, sans passer par un administrateur.

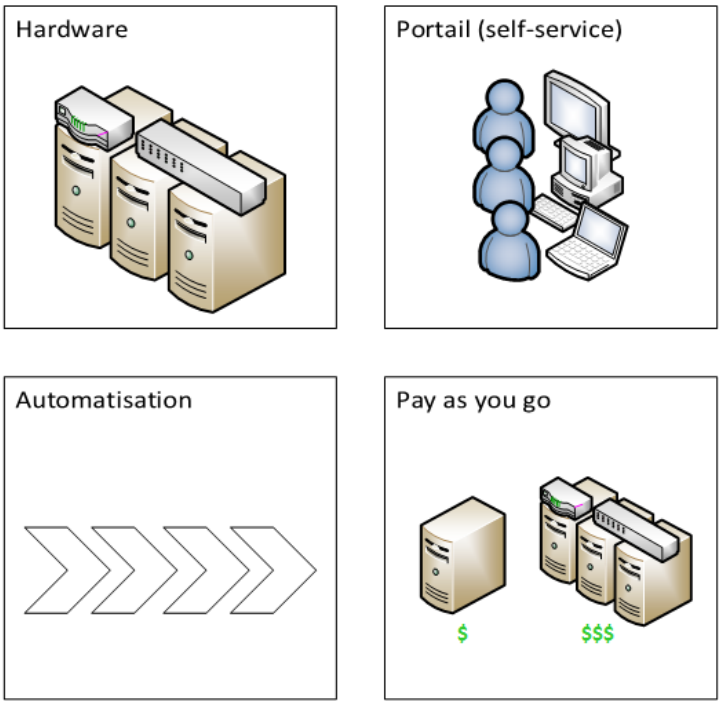
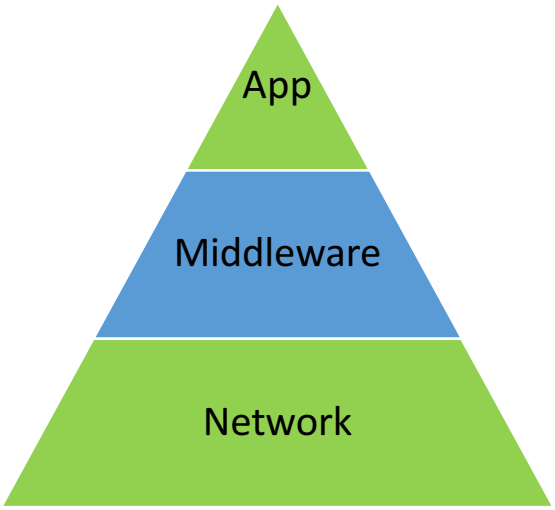


Figure 4 : piliers fondamentaux nécessaire pour le Cloud

Ces quatre piliers permettent la mise à disposition de services. D'où tous les acronymes de type *aaS*. La notation *aaS* signifie *as a Service* est inhérente à l'arrivée des solutions *clouds*. Tous les types de service ne sont pas compatibles avec tous les types de *cloud*. Dans le cadre de *cloud* public, *IaaS*, *PaaS* et *SaaS* existent. Dans le cadre du cloud privé, le *PaaS* et *SaaS* n'ont pas lieu d'exister : nous le verrons en détail plus loin dans ce document mais s'il était possible de faire du *PaaS* en *cloud* privé, cela voudrait dire que nous serions responsables de la partie applicative et réseau mais pas de la partie *middleware* (intergiciel). Ce genre de saut de responsabilité n'est pas possible (voir schéma ci-dessous).

[5] http://www.inficiences.com/fr/ge/ressources/strategie-produit-services/delai-sur-marche-time-market



	Public	Privé
IaaS	☺	☺
PaaS	☺	☹
SaaS	☺	☹

Figure 5 : pyramide et matrice de dépendance des couches

IaaS

C'est l'abréviation d'*Infrastructure as a Service*. Cela permet la location d'une infrastructure virtualisée (ou non, appelée *bare metal*). En mots plus simples, pour le client, c'est la mise à disposition d'un système d'exploitation qu'il aura choisi et de performances qu'il aura calibrées. Cela évite d'acheter un ensemble de matériel pour installer une infrastructure. On se contente de louer du matériel virtuel comme s'il nous appartenait et y installer les serveurs virtuels que nous souhaitons utiliser. L'installation et la mise à jour des applicatifs est sous la responsabilité du client. Des exemples d'utilisation concrète de *IaaS* est la création de *VM* sur Microsoft Azure ou via Amazon Web Service (AWS).

PaaS

C'est l'abréviation de *Platform as a Service.* Le client accède à des *middlewares* préinstallés sur un système d'exploitation. Le *PaaS* offre un cadre de travail permettant une rapide utilisation et customisation d'applications. Cela permet donc d'héberger des applications plus adaptables face à des solutions clefs en main qu'offre le *SaaS*. Des exemples récurrents de l'utilisation de *PaaS* dans les entreprises sont des services de base de données, serveur web (apache) ou de serveur d'application (jboss, tomcat, etc.).

SaaS

C'est l'abréviation de *Software as a Service*. Le client paye l'utilisation d'un logiciel sans se soucier de son installation, de sa configuration, de sa maintenance, etc. Le *SaaS* passe par la plus grande délégation : *applications*, *runtime*, *data*, *middleware*, *OS*, *virtualization*, *storage*, *servers*, et *networking*. Un exemple concret de l'utilisation du *SaaS* est l'*emailing*. La frontière entre le *PaaS* et le *Saas* peut s'avérer difficile à saisir, une manière de les distinguer est de voir le *PaaS* comme la mise à disposition d'outils intermédiaires à la différence de *SaaS* qui répond à un service pour une utilisation finale. Le service rendu par l'*emailing* est l'envoi et la réception de mail.

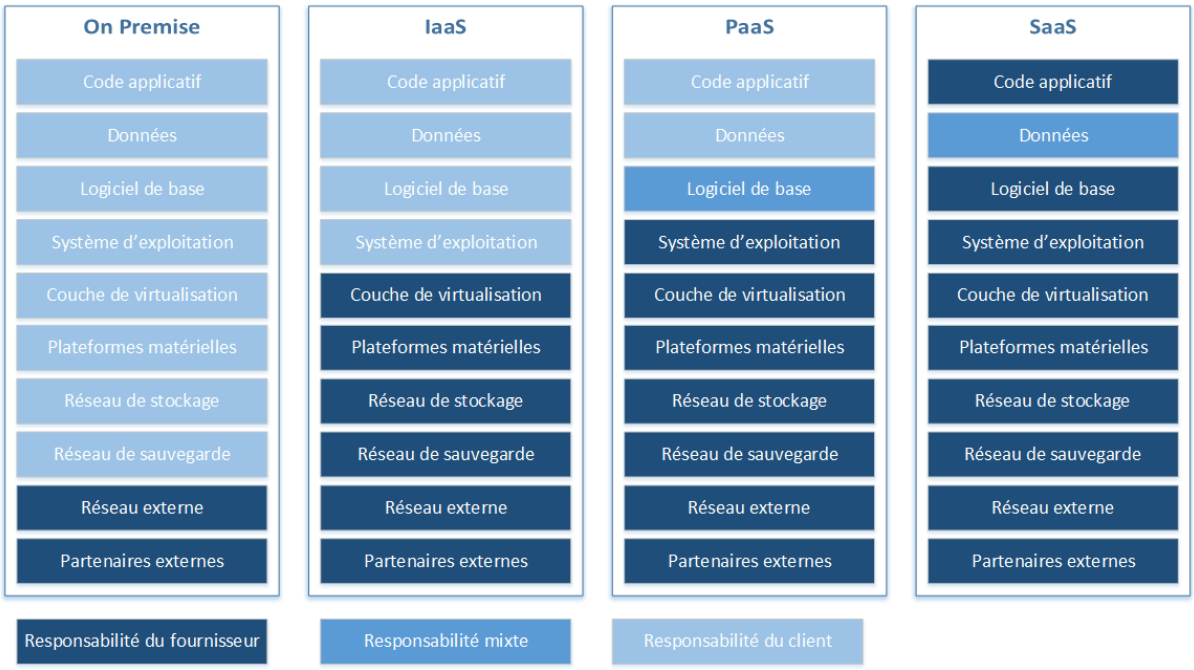


Figure 6 : tableau récapitulatif des briques dont chacune des solutions est responsable

Pour récapituler, les *aaS* sont des services de granularités différentes permettant d'avoir plus ou moins de flexibilité (le *IaaS* en a plus que le *PaaS* qui en a plus que le *SaaS*). Tout l'enjeu réside au bon dosage des responsabilités engagées par le fournisseur et de la facilité / rapidité de la mise en exploitation des services. Ces types de service sont apparus à l'émergence du *cloud* afin que les utilisateurs puissent choisir quelles briques du système d'information ils souhaitent déléguer.

Cloud

Le cloud tire parti de la mutualisation des ressources, ce qui lui donne cette capacité à tenir les montées en charge : pic de demande de ressources processeur (*CPU*) et mémoire vive (*RAM*), etc. En effet, la répartition de la charge des applications du client sur les machines d'un fournisseur est possible. Cette capacité à tenir la charge, appelée *scalability* (mise à l'échelle) dépend :

- de l'infrastructure du fournisseur. Celle-ci pouvant être *a priori* très grande ou, dans une moindre mesure, généralement plus grande que la capacité maximale dont l'application a besoin ;
- du design des applications qu'on souhaite "scalabiliser". Si la plupart des sites fournisseurs nous vendent une scalabilité d'exception, il ne convient pas de l'appliquer à n'importe quel usage. Seules les applications *Stateless* peuvent être scalabilisées horizontalement : augmentation des performances par l'augmentation du nombre de machines (voir schéma ci-dessous). Le design *stateless* d'une application permet de rendre toutes les requêtes d'utilisateurs indépendantes les unes des autres. Ainsi la requête A1 de l'utilisateur Antoine pourra être traitée par le serveur S1, la requête A2 du même utilisateur pourra être traitée par S2.

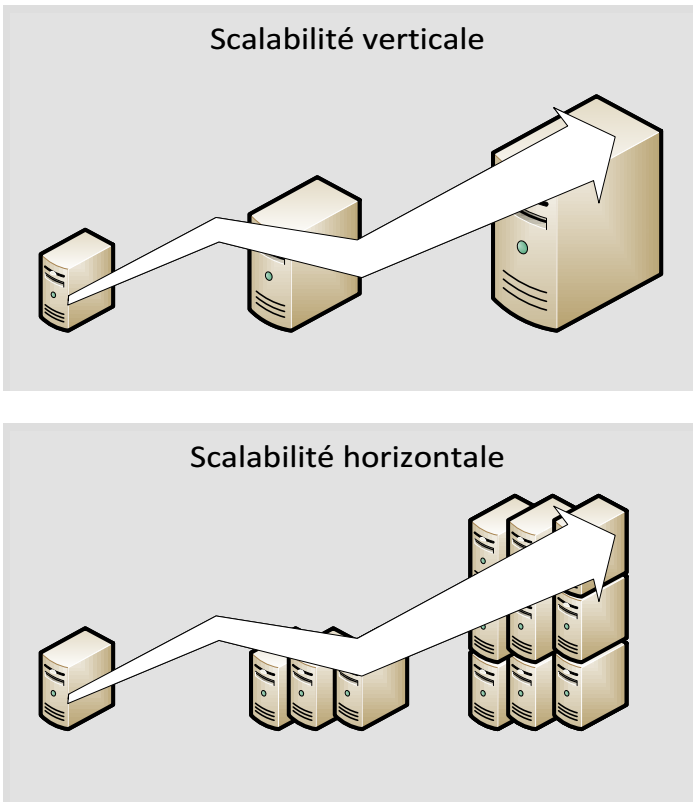


Figure 7 : différence entre scalabilité verticale et horizontale

Des questions liées à la sécurité surviennent alors : certaines entreprises sont frileuses d'opter pour de telles solutions dont l'architecture est partagée par plusieurs clients ou du fait que la solution soit externalisée. De ce fait, se sont développés plusieurs types de *cloud* répondant à différents cas d'usages.

Cloud public

La principale particularité du *cloud public* est la facilité d'accès aux ressources. Les serveurs sont accessibles via internet et possède une IP publique. L'utilisation est assez simple car aucune configuration réseau n'est requise au sein de l'infrastructure. De cette facilité d'utilisation, ces solutions se démocratisent de plus en plus dans les nouvelles sociétés. L'infrastructure à la demande permet d'être modulaire et proportionnée aux besoins métier. C'est d'ailleurs de cette manière que la facturation s'effectue : il ne devient pas nécessaire de devoir investir dans une infrastructure coûteuse, plus on stocke, utilise ou télécharge, plus la facture sera élevée (*pay as you go*). Les détails de facturation varient légèrement en fonction des fournisseurs mais nous aurons le temps de rentrer plus en détail sur les offres d'Amazon ou Microsoft dans une partie dédiée.

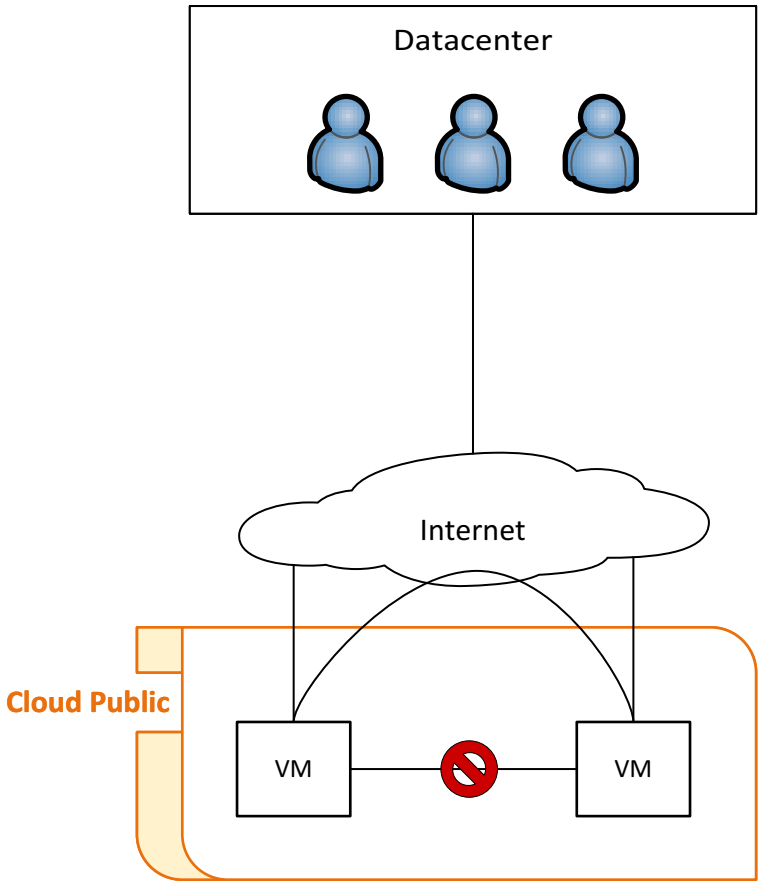


Figure 8 : l'accès des utilisateurs au cloud public

Certaines limitations comme l'impossibilité de gérer son réseau privé ou l'IP publique ne répondent pas à tous les cas d'usages de ce genre de *cloud*. L'IP publique apporte, certes, de la flexibilité mais ne répond pas toujours aux exigences de sécurité : faire transiter l'état de santé d'un serveur par internet n'est pas toujours envisageable dans un contexte client.

Les solutions clefs en main proposées par les *clouds publics* permettent un *Time to Market* rapide. L'envers de la médaille est que la solution ne rend pas forcément adaptée aux besoins logiciels. C'est pourquoi s'est développée une version plus paramétrable et dont l'accès est plus limité, appelée le *cloud privé*.

Cloud privé

La principale différence du *cloud privé* face au *cloud public* est l'accessibilité aux ressources : il n'y a généralement pas une IP publique. C'est un gage de sécurité, le *cloud privé* est moins exposé que son jumeau car pas relié directement aux réseaux publics. L'entreprise ou le collaborateur n'y ont accès uniquement par des liaisons louées ou éventuellement des connections chiffrées sur des réseaux publics.

Le contrôle est beaucoup plus complet sur les données et sur l'infrastructure d'une solution *cloud privé*. Un réseau dédié est mis à disposition permettant de plus grandes marges de manœuvre et de customisation des applications et de la sécurité de l'infrastructure.

Cette customisation rend le *cloud privé* plus ressemblant que le *cloud public* à l'informatique traditionnelle. Les entreprises ont adopté le *cloud* pour de multiples raisons : manque de confiance

des équipes en interne et aussi beaucoup par effet de mode. Or la mode a un prix et le *cloud privé* est un investissement moins risqué que son cousin public car il ressemble plus aux solutions traditionnelles : son adoption est douce et plus réaliste que le public.

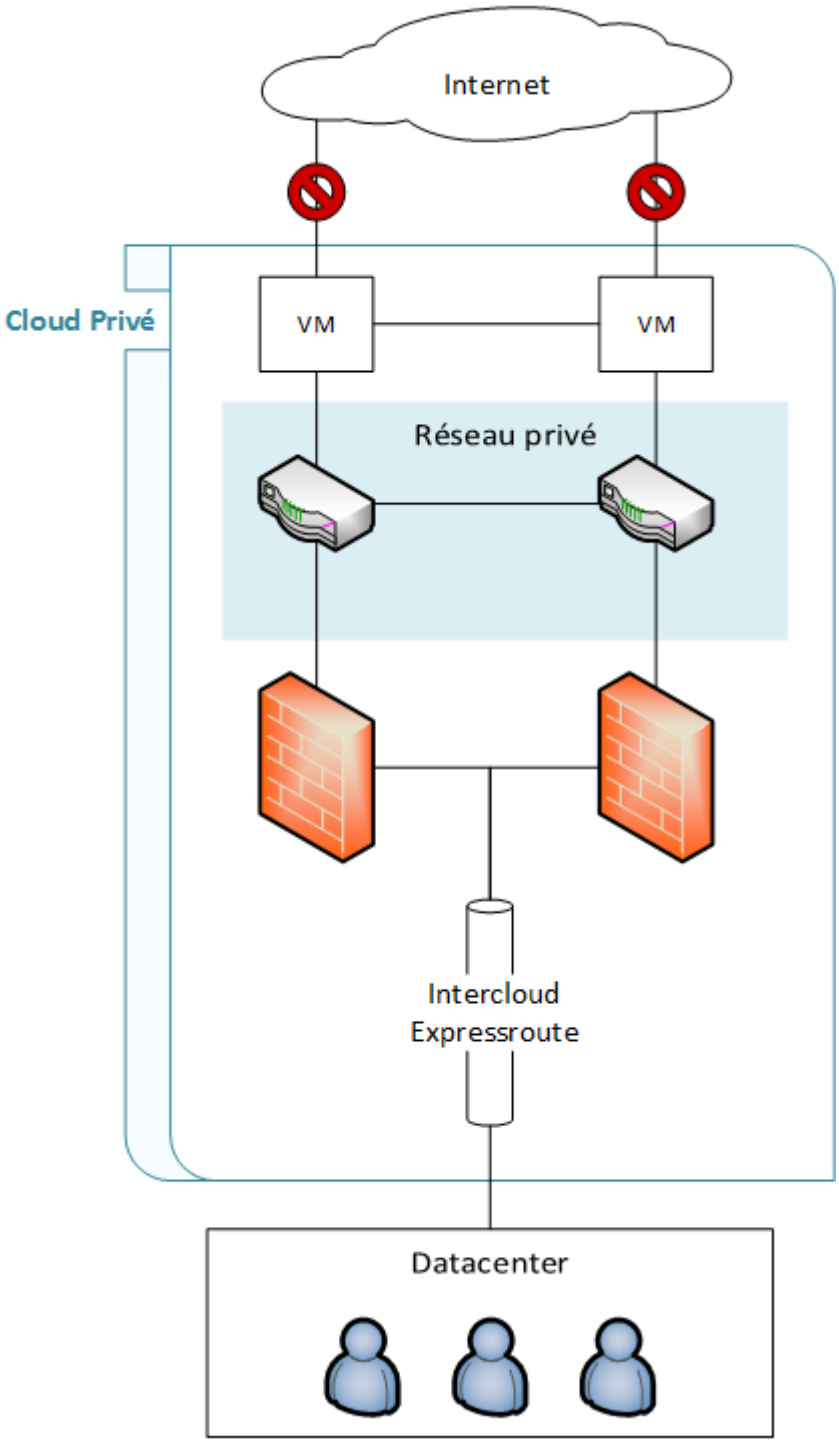


Figure 9 : l'accès des utilisateurs au cloud privé

Une confusion facile à faire est d'imaginer que le *cloud privé* permet d'avoir une infrastructure dont le matériel est dédié. La mutualisation du *hardware* existe toujours, même au sein d'un cloud privé. Sans mutualisation *hardware*, le *cloud*, quelle que soit sa forme, ne permettrait pas une bonne scalabilité. C'est le gagne-pain principal des *providers*, c'est pourquoi il faut bien comprendre que

”Cloud Privé” ne signifie pas “ressources physiquement dédiées”. Quand les providers parlent de *ressources dédiées* dans le cadre de *cloud privé*, ils parlent dans la majorité des cas de l’infrastructure réseau qui peut être paramétrée. Et pourtant, même elle, est virtualisée.

Une autre confusion possible vient du fait que les comparatifs trouvés sur internet mettent rarement face à face le *cloud privé* et *cloud public*. Il faut bien comprendre que le *cloud privé*, à la différence du *cloud public*, ajoute une complexité provenant de la gestion de l’infrastructure réseau, avec ses avantages et inconvénients :

Il n’y a finalement pas tant de différences entre les solutions de *cloud* privé et public, on trouve d’ailleurs beaucoup d’emmêlements dans les articles du net. L’adaptation des entreprises dans cette technologie est moins risquée car le cloud privé ressemble plus à l’infrastructure traditionnelle qu’au *cloud* public. C’est pourquoi certaines entreprises ont décidé de faire leur propre *cloud* privé.

Ce qui paraissait inhérent au cloud, à savoir que l’hébergeur et le client sont des entités différentes, ne fait pas partie des quatre fondamentaux décrivant le cloud : *cloud* privé *on Premise*.

Cloud privé on premise

La solution de *cloud privé on premise* rassemble les mêmes caractéristiques du *cloud privé* à la différence près que l’infrastructure est gérée par le client (*on premise* signifie “chez soi”). La différenciation peut s’avérer difficile mais rappelez-vous les quatre fondamentaux qui portent le *cloud*. Ils restent valables même lorsque la solution est hébergée chez le client. Le seul détail qui change véritablement est la facturation qui varie en fonction de la politique des entreprises (refacturation interne, suivis de consommation, etc.).

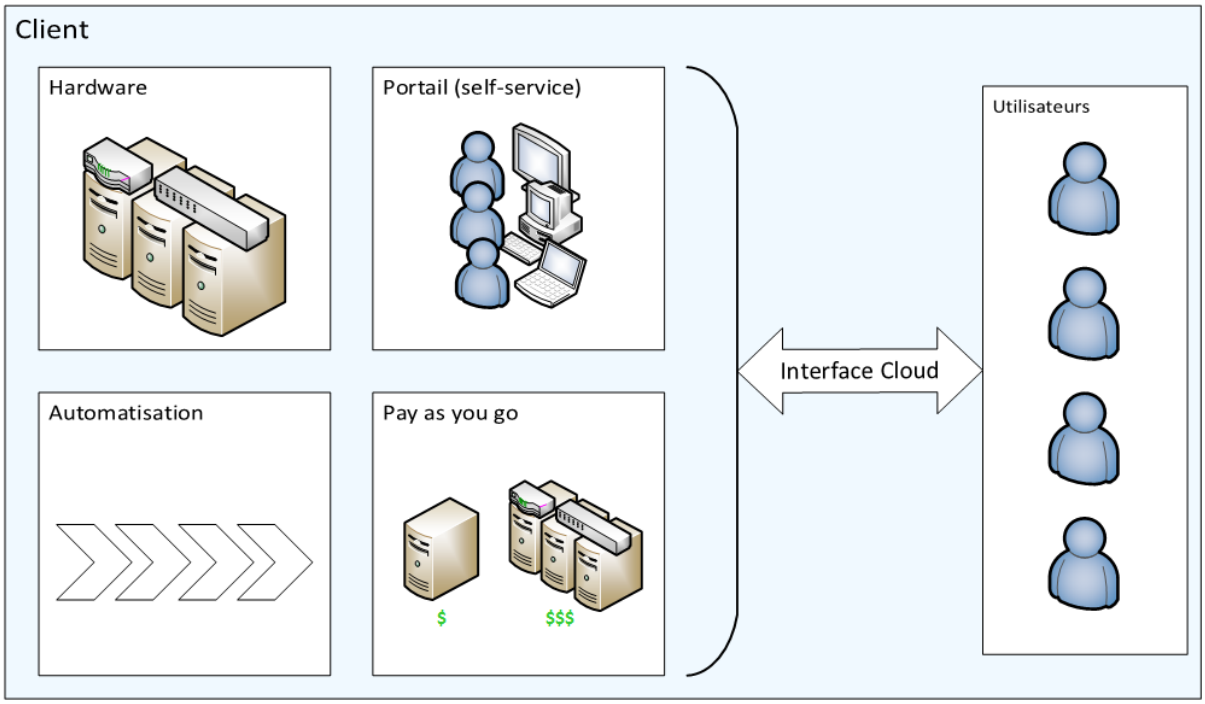


Figure 10 : l'accès des utilisateurs au cloud privé on Premise

Les avantages et inconvénients des solutions entièrement gérées par le client reviennent dans les solutions *cloud privé on premise*. La solution hébergée par le client apporte des lenteurs de déploiement, des coûts important d'acquisition et de maintenance. L’infrastructure ne bénéficie plus d’une aussi bonne scalabilité en comparaison à des hébergeurs dont les infrastructures sont plus

importantes. En revanche, la réversibilité (passage d'un fournisseur à un autre) n'est plus un problème, la customisation et les dépendances logicielles sont sous la direction du service informatique. La sécurité et les niveaux de services sont aussi à la charge du DSI.

Finalement, le *cloud privé on premise* n'est pas tellement plus qu'une informatique traditionnelle dont certains processus ont été améliorés par l'instauration des trois fondamentaux du *cloud* (*self-service*, *automatisation*, *pay-as-you-go*).

L'enjeu des entreprises est alors d'allier le meilleur de chacune des solutions afin que pour chaque cas d'utilisation, il soit possible de profiter des avantages en réduisant l'impact des inconvénients. Ce genre de mélange de solution est appelé *cloud hybrid*.

Cloud hybrid

La définition qu'on trouve sur internet du *cloud hybrid* est l'intégration des *clouds privés* et des *clouds publics* pour remplir différentes fonctions au sein d'une même organisation. Dans le cadre du sujet de ce stage, le *cloud hybrid* passe par l'utilisation simultanée de plusieurs de ses solutions, par exemple :

- *Cloud privé* + solution traditionnelle
- *Cloud public* + solution traditionnelle
- *Cloud privé* + *cloud public* + solution traditionnelle
- Etc.

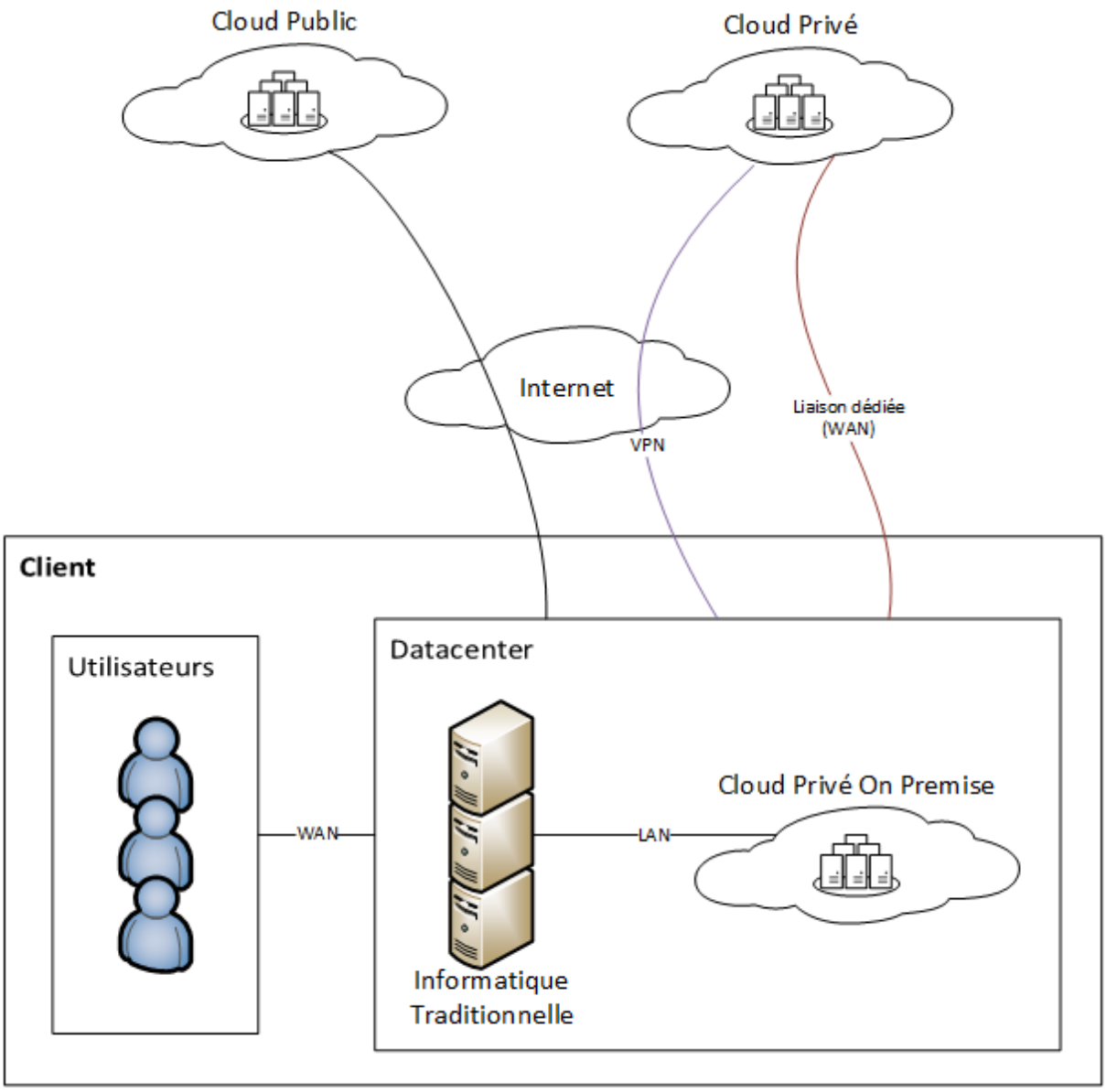


Figure 11 : relations entre les types de cloud

La mise en place d'une infrastructure hybride apporte une complexité supplémentaire dans l'étude des niveaux de services rendus par chacune des parties la composant. Ces problèmes surviennent notamment par le nombre d'acteurs importants intervenant dans la solution. En revanche, elle permet de se soustraire d'une bonne partie des obligations qu'apporte le cloud géré par un fournisseur externe : offres de service immuables, maintenances, etc.

Limitations des clouds (hors privé on premise)

La délégation à un prestataire extérieur ouvre nécessairement la porte à des dépendances multiples, à commencer par les maintenances ou les pannes qui peuvent arriver voire qui sont nécessaires et suspendent donc toutes ou une partie des activités. Aussi trivial que cela puisse paraître, la liaison (internet, dédiée, WAN, etc.) est une condition nécessaire : si une panne sur cette liaison se produit, l'accès aux applications du *cloud* est compromis.

L'accès aux ressources à distance amène avec lui des latences. L'emplacement des *datacenters* peuvent ne pas convenir aux besoins du client : il n'y a par exemple pas de datacenters Azure en Russie ou en Afrique (voir schéma ci-dessous). A titre d'exemple, la latence moyenne entre Paris et les datacenters d'Europe est inférieure à 100ms, celle avec l'Australie est de plus de 300ms[6]. Pour une application dont l'architecture hybride fait intervenir une composante sur Azure, il faut prendre compte de ce genre de latence qui peut intervenir.

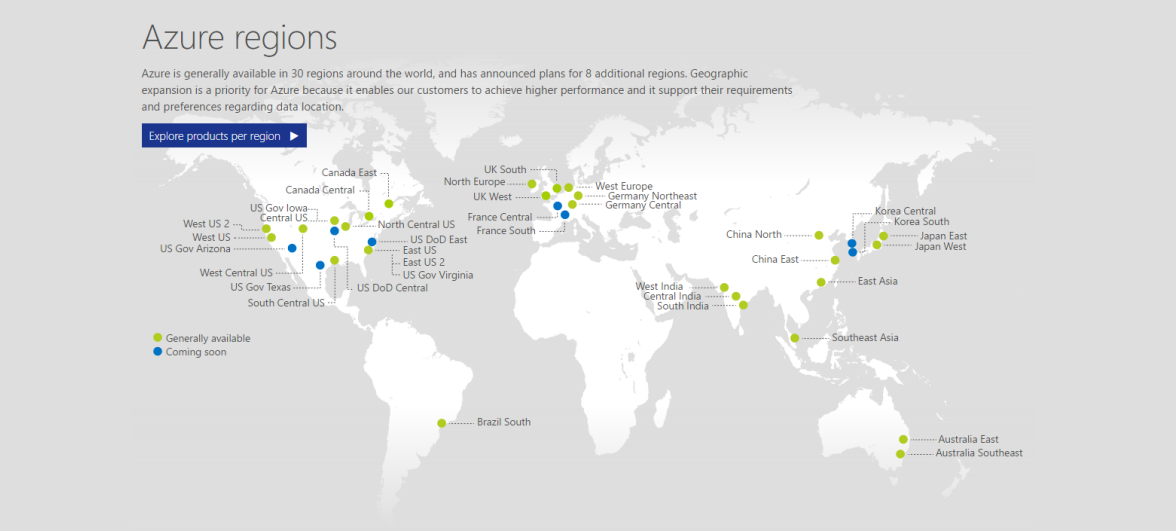


Figure 12 : présence des datacenters Azure dans le monde[7]

Ce plus certaines restrictions peuvent intervenir en fonction des pays sources. Un exemple réel de la limitation du *cloud* contextualisé internationalement : la Turquie, suite à des fuites de mails portant préjudices au gouvernement, a décidé de fermer les accès Dropbox, Google drive, OneDrive et GitHub[8].

D'un point de vu réglementaire, il existe des problématiques géographiques : certaines données doivent être hébergées uniquement sur le territoire, or même s'il est généralement possible de choisir la localisation des *datacenters*, les fournisseurs assurent des sauvegardes hébergées sur d'autres pays non communiqués. Il arrive aussi que la connaissance de l'ensemble des fournisseurs sous-jacents ne soit pas nécessairement connue. Pour ces raisons, ce genre de solution ne peuvent être optés pour certains cas d'usage comme par exemple les systèmes bancaires qui ont obligation de détenir les informations sur le territoire.

Concernant les performances, en général, rares sont les applications designées pour supporter la scalabilité horizontale (*stateless* par exemple). Or, si l'application nécessite une scalabilité verticalement (augmentation des ressources CPU/mémoire d'une seule machine) la remise en question de l'architecture portée par le *cloud* est à envisager. Cela pour plusieurs raisons :

- Augmenter le nombre d'*IOPS* (*Input/Output Operations Per Second*) n'est pas possible sans artifices au niveau de l'application (si tenté que l'infrastructure sous-jacente le permette) ;

[6] http://www.azurespeed.com/
[7] https://azure.microsoft.com/en-us/regions/
[8] http://www.frandroid.com/actualites-generales/382262_google-drive-onedrive-dropbox-github-bloques-turquie

- Augmenter la fréquence des *vCPU* (*Virtual CPU*) n'est pas envisageable, il se peut que l'on n'ait même pas connaissance des *CPU* physiques dont les *vCPU* proviennent.
- Augmenter le nombre de *vCPU* est possible mais n'est pas forcément pertinent. Les *vCPU* sont des cœurs virtuels simulés par l'hyperviseur, leurs nombres sont généralement supérieurs au nombre de cœurs physiques. Toujours dans une optique de mutualisation, l'augmentation des *vCPU* peut provoquer de la contention (voir figure ci-dessous) voire dans le pire des cas ralentir la *VM*. Enfin, les algorithmes d'ordonnancement *multi-vCPU* provoquent eux aussi des latences[9].
- Augmenter la *RAM* n'est pas non plus forcément significatif et cela pour même raison que la contention des vCPU (voir figure ci-dessous) : la *RAM* est elle aussi mutualisée et ne permet pas de répondre forcément à un pic de charge dans le pire des cas.

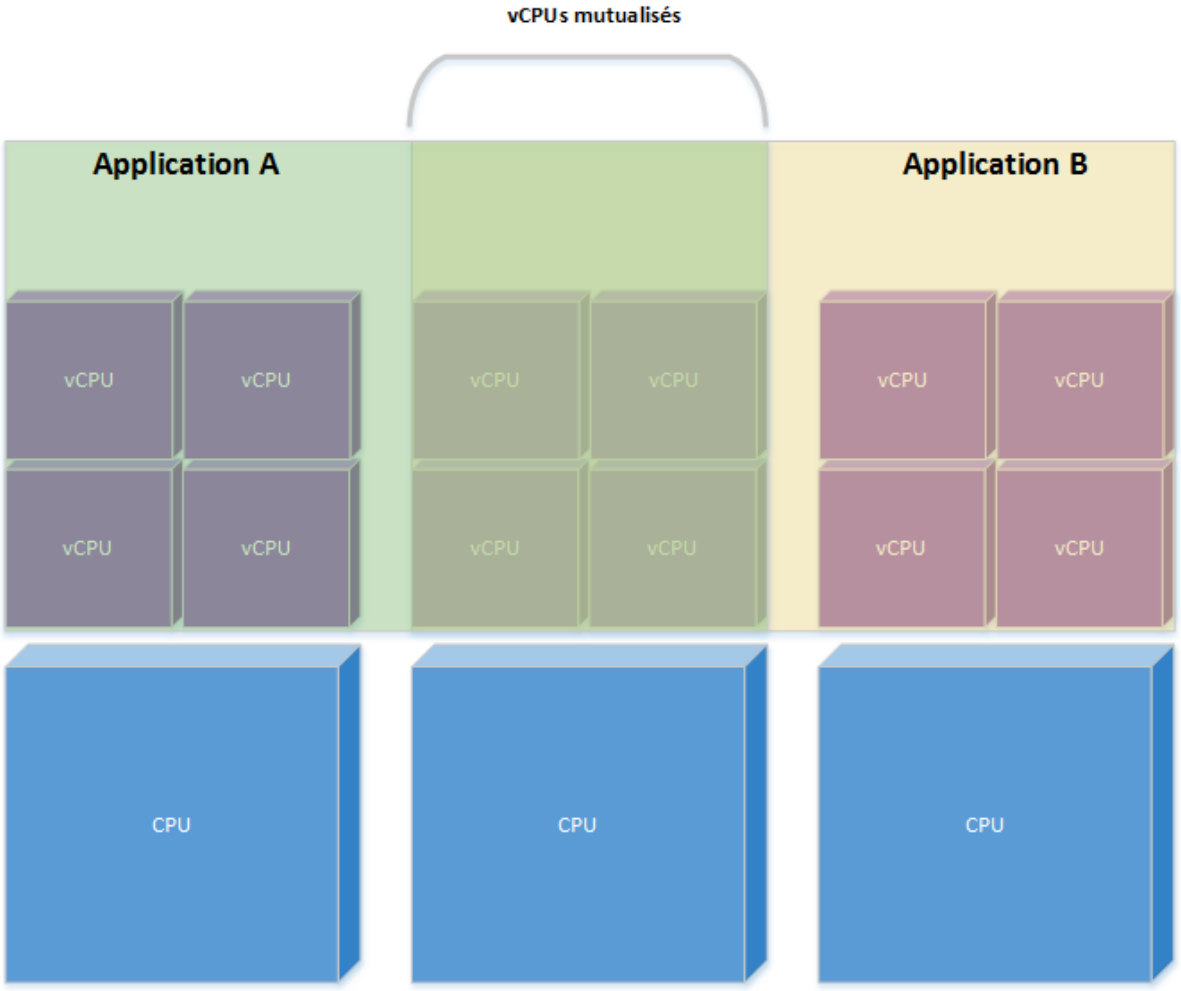


Figure 13 : contention des *vCPU*

[9] https://en.wikipedia.org/wiki/Gang_scheduling

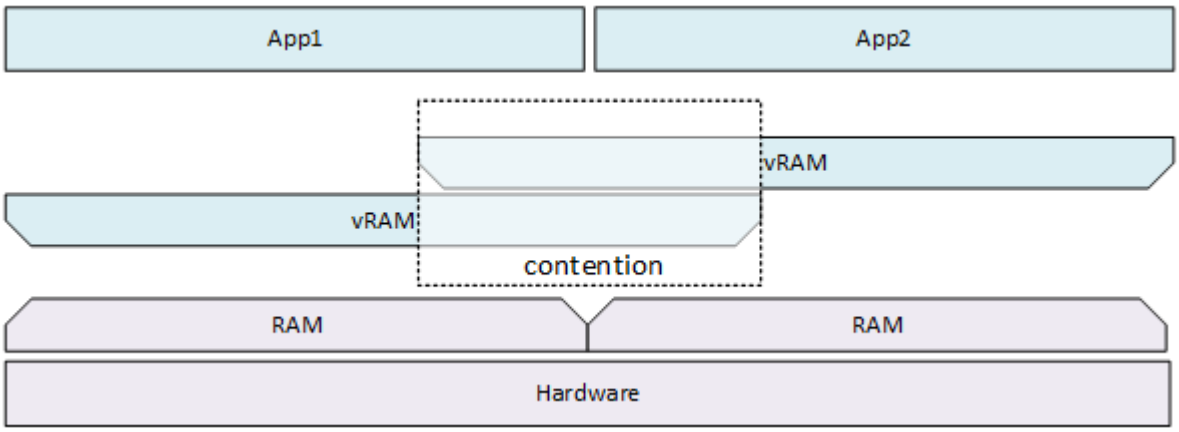


Figure 14 : Contention de la *RAM*

Enfin, il faut bien comprendre que se lancer dans le cloud implique que les services jusque-là rendus par la DSI deviennent la responsabilité des fournisseurs. L'utilisation de solutions hybrides optée par les entreprises génère quant à elle une architecture sophistiquée : la multiplicité des acteurs et des flux complexifie la lecture des performances, disponibilités, dépendances qu'une application peut rendre. Il devient important de savoir si les applications répondent bien aux exigences du *Business*, plus précisément aux niveaux de services décrits dans les *SLO*/*SLA*.

Indicateurs des niveaux de service

Définitions

Afin de pouvoir répondre correctement aux exigences du *Business*, il est important de comprendre comment elles sont décrites. Quelques définitions ci-dessous viennent directement d'ouvrages[10] en référence à *ITIL*, elles sont nécessaires pour comprendre les enjeux avant de s'engager dans le *cloud*.

SLO

Pour *Service Level Objectives* ou Objectifs de Niveau de Services. Ce sont les exigences du client. Prenons un exemple concret. L'entreprise Hé-Nokia souhaite faire une application de bibliothèque photo. Une exigence vis à vis du fournisseur qui hébergera les photos peut porter sur les performances d'accès à ces ressources. En des termes plus simples, l'exigence du client : "Je souhaite qu'en cas d'incident, je ne puisse pas perdre plus d'une journée de donnée".

SLA

Pour *Service Level Agreement* ou Contrat de Niveau de Service. C'est un contrat entre l'organisation informatique et les clients qui expriment en détails :

- Le type des services à fournir avec le niveau de qualité souhaité ;
- Les caractéristiques quantitatives telles que les performances et la disponibilité.

[10] http://www.itilfrance.com/

En des termes simples, les *SLA* permettent de se mettre d'accord sur les performances auxquelles l'application devra répondre. Par exemple la compagnie FWFT, au vue des exigences de l'entreprise Hé-Nokia, propose un accord stipulant : "OK pour l'accès des photos rapide, on vous propose 0.8ms".

Dans un monde idéal, les *SLA* offrent des performances supérieures ou égales au *SLO*, dans la vraie vie des discussions peuvent amener à faire bouger les exigences du client. Pour reprendre l'exemple précédent de manière vulgarisée :

Hé-Nokia (client) : "Je veux que mes photos soient accessibles de jour comme de nuit 7/7"
FWFT (fournisseur) : "On vous propose une disponibilité à 99% du temps pour 100€/mois"
Hé-Nokia (client) : "Je veux 100% !"
FWFT (fournisseur) : "On vous propose une disponibilité à 100% du temps pour 1000€/mois"
Hé-Nokia (client) : "Va pour 99% alors."

Normalement, les *SLO* se décident avant même la rencontre avec les fournisseurs. En effet, ceux-ci :

- en connaissance de cause, peuvent proposer des *SLA* en adéquations avec les besoins du client ;
- proposent des services encadrés/décris par des *SLA*. C'est alors aux clients de se soumettre aux contraintes que les contrats imposent (c'est le cas du *cloud*).

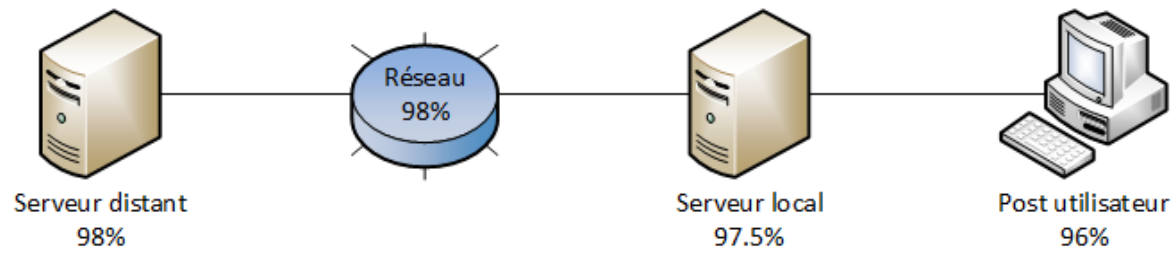
Les *SLA*/*SLO* font appels à des notions permettant de rendre compte la disponibilité d'un système, sa fiabilité. Ainsi, en connaissance de cause, il devient possible de faire correspondre une infrastructure capable de répondre à ces exigences. Dans l'optique de "cloudification", il devient important pour le client de formuler ses exigences afin qu'avec les *SLA,* il puisse faire valoir ses droits en cas de non-respect de celles-ci. Ces exigences peuvent être définies par :

Disponibilité (Availability)

La disponibilité, au sens *ITIL*, est l'aptitude d'un composant ou d'un service à remplir les fonctions requises à un instant donné ou sur une période donnée. Dans la plupart du temps, elle est exprimée en pourcentage.

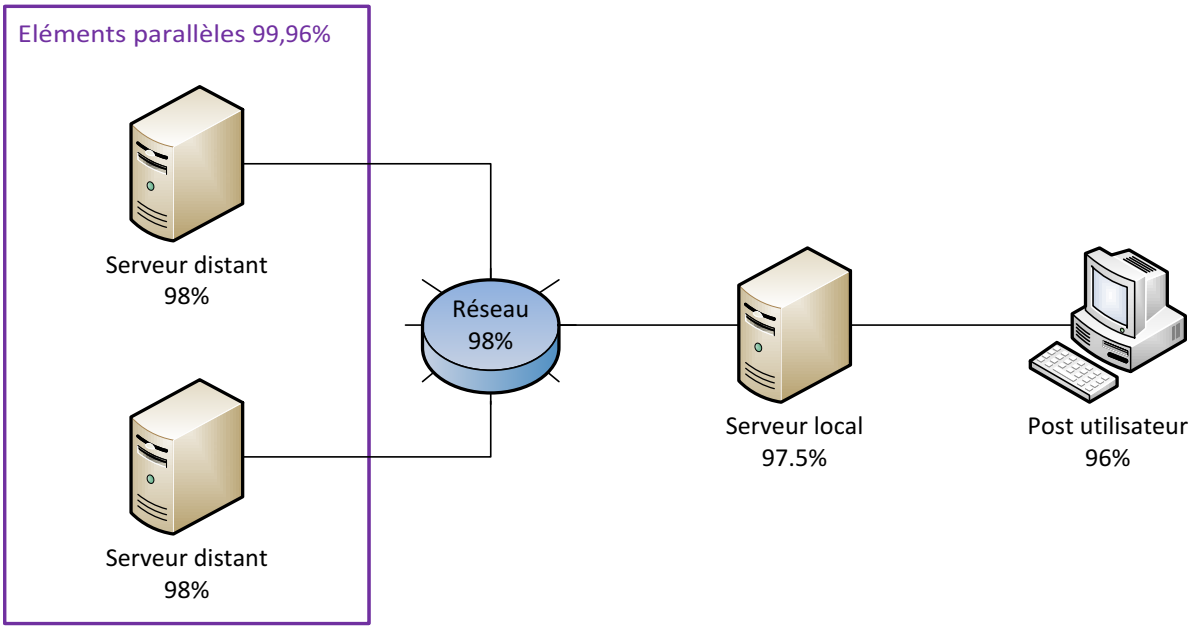
$$Disponibilité = \frac{Temps\ de\ service\ convenu - Temps\ d'indisponibilité}{Temps\ de\ service\ convenu} \times 100$$

Exemple de calcul de disponibilité d'élément en série : quand les éléments sont en série, la disponibilité de l'élément en bout de chaîne est le produit des disponibilités des éléments précédents.



$$Disponibilité\ totale = 0{,}98 \times 0{,}98 \times 0{,}975 \times 0{,}96 = 89{,}89\%$$

Exemple de calcul de disponibilité d'élément en parallèle :



$$p = 1 - ((1 - 0{,}98) \times (1 - 0{,}98)) = 99{,}96\%$$

$$Disponibilité\ totale = p \times 0{,}98 \times 0{,}975 \times 0{,}96 = 91{,}96\%$$

Fiabilité (Reliability)

La fiabilité, au sens *ITIL*, est l'aptitude d'un composant ou d'un système à se maintenir en fonctionnement (à ne pas tomber en panne). Elle est souvent mesurée et rapportée sous deux formes :

- Intervalle moyen entre les incidents de service (*MTBSI* ou *Mean Time Between Service Incidents*)

$$MTBSI = \frac{Temps\ disponible}{Nombre\ de\ coupures}$$

- Intervalle moyen entre les défaillances (*MTBF ou Mean Time Between Failures*). En générale

$$MTBF = \frac{Temps\ disponible - Temps\ Total\ d'Indisponibilité}{Nombre\ de\ coupures}$$

Maintenabilité (Maintainability)

La maintenabilité, au sens *ITIL*, est l'aptitude d'un composant ou d'un système à être maintenu ou rétabli en état de fonctionnement (rapidité de réparation). Elle est mesurée et rapportée sous la forme d'un délai moyen de restauration du service (*MTRS* ou *Mean Time to Restore Service*). Cette valeur est bien une moyenne, dans le meilleur des mondes, le *MTRS* doit être inférieur au *RTO*. Autrement dit, le *MTRS* donne la valeur moyenne du temps de réparation (des incidents qui se sont réellement produits) qui doit être inférieur aux objectifs de temps de réparation (les *RTO*).

$$MTRS = \frac{Temps\ d'indisponibilité}{Nombre\ de\ coupures}$$

Le temps moyen des temps de réparation (*MTTR* pour *Mean time to Repair*) ne s'utilise plus trop car décoré de l'expérience utilisateur : le composant peut être réparé sans que le service soit de nouveau disponible[11].

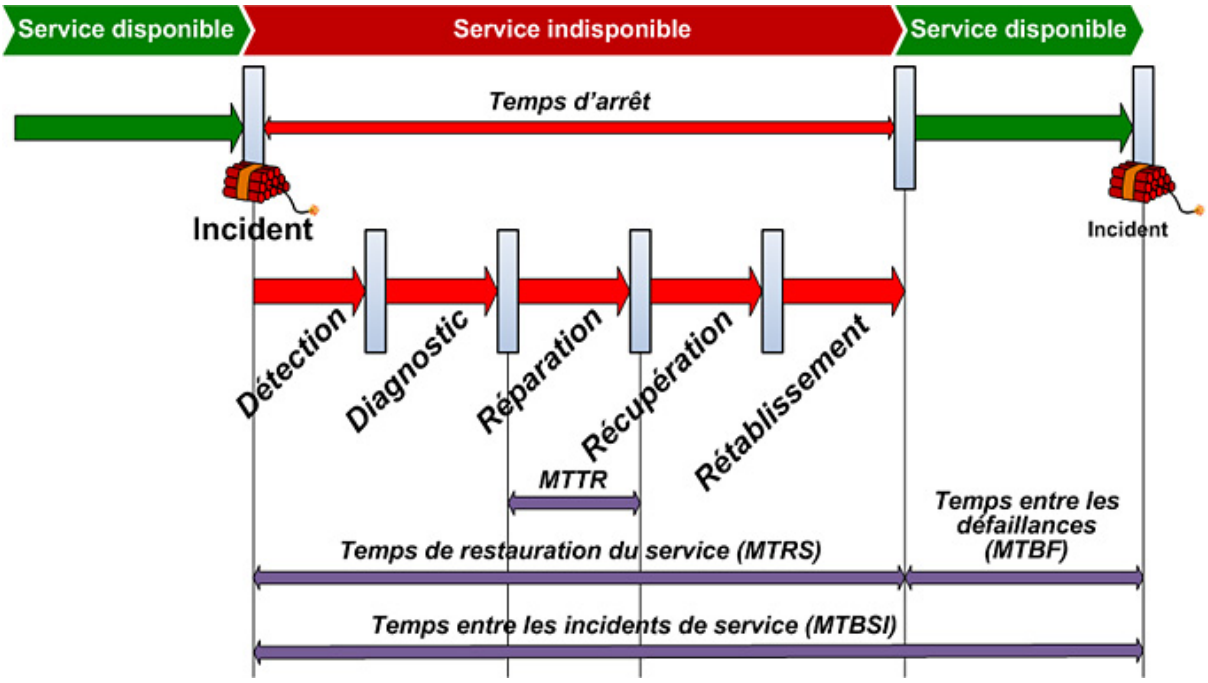


Figure 15 : récapitulatif des types d'indisponibilité[12]

RTO

Pour *Recovery Time Objective*, est la durée maximale d'interruption admissible (*DMIA*) est l'expression de besoin de disponibilité des différents métiers ou services, dans une organisation. Celle-ci peut varier en fonction de si l'on parle du point de vu métier, infrastructure ou applicatif. Dans des termes plus simple l'entreprise Hé-nokia souhaite, par rapport à ses besoins métiers, qu'en cas de panne sur sa base de données, le temps entre le début de l'incident (quand rien ne marche) et le début de la reprise d'activité soit de 8 heures. Ces 8 heures sont un RTO. Les RTO font partie des contraintes permettant la mise en œuvre d'infrastructure permet d'atteindre cet objectif de ces 8 heures.

RPO

Pour *Recovery Point Objective* :

"Temps associé à la fréquence de sauvegarde, par exemple, si le RPO est défini à 24 heures et que la volumétrie est faible, alors on peut considérer qu'une sauvegarde complète en fin de journée suffit. En revanche, si le RPO est très faible comme c'est le cas dans les secteurs de la banque ou des télécommunications, alors plusieurs sauvegardes seront

[11] http://blog.certification.info/2007/08/mttr-out-mtrs-in.html
[12] http://www.itilfrance.com/pages/docs/itilv3-02/img/11200-22-incident.jpg

nécessaires par jour, et en fonction de la volumétrie, différentes techniques de sauvegarde seront utilisées." - Wikipédia[13]

En d'autres termes, plus un *RPO* est long, moins régulièrement les sauvegardes sont effectuées. Il est à la fois plus économique mais plus dangereux d'augmenter le *RPO* : si la solution casse la quantité de donnée impactée sera plus grande (exemple de la corruption d'une base de données).

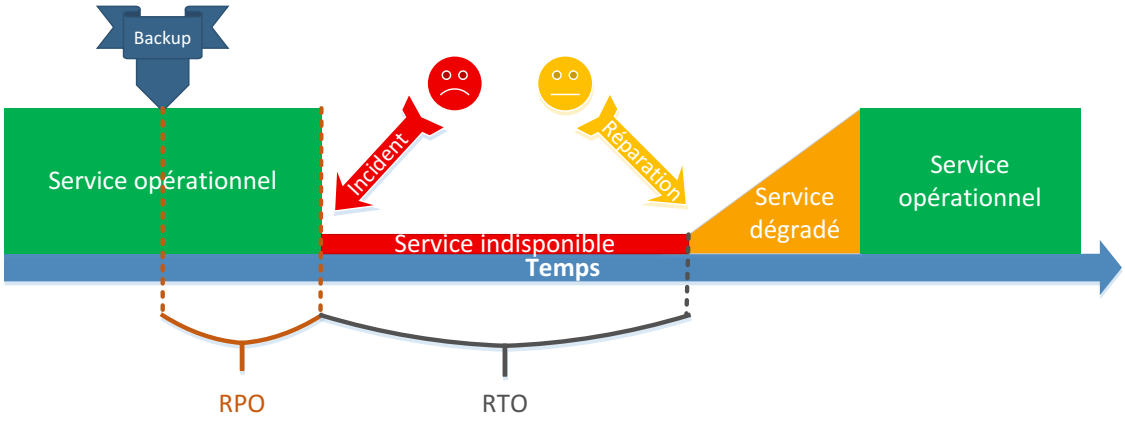


Figure 16 : récapitulatif des *RTO* / *RPO*[14]

Importances / Limites

Dans l'optique de "cloudification", les *SLA* deviennent la priorité qui nécessite de savoir lire entre les lignes des fournisseurs et comprendre les enjeux que leurs services engagent d'un point de vu *Business*. Les *SLA* donnent une vision sur les niveaux services qu'on peut attendre, malgré cela il ne faut pas non plus s'imaginer qu'ils offrent une sécurité absolue. Si un fournisseur s'engage à rembourser 20% du prix de la location de ses services pour une disponibilité inférieure à 99%, rien ne l'empêche de fournir une disponibilité de 80%, pouvant mettre en péril le client. Cela peut faire partie de la marge prise par le fournisseur de ne pas investir dans une infrastructure aussi performante/disponible qu'il prétend avoir. En ce sens, il est important de connaître ces quelques points à avoir en tête avant de se lancer dans la signature de n'importe quel service.

Les temps de disponibilité ou d'indisponibilité sont dans la majorité des cas mentionnés dans les *SLA*, il arrive que ce soit même les seules valeurs affichées. Les définitions peuvent varier en fonction du fournisseur qui en parle, c'est pourquoi il est toujours bon de bien regarder comment les fournisseurs les définissent.

C'est en lisant bien les *SLA* qu'il est possible de se rendre compte de quelques bizarreries, par exemple les contrats qui couvrent les composantes de manière indépendante. De ce fait, l'indisponibilité en cascade (voir figure ci-dessous) peut, dans le pire des cas, rendre la solution générale indisponible et pourtant rentrer dans les termes des *SLA* de chaque composante.

[13] https://fr.wikipedia.org/wiki/Perte_de_donn%C3%A9es_maximale_admissible
[14] https://fr.wikipedia.org/wiki/Dur%C3%A9e_maximale_d%27interruption_admissible

Figure 17 : indisponibilité en cascade[15]

Un autre exemple, ces temps d'indisponibilité, à eux seuls, ne disent pas grand-chose. Pour 8h de d'indisponibilité fournisseur, rien ne nous assure que ce sera 8h d'indisponibilité client : si la reprise d'activité est longue après un incident et que les indisponibilités du fournisseur sont récurrentes, il y aura une dé-corrélation entre les temps d'indisponibilité fournisseur et client (voir schéma ci-dessous). En ce sens il est important d'avoir plus d'informations sur les ces valeurs comme le nombre d'occurrences maximum de panne, le temps d'intervention moyen ou la garantie de temps d'intervention.

Figure 18 : récurrence d'indisponibilité

Le *RTO* a lui aussi de multiples interprétations. Il est possible de comprendre le *RTO* comme un objectif du temps entre le début d'une panne mettant fin à un service jusqu'au retour à la normale de ce dernier. Or le "retour à la normale" comprend-il la notion de service dégradé (voir figure ci-dessous) ? Cette vision est variable en fonction des acteurs et il est nécessaire que l'ambiguïté soit levée lors des rédactions des garanties en adéquations avec les *RTO*.

[15] @litterock

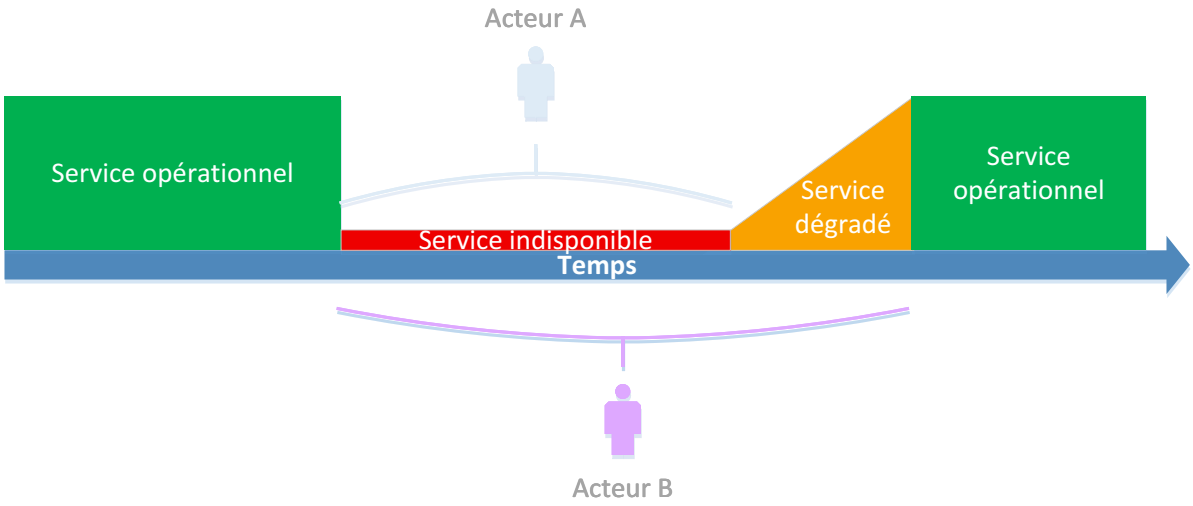


Figure 18 : *RTO* avec ou sans service dégradé

Une autre ambiguïté récurrente concernant toujours les *RTO* est l'identification du périmètre (matériel, applicatif, métier) de la composante impliquée dans la panne. La durée de 8h de rétablissement concerne-t-elle la remise en production (temporaire ou non) :

- des disques de la base de données qui tombe en panne ? (matériel)
- des tables de la bases de données qui ont été corrompues ? (applicatif)
- d'une suppression accidentelle par un utilisateur de données métiers ? (métier)

Par exemple, le temps pour la reprise d'activé d'une base de données est à ajouter au temps de reprise d'activité du serveur physique où elle est installée.

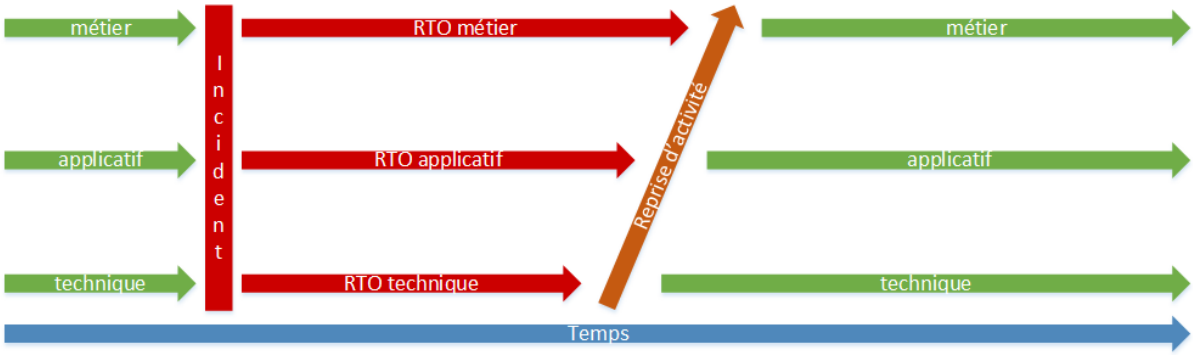


Figure 19 : reprise d'activé en fonction du niveau de l'application impacté

Aussi, les performances doivent être suffisamment explicitée dans les *SLA* afin de répondre assez finement aux questions suivantes :

- un service ayant de la latence est-il considéré comme indisponible ?
- à partir de quel niveau de latence dois-je considérer mon service comme indisponible ? Nous verrons comment Azure définit son service de base donnée comme injoignable un peu plus tard. IBM par exemple assure le temps de réponse aux requêtes des bases de données quand il infogérait la SI de Danone ;

Ces quelques notions nous arment pour comprendre précisément les offres de service que proposent les gros acteurs du marché.

Amazon, Citrix, Google, HP, IBM, Intel, Microsoft ou SalesForce figurent parmi les principales entreprises du secteur. Pour se donner une vue de l'esprit, le graphique ci-dessous indique la répartition de parts de marché entre les grands acteurs 2015[16]. J'aborderai dans cette partie les services proposés, leurs spécificités et la nature des *SLA* des deux plus gros acteurs du marché : Amazon Web Service (AWS) et son concurrent Microsoft Azure.

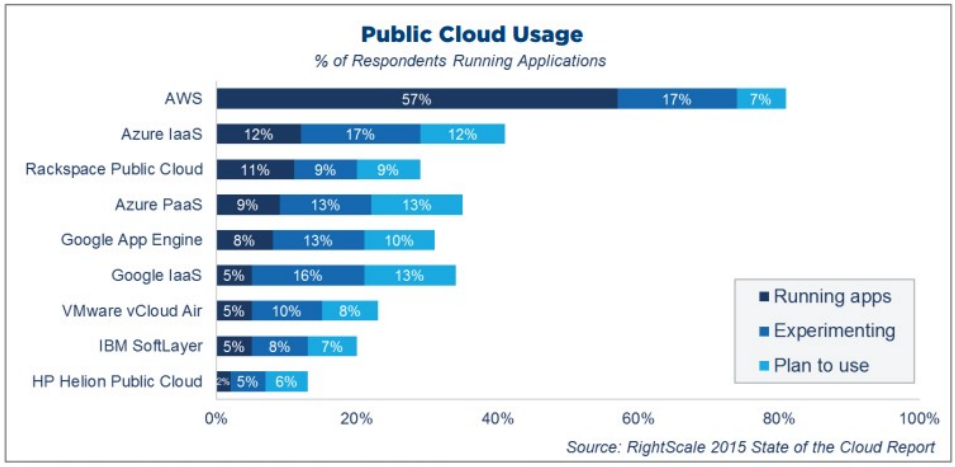


Figure 20 : Répartition des parts de marché du cloud

Microsoft Azure

Azure met à disposition ses contrats de niveaux de services en lignes classés par service (*VM*, *storage account*, *databases*, etc.). Ces documents sont dans l'ensemble légers car peu d'indicateurs sur les niveaux de service y sont mentionnés. Les *SLA* concernent chaque service de manière indépendante, de ce fait, le problème est qu'Azure ne protège pas des indisponibilités en cascades mentionnés dans les limites des niveaux de service.

Des contraintes existent pour que les *SLA* soient applicables. En ce qui concerne le service de *Virtual Machines*, le *SLA* mentionne qu'un pourcentage de disponibilité : 99.95% et encore, ce temps concerne uniquement l'une des deux *VM*. Ces deux *VM* doivent faire office d'instances redondantes (ce que Microsoft appelle des Groupes à Haute Disponibilité). C'est la condition nécessaire pour que le *SLA* soit appliqué.

Rien n'assure le nombre d'occurrence de ces temps d'indisponibilités, or si l'application métier (ou une application tierce dépendant d'elle) met du temps à se recharger en cas de coupure, ce temps de reprise ne sera pas comptabilisé pour Azure et peut mettre en danger l'expérience utilisateur.En parlant d'expérience utilisateur, les définitions de disponibilité ont toutes un sens différent en fonction du service proposé : concernant l'*Active Directory* (*AD*), le service est considéré comme disponible que s'il est possible d'effectuer certaines actions (parmi l'ensemble des actions normalement possible). De plus, le temps d'indisponibilité ne compte pas les maintenances imposées

[16] http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey

par l'hébergeur. Concernant Azure : "*nous publierons une notification ou vous informerons au minimum cinq (5) jours avant le début d'un tel Temps d'Indisponibilité*".

Un autre problème est l'absence de garantie de temps d'intervention (GTI). Microsoft ne spécifie pas le temps de la remise en production suite à un incident. Il faut se contenter du temps d'indisponibilité pour approximer le temps d'interruption maximum avant un remboursement.

Les performances attendues dépendent du service, l'accès des bases de données par exemple spécifie qu'"*une minute est comptabilisée dans le Temps d'Indisponibilité d'une Base de Données lorsque toutes les tentatives continues du Client pour établir une connexion avec la Base de Données au cours de cette minute échouent*".

L'absence d'informations précises sur les niveaux de performances peuvent ne pas répondre aux objectifs du métier : il est possible d'augmenter les ressources utilisées par les services mais les définitions de disponibilité mentionnées par les contrats d'Azure elles sont indiscutables. Azure propose néanmoins des tests de performances intégrés permettant par exemple de tester la tenue en charge d'une application web.

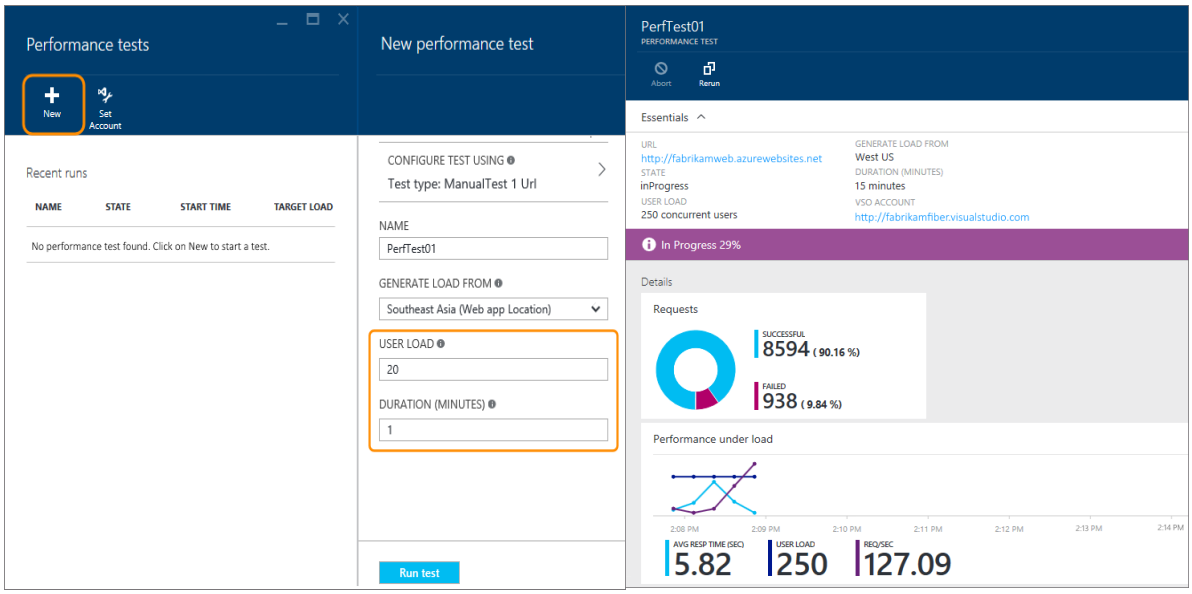


Figure 21 : tests de performances Azure d'une application Web

Les *VM* se payent à leurs utilisations et ont des prix variables. La tarification dépend des ressources allouées (c'est à dire le nombre de *vCPU*, la quantité de *RAM*, le type de stockage, etc.) et des licences impliquées. Par exemple, une *VM* hébergeant *SQL server* coûtera plus chère à l'heure qu'une VM sous *Linux*. Pour cette raison, il est important de bien éteindre les services quand ils n'ont pas besoin d'être allumés (Azure propose l'implémentation d'alertes sur la facturation). Si Microsoft facture à la minute, ce n'est pas le forcément le cas de tous les hébergeurs, certains facturent par plus large granularité, facturant par exemple 2 heures les 10 minutes d'utilisation réelle. Dans le cadre d'une architecture hybride, c'est une composante qu'il faut prendre en compte dans une optique de continuité de service et dans des soucis de supervision : différencier une *VM* normalement éteinte d'une *VM* en panne.

Certains[17] mêmes adaptent (*resize*) les performances des machines virtuelles pour répondre exactement aux attentes de service : ils partent du principe qu'il n'est pas nécessaire d'avoir une

[17] http://fr.slideshare.net/Nuno.Godinho/tips-tricks-on-architecting-windows-azure-for-costs

puissance de 100% pendant toute la journée mais uniquement pendant les fortes activités et ainsi baisser les nombres de *vCPU* pendant les périodes creuses. L'idée est de faire baisser le prix de l'utilisation des *VM* à l'heures tout en garantissant un niveau de service exigé par le métier. Cette scalabilité (*CPU*, réseau, disque) peut à l'inverse être augmentée afin de mieux répondre aux exigences. Le *resizing* a cependant quelques limitations :

- Il dépend du *hardware* disponible de la région où sont localisées les *VM*. Si la taille souhaitée de la *VM* n'est pas disponible dans la région, il faut soit attendre que le *hardware* nécessaire soit disponible, soit migrer la *VM* dans une autre région ;
- Il nécessite généralement le redémarrage de la *VM* ;
- Le *resizing* d'une VM nécessite souvent le *resize*, et donc le redémarrage (*reboot*), d'autres *VM*. C'est par exemple le cas des *VM* d'une Zone Haute Disponibilité (*Availability Set)*. "*If the VM you wish to resize is part of an availability set, then you must stop all VMs in the availability set before changing the size of any VM in the availability set. The reason all VMs in the availability set must be stopped before performing the resize operation is that all running VMs in the availability set must be using the same physical hardware cluster.*"[18]
- Enfin, cette action n'est pas instantanée : "*[...] the resize may not be immediate. If you change from say an A1 to an A7, or even from an A6 to A7, your VM may have to move to another Hyper-V host depending on how our hosts are loaded. Once the move is complete, then the VM will be brought up.*"[19]

Concernant le support, Azure le sépare en différents niveaux, le meilleur permettant d'être conseillé sur divers sujet comme la migration, le déploiement, la conception, l'implémentation, etc. Il permet aussi de bénéficier d'une vérification du code et de l'architecture. C'est un point à prendre en compte si le client souhaite connaitre les bonnes pratiques que préconise son fournisseur concernant la mise en place d'une architecture cloud (classique ou hybride).

Amazon Web Services

Les *SLA* d'Amazon Web Services (AWS) ressemblent à ceux d'Azure, pour le service EC2 (qui permet de faire des *VM*), seule la disponibilité -hors maintenances- est spécifiée. La contrainte de redondance est présente sous un autre nom : "*Availability Zone*" qui permet de déployer des *VM* sur des datacenters dans des zones différentes. Cette contrainte impose au client une architecture afin que les *SLA* puissent être appliqués.

Tout comme Azure, la plupart des services ne proposent qu'un temps de disponibilité défini spécifiquement en fonction du type de service qu'ils rendent avec les mêmes contraintes de performances et de scalabilités. La documentation d'AWS est cependant plus étoffée en ce qui concerne le support. En fonction de celui dont le client bénéficie, il est possible de se faire assister sur plus ou moins de sujet, plus ou moins prioritairement sur des plages horaires plus ou moins larges :

[18] https://azure.microsoft.com/en-us/blog/resize-virtual-machines/

[19] https://blogs.technet.microsoft.com/uspartner_ts2team/2015/01/20/resizing-an-azure-virtual-machine/

	Critique	Urgente	Elevée	Normale	Faible
Formule Entreprise 24h/24, 7j/7	15 minutes maximum	1 heure maximum	4 heures maximum	12 heures maximum	24 heures maximum
Formule Professionnel 24h/24, 7j/7		1 heure maximum	4 heures maximum	12 heures maximum	24 heures maximum
Formule Développeur (Heures d'ouverture*)				12 heures maximum	24 heures maximum

Le support permet de répondre à des questions sur les procédés en matière de services et de fonctionnalités AWS, aux bonnes pratiques pour nous aider à intégrer, déployer et gérer des applications dans le cloud, à installer, configurer et dépanner certains OS, applicatif, etc. mais n'aide pas dans la conception de code. Cependant en souscrivant à des offres support de niveau *Business* / *Entreprise*, il est possible de demander des conseils en termes d'architecture et de dimensionnement.

Dans un cas concret de panne, et dans une optique de réclamation, chez AWS comme chez Azure, c'est au client de faire la demande avec les informations permettant de prouver que l'incident a bien eu lieu[20] : dates et temps d'indisponibilités, instance qui subit le problème, *logs* qui attestent des erreurs confirmant la panne, etc. En ce sens, il est important d'avoir un suivi de son exploitation pour :

- pouvoir attester qu'une panne à bien eu lieu si l'on souhaite faire jouer les *SLA :* donner de la responsabilité à son fournisseur c'est se donner la responsabilité de le challenger en vérifiant que ce qu'il fait est réellement fait. ;
- rendre compte de la bonne exploitation du SI vis à vis des exigences métiers ;
- évaluer les capacités totales du SI et faire du *capacity planning* : l'objectif de la gestion de la capacité est de garantir que l'infrastructure informatique est fournie au bon moment, au bon prix et en quantité adéquate pour tenir la qualité de service en alignement avec les besoins métiers.

Amazon et Microsoft proposent des services similaires encadrés par des contrats eux aussi très ressemblant. Ces contrats reposent, comment nous l'avons vu plus tôt, essentiellement sur les temps de disponibilité. Dans les faits, comment ces deux acteurs se placent-ils autours de leurs concurrents ?

Analyse comparative

En 2014, le service Amazon *Elastic Compute Cloud* (EC2) a battu le record de 2,41 heures de d'indisponibilité (*downtime*) réparties sur 20 pannes. Cela signifie que les services étaient disponibles à 99.9974% du temps. Rien à voir en comparaison à l'évènement du réveillon de noël 2012, quand les services *Web Cloud Public* d'Amazon ont rencontré une panne de 12 heures dans l'est des Etats-Unis, mettant hors d'usage de nombreuse compagnie dont Netflix. Mais y a-t-il vraiment une relation entre les temps de disponibilité des *SLA* et les temps de disponibilité réellement constatés ? Un article de 2011[21] observe qu'en général, les *SLA* de 100% sont rarement respectés (17%). A l'inverse certains *SLA*

[20] http://www.networkworld.com/article/2885818/cloud-computing/10-tips-of-what-to-watch-for-in-microsoft-and-other-cloud-slas.htm

[21] http://blog.cloudharmony.com/2011/01/do-slas-really-matter-1-year-case-study.html

comme ceux d'AWS EC2 figurent parmi les plus mauvais et se trouvent pourtant dans le meilleur temps de disponibilité.

Les performances que délivre le cloud sont dures à analyser et à comparer. L'essence même de la « multi-location » (mutualisation) rend la solution non déterministe : la non-reproductibilité des tests de performances ne permettent pas d'obtenir de conclusions aussi précises que sur des solutions traditionnelles. Le graphe ci-dessous affiche les différents fournisseurs et instances qu'ils proposent. La couleur représente le ratio prix/performance (bleu étant meilleur que le marron), la taille des cercles représentant la performance de l'instance (les plus grandes représentent les plus performantes)[22].

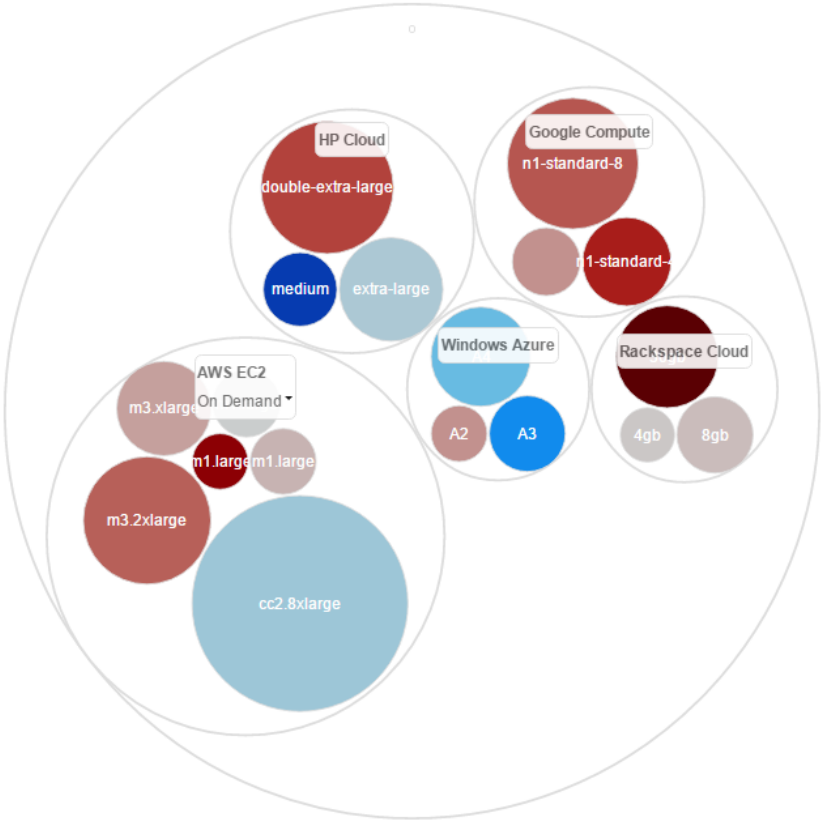


Figure 22 : ratio performance / prix par instances par fournisseurs

La figure ci-dessous représente elle les performances suivant deux métriques : SPECint et SPECfp qui sont des outils permettant de rendre compte les performances *CPU* à travers 29 tests « workloads ».

[22] http://blog.cloudharmony.com/2013/06/value-of-the-cloud-cpu-performance.html

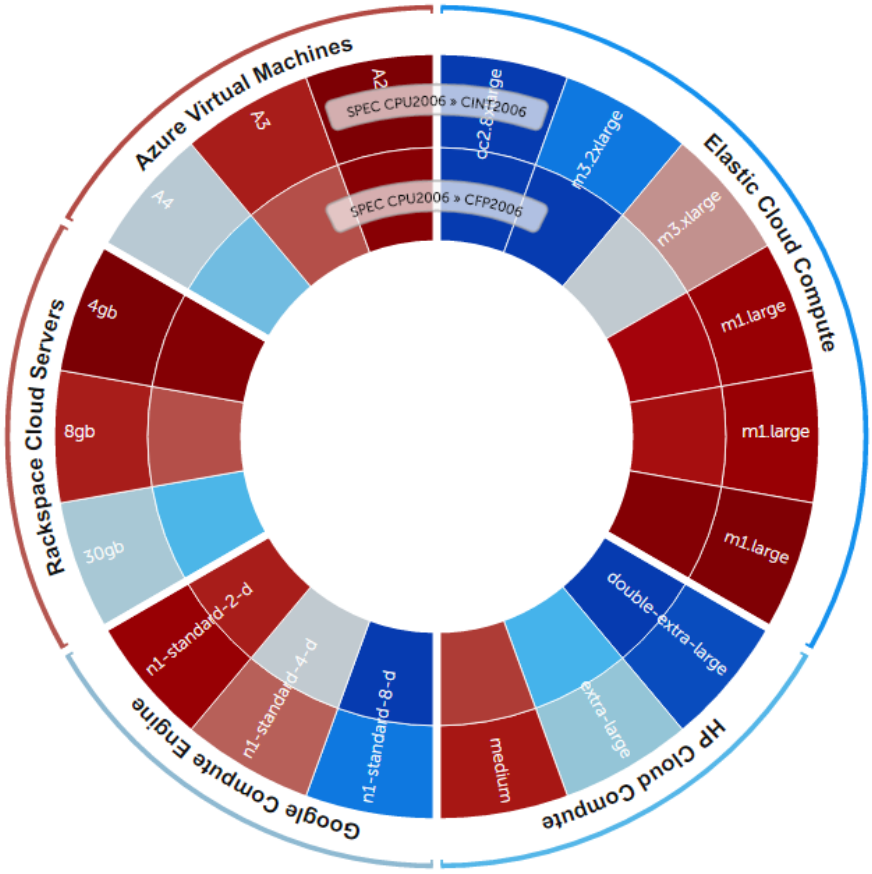


Figure 23 : performance SPECint et SPECfp par instances par fournisseurs

Les performances à prendre en compte sont relatives au cas d'usage : les applications web sont gourmandes en I/O en lecture et réseaux externe à la différence des bases de données qui elles demande plus I/O en lecture/écriture et en réseaux interne.

Les performances sont enfin corrélées à l'emplacement des *datacenters* : tous les continents ne sont pas fournis équitablement par les fournisseurs et imposent donc des contraintes sur les niveaux de services pour les infrastructures internationales. Certains providers imposent même des quotas par type d'instance par région, pouvant affecter des entreprises ayant un rapide croissance. Un article[23] (de 2014) détaille en profondeur les différences qu'apportent les plus gros *providers* du cloud et liste notamment les politiques de restrictions, en voici quelques-uns pour se donner une idée :

Service	Policy
Amazon EC2	20 instances par région pour la plupart des types de unités de traitement (*compute instances*)
DigitalOcean	10 unités de traitement (Droplets)

[23] http://blog.cloudharmony.com/2014/07/comparing-cloud-compute-services.html

Google Compute Engine	24 CPU *cores* par régions
Microsoft Azure	20 CPU *cores*
Rackspace	128 GB mémoire
SoftLayer	20 unités de traitement par jour, revue des instances supplémentaires manuellement pour approbation

Instance Size	Amazon EC2	DigitalOcean	Google Compute Engine	Microsoft Azure	Rackspace
2 GB / 1 Core	160 (20 par région)	10	72 (24 par région)	20	64
4 GB / 2 Core	160 (20 par région)	10	32 (12 par région)	10	32
8 GB / 4 Core	160 (20 par région)	10	18 (6 par région)	5	16
16 GB / 8 Core	160 (20 par région)	10	9 (3 par région)	2	8
32 GB / 16 Core	160 (20 par région)	10	3 (1 par région)	1	4

La sécurité peut dépendre elle aussi des *providers*. Même si les *OS* intègrent des outils palliant aux problématiques de sécurité, par exemple des pare-feu (*firewalls*) internes. Il est toutefois recommandé de mettre en place des ressources dédiées.

Il existe des outils permettant d'attester du bon fonctionnement de l'exploitation du *cloud*. Les *providers* tels que Amazon ou Microsoft, conscients de ce besoin, intègrent directement des solutions de supervision dans leurs plateformes d'administration.

Exemple de superviseur intégré

Microsoft Azure Monitoring (MAM)

Microsoft Azure Monitoring (*MAM*) est un ensemble d'outil permettant de monitorer les ressources hébergées sur Azure par la récupération de métriques. Leurs natures varient en fonction du type de ressources[24] : une machine virtuelle remonte des informations sur les capacités *CPU*, utilisation de la *RAM*, d'autres ressources comme les applications web (PaaS) remontent le nombre de réponse http 200, etc.

Les métriques sont dans notre cas ce qui nous intéresse le plus. Elles ont l'avantage d'être disponibles par défaut lorsqu'on déploie sur Azure. Leurs durées de conservation sont de 30 jours pour chaque métrique et d'une fréquence d'une minute entre deux collectes.

Azure propose la création d'alertes cependant cette création reste assez primitive : l'alerte déclenchée peut envoyer des mails, déclencher un script sur Azure où lancer une requête http mais l'escalade ou l'acquittement n'ont pas encore développés.

La nouvelle console de supervision relève un problème : si l'entreprise considère que l'infrastructure déployée sur le *cloud* est 'gérée' de la même manière que l'infrastructure traditionnelle, elle souhaitera connaitre l'état de son système via une unique console de supervision. Ce genre de problématique questionne l'organisation et le processus.

L'impact organisationnel

Définition ITIL

> ITIL (« Information Technology Infrastructure Library » pour « Bibliothèque pour l'infrastructure des technologies de l'information ») est un ensemble d'ouvrages recensant les bonnes pratiques (« best practices ») du management du système d'information. ITIL permet, grâce à une approche par processus clairement définie et contrôlée, d'améliorer la qualité des SI et de l'assistance aux utilisateurs en créant notamment la fonction (au sens « département de l'entreprise ») de Centre de services ou « Service Desk » (extension du « help desk ») qui centralise et administre l'ensemble de la gestion des systèmes d'informations. ITIL est finalement une sorte de « règlement intérieur » du département informatique des entreprises qui l'adoptent. - Wikipédia[25]

[24] https://azure.microsoft.com/en-us/documentation/articles/monitoring-supported-metrics/
[25] https://fr.wikipedia.org/wiki/Information_Technology_Infrastructure_Library

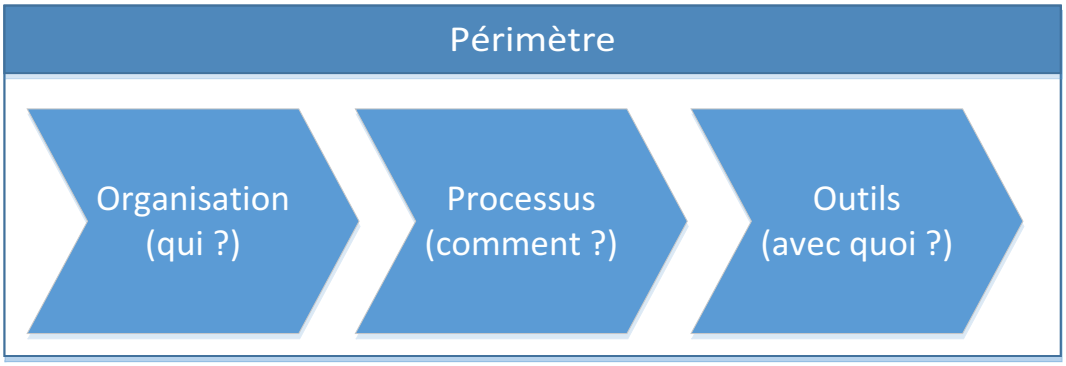


Figure 24 : intégration processus ITIL dans le périmètre

Les processus décris par ITIL s'intègrent entre l'organisation et l'outillage, à eux trois ils forment un périmètre (voir schéma ci-dessus). C'est l'organisation de l'entreprise qui définit les processus à adopter. Les processus sont ensuite implémentés à travers de l'outillage. Il est important que les choix se fassent dans ce sens et pas l'inverse : ce n'est pas grâce aux fonctionnalités des outils qu'une entreprise décide ses processus et son organisation. Pour répondre à des problématiques de gestion niveaux de service, certains processus d'ITIL sont impactés, à commencer par celui de la gestion d'incident.

Gestion des incidents

Un incident au sens ITIL est une interruption non planifiée ou une dégradation de la qualité d'un service informatique. Les objectifs du processus de gestion des incidents sont de :

- rétablir le fonctionnement normal du service aussi rapidement que possible ;
- réduire au minimum l'impact défavorable sur le fonctionnement des clients et des utilisateurs[26].

Le processus se déroule par étapes qui sont :

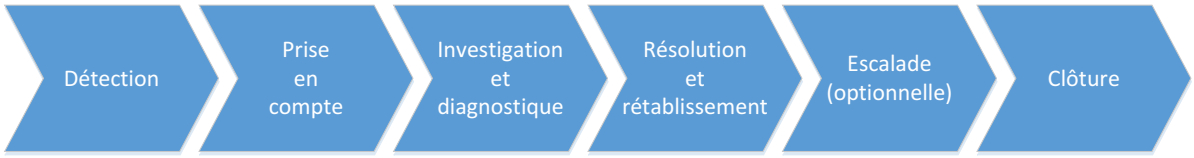


Figure 25 : étapes du processus de gestion des incidents

La gestion d'incident est le processus le sollicité dans ce sujet de stage puisque les premières étapes correspondent directement aux problématiques de supervision : comment gérer la détection d'un incident ? Dans le cadre d'une architecture de cloud hybride ces problématiques se déclinent par les questions suivantes :

- Comment suis-je capable de m'assurer de la santé de mon SI à distance ?
- Sur quel tronçon l'incident intervient-il ? *On Premise*, sur le *cloud* ?

[26] http://www.itilfrance.com/index.php?pc=pages/docs/itilv3-04/120-01.inc&pe=haut_entete_itil2007.inc&pt=La%20gestion%20des%20incidents

L'analyse des causes racines se déroule après la clôture de l'incident. Elle n'est pas du ressort de la gestion d'incident mais de la gestion des problèmes.

Gestion des problèmes

Au sens ITIL, un problème s'agit de la cause d'un ou de plusieurs incidents. Le problème est initié principalement à la fermeture d'un incident. Les objectifs du processus de la gestion des problèmes sont les suivants :

- empêcher les problèmes et les incidents qui en découlent de se produire ;
- éliminer les incidents récurrents : il s'agit d'une série d'incidents similaires (ou perçus comme similaires) ;
- réduire au minimum l'impact des incidents ne pouvant être empêchés[27].

Dans le cadre d'une architecture de cloud hybride, l'implémentation de la gestion des problèmes ne change fondamentalement à l'exception de l'utilisation d'outil de métrologie aidant à l'investigation.

Gestion des niveaux de services

Les objectifs du processus de la gestion des niveaux de services sont les suivants :

- s'assurer que les niveaux services de solutions clouds adoptées soient en phase avec le besoin métier ;
- vérifier que les niveaux de services (*SLA*) soient respectés grâce aux outils de monitoring.

Dans le cadre d'une architecture de cloud hybride, la principale différence avec l'architecture traditionnelle est que les applications hébergées sur le *cloud* sont des offres de services. Il n'y a donc pas de négociation possible de la part des *services managers* (les personnes en charge de s'assurer des niveaux de service) sur les offres proposées par le *provider*.

Les trois processus décris ci-dessus s'implémentent en partie (détection) par des outils d'exploitation appelés outils de monitoring.

Les outils d'exploitation

"Les besoins en termes de supervision (au sens large), ne sont pas du tout les mêmes selon les cibles :

Les exploitants ont besoin de connaître l'état de chaque indicateur technique à un instant t et d'être alerté en cas de défaillance afin de corriger le problème au plus vite (Gestion de la disponibilité).

Les responsables fonctionnels, on besoin de savoir si les applications dont ils ont la charge fonctionnent et cela avec des temps de réponse acceptables du point de vue des utilisateurs (Gestion de la performance).

[27] http://www.itilfrance.com/index.php?pc=pages/docs/itilv3-04/120-05.inc&pt=La%20gestion%20des%20probl%E8mes&pe=haut_entete_itil2007.inc

La supervision du ressenti utilisateur commence là ou s'arrête la supervision technique et adresse de nouveaux besoins de supervision. Votre application peut parfaitement fonctionner du point de vue technique (disponibilité), mais être totalement inutilisable à cause de temps de réponse très longs (ressenti utilisateur). La supervision du ressenti utilisateur permet de se mettre à la place de l'utilisateur en exécutant des scénarios d'utilisation d'applications et en plaçant des points de vérification et de mesure des temps de réponse à chaque étape du scénario applicatif. On parle aussi de supervision de bout en bout ou de End User Experience." – Wikimonitoring-fr[28]

Définitions supervision / métrologie / monitoring

Cette partie se concentre sur les outils qui permettent le *monitoring* des solutions qu'on souhaite surveiller. Ces outils, à la différence de ceux évoqués plus haut, ne sont pas intégrés aux solutions *clouds*. Avant de débuter la présentation des outils, je compte m'attarder aux relations qui existent entre la supervision, la métrologie et le *monitoring*. Le terrain est glissant et les confusions entre les trois notions peuvent vite survenir. Un article[29] d'it-connect.fr m'a permis de démystifier les appellations, en voici un extrait adapté :

La métrologie, dans laquelle on retrouve la notion de mètre/métrique, est le fait d'obtenir, de garder et de tracer la valeur numérique d'une charge. Par exemple le pourcentage de CPU utilisé sur un serveur, le nombre de personnes connectées sur une site web, le trafic sortant et entrant sur un switch. Bien souvent, la métrologie permet tout simplement de tracer des graphiques, bien connus dans le domaine du monitoring. La métrologie est donc le fait de récupérer la charge (chiffrée) permettant de tracer son évolution dans le temps.

Figure 26 : Logo métrologie[30]

La supervision vise à répondre à la question « le service est-il joignable ? ». Au sens strict du terme, aucune valeur numérique n'entre en compte et l'aspect historique de charge ne fait donc pas partie de la supervision. Une « extension » de la supervision porte sur le fait de récupérer une valeur chiffrée comme une charge mémoire et de lui appliquer un seuil d'alerte. Dans ce cas, aucun historique

[28] https://wiki.monitoring-fr.org/supervision/eue/start
[29] http://www.it-connect.fr/monitoring-supervision-et-metrologie/
[30] https://thenounproject.com/term/mesure/645304/

n'est gardé mais l'on est capable d'obtenir et de surveiller non plus des états, mais des valeurs numériques.

Figure 27 : Logo supervision

Le *monitoring* (monitorage/surveillance) est le fait de surveiller quelque chose, ce qui revient à connaître son état actuel mais aussi l'historique de ses états passés, par l'intermédiaire de valeurs (*UP*/*DOWN*) et de données chiffrées (des pourcentages par exemple). Avec le temps, supervision et métrologie tendent à complètement se confondre dans les solutions proposées. Ainsi, lorsque l'on cherche à récupérer une information, on parle alors de monitoring. On est maintenant capable d'appliquer de la supervision sur la métrologie, pour faire plus clair, on va appliquer des alertes sur des seuils de charge.

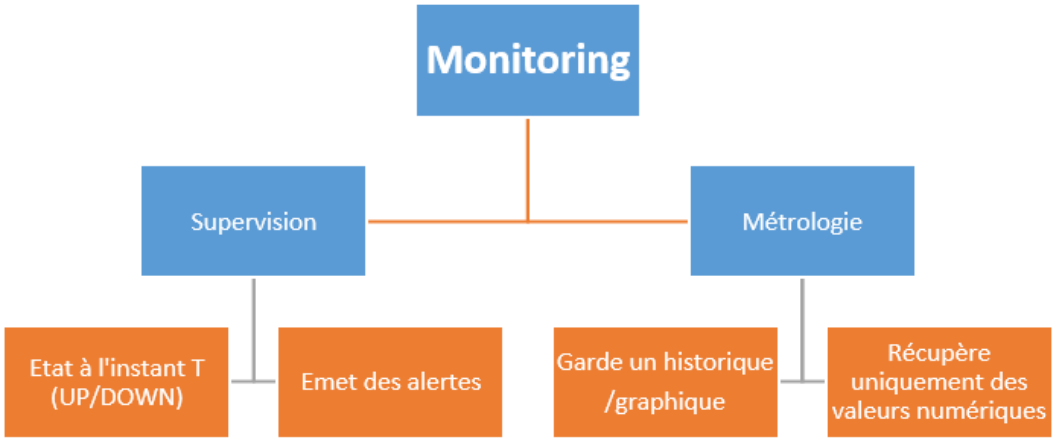


Figure 28 : relations métrologie supervision métrologie[31]

[31] http://www.it-connect.fr/monitoring-supervision-et-metrologie/

Exemples d'outils

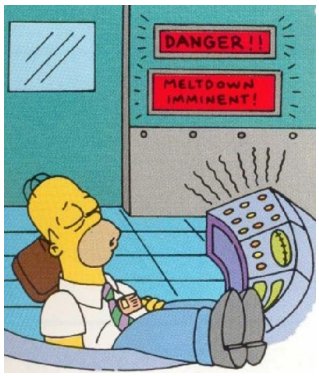


Figure 29 : supervision inefficace[32]

Nous allons dans cette partie nous intéresser aux outils qui existent sur le marché et qui implémente la détection du processus de la gestion d'incident. Les différences fonctionnelles entre supervision et métrologie s'effacent : les outils à l'origine spécialisés intègrent des fonctionnalités de chaque composante les rendant plus vraiment "pur outil de métrologie" ou "pur outil de supervision".

Zabbix

Figure 30 : logo Zabbix[33]

Les explications sur le fonctionnement de Zabbix sont tirées en grande partie d'un article[34]. Vendu comme un outil de supervision, son fonctionnement repose sur quatre composants :

- Le serveur, composant principal, disponible uniquement sous Linux, il permet la surveillance des systèmes. Il fonctionne habituellement grâce à des agents mais ceux-ci ne sont pas une obligation. Sans agent, Zabbix peut atteindre certaines ressources mais remonte moins d'informations.
- L'agent qui, comme nous l'indique, est optionnel et permet de remonter un plus grand volume d'informations sur la ressource sur laquelle il est installé. L'agent est déployable sur plus de plateformes que le serveur : linux, Unix, OS X, Microsoft. Il permet la surveillance des hôtes sur lequel il est installé. Les informations sont communiquées directement au serveur ou via un *proxy* Zabbix par des requêtes en JSON.
- Le *proxy* permet de collecter les informations d'un hôte (de la même manière que le serveur) avant de les envoyer au serveur. Son utilisation évite l'ouverture excessive de flux au sein du réseau et trouve donc son utilité dans la supervision de système à distance (donc des architectures *clouds*) par son rôle de relais. Cela permet aussi d'alléger le traitement du serveur puisqu'une partie de ces traitements sont effectués sur la machine hébergeant le proxy. A la différence du serveur, le *proxy* ne peut pas être administré par le *frontend*.

[32] https://www.techpowerup.com
[33] http://www.zabbix.com/img/logo/zabbix_logo_500x131.png
[34] https://wiki.monitoring-fr.org/zabbix/zabbix-work

- Le *frontend*, la partie immergée de l'iceberg, est l'interface permettant de visualiser et configurer l'outil. C'est une interface web, donc accessible via un navigateur.

Les alertes sont gérées de ma manière suivante : les informations (JSON) entre un agent et un serveur (ou un *proxy*) sont appelées *checks* et permettent de remonter des métriques. Chaque item est focalisé sur la collecte de donnée sur une ressource par exemple l'utilisation *CPU* ou *RAM*. Des *triggers* (déclencheurs) permettent de générer des évènements en réactions aux valeurs remontées par les items. Enfin, les évènements sont soumis (ou non) à un ensemble d'actions tel que l'envoi de mails, sms, etc.

Dans un contexte d'architecture cloud hybride (voir schéma ci-dessous), les métriques peuvent être remontées essentiellement par des agents qui sont installés sur les machines. Elles sont ensuite véhiculées directement vers le Zabbix serveur, ou par l'intermédiaire d'un Zabbix *proxy*. Dans le cadre du *cloud*, il n'est question que de *IaaS*, car l'agent doit être installé sur un *OS*. Il convient aussi de prendre en compte que les flux entre Zabbix agent et Zabbix serveur ne sont pas, par défaut, chiffrés. En ce sens, il est envisageable de monitorer du *IaaS* privé avec un Zabbix *proxy* puisque la liaison est sécurisée. Monitorer du IaaS *public* expose potentiellement des données de l'état des serveurs à Internet. Des solutions plus exotiques comme le *Public Peering* proposées par Microsoft existent : capacité de discuter à une IP publique en passant par une liaison dédiée, évitant donc l'exposition de flux sensibles sur Internet.

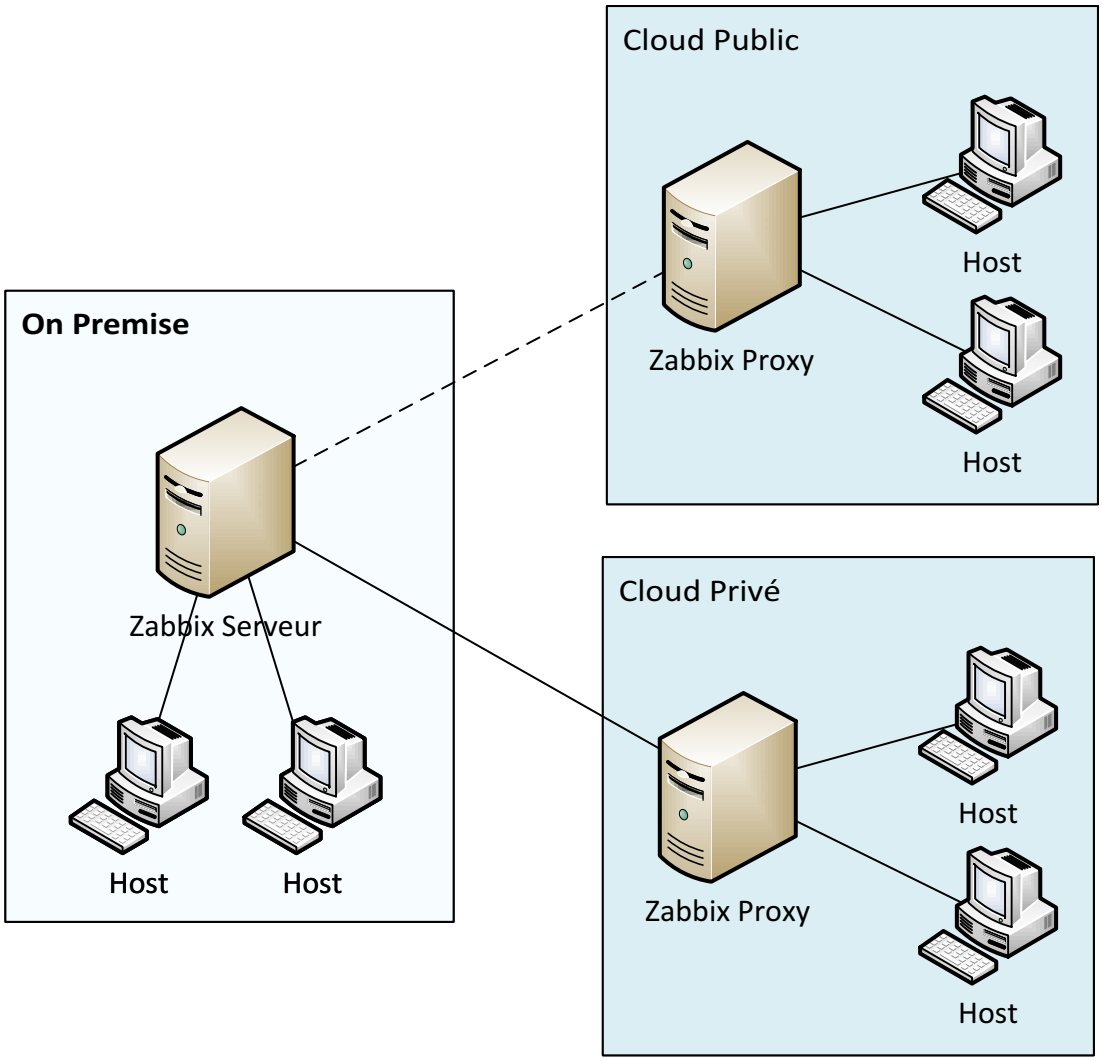


Figure 31 : architecture hybride avec Zabbix

Dans le cadre d'XXX, l'outil Zabbix a été utilisé pour ses fonctionnalité de supervision (alerte + up/down). Pour la partie métrologie, c'est l'outil Splunk qui a été adopté. Son rôle est d'indexer des données de diverses sources de manière centralisée. Cette centralisation vient d'un besoin simple : lire les journaux de bord (*logs*) d'un système permet de connaître l'état de ce système, or une architecture contient généralement un ensemble multiple de système de différentes natures nécessitant d'être surveillés.

Splunk

Figure 32 : logo Splunk

Si Zabbix pallie essentiellement à la gestion d'incident, Splunk, lui, participe à l'outillage pour la gestion des problèmes : il facilite l'investigation d'erreurs par des recherches avancées des logs, permet de faire de la gestion de la capacité.

…et d'autres…

Dans le cadre de mon stage et de du contexte client dans lequel je suis plongé, je me suis essentiellement focalisé sur l'outil Zabbix. Ils existent néanmoins d'autres outils :

Importances / Limites

L'outil de supervision, en l'occurrence Zabbix chez XXX, tente de couvrir une partie du processus de la gestion d'incident à savoir :

- La détection grâce aux remontés des métriques, l'envoi d'email ;
- La prise en compte grâce à la fonction d'*acknowledgement* ;
- L'investigation et diagnostic en partie grâce aux descriptions des alertes remontées ;
- Etc.

La première chose est de s'assurer que l'outil de *monitoring* ne soit pas lui aussi *down*. En cas de panne du serveur hébergeant l'outil de monitoring, aucune alerte n'est remontée. Si d'autres applications sont hébergées sur le même serveur, aucune notification n'alerte qu'elles ne marchent plus.

Une utilisation massive de monitoring peut impacter des baisses de performances et des failles de sécurité (flux sur l'état de santé d'un serveur), notamment sur le serveur ou est hébergé la solution de supervision. Un exemple concret de mauvaise pratique est l'utilisation intensive de scripts externes (*external scripts*)[35] sur Zabbix qui ouvre et ferme perpétuellement des connexions avec le serveur supervisé.

D'un point de vue organisationnel, la mise en place d'outils de monitoring nécessitent des ressources humaines dédiées s'occupant de l'installation / la configuration / le déploiement / la maintenabilité, notamment si ces outils se connectent à des sources susceptibles de changer (API Azure par exemple[36]).

Il est nécessaire d'instaurer des processus en adéquation avec l'organisation. La configuration des alertes par exemple, se doit d'être particulièrement bien dosée : si trop d'alertes sont levées, les équipes ne prendront pas la peine de les lire et un risque d'incident majeur risque de passer inaperçu. Ce genre d'incident peut aussi arriver, si trop peu d'alertes sont levées. Si une alerte est levée, il faut s'assurer qu'une procédure documentée permette de remédier à l'incident.

Les processus qui ne sont pas gérés par l'outil de *monitoring* doit permettre une bonne intégration avec d'autres outils comme par exemple des outils de *ticketing*. Ces outils permettent d'avoir un suivi précis des incidents notamment sur leurs dates d'ouverture, de clôture. Cette intégration ne concerne pas uniquement des logiciels fonctionnellement différents, il est parfois nécessaire de faire communiquer des logiciels de *monitoring* entre eux.

Comment s'intègrent-ils dans les architectures hybrides ?

S'il s'agit de solutions *IaaS* ou plus généralement d'un *monitoring* à faire sur des *hosts* à la couche de l'*OS*, l'intégration d'outils dédiés tel que Zabbix, Splunk ou équivalent n'est pas un problème. Avec leurs agents, ils sont conçus pour monitorer des équipements locaux ou à distance par une architecture adaptée comme nous l'avons vu avec les Zabbix *proxy*.

Dès lors qu'on quitte les solutions *IaaS*, le *monitoring* change d'aspect. Les « boîtes noires » telles que les solutions *PaaS* ou *SaaS* ne nous permettent pas d'installer de tels agents. Il n'est plus aussi facile de pouvoir remonter ces informations qui peuvent toujours être impactantes : des bases de données *PaaS* reposent nécessairement sur des infrastructures. Certaines métriques (les locks de base de données par exemple) ne sont des informations accessibles qu'au niveau de l'*OS*.

Les hébergeurs comme Azure ou AWS mettent à dispositions comme nous l'avons vu précédemment des outils intégrés de supervision. Ces outils sont accessibles depuis le portail de l'hébergeur, ce qui peut ne pas correspondre au client. Ces outils pourraient potentiellement suffire dans le cas d'architectures dédiées au cloud d'un seul hébergeur. Pour des architectures *clouds* hybrides ou *muti-providers*, la complexité est tout autre : à commencer par le nombre de consoles de supervision à surveiller.

[35] https://www.zabbix.com/documentation/3.0/manual/config/items/itemtypes/external
[36] https://docs.microsoft.com/en-us/rest/api/apimanagement/

Certaines entreprises souhaitent avoir qu'une seule plateforme à superviser. Dans cette optique, il faut interfacer la console de supervision avec les différentes sources d'objets à monitorer. Ces interfaces sont pour le cas d'Azure gérées par des *API* ou des commandes PowerShell.

Les *API* (*Application Programming Interface*) proposés par AWS et Azure sont des services web qui permettent de requêter le cloud. Microsoft participe au développement de *SDK open source* sur GitHub. Ses *SDK* sont des boites à outils qu'on utilise via un langage et qui facilitent les communications avec les *API* (voir schéma plus bas). C'est un moyen pour remonter des métriques et s'assurer de l'état de santé des ressources sur Azure. L'interrogation de ces interfaces permet d'avoir un aperçu sur les « métriques internes » (métriques liées à des couches plus basses) du *PaaS* ou du *IaaS*.

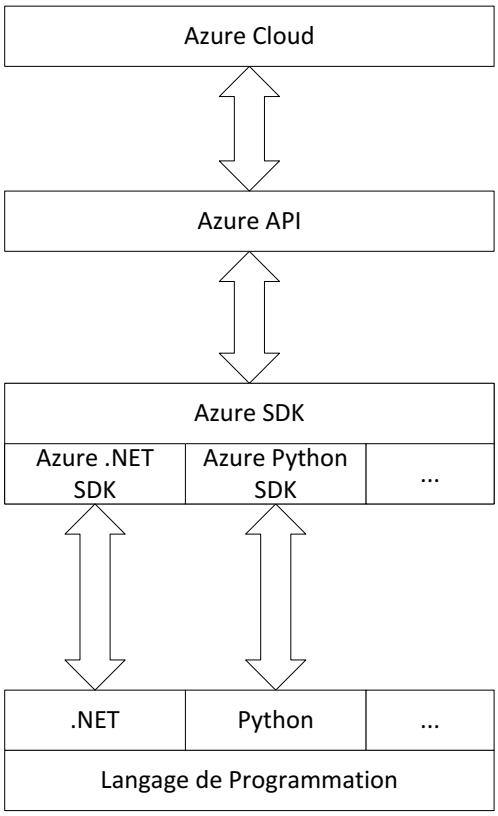


Figure 33 : API et SDK intégrations

Un autre moyen d'interfacer le *monitoring* intégré du *cloud* avec les outils on premise consiste en l'installation d'un méta dashboard. Des outils sur le marché s'interfacent avec Azure, c'est le cas de PagerDuty [37] qui remonte les alertes déclenchées depuis Azure vers un dashboard unifié. L'intégration fonctionne grâce au *webhook* : les requêtes http envoyées lorsqu'une alerte est déclenchée sur Azure. D'autres outils comme dynatrace [38] utilisent les *API* pour remonter des métriques permettant d'unifier l'arrivée de donnée de multiples fournisseurs.

Que ce soit par le développement d'outils internes ou par l'achat de licences, il faut comprendre que les interfaces avec le *cloud* sont sous la responsabilité de l'hébergeur. Cela signifie que c'est au client de s'adapter quand le fournisseur décide de changer ses *API*, voire de les supprimer.

[37] https://www.pagerduty.com/
[38] https://www.dynatrace.com

Conclusion et retour d'expérience

Ce sujet de stage est focalisé sur les problématiques de monitoring et particulièrement de la supervision. Les problématiques liées à la sauvegarde, les rôles et responsabilités (RACI[39]), l'inventaire, etc. ne sont pas décrites dans ce sujet de stage mais contribuent aussi à garantir les niveaux de service du SI.

L'expérience d'XXX m'a fait réaliser à quel point la gestion des niveaux de service est importante et complexe à mettre en place dans un SI. J'ai aussi pris conscience des difficultés de la mise en place d'outils de supervision qui permettent de s'assurer de ces niveaux de service au sein d'architectures de *cloud* et plus particulièrement de *cloud* hybrides.

L'adoption du *cloud* chez XXX s'est faite indépendamment de l'organisation et des processus existants. De ce fait, il a été nécessaire de trouver des moyens de supervision qui techniquement sont des solutions prématurées ou mal adaptées : faire communiquer Azure et Zabbix n'est pas nativement prévu. A titre d'exemple, j'ai détecté un bogue de la *SDK* Python développée par Microsoft Azure permettant de remonter les métriques sur les bases de données[40] via l'*API REST* (*PaaS*). Ce genre de fonctionnalités bancales prouvent que les technologies / interfaces sont jeunes et que peu d'acteurs du marché semblent les utiliser.

Dans la problématique de supervision corrélée au contexte d'XXX, le *cloud* n'apporte pas de complexité tant qu'il reste **uniquement** une abstraction infrastructurelle. La supervision se greffe sans mal sur du *IaaS*. Sa transparence et sa ressemblance avec l'infrastructure traditionnelle est son meilleur atout.

Beaucoup d'importance est attribuée aux autres services tels que le *PaaS* ou le *SaaS*, trop à mon gout. Les services *PaaS* proposés par les fournisseurs de *cloud* sont des surcouches aux *IaaS* qui trouvent sens pour des cas d'utilisation bien particuliers. Ces surcouches sont l'assemblages de *IaaS* auxquels ont été ajoutés des processus '*home made*' développés par le fournisseur de *cloud*. Or, c'est dans les processus qui forment les surcouches *IaaS* que peuvent intervenir des spécificités du métier et dont les *providers* tel que Microsoft ne peuvent / veulent avoir connaissance.

Le *PaaS* est facile à déployer mais apporte une difficulté de supervision. Comme nous l'avons vu plus haut, les interfaces du *cloud* changent rapidement et indépendamment de la volonté du client. De plus, rien n'assure qu'Azure sera l'unique fournisseur d'XXX. AWS ou le *cloud* de Google sont des *providers* potentiels qui ajoutent une couche de complexité dans le design d'une architecture harmonieuse.

Des outils de management d'infrastructure pluri-fournisseurs voient le jour, une des solutions la plus aboutie[41] et *open source* est ManageIQ. Projet racheté par RedHat en 2012, il apporte une couche supplémentaire d'abstraction reposant sur la communion d'infrastructures multi-*clouds* et traditionnelles. Cette manière d'aborder l'industrialisation sont des voies d'exploration qui peuvent s'avérer prometteuses à l'avenir.

Une tout autre vision consiste à considérer le *cloud* et l'infrastructure traditionnelle comme des périmètres indépendants. Choisir l'outil (ici le *cloud*) avant l'organisation est une mauvaise pratique, cependant si les entreprises s'entêtent à migrer vers le *cloud*, elles devront investir du temps dans l'élaboration de nouvelles organisations et de nouveaux processus. L'exemple des périmètres

[39] https://fr.wikipedia.org/wiki/RACI
[40] https://github.com/Azure/azure-sdk-for-python/issues/909
[41] http://blog.octo.com/cloud-management-platform-vers-la-gouvernance-de-votre-empreinte-it/

autour des systèmes centralisés[42] et distribués[43] montre qu'un changement trop important de paradigme oblige une réorganisation. Je fais le pari que le *cloud* subira un traitement particulier et fera objet d'un périmètre distinct de l'infrastructure traditionnelle.

[42] https://fr.wikipedia.org/wiki/Serveur_central
[43] https://fr.wikipedia.org/wiki/Architecture_distribu%C3%A9e

Sources complémentaires

- https://wikipedia.org
- http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey
- http://www.forbes.com/sites/louiscolumbus/2015/09/27/roundup-of-cloud-computing-forecasts-and-market-estimates-q3-update-2015/#138ef5f86c7a
- http://france.emc.com/collateral/emc-perspective/h6870-consulting-cloud-ep.pdf
- http://www.informationweek.com/cloud/cloud-vs-on-premises-6-benefits-of-keeping-data-private/d/d-id/1323089?image_number=5
- http://www.dell.com/learn/fr/fr/frbsdt1/campaigns/revueit-cloud-public-prive-match
- http://www.inficiences.com/fr/ge/ressources/strategie-produit-services/delai-sur-marche-time-market
- https://fr.scribd.com/doc/98645235/Gestion-de-la-disponibilite-ITIL-V3
- https://wiki.monitoring-fr.org/zabbix/zabbix-work
- http://blog.zabbix.com/multiple-servers-for-active-agent-sure/858/
- http://blog.nicolargo.com/wp-content/plugins/download-monitor/download.php?id=1
- https://fr.wikipedia.org/wiki/Centreon
- http://publib.boulder.ibm.com/tividd/td/ITM/SH19-4569-03/fr_FR/PDF/dmumst.pdf
- http://blog.neoxia.com/qu-est-ce-qu-une-metrologie/
- https://www.splunk.com/web_assets/pdfs/secure/Splunk_for_the_Cloud.pdf
- https://www.splunk.com/pdfs/technical-briefs/splunk-and-aws.pdf
- https://www.hpe.com/h20195/V2/GetPDF.aspx/4AA1-6028ENW.pdf
- https://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems
- https://www.opennms.org/en/opennms/the-platform
- http://hn.premii.com/#/article/12722586
- http://www.networkworld.com/article/2885818/cloud-computing/10-tips-of-what-to-watch-for-in-microsoft-and-other-cloud-slas.html
- https://blog.codingoutloud.com/category/cloud-computing/azure-cloud-computing/stupid-azure-tricks/
- http://readwrite.com/2013/08/21/gartner-aws-now-5-times-the-size-of-other-cloud-vendors-combined/
- http://www.infoworld.com/article/2614685/amazon-web-services/why-google-is-freaking-out-amazon.html
- http://www.networkworld.com/article/2162488/cloud-computing/how-long-will-big-name-customers-like-netflix-put-up-with-amazon-cloud-outages-.html
- http://www.networkworld.com/article/2866950/cloud-computing/which-cloud-providers-had-the-best-uptime-last-year.html
- https://azure.microsoft.com/en-us/documentation/articles/app-service-web-app-performance-test/

Index