Livret d'Epargne AWS

www.gekko.fr

Livret d'épargne AWS

Introduction aux Cloud FinOps :

- Reporting
- Cost allocation
- Cost optimisation

Félicitations !

Vous êtes l'heureux détenteur d'un ou plusieurs comptes AWS.
Vous avez pu savourer le plaisir inégalé d'obtenir des ressources en quelques minutes plutôt que plusieurs semaines, construire des architectures sécurisées et performantes en quelques clics, tester des services innovants pour supporter vos projets digitaux sans que l'on vous explique pourquoi ce n'était pas possible, etc etc... et tout ça pour quelques euros par heure.

Mais ça n'a pas duré.
La facture qui pesait quelques centaines d'euros par mois est passée à plusieurs milliers.
Et quand elle a dépassé 5 chiffres, les questions ont surgi :
Comment avait-on ouvert ce compte, déjà ?
Avec la carte bancaire de qui ? Et c'est sur quel budget, d'ailleurs ?
On en a vraiment besoin ? Est-ce que ça ne fait pas double usage avec ce qu'on a dans nos datacenters ? Qui a la main sur la dépense, et qui surveille ce que l'on fait ? Et est-ce qu'on ne peut pas optimiser tout ça ?

Outre que ne pas y répondre signifie souvent la fin de l'expérience - et c'est dommage -, il faut bien admettre que ce sont de bonnes questions : nous parlons ni plus ni moins de la mise en place d'une gouvernance opérationnelle et financière d'un composant appelé à jouer un rôle majeur dans l'IT de toutes les entreprises.

Or c'est compliqué... pour au moins deux bonnes raisons
et deux mauvaises !

Les bonnes, tout d'abord.
Au delà des usages qu'il permet, le Cloud a profondément «disrupté» les processus d'achat et d'exploitation de l'IT en tant que ressource.
Les entreprises fonctionnent par budget, et rationalisent leurs dépenses via leur fonction achats ; on achète au meilleur prix en maximisant le levier du volume et donc de la centralisation.
Conséquence du point précédent, les achats d'infrastructures reposent sur un capacity planning long terme : on s'équipe pour pouvoir traiter le besoin d'aujourd'hui comme celui de demain, et pour pouvoir traiter la crête de charge aussi bien que le train-train quotidien.
C'est pourquoi les serveurs des datacenters sont souvent à la fois excédentaires et modérément chargés, allumés jour et nuit, hébergeant des projets actifs aussi bien qu'arrêtés, tout ceci ne posant pas de problème tant que l'on reste dans le «stock» d'infrastructures qui a été planifié et investi.

Or le Cloud est par construction doublement orthogonal à ce modèle :
››› D'un point de vue achat, plusieurs personnes ont la main sur le robinet de la dépense, et celle-ci devient plus difficile à anticiper, réguler et optimiser.
››› D'un point de vue exploitation, la tentation sera grande pour les équipes d'appliquer les mêmes principes de dimensionnement, de tolérance à l'inutilisation et de fonctionnement jour et nuit que sur les datacenters, ce qui sera un tueur de business case.

Outre les bien connus glissements de modèle *«Capex vs. Opex»* et *«Make vs. Buy»*, le Cloud amène donc une gestion de l'IT orientée «flux» plutôt que «stocks» : là où l'on cherchait l'équilibre entre trop (incidence financière) et pas assez (incidence business) d'un stock d'IT, on s'intéresse à présent à la régulation de la consommation d'un flux d'IT en fonction du besoin.

Tout ceci nécessite outillage et pédagogie, et sans vouloir trop dévoiler la suite, le portefeuille est souvent un excellent outil pédagogique : un bon reporting opérationnel et une bonne ventilation des coûts seront le commencement de la sagesse...

Et c'est là qu'intervient la première mauvaise raison.
AWS est une plateforme fabuleuse en termes de puissance, de possibilités, d'innovation... mais nous pensons que ce ne sera pas leur faire offense que d'estimer qu'ils ne facilitent pas vraiment la tâche des gestionnaires (et pour avoir regardé d'autres plateformes Cloud, ils ne sont ni les seuls, ni les pires).
Quiconque s'est déjà escrimé sur des exports de Cost Explorer pour retraiter les chiffres en vue d'une analyse de consommation ou une re-allocation des coûts verra de quoi nous parlons : manque d'information clés, profusion d'informations parasites, conventions de nommage inconstantes, lourdeur des fichiers etc etc...
Les rapports de la Billing Console sont par ailleurs souvent insuffisants, et les vues opérationnelles ne sont pas très pratiques (qui n'a pas oublié des ressources sur une plaque géographique du fait de l'approche de la console par région ? qui arrive à se faire simplement une idée claire du niveau effectif d'utilisation de ses ressources pour un projet ? etc...).
Précisons que ceci n'est déjà pas évident pour un compte, et que cela devient quasi-impossible dès que l'on gère une matrice de plusieurs comptes AWS (par exemple Dev, Intégration, Production) servant plusieurs projets ou métiers consommant tous des ressources dans chacun des comptes.
Un outillage additionnel (du marché ou maison) est donc nécessaire, du moins tant qu'AWS n'améliore pas ses outils (ce dont nous ne doutons pas, à terme), ainsi que la mise en place des pratiques associées.

La seconde mauvaise raison part d'une bonne intention : c'est la surabondance du choix.
Chez **Gekko** nous l'appelons l'effet «Subway», du nom de la célèbre chaîne de restauration, par analogie avec la perplexité dans laquelle nous plonge la perspective de devoir faire notre choix dans une combinatoire de 5 types de pains, 4 variétés de viande, un grand nombre de crudités et légumes, plusieurs sortes de sauces et condiments, le tout en taille de 10, 15 ou 30 cm… C'est forcément plus riche que «jambon/crudités» ou «saucisson sec/cornichons», mais cela augmente le risque de déception, et impose donc plus de réflexion…
Quel rapport avec AWS ?
Eh bien dans EC2, nous avons le choix entre une dizaines de familles d'instances, réparties en 5 catégories (générales, optimisées pour le calcul, pour la mémoire, pour le stockage ou le calcul intensif), existant chacune sur 1 à 5 générations, et disponibles en 3 à 7 tailles selon les modèles ; si on ne retient que les générations actives (N et N-1), ceci ne représente environ «que» 70 types d'instances.

Celles-ci sont disponibles en 4 versions Windows et 4 versions Linux, dans 19 régions (dont une spéciale pour l'administration US) et 55 zones de disponibilité (situation mai 2018) ; 4 régions et 12 zones de disponibilité supplémentaires sont officiellement annoncées et en déploiement.
Enfin au-delà de la technique, il faudra choisir entre une tarification à la demande ou par réservation ; et dans ce dernier cas, il faudra décider si l'on veut que les instances soient convertibles ou non, avec une option régionale ou pas, et opter pour une durée de réservation (1 an ou 3 ans), ainsi que pour un mode de paiement (upfront, partial upfront, no upfront)…
Le Gartner a dénombré le nombre de choix possibles par factorisation de tous ces critères (et quelques autres), et arrivait en avril 2017 à un peu plus de 1.750.000 possibilités… pour EC2.

Et EC2 n'est qu'un des services, parmi la centaine offerte par AWS (103 en mai 2018)... Comme pour notre fameux sandwich, il est donc difficile de choisir ; et une fois choisi, il est difficile de se dire qu'on n'aurait pas pu faire mieux. Mais heureusement, à la différence des sandwichs, il existe là encore des outils pour nous aider à objectiver et optimiser nos choix, et c'est ce que nous verrons plus loin.

Un métier est donc né, absolument clé dans le processus d'adoption du Cloud dans les entreprises car garant de l'adhérence entre le design technique, le comportement de consommation et la finance.
Nous l'appelons pour notre part Cloud FinOps, et nous y distinguons 4 activités complémentaires :

››› Budget & Plan : business case, suivi du réel vs le budget (généralement avec un rebaselining car les activités basculées diffèrent de ce qui était prévu), analyse des changements, alertes si écarts importants, forecast, etc…

››› Showback/Chargeback : suivi détaillé de la consommation cloud (par compte, par service, par appli etc…), reventilation des coûts (avec ou sans refacturation interne), etc…

››› Cost optimisation : actions de baisse des coûts, par éliminiation du gaspillage, saturation des ressources, sélection des options tarifaires ou des services les plus appropriés, etc…

››› Scorecard : définition et suivi de KPIs pour aligner et impliquer les différents acteurs de l'entreprise dans un plan d'amélioration continue, communication aux métiers et fonctions support, etc…

Chez **Gekko**, nous avons eu la chance de participer à quelques-uns des premiers projets de Cloud FinOps, et d'échanger avec les principaux acteurs internationaux de ce domaine.
Nous vous présentons dans les pages qui suivent notre retour d'expérience, avec un focus sur l'optimisation financière.
Il ne tient plus qu'à vous de venir l'enrichir avec nous.

Les fondamentaux : RRRR ! (Reduce, Righsize, Reserve, Report)

Une célèbre styliste britannique disait à propos des habits :
«Buy less. Choose well. Make it last».
Nous trouvons pour notre part que cette maxime résume bien les fondamentaux de n'importe quelle méthodologie de réduction de coûts - si on y ajoute «l'ownership», la responsabilisation.

Appliqué au Cloud, nous classons les initiatives en 4 catégories.

Reduce : réduire la quantité consommée

Sur ce sujet, on pense d'abord spontanément au gaspillage, et c'est effectivement la première action évidente à entreprendre.
Toutes les sociétés ne sont pas logées à la même enseigne sur ce point, mais les datacenters hébergent assez naturellement des ressources oubliées ou inutiles (chez **Gekko**, nous les avons affectueusement nommées les «zombis») : volumes non attachés, VMs (voire serveurs) inutilisées, équipements réseau inactifs etc...
Ces zombis se développent d'autant plus facilement que les environnements sont instables ou dynamiques (environnements de test ou d'intégration, expérimentations d'applications, métiers innovants etc...), au gré des tests, des projets suspendus jusqu'à nouvel ordre, des ressources commisionnées «au cas où» ou «en attendant que», etc...

Cela n'est pas très gênant dans un datacenter dans la mesure où les zombis mobilisent des infrastructures existantes et utilisées par ailleurs, et génèrent donc un coût marginal faible.

Dans le Cloud en revanche, ces ressources inutilisées représentent un coût inutile direct qui peut être évité ; leur identification et leur élimination quotidienne (manuelle ou automatique, nous reviendrons sur ce point) sont donc indispensables.

Au delà de la chasse au gaspi/zombi, le deuxième levier de réduction de la quantité consommée repose sur une utilisation maximale d'une des caractéristiques fondamentales du Cloud - l'élasticité :

››› Il y a 168 heures dans une semaine.

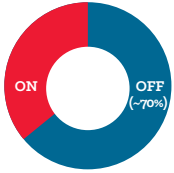
››› Le week-end représente 48 heures, soit près de 30% de l'ensemble ; une application pouvant être arrêtée le week-end devrait donc générer une économie de 30% sur les ressources associées.

››› Sur la base d'une journée travaillée de 10 heures, les nuits représentent 70 heures, soit près de 60% de la semaine hors week-end.

Il y a donc un nouveau gisement d'économies pour les applications ne supportant que les heures ouvrées : de manière globale, on a sur une semaine un gain de plus de 65% atteignable sur les applications qui peuvent être éteintes en heures non ouvrées.

››› Et il est possible d'aller plus loin en déclinant les jours fériés (une dizaine par an en France, selon les années et la coïncidence avec un week-end ou non), les vacances ou tout particularisme saisonnier d'une entreprise.

168 heures
dans une semaine



118 heures
de nuit & week-ends

Un exemple ?

Prenons une instance m5.xl, c'est à dire l'équivalent d'un serveur généraliste de 4 CPUs et 16 Gb de RAM.
Elle coûte environ 22 cents par heure (mai 2018, région Paris, Linux).

Quel sera son coût annuel si...

- elle n'est jamais éteinte, comme sur un datacenter	$1 962,24	
- elle est éteinte le week-end	$1 446,14	soit 26%d'économies
- elle est également éteinte la nuit	$602,56	soit 69% d'économies
- elle est éteinte 10 jours fériés et 3 semaines de vacances	$546,56	soit 72% d'économies

Multiplié par quelques dizaines d'instances (par exemple sur des environnements de test ou de développement), l'impact n'est pas négligeable.

Rightsize : maximiser l'usage

Si la première des habitudes à changer concerne l'extinction des ressources lors des périodes d'inutilisation, la seconde porte sur le dimensionnement des serveurs.

Sur ce point, le raisonnement se fonde sur la quantité de mémoire, le nombre et la puissance des processeurs, ainsi que sur les variations de charge dans une journée ou sur une période donnée : on dimensionne sur la crête afin de ne pas mettre en risque l'application business.

Il s'y ajoute souvent une composante de montée en charge ou d'évolution dans le temps : on dimensionne ainsi sur la crête à venir - en intégrant le fait que les serveurs devront tenir un certain temps - souvent 3 ans.

Résultat, l'utilisation moyenne des serveurs dans les datacenters est souvent très faible, cumulant ces multiples réserves de puissance.

Le problème est que de nombreux dimensionnements dans le Cloud intègrent ces pratiques, alors qu'il y a de multiples raisons de ne pas le faire :

››› Facilité à changer d'instance en cas d'évolution du besoin dans le temps - pas de nécessité de prévoir à trop long terme ;

››› Pour les applications supportant le scale-out, possibilité d'ajouter des ressources - éventuellement via auto-scale ;

››› Pour les applications ne le supportant pas, possibilité de « burster », c'est à dire d'augmenter temporairement la capacité des ressources ; sur ce point, l'introduction des instances t2 dîtes « illimitées » en décembre 2017 a levé un frein important [1] ;

››› Enfin et surtout, facilité à changer de type d'instance si on s'est trompé.

Un exemple ?

Reprenons notre m5.xl et son coût annuel de $ 1962
(donc environ $ 163 par mois).

Si elle a été dimensionnée trop largement dans la perspective d'une évolution du business, elle tournera sans doute sur une instance de la même famille mais plus petite : la m5.l coûtera la moitié ; par exemple, commencer sur une m5.l pendant 9 mois et ne passer en x.l que lorsque c'est nécessaire coûtera $ 1226 sur un an, soit 38% d'économies.
Si elle a été dimensionnée trop largement dans la perspective d'un appel de charge à hauteur de 1 à 2 heures par jour, mais que l'application autorise le scale-out, elle pourra également tourner sur une m4.l la majorité du temps et 2 m4.l lorsque nécessaire (coût annuel $ 1063, soit 46% d'économies) si l'application le permet ; si l'application ne le permet pas, elle pourra peut-être tourner sur une t2.l aux performances assez proches, et «burster» lorsque nécessaire afin de délivrer de la puissance supplémentaire occasionellement (coût annuel $ 925, soit 53% d'économies).

[1] *Les instances t2 offrent la capacité de burster, c'est-à-dire d'augmenter temporairement de puissance pendant une période liée à des niveaux de crédits, accumulés dans le temps, en consommant moins que la puissance nominale de l'instance ; très adaptés à des crêtes, ces bursts ont toutefois une durée limitée, ce qui peut susciter des problèmes de performance ; les nouvelles instances t2 «illimitées» s'affranchissent de ce frein, en autorisant le dépassement de puissance aussi longtemps que nécessaire – moyennant facturation complémentaire.*

Nous voudrions conclure cette section par un type de rightsizing propre au Cloud, qui est la chasse aux instances d'anciennes générations.

AWS fait effectivement évoluer rapidement sa gamme d'instances, les nouvelles offrant généralement des performances meilleures pour un coût légèrement plus élevé ; mais cette situation tarifaire évolue très vite, et les instances de nouvelles génération sont bientôt au prix des anciennes - voire moins chères.

Exemple avec des instances de famille « calcul optimisé » (type c), très répandues dans les configurations client : les instances c5 ont été lancées en fin d'année 2017, les parcs sont principalement constitués de c4, mais on trouve encore énormément de c3.

- Prix d'une c3.xl (7.5 Gb RAM, 4 vCPUs) Linux en Irlande : $ 0,239 / heure (mai 2018)
- Prix d'une c4.xl (7.5 Gb RAM, 4 vCPUs) Linux en Irlande : $ 0,226 / heure (mai 2018), soit 5% moins cher
- Prix d'une c5.xl (8 Gb RAM, 4 vCPUs) linux en Irlande : $ 0,192 / heure (mai 2018), soit 15% moins cher

Or les c5 sont plus récentes et plus puissantes que les c4, lesquelles le sont plus que les c3 : nous avons appliqué un benchmark avec l'outil Geekbench 3 (21 tests integer, floating point et memory) et obtenons les résultats suivants : score de 6421 pour la c3, de 7504 pour la c4 (17% plus puissante) et de 8485 (32% plus puissante vs la c3 et 13% vs la c4).

Il y a donc possibilité de gagner de la puissance en baissant la facture (voire de baisser beaucoup la facture, par exemple en profitant pour passer sur une c5.l si le taux d'utilisation n'était pas optimal).

Notons que le problème est moindre dans les régions récentes (par exemple il n'y a pas d'intances c3 en France – mais beaucoup de clients français utilisent la région Irlande).

Reserve : tirer partie du modèle tarifaire d'AWS

AWS offre la possibilité de bénéficier d'une réduction tarifaire significative (moyenne de 40 à 60% selon durée et modalités de paiement en contrepartie d'un engagement pour une période donnée (1 ou 3 ans), sur un type de ressources donné.

Voici par exemple ce que cela donne pour notre instance m5.xl Linux en France (toujours en données de mai 2018) :

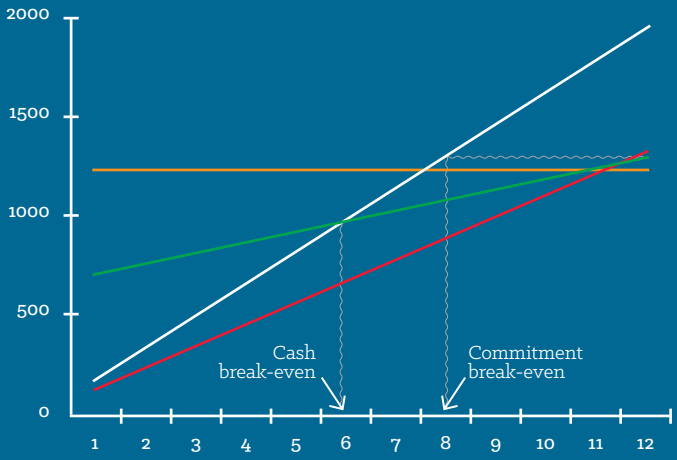
››› Pour mémoire son coût horaire est de $ 0,224, soit environ $ 1962 par an ;

››› Si on réserve sur un an (mode «standart») en payant tout en avance, on paiera $ 1288 pour un an, soit 34% d'économies ;

››› Cette même réservation sur un an en payant partiellement d'avance donne un premier paiement de $ 656 et un paiement horaire de $ 0,076, soit un total annuel de $ 1314, une économie de 33% ;

››› Enfin, la réservation sur 1 an sans paiement upfront donne un paiement horaire de $ 0,157, soit un total annuel de $ 1375, une économie de 30%

En voici une illustration graphique :



à l'heure

RI standard 1 an (partial upfront)

RI standard 1 an (no upfront)

RI standard 1 an (full upfront)

Commentaires importants :

Il y a bien un engagement pris sur un an (dans l'exemple) à payer la totalité des heures, qu'elles soient consommées ou pas ; il faut donc bien distinguer le point d'équilibre en trésorerie (croisement des lignes blanche et verte dans l'exemple : environ 6 mois) du point d'équilibre d'engagement (nombre de mois sur la courbe blanche correspondant à la dépense d'une année de la courbe verte), qui reflète la vraie charge annuelle.
Comme il y a engagement à payer toutes les heures, c'est une évidence mais il faut être assez sûr de la permanence du besoin : la courbe bleue est établie avec une saturation de 100%, mais ne croiserait plus aucune courbe à 60%, et les gains seraient insignifiants à 70%.
D'autre part cet engagement se base sur le coût de la ressource au moment de la souscription - dans un contexte de baisse fréquente des prix (64 baisses depuis la création, soit plus d'une par trimestre - situation avril 2018)

On peut donc se dire à première vue que c'est intéressant mais totalement contradictoire avec les objectifs de flexibilité précédemment vus.
En fait non, car AWS a introduit une grande dose de subtilité (mais aussi de complexité) qui permet de combiner flexibilité et réservation :

Sur le principe même tout d'abord, la réservation ne s'applique pas à une instance donnée mais à un type d'instance. Il faut voir chaque réservation comme un «coupon» de 730 heures par mois (nombre moyen d'heures par mois dans une année de 365 jours ; dans les faits, chaque mois dispose d'un coupon correspondant au nombre réel d'heures) achetées à prix discounté, et qui va être utilisé à chaque fois qu'une instance cadrant avec l'objet de la réservation sera démarrée.

Ce coupon s'appliquera donc aussi bien à une instance précise qui reste allumée tout le temps, qu'à plusieurs instances de même type qui fonctionneraient de manière alternative - avec toutefois un plafond horaire (il ne peut y avoir plus de 3600 secondes de fonctionnement éligible au coupon par heure : 4 instances fonctionnant chacune 15 minutes dans une heure donnée utilisent le même coupon, même si leur fonctionnement est simultané ; mais si l'une d'elle travaille 20 minutes au sein de la même heure, les 5 minutes supplémentaires seront au tarif à la demande).
Mieux, ce coupon s'appliquera non seulement aux instances du compte AWS dans lequel il a été souscrit, mais il bénéficiera également aux autres comptes du même groupe de facturation (cas des comptes liés et de la facturation consolidée, s'il y en a).

NB : ceci implique le grand intérêt qu'il y a à limiter le nombre de types d'instances utilisés : face à la richesse du catalogue AWS, il est tentant de multiplier les familles et les tailles afin de coller au plus près des demandes des développeurs ou des spécificités des applications, mais la standardisation des instances augmente le levier d'utilisation des instances réservées.

Sur les possibilités de modification ensuite, la réservation n'est pas figée et permet en théorie de changer l'objet auquel elle s'applique en cas d'évolution du besoin (on reste engagé, mais on peut utiliser cet engagement sur quelque chose d'utile - sauf si l'on quitte AWS auquel cas cet engagement est dû et perdu).

En pratique c'est assez complexe, du fait du grand nombre d'options qu'offre AWS et des fréquentes évolutions de modèle.

Dans les grandes lignes, il existe :
››› Des instances réservées «standard», qui peuvent s'appliquer à une zone de disponibilité (AZ) ou à une région. La principale possibilité de changement est limitée à la taille de l'instance sur Linux, via une table de normalisation (par exemple passage d'une M4.xl à deux M4.l) ; à noter que ce changement s'applique automatiquement (Size Flex) pour les instances Linux ou Linux AWS régionales.
››› Des instances réservées «convertibles», qui peuvent être échangées contre d'autres modèles (ce qui dépasse donc la taille pour inclure la famille et l'OS) sous réserve que l'engagement financier correspondant soit supérieur ou égal ; à noter qu'elles offrent un discount inférieur aux instances «standard».

Précisons également qu'il existe une marketplace pour revendre à d'autres clients des instances réservées inutiles ; ce sera peut-être un jour une possibilité intéressante, mais c'est actuellement très théorique : fonctionnement beta (depuis 2012), nécessité d'avoir un compte en dollars, très très faible liquidité du marché (~2000 instances sur le marché en mai 2018 sur plusieurs millions fonctionnant chez AWS), mouvements à 80% sur les régions et AZ US.

Le tableau ci-contre résume la situation à ce jour (mai 2018) :

	Standard		Convertible	
	Zonale	Régionale	Zonale	Régionale
Durée	1 an - 3 ans		1 an - 3 ans	
Paiement	No/Partial (50%) / Full upfront		No/Partial (50%) / Full upfront	
Économie	≈ 40-60%		≈ 45%	
Capacité garantie	Oui	Non	Oui	Non
Changement AZ dans région	Oui	n/a	Oui	n/a
Changement région	Non		Non	
Changement OS	Non		Oui	
Changement taille	Oui (Linux)	Size flex (Linux)	Oui (si engagement financier supérieur ou égal)	
Changement famille	Non		Oui (si engagement financier supérieur ou égal)	
Eligible aux baisses de prix	Non		Oui (si engagement financier supérieur ou égal)	
Revendable sur market place	Oui (mais liquidité du marché ?...)		À venir	

Comment se repérer et faire son choix dans tout cela ?
Il n'y a évidemment pas de règle absolue et tout dépend de l'entreprise, des profils de charge, des politiques de gestion de trésorerie, etc...
D'une manière générale, la question peut se poser pour des environnements très flexibles et fréquemment éteints (dev & test, fonctionnement limité aux heures ouvrées) mais les instances réservées sont globalement rentables dès qu'il y a des charges intensives et une forte standardisation des instances ; ceci est notamment dû à la rapidité d'atteinte du break-even (très proche du 8ème mois pour les engagements d'un an, et du 15ème pour les engagements de trois ans), qui permet éventuellement de «jeter» des instances réservées sans trop de casse si une modification stratégique ou technique de grande ampleur intervient.

Petite digression non financière sur les instances réservées

Avant d'être une option tarifaire, les instances réservées sont... réservées.
Il faut se souvenir que le Cloud ne garantit pas à un client que les ressources qu'il souhaite acheter soient effectivement disponibles «en stock» (ni sur AWS ni sur les autres plateformes) : il peut arriver que tel ou tel modèle d'instance ne soit pas disponible à un instant donné sur telle zone de disponibilité, dans telle région (c'est rare, mais nous l'avons vu). On peut d'ailleurs imaginer des situations (sinistre de zone) générant un afflux de demandes simultanées qui ne pourraient être servies.
La réservation d'instance permet de pallier à cela et de garantir l'accès à la ressource, puisqu'elle est effectivement «réservée» dans le capacity planning d'AWS ; c'est donc une caractéristique intéressante en soi, par exemple pour une solution de Disaster Recovery.

(NB : la réservation implique la sélection d'une zone de disponibilité ; l'option «régionale» fait perdre cette garantie).
Cette caractéristique s'accompagne d'une particularité dans le cas de comptes liés : si le bénéfice tarifaire se partage entre les comptes (application en priorité au compte ayant souscrit, usage de l'excédent par les autres comptes), la capacité garantie ne s'applique en revanche qu'aux seuls comptes souscrivant des réservations. Il faut donc bien les choisir (par exemple comptes de production ; proscrire à l'inverse le compte de facturation consolidée, qui se limite à un bucket S3…).

Report : responsabiliser les utilisateurs

Les dépenses dont personne ne se sent responsable grandissent vite. Il est à ce titre salutaire de fournir à chaque classe d'utilisateurs une information claire sur les ressources dont il a la charge et sur lesquelles il peut agir.
Cette information est trop souvent mensuelle, à l'occasion de la facturation, ce qui fait perdre l'occasion d'enclencher des actions au fil de leur nécessité (c'est d'ailleurs pour cela que l'optimisation de coûts cloud n'est pas une action «coup de poing» mais bien une discipline quotidienne à mettre en place et faire vivre…) :
››› Suivi quotidien des ressources sous ou pas utilisées, pour les équipes techniques,
››› Suivi technique et financier régulier sur les différents services utilisés, et leur tendance, pour le responsable de production,
››› Bilan financier au minimum hebdomadaire pour les responsables de centres de coût (applications, domaines ou métiers), et le contrôleur de gestion,
››› Heatmap (matrice présentant des taux d'utilisation pour un groupe de ressources par jour/heure) pour des responsables applicatifs afin d'aider le dimensionnement,
››› etc…

Afin d'effectuer ceci de la manière la plus parlante, il est fondamental de ventiler les coûts - qu'il y ait ou pas une facturation interne - selon les différentes entités utilisatrices.
AWS permet d'allouer des tags à chaque ressource, couvrant aussi bien des informations techniques que fonctionnelles.
Il est donc possible de relier chaque ressource à différents agrégats : application, domaine (ex : développement, intégration, production…), service utilisateur (et donc payeur) etc…
Le nombre de tags crées par l'utilisateur est limité à 50 mais ce n'est généralement pas un problème. Les difficultés se situent plutôt dans la taxonomie des tags (cohabitation de «dev», «DEV», «dévelop» etc…), dans leur non systématisation (dans une perspective de facturation, il faut que chaque ressource ait un tag permettant l'allocation de coûts - faute de quoi on s'oblige à de pénibles retraitements de coûts sous-absorbés), et dans la consolidation d'informations provenant de plusieurs comptes selon des systèmes de tags communs.
A noter qu'AWS permet dorénavant (depuis mars 2017) d'automatiser la création des tags lors de la création d'instances EC2 ou de volumes EBS, ou de créer des politiques utilisateurs forçant l'adjonction de tags lors de la création de ressources.
Attention, les tags ne sont pas retro-actifs, et il sera donc difficile de construire a posteriori une ventilation des coûts : si celle-ci est envisagée (et elle finira par l'être), il faut tagger rapidement.

La Scorecard Cloud FinOps, un excellent outil de communication et d'alignement avec les métiers et les fonctions support.

Plus la facture cloud grandit, plus il devient fondamental de démontrer aux différents acteurs de l'entreprise que cet argent n'est pas dépensé en vain… Il est même judicieux de démontrer qu'au contraire, la quantité de workloads ou le nombre de services ont augmenté comparativement plus vite que la facture…

Pour cela, la solution idéale est de définir une ou plusieurs métriques «business», sur lesquelles indéxer la facture ou certains de ses composants : coût du clic ou d'un affichage de message pour des services internet, coût de traitement d'une transaction ou d'un process, coût de l'utilisateurs etc…

D'autres KPIs peuvent ête suivis avec intérêt : taux de couverture RI, taux de perte d'heures RI, ratio d'élasticité (coût horaire moyen en jour vs nuit, semaine vs week-end) etc…

La définition d'objectifs chiffrés et d'actions associées, ainsi que leur suivi régulier, sont les meilleurs moyens d'assurer une amélioration continue de la gestion financière du cloud.

Aller plus loin

L'application des mesures RRRR fait généralement gagner 20-25% de la facture lors des actions initiales, et permet surtout de maintenir l'efficacité dans le temps.
Il y a toutefois de nombreux autres leviers d'action, et nous proposons ci-dessous une liste non limitative (et surtout rapide : beaucoup de ces sujets sont complexes - par exemple tous ceux qui touchent au re-platforming, ce n'est pas le but de ce document de les développer ici) :

Dépasser EC2

EC2 représente généralement la majorité de la facture AWS mais cela ne signifie pas qu'il n'y a rien d'autre à optimiser : le tiering du stockage (type de support sur EBS ; possibilité de «infrequent access» sur S3 ; Glacier) offre de nombreuses possibilités, la tarification par réservation ne s'applique pas qu'aux serveurs virtuels (ex : RDS, DynamoDB, Aurora, Redshift, Elasticache) et les coûts de réseau justifieraient un document dédié à eux seuls (minimisation du transfert inter-régions, utilisation d'adresses IP privées, utilisation de Cloudfront, utilisation de DirectConnect...).

Buy vs Make

Utiliser le Cloud suppose déjà une ouverture à consommer un service plutôt que le construire. AWS permet d'aller très loin dans cette approche, avec de très nombreux services applicatifs qui produisent des économies de licences et de montée en charge.

Au niveau des infrastructures, le service RDS permet de créer un moteur de base de données sécurisé, incluant la redondance et les sauvegardes ; outre la simplification qu'il apporte, il génère des économies directes (notamment de licences grâce au modèle de paiement à l'usage) et indirectes (charge de service évitée pour les backups, par exemple) ; nous n'évoquons pas ici les migrations de base de données (Oracle vers Aurora, par exemple), mais c'est évidemment une source d'économies considérables.

Au niveau du développement, il est possible de maximiser l'usage des ressources via l'utilisation de conteneurs (AWS ECS), ou même de s'affranchir de l'infrastructure par une approche serverless (AWS Lambda). Les niveaux d'intégration sont différents (on continue à construire son infrastructure, son mode de scaling etc... dans le premier cas, on ne s'en occupe plus du tout dans le second ; on est totalement libre de ses choix de langages et de programmation dans le premier, moins dans le second ; la réversibilité est assez simple dans le premier, moins dans le second) et les leviers d'économie sont différents (densification dans un cas ; facturation sur événement dans le second, particulièrement adaptée à des montées en charge ou à une variabilisation vs un usage business, par exemple IoT) ; les résultats en termes de facilité et de rapidité de déploiement ou d'économies sont en revanche dans les deux cas importants.

Oser les instances Spot

AWS offre la possibilité d'acheter des instances à très bas prix en effectuant une offre sur leurs capacités libres. Le prix dépend de l'offre et la demande - on constate généralement un range de 60% à 90% d'économies par rapport au prix à la demande.

Il y a évidemment une contrepartie : AWS reprend ces instances avec un préavis très court (2 minutes) si le prix de ces capacités vient à excéder le seuil maximal que l'on a défini.

C'est donc un usage réservé à des charges interruptibles ou (et surtout) à des charges répartissables par design sur une flotte d'instances (AWS propose d'ailleurs des automatismes permettant de définir de tels groupes).
Il est également possible de définir un prix maximal assez haut (mais cependant éloigné du prix à la demande) afin de limiter le risque d'interruption (la console donne accès à l'historique du prix sur 3 mois pour chaque instance).

Consolider les comptes

La plupart des services d'AWS offrent un discount selon le volume.
Ceci vaut également pour les frais de support (passage de 10% pour la première tranche à 3% pour la dernière).
Il y a donc un intérêt évident à lier tous les comptes dans un compte de facturation consolidé - la difficulté résultant souvent à identifier ceux qui ont peut-être été ouverts hors des circuits traditionnels de l'IT.

Considérations pratiques

L'outillage

Comme nous l'évoquions, il est certainement possible de faire tout cela à partir des outils standards AWS… mais en ce qui nous concerne nous n'y arrivons pas : trop de données à rapprocher, retraiter, visualiser pour arriver facilement à des conclusions efficaces.
Il existe plusieurs outils sur le marché (Cloudhealth, Cloudcheckr, Cloudability, Cloudyn etc…), dont un français (Teevity), construit sur la base de la plateforme opensource ICE issue du projet FinOps de Netflix.
Ils s'avèrent rapidement indispensables pour effectuer très facilement les tâches que nous avons évoquées précédemment : traquer le gaspillage, évaluer la sous-utilisation, calculer les meilleurs compromis d'instances réservées, bâtir divers niveaux de reporting pour les différentes fonctions de l'entreprise, faciliter la facturation interne etc…

Ils ont tous leurs forces ou faiblesses (orientation reporting ou optimisation, couverture multicloud, fonctionnalités, ergonomie, tarification, souplesse et personnalisation…) et correspondent à des typologies différentes de taille de projet client : on ne cherchera pas le même outil pour rationaliser l'usage d'un environnement monocompte de 15 ou 20 k€/mois que pour établir une facturation multicloud/multi-comptes de plusieurs centaines de k€ mensuelles, avec introduction d'unités d'oeuvre métier.
Chez **Gekko** nous en utilisons plusieurs, en fonction des cas que nous rencontrons ; nous avons également construit notre propre plateforme avec l'un d'entre eux afin d'administrer la fonction FinOps (reporting, facturation, optimisation) de certains de nos clients.

Nous voyons également quelques cas (généralement de très grands comptes), où des outillages «maison» ont été construits - solutions sans doute très bonnes, au bémol de leur maintenabilité et évolutivité (ou du coût de celle-ci).
Enfin nous voyons aussi... ce que nous ne voyons pas encore, et que nous appelons de nos voeux : des outils de mesure et de prévision financière sur lambda (OK, nous savons bien que cela dépend du codage et du contexte applicatif, mais avec du big data et du machine learning, on peut rêver...), des outils de modélisation de budget selon des projections d'usage - dans un contexte de migration ou de meilleure utilisation etc... Si vous aussi vous pensez à ce genre de choses, faites-nous signe, nous serons ravis de réfléchir ensemble !

L'automatisation

D'une manière générale, le cloud perd de sa saveur sans automatisation...
Les FinOps n'échappent pas à la règle et il y a de nombreux sujets à creuser : détection automatique de ressources insuffisamment utilisées ou orphelines, automatisation de start/stop, détection de ressources non taggées, surveillance et automatisation de re-création d'instances spot, automatisation de changement de taille d'instances etc...

Certaines de ces fonctions sont couvertes par les outils pré-cités, d'autres gagnent de toute manière à être insérées dans les process de production.

La durée

Nous avons rencontré beaucoup de clients qui souhaitaient une analyse de leur facturation assortie de recommandations, mais qui voyaient mal l'intérêt de s'outiller dans la durée.

Il est vrai que la campagne initiale est généralement spectaculaire, avec ses 20-25% de gains.

L'expérience montre qu'ils peuvent être vite perdus.

D'abord ils peuvent tout simplement ne jamais être mis en oeuvre…

La nature humaine étant ce qu'elle est, un suivi dans la durée favorise l'exécution effective des actions, qui auraient pu être vite oubliées sinon.

Ensuite, à moins d'avoir une production totalement stable (mais alors, faut-il vraiment être dans le cloud ?!), de nouveaux services vont arriver, de nouveaux composants vont être intégrés et il faudra les optimiser ; tout ceci sans même parler de tirer profit des évolutions tarifaires d'AWS.

Notre sujet relève donc bien de la course de fond et non du sprint.

L'ownership

Conséquence du point précédent, nous relevons que les clients qui sont les plus performants sur le sujet sont ceux qui en ont clairement confié la responsabilité à quelqu'un.
Cela ne signifie pas que c'est un travail à plein temps (mais cela peut l'être chez les grands utilisateurs de cloud, et les gains le justifient pleinement), mais compter sur la bonne volonté ou le comportement vertueux des uns et des autres ne suffit généralement pas.

En guise de conclusion

Nous espérons que ce petit document vous aura intéressé,
ou au moins convaincu de l'importance du sujet.

Si c'est le cas et que nous pouvons vous aider, contactez-nous,
nous serons ravis de le faire.
Si c'est le cas et que vous souhaitez contribuer au développement de nos réflexions et expériences sur le sujet, contactez-nous également ! L'utilisation du cloud à grande échelle dans les entreprises est encore un phénomène suffisamment récent, la discipline de Cloud FinOps est encore jeune et nous sommes pleinement conscients qu'il reste beaucoup à faire et à co-construire. C'est dans cet esprit que nous avons créé le meetup de Cloud FinOps, afin que les bonnes expériences et pratiques puissent être partagées, vous y êtes les bienvenus.

Et si ce n'est pas le cas, peut-être serez-vous plus intéressé par notre anti-guide final ?!

5 moyens sûrs de perdre de l'argent avec le Cloud :

Faire comme d'habitude

«Notre IT fonctionne en 24x7, il n'y a rien qu'on peut arrêter. Bon peut-être le développement et les tests la nuit et le week-end.
Et puis cette appli aussi. Mais c'est compliqué.
Et on n'est pas sûrs que ça redémarrera. Et puis ça ne va pas chercher bien loin».

Attendre que la situation soit stable pour réserver des instances

«Là on a bien ces instances qui sont 'on' depuis 6 mois, c'est dommage on aurait déjà gagné de l'argent avec des réservations, mais on n'est pas sûrs qu'on ne va pas en changer donc on attend ; et il faut encore qu'on y voit clair avec les futurs besoins avant de s'engager.»

Dimensionner comme d'habitude

«Nos applications ont toujours eu besoin de 8 vCPUs et 32 Gb de RAM sur le datacenter, donc si on veut être sûrs que ça marche toujours, il faut la même chose dans le cloud. D'ailleurs on sera bien contents s'il y a un gros pic comme on avait eu ce fameux jour l'an dernier».

Compter sur la discipline des utilisateurs

«OK ils ont démarré plein d'instances, mais c'était pour essayer - d'ailleurs le cloud c'est fait pour ça, non ?
Et nos équipes savent se tenir, ils les arrêteront quand ils n'en auront plus besoin».

Laisser les budgets où ils sont

«Ce compte est associé au projet big data, c'est noyé dans l'ensemble et porté par le métier ; celui-ci est dans le budget de développement, tout le monde contribue ; et celui-là est dans les frais généraux, personne n'y verra rien.»

À propos de Gekko

Société de conseil et d'intégration, **Gekko** est spécialisée dans la stratégie et la réalisation des migrations vers le Cloud.

Nous accompagnons nos clients dans la conception, le déploiement et la maintenance d'une infrastructure Cloud flexible, connectée et sécurisée, 100% DevOps.

En combinant notre expertise technique pointue et notre profonde connaissance de la production informatique, nous vous permettons de tirer le meilleur parti du Cloud.

Créée fin 2015, **Gekko** est Advanced Partner AWS et compte aujourd'hui près de 50 consultants spécialisés AWS, DevOps et microservices.

Gekko
12 rue d'Alsace,
92300 Levallois-Perret
T + 33 (0) 158 744 600

Dans ce petit livre, **Gekko** vous présente une discipline naissante mais indispensable pour mettre le Cloud sous contrôle dans l'entreprise : les Cloud FinOps.

Découvrez comment faire baisser votre facture Cloud, optimiser l'utilisation de vos ressources, impliquer vos utilisateurs, ventiler vos coûts et préparer la facturation interne.

gekko
les ingénieurs du cloud