Chapitre 3
Les services de calcul sur AWS

1. Introduction

Dans les coulisses de tout projet cloud, la question du calcul, autrement dit la façon dont s'exécute le code et dont on alloue les ressources informatiques demeurent essentielles.

Que vous soyez développeur, architecte ou simple curieux des technologies AWS, le choix de la solution de calcul adaptée a un impact sur la performance, la fiabilité et même la facture de votre application.

Dans ce chapitre, nous explorons les principaux services de calcul proposés par AWS : EC2, ECS, EKS et Lambda. Chacune de ces options possède sa propre logique d'utilisation : d'un côté des machines virtuelles configurables (EC2), de l'autre un environnement « serverless » où l'on s'affranchit de la gestion des serveurs (Lambda). Entre ces deux extrêmes se trouvent les conteneurs managés via ECS ou orchestrés via Kubernetes (EKS), qui offrent un compromis intéressant pour la mise en place de microservices portables et évolutifs.

Nous commencerons par une vue d'ensemble, puis nous plongerons dans Amazon EC2 et ses bonnes pratiques. Nous examinerons ensuite la montée en charge à l'aide d'Autoscaling et de Load Balancers. Nous verrons comment choisir la bonne solution de conteneurs entre ECS, EKS et Fargate et découvrirons l'univers serverless d'AWS Lambda. Enfin, un comparatif détaillé vous aidera à sélectionner le service le mieux adapté à vos projets.

Que vous découvriez AWS ou souhaitiez affiner vos connaissances, ce chapitre vise à offrir une vue d'ensemble et des conseils pratiques.

2. Présentation des options de calcul (EC2, ECS, EKS, Lambda)

2.1 Une brève histoire du compute sur AWS

Lancé publiquement en 2006, Amazon EC2 (*Elastic Compute Cloud*) a rapidement pris la forme de «vitrine» du cloud, permettant de louer de la puissance de calcul à l'heure, aujourd'hui facturée à la seconde pour la plupart des instances Linux (avec un minimum de 60 secondes), tandis que Windows et certaines distributions commerciales restent facturées à l'heure. Cette flexibilité était en rupture avec l'achat et la gestion de serveurs physiques traditionnels.

Les besoins ont progressivement évolué vers davantage d'abstraction, notamment via les conteneurs Docker. AWS a répondu en introduisant Amazon ECS (*Elastic Container Service*), puis un service Kubernetes managé, Amazon EKS (*Elastic Kubernetes Service*). Plus tard, le service AWS Lambda a fait son apparition pour permettre l'exécution de code sans gestion de serveurs, selon un modèle « serverless ».

On peut résumer ces quatre piliers de calcul comme suit :

– **EC2** procure des machines virtuelles (VM) offrant un contrôle total.
– **ECS** déploie des conteneurs Docker managés par AWS.
– **EKS** orchestre des conteneurs Kubernetes, mais sur AWS.
– **Lambda** exécute des fonctions sans nécessiter la gestion d'infrastructures.

Le choix dépend principalement du degré de contrôle souhaité sur l'OS, du type d'application, de la culture d'entreprise (Kubernetes ou non) et de l'effort à fournir pour gérer les serveurs et le système d'exploitation.

2.2 Les différents modèles d'exécution

Dans le modèle IaaS (EC2), l'utilisateur est responsable de tout ce qui se situe au-dessus de la virtualisation. AWS fournit les serveurs physiques, la couche hyperviseur et la mise à disposition de l'instance, tandis que vous devez installer et maintenir l'OS, les dépendances et votre code. Ce degré de liberté se paie par la nécessité de patcher et de monitorer vos machines, ainsi que de concevoir vos stratégies d'échelle.

Pour le modèle CaaS (ECS ou EKS), l'application est emballée dans un conteneur Docker. L'orchestrateur (ECS ou Kubernetes via EKS) planifie et gère ces conteneurs. L'isolement est semblable à un mini système d'exploitation, et plusieurs conteneurs peuvent s'exécuter sur la même machine ou être dispersés. L'avantage est de se concentrer davantage sur le code et le déploiement, tout en acceptant qu'une couche d'infrastructure reste à administrer (ou déléguée, via Fargate).

Enfin, le modèle FaaS (Lambda) correspond à un environnement «serverless» où l'on écrit uniquement des fonctions invoquées à la demande. On ne paie que le temps d'exécution, sans devoir gérer des instances en continu. C'est une solution idéale pour des tâches ponctuelles ou événementielles, avec une facturation granulaire à la milliseconde, mais soumise à certaines limites (durée, mémoire, accès disque).

Quand privilégier l'une ou l'autre ?

EC2 est un bon choix pour les migrations *lift & shift* (réhéberger une application telle quelle, sans la réécrire) ou lorsque le contrôle du système d'exploitation est indispensable. ECS/EKS conviennent bien à ceux qui utilisent déjà Docker ou Kubernetes et recherchent un déploiement flexible. Lambda s'applique particulièrement à l'événementiel, aux workloads de courte durée et aux contextes où l'on veut réduire la maintenance système.

3. Amazon EC2 : fonctionnement et bonnes pratiques

EC2 en quelques mots

Amazon EC2 (*Elastic Compute Cloud*) permet de déployer et configurer des machines virtuelles (instances) dans le cloud AWS. Chaque instance correspond à des ressources choisies (vCPU, RAM, stockage, débit réseau), que l'on peut adapter parmi de nombreuses catégories (T, M, C, R, G, P, etc.). On est facturé en fonction de l'utilisation, et il est possible de redimensionner, d'arrêter ou de démarrer les instances à volonté.

Historiquement, EC2 est l'un des services phares d'AWS, car il offre la possibilité de tout faire sur une instance, en contrepartie de la charge liée à l'OS (patching, sécurité, mises à l'échelle).

3.1 Fonctionnement

Pour créer une instance, on sélectionne une AMI (*Amazon Machine Image*) qui définit l'OS et les packages initiaux. On choisit ensuite le type d'instance (t2.micro, m5.large, etc.), les paramètres réseau (VPC, subnet, IP), ainsi que les options de sécurité (Security Group, clés SSH, mots de passe). Après le lancement, on peut se connecter et configurer librement l'instance.

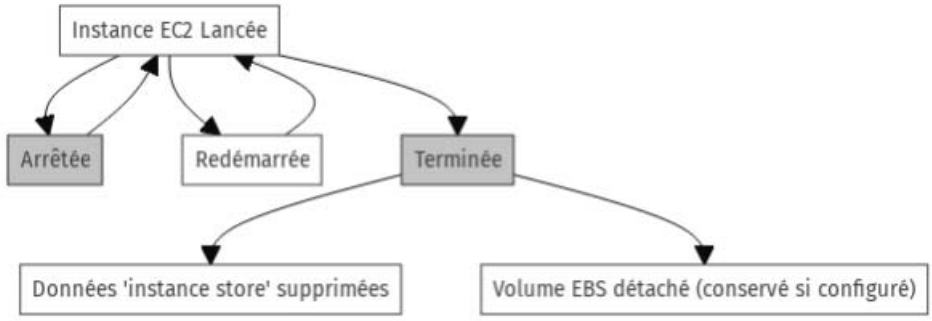
Écran de création d'un EC2 sur AWS

Volumes EBS

Par défaut, on utilise souvent un volume EBS (*Elastic Block Store*) comme disque principal. C'est un stockage persistant et chiffrable dont la taille et les performances varient selon les types : gp3 (usage général), io2 (usage intensif) ou st1/sc1 (HDD moins coûteux).

Cycle de vie des instances

Une instance peut être **démarrée, arrêtée, redémarrée ou terminée**. À l'arrêt, vous ne payez plus l'instance, mais vous continuez à payer le volume EBS. En cas de terminaison, l'instance est détruite et les données de type « instance store » disparaissent, alors qu'un volume EBS détaché peut être conservé.



Cycle de vie des instances EC2

Bonnes pratiques

Il est conseillé d'ajouter des tags à vos instances (indiquant par exemple l'équipe propriétaire ou l'environnement) pour simplifier la gestion. Sécurisez vos instances via un rôle IAM (*Instance Profile*) avec des permissions minimales, et appliquez régulièrement les correctifs de sécurité. Surveillez également les métriques avec CloudWatch, planifiez des sauvegardes (snapshots EBS) et n'hésitez pas à changer de type d'instance pour ajuster vos ressources.

Exemple d'utilisation

Pour un petit serveur web, on peut lancer une instance **t2.micro**, y installer un serveur web (Apache, Nginx), attacher une IP publique et ouvrir le trafic HTTP/HTTPS via un Security Group. Ce type d'instance coûte peu si elle reste allumée en continu. Pour une configuration plus robuste, on peut créer plusieurs instances dans un Autoscaling Group réparti sur plusieurs zones de disponibilité, avec un basculement automatique en cas de panne.

3.2 Autoscaling et Load Balancing : gérer la montée en charge

Le principe du scale-out et scale-in

L'un des grands avantages du cloud est la capacité d'ajuster dynamiquement les ressources selon la charge. On peut ajouter des instances (*scale-out*) lorsqu'un pic survient, puis en retirer (*scale-in*) pour réduire les coûts lorsque l'activité baisse. Ce mécanisme repose sur des métriques telles que l'utilisation CPU ou le nombre de requêtes HTTP.

Les politiques de scaling (simple, step, ou target tracking) permettent de définir les seuils et les réactions souhaitées. Avec le « *target tracking* », on cherche à maintenir un pourcentage cible (ex. 50 % CPU) ; l'Autoscaling Group (ASG) crée ou supprime des instances pour rester près de cet objectif.

Elastic Load Balancing (ELB)

Un Load Balancer distribue le trafic entre plusieurs instances et surveille leur état. AWS propose plusieurs types de load balancers : l'Application Load Balancer (couche 7), le Network Load Balancer (couche 4) et le Gateway Load Balancer (pour la sécurité et le filtrage). Coupler un ELB à un ASG permet de ne router le trafic qu'aux instances saines et de démarrer ou d'arrêter automatiquement des instances au besoin.

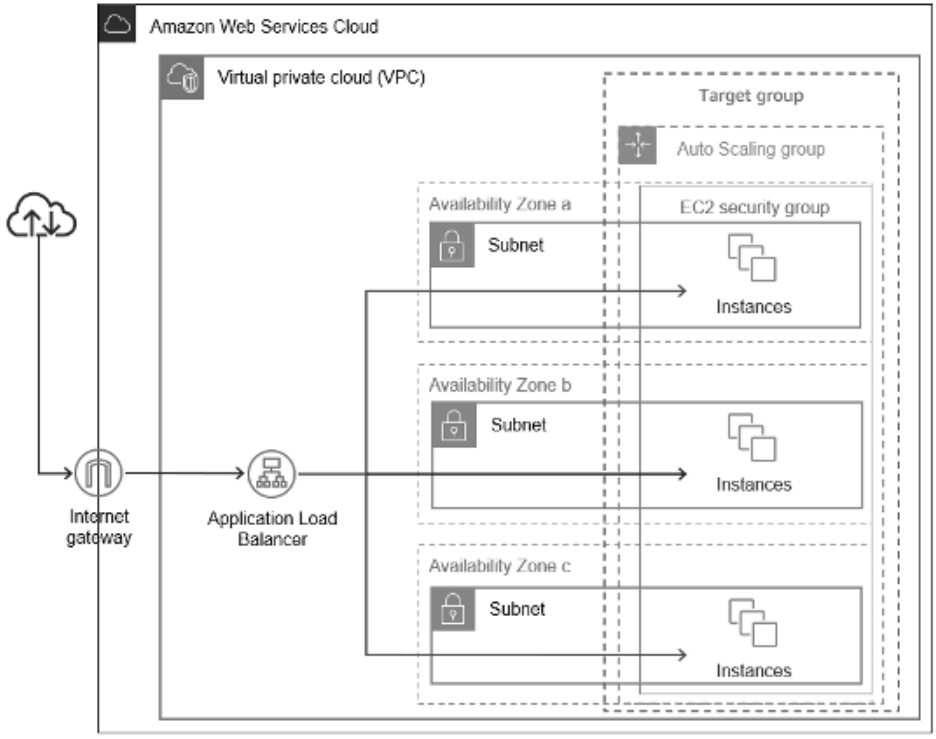
Stratégies d'Autoscaling

Le scaling peut être horizontal (ajout d'instances) ou vertical (changement de type d'instance pour une configuration plus puissante). Le scaling horizontal est plus flexible et plus tolérant aux pannes, tandis que le scaling vertical est limité par les tailles disponibles et nécessite souvent un redémarrage.

Au sein d'un ASG, on définit un nombre minimum (pour assurer la haute disponibilité), un nombre désiré pour la charge courante et un nombre maximum pour éviter les coûts démesurés. On configure également une période de « cooldown » pour éviter des oscillations trop fréquentes.

Exemple concret

Pour un site d'e-commerce dont la charge varie fortement, il peut être contre-productif de dimensionner un cluster pour le pic de fin d'année. En optant pour un ASG, on maintient une base de deux instances en permanence (une dans chaque zone de disponibilité), puis, si le CPU dépasse 70 % sur une durée de quelques minutes, l'ASG lance une instance supplémentaire (jusqu'à un plafond fixé). Quand la charge retombe, l'ASG retire des instances pour réduire les coûts.



Infrastructure AWS tirant parti de l'Autoscaling

Ce schéma illustre le fonctionnement combiné d'un load balancer et d'un auto scaling group dans un environnement AWS.

Le trafic provenant d'Internet entre tout d'abord via une Internet Gateway, avant d'être redirigé vers un Application Load Balancer. Ce dernier agit comme un répartiteur intelligent : il examine les requêtes entrantes et les distribue de manière équilibrée entre plusieurs instances EC2 situées dans différentes zones de disponibilité (a, b, c). Cela permet non seulement d'optimiser les performances en évitant qu'une seule instance ne soit surchargée, mais aussi d'assurer une haute disponibilité en cas de défaillance d'une zone.

Les instances EC2 ne sont pas déployées statiquement : elles sont gérées dynamiquement par un groupe de mise à l'échelle automatique (Autoscaling group). Ce mécanisme permet d'ajuster le nombre d'instances en fonction de la charge : lorsque le trafic augmente, de nouvelles instances sont automatiquement lancées pour absorber la montée en charge ; inversement, lorsqu'il diminue, les instances inutiles sont arrêtées afin de limiter les coûts.

L'ensemble est encapsulé dans un *Virtual Private Cloud* (VPC), qui définit l'environnement réseau sécurisé dans lequel se déploient les ressources. Chaque instance EC2 appartient également à un groupe de sécurité qui contrôle les flux autorisés, ajoutant ainsi une couche de sécurité réseau à l'architecture.

Ce type d'architecture combine donc élasticité, performance et résilience, en s'adaptant en temps réel aux besoins des utilisateurs tout en garantissant une continuité de service.

4. Conteneurs : choisir entre ECS, EKS et Fargate

L'essor des conteneurs

Docker a radicalement changé la façon de déployer des applications. En regroupant code, dépendances et configuration dans une image, on garantit un déploiement identique sur tous les environnements Docker. Cela simplifie le cycle de développement et encourage la portabilité.