Architectures multiniveaux sans serveur AWS

Utilisation d'Amazon API Gateway et d'AWS Lambda

Novembre 2015

Mentions légales

Ce document est fourni à titre informatif uniquement. Il présente l'offre de produits et les pratiques actuelles d'AWS à la date de publication de ce document, des informations qui sont susceptibles d'être modifiées sans avis préalable. Il incombe aux clients de procéder à leur propre évaluation indépendante des informations contenues dans ce document et chaque client est responsable de son utilisation des produits ou services AWS, chacun étant fourni « en l'état », sans garantie d'aucune sorte, qu'elle soit explicite ou implicite. Ce document ne crée pas de garanties, représentations, engagements contractuels, conditions ou assurances à l'encontre d'AWS, de ses affiliés, fournisseurs ou donneurs de licence. Les responsabilités et obligations d'AWS vis-à-vis de ses clients sont régies par les contrats AWS. Le présent document ne fait partie d'aucun et ne modifie aucun contrat entre AWS et ses clients.

Table des matières

Résumé

Le présent livre blanc vous montre comment les innovations d'Amazon Web Services (AWS) peuvent modifier votre façon de concevoir les architectures multiniveaux de modèles répandus, tels que les microservices, les applications mobiles et les sites web publics. Les architectes et les développeurs peuvent désormais utiliser un modèle d'implémentation incluant Amazon API Gateway et AWS Lambda afin de réduire les cycles de développement et d'opérations requis pour créer et gérer efficacement les applications multiniveaux.

Introduction

L'application multiniveau (à trois niveaux, à n niveaux, etc.) a constitué un modèle d'architecture indépassable pendant des décennies. Ce modèle propose d'excellentes recommandations à suivre pour garantir des composants d'application découplés et évolutifs, à même d'être gérés et maintenus de façon séparée (souvent par des équipes distinctes). Les applications multiniveaux sont souvent conçues à l'aide d'une approche de type architecture orientée services en matière d'utilisation des services Web. Dans cette approche, le réseau fait office de frontière entre les niveaux. Cependant la création d'un service Web comme partie intégrante de votre application présente de nombreux aspects indifférenciés. Une grande part du code écrit au sein d'une application web multiniveau est un résultat direct du modèle lui-même. Parmi les exemples, citons le code intégrant un niveau à un autre, le code définissant une API et un modèle de données que les niveaux utilisent pour se comprendre, et le code relatif à la sécurité garantissant que les points d'intégration des niveaux ne sont pas exposés de manière non souhaitée.

Amazon API Gateway[1], service de création et de gestion des API, et AWS Lambda[2], service d'exécution de fonctions de code arbitraire, peuvent être utilisés conjointement pour simplifier la création d'applications multiniveaux robustes.

L'intégration d'Amazon API Gateway à AWS Lambda permet que les fonctions de code définies par l'utilisateur soient déclenchées directement via une demande HTTPS définie par l'utilisateur. Quel que soit le volume de demandes requis, API Gateway et Lambda se redimensionnent automatiquement pour prendre en charge les besoins exacts de votre application. Lorsque les deux services sont associés, vous pouvez créer un niveau pour votre application qui vous permet d'écrire le code important de votre application et de *ne pas* vous focaliser sur d'autres aspects sans différenciation de l'implémentation d'une architecture multiniveau, comme l'architecture destinée à une haute disponibilité, l'écriture de SDK côté client, la gestion du serveur/système d'exploitation, ou le dimensionnement et l'implémentation d'un dispositif d'autorisation client.

Plus récemment, AWS a annoncé la possibilité de créer des fonctions Lambda s'exécutant au sein de votre Amazon Virtual Private Cloud (Amazon VPC)[3]. Une telle fonctionnalité étend les avantages liés à l'association d'API Gateway et Lambda pour inclure une grande diversité de cas d'utilisation où la confidentialité du réseau est requise. Par exemple, lorsque vous devez intégrer votre service Web à une base de données relationnelle contenant des informations sensibles. L'intégration de Lambda et d'Amazon VPC a étendu indirectement les capacités d'Amazon API Gateway, car elle offre la possibilité aux développeurs de définir leur propre ensemble d'API HTTPS accessibles par Internet devant un serveur principal qui demeure privé et sécurisé à titre de partie intégrante d'Amazon VPC. Vous pouvez observer les avantages de ce puissant modèle à travers chaque couche d'une architecture multiniveau. Ce livre blanc se concentre sur l'exemple le plus connu d'architecture multiniveau, à savoir l'application web à **trois niveaux**. Cependant, vous pouvez appliquer ce modèle multiniveau au-delà d'une application web classique à trois niveaux.

Présentation de l'architecture à trois niveaux

L'architecture à trois niveaux est un modèle répandu pour les applications auxquelles sont confrontés les utilisateurs. Les niveaux composant cette architecture incluent le **niveau présentation**, le **niveau logique** et le **niveau données**. Le niveau présentation correspond au composant avec lequel les utilisateurs interagissent directement (comme une page web, une interface utilisateur d'application mobile, etc.). Le niveau logique contient le code requis pour traduire les actions utilisateur du niveau présentation en fonctionnalités qui orientent le comportement de l'application. Le niveau données se compose des supports de stockage (bases de données, magasins d'objets, caches, systèmes de fichiers, etc.) contenant les données appropriées de l'application. La figure 1 illustre un exemple d'application simple à trois niveaux.

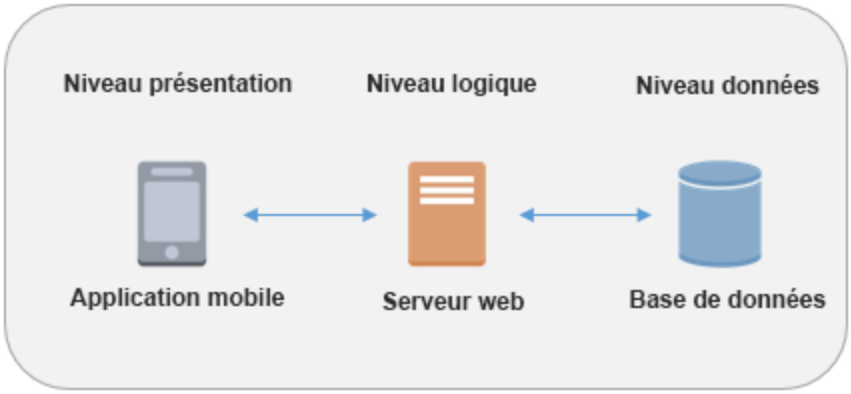

Figure 1 : modèle d'architecture pour une application simple à trois niveaux

Il existe en ligne de multiples ressources excellentes grâce auxquelles vous pouvez en savoir plus sur le *modèle général de l'*architecture à trois niveaux. Le livre blanc est axé sur le modèle d'implémentation spécifique de cette architecture à l'aide d'Amazon API Gateway et d'AWS Lambda.

Niveau logique sans serveur

Le niveau logique de l'architecture à trois niveaux constitue le cerveau de l'application. Telle est la raison pour laquelle l'intégration d'Amazon API Gateway et d'AWS Lambda pour former votre niveau logique peut être si révolutionnaire. Les fonctionnalités des deux services vous permettent de développer une application de production sans serveur hautement disponible, évolutive et sécurisée. Même si votre application utilise des milliers de serveurs, en mettant à profit ce modèle, vous n'avez pas à en gérer un seul. De plus, en utilisant ensemble ces services gérés, vous bénéficiez des avantages suivants :

- Aucun système d'exploitation à choisir, sécuriser, corriger ou gérer.
- Aucun serveur à dimensionner correctement, à surveiller ou à agrandir.
- Aucun risque de coût par sur-approvisionnement.
- Aucun risque de performance par sous-approvisionnement.

De plus, il existe des fonctions spécifiques au sein de chaque service qui sont bénéfiques au modèle d'architecture multi-niveau.

Amazon API Gateway

Amazon API Gateway est un service entièrement géré de définition, déploiement et maintenance des API. Les clients utilisent les API à l'aide de demandes HTTPS standard. Son applicabilité à une architecture multiniveau orienté services est manifeste. Cependant, le service possède des fonctionnalités et des qualités spécifiques qui en font un puissant avantage pour votre niveau logique.

Intégration à AWS Lambda

Amazon API Gateway offre à votre application un moyen simple (demandes HTTPS) de mettre directement à profit le caractère novateur d'AWS Lambda. API Gateway constitue le pont qui connecte votre niveau présentation et les fonctions que vous écrivez dans AWS Lambda. Après avoir défini la relation client/serveur à l'aide de votre API, le contenu de la demande HTTPS du client est transmis à Lambda pour être exécuté. Le contenu inclut les métadonnées, les en-têtes et le corps de la demande.

Performances des API stables à travers le monde

Chaque déploiement d'Amazon API Gateway inclut une distribution d'Amazon CloudFront[4]. Amazon CloudFront est un service Web de diffusion de contenu qui utilise le réseau mondial Amazon des emplacements périphériques (edge locations) comme points de connexion pour les clients qui recourent à votre API. Il est ainsi possible de réduire la latence totale du temps de réponse de votre API. Au travers de l'utilisation de plusieurs emplacements périphériques à travers le monde, Amazon CloudFront met aussi à votre disposition les capacités nécessaires pour combattre les scénarios d'attaques par déni de service distribué (DDoS). Pour plus d'informations, consultez le livre blanc AWS Best Practices for Combatting DDoS Attacks (Bonnes pratiques AWS pour combattre les attaques DDoS)[5].

Vous pouvez améliorer les performances de demandes d'API spécifiques en utilisant Amazon API Gateway pour stocker les réponses dans un cache en mémoire facultatif. Il en résulte non seulement des avantages en matière de performances pour les demandes d'API répétées, mais aussi une réduction des exécutions du serveur principal, ce qui peut diminuer votre coût global.

Encourager l'innovation

Le travail de développement requis pour créer une application constitue un investissement. Vous devez le justifier pour que le projet commence. En réduisant la quantité d'investissement requise pour les tâches de développement, ainsi que le temps nécessaire, vous êtes libre de procéder à plus d'expériences et d'innover plus ouvertement.

Pour de nombreuses applications multiniveaux basées sur les services Web, le niveau présentation est facilement réparti entre les utilisateurs (périphériques mobiles séparés, navigateurs web, etc.). Souvent, ces utilisateurs ne sont pas liés géographiquement. Cependant, un niveau logique découplé n'est pas fragmenté physiquement par les utilisateurs. Tous les utilisateurs dépendent de la même infrastructure pour l'exécution de votre niveau logique, ce qui amplifie l'importance de l'infrastructure. Faire des économies lors de l'implémentation initiale de votre niveau logique (« nous n'avons pas besoin de métriques au lancement initial », « l'utilisation sera faible au début, nous nous préoccuperons de dimensionner ultérieurement », etc.) est souvent proposé comme solution pour diffuser une nouvelle application plus rapidement. Il peut en résulter une dette technique et un risque opérationnel quand vous devez déployer ces modifications sur une application s'exécutant déjà en production. Amazon API Gateway *vous* permet de réaliser de telles économies tout en diffusant plus rapidement, parce que le service les a déjà implémentées automatiquement.

La durée de vie globale d'une application pourrait être inconnue ou connue comme étant brève. La création d'un script commercial pour une application multiniveau peut être difficile pour ces raisons-là. Elle peut être rendue plus simple quand votre point de démarrage inclut les fonctionnalités gérées fournies par Amazon API Gateway et que vous commencez seulement à être exposé aux coûts d'infrastructure après que votre API a démarré la réception des demandes. Pour plus d'informations, consultez Amazon API Gateway Pricing (Tarification d'Amazon API Gateway).[6]

Itérer rapidement, demeurer agile

Avec les nouvelles applications, la base d'utilisateur peut continuer à être mal définie (taille, modèles d'utilisation, etc.). Le niveau logique doit demeurer agile tandis que la base d'utilisateurs prend forme. Votre application et votre activité doivent pouvoir évoluer et accueillir les attentes changeantes de vos premiers clients. Amazon API Gateway réduit le nombre de cycles de développement requis pour conduire une API de la création au déploiement. Amazon API Gateway permet de créer des intégrations fictives[7] grâce auxquelles vous pouvez générer les réponses des API directement depuis API Gateway et par rapport auxquelles les applications clientes peuvent se développer, tandis qu'en parallèle, la logique principale complète est développée. Cet avantage s'applique non seulement lors du premier déploiement d'une API, mais aussi après que l'entreprise a décidé que l'application (et l'API existante) doit évoluer rapidement en réponse à vos utilisateurs. Comme API Gateway et AWS Lambda autorisent le versioning, les fonctionnalités existantes et les dépendances des clients peuvent continuer en l'état, alors que de nouvelles fonctionnalités sont diffusées sous forme de version d'API/de fonction distincte.

Sécurité

L'implémentation du niveau logique d'une application web publique à trois niveaux en tant que service Web promeut immédiatement le thème de la sécurité. L'application doit s'assurer que seuls les clients autorisés ont accès à votre niveau logique (qui est exposé sur le réseau). Amazon API Gateway traite le thème de la sécurité au travers de solutions qui vous garantissent la sécurité de votre serveur principal. Pour le contrôle d'accès, ne vous fiez pas à la fourniture à vos applications clientes de chaînes de clé API statique ; elles peuvent être extraites des clients et utilisées ailleurs. Vous pouvez tirer profit des multiples façons par lesquelles Amazon API Gateway contribue à sécuriser votre niveau logique :

- Toutes les demandes adressées à vos API peuvent s'effectuer via HTTPS pour autoriser le chiffrement en transit.
- Vos fonctions AWS Lambda peuvent restreindre l'accès de telle sorte qu'il n'existe une relation d'approbation qu'entre une API particulière au sein d'Amazon API Gateway et une fonction particulière d'AWS Lambda. Il n'y a aucun autre moyen d'invoquer cette fonction Lambda, si ce n'est en utilisant l'API via laquelle vous avez choisi de l'exposer.

- Amazon API Gateway vous permet de générer les SDK client pour les intégrer à vos API. Ce SDK gère aussi la signature des demandes lorsque les API nécessitent une authentification. Ces informations d'identification des API utilisées sur le côté client pour l'authentification sont transmises directement à votre fonction AWS Lambda, où une authentification supplémentaire peut se produire au sein du code que vous possédez et écrivez, si nécessaire.
- Chaque combinaison ressource/méthode que vous créez comme partie intégrante de votre API se voit attribuer son propre ARN (Amazon Resource Name) qui peut être référencé dans les stratégies AWS Identity and Access Management (IAM)[8].
  - Cela signifie que vos API sont traitées comme citoyens de première classe, au même titre que les autres API que possède AWS. Les stratégies IAM peuvent être détaillées et référencer les ressources/méthodes spécifiques d'une API créée à l'aide d'Amazon API Gateway.
  - L'accès des API est appliqué par les stratégies IAM que vous créez en dehors du contexte de votre code d'application. Il s'ensuit que vous n'avez à écrire aucun code pour être conscient de ces niveaux d'accès ou les mettre en vigueur. Le code ne peut pas contenir de bogues ou être exploité s'il n'existe pas.
  - L'autorisation des clients à l'aide de l'autorisation AWS Signature version 4 (SigV4)[9] et des stratégies IAM pour l'accès des API permet à ces mêmes informations d'identification de restreindre l'accès ou de l'autoriser à d'autres ressources et services AWS comme nécessaire (par exemple, les compartiments Amazon S3 ou les tables Amazon DynamoDB).

AWS Lambda

En son cœur, AWS Lambda autorise que le code arbitraire écrit en l'un quelconque des langages pris en charge (Node, basé sur JVM et Python à compter de novembre 2015) soit déclenché en réponse à un événement. Cet événement peut être l'un des multiples déclencheurs programmatiques qu'AWS rend disponibles, appelés **source de l'événement** (voir ici les sources d'événement actuellement prises en charge[10]). Les cas d'utilisation les plus répandus d'AWS Lambda tournent autour des flux de travail de traitement des données pilotés par les événements, comme le traitement des fichiers stockés dans Amazon Simple Storage Service (Amazon S3)[11] ou la diffusion d'enregistrements de données depuis Amazon Kinesis[12].

Lors d'une utilisation conjointe avec Amazon API Gateway, une fonction AWS Lambda peut exister au sein du contexte d'un service Web classique, et elle peut être déclenchée directement par une demande HTTPS. Amazon API Gateway fait office de porte d'entrée de votre niveau logique, mais vous devez désormais exécuter la logique à l'arrière de ces API. C'est ici qu'AWS Lambda entre en scène.

Emplacement de votre logique métier

AWS Lambda vous permet d'écrire des fonctions de code, appelées **gestionnaires**, qui s'exécuteront lorsqu'elles sont déclenchées par un événement. Par exemple, vous pouvez écrire un gestionnaire qui se déclenchera quand un événement tel qu'une demande HTTPS à votre API se produit. Lambda vous permet de créer des gestionnaires modulaires au niveau de précision de votre choix (un par API ou un par méthode d'API) qui peuvent être mis à jour, appelée et modifiés de façon indépendante. Le gestionnaire est alors libre d'accéder à toute autre de ses dépendances (comme les autres fonctions que vous avez chargées avec votre code, les bibliothèques, les bibliothèques natives ou même les services Web externes). Lambda vous permet d'empaqueter la totalité des dépendances requises dans votre définition de fonction lors de la création. Lorsque vous créez votre fonction, vous spécifiez la méthode à l'intérieur de votre package de déploiement qui fera office de gestionnaire de demandes. Vous êtes libre de réutiliser le même package de déploiement pour plusieurs définitions de fonction Lambda, chaque fonction Lambda pouvant avoir un seul gestionnaire au sein du même package de déploiement. Dans le modèle d'architecture multiniveau sans serveur, chacune des API que vous créez dans Amazon API Gateway s'intègre à une fonction Lambda (et à son gestionnaire) qui exécute la logique business requise.

Intégration d'Amazon VPC

AWS Lambda, le cœur de votre niveau logique, est le composant qui s'intègre directement au niveau données. Comme le niveau données contient souvent des informations métier ou utilisateur, il doit être fermement sécurisé. Dans le cas des services AWS grâce auxquels vous pouvez intégrer à partir d'une fonction Lambda, vous pouvez gérer le contrôle d'accès à l'aide des stratégies IAM. Ces services incluent, entre autres, Amazon S3, Amazon DynamoDB, Amazon Kinesis, Amazon Simple Queue Service (Amazon SQS) et Amazon Simple Notification Service (Amazon SNS), ainsi que d'autres fonctions AWS Lambda. Cependant, il se peut que vous ayez un composant qui régisse son propre contrôle d'accès, comme une base de données relationnelle. Avec de tels composants, vous pouvez parvenir à une meilleure sécurité en les déployant au sein d'un environnement de mise en réseau privé, à savoir un Amazon Virtual Private Cloud (Amazon VPC)[13].

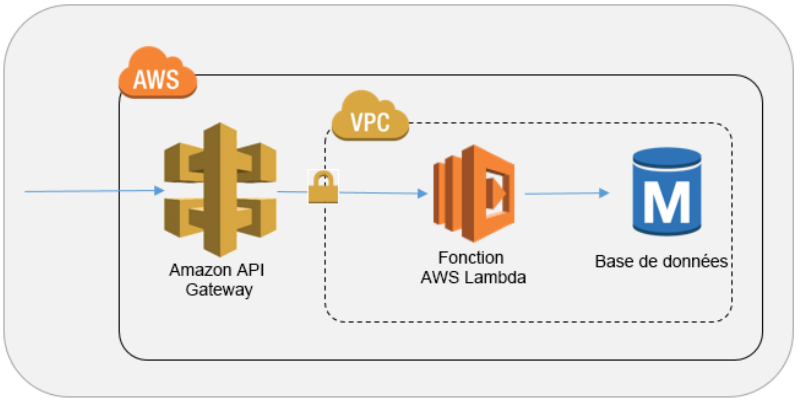

Figure 2 : modèle d'architecture à l'aide d'un VPC

L'utilisation d'un VPC signifie que les bases de données et autres supports de stockage dont dépend votre logique business peuvent devenir inaccessibles sur Internet. Le VPC assure aussi que la *seule* façon d'interagir avec vos données depuis Internet se fera au travers des API que vous avez définies et des fonctions de code Lambda que vous avez écrites.

Sécurité

Pour qu'une fonction Lambda soit exécutée, elle doit être déclenchée par un événement ou un service ayant été autorisé à agir ainsi via une stratégie IAM. Il est possible de créer une fonction Lambda qui ne puisse pas être *du tout* exécutée, à moins qu'elle ne soit invoquée par une demande API Gateway que vous définissez. Votre code ne sera traité que comme partie de votre cas d'utilisation valide, défini par l'API que vous avez créée.

Chaque fonction Lambda assume elle-même un rôle IAM, possibilité qui doit être accordée via une relation d'approbation IAM. Ce rôle IAM définit les autres services/ressources AWS avec lesquels votre fonction Lambda pourra interagir (comme une table Amazon DynamoDB ou un compartiment Amazon S3). Les services auxquels votre fonction a accès seront définis et contrôlés depuis l'extérieur de la fonction elle-même. Ce comportement est subtil, mais puissant. Il permet au code que vous écrivez d'être libre de stocker ou d'extraire les informations d'identification AWS : cela signifie que vous n'avez pas à coder en dur les clés d'API, ni à écrire le code pour les extraire et les stocker en mémoire. L'activation de votre fonction Lambda pour appeler les services auxquels elle a droit, comme défini par son rôle IAM, est géré pour vous par le service lui-même.

Niveau données

En utilisant AWS Lambda comme niveau logique, vous disposez d'un large éventail d'options de stockage de données pour votre niveau données. Ces options se divisent en deux grandes catégories : les magasins de données hébergés par Amazon VPC et les magasins de données IAM. AWS Lambda a la possibilité de s'intégrer aux deux en toute sécurité.

Magasins de données hébergés sur Amazon VPC

L'intégration d'AWS Lambda à Amazon VPC permet aux fonctions d'intégrer une grande diversité de technologies de stockage de manière privée et sécurisée.

- Amazon RDS[14]

  Utilisez l'un des moteurs rendus disponibles par Amazon Relational Database Service (Amazon RDS). Connectez-vous directement au code que vous avez écrit dans Lambda, tout comme vous le feriez en dehors de Lambda, mais avec l'avantage d'une simple intégration à AWS Key Management Service (AWS KMS) pour le chiffrement des informations d'identification de la base de données.

- Amazon ElastiCache[15]

  Intégrez vos fonctions Lambda à un cache en mémoire géré pour optimiser les performances de votre application.

- Amazon Redshift[16]

  Vous pouvez développer des fonctions qui interrogent en toute sécurité l'entrepôt de données d'une entreprise à des fins de génération de rapports, de tableaux de bord ou d'extraction de résultats de requête appropriés.

- Service Web privé hébergé par Amazon Elastic Compute Cloud (Amazon EC2)[17]

  Il se peut que vous ayez des applications existantes qui s'exécutent comme service Web de façon privée au sein d'un VPC. Effectuez vos demandes HTTP sur votre réseau VPC logiquement privé à partir d'une fonction Lambda.

Magasins de données IAM

Comme AWS Lambda est intégré à IAM, il peut utiliser IAM pour sécuriser l'intégration à n'importe quel service AWS pouvant être exploité directement à l'aide des API AWS.

- Amazon DynamoDB[18]

  Amazon DynamoDB est la base de données NoSQL évolutive de façon illimitée d'AWS. Pensez à Amazon DynamoDB lorsque vous voulez extraire des enregistrements de données (400 Ko ou moins à la date de rédaction du livre blanc) avec des performances inférieures à 10 millisecondes, indépendamment de l'échelle. Grâce au contrôle d'accès précis d'Amazon DynamoDB, vos fonctions Lambda peuvent suivre les bonnes pratiques du moindre privilège lors de l'interrogation de données spécifiques dans DynamoDB.

- Amazon S3[19]

  Amazon Simple Storage Service (Amazon S3) fournit un stockage d'objets à l'échelle d'Internet. Comme Amazon S3 est conçu pour une durabilité de 99,999999999 % des objets, pensez à l'utiliser quand votre application nécessite un stockage bon marché et hautement durable. De plus, comme Amazon S3 est conçu pour une disponibilité des objets pouvant atteindre jusqu'à 99,99 % sur une année donnée, envisagez de l'utiliser quand votre application requiert un stockage hautement disponible. Il est possible d'accéder directement aux objets stockés dans Amazon S3 (fichiers, images, journaux, données binaires) via HTTP. Les fonctions Lambda peuvent communiquer en toute sécurité avec Amazon S3 via les points de terminaison privés virtuels, tandis que les données au sein de S3 peuvent être restreintes à la seule stratégie IAM associée à la fonction Lambda.

- Amazon Elasticsearch Service[20]

  Amazon Elasticsearch Service (Amazon ES) est une version managée d'Elasticsearch, célèbre moteur de recherche et d'analyse. Amazon ES fournit l'allocation managée des clusters, la détection des défaillances et le remplacement des nœuds ; vous pouvez l'imiter l'accès à l'API Amazon ES à l'aide des stratégies IAM.

Niveau présentation

Amazon API Gateway ouvre une grande diversité de possibilités au niveau présentation. Une API HTTPS accessible par Internet peut être utilisée par tout client capable d'une communication HTTPS. La liste suivante contient quelques exemples que vous pouvez envisager d'utiliser pour le niveau présentation de votre application :

- Application mobile : en plus de l'intégration à la logique métier personnalisée via Amazon API Gateway et AWS Lambda, vous pouvez utiliser Amazon Cognito[21] comme mécanisme pour créer et gérer les identités utilisateur.
- Contenu de site Web statique (tel que les fichiers hébergés dans Amazon S3) : vous pouvez activer vos API Amazon API Gateway pour qu'elles soient compatibles CORS (Cross-Origin Resource Sharing). Les navigateurs web peuvent ainsi appeler directement vos API depuis les pages web statiques.
- Autre périphérique client HTTPS : la plupart des appareils connectés sont capables de communiquer via HTTPS. Il n'y a rien d'unique ou de propriétaire quant à la façon dont les clients communiquent avec les API que vous créez à l'aide d'Amazon API Gateway ; c'est du HTTPS à l'état pur. Ni logiciel client ou licence spécifique n'est requis.

Exemples de modèles d'architecture

Vous pouvez implémenter les célèbres modèles d'architecture suivants à l'aide d'Amazon API Gateway et d'AWS Lambda comme l'assemblage qui compose votre niveau logique. Pour chaque exemple, nous utilisons uniquement les services AWS qui ne nécessitent pas que les utilisateurs gèrent leur propre infrastructure.

Application mobile

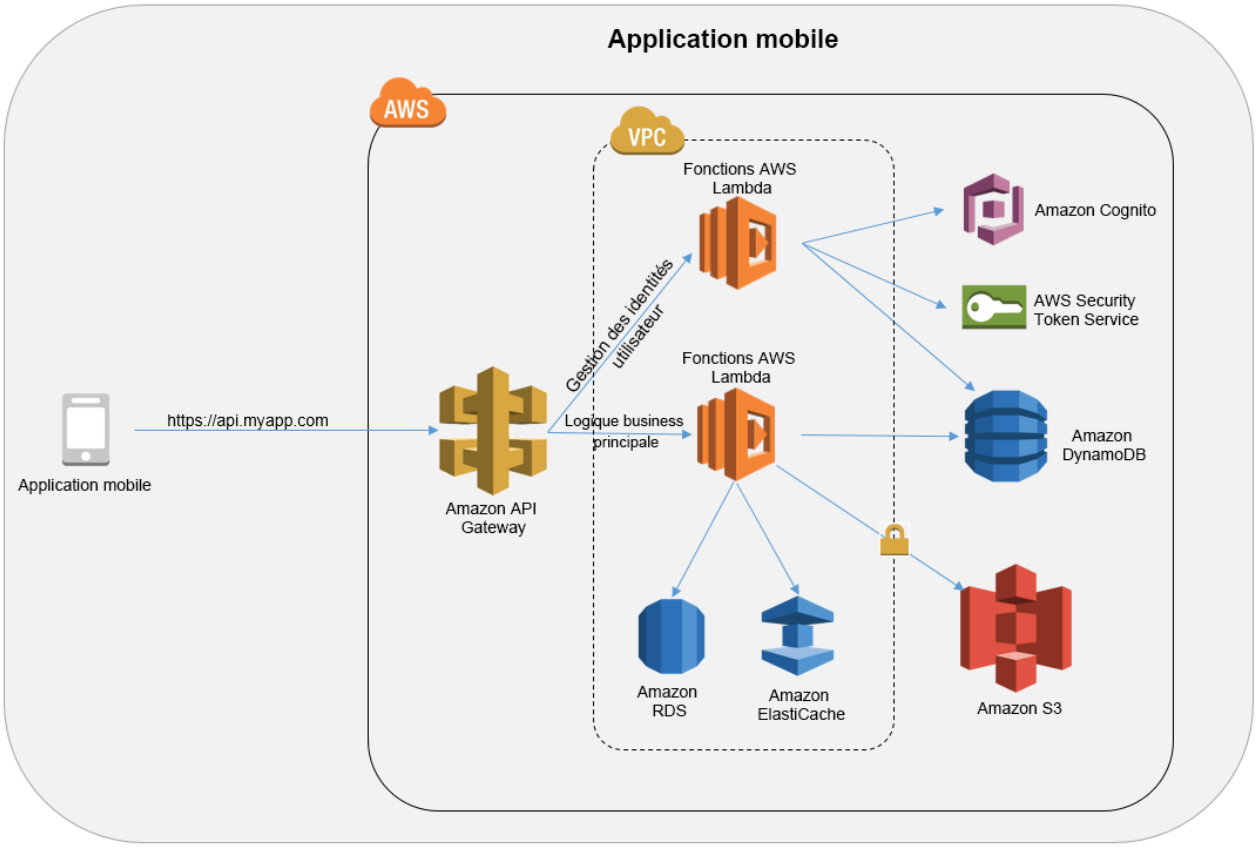

Figure 3 : modèle d'architecture pour application mobile

- **Niveau présentation :** application mobile s'exécutant sur le smartphone de chaque utilisateur.
- **Niveau logique :** Amazon API Gateway et AWS Lambda. Le niveau logique est globalement réparti par la distribution Amazon CloudFront créée comme partie intégrante de chaque API Amazon API Gateway. Un ensemble de fonctions Lambda peut être spécifique à l'authentification et à la gestion des identités utilisateur/périphérique, et géré par Amazon Cognito, qui fournit l'intégration à IAM pour les informations d'identification d'accès utilisateur temporaires, ainsi qu'aux fournisseurs d'identité tiers les plus connus. D'autres fonctions Lambda peuvent définir la logique business principale de votre application mobile.

- **Niveau données :** les différents services de stockage peuvent être mis à profit selon vos besoins ; les options sont présentées plus haut dans le livre blanc.

Site web hébergé sur Amazon S3

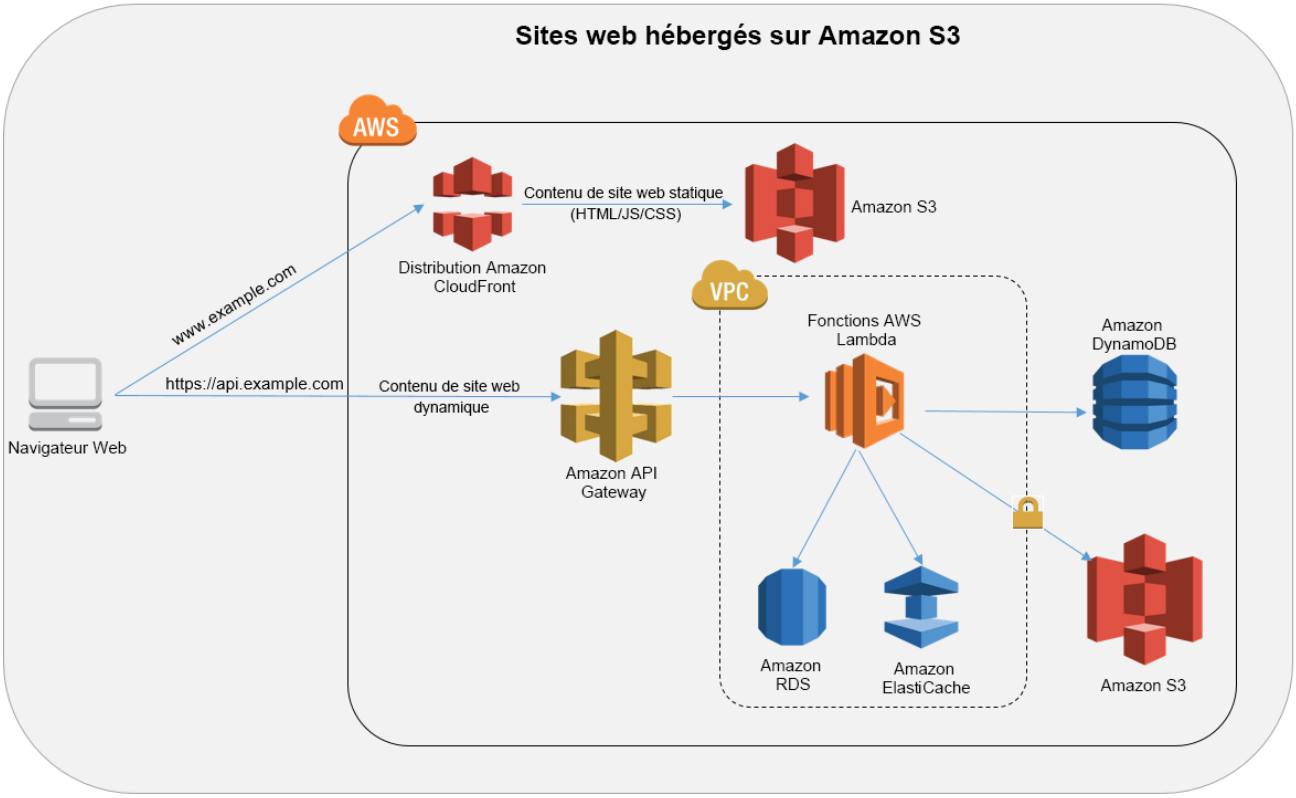

Figure 4 : modèle d'architecture pour une application web statique hébergée sur Amazon S3

- **Niveau présentation :** contenu de site web statique hébergé dans Amazon S3, distribué par Amazon CloudFront. L'hébergement d'un contenu de site web statique sur Amazon S3 est une alternative économique à l'hébergement de contenu sur une infrastructure serveur. Cependant, dans le cas d'un site web qui contient de riches fonctionnalités, le contenu statique doit souvent s'intégrer à un backend dynamique.
- **Niveau logique :** Amazon API Gateway et AWS Lambda. Le contenu d'un site web statique hébergé dans Amazon S3 peut s'intégrer directement à Amazon API Gateway, qui peut être compatible CORS.

- **Niveau données :** les différents services de stockage de données peuvent être mis à profit selon les besoins. Ces options sont présentées plus haut dans le livre blanc.

Environnement de microservices

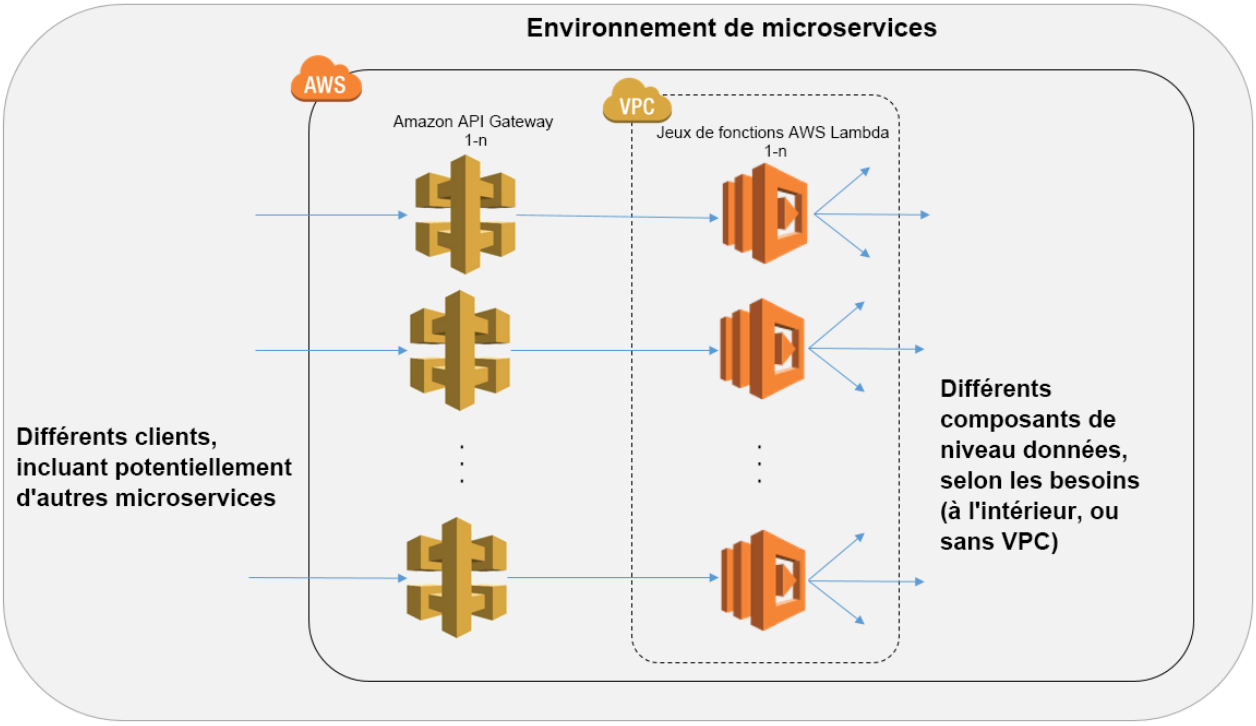

Figure 5 : modèle d'architecture pour un environnement de microservices

Le modèle d'architecture de **microservices** n'est pas lié à l'architecture classique à trois niveaux que nous avons abordée dans ce livre blanc. Dans une architecture de microservices, il existe un découplage important des composants logiciels, si bien que les avantages de l'architecture multiniveau sont amplifiés d'un bout à l'autre. Une API créée avec Amazon API Gateway, ainsi que les fonctions exécutées ensuite par AWS Lambda, est tout ce qu'il vous faut pour créer un microservice. Votre équipe est libre d'utiliser ces services pour découpler et fragmenter votre environnement jusqu'au niveau de granularité souhaité.

En général, un environnement de microservices peut présenter les difficultés suivantes : surcharge répétée pour la création de chaque microservice, problèmes liés à l'optimisation de l'utilisation et de la densité du serveur, complexité de l'exécution simultanée de plusieurs versions de différents microservices et prolifération des exigences du code côté client à intégrer à différents appareils distincts.

Cependant, lorsque vous créez des microservices à l'aide du modèle sans serveur AWS, ces problèmes deviennent plus simples à résoudre et, dans certains cas, disparaître tout simplement. Le modèle de microservices AWS atténue l'obstacle lié à la création de chaque microservice suivant (Amazon API Gateway autorise même le clonage d'API existantes). L'optimisation de l'utilisation de ce serveur n'est plus pertinente dans le cas de ce modèle. API Gateway et Lambda autorisent tous deux des fonctionnalités de versioning simples. Enfin, Amazon API Gateway fournit des SDK client générés par programmation dans un certain nombre de langages répandus afin de réduire la surcharge associée à l'intégration.

Conclusion

Le modèle de l'architecture multiniveau encourage la bonne pratique consistant à créer des composants d'application faciles à maintenir, découplés et évolutifs. Lorsque vous créez un niveau logique où l'intégration se produit via Amazon API Gateway et où le calcul s'effectue dans AWS Lambda, vous êtes en bonne voie pour atteindre ces objectifs, tout en réduisant l'effort requis pour y parvenir. Ensemble, ces services fournissent un serveur frontal d'API HTTPS à vos clients et un environnement sécurisé au sein de votre VPC pour exécuter la logique business. Vous pouvez ainsi tirer parti de nombreux scénarios populaires dans lesquels vous pouvez utiliser ces services managés au lieu de gérer vous-même l'infrastructure serveur classique.

Collaborateurs

Les personnes et organisations suivantes ont participé à l'élaboration de ce document :

Andrew Baird, architecte solutions AWS

Stefano Buliani, responsable produit senior, Tech, AWS Mobile

Vyom Nagrani, responsable produit senior, AWS Mobile

Ajay Nair, responsable produit senior, AWS Mobile

Notes

[1] http://aws.amazon.com/api-gateway/

[2] http://aws.amazon.com/lambda/

[3] https://aws.amazon.com/vpc/

[4] https://aws.amazon.com/cloudfront/

[5] https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

[6] https://aws.amazon.com/api-gateway/pricing/

[7] http://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html

[8] http://aws.amazon.com/iam/

[9] http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

[10] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources

[11] https://aws.amazon.com/s3/

[12] https://aws.amazon.com/kinesis/

[13] https://aws.amazon.com/vpc/

[14] https://aws.amazon.com/rds/

[15] https://aws.amazon.com/elasticache/

[16] https://aws.amazon.com/redshift/

[17] https://aws.amazon.com/ec2/

[18] https://aws.amazon.com/dynamodb/

[19] https://aws.amazon.com/s3/storage-classes/

[20] https://aws.amazon.com/elasticsearch-service/

[21] https://aws.amazon.com/cognito/