Livre blanc AWS

Architectures à plusieurs niveaux sans serveur AWS avec Amazon API Gateway et AWS Lambda

Architectures à plusieurs niveaux sans serveur AWS avec Amazon API Gateway et AWS Lambda : Livre blanc AWS

Table of Contents

Architectures à plusieurs niveaux sans serveur AWS avec Amazon API Gateway et AWS Lambda

Date de publication : 20 octobre 2021 (Révisions du document)

Résumé

Ce livre blanc montre comment les innovations d'Amazon Web Services (AWS) peuvent aider à transformer la conception des architectures à plusieurs niveaux et la mise en œuvre des modèles courants, tels que les microservices, les backends mobiles et les applications d'une seule page. Les architectes et les développeurs peuvent utiliser Amazon API Gateway, AWS Lambda et d'autres services pour réduire les cycles de développement et d'exploitation nécessaires à la création et à la gestion d'applications à plusieurs niveaux.

Introduction

L'application à plusieurs niveaux (trois niveaux, multi-niveaux, etc.) est un modèle d'architecture fondamental depuis des décennies et reste un modèle populaire pour les applications destinées aux utilisateurs. Bien que le langage utilisé pour décrire une architecture à plusieurs niveaux varie, une application à plusieurs niveaux comprend généralement les composants suivants :

- Niveau de présentation : composant avec lequel l'utilisateur interagit directement (par exemple, les pages web et les interfaces utilisateur des applications mobiles).
- Niveau logique : code requis pour traduire les actions des utilisateurs en fonctionnalités de l'application (par exemple, les opérations de base de données CRUD et le traitement des données).
- Niveau de données : supports de stockage (par exemple, bases de données, magasins d'objets, caches et systèmes de fichiers) qui contiennent les données pertinentes pour l'application.

Le modèle d'architecture à plusieurs niveaux fournit un cadre général pour garantir que les composants d'application découplés et évolutifs de manière indépendante peuvent être développés, gérés et maintenus séparément (souvent par des équipes distinctes).

En raison de ce modèle dans lequel le réseau (un niveau doit effectuer un appel réseau pour interagir avec un autre niveau) agit comme la frontière entre les niveaux, le développement d'une application à plusieurs niveaux nécessite souvent la création de nombreux composants d'application indifférenciés. Certains de ces composants incluent :

- Le code définissant une file d'attente de messages pour la communication entre les niveaux
- Le code définissant une interface de programme d'application (API) et un modèle de données
- Le code lié à la sécurité qui garantit un accès approprié à l'application

Tous ces exemples peuvent être considérés comme des composants « standard » qui, bien que nécessaires dans des applications à plusieurs niveaux, ne varient pas beaucoup dans leur mise en œuvre d'une application à une autre.

AWS propose un certain nombre de services qui permettent la création d'applications à plusieurs niveaux sans serveur, simplifiant ainsi considérablement le processus de déploiement de ces applications en production et supprimant la surcharge associée à la gestion traditionnelle des serveurs. Amazon API Gateway, un service de création et de gestion d'API et AWS Lambda, un

service d'exécution de fonctions de code arbitraires, peuvent être utilisés ensemble pour simplifier la création d'applications à plusieurs niveaux robustes.

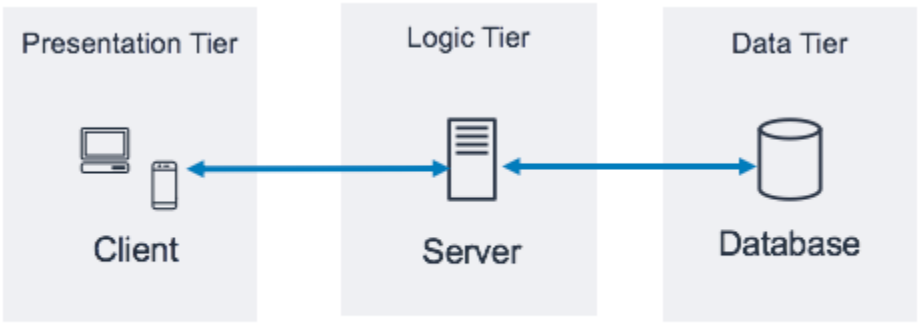
L'intégration d'Amazon API Gateway à AWS Lambda permet de lancer des fonctions de code définies par l'utilisateur directement par le biais de requêtes HTTPS. Quel que soit le volume de demandes, API Gateway et Lambda se mettent à l'échelle automatiquement pour prendre en charge exactement les besoins de votre application (veuillez consulter Quotas Amazon API Gateway et remarques importantes pour obtenir des informations sur la capacité de mise à l'échelle). En combinant ces deux services, vous pouvez créer un niveau qui vous permet d'écrire uniquement le code qui compte pour votre application et de ne pas vous concentrer sur divers autres aspects non différenciés de la mise en œuvre d'une architecture à plusieurs niveaux, tels que l'architecture pour la haute disponibilité, l'écriture de kits SDK clients, la gestion du serveur et du système d'exploitation (OS), la mise à l'échelle et la mise en œuvre d'un mécanisme d'autorisation du client.

API Gateway et Lambda permettent la création d'un niveau logique sans serveur. En fonction des exigences de votre application, AWS propose également des options pour créer un niveau de présentation sans serveur (par exemple, avec Amazon CloudFront et Amazon Simple Storage Service) et un niveau de données (par exemple, Amazon Aurora ou Amazon DynamoDB).

Ce livre blanc se concentre sur l'exemple le plus populaire d'architecture à plusieurs niveaux, l'application web à trois niveaux. Toutefois, vous pouvez appliquer ce modèle à plusieurs niveaux bien au-delà d'une application web classique à trois niveaux.

Présentation de l'architecture à trois niveaux

L'architecture à trois niveaux est la mise en œuvre la plus populaire d'une architecture à plusieurs niveaux et se compose d'un niveau de présentation unique, d'un niveau logique et d'un niveau de données. L'illustration suivante montre un exemple d'application simple et générique à trois niveaux.



Motif architectural pour une application à trois niveaux

Il existe de nombreuses ressources en ligne intéressantes pour en savoir plus sur le modèle général d'architecture à trois niveaux. Ce livre blanc se concentre sur un modèle de mise en œuvre spécifique pour cette architecture à l'aide d'Amazon API Gateway et AWS Lambda.

Niveau logique sans serveur

Le niveau logique de l'architecture à trois niveaux représente le cerveau de l'application. C'est là qu'utiliser Amazon API Gateway et AWS Lambda peut avoir le plus d'impact par rapport à une mise en œuvre traditionnelle basée sur un serveur. Les fonctions de ces deux services vous permettent de créer une application sans serveur hautement disponible, évolutive et sécurisée. Dans un modèle traditionnel, votre application peut nécessiter des milliers de serveurs ; toutefois, en utilisant Amazon API Gateway et AWS Lambda, vous n'êtes pas responsable de la gestion des serveurs à quelque titre que ce soit. En outre, en utilisant ces services gérés ensemble, vous bénéficiez des avantages suivants :

- AWS Lambda :
  - Aucun système d'exploitation à choisir, à sécuriser, à corriger ou à gérer
  - Aucun serveur à correctement dimensionner, contrôler ou mettre à l'échelle
  - Réduction des risques liés au surapprovisionnement pour vos coûts
  - Réduction des risques liés au sous-approvisionnement pour vos performances

- Amazon API Gateway :
  - Mécanismes simplifiés pour déployer, contrôler et sécuriser les API
  - Amélioration des performances de l'API grâce à la mise en cache et à la diffusion de contenu

AWS Lambda

AWS Lambda est un service de calcul qui vous permet d'exécuter des fonctions de code arbitraires dans l'un des langages pris en charge (Node.js, Python, Ruby, Java, Go, .NET, pour plus d'informations, veuillez consulter FAQ Lambda) sans allouer, gérer ni dimensionner des serveurs. Les fonctions Lambda sont exécutées dans un conteneur géré et isolé, et sont lancées en réponse à un événement qui peut être l'un des nombreux déclencheurs programmatiques qu'AWS met à disposition, appelé source d'événement. Veuillez consulter FAQ Lambda pour connaître toutes les sources d'événements.

De nombreux cas d'utilisation courants de Lambda concernent les flux de traitement des données orientées événements, tels que le traitement de fichiers stockés dans Amazon S3 ou les registres de données en streaming d'Amazon Kinesis. Lorsqu'elle est utilisée conjointement avec Amazon API Gateway, une fonction Lambda exécute les fonctionnalités d'un service web classique : elle initie du

code en réponse à une demande HTTPS du client ; API Gateway fait office de porte d'entrée pour votre niveau logique et AWS Lambda appelle le code d'application.

Votre logique métier se trouve ici, sans aucun serveur nécessaire

Lambda nécessite que vous écriviez des fonctions de code, appelées gestionnaires, qui s'exécuteront lorsqu'elles seront initiées par un événement. Pour utiliser Lambda avec API Gateway, vous pouvez configurer API Gateway pour lancer des fonctions de gestionnaire lorsqu'une requête HTTPS est envoyée à votre API. Dans une architecture à plusieurs niveaux sans serveur, chacune des API que vous créez dans API Gateway s'intègre à une fonction Lambda (et au gestionnaire qu'elle contient) qui invoque la logique métier requise.

Utiliser les fonctions AWS Lambda pour composer le niveau logique vous permet de définir le niveau de granularité souhaité pour exposer la fonctionnalité de l'application (une fonction Lambda par API ou une fonction Lambda par méthode d'API). À l'intérieur de la fonction Lambda, le gestionnaire peut atteindre toutes les autres dépendances (par exemple, d'autres méthodes que vous avez téléchargées avec votre code, des bibliothèques, des binaires natifs et des services web externes) ou même d'autres fonctions Lambda.

La création ou la mise à jour d'une fonction Lambda nécessite soit le chargement de code en tant que package de déploiement Lambda dans un fichier zip vers un compartiment Amazon S3, soit le code d'empaquetage sous forme d'image de conteneur avec toutes les dépendances. Les fonctions peuvent utiliser différentes méthodes de déploiement, telles que la console de gestion AWS, l'exécution d'AWS Command Line Interface (AWS CLI) ou l'exécution de modèles ou de cadres d'Infrastructure as Code, tels que AWS CloudFormation, AWS Serverless Application Model (AWS SAM) ou AWS Cloud Development Kit (AWS CDK). Lorsque vous créez votre fonction à l'aide de l'une de ces méthodes, vous spécifiez quelle méthode de votre package de déploiement fera office de gestionnaire de la demande. Vous pouvez réutiliser le même package de déploiement pour plusieurs définitions de fonction Lambda, chaque fonction Lambda pouvant avoir un gestionnaire unique au sein du même package de déploiement.

Sécurité Lambda

Pour exécuter une fonction Lambda, elle doit être appelée par un événement ou un service autorisé par une politique AWS Identity and Access Management (IAM). Avec les politiques IAM, vous pouvez créer une fonction Lambda qui ne peut être initiée que si elle est appelée par une ressource API Gateway que vous définissez. Cette politique peut être définie à l'aide d'une politique basée sur les ressources pour divers services AWS.

Chaque fonction Lambda assume un rôle IAM qui est attribué lors du déploiement de la fonction Lambda. Ce rôle IAM définit les autres services et ressources AWS avec lesquels votre fonction Lambda peut interagir (par exemple, Amazon DynamoDB Amazon S3). Dans le contexte de la fonction Lambda, cela s'appelle un rôle d'exécution.

Ne stockez pas d'informations sensibles dans une fonction Lambda. IAM gère l'accès aux services AWS via le rôle d'exécution Lambda ; si vous devez accéder à d'autres informations d'identification (par exemple, des informations d'identification de base de données et des clés d'API) à partir de votre fonction Lambda, vous pouvez utiliser AWS Key Management Service (AWS KMS) avec des variables d'environnement ou utiliser un service tel que AWS Secrets Manager pour protéger ces informations lorsqu'elles ne sont pas utilisées.

Performance à grande échelle

Le code extrait en tant qu'image de conteneur à partir d'Amazon Elastic Container Registry (Amazon ECR) ou d'un fichier zip chargé sur Amazon S3, s'exécute dans un environnement isolé géré par AWS. Vous n'avez pas à mettre à l'échelle vos fonctions Lambda : chaque fois qu'une notification d'événement est reçue par votre fonction, AWS Lambda localise la capacité disponible au sein de sa flotte de calcul et exécute votre code avec les configurations d'exécution, de mémoire, de disque et de délai d'expiration que vous définissez. Avec ce modèle, AWS peut démarrer autant de copies de votre fonction que nécessaire.

Un niveau logique basé sur Lambda est toujours dimensionné aux besoins de vos clients. La capacité à absorber rapidement les pics de trafic grâce à la mise à l'échelle gérée et au lancement de code simultané, combinés à la tarification à l'utilisation de Lambda, vous permet de toujours répondre aux demandes des clients tout en ne payant pas pour la capacité de calcul inactive.

Déploiement et gestion sans serveur

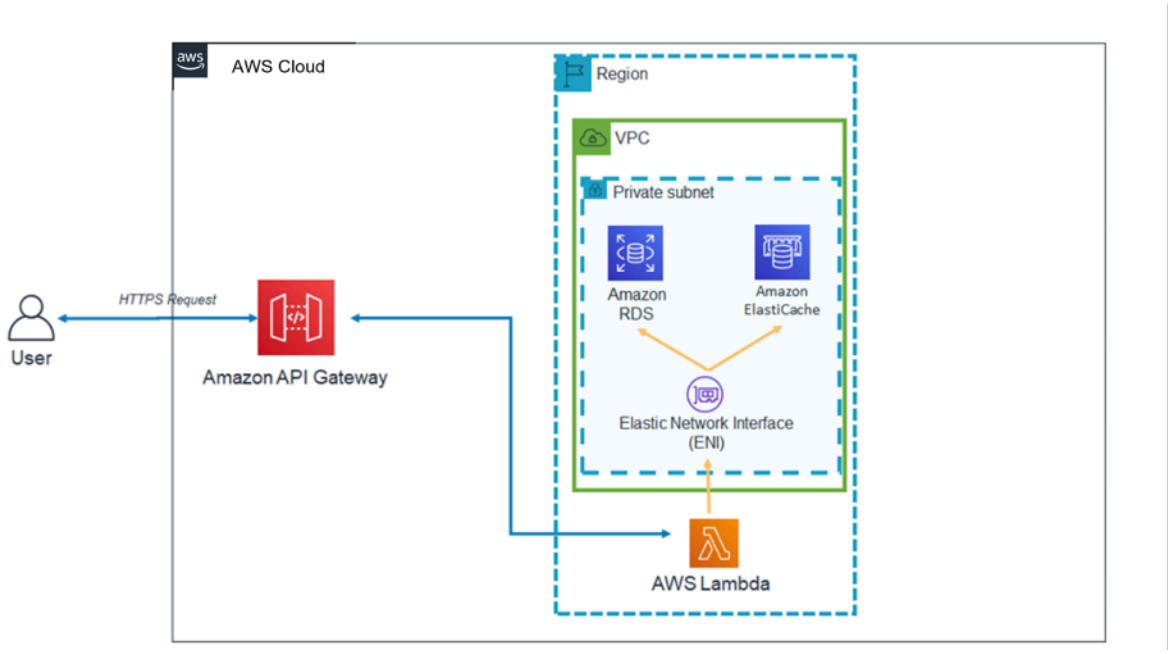
Pour vous aider à déployer et à gérer vos fonctions Lambda, utilisez AWS SAM AWS Serverless Application Model (AWS SAM), un cadre open source qui inclut :

- Spécification du modèle AWS SAM : syntaxe utilisée pour définir vos fonctions et décrire leurs environnements, leurs autorisations, leurs configurations et leurs événements afin de simplifier le chargement et le déploiement.
- CLI AWS SAM : commandes qui vous permettent de vérifier la syntaxe du modèle SAM, d'appeler des fonctions localement, de déboguer des fonctions Lambda et des fonctions de package de déploiement.

Vous pouvez également utiliser AWS CDK, qui est un cadre de développement logiciel permettant de définir une infrastructure cloud à l'aide de langages de programmation et de l'allouer via CloudFormation. CDK fournit un moyen impératif de définir les ressources AWS, tandis qu'AWS SAM fournit un moyen déclaratif.

En règle générale, lorsque vous déployez une fonction Lambda, elle est appelée avec des autorisations définies par le rôle IAM qui lui est attribué et peut atteindre les points de terminaison connectés à Internet. Au cœur de votre niveau logique, AWS Lambda est le composant qui s'intègre directement au niveau des données. Si votre niveau de données contient des informations commerciales ou utilisateur sensibles, il est important de s'assurer que ce niveau de données est correctement isolé (dans un sous-réseau privé).

Vous pouvez configurer une fonction Lambda pour vous connecter à des sous-réseaux privés dans un cloud privé virtuel (VPC) de votre compte AWS si vous souhaitez que la fonction Lambda accède à des ressources que vous ne pouvez pas exposer publiquement, comme une instance de base de données privée. Lorsque vous connectez une fonction à un VPC, Λ crée une interface réseau Elastic pour chaque sous-réseau de la configuration VPC de votre fonction et l'interface réseau Elastic est utilisée pour accéder à vos ressources internes en privé.



Modèle d'architecture Lambda à l'intérieur d'un VPC

L'utilisation de Lambda avec un VPC signifie que les bases de données et autres supports de stockage dont dépend votre logique métier peuvent être rendus inaccessibles sur Internet. Le VPC

garantit également que le seul moyen d'interagir avec vos données provenant d'Internet passe par les API que vous avez définies et les fonctions de code Lambda que vous avez écrites.

Amazon API Gateway

Amazon API Gateway est un service entièrement géré qui permet aux développeurs de créer, publier, gérer, contrôler et sécuriser facilement des API à n'importe quelle échelle.

Les clients (c'est-à-dire les niveaux de présentation) s'intègrent aux API exposées via API Gateway à l'aide de requêtes HTTPS standard. L'applicabilité des API exposées via API Gateway à une architecture à plusieurs niveaux orientée services est la capacité à séparer les différentes fonctionnalités de l'application et à exposer ces fonctionnalités via des points de terminaison REST. Amazon API Gateway possède des fonctions et des qualités spécifiques qui peuvent ajouter de puissantes fonctionnalités à votre niveau logique.

Intégration à AWS Lambda

Amazon API Gateway prend en charge les types d'API REST et HTTP. Une API REST API Gateway se compose de ressources et de méthodes. Une ressource est une entité logique à laquelle une application peut accéder via un chemin de ressource (par exemple, `/tickets`). Une méthode correspond à une demande d'API soumise à une ressource d'API (par exemple, `GET /tickets`). API Gateway vous permet de sauvegarder chaque méthode avec une fonction Lambda, c'est-à-dire que lorsque vous appelez l'API via le point de terminaison HTTPS exposé dans API Gateway, API Gateway appelle la fonction Lambda.

Vous pouvez connecter les fonctions API Gateway et Lambda à l'aide d'intégrations proxy et d'intégrations non proxy.

Intégrations de proxy

Dans une intégration proxy, l'intégralité de la demande HTTPS du client est envoyée telle quelle à la fonction Lambda. API Gateway transmet l'intégralité de la demande du client en tant que paramètre d'événement de la fonction de gestionnaire Lambda, et la sortie de la fonction Lambda est renvoyée directement au client (y compris le code d'état, les en-têtes, etc.).

Intégrations non proxy

Dans une intégration non proxy, vous configurez la façon dont les paramètres, les en-têtes et le corps de la demande client sont transmis au paramètre d'événement de la fonction de gestionnaire Lambda. En outre, vous configurez la façon dont la sortie Lambda est retraduite vers l'utilisateur.

> **Note**
>
> API Gateway peut également utiliser un proxy vers des ressources sans serveur supplémentaires extérieures à AWS Lambda, telles que des intégrations fictives (utiles pour le développement initial d'applications) et un proxy direct vers des objets S3.

Performances d'API stables dans plusieurs régions

Chaque déploiement d'Amazon API Gateway inclut une distribution Amazon CloudFront intégrée. CloudFront est un service de diffusion de contenu qui utilise le réseau mondial d'emplacements périphériques d'Amazon comme points de connexion pour les clients utilisant votre API. Cela permet de réduire la latence de réponse de votre API. En utilisant plusieurs emplacements périphériques à travers le monde, Amazon CloudFront fournit également des fonctionnalités pour lutter contre les scénarios d'attaque par déni de service distribué (DDoS). Pour de plus amples informations, veuillez consulter le livre blanc Bonnes pratiques AWS pour la résilience DDoS.

Vous pouvez améliorer les performances de demandes d'API spécifiques en utilisant API Gateway pour stocker les réponses dans un cache en mémoire facultatif. Cette approche offre non seulement des bénéfices en termes de performances pour les demandes d'API répétées, mais elle réduit également le nombre d'appels de vos fonctions Lambda, ce qui peut réduire votre coût global.

Encouragez l'innovation et réduisez les frais généraux grâce aux fonctions intégrées

Le coût de développement d'une nouvelle application est un investissement. L'utilisation d'API Gateway peut réduire le temps requis pour certaines tâches de développement ainsi que son coût total, ce qui permet aux organisations d'expérimenter et d'innover plus librement.

Au cours des phases initiales de développement d'applications, la mise en œuvre de la journalisation et la collecte des métriques sont souvent négligées pour livrer une nouvelle application plus rapidement. Cela peut entraîner un endettement technique et un risque opérationnel lors du déploiement de ces fonctions sur une application exécutée en production. Amazon API Gateway s'intègre parfaitement à Amazon CloudWatch, qui collecte et traite les données brutes d'API Gateway en métriques lisibles quasiment en temps réel pour contrôler l'exécution des API. API Gateway prend également en charge la journalisation des accès avec des rapports configurables et le suivi AWS X-Ray pour le débogage. Chacune de ces fonctions ne nécessite aucun code pour être écrite et

peut être ajustée dans les applications exécutées en production sans risque pour la logique métier principale.

La durée de vie globale d'une application peut être inconnue ou être connue pour être de courte durée. La création d'une étude de cas pour le développement de telles applications peut être facilitée si votre point de départ inclut déjà les fonctions gérées fournies par API Gateway et si vous n'engagez des coûts d'infrastructure qu'une fois que vos API ont commencé à recevoir des demandes. Pour de plus amples informations, veuillez consulter Tarification d'Amazon API Gateway.

Itérez rapidement, restez agile

L'utilisation d'Amazon API Gateway et AWS Lambda pour créer le niveau logique de votre API vous permet de vous adapter rapidement aux demandes changeantes de votre base d'utilisateurs en simplifiant le déploiement des API et la gestion des versions.

Étape de déploiement

Lorsque vous déployez une API dans API Gateway, vous devez associer le déploiement à une étape API Gateway. Chaque étape est un instantané de l'API et est disponible pour les applications clientes à appeler. En utilisant cette convention, vous pouvez facilement déployer des applications vers des étapes de développement, de test ou de production, et déplacer des déploiements entre les étapes. Chaque fois que vous déployez votre API à une étape, vous créez une version différente de l'API qui peut être annulée si nécessaire. Ces fonctions permettent aux fonctionnalités existantes et aux dépendances client de continuer sans être perturbées tandis que de nouvelles fonctionnalités sont publiées en tant que version d'API distincte.

Intégration découplée avec Lambda

L'intégration entre l'API dans API Gateway et la fonction Lambda peut être découplée à l'aide de variables d'étape API Gateway et d'un alias de fonction Lambda. Cela simplifie et accélère le déploiement de l'API. Au lieu de configurer le nom ou l'alias de la fonction Lambda directement dans l'API, vous pouvez configurer une variable d'étape dans l'API qui peut pointer vers un alias particulier dans la fonction Lambda. Pendant le déploiement, modifiez la valeur de la variable d'étape pour pointer vers un alias de fonction Lambda et l'API exécutera la version de la fonction Lambda derrière l'alias Lambda pour une étape particulière.

Déploiement d'une version canary

La version canary est une stratégie de développement logiciel qui consiste à déployer une nouvelle version d'API à des fins de test, tandis que la version de base reste déployée en tant que version

de production pour l'exploitation normale dans la même étape. Dans le cadre du déploiement d'une version canary, l'ensemble du trafic d'API est séparé de façon aléatoire entre la version de production et la version canary selon un rapport préconfiguré. Les API d'API Gateway peuvent être configurées pour le déploiement de la version canary afin de tester de nouvelles fonctions auprès d'un nombre limité d'utilisateurs.

Noms de domaine personnalisés

Vous pouvez fournir un nom d'URL intuitif et convivial à l'API au lieu de l'URL fournie par API Gateway. API Gateway fournit des fonctions permettant de configurer un domaine personnalisé pour les API. Avec des noms de domaine personnalisés, vous pouvez configurer le nom d'hôte de votre API et choisir un chemin de base à plusieurs niveaux (par exemple, `myservice`, `myservice/cat/v1` ou `myservice/dog/v2`) pour cartographier l'URL alternative à votre API.

Prioriser la sécurité des API

Toutes les applications doivent s'assurer que seuls les clients autorisés ont accès à leurs ressources d'API. Lors de la conception d'une application à plusieurs niveaux, vous pouvez tirer parti de différentes manières dont Amazon API Gateway contribue à sécuriser votre niveau logique :

Sécurité du transit

Toutes les demandes adressées à vos API peuvent être effectuées via HTTPS pour activer le chiffrement en transit.

API Gateway fournit des certificats SSL/TLS intégrés. Si vous utilisez l'option de nom de domaine personnalisé pour les API publiques, vous pouvez fournir votre propre certificat SSL/TLS à l'aide d'AWS Certificate Manager. API Gateway prend également en charge l'authentification TLS mutuelle (mTLS). L'authentification TLS mutuelle améliore la sécurité de votre API et vous aide a protéger vos données contre les attaques telles que les clients frauduleux ou les attaques de type HDM.

Autorisation d'API

Chaque combinaison ressource/méthode que vous créez dans le cadre de votre API se voit attribuer un Amazon Resource Name (ARN) unique qui peut être référencé dans les politiques AWS Identity and Access Management (IAM).

Il existe trois méthodes générales pour ajouter une autorisation à une API dans API Gateway :

- Rôles et politiques IAM : les clients utilisent l'autorisation AWS Signature Version 4 (SigV4) et les politiques IAM pour l'accès aux API. Les mêmes informations d'identification peuvent restreindre

ou autoriser l'accès à d'autres services et ressources AWS selon les besoins (par exemple, des compartiments Amazon S3 ou des tables Amazon DynamoDB).

- Groupes d'utilisateurs Amazon Cognito : les clients se connectent via un groupe d'utilisateurs Amazon Cognito et obtiennent des jetons, qui sont inclus dans l'en-tête d'autorisation d'une demande.
- Dispositif d'autorisation Lambda : définissez une fonction Lambda qui implémente un schéma d'autorisation personnalisé utilisant une stratégie de jeton de support (par exemple, OAuth et SAML) ou des paramètres de demande pour identifier les utilisateurs.

Restrictions d'accès

API Gateway prend en charge la génération de clés d'API et l'association de ces clés à un plan d'utilisation configurable. Vous pouvez contrôler l'utilisation des clés d'API avec CloudWatch.

API Gateway prend en charge la limitation, les limites de débit et les limites de taux de rafale pour chaque méthode de votre API.

API privées

En utilisant API Gateway, vous pouvez créer des API REST privées auxquelles vous pouvez accéder uniquement à partir de votre cloud privé virtuel dans Amazon VPC en utilisant un point de terminaison d'un VPC d'interface. Il s'agit d'une interface réseau de point de terminaison que vous créez dans votre VPC.

À l'aide des politiques de ressources, vous pouvez accorder ou refuser l'accès à votre API depuis des VPC et des points de terminaison d'un VPC sélectionnés, y compris entre plusieurs comptes AWS. Chaque point de terminaison peut être utilisé pour accéder à plusieurs API privées. Vous pouvez également utiliser AWS Direct Connect pour établir une connexion à partir d'un réseau sur site à Amazon VPC et accéder à votre API privée avec cette connexion.

Dans tous les cas, le trafic vers votre API privée utilise des connexions sécurisées et ne quitte pas le réseau Amazon : il est isolé de l'Internet public.

Protection par pare-feu à l'aide d'AWS WAF

Les API orientées Internet sont vulnérables aux attaques malveillantes. AWS WAF est un pare-feu d'application web qui permet de protéger les API contre de telles attaques. Il protège les API contre les exploits web courants tels que l'injection SQL et les attaques de scripting intersites. Vous pouvez utiliser AWS WAF avec API Gateway pour protéger les API.

Niveau de données

Le fait d'utiliser AWS Lambda comme niveau logique ne limite pas les options de stockage des données disponibles dans votre niveau de données. Les fonctions Lambda se connectent à n'importe quelle option de stockage de données en incluant le lecteur de base de données approprié dans le package de déploiement Lambda et utilisent un accès basé sur les rôles IAM ou des informations d'identification chiffrées (via AWS KMS ou AWS Secrets Manager).

Le choix d'un magasin de données pour votre application dépend fortement des exigences de celle-ci. AWS propose un certain nombre de magasins de données avec et sans serveur que vous pouvez utiliser pour composer le niveau de données de votre application.

Options de stockage des données sans serveur

Amazon S3 est un service de stockage d'objets qui offre une capacité de mise à l'échelle, une disponibilité des données, une sécurité et des performances de pointe.

Amazon Aurora est une base de données relationnelle compatible avec MySQL et PostgreSQL, conçue pour le cloud, qui combine les performances et la disponibilité des bases de données d'entreprise traditionnelles avec la simplicité et la rentabilité des bases de données open source. Aurora propose des modèles d'utilisation traditionnels et sans serveur.

Amazon DynamoDB est une base de données clé-valeur et document offrant des performances de latence de quelques millisecondes, quel que soit l'ordre de grandeur. Il s'agit d'une base de données sans serveur durable, multi-région et multi-maître entièrement gérée, intégrant la sécurité, la sauvegarde et la restauration, ainsi que la mise en cache en mémoire pour les applications à l'échelle d'Internet.

Amazon Timestream est un service de base de données en séries chronologiques rapide, évolutif et entièrement géré pour l'IoT et les applications opérationnelles. Il facilite le stockage et l'analyse de milliards d'événements par jour, pour un coût dix fois moins élevé que les bases de données relationnelles. Avec la montée en puissance des appareils IoT, des systèmes informatiques et des machines industrielles intelligentes, les données en séries chronologiques, à savoir les données qui mesurent des changements au fil du temps, constituent l'un des types de données ayant la plus forte croissance.

Amazon Quantum Ledger Database (Amazon QLDB) est une base de données de registre entièrement gérée qui fournit un journal de transaction transparent, inaltérable et vérifiable par

chiffrement appartenant à une autorité de confiance. Amazon QLDB suit chaque modification de données d'application et maintient un historique complet et vérifiable des modifications au fil du temps.

Amazon Keyspaces (pour Apache Cassandra) est un service de base de données évolutif, hautement disponible et géré, compatible avec Apache Cassandra. Avec Amazon Keyspaces, vous pouvez exécuter vos charges de travail Cassandra sur AWS à l'aide du même code d'application Cassandra et des mêmes outils pour développeur que ceux que vous utilisez aujourd'hui. Vous n'avez pas besoin de configurer, de corriger ou de gérer des serveurs, ni d'installer, de tenir à jour ou d'exploiter un logiciel. Amazon Keyspaces est sans serveur. Vous payez donc uniquement les ressources que vous utilisez, et le service se met automatiquement à l'échelle en augmentant ou en diminuant en fonction du trafic des applications.

Amazon Elastic File System (Amazon EFS) fournit un système de fichiers élastique set-and-forget simple et sans serveur qui vous permet de partager des données de fichiers sans allouer ni gérer de stockage. Il peut être utilisé avec AWS Cloud services et des ressources sur site. Il est conçu pour se mettre à l'échelle à la demande jusqu'à plusieurs pétaoctets, sans perturber les applications. Avec Amazon EFS, vous pouvez augmenter et réduire vos systèmes de fichiers automatiquement à mesure que vous ajoutez et supprimez des fichiers, ce qui élimine la nécessité d'allouer et de gérer la capacité pour répondre à la croissance. Amazon EFS peut être monté avec la fonction Lambda, ce qui en fait une option de stockage de fichiers viable pour les API.

Options de stockage de données avec serveur

Amazon Relational Database Service (Amazon RDS) est un service web géré qui facilite la configuration, l'exploitation et la mise à l'échelle d'une base de données relationnelle à l'aide de l'un des moteurs disponibles (Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle et Microsoft SQL Server) et exécuté sur différents types d'instance de base de données optimisés pour la mémoire, les performances ou les I/O.

Amazon Redshift est un service entièrement géré d'entreposage de pétaoctets de données dans le cloud.

Amazon ElastiCache est un déploiement entièrement géré de Redis ou Memcached. Déployez, exécutez et mettez à l'échelle de manière transparente les magasins de données en mémoire compatibles open source.

Amazon Neptune est un service de base de données orientée graphe fiable, rapide et entièrement gérée qui facilite la création et l'exécution d'applications utilisant des jeux de données hautement

connectés. Neptune prend en charge les modèles graphiques courants (graphiques de propriétés et RDF (Resource Description Framework) de W3C, ainsi que leurs langages de requête respectifs, ce qui vous permet de créer facilement des requêtes qui parcourent efficacement des jeux de données hautement connectés.

Amazon DocumentDB (compatible avec MongoDB) est un service de base de données document rapide, évolutif, hautement disponible et entièrement géré, prenant en charge les charges de travail MongoDB.

Enfin, vous pouvez également utiliser des magasins de données s'exécutant indépendamment sur Amazon EC2 en tant que niveau de données d'une application à plusieurs niveaux.

Niveau de présentation

Le niveau de présentation est chargé d'interagir avec le niveau logique via les points de terminaison REST API Gateway exposés sur Internet. Tout client ou appareil compatible avec HTTPS peut communiquer avec ces points de terminaison, ce qui donne à votre niveau de présentation la flexibilité nécessaire pour prendre de nombreuses formes (applications de bureau, applications mobiles, pages web, appareils IoT, etc.). Selon vos besoins, votre niveau de présentation peut utiliser les offres sans serveur AWS suivantes : tout client ou appareil compatible avec HTTPS peut communiquer avec ces points de terminaison, ce qui donne à votre niveau de présentation la flexibilité nécessaire pour prendre de nombreuses formes (applications de bureau, applications mobiles, pages web, appareils IoT, etc.). Selon vos besoins, votre niveau de présentation peut utiliser les offres sans serveur AWS suivantes :

- Amazon Cognito : service de synchronisation des données et de l'identité des utilisateurs sans serveur qui vous permet d'ajouter l'inscription, la connexion et le contrôle d'accès des utilisateurs à vos applications web et mobiles rapidement et efficacement. Amazon Cognito s'adapte à des millions d'utilisateurs et prend en charge la connexion avec les fournisseurs d'identité sociale comme Facebook, Google et Amazon, ainsi qu'avec les fournisseurs d'identité entreprise via SAML 2.0.
- Amazon S3 avec CloudFront : vous permet de diffuser des sites web statiques, tels que des applications d'une seule page, directement à partir d'un compartiment S3 sans devoir fournir de serveur web. Vous pouvez utiliser CloudFront en tant que réseau de diffusion de contenu géré (CDN) pour améliorer les performances et activer le protocole SSL/TL à l'aide de certificats gérés ou personnalisés.

AWS Amplify est un ensemble d'outils et de services qui peuvent être utilisés ensemble ou séparément, pour aider les développeurs frontend web et mobiles à créer des applications complètes et évolutives à technologie AWS. Amplify propose un service entièrement géré permettant de déployer et d'héberger des applications web statiques à l'échelle mondiale, utilisé via le CDN fiable d'Amazon, qui dispose de centaines de points de présence à travers le monde et de flux de travail CI/CD intégrés qui permettent d'accélérer le cycle de publication de vos applications. Amplify prend en charge les cadres web populaires, tels que JavaScript, React, Angular, Vue, Next.js, et les plateformes mobiles, telles qu'Android, iOS, React Native, Ionic et Flutter. Selon vos configurations réseaux et les exigences de vos applications, vous devrez peut-être activer vos API Gateway pour qu'elles soient conformes au partage des ressources d'origine croisée (CORS). La conformité CORS permet aux navigateurs web d'invoquer directement vos API à partir de pages web statiques.

Lorsque vous déployez un site web avec CloudFront, un nom de domaine CloudFront vous est fourni pour atteindre votre application (par exemple, `d2d47p2vcczkh2.cloudfront.net`). Vous pouvez utiliser Amazon Route 53 pour enregistrer des noms de domaine et les diriger vers votre distribution CloudFront, ou diriger des noms de domaine déjà détenus vers votre distribution CloudFront. Cela permet aux utilisateurs d'accéder à votre site en utilisant un nom de domaine familier. Notez que vous pouvez également attribuer un nom de domaine personnalisé à l'aide de Route 53 à votre distribution API Gateway, ce qui permet aux utilisateurs d'appeler des API à l'aide de noms de domaine familiers.

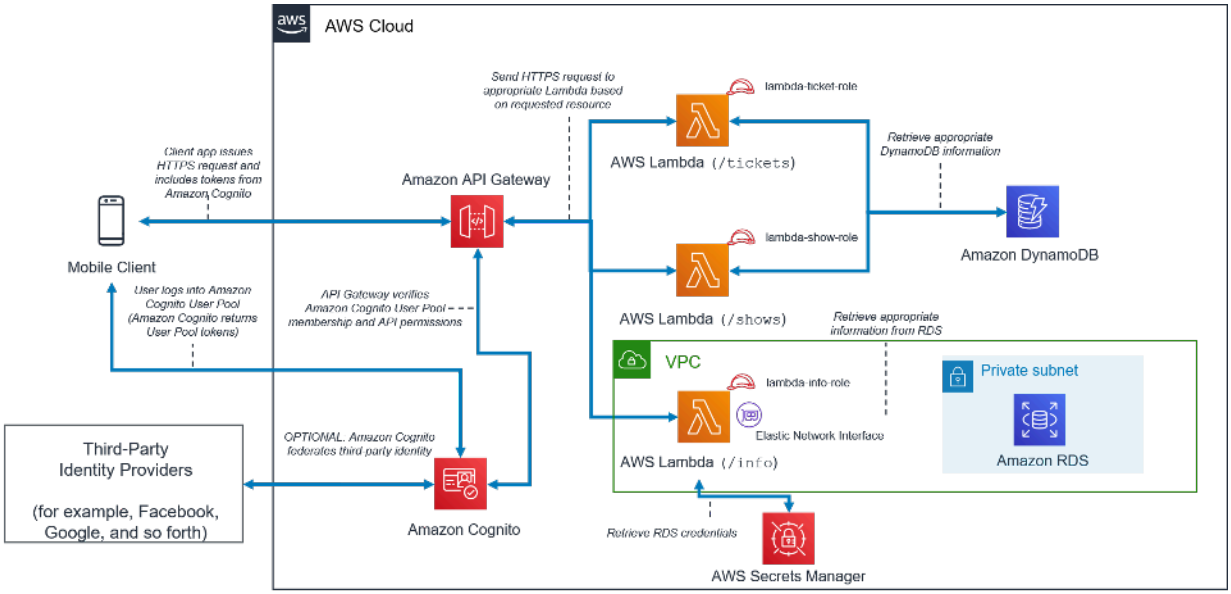
Exemples de modèles d'architecture

Vous pouvez mettre en œuvre des modèles d'architecture populaires à l'aide d'API Gateway et AWS Lambda comme votre niveau logique. Ce livre blanc présente les modèles d'architecture les plus populaires qui exploitent les niveaux logiques basés sur AWS Lambda :

- Backend mobile : une application mobile communique avec API Gateway et Lambda pour accéder aux données de l'application. Ce modèle peut être étendu aux clients HTTPS génériques qui n'utilisent pas de ressources AWS sans serveur pour héberger des ressources du niveau de présentation (tels que les clients de bureau, le serveur web s'exécutant sur EC2, etc.).
- Application d'une seule page : une application d'une seule page hébergée dans Amazon S3 et CloudFront communique avec API Gateway et AWS Lambda pour accéder aux données de l'application.
- Application web : l'application web est un backend d'application web à usage général, piloté par des événements, qui utilise AWS Lambda avec API Gateway pour sa logique métier. Elle utilise également DynamoDB comme base de données et Amazon Cognito pour la gestion des utilisateurs. Tout le contenu statique est hébergé à l'aide d'Amplify.

Outre ces deux modèles, ce livre blanc traite de l'applicabilité de Lambda et d'API Gateway à une architecture générale de microservices. Une architecture de microservices est un modèle populaire qui, bien qu'il ne s'agisse pas d'une architecture à trois niveaux standard, implique le découplage des composants d'application et leur déploiement en tant qu'unités individuelles de fonctionnalités sans état qui communiquent entre elles.

Backend mobile



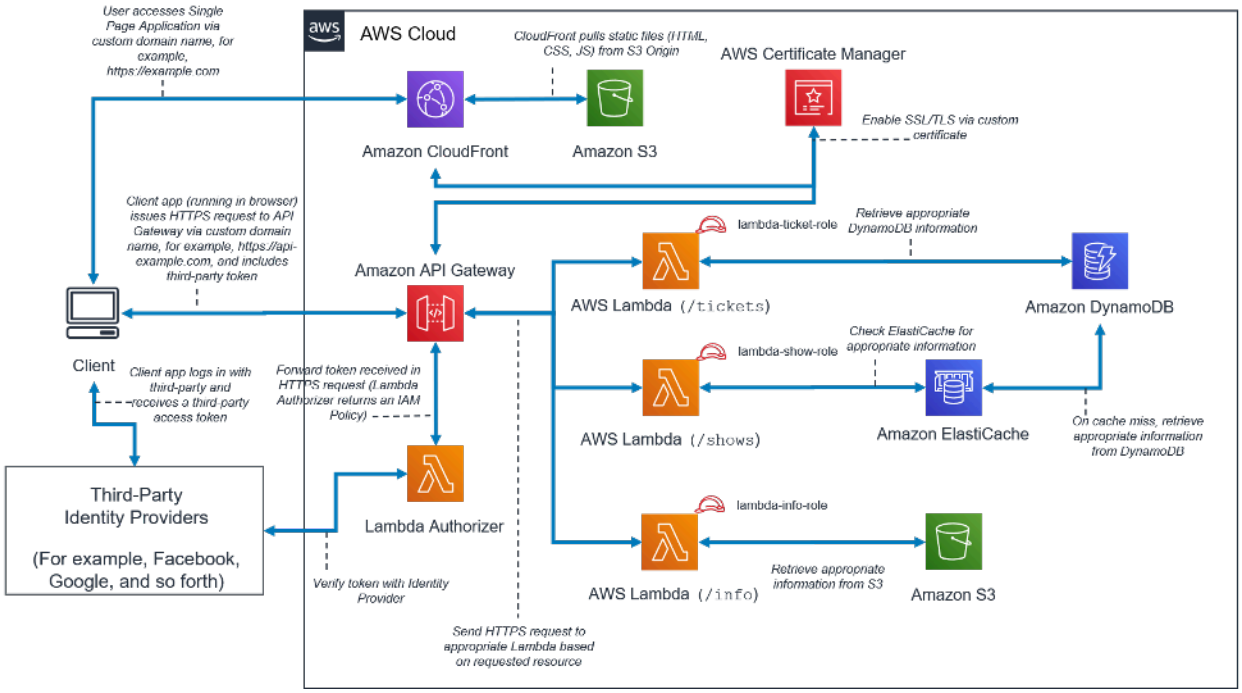
Modèle architectural de backend mobile sans serveur

Tableau 1 : composants de niveau backend mobile

Niveau	Composants
Présentation	Application mobile exécutée sur l'appareil d'un utilisateur.
Logique	Amazon API Gateway avec AWS Lambda. Cette architecture présente trois services exposés (`/tickets`, `/shows` et `/info`). Les points de terminaison API Gateway sont sécurisés par des groupes d'utilisateurs Amazon Cognito. Dans cette méthode, les utilisateurs se connectent aux groupes d'utilisa teurs Amazon Cognito (en utilisant un tiers fédéré si nécessaire) et reçoivent des jetons d'accès et d'identification qui sont utilisés pour autoriser les appels API Gateway.

Niveau	Composants
	Chaque fonction Lambda se voit attribuer son propre rôle Identity and Access Managemen t (IAM) pour fournir un accès à la source de données appropriée.
Données	DynamoDB est utilisé pour les services `/tickets` et `/shows`. Amazon RDS est utilisé pour le service `/info`. Cette fonction Lambda récupère les informati ons d'identification Amazon RDS auprès d'AWS Secrets Manager et utilise une interface réseau Elastic pour accéder au sous-réseau privé.

Application d'une seule page

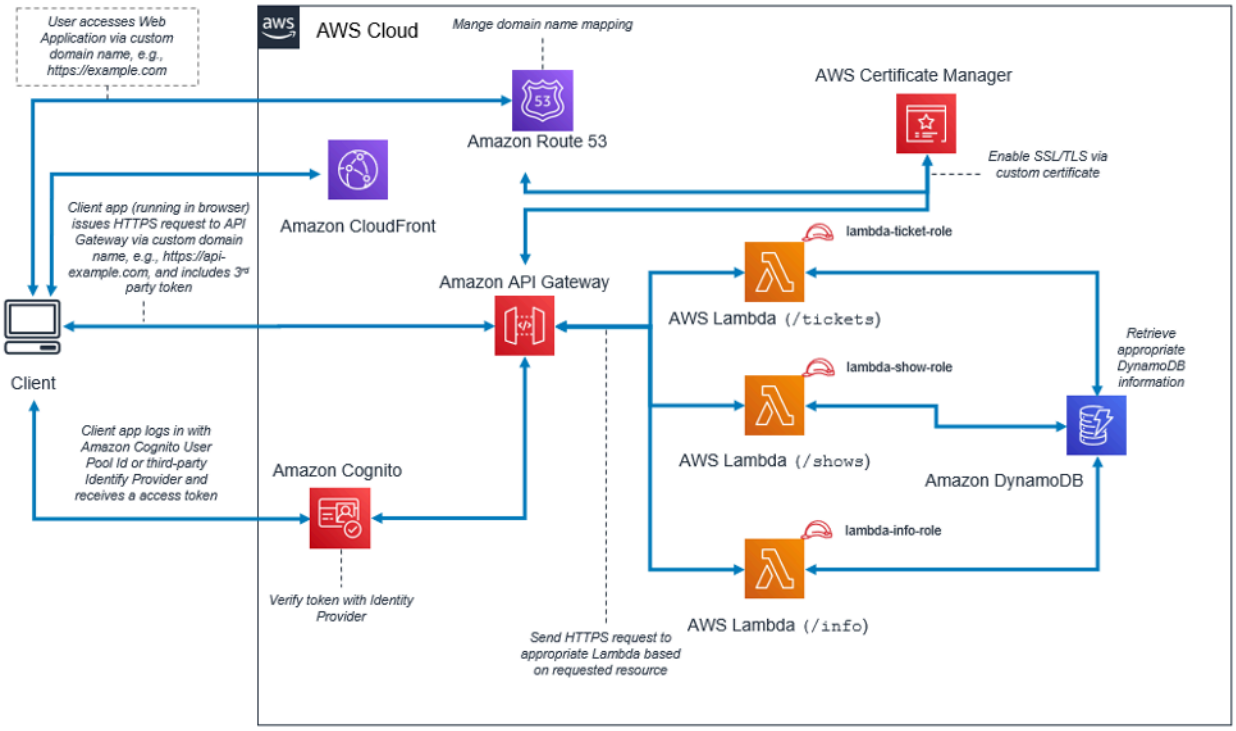


Modèle architectural d'une application d'une seule page sans serveur

Tableau 2 : composants d'une application d'une seule page

Niveau	Composants
Présentation	Contenu de site web statique hébergé dans Amazon S3, distribué par CloudFront. AWS Certificate Manager permet d'utiliser un certificat SSL/TLS personnalisé.
Logique	API Gateway avec AWS Lambda. Cette architecture présente trois services exposés (`/tickets`, `/shows` et `/info`). Les points de terminaison API Gateway sont sécurisés par un mécanisme d'autorisation Lambda. Dans cette méthode, les utilisateurs se connectent via un fournisseur d'identité tiers et obtiennent des jetons d'accès et d'identification. Ces jetons sont inclus dans les appels API Gateway. Le mécanisme d'autorisation Lambda valide ces jetons et génère une politique IAM contenant des autorisations d'initiation d'API. Chaque fonction Lambda se voit attribuer son propre rôle IAM afin de fournir un accès à la source de données appropriée.
Données	Amazon DynamoDB est utilisé pour les services `/tickets` et `/shows`. Le service `/shows` utilise Amazon ElastiCache pour améliorer les performances de la base de données. Les échecs du cache sont envoyés à DynamoDB. Amazon S3 est utilisé pour héberger du contenu statique utilisé par le `/info service`.

Application web



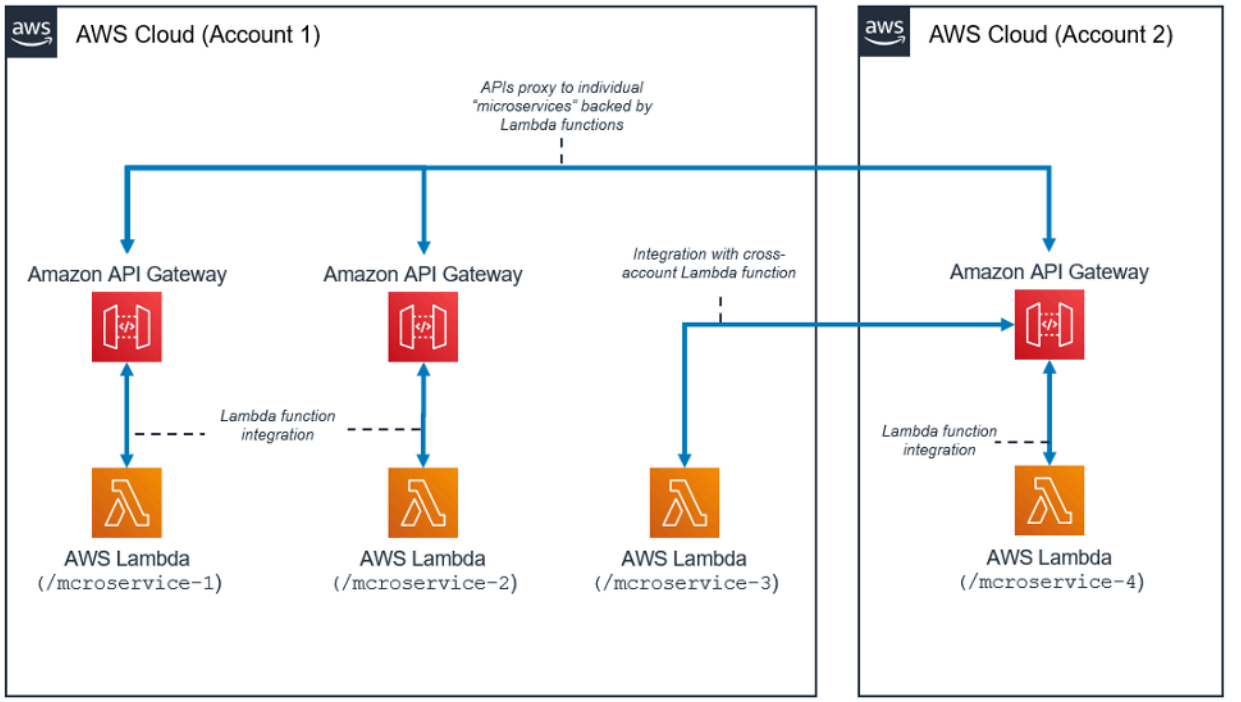
Modèle architectural d'une application web

Tableau 3 : composants de l'application web

Niveau	Composants
Présentation	L'application frontend est constituée de tout le contenu statique (HTML, CSS, JavaScript et images) qui est généré par des utilitaires React tels que create-react-app. Amazon CloudFront héberge tous ces objets. L'application web, lorsqu'elle est utilisée, télécharge toutes les ressources sur le navigateur et commence à s'y exécuter. L'application web se connecte au backend en appelant les API.
Logique	La couche logique est construite à l'aide de fonctions Lambda gérées par les API REST API Gateway.

Niveau	Composants
	Cette architecture présente plusieurs services exposés. Il existe plusieurs fonctions Lambda différentes, chacune gérant un aspect différent de l'application. Les fonctions Lambda se trouvent derrière API Gateway et sont accessibles à l'aide de chemins d'URL d'API. L'authentification des utilisateurs est gérée à l'aide de groupes d'utilisateurs Amazon Cognito ou de fournisseurs d'utilisateurs fédérés. API Gateway utilise l'intégration prête à l'emploi avec Amazon Cognito. Ce n'est qu'après l'authentification d'un utilisateur que le client reçoit un jeton JSON Web Token (JWT) qu'il doit ensuite utiliser lors des appels d'API. Chaque fonction Lambda se voit attribuer son propre rôle IAM afin de fournir un accès à la source de données appropriée.
Données	Dans cet exemple particulier, DynamoDB est utilisé pour le stockage des données, mais d'autres bases de données ou services de stockage Amazon spécialement conçus peuvent être utilisés en fonction du cas d'utilisation et du scénario d'utilisation.

Microservices avec Lambda



Modèle architectural pour les microservices avec Lambda

Le modèle architectural de microservice n'est pas lié à l'architecture classique à trois niveaux ; cependant, ce modèle populaire peut tirer des avantages significatifs de l'utilisation de ressources sans serveur.

Dans cette architecture, chacun des composants d'application est découplé, avec un déploiement et un fonctionnement indépendants. Une API créée avec Amazon API Gateway et des fonctions lancées par la suite par AWS Lambda suffisent pour créer un microservice. Votre équipe peut utiliser ces services pour découpler et fragmenter votre environnement au niveau de granularité souhaité.

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

Lorsque vous créez des microservices à l'aide de ressources sans serveur, ces problèmes deviennent moins difficiles à résoudre et, dans certains cas, disparaissent tout simplement. Le modèle de microservices sans serveur facilite la création de chaque microservice suivant (API Gateway permet même le clonage d'API existantes et l'utilisation de fonctions Lambda dans d'autres

comptes). L'optimisation de l'utilisation du serveur n'est plus pertinente avec ce modèle. Enfin, Amazon API Gateway fournit des kits SDK client générés par programme dans un certain nombre de langages populaires afin de réduire les frais d'intégration.

Conclusion

Le modèle d'architecture à plusieurs niveaux encourage les bonnes pratiques de création de composants d'application simples à maintenir, découpler et mettre à l'échelle. Lorsque vous créez un niveau logique dans lequel l'intégration s'effectue par API Gateway et le calcul s'effectue dans AWS Lambda, vous atteignez ces objectifs tout en réduisant les efforts nécessaires pour les atteindre. Ensemble, ces services fournissent un frontend d'API HTTPS pour vos clients et un environnement sécurisé pour appliquer votre logique métier tout en supprimant la surcharge liée à la gestion d'une infrastructure basée sur un serveur typique.

Participants

Ont contribué à la préparation du présent document :

- Andrew Baird, architecte de solutions AWS
- Bryant Bost, consultant AWS ProServe
- Stefano Buliani, chef de produit principal, Tech, AWS Mobile
- Vyom Nagrani, chef de produit principal, AWS Mobile
- Ajay Nair, chef de produit principal, AWS Mobile
- Rahul Popat, architecte de solutions mondiales
- Brajendra Singh, architecte de solutions principal

Autres lectures

Pour plus d'informations, veuillez consulter :

- Livres blancs et guides AWS

Révisions du document

Pour être informé des mises à jour de ce livre blanc, abonnez-vous au flux RSS.

update-history-change	update-history-description	update-history-date
Livre blanc mis à jour	Mis à jour avec les nouvelles fonctions et les nouveaux modèles de service.	20 octobre 2021
Livre blanc mis à jour	Mis à jour avec les nouvelles fonctions et les nouveaux modèles de service.	1er juin 2021
Livre blanc mis à jour	Mis à jour avec les nouvelles fonctions de service.	25 septembre 2019
Publication initiale	Livre blanc publié.	1er novembre 2015

Mentions légales

Les clients sont responsables de leur propre évaluation indépendante des informations contenues dans ce document. Le présent document : (a) est fourni à titre informatif uniquement, (b) représente les offres et pratiques actuelles de produits AWS, qui sont susceptibles d'être modifiées sans préavis, et (c) ne crée aucun engagement ou assurance de la part d'AWS et de ses affiliés, fournisseurs ou concédants de licences. Les produits ou services AWS sont fournis « en l'état » sans garantie, représentation ou condition, de quelque nature que ce soit, explicite ou implicite. Les responsabilités et obligations d'AWS envers ses clients sont déterminées par les contrats AWS, et le présent document ne fait pas partie d'un contrat entre AWS et ses clients, ni le modifie.