Livre blanc AWS

Implémentation des microservices sur AWS

Implémentation des microservices sur AWS: Livre blanc AWS

Table of Contents

Implémentation des microservices sur AWS

Date de publication : 9 novembre 2021 (Historique du document et contributeurs)

Résumé

Les microservices s'intègrent dans une approche architecturale et organisationnelle du développement logiciel créée pour accélérer les cycles de déploiement, à favoriser l'innovation et l'appropriation, à améliorer la maintenabilité et la capacité de mise à l'échelle des applications logicielles et à faire évoluer les organisations qui fournissent des logiciels et des services en utilisant une approche agile qui aide les équipes à travailler indépendamment. Avec cette approche basée sur les microservices, les logiciels sont constitués de petits services qui communiquent via des API bien définies qui peuvent être déployés indépendamment. Ces services sont la propriété de petites équipes autonomes. Cette approche agile est la clé de la mise à l'échelle réussie de votre organisation.

Trois modèles courants ont été observés lorsque des clients AWS créent des microservices : orienté API, orienté événement et livraison de données. Le présent livre blanc présente les trois approches et résume les caractéristiques communes des microservices, aborde les principaux défis que présente la conception de microservices et décrit comment les équipes produit peuvent s'appuyer sur Amazon Web Services (AWS) pour relever ces défis.

En raison de la nature plutôt complexe des divers sujets abordés dans ce livre blanc, notamment le stockage des données, la communication asynchrone et la découverte de services, il est recommandé de prendre en compte les exigences spécifiques et les cas d'utilisation de leurs applications, en plus des conseils fournis, avant d'effectuer choix architecturaux.

Introduction

Les architectures de microservices ne sont pas une approche complètement nouvelle de l'ingénierie logicielle, mais plutôt une combinaison de divers concepts réussis et éprouvés tels que :

- Développement logiciel agile
- Architectures orientées services
- Conception privilégiant les API
- Intégration continue / livraison continue (CI/CD)

Dans de nombreux cas, des modèles de conception de l'application à douze facteurs sont utilisés pour les microservices.

Dans un premier temps, ce livre blanc décrit les différents aspects qui caractérisent une architecture de microservices hautement évolutive et tolérante aux pannes (interface utilisateur, implémentation de microservices et magasin de données) et comment la concevoir sur AWS à l'aide des technologies de conteneurs. Les services AWS sont ensuite recommandés pour l'implémentation d'une architecture de microservices sans serveur classique afin de réduire la complexité opérationnelle.

L'approche sans serveur est définie comme un modèle opérationnel par les principes suivants :

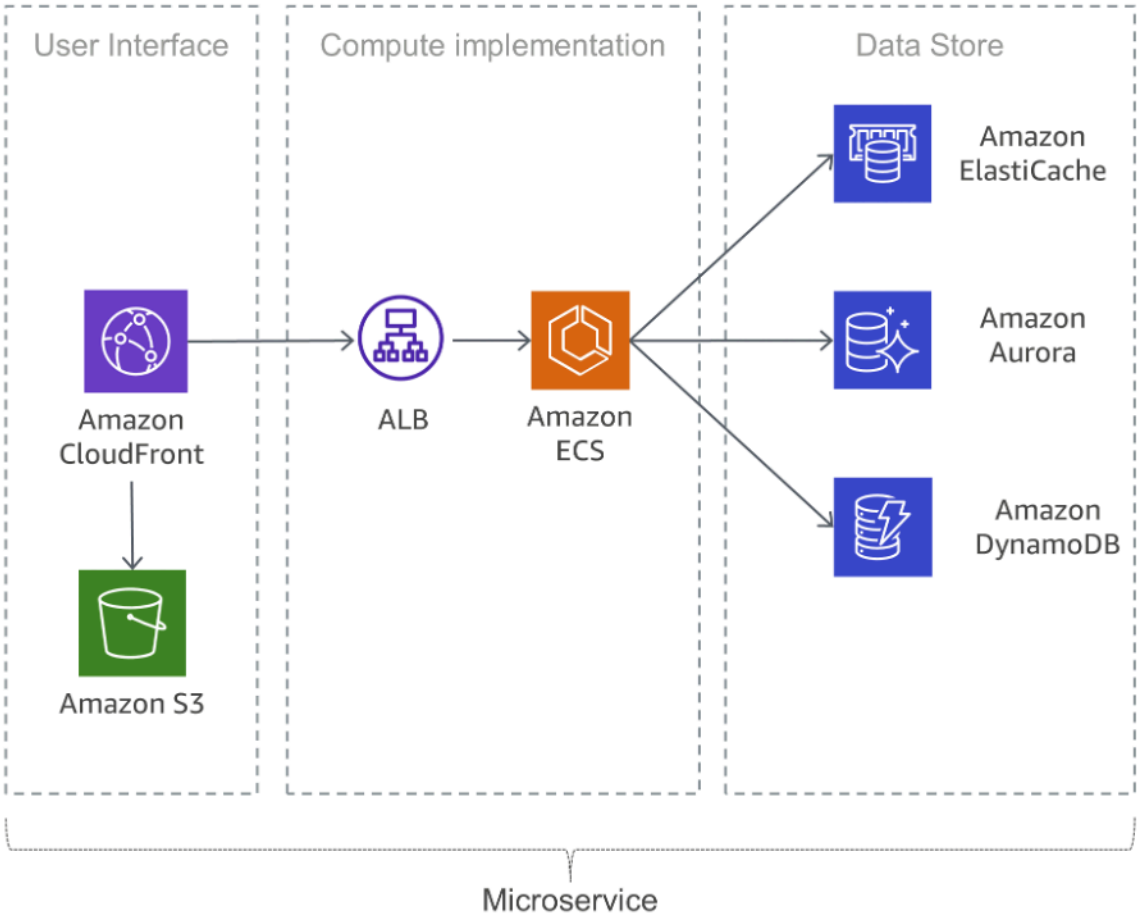
- Aucune infrastructure à allouer ou gérer
- Scalabilité automatique en fonction des unités de consommation
- Modèle de facturation en fonction de la valeur proposée
- Disponibilité et tolérance aux pannes intégrées

Enfin, ce livre blanc aborde le système global et es aspects inter-services d'une architecture de microservices, tels que la surveillance et l'audit distribués, la cohérence des données, ainsi que la communication asynchrone.

Ce livre blanc se concentre uniquement sur les charges de travail exécutées dans le cloud AWS. Il n'aborde pas les scénarios hybrides ni les stratégies de migration. Pour plus d'informations sur la migration, reportez-vous au livre blanc Méthodologie de migration de conteneurs.

Architecture de microservices sur AWS

Les applications monolithiques classiques sont conçues à l'aide de différentes couches, telles qu'une couche d'interface utilisateur (IU), une couche business et une couche de persistance. L'idée centrale d'une architecture de microservices consiste à diviser les fonctionnalités en verticaux cohésifs (et non par couches technologiques, mais par l'implémentation d'un domaine). La figure suivante illustre une architecture de référence pour une application de microservices classique sur AWS.



Application de microservices classique sur AWS

Rubriques

- Interface utilisateur
- Microservices
- Magasin de données

Interface utilisateur

Les applications web modernes utilisent souvent des frameworks JavaScript pour implémenter une application d'une seule page qui communique avec une API REST (Representational State Transfer) ou RESTful. Le contenu web statique peut être diffusé à l'aide d'Amazon Simple Storage Service (S3) et d'Amazon CloudFront.

Dans la mesure où les clients d'un microservice sont traités à partir de l'emplacement périphérique le plus proche et reçoivent des réponses d'un cache ou d'un serveur proxy avec des connexions optimisées à l'origine, les latences peuvent être considérablement réduites. Toutefois, les microservices qui s'exécutent à proximité l'un de l'autre ne bénéficient pas d'un réseau de diffusion de contenu. Dans certains cas, cette approche peut même ajouter une latence supplémentaire. Une bonne pratique consiste à mettre en place d'autres mécanismes de mise en cache pour réduire les communications et les latences. Pour de plus amples informations, veuillez consulter la rubrique the section called "Bavardage").

Microservices

Les API sont la porte d'entrée des microservices. Cela signifie que les API servent de point d'entrée pour la logique des applications derrière un ensemble d'interfaces de programmation, généralement une API de services web REST. Cette API accepte et traite les appels des clients, et peut implémenter des fonctionnalités telles que la gestion du trafic, le filtrage des demandes, l'acheminement, la mise en cache, l'authentification et l'autorisation.

Implantation de microservices

AWS a intégré des éléments de base qui soutiennent le développement des microservices. Deux approches populaires consistent à utiliser AWS Lambda et les conteneurs Docker avec AWS Fargate.

Avec AWS Lambda, chargez votre code et de laissez Lambda s'occuper de tout pour que vous puissiez exécuter et mettre à l'échelle l'implémentation en fonction de votre courbe de demande réelle, tout en assurant une haute disponibilité. Aucune administration de l'infrastructure n'est nécessaire. Lambda prend en charge plusieurs langages de programmation et peut être déclenché à partir d'autres services AWS, ou appelé directement depuis n'importe quelle application web ou mobile. L'un des atouts principaux d'AWS Lambda est de vous rendre agile : vous pouvez vous concentrer sur votre logique métier car la sécurité et la mise à l'échelle sont gérées par AWS. L'approche résolument innovante de Lambda est à la base de la plateforme évolutive.

Le déploiement basé sur des conteneurs est l'une des approches courante permettant de réduire les efforts opérationnels pour le déploiement. Les technologies de conteneur comme Docker ont gagné en popularité au cours de ces dernières années en raison d'avantages comme la portabilité, la productivité et l'efficacité. Avec les conteneurs, la phase d'apprentissage peut être difficile, et vous devez penser à des correctifs de sécurité pour la surveillance et vos images Docker. Amazon Elastic Container Service (Amazon ECS) et Amazon Elastic Kubernetes Service (Amazon EKS) vous évitent de devoir installer, exploiter et mettre à l'échelle votre infrastructure de gestion de clusters. Avec de simples appels d'API, vous pouvez lancer et stopper des applications activées par Docker, interroger l'état complet de votre cluster et accéder à de nombreuses fonctions connues, comme les groupes de sécurité, les répartitions de charge, les volumes Amazon Elastic Block Store (Amazon EBS) et les rôles AWS Identity and Access Management (IAM).

AWS Fargate est un moteur de calcul sans serveur pour les conteneurs qui fonctionne à la fois avec Amazon ECS et Amazon EKS. Avec Fargate, vous n’avez plus à vous soucier de provisionner suffisamment de ressources de calcul pour vos applications conteneur. Fargate peut lancer des dizaines de milliers de conteneurs et s’adapter facilement pour exécuter vos applications les plus stratégiques.

Amazon ECS prend en charge les stratégies et contraintes de positionnement de conteneurs afin de personnaliser la façon dont Amazon ECS positionne les tâches et y met fin. La contrainte de positionnement d'une tâche est une règle prise en compte lors du positionnement des tâches. Vous pouvez associer des attributs, qui sont des paires clé/valeur, à vos instances de conteneur, puis utiliser une contrainte pour positionner des tâches en fonction de ces attributs. Par exemple, vous pouvez utiliser des contraintes afin de positionner certains microservices en fonction du type d'instance ou de la capacité d'instance, comme les applications alimentées par le GPU.

Amazon EKS s'appuie sur les versions à jour du logiciel open source Kubernetes, ce qui vous permet d'utiliser tous les plug-ins et outils existants de la communauté Kubernetes. Les applications s’exécutant sur Amazon EKS sont entièrement compatibles avec les applications s’exécutant sur n’importe quel environnement Kubernetes standard, qu’il s’agisse de centres de données sur site ou de clouds publics. Amazon EKS intègre IAM dans Kubernetes, vous permettant ainsi d’enregistrer les entités IAM avec le système d’authentification natif de Kubernetes. Il n'est pas nécessaire de configurer manuellement les informations d'identification pour l'authentification auprès du plan de contrôle Kubernetes. L'intégration IAM vous permet d'utiliser IAM pour vous authentifier directement auprès du plan de contrôle lui-même et fournir un accès granulaire précis au point de terminaison public de votre plan de contrôle Kubernetes.

Les images Docker utilisées dans Amazon ECS et Amazon EKS peuvent être stockées dans Amazon Elastic Container Registry (Amazon ECR). Avec Amazon ECR, vous n'avez plus besoin de gérer ni de mettre à l'échelle l'infrastructure requise pour faire fonctionner votre registre de conteneurs.

L'intégration et la livraison continues (CI/CD) constituent des bonnes pratiques et un élément essentiel d'une initiative DevOps qui autorise les changements logiciels rapides tout en maintenant la stabilité et la sécurité du système. Toutefois, cela n'entre pas dans le cadre de ce livre blanc. Pour de plus amples informations, veuillez consulter le livre blanc Application de l'intégration continue et de la livraison continue sur AWS.

Liens privés

AWS PrivateLink est une technologie hautement disponible et évolutive qui vous permet de connecter de façon privée votre VPC aux services AWS compatibles, aux services hébergés par d'autres comptes AWS (services de points de terminaison VPC) et aux services partenaires compatibles AWS Marketplace. Pour communiquer avec le service, vous n'avez pas besoin d'une passerelle Internet, d'un périphérique NAT, d'une adresse IP publique, d'une connexion AWS Direct Connect ou d'une connexion VPN. Le trafic entre votre VPC et le service ne quitte pas le réseau Amazon.

Les liens privés constituent un excellent moyen d'améliorer l'isolation et la sécurité de l'architecture des microservices. Un microservice, par exemple, peut être déployé dans un VPC totalement séparé, avec un équilibreur de charge en avant-plan, et exposé à d'autres microservices via un point de terminaison AWS PrivateLink. Avec cette configuration, l'utilisation de AWS PrivateLink permet au trafic réseau à destination et en provenance du microservice de ne jamais traverser l'Internet public. La conformité réglementaire pour les services traitant des données sensibles tels que des données PCI, HIPPA et le bouclier de protection des données UE/États-Unis constitue un cas d'utilisation pour ce type d'isolement. En outre, AWS PrivateLink permet de connecter des microservices entre différents comptes et VPC Amazon, sans avoir besoin de règles de pare-feu, de définitions de chemin ou de tables de routage, ce qui simplifie la gestion du réseau. En utilisant PrivateLink, les fournisseurs de logiciels en tant que service (SaaS) et les ISV peuvent également proposer leurs solutions basées sur des microservices avec une isolation opérationnelle complète et un accès sécurisé.

Magasin de données

Le magasin de données est utilisé pour conserver les données nécessaires aux microservices. Les magasins les plus populaires pour les données de session sont des caches en mémoire tels

que Memcached ou Redis. AWS propose les deux technologies dans le cadre du service géré Amazon ElastiCache.

L'intégration d'un cache entre les serveurs d'application et une base de données est un mécanisme courant qui permet de réduire la charge de lecture sur la base de données, ce qui, en retour, permet aux ressources d'être utilisées pour prendre en charge plus d'écritures. Les caches peuvent également améliorer la latence.

Les bases de données relationnelles sont toujours très populaires pour stocker les données structurées et les objets métier. AWS propose six moteurs de base de données (Microsoft SQL Server, Oracle, MySQL, MariaDB, PostgreSQL et Amazon Aurora) en tant que services managés via Amazon Relational Database Service (Amazon RDS).

Les bases de données relationnelles, toutefois, ne sont pas conçues pour une mise à l'échelle illimitée, ce qui peut rendre difficile et énergivore le recours à des techniques pour prendre en charge un grand nombre de demandes.

Les bases de données NoSQL ont été conçues pour favoriser la capacité de mise à l'échelle, les performances et la disponibilité à la cohérence des bases de données relationnelles. Un des aspects importants réside dans le fait que les bases de données NoSQL n’imposent pas de schéma strict. Les données sont réparties entre les partitions qui peuvent être mises à l’échelle horizontalement, et sont récupérées à l’aide des clés de partition.

Étant donné que chacun des microservices est conçu pour n'avoir qu'une seule utilité, ils ont généralement un modèle de données simplifié qui peut être parfaitement adapté à la persistance NoSQL. Il est important de comprendre que les bases de données NoSQL présentent des modèles d’accès différents de ceux des bases de données relationnelles. Par exemple, il est impossible de joindre des tables. Si cela s’avère nécessaire, la logique doit être implémentée dans l’application. Vous pouvez utiliser Amazon DynamoDB pour créer une table de base de données dotée de capacités de stockage et d'extraction de données, quels que soient le volume et le nombre de requêtes. DynamoDB offre des performances de latence de l'ordre de quelques millisecondes ; toutefois, certains cas d'utilisation requièrent des temps de réponse de l'ordre de microsecondes. Amazon DynamoDB Accelerator (DAX) fournit des fonctionnalités de mise en cache pour accéder aux données.

DynamoDB offre également une fonctionnalité de mise à l’échelle automatique pour ajuster dynamiquement la capacité de débit en fonction du trafic réel. Toutefois, dans certains cas, la planification des capacités est difficile ou impossible en raison de pics d’activité importants et brefs dans votre application. Pour de telles situations, DynamoDB offre une option à la demande, qui

s’accompagne d’une tarification à la demande simple. DynamoDB à la demande est capable de répondre instantanément à des milliers de requêtes par seconde sans planification de capacité.

Réduction de la complexité opérationnelle

L'architecture décrite précédemment dans ce livre blanc utilise déjà des services managés, mais les instances Amazon Elastic Compute Cloud (Amazon EC2) doivent toujours être gérées. Il est possible de réduire davantage les efforts opérationnels nécessaires pour exécuter, gérer et surveiller les microservices à l'aide d'une architecture entièrement sans serveur.

Rubriques

- Implémentation d'API
- Microservices sans serveur
- Reprise après sinistre
- Haute disponibilité
- Déploiement d'applications basées sur Lambda

Implémentation d'API

L'architecture, l'amélioration continue, le déploiement, la surveillance et la gestion d'une API peuvent être des tâches fastidieuses. Parfois, différentes versions d'API doivent être exécutées pour assurer la rétrocompatibilité pour tous les clients. Les différentes étapes du cycle de développement (par exemple, le développement, le test et la production) multiplient davantage les efforts opérationnels.

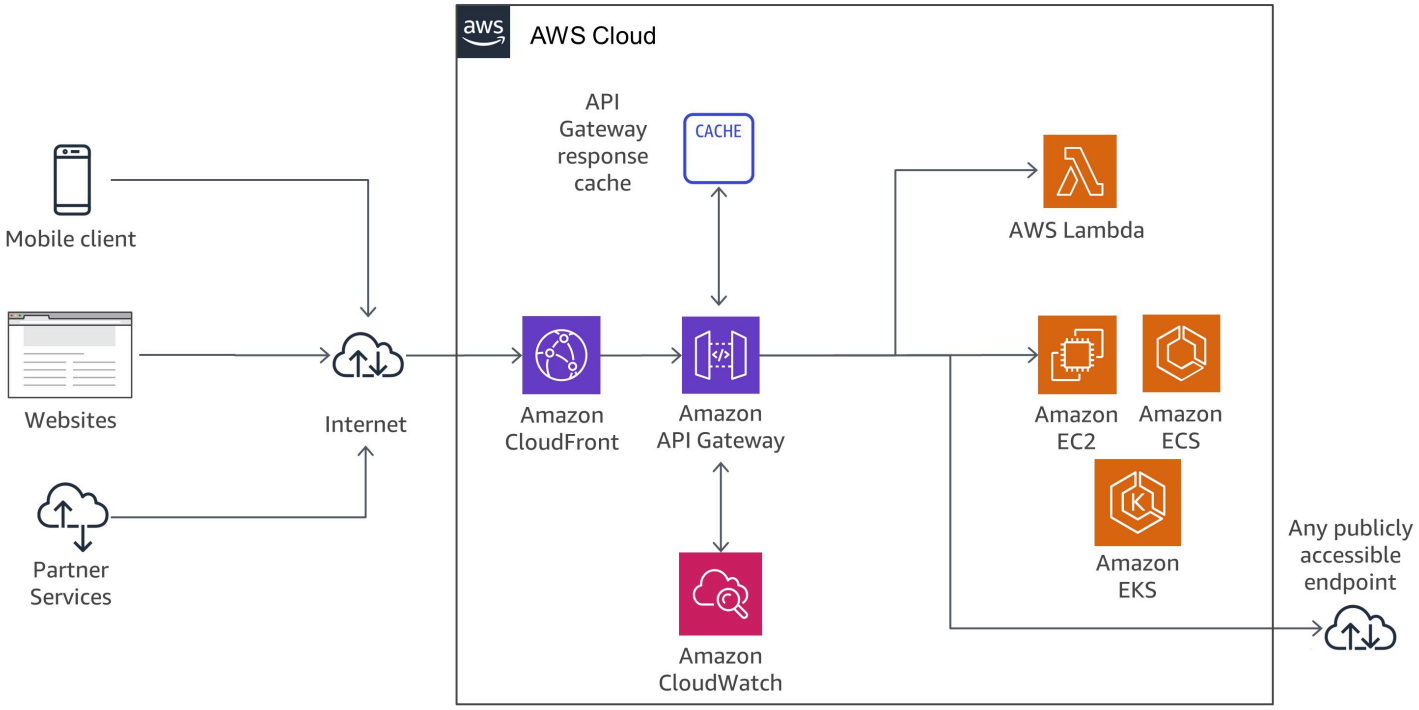
L'autorisation est une caractéristique essentielle pour toutes les API, mais elle est généralement complexe à mettre en place et implique des tâches répétitives. Lorsqu'une API est publiée et aboutit, le défi suivant consiste à gérer, surveiller et monétiser l'écosystème des développeurs tiers utilisant l'API.

D'autres caractéristiques et défis importants incluent la limitation des demandes pour protéger les services backend, la mise en cache des réponses d'API, la gestion de la transformation des demandes et des réponses et la génération de définitions d'API et de la documentation à l'aide d'outils tels que Swagger.

Amazon API Gateway répond à ces défis et réduit la complexité opérationnelle liée à la création et à la gestion d'API RESTful. API Gateway vous permet de créer vos API par programmation en important des définitions Swagger à l'aide de l'API AWS ou de la Console de gestion AWS Management Console. API Gateway sert de porte d'entrée à n'importe quelle application web

s'exécutant sur Amazon EC2, Amazon ECS, AWS Lambda ou sur n'importe quel environnement sur site. Fondamentalement, API Gateway vous permet d'exécuter des API sans avoir à gérer de serveurs.

La figure suivante illustre la façon dont API Gateway gère les appels d'API et interagit avec d'autres composants. Les demandes provenant d'appareils mobiles, de sites web, ou d'autres services backend sont acheminées vers le point de présence (PoP) CloudFront le plus proche afin de réduire au minimum la latence et de garantir une expérience utilisateur optimale.



Flux d'appels API Gateway

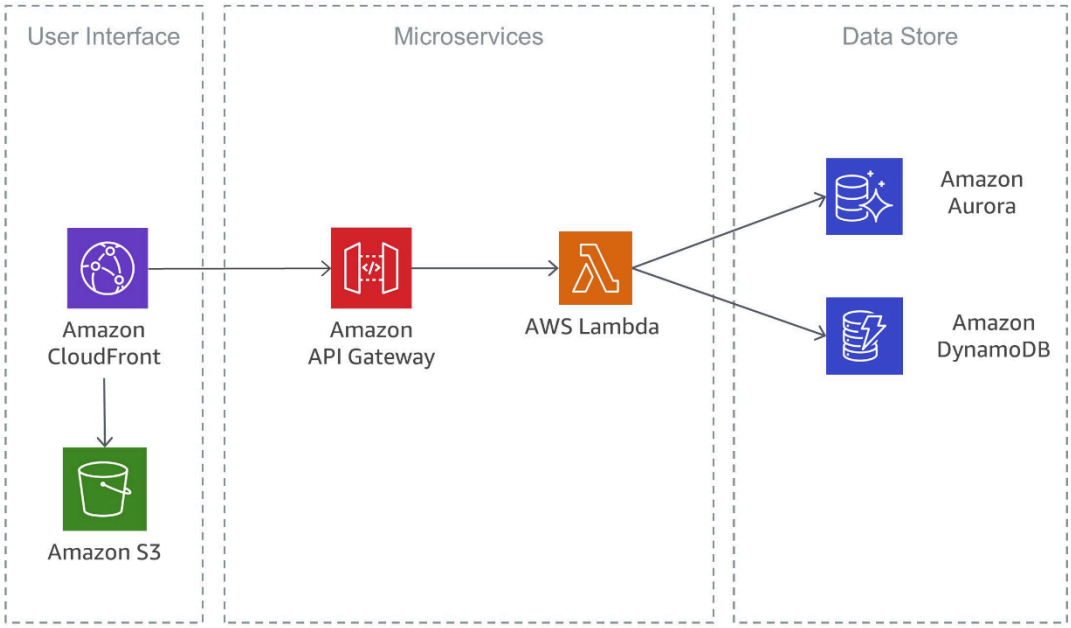
Microservices sans serveur

« Aucun serveur n'est plus facile à gérer qu'un modèle sans serveur. »

La suppression des serveurs est un excellent moyen d'éliminer la complexité opérationnelle.

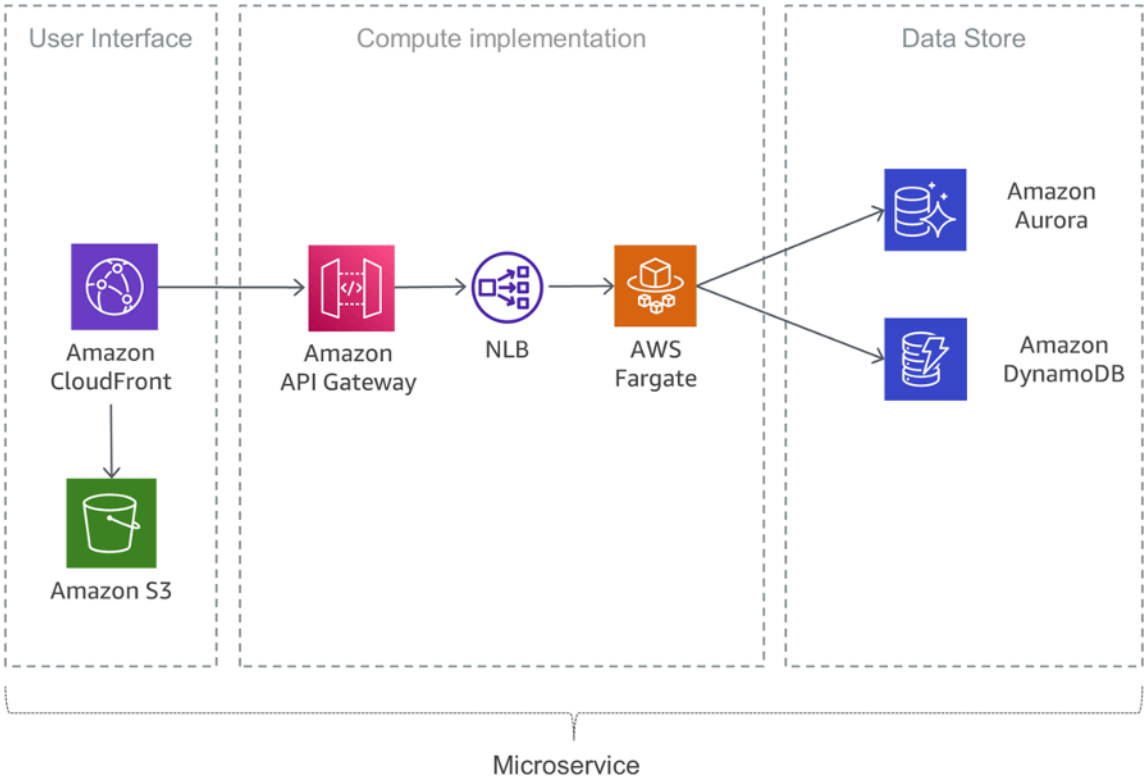
La plateforme Lambda est étroitement intégrée à API Gateway. La capacité d'effectuer des appels synchrones depuis API Gateway vers Lambda permet de créer des applications complètement sans serveur, ce qui est présenté en détail dans le Manuel du développeur Amazon API Gateway.

La figure suivante illustre l'architecture d'un microservice sans serveur s'exécutant sur AWS Lambda, dans laquelle le service complet est conçu à partir de services managés. Cela élimine la charge architecturale de la conception pour la mise à l'échelle et la haute disponibilité, mais aussi les efforts opérationnels nécessaires pour exécuter et surveiller l'infrastructure sous-jacente du microservice.



Microservice sans serveur utilisant AWS Lambda

Une implémentation similaire, également basée sur des services sans serveur, est illustrée dans la figure suivante. Dans cette architecture, les conteneurs Docker sont associés à Fargate. L'infrastructure sous-jacente n'est donc plus un sujet de préoccupation. Outre DynamoDB, cette architecture utilise Amazon Aurora Serverless, une configuration à scalabilité automatique à la demande dédiée à Amazon Aurora (édition compatible MySQL) dans laquelle la base de données démarre, s'arrête et augmente ou diminue la capacité en fonction des besoins de votre application, et ce de manière automatique.



Microservice sans serveur utilisant Fargate

Reprise après sinistre

Comme indiqué précédemment dans l'introduction du présent livre blanc, les applications de microservices standard sont implémentées à l'aide des modèles d'application à douze facteurs. La section Processus indique que « les processus à douze facteurs sont sans état et ne partagent rien. Toutes les données qui doivent être conservées doivent être stockées dans un service de sauvegarde avec état, généralement une base de données. »

Pour une architecture de microservices classique, cela signifie que la reprise après sinistre doit se concentrer principalement sur les services en aval qui maintiennent l'état de l'application. Il peut s'agir par exemple de systèmes de fichiers, de bases de données ou de files d'attente. Lors de la création d'une stratégie de reprise après sinistre, les organisations planifient le plus souvent l'objectif de délai de reprise et l'objectif de point de reprise.

L'objectif de délai de reprise correspond au délai maximum acceptable entre l'interruption du service et la restauration du service. Cet objectif détermine ce qui est considéré comme un créneau acceptable d'indisponibilité du service. Il est défini par l'organisation.

L'objectif de point de reprise correspond à la durée acceptable depuis le dernier point de reprise des données. Cet objectif détermine ce qui est considéré comme étant une perte de données acceptable entre le dernier point de reprise et l'interruption du service. Il est défini par l'organisation.

Pour de plus amples informations, veuillez consulter le livre blanc Récupération des charges de travail après un sinistre sur AWS : récupération dans le cloud.

Haute disponibilité

Cette section examine de plus près la haute disponibilité pour différentes options de calcul.

Amazon EKS exécute des instances de plan de données et un contrôle Kubernetes sur plusieurs zones de disponibilité, ce qui permet d'assurer une haute disponibilité. Amazon EKS détecte et remplace automatiquement les instances de plan de contrôle défectueuses. Il fournit des mises à niveau de version automatiques et des correctifs pour les instances de plan de contrôle. Ce plan de contrôle se compose d'au moins deux nœuds de serveur d'API et de trois nœuds etcd qui s'exécutent sur trois zones de disponibilité au sein d'une région. Amazon EKS utilise l'architecture de Régions AWS pour maintenir une haute disponibilité.

Amazon ECR héberge des images dans une architecture hautement disponible et hautes performances, vous permettant de déployer de manière fiable des images pour vos applications de conteneur sur plusieurs zones de disponibilité. Amazon ECR fonctionne avec Amazon EKS, Amazon ECS et AWS Lambda, ce qui simplifie les flux de travail, du développement à la production.

Amazon ECS est un service régional qui simplifie l'exécution des conteneurs de manière hautement disponible sur plusieurs zones de disponibilité au sein d'une Région AWS. Amazon ECS inclut plusieurs stratégies de planification qui placent vos conteneurs sur vos différents clusters en fonction des ressources requises (par exemple, CPU ou RAM) et des exigences en termes de disponibilité.

AWS Lambda exécute votre fonction dans plusieurs zones de disponibilité afin de vous assurer qu'elle est disponible pour traiter les événements en cas d'interruption de service dans une seule zone. Si vous configurez votre fonction pour vous connecter à un cloud privé virtuel (VPC) de votre compte, spécifiez les sous-réseaux dans plusieurs zones de disponibilité, pour garantir la haute disponibilité.

Déploiement d'applications basées sur Lambda

Vous pouvez utiliser AWS CloudFormation pour spécifier, déployer et configurer des applications sans serveur.

Le AWS Serverless Application Model (AWS SAM) est un moyen pratique de définir des applications sans serveur. AWS SAM est pris en charge en mode natif par CloudFormation et définit une syntaxe simplifiée pour exprimer des ressources sans serveur. Pour déployer votre application, spécifiez les ressources dont vous avez besoin dans le cadre de votre application, ainsi que les politiques d'autorisations associées à un modèle CloudFormation, d'empaqueter vos artefacts de déploiement et de déployer le modèle. Basé sur AWS SAM, SAM Local est un outil AWS Command Line Interface (AWS CLI) qui vous fournit un environnement pour développer, tester et analyser localement vos applications sans serveur avant de les charger sur le runtime Lambda. Vous pouvez utiliser AWS SAM Local pour créer un environnement local de test qui simule l'environnement d'exécution AWS.

Composants de systèmes distribués

Après avoir découvert comment AWS peut relever les défis liés à chaque microservice, l'accent est mis sur les défis inter-services tels que la découverte de services, la cohérence des données, la communication asynchrone et la surveillance et l'audit distribués.

Rubriques

Découverte de services

L'un des principaux défis des architectures de microservices est de permettre aux services de se détecter mutuellement et d'interagir entre eux. Les caractéristiques distribuées des architectures de microservices compliquent non seulement la communication entre les services, mais posent également d’autres difficultés, telles que la vérification de l'état de ces systèmes et l'annonce de la disponibilité de nouvelles applications. Vous devez décider comment et où stocker les métadonnées de stockage, telles que les données de configuration, qui peuvent être utilisées par les applications. Dans cette section sont exposées plusieurs techniques permettant d'effectuer la découverte de services sur AWS pour les architectures orientées microservices.

Découverte de services basée sur un DNS

Amazon ECS inclut désormais la fonction intégrée de découverte de services qui permet aux services conteneurisés de se détecter mutuellement et de s'interconnecter, et ce en toute simplicité.

Jusqu'ici, pour rendre possible cette détection et cette connexion réciproques, vous deviez configurer et exécuter votre propre système de découverte de services basé sur Amazon Route 53, AWS Lambda et ECS Event Stream, ou connecter chaque service à un équilibreur de charge.

Amazon ECS crée et gère un registre de noms de services à l’aide de l’API Auto Naming de Route 53. Les noms sont automatiquement mis en correspondance avec un ensemble

d’enregistrements DNS pour vous permettre de vous référer à un service à partir du nom figurant dans votre code et d’écrire des requêtes DNS pour que le nom mène au point de terminaison du service lors de l’exécution. Vous pouvez spécifier les conditions du bilan d’intégrité dans la définition des tâches d’un service, et Amazon ECS s’assure que seuls les points de terminaison du service en bon état soient retournés lors de la recherche d’un service.

De plus, vous pouvez également utiliser la découverte unifiée de services pour les services gérés par Kubernetes. Pour activer dette intégration, AWS a participé au projet External DNS, un projet d'incubateur Kubernetes.

Une autre option consiste à tirer parti des capacités d'AWS Cloud Map. AWS Cloud Map optimise les capacités des API Auto Naming en fournissant un registre de services dédié aux ressources, par exemple les protocoles Internet (IP), les URL et les ressources Amazon Resource Name (ARN), mais aussi un mécanisme de découverte de services basés sur les API doté d'un système rapide de propagation des modifications et de la capacité d'utiliser des attributs pour circonscrire l'ensemble des ressources découvertes. Les ressources Auto Naming de Route 53 existantes sont automatiquement mises à niveau vers AWS Cloud Map.

Logiciel tiers

Une autre approche permettant de mettre en œuvre la découverte de services consiste à utiliser un logiciel tiers comme HashiCorp Consul, etcd ou Netflix Eureka. Ces trois exemples sont des magasins clé-valeur distribués fiables. Pour HashiCorp Consul, un Quick Start AWS permet de configurer un environnement de Cloud AWS flexible et évolutif, et de lancer HashiCorp Consul automatiquement dans la configuration de votre choix.

Maillages de services

Dans une architecture de microservices avancée, une application peut être composée de centaines, voire de milliers de services. Et en règle générale, ce ne sont pas les services eux-mêmes qui constituent la partie la plus complexe de l'application, mais la communication entre ces services. Les maillages de services constituent une couche supplémentaire de gestion de la communication inter-services, chargée de surveiller et de contrôler le trafic dans les architectures de microservices. Cela permet aux tâches, telles que la découverte de services, d'être entièrement gérées par cette couche.

En règle générale, un maillage de services est divisé en un plan de données et un plan de contrôle. Le plan de données consiste en un ensemble de proxys intelligents qui sont déployés avec le code de l'application en tant que proxy secondaire spécial qui intercepte toutes les communications réseau entre les microservices Le plan de contrôle est responsable de la communication avec les proxys.

Les maillages de services sont transparents, ce qui signifie que les développeurs d'application n'ont pas besoin de se préoccuper de cette couche supplémentaire, ni de modifier le code d'application existant. AWS App Mesh est un maillage de services qui fournit un réseau de niveau application pour permettre la communication entre vos services sur plusieurs types d'infrastructure de calcul. App Mesh standardise la façon dont vos microservices communiquent, en vous donnant une visibilité complète et en garantissant une haute disponibilité pour vos applications.

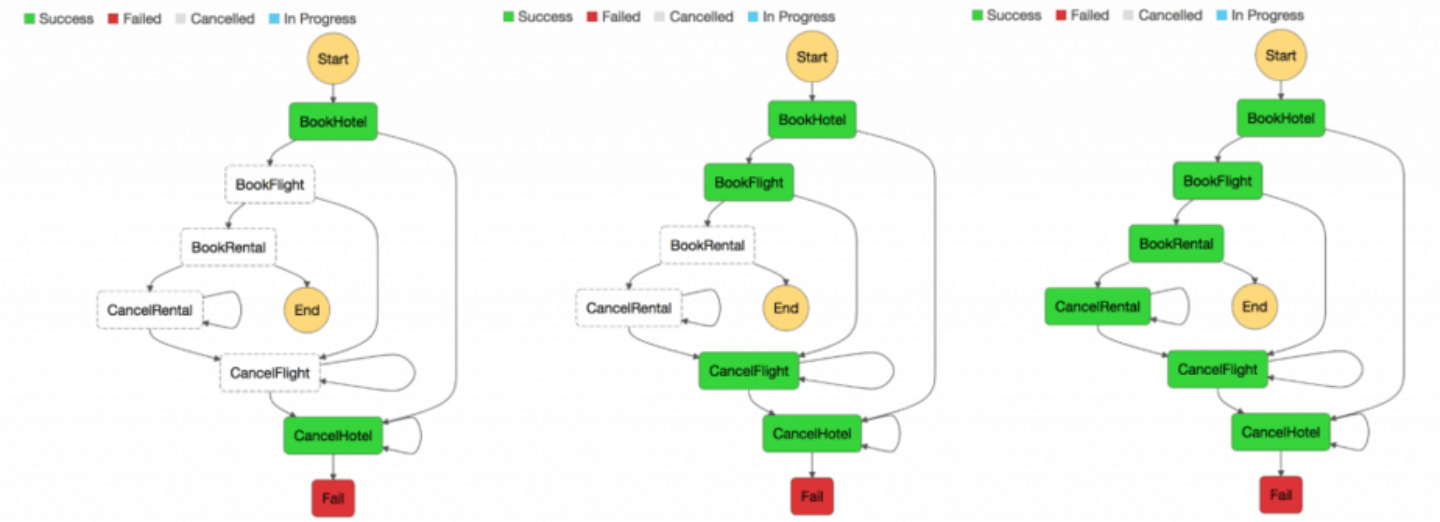
Vous pouvez utiliser AWS App Mesh avec des microservices existants ou nouveaux s'exécutant sur AWS Fargate, Amazon ECS, Amazon EKS et Kubernetes auto-géré sur AWS. App Mesh peut surveiller et contrôler les communications pour les microservices exécutés sur des clusters, des systèmes d'orchestration ou des VPC en tant qu'application unique, sans aucune modification du code.

Gestion des données distribuées

Les applications monolithiques sont généralement soutenues par une grande base de données relationnelle, qui définit un modèle de données unique commun à tous les composants de l'application. Dans une approche de microservices, une telle base de données centrale empêcherait d'atteindre l'objectif visant à créer des composants décentralisés et indépendants. Chaque composant de microservice doit disposer de sa propre couche de persistance des données.

Cependant, la gestion des données distribuées génère de nouveaux défis. Conséquence du Théorème CAP, les architectures de microservices distribuées nécessitent de faire un compromis sur la cohérence, au profit des performances. Elles doivent donc adopter le principe de la cohérence éventuelle.

Dans un système distribué, les transactions commerciales peuvent couvrir plusieurs microservices. Comme ces derniers ne peuvent pas exploiter une unique transaction ACID, vous risquez d'obtenir des exécutions partielles. Dans ce cas, une logique de contrôle serait nécessaire pour reprendre les transactions déjà traitées. À cet effet, le modèle distribué Saga pattern est couramment utilisé. En cas d'échec d'une transaction commerciale, Saga orchestre une série de transactions compensatoires qui annulent les modifications qui ont été opérées par les transactions précédentes. Les AWS Step Functions facilitent la mise en œuvre d'un coordinateur d'exécution Saga comme illustré dans la figure suivante.



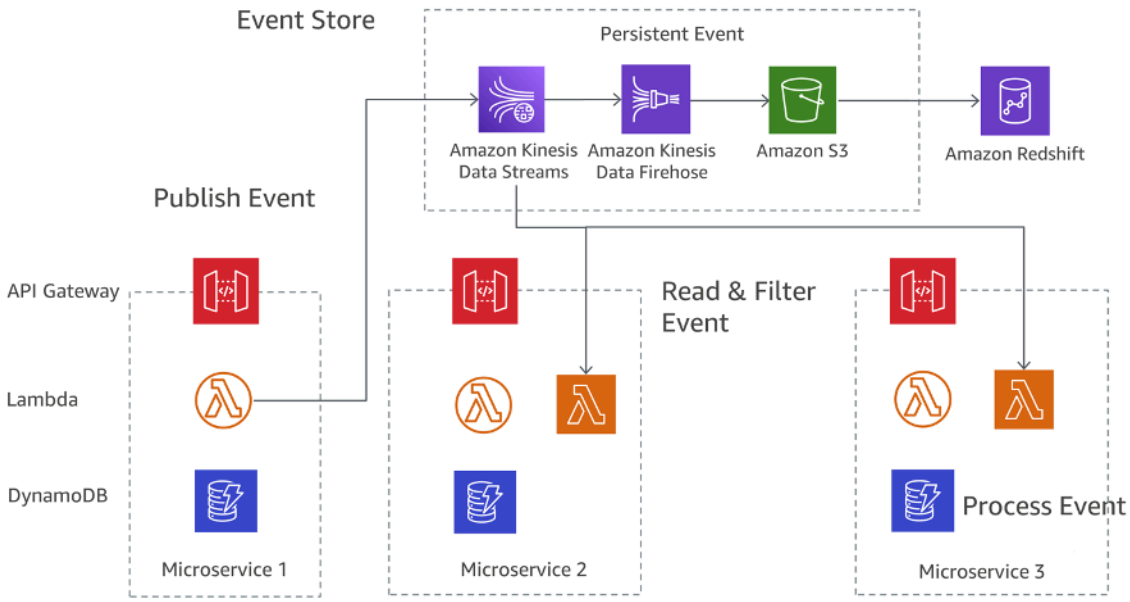
Coordinateur d'exécution Saga

La création d'un magasin centralisé de données de référence critiques, géré par des outils et des procédures de base de gestion de données, permet aux microservices de synchroniser leurs données critiques et de restaurer un état précédent. En utilisant AWS Lambda avec des événements Amazon CloudWatch Events planifiés, vous pouvez créer un mécanisme simple de nettoyage et de déduplication.

Les changements d'état affectent fréquemment plusieurs microservices. Dans ces cas, le sourcing d'événements s'est avéré être une méthode utile. L'idée de base des sources d'événements consiste à représenter et à conserver chaque changement d'état de l'application sous la forme d'un enregistrement d'événements. Au lieu de conserver l'état de l'application, les données sont stockées sous la forme d'un flux d'événements. Les systèmes de journalisation des transactions de base de données et de contrôle de version sont deux exemples bien connus de sources d'événements. Les sources d'événement présentent différents avantages : leur état peut être déterminé et reconstitué pour n'importe quel point dans le temps. Elles produisent naturellement une piste d'audit permanente et facilitent également le débogage.

Dans le contexte d'architectures de microservices, le sourcing d'événements permet de découpler différentes parties d'une application à l'aide d'un modèle de publication et d'abonnement. Il permet d'importer les mêmes données d'événements dans différents modèles de données pour des microservices distincts. Le sourcing d'événements est souvent utilisé conjointement avec le modèle CQRS (Command, Query, Responsibility, Segregation) pour découpler les charges de travail en écriture et en lecture et en optimiser les performances, la capacité de mise à l'échelle et la sécurité. Dans les systèmes de gestion de données traditionnels, les commandes et les requêtes sont exécutées sur le même référentiel de données.

La figure suivante montre comment le modèle de sourcing d'événements peut être mis en œuvre sur AWS. Amazon Kinesis Data Streams constitue le composant principal du magasin d'événements central qui capture les modifications d'application en tant qu'événements et les conserve sur Amazon S3. La figure décrit trois différents microservices composés d'Amazon API Gateway, d'AWS Lambda et d'Amazon DynamoDB. Les flèches indiquent le flux des événements : quand le microservice 1 subit un événement de modification de l'état, il publie un événement en envoyant un message à Kinesis Data Streams. Tous les microservices exécutent leur propre application Kinesis Data Streams dans AWS Lambda, qui lit une copie du message, la filtre en fonction de sa pertinence pour le microservice et la transmet éventuellement en vue d'un traitement ultérieur. Si votre fonction renvoie une erreur, Lambda retente de traiter le lot jusqu'à ce que le traitement réussisse ou que les données expirent. Pour éviter les partitions bloquées, vous pouvez configurer le mappage de source d'événement pour réessayer avec une taille de lot plus petite, limiter le nombre de nouvelles tentatives ou supprimer les enregistrements qui sont trop vieux. Afin de conserver les événements supprimés, vous pouvez configurer le mappage de source d'événement pour envoyer les informations détaillées sur les lots ayant échoué à une file d'attente Amazon Simple Queue Service (Amazon SQS) ou à une rubrique Amazon Simple Notification Service (Amazon SNS).



Modèle de sourcing d'événements sur AWS

Amazon S3 stocke durablement tous les événements pour l'ensemble des microservices. Il s'agit de la seule source de référence pour les opérations de débogage, de restauration d'état d'application ou d'audit des modifications d'application. Il y a deux raisons principales pour lesquelles les

enregistrements peuvent être distribués plusieurs fois à votre application Kinesis Data Streams : nouvelles tentatives de l'application producteur et nouvelles tentatives de l'application consommateur. Votre application doit anticiper et gérer de façon appropriée le traitement des enregistrements plusieurs fois.

Gestion de la configuration

Dans une architecture de microservices classique comportant des dizaines de services différents, chaque service doit avoir accès à plusieurs services en aval et composants d'infrastructure qui exposent les données au service. Il peut s'agir de files d'attente de messages, de bases de données et d'autres microservices. L'un des principaux défis consiste à configurer chaque service de manière cohérente afin de fournir des informations sur la connexion aux services et à l'infrastructure en aval. En outre, la configuration doit également contenir des informations sur l'environnement dans lequel le service fonctionne, et le redémarrage de l'application pour utiliser de nouvelles données de configuration ne devrait pas être nécessaire.

Le troisième principe des modèles d'application à douze facteurs aborde ce sujet : « L'application à douze facteurs stocke la configuration dans des variables d'environnement (souvent appelées var. env. ou env.). ». Pour Amazon ECS, les variables d'environnement peuvent être transmises au conteneur à l'aide du paramètre de définition du conteneur d'environnement qui fait correspondre l'option `--env` à l'exécution Docker. Les variables d'environnement peuvent être transmises en bloc à vos conteneurs en utilisant le paramètre de définition du conteneur `environmentFiles` pour répertorier un ou plusieurs fichiers contenant les variables d'environnement. Le fichier doit être hébergé dans Amazon S3. Dans AWS Lambda, l'exécution met les variables d'environnement à la disposition de votre code et définit les variables d'environnement supplémentaires qui contiennent des informations sur la fonction et la demande d'appel. Pour Amazon EKS, vous pouvez définir des variables d'environnement dans le champ env du manifeste de configuration du pod correspondant. Utiliser une ConfigMap constitue une autre façon d'utiliser des variables d'environnement.

Communication asynchrone et messagerie légère

La communication dans les applications monolithiques traditionnelles est simple : une partie de l'application utilise des appels de méthode ou un mécanisme de distribution d'événements interne pour communiquer avec les autres parties. Si la même application est mise en œuvre avec des microservices découplés, la communication entre les différentes parties de l'application doit être déployée à l'aide de la communication réseau.

Communication basée sur REST

Le protocole HTTP/S est la solution la plus courante pour mettre en œuvre une communication synchronisée entre les microservices. Dans la plupart des cas, les API RESTful utilisent le protocole HTTP comme une couche de transport. Le type d'architecture REST s'appuie sur la communication sans état, des interfaces uniformes et des méthodes standard.

Avec API Gateway, vous pouvez créer une API qui agit en tant que porte d'entrée pour permettre aux applications d'accéder aux données, à la logique métier ou aux fonctionnalités de vos services backend. Les développeurs d'API peuvent créer des API qui accèdent à AWS ou à d'autres services web, ainsi qu'à des données stockées dans le cloud AWS. Un objet d'API défini avec le service API Gateway est un groupe de ressources et de méthodes.

Une ressource désigne un objet saisi dans le domaine d'une API et qui peut associer un modèle de données ou des relations avec d'autres ressources. Chaque ressource peut être configurée pour répondre à une ou plusieurs méthodes, c'est-à-dire des verbes HTTP standard tels que GET, POST ou PUT. Les API REST peuvent être déployées à différentes étapes, avec un système de gestion des versions ou clonées vers de nouvelles versions.

API Gateway gère toutes les tâches liées à l'acceptation et au traitement de plusieurs centaines de milliers d'appels d'API simultanés, notamment la gestion du trafic, le contrôle des autorisations et des accès, la surveillance et la gestion de la version de l'API.

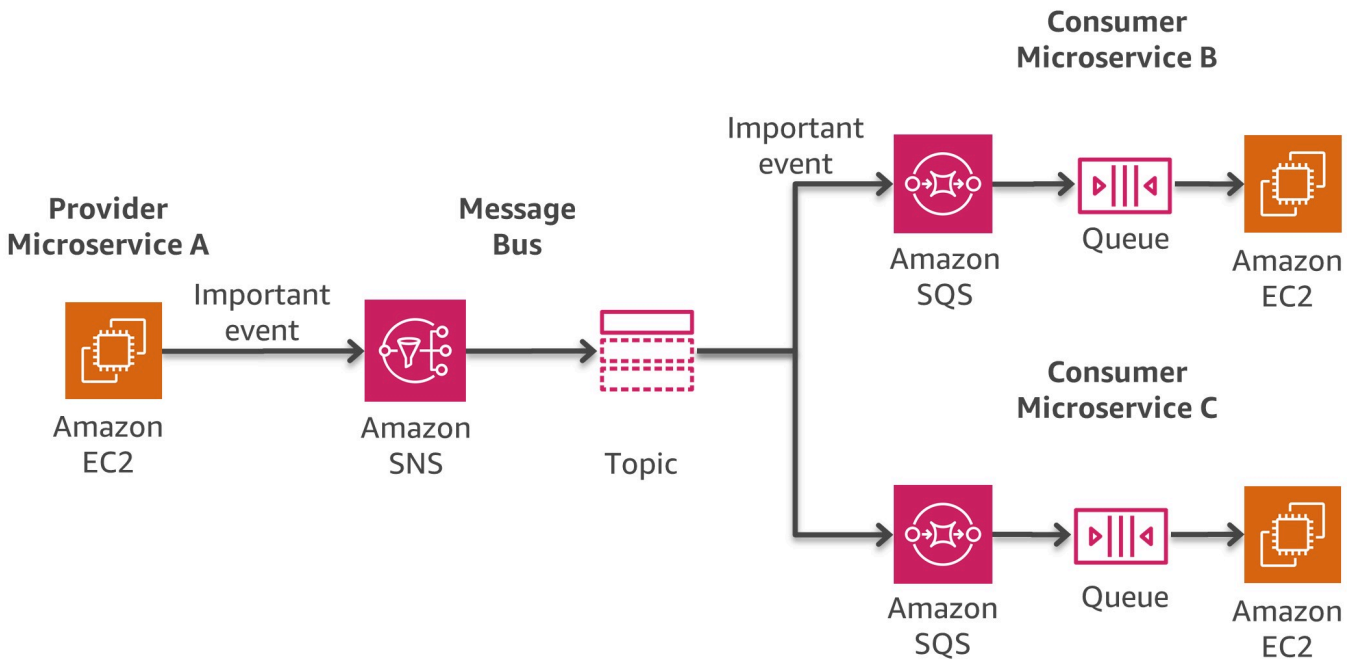
Messagerie asynchrone et transmission d'événement

La transmission de message est un autre modèle utilisé pour mettre en œuvre la communication entre microservices. Les services communiquent en échangeant des messages via une file d'attente. L'un des principaux avantages de ce style de communication est qu'il n'est pas nécessaire de disposer d'un service de découverte et que les services sont faiblement couplés.

Les systèmes synchrones sont étroitement couplés, ce qui signifie qu'un problème de dépendance synchrone en aval a un impact immédiat sur les appelants en amont. Les relances des appelants en amont peuvent rapidement se disperser et amplifier les problèmes.

En fonction d'exigences spécifiques, telles que les protocoles, AWS propose différents services facilitant la mise en œuvre de ce modèle. Une implémentation possible met en œuvre une combinaison d'Amazon Simple Queue Service (Amazon SQS) ou d'Amazon Simple Notification Service (Amazon SNS).

Les deux services fonctionnent en étroite collaboration : Amazon SNS permet aux applications d'envoyer des messages à plusieurs abonnés via un mécanisme Push. L'utilisation en conjonction d'Amazon SNS et d'Amazon SQS permet de remettre un message à plusieurs consommateurs. La figure suivante illustre l'intégration d'Amazon SNS et d'Amazon SQS.



Modèle de bus de messagerie sur AWS

Lorsque vous abonnez une file d'attente Amazon SQS à une rubrique SNS, vous pouvez publier un message dans la rubrique et Amazon SNS envoie un message à la file d'attente Amazon SQS associée. Le message contient l'objet et le message publié dans la rubrique, ainsi que les métadonnées au format JSON.

Amazon EventBridge est une autre option qui permet de créer des architectures pilotées par les événements avec des sources d'événements couvrant des applications internes, des applications SaaS tierces et des services AWS, à grande échelle. Service de bus d'événements entièrement géré, EventBridge reçoit des événements provenant de différentes sources, identifie une cible en fonction d'une règle de routage et fournit des données en temps quasi réel à cette cible, notamment AWS Lambda, Amazon SNS et Amazon Kinesis Streams, entre autres. Un événement entrant peut également être personnalisé, par transformateur d'entrée, avant la livraison.

Pour développer des applications pilotées par les événements nettement plus rapidement, les registres de schémas EventBridge collectent et organisent les schémas, y compris les schémas de tous les événements générés par les services AWS. Les clients peuvent également définir des schémas personnalisés ou utiliser une option de déduction de schéma pour découvrir les schémas automatiquement. Dans l'ensemble, toutefois, une valeur de latence relativement plus élevée pour la diffusion d'EventBridge constitue un compromis potentiel pour toutes ces fonctions. En outre, le débit et les quotas par défaut pour EventBridge peuvent nécessiter une augmentation, via une demande de support, en fonction du cas d'utilisation.

Une stratégie de déploiement différente repose sur Amazon MQ, qui peut être utilisé si les logiciels existants utilisent des API et des protocoles standard ouverts pour la messagerie, y compris JMS, NMS, AMQP, STOMP, MQTT et WebSocket. Amazon SQS expose une API personnalisée, ce qui signifie que si vous souhaitez migrer une application existante depuis un environnement sur site vers AWS par exemple, des modifications du code sont nécessaires. Avec Amazon MQ, cela n'est pas nécessaire dans de nombreux cas.

Amazon MQ gère l'administration et la maintenance d'ActiveMQ, un agent de messages open source courant. L'infrastructure sous-jacente est automatiquement configurée pour assurer une haute disponibilité et une durabilité des messages afin d'assurer la fiabilité de vos applications.

Orchestration et gestion d'états

Le caractère distribué des microservices rend difficile l'orchestration des flux de travail lorsque plusieurs microservices sont impliqués. Les développeurs peuvent être tentés d'ajouter un code d'orchestration directement dans leurs services. Cette méthode est à éviter, car elle renforce le couplage et empêche de remplacer rapidement des services individuels.

Vous pouvez utiliser AWS Step Functions pour construire des applications à partir de composants individuels qui exécutent chacun une fonction discrète. Step Functions fournit une machine d'état qui masque la complexité des opérations d'orchestration des services, telles que la gestion des erreurs, la sérialisation et la parallélisation. Cela vous permet de mettre à l'échelle et de modifier les applications rapidement, tout en évitant d'avoir à coordonner davantage le code interne des services.

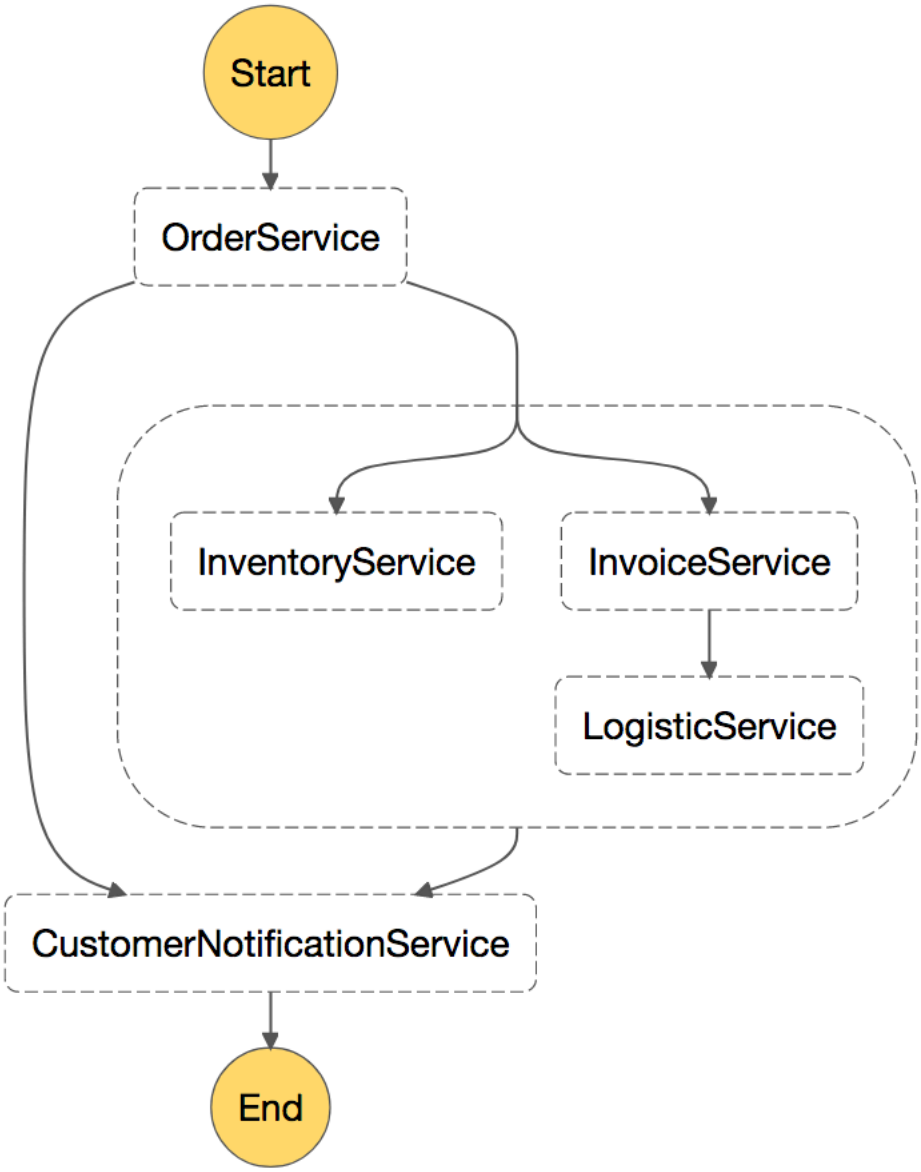
Step Functions est un moyen fiable de coordonner des composants et suivre les fonctions de votre application. Step Functions fournit une console graphique pour organiser et visualiser les composants de votre application en une série d'étapes. Cela facilite la création et l'exécution des services distribués.

Step Functions démarre automatiquement chaque étape, en fait le suivi et la relance en cas d'erreur. Votre application est donc exécutée dans l'ordre et comme prévu. Step Functions consigne l'état de chaque étape pour que vous puissiez diagnostiquer et résoudre rapidement les problèmes éventuels. Vous pouvez modifier et ajouter des étapes sans écrire de code, afin de faire évoluer votre application et d'innover plus rapidement.

Step Functions est intégré à la plateforme sans serveur d'AWS et prend en charge l'orchestration des fonctions Lambda ainsi que des applications basées sur des ressources de calcul, telles qu'Amazon EC2, Amazon EKS, Amazon ECS et des services supplémentaires tels qu'Amazon SageMaker et AWS Glue. Step Functions gère les opérations et l'infrastructure sous-jacente à votre place pour vous aider à assurer la disponibilité de votre application, quelle que soit l'échelle.

Pour créer des flux de travail, Step Functions utilise le langage des états d'Amazon. Les flux de travail peuvent contenir des étapes parallèles ou séquentielles, ainsi que des étapes de ramifications.

La figure suivante présente un exemple de flux de travail pour une architecture de microservices combinant des étapes parallèles et séquentielles. L'appel d'un tel flux de travail peut être effectué via l'API Step Functions ou avec API Gateway.



Exemple de flux de travail de microservices appelé par Step Functions

Surveillance distribuée

Une architecture de microservices se compose de différents composants distribués qui doivent être surveillés. Vous pouvez utiliser Amazon CloudWatch pour collecter et suivre des métriques, centraliser et surveiller des fichiers journaux, régler des alarmes et réagir automatiquement aux modifications apportées à votre environnement AWS. CloudWatch peut surveiller les ressources AWS de la même façon que les instances Amazon EC2, les tables DynamoDB et les instances

de base de données Amazon RDS, ainsi que les métriques personnalisées générées par vos applications et services, et tous les fichiers journaux générés par vos applications.

Surveillance

Vous pouvez utiliser CloudWatch pour gagner une visibilité à l'échelle du système sur l'utilisation des ressources, la performance de l'application et la santé opérationnelle. CloudWatch fournit une solution de surveillance fiable, évolutive et flexible que vous pouvez commencer à utiliser en seulement quelques minutes. Vous n'avez plus besoin de régler, gérer et redimensionner vos propres systèmes et infrastructures de surveillance. Dans une architecture de microservices, la capacité de surveillance des métriques personnalisées à l'aide de CloudWatch est un avantage supplémentaire, car les développeurs peuvent décider quelles métriques doivent être collectées pour chaque service. En outre, la mise à l'échelle dynamique peut être implémentée en fonction de métriques personnalisées.

Outre Amazon CloudWatch, vous pouvez également utiliser CloudWatch Container Insights pour collecter, regrouper et résumer les métriques et les journaux provenant de vos applications et microservices conteneurisés. CloudWatch Container Insights collecte automatiquement des métriques pour de nombreuses ressources, telles que le processeur, la mémoire, le disque et le réseau, et les regroupe sous forme de métriques CloudWatch au niveau du cluster, du nœud, du pod, de la tâche et du service. En utilisant CloudWatch Container Insights, vous pouvez accéder aux métriques du tableau de bord CloudWatch Container Insights. Le service fournit également des informations de diagnostic (par exemple sur les échecs de redémarrage des conteneurs) pour vous aider à isoler les problèmes et à les résoudre rapidement. Vous pouvez également définir des alarmes CloudWatch sur les métriques collectées par Container Insights.

Container Insights est disponible pour les plateformes Amazon ECS, Amazon EKS et Kubernetes sur Amazon EC2. Le support Amazon ECS inclut la prise en charge de Fargate.

L'utilisation de Prometheus est une autre option courante, en particulier pour Amazon EKS. Prometheus est une boîte à outils open source de surveillance et d'alerte souvent utilisée avec Grafanapour afficher les métriques collectées. Plusieurs composants Kubernetes stockent des métriques dans `/metrics` et Prometheus peut les capturer à intervalles réguliers.

Amazon Managed Service for Prometheus (AMP) est un service de surveillance compatible avec Prometheus qui vous permet de surveiller les applications conteneurisées à grande échelle. Avec AMP, vous pouvez utiliser le langage de requête open source Prometheus (PromQL) pour surveiller les performances des charges de travail conteneurisées, sans avoir à gérer l'infrastructure

sous-jacente nécessaire pour la gestion, l'ingestion, le stockage et la requête des métriques opérationnelles. Vous pouvez collecter les métriques de Prometheus à partir des environnements Amazon EKS et Amazon ECS, à l'aide d'AWS Distro for OpenTelemetry ou les serveurs Prometheus en tant qu'agents de recouvrement.

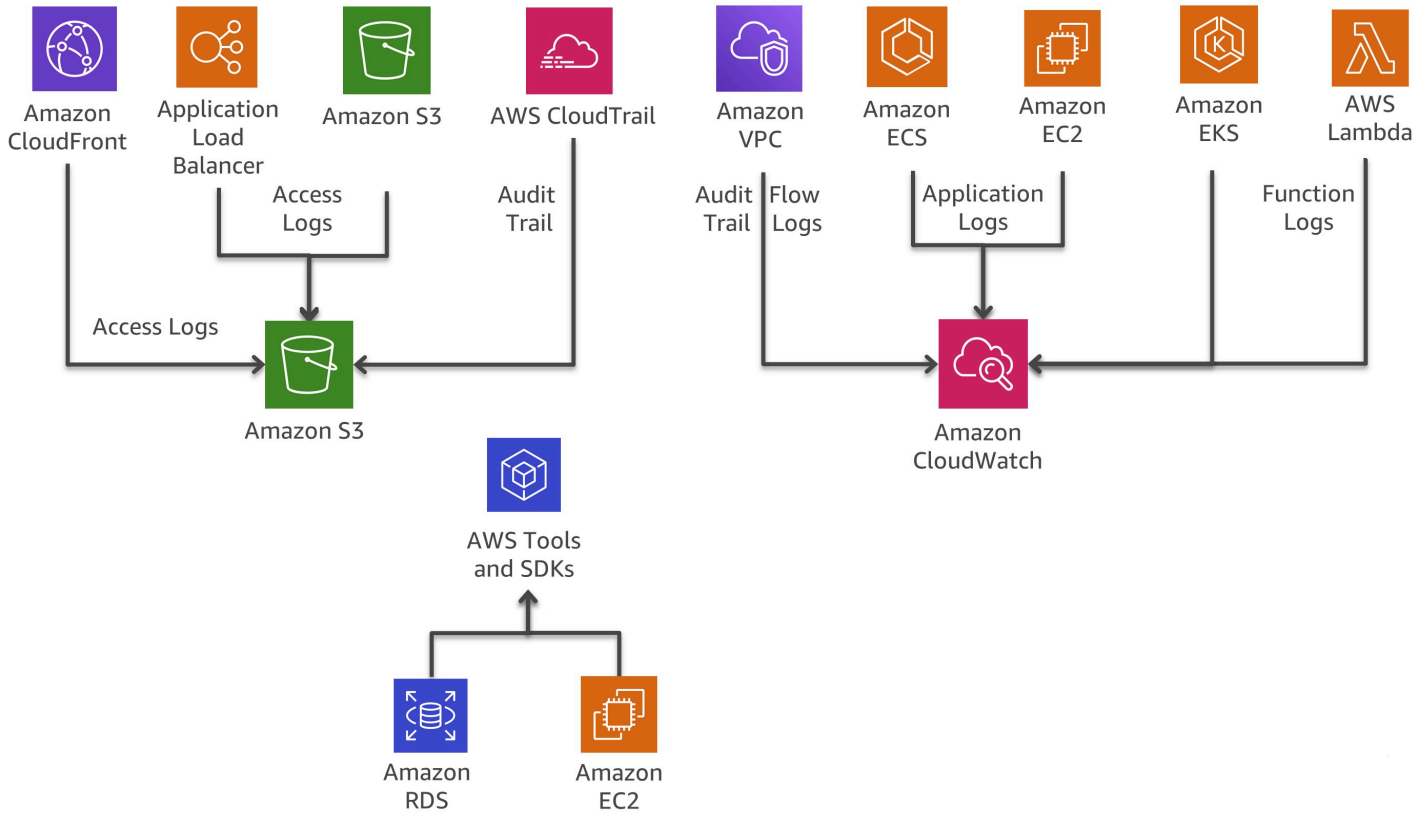
AMP est souvent utilisé en combinaison avec Amazon Managed Service for Grafana (AMG). AMG facilite l'interrogation, la visualisation, les alertes et la compréhension de vos métriques, quel que soit l'endroit où elles sont stockées. Avec AMG, vous pouvez analyser vos métriques, vos journaux et vos traces sans avoir à allouer des serveurs, à configurer et à mettre à jour des logiciels, ou à effectuer les tâches lourdes liées à la sécurisation et à la mise à l'échelle de Grafana en production.

Centralisation des journaux

Une journalisation cohérente est essentielle pour l'identification et la résolution des problèmes. Les microservices permettent aux équipes de publier beaucoup plus de versions qu'auparavant et encouragent les équipes d'ingénieurs à expérimenter avec de nouvelles fonctions dans les environnements de production. La compréhension de l'impact sur le client est essentielle pour améliorer progressivement une application.

Par défaut, la plupart des services AWS centralisent leurs fichiers journaux. Les destinations principales pour les fichiers journaux sur AWS sont Amazon S3 et Amazon CloudWatch Logs. Pour les applications qui s'exécutent sur des instances Amazon EC2, un démon est disponible pour envoyer les fichiers journaux vers CloudWatch Logs. Les fonctions Lambda envoient nativement leurs données de journalisation à CloudWatch Logs. Amazon ECS inclut la prise en charge du pilote de journal awslogs, qui permet la centralisation des journaux de conteneurs dans CloudWatch Logs. Pour Amazon EKS, Fluent Bit ou Fluentd peut transférer les journaux d'instances individuelles du cluster vers le service CloudWatch Logs de journalisation centralisée, où ces journaux sont regroupés pour établir des rapports de haut niveau à l'aide d'Amazon OpenSearch Service et de Kibana. En raison de son faible encombrement et de ses avantages en termes de performances, Fluent Bit est recommandé plutôt que FluentD.

La figure suivante illustre les fonctionnalités de journalisation de certains des services. Les équipes sont ensuite en mesure de rechercher et d'analyser ces journaux à l'aide d'outils tels qu'Amazon OpenSearch Service et Kibana. Amazon Athena peut être utilisé pour exécuter une requête ponctuelle sur des fichiers journaux centralisés dans Amazon S3.



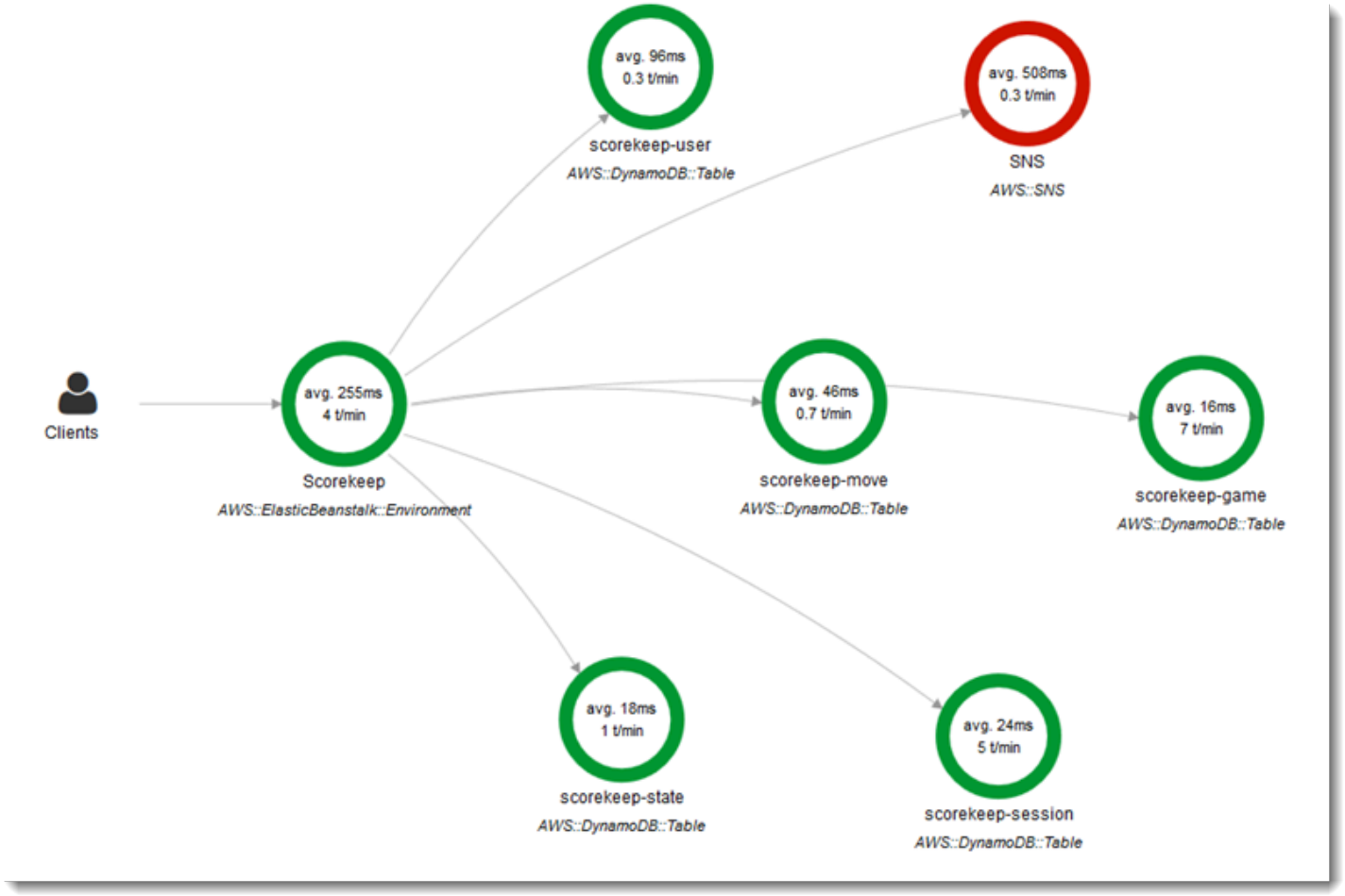
Capacités de journalisation des services AWS

Suivi distribué

Dans de nombreux cas, un ensemble de microservices fonctionne conjointement pour gérer une demande. Imaginez un système complexe composé de dizaines de microservices dans lequel une erreur se produit dans l'un des services de la chaîne d'appel. Même si chaque microservice assure correctement sa journalisation et que les journaux sont regroupés dans un système central, il peut être difficile de trouver tous les messages de journal pertinents.

L'idée centrale derrière AWS X-Ray est l'utilisation d'identifiants de corrélation, qui sont des identifiants uniques attachés à toutes les requêtes et tous les messages liés à une chaîne d'événements spécifique. Les identifiants de suivi sont ajoutés aux requêtes HTTP dans des en-têtes de suivi spécifiques nommés `X-Amzn-Trace-Id` lorsque la requête atteint le premier service intégré à X-Ray (par exemple, un équilibreur Application Load Balancer ou API Gateway). Ces identifiants sont également inclus dans la réponse. Avec le kit SDK X-Ray, n'importe quel microservice peut lire cet en-tête, l'ajouter ou le mettre à jour.

X-Ray fonctionne avec Amazon EC2, Amazon ECS, Lambda et AWS Elastic Beanstalk. Vous pouvez utiliser X-Ray avec les applications écrites en Java, Node.js et .NET qui sont déployées sur ces services.



Cartographie du service AWS X-Ray

Epsagon est un SaaS entièrement géré qui inclut le suivi de tous les services AWS, des API tierces (via des appels HTTP) et d'autres services courants tels que Redis, Kafka et Elastic. Le service Epsagon inclut des fonctionnalités de surveillance, des alertes sur les services les plus courants et une visibilité de la charge utile pour chaque appel passé par votre code.

AWS Distro for OpenTelemetry est une distribution sécurisée, prête pour la production et soutenue par AWS du projet OpenTelemetry. Partie intégrante de la Cloud Native Computing Foundation, AWS Distro for OpenTelemetry fournit des API, des bibliothèques et des agents open source pour collecter des traces et des métriques distribuées pour la surveillance des applications. Avec AWS Distro for OpenTelemetry, vous pouvez instrumenter vos applications une seule fois pour envoyer des métriques et des traces corrélées à plusieurs solutions de surveillance AWS et partenaires.

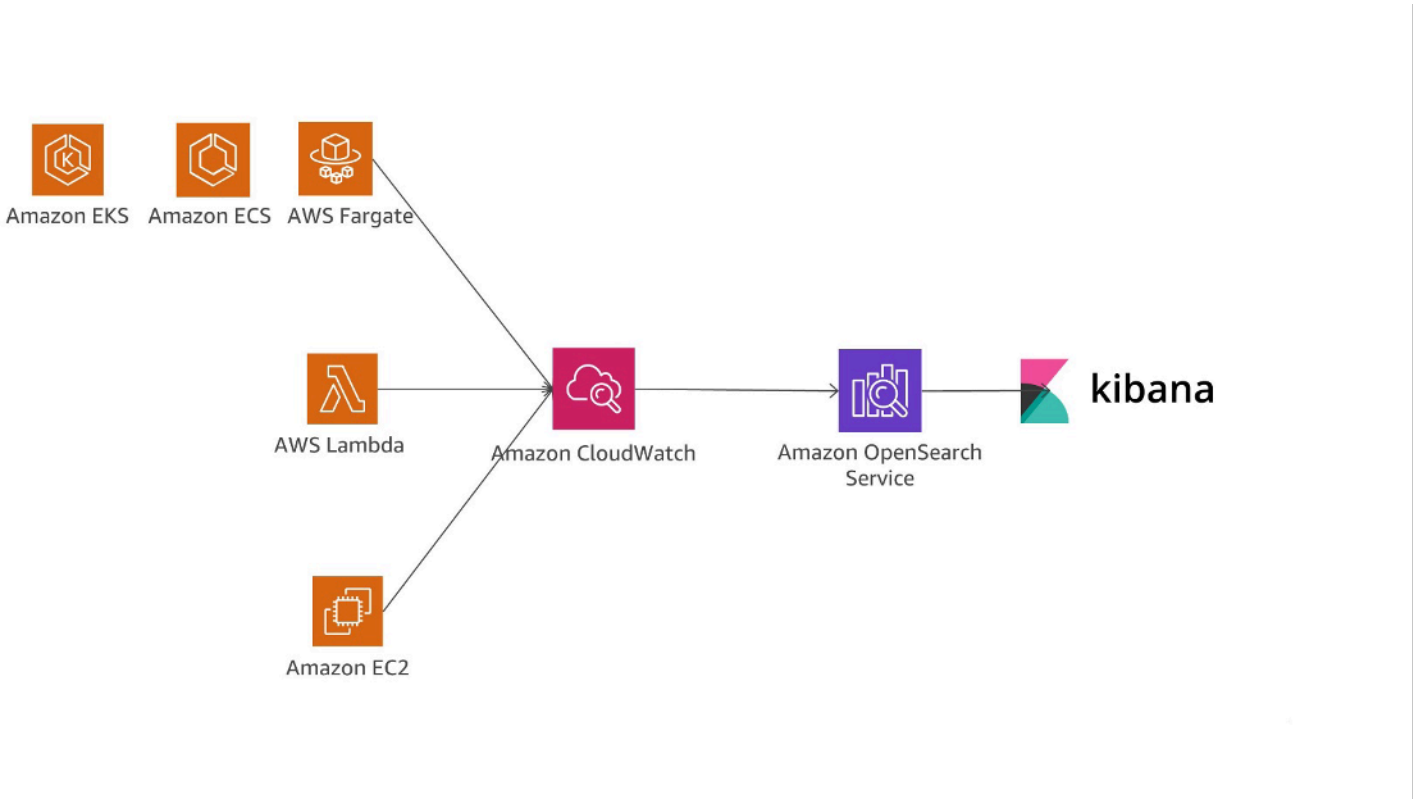
Utilisez des agents d'auto-instrumentation pour collecter des traces sans modifier votre code. AWS Distro for OpenTelemetry collecte également les métadonnées de vos ressources AWS et de vos services managés, afin de corréler les données de performance des applications avec les données d'infrastructure sous-jacentes, et de réduire le délai moyen de résolution des problèmes. Utilisez AWS Distro for OpenTelemetry pour instrumenter vos applications exécutées sur Amazon EC2, Amazon ECS, Amazon EKS sur Amazon EC2, Fargate et AWS Lambda, ainsi que sur site.

Options pour l'analyse de journaux sur AWS

La recherche, l'analyse et la visualisation des données de journal sont autant d'activités cruciales pour la compréhension des systèmes distribués. Amazon CloudWatch Logs Insights vous permet d'explorer, d'analyser et de visualiser instantanément des journaux. Il vous permet de résoudre les problèmes opérationnels auxquels vous faites face. Utiliser Amazon OpenSearch Service avec Kibana constitue une autre option pour l'analyse des fichiers journaux.

Amazon OpenSearch Service peut être utilisé pour la recherche en texte intégral, la recherche structurée, l'analyse ou ces trois activités en combinaison. Kibana est un plugin de visualisation de données open source qui s'intègre parfaitement à Amazon OpenSearch Service.

La figure suivante illustre l'analyse des journaux avec Amazon OpenSearch Service et Kibana. CloudWatch Logs peut être configuré pour diffuser les entrées de journaux vers Amazon OpenSearch Service en temps quasi réel via un abonnement CloudWatch Logs. Kibana visualise les données et offre une interface de recherche pratique pour les magasins de données dans Amazon OpenSearch Service. Cette solution peut être utilisée en association avec un logiciel comme ElastAlert pour mettre en œuvre un système d'alarme afin d'envoyer des notifications SNS et des e-mails, créer des tickets JIRA, etc., si des anomalies, des pics, ou d'autres tendances intéressantes sont détectés dans les données.



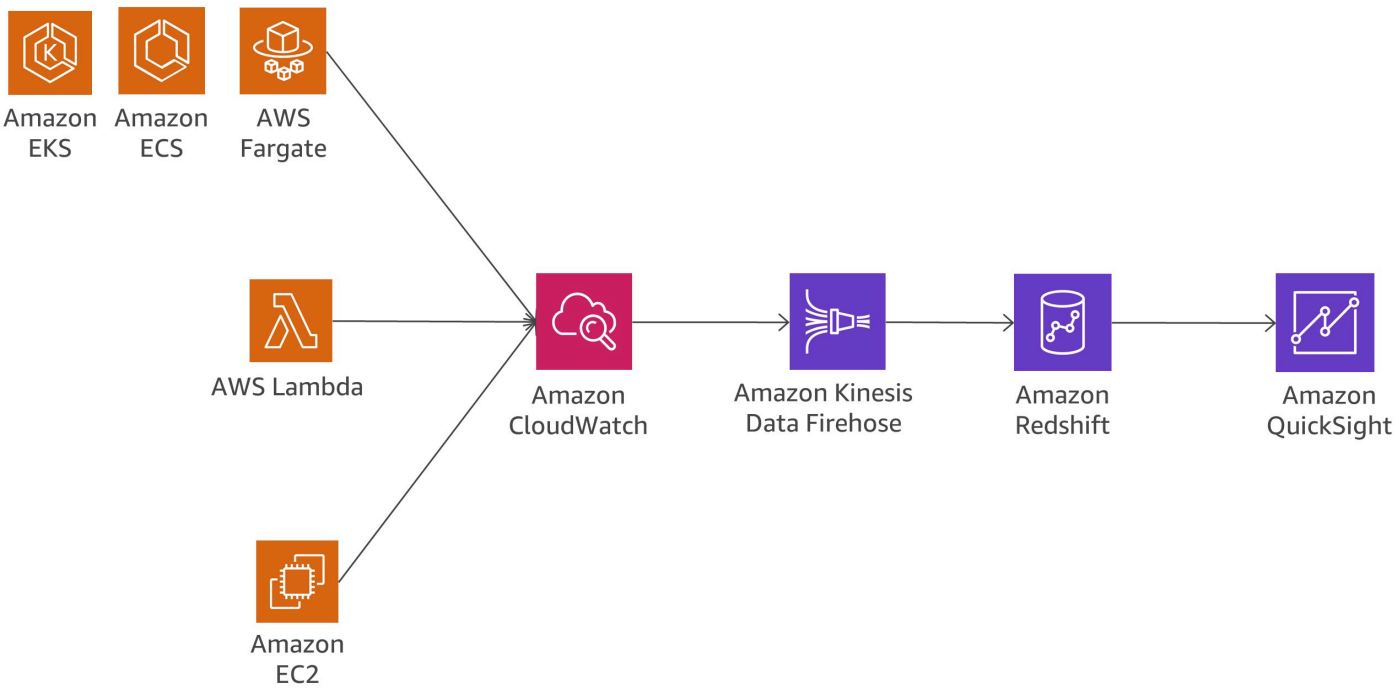
Analyse de journal avec Amazon OpenSearch Service et Kibana

Une autre option pour l'analyse des fichiers journaux consiste à utiliser Amazon Redshift avec Amazon QuickSight.

QuickSight peut être facilement connecté aux services de données AWS, notamment Amazon Redshift, Amazon RDS, Amazon Aurora, Amazon EMR, DynamoDB, Amazon S3 et Amazon Kinesis.

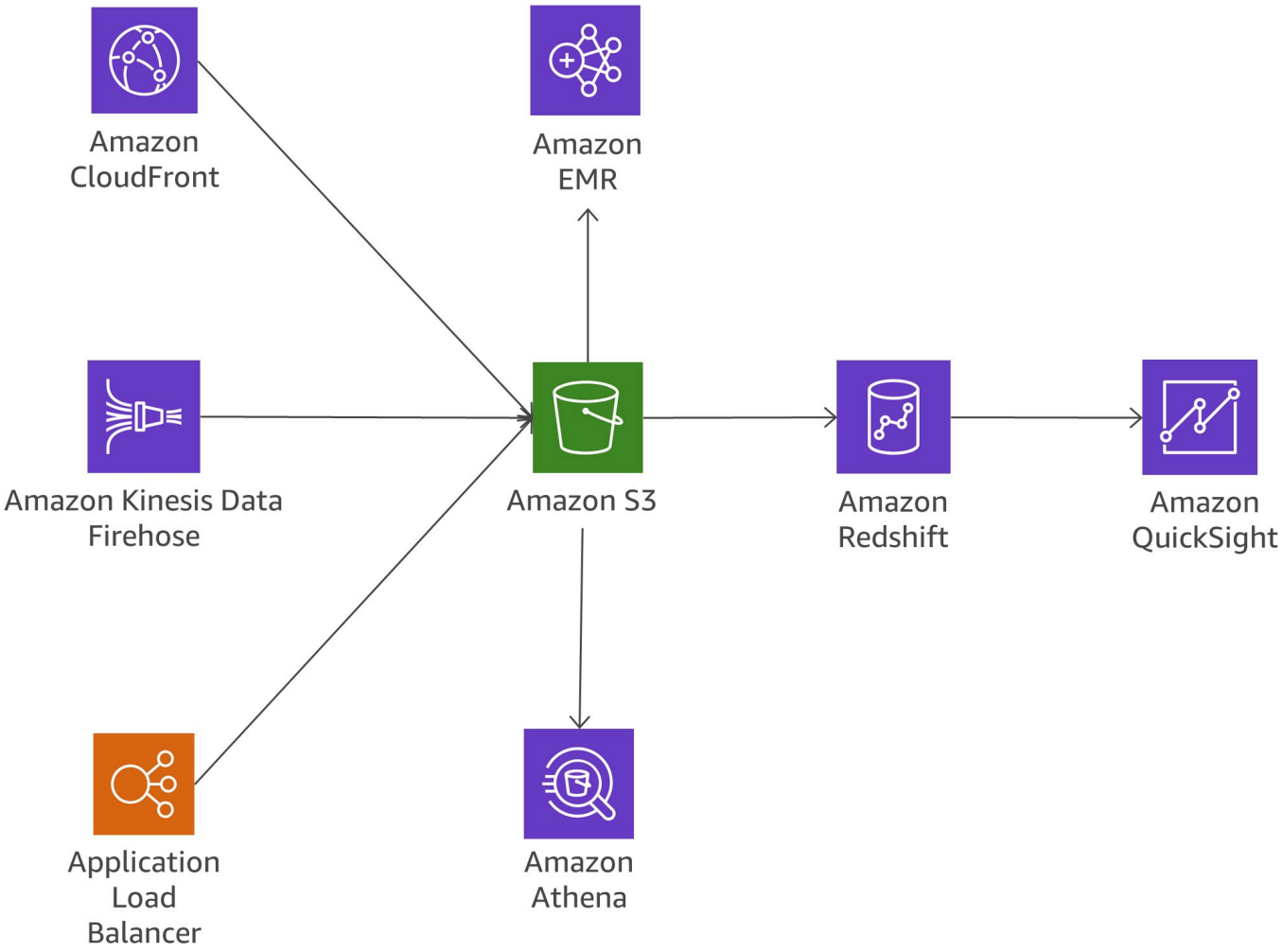
CloudWatch Logs peut agir en tant que magasin centralisé pour les données de journalisation et, en plus de seulement stocker les données, il est possible de diffuser des entrées de journaux vers Amazon Kinesis Data Firehose.

La figure suivante illustre un scénario où les entrées de journal sont diffusées en streaming à partir de différentes sources vers Amazon Redshift à l'aide de CloudWatch Logs et Kinesis Data Firehose. Amazon QuickSight utilise les données stockées dans Amazon Redshift à des fins d'analyse, de rapports et de visualisation.



Analyse de journal avec Amazon Redshift et Amazon QuickSight

La figure suivante illustre un scénario d'analyse de journal sur Amazon S3. Lorsque les journaux sont stockés dans des compartiments Amazon S3, les données de journaux peuvent être chargées dans différents services de données AWS, par exemple, Amazon Redshift ou Amazon EMR, pour analyser les données stockées dans le flux de journaux et rechercher des anomalies.



Analyse de journal sur Amazon S3

Bavardage

Quand les applications monolithiques sont divisées en petits microservices, les frais de communication augmentent, car les microservices doivent communiquer entre eux. Dans bien des implémentations, le service REST sur HTTP est utilisé, car il s'agit d'un protocole de communication léger. Cependant, il peut être confronté à des problèmes face à de grands volumes de messages. Dans certains cas, vous pourriez envisager d'utiliser des services de consolidation qui échangent beaucoup de messages. Si vous vous retrouvez dans une situation où vous consolidez de plus en plus de services dans le seul but de limiter les communications, vous devez vérifier les domaines qui posent problème et votre modèle de domaine.

Protocoles

Différents protocoles possibles sont présentés plus haut dans ce livre blanc, dans la section the section called “Communication asynchrone et messagerie légère”. Pour les microservices, il est fréquent d'utiliser des protocoles simples tels que HTTP. Les messages échangés par les services peuvent être encodés de différentes façons, par exemple, dans un format compréhensible par les utilisateurs tels que JSON ou YAML, ou dans un format binaire efficace comme Avro ou Protocol Buffers.

Mise en cache

Les caches sont un excellent moyen de réduire la latence et le volume de communication des architectures de microservices. Plusieurs couches de mise en cache sont possibles en fonction du cas d'utilisation réel et des goulots d'étranglement. De nombreuses applications à microservices s'exécutant sur AWS utilisent ElastiCache pour réduire le volume des appels vers d'autres microservices en mettant en cache les résultats localement. API Gateway fournit une couche de mise en cache intégrée afin de réduire la charge sur les serveurs backend. De plus, la mise en cache est utile pour réduire la charge de la couche de persistance des données. Le défi, pour tout mécanisme de mise en cache, consiste à trouver le juste équilibre entre un bon taux d'accès au cache, et la rapidité et la cohérence des données.

Audit

Dans les architectures de microservices, qui peuvent comporter potentiellement des centaines de services distribués, un autre défi à consiste à assurer la visibilité des actions d'utilisateur sur chaque service et à être en mesure d'obtenir une vision globale claire de tous les services au sein de l'organisation. Pour vous aider à appliquer des politiques de sécurité, il est important d'auditer à la fois l'accès aux ressources et les activités conduisant à des modifications du système.

Les modifications doivent être suivies au niveau des services individuels, mais aussi dans les services s’exécutant sur le système dans son ensemble. En général, les modifications sont fréquentes dans les architectures de microservices, ce qui fait de l’audit des modifications une activité plus qu’essentielle. Cette section examine les principaux services et fonctions AWS qui peuvent vous aider à auditer votre architecture de microservices.

Piste d'audit

AWS CloudTrail est un outil utile pour le suivi des modifications des microservices, car il permet à tous les appels d'API effectués sur le AWS Cloud d'être journalisés et envoyés vers CloudWatch Logs en temps réel ou vers Amazon S3 en quelques minutes.

Toutes les actions d'utilisateur et de systèmes automatisés deviennent accessibles et peuvent être analysés pour détecter un comportement inattendu ou des violations de politique d'entreprise ou pour soutenir les activités de débogage. Les informations enregistrées comprennent un horodatage, des informations sur l'utilisateur et le compte, le service qui a été appelé, l'action de service demandée, l'adresse IP de l'appelant, ainsi que les paramètres de requête et les éléments de la réponse.

CloudTrail permet de définir plusieurs journaux d'activité pour un même compte, ce qui permet à différentes parties prenantes, par exemple les administrateurs de sécurité, les développeurs de logiciels ou les auditeurs informatiques, de créer et de gérer leur propre journal d'activité. Si les équipes de microservices possèdent différents comptes AWS, il est possible de regrouper les journaux d'activité dans un même compartiment S3.

Le stockage des pistes d'audit dans CloudWatch présente plusieurs avantages : les données des pistes d'audit sont capturées en temps réel, et la redirection des informations vers Amazon OpenSearch Service pour la recherche et la visualisation est facile. Vous pouvez configurer CloudTrail pour vous connecter à Amazon S3 et CloudWatch Logs.

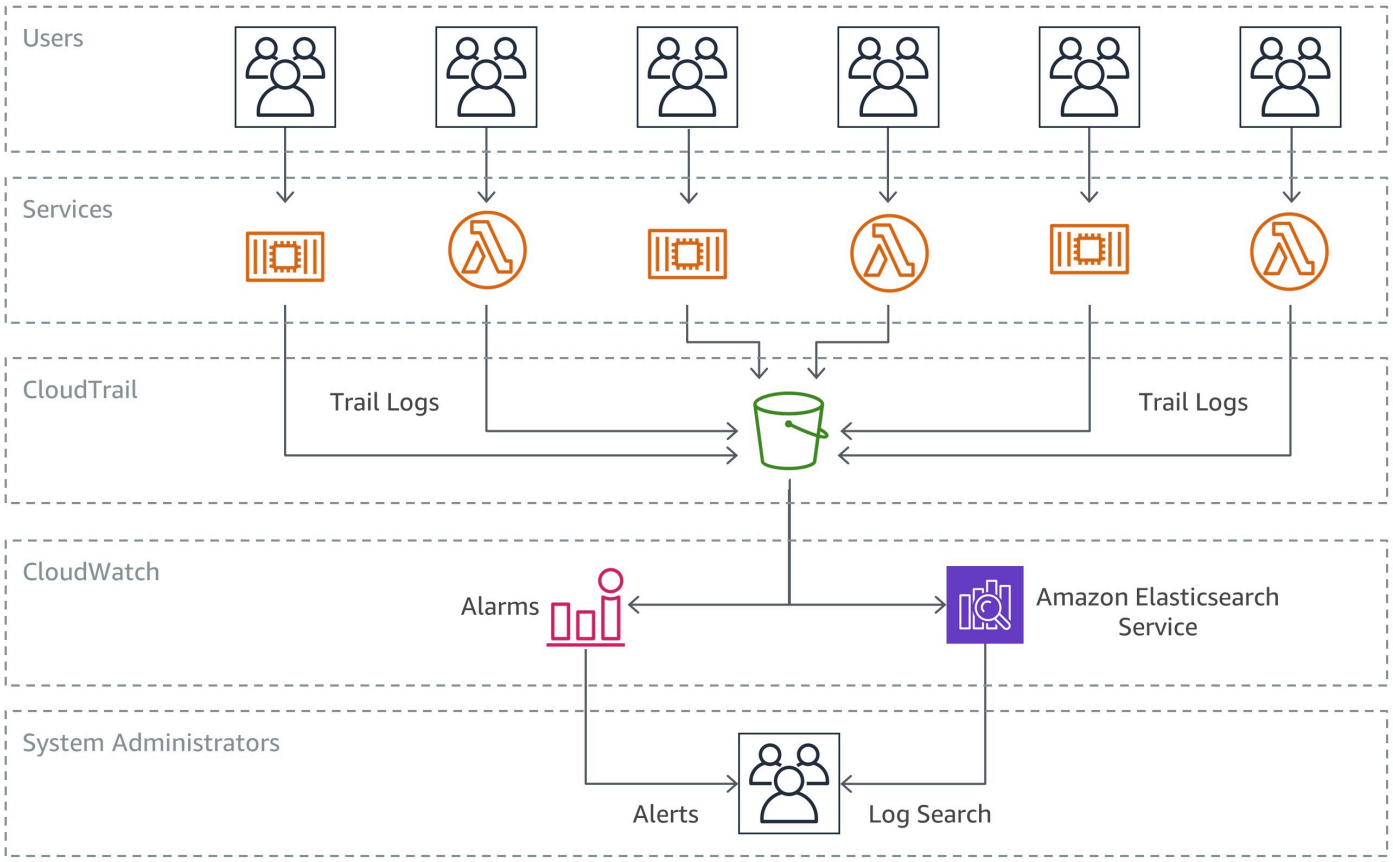
Actions en temps réel et événements

Certaines modifications dans les architectures de systèmes doivent faire l'objet d'une réponse rapide, et être suivies d'une action corrective ou de procédures de gouvernance spécifiques visant à autoriser la modification. L'intégration d'Amazon CloudWatch Events avec CloudTrail permet de générer des événements pour tous les appels d'API en mutation parmi tous les services AWS. Il est également possible de définir des événements personnalisés ou de générer des événements selon un calendrier fixe.

Lorsqu'un événement est déclenché et correspond à une règle définie, un groupe prédéfini de personnes de votre organisation peut être immédiatement averti, afin que ces personnes puissent prendre les mesures appropriées. Si l’action requise peut être automatisée, alors la règle peut automatiquement déclencher un flux de travail intégré ou appeler une fonction Lambda permettant de résoudre le problème.

La figure suivante illustre un environnement dans lequel CloudTrail et CloudWatch Events fonctionnent ensemble pour répondre aux exigences en matière d'audit et de correction au sein

d'une architecture de microservices. Tous les microservices sont suivis par CloudTrail et la piste d'audit est stockée dans un compartiment Amazon S3. CloudWatch Events prend connaissance des changements opérationnels au fur et à mesure qu'ils surviennent. CloudWatch Events répond à ces changements opérationnels et, le cas échéant, prend des mesures correctives en envoyant des messages pour répondre à l'environnement, en activant des fonctions, en procédant à des modifications et en capturant des informations de statut. CloudWatch Events repose sur CloudTrail et déclenche des alertes lorsqu'une modification spécifique est apportée à votre architecture.



Audit et correction

Inventaire des ressources et gestion des modifications

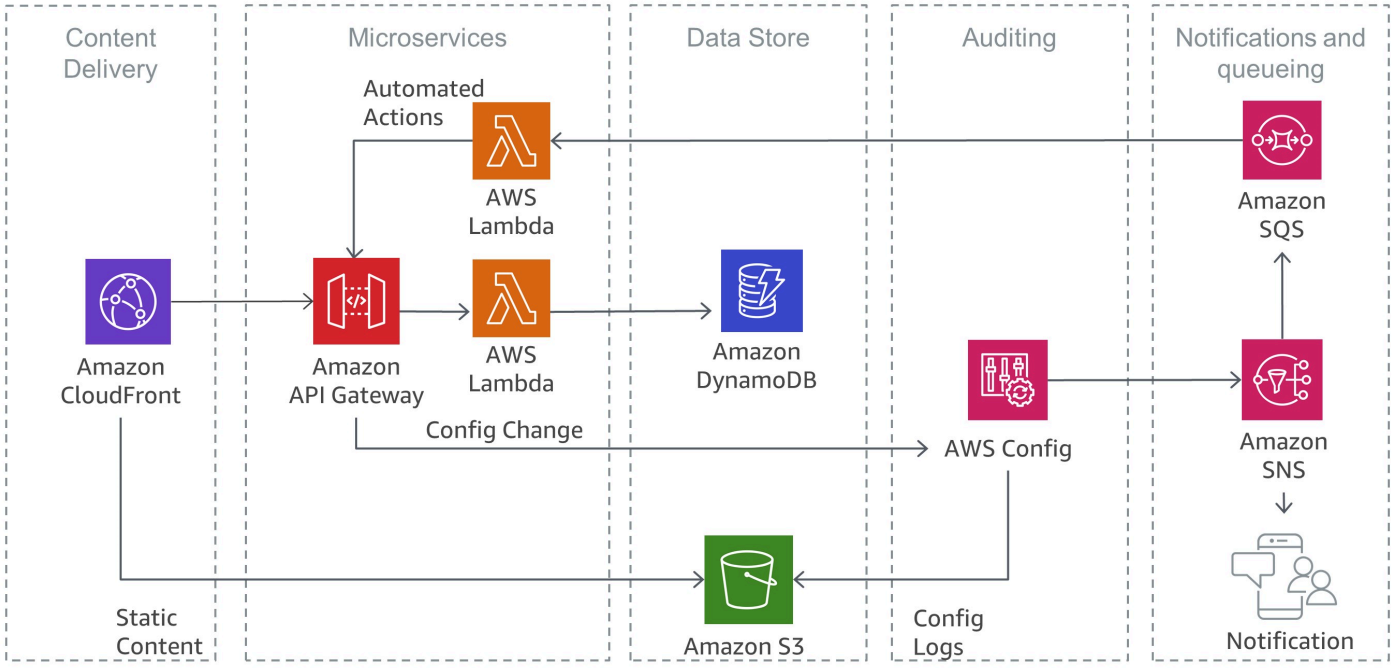
Afin de garder le contrôle des configurations d'infrastructure en évolution rapide dans un environnement de développement agile, il est essentiel d'adopter une approche plus automatisée et gérée pour l'audit et le contrôle de votre architecture.

Même si CloudTrail et CloudWatch Events s'avèrent être des composants importants pour le suivi et la correction des modifications d'infrastructure dans les microservices, les règles AWS Config

permettent à une entreprise de définir des politiques de sécurité assorties de règles spécifiques, visant à détecter, suivre et signaler automatiquement les violations de politiques.

L'exemple suivant illustre la façon dont il est possible de détecter, notifier et automatiquement réagir face aux modifications de configuration non conformes au sein de votre architecture de microservices. Un membre de l'équipe de développement apporte une modification à API Gateway pour permettre à un microservice d'autoriser le point de terminaison à accepter le trafic HTTP entrant, plutôt que d'autoriser uniquement les requêtes HTTPS.

Étant donné que l'organisation a préalablement identifié cette situation comme étant un problème de conformité à la sécurité, une règle AWS Config assure d'ores et déjà le suivi de la situation en question. Cette règle identifie la modification comme une violation de la sécurité et effectue deux actions : dans un compartiment Amazon S3, elle crée un journal relatif au changement détecté à des fins d'audit, et elle crée une notification SNS. Dans notre scénario, le service Amazon SNS est utilisé pour deux raisons : pour envoyer un e-mail à un groupe spécifié et l'informer de la violation de la sécurité, et pour ajouter un message à une file d'attente SQS. Ensuite, le message est récupéré, et l'état conforme est restauré en modifiant la configuration d'API Gateway.



Détection des violations de sécurité avec AWS Config

Conclusion

L'architecture de microservices est une approche de conception distribuée qui vise à surmonter les limitations des architectures monolithiques traditionnelles. Les microservices permettent de mettre à l'échelle des applications et des organisations, tout en améliorant les cycles de développement. Cependant, ils présentent également quelques défis susceptibles d'augmenter la complexité de l'architecture et la charge opérationnelle.

AWS propose une large gamme de services managés qui peuvent aider les équipes produit à créer des architectures de microservices et à minimiser la complexité architecturale et opérationnelle. Ce livre blanc présente les services AWS pertinents et explique comment implémenter des modèles classiques, tels que la découverte de services ou les sources d'événements, en mode natif avec les services AWS.

Ressources

- Centre d'architecture AWS
- Livres blancs AWS
- AWS Architecture Monthly
- Blog sur l'architecture AWS
- Vidéos This Is My Architecture
- AWS Answers
- Documentation AWS

Historique du document et contributeurs

Historique 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	Intégration d'Amazon EventBridge, AWS OpenTelemetry, AMP, AMG, Container Insights, modifications mineures du texte.	9 novembre 2021
Mises à jour mineures	Ajustement de la mise en page	30 avril 2021
Mises à jour mineures	Modifications mineures du texte.	1 août 2019
Livre blanc mis à jour	Intégration d'Amazon EKS, AWS Fargate, Amazon MQ, AWS PrivateLink, AWS App Mesh et AWS Cloud Map	1 juin 2019
Livre blanc mis à jour	Intégration d'AWS Step Functions, d'AWS X-Ray et des flux d'évènements ECS.	1 septembre 2017
Publication initiale	Implémentation des microservices sur AWS publiée.	1 décembre 2016

> **Note**
>
> Pour vous abonner aux mises à jour RSS, vous devez activer un plug-in RSS pour le navigateur que vous utilisez.

Participants

Les personnes et organisations suivantes ont participé à la préparation du présent document :

- Sascha Möllering, Architecture de solutions, AWS
- Christian Müller, Architecture de solutions, AWS
- Matthias Jung, Architecture de solutions, AWS
- Peter Dalbhanjan, Architecture de solution, AWS
- Peter Chapman, Architecture de solution, AWS
- Christoph Kassen, Architecture de solution, AWS
- Umair Ishaq, Architecture de solutions, AWS
- Rajiv Kumar, Architecture de solutions, AWS

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.