AWS Livre blanc

Implémentation de microservices sur AWS

Implémentation de microservices sur AWS: AWS Livre blanc

Table of Contents

Implémentation de microservices sur AWS

Date de publication : 31 juillet 2023 (Historique du document)

Les microservices offrent une approche rationalisée du développement logiciel qui accélère le déploiement, encourage l'innovation, améliore la maintenabilité et améliore l'évolutivité. Cette méthode repose sur de petits services faiblement couplés qui communiquent par le biais de réseaux bien définis APIs et gérés par des équipes autonomes. L'adoption de microservices offre des avantages, tels qu'une évolutivité, une résilience, une flexibilité améliorées et des cycles de développement plus rapides.

Ce livre blanc explore trois modèles de microservices courants : piloté par API, piloté par les événements et streaming de données. Nous donnons un aperçu de chaque approche, décrivons les principales fonctionnalités des microservices, abordons les défis liés à leur développement et illustrons comment Amazon Web Services (AWS) peut aider les équipes chargées des applications à surmonter ces obstacles.

Compte tenu de la nature complexe de sujets tels que le stockage des données, la communication asynchrone et la découverte de services, nous vous encourageons à évaluer les besoins et les cas d'utilisation spécifiques de votre application en fonction des conseils fournis lors de la prise de décisions architecturales.

Introduction

Les architectures de microservices combinent des concepts éprouvés et éprouvés issus de différents domaines, tels que :

- Développement logiciel agile
- Architectures orientées services
- Conception axée sur les API
- ContinuIntegration/Continuous Delivery (CI/CD)

Les microservices intègrent souvent des modèles de conception issus de l'application Twelve-Factor.

Bien que les microservices offrent de nombreux avantages, il est essentiel d'évaluer les exigences uniques de votre cas d'utilisation et les coûts associés. Une architecture monolithique ou des approches alternatives peuvent être plus appropriées dans certains cas. Le choix entre les

microservices ou les monolithes doit être fait sur la case-by-case base de facteurs tels que l'échelle, la complexité et les cas d'utilisation spécifiques.

Nous explorons d'abord une architecture de microservices hautement évolutive et tolérante aux pannes (interface utilisateur, implémentation de microservices, magasin de données) et montrons comment l'intégrer à l'aide de technologies de conteneurs. AWS Nous suggérons ensuite AWS des services pour implémenter une architecture typique de microservices sans serveur, afin de réduire la complexité opérationnelle.

Le mode Serverless se caractérise par les principes suivants :

- Aucune infrastructure à fournir ou à gérer
- Mise à l'échelle automatique par unité de consommation
- Modèle de facturation « Pay for value »
- Disponibilité et tolérance aux pannes intégrées
- Architecture axée sur les événements (EDA)

Enfin, nous examinons l'ensemble du système et abordons les aspects interservices d'une architecture de microservices, tels que la surveillance distribuée, la journalisation, le suivi, l'audit, la cohérence des données et la communication asynchrone.

Ce document se concentre sur les charges de travail exécutées dans le AWS Cloud, à l'exception des scénarios hybrides et des stratégies de migration. Pour plus d'informations sur les stratégies de migration, consultez le livre blanc sur la méthodologie de migration des conteneurs.

Êtes-vous Well-Architected ?

Le AWS Well-Architected Framework vous aide à comprendre les avantages et les inconvénients des décisions que vous prenez lors de la création de systèmes dans le cloud. Les six piliers du cadre vous permettent d'apprendre les meilleures pratiques architecturales pour concevoir et exploiter des systèmes fiables, sécurisés, efficaces, rentables et durables. À l'aide du AWS Well-Architected Tool, disponible gratuitement dans le AWS Management Console, vous pouvez évaluer votre charge de travail par rapport à ces meilleures pratiques en répondant à une série de questions pour chaque pilier.

Dans le Serverless Application Lens, nous nous concentrons sur les meilleures pratiques en matière d'architecture de vos applications sans serveur. AWS

Pour obtenir des conseils d'experts supplémentaires et les meilleures pratiques relatives à votre architecture cloud (déploiements d'architecture de référence, diagrammes et livres blancs), consultez le Centre d'architecture.AWS

Modernisation vers les microservices

Les microservices sont essentiellement de petites unités indépendantes qui constituent une application. La transition des structures monolithiques traditionnelles vers les microservices peut suivre différentes stratégies.

Cette transition a également un impact sur le mode de fonctionnement de votre organisation :

- Il encourage le développement agile, dans le cadre duquel les équipes travaillent selon des cycles rapides.
- Les équipes sont généralement petites, parfois décrites comme deux équipes de pizza, suffisamment petites pour que deux pizzas puissent nourrir toute l'équipe.
- Les équipes assument l'entière responsabilité de leurs services, de la création au déploiement et à la maintenance.

Architecture de microservices simple sur AWS

Les applications monolithiques classiques se composent de différentes couches : une couche de présentation, une couche d'application et une couche de données. Les architectures de microservices, quant à elles, séparent les fonctionnalités en secteurs verticaux cohérents en fonction de domaines spécifiques, plutôt que de couches technologiques. La figure 1 illustre une architecture de référence pour une application de microservices typique sur AWS.

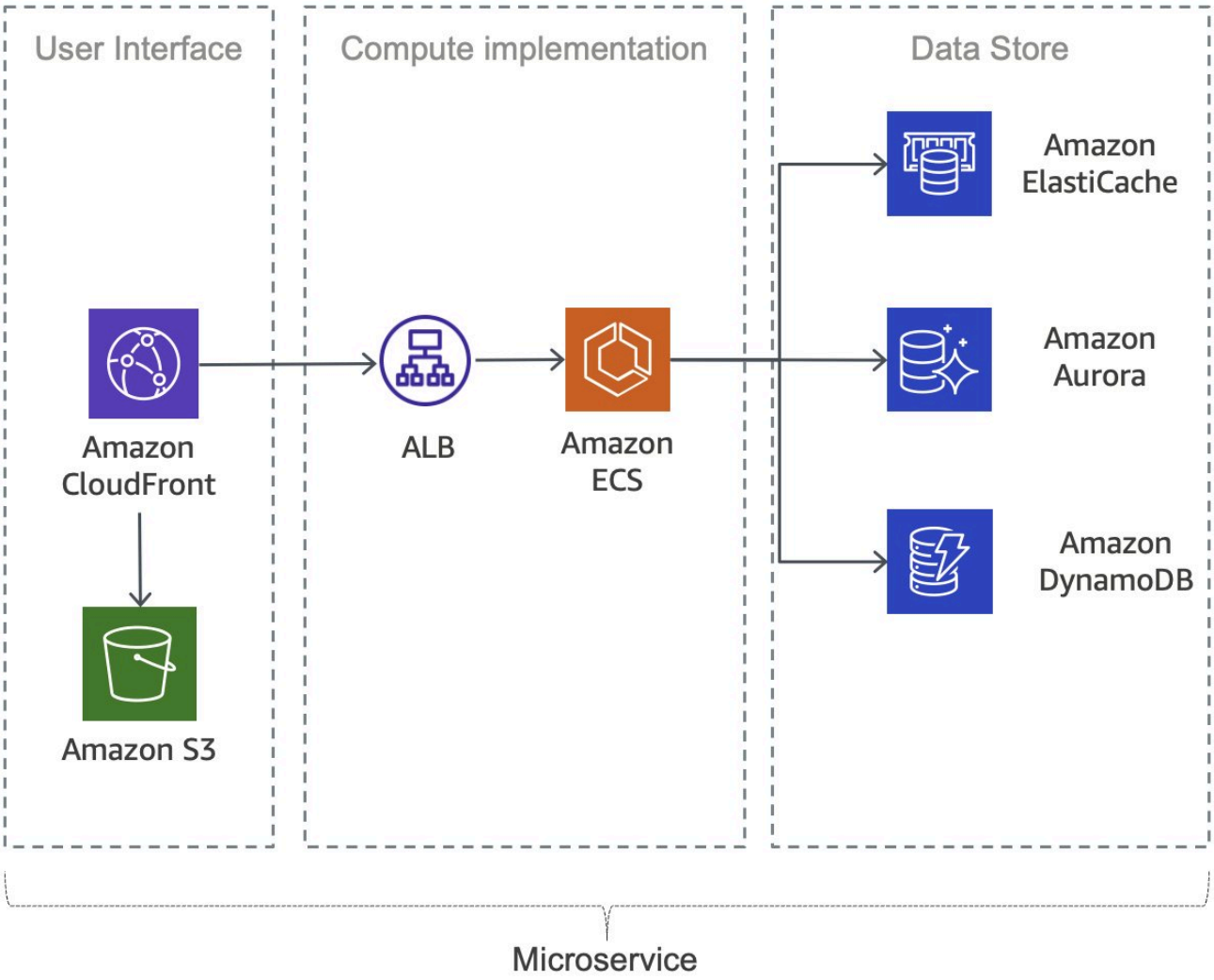

Figure 1 : Application de microservices typique sur AWS

Interface utilisateur

Les applications Web modernes utilisent souvent JavaScript des frameworks pour développer des applications d'une seule page qui communiquent avec le backend APIs. Ils APIs sont généralement créés à l'aide de Representational State Transfer (REST) ou RESTful APIs GraphQL APIs. Le contenu Web statique peut être diffusé à l'aide d'Amazon Simple Storage Service (Amazon S3) et d'Amazon CloudFront.

Microservices

APIs sont considérés comme la porte d'entrée des microservices, car ils constituent le point d'entrée de la logique des applications. Généralement, l'API des services RESTful Web ou GraphQL APIs sont utilisés. Ils APIs gèrent et traitent les appels des clients, en gérant des fonctions telles que la gestion du trafic, le filtrage des demandes, le routage, la mise en cache, l'authentification et l'autorisation.

Implémentations de microservices

AWS propose des éléments de base pour développer des microservices, notamment Amazon ECS et Amazon EKS en tant que choix pour les moteurs d'orchestration de conteneurs AWS Fargate et en EC2 tant qu'options d'hébergement. AWS Lambda est un autre moyen sans serveur de créer des microservices. AWS Le choix entre ces options d'hébergement dépend des exigences du client en matière de gestion de l'infrastructure sous-jacente.

AWS Lambda vous permet de télécharger votre code, de le dimensionner automatiquement et de gérer son exécution avec une haute disponibilité. Cela élimine le besoin de gestion de l'infrastructure, ce qui vous permet d'agir rapidement et de vous concentrer sur votre logique métier. Lambda prend en charge plusieurs langages de programmation et peut être déclenché par d'autres AWS services ou appelé directement depuis des applications Web ou mobiles.

Les applications basées sur des conteneurs ont gagné en popularité en raison de leur portabilité, de leur productivité et de leur efficacité.AWS propose plusieurs services pour créer, déployer et gérer des conteneurs.

- App2Container, un outil de ligne de commande pour la migration et la modernisation des applications Web Java et .NET au format conteneur. AWS A2C analyse et dresse un inventaire des applications exécutées sur du matériel nu, des machines virtuelles, des instances Amazon Elastic Compute Cloud (EC2) ou dans le cloud.

- Amazon Elastic Container Service (Amazon ECS) et Amazon Elastic Kubernetes Service (Amazon EKS) gèrent votre infrastructure de conteneurs, ce qui facilite le lancement et la maintenance des applications conteneurisées.
  - Amazon EKS est un service Kubernetes géré permettant d'exécuter Kubernetes dans le AWS cloud et dans des centres de données sur site (Amazon EKS Anywhere). Cela étend les services cloud aux environnements sur site pour le traitement local des données à faible latence, les coûts de transfert de données élevés ou les exigences en matière de résidence des données (consultez le livre blanc intitulé « Running Hybrid Container Workloads with Amazon EKS Anywhere »). Vous pouvez utiliser tous les plug-ins et outils existants de la communauté Kubernetes avec EKS.
  - Amazon Elastic Container Service (Amazon ECS) est un service d'orchestration de conteneurs entièrement géré qui simplifie le déploiement, la gestion et le dimensionnement des applications conteneurisées. Les clients choisissent ECS pour sa simplicité et son intégration approfondie avec AWS les services.

Pour en savoir plus, consultez le blog Amazon ECS vs Amazon EKS : making sense of AWS container services.

- AWS App Runnerest un service d'applications de conteneur entièrement géré qui vous permet de créer, de déployer et d'exécuter des applications Web conteneurisées et des services d'API sans expérience préalable en matière d'infrastructure ou de conteneur.
- AWS Fargate, un moteur de calcul sans serveur, fonctionne à la fois avec Amazon ECS et Amazon EKS pour gérer automatiquement les ressources de calcul pour les applications de conteneurs.
- Amazon ECR est un registre de conteneurs entièrement géré offrant un hébergement performant, qui vous permet de déployer de manière fiable des images et des artefacts d'applications où que vous soyez.

Intégration continue et déploiement continu (CI/CD)

L'intégration continue et la livraison continue (CI/CD) constituent un élément crucial d'une DevOps initiative visant à modifier rapidement les logiciels. AWS propose des services pour implémenter le CI/CD pour les microservices, mais une discussion détaillée dépasse le cadre de ce document. Pour plus d'informations, consultez le AWS livre blanc Practizing Continuous Integration and Continuous Delivery on.

Réseaux privés

AWS PrivateLink est une technologie qui améliore la sécurité des microservices en permettant des connexions privées entre votre Virtual Private Cloud (VPC) et les services pris en charge. AWS Il permet d'isoler et de sécuriser le trafic des microservices, en veillant à ce qu'il ne traverse jamais l'Internet public. Cela est particulièrement utile pour se conformer à des réglementations telles que PCI ou HIPAA.

Banque 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 les caches en mémoire tels que Memcached ou Redis. AWS propose les deux technologies dans le cadre du ElastiCache service géré Amazon.

La mise en cache entre les serveurs d'applications et une base de données est un mécanisme courant pour réduire la charge de lecture sur la base de données, ce qui peut permettre d'utiliser des ressources pour prendre en charge un plus grand nombre d'écritures. Les caches peuvent également améliorer la latence.

Les bases de données relationnelles sont toujours très populaires pour stocker des données structurées et des objets métiers. AWS propose six moteurs de base de données (Microsoft SQL Server, Oracle, MySQL, MariaDB, PostgreSQL et Amazon Aurora) sous forme de services gérés via Amazon Relational Database Service (Amazon RDS).

Cependant, les bases de données relationnelles ne sont pas conçues pour une échelle infinie, ce qui peut rendre difficile et chronophage l'application de techniques permettant de prendre en charge un grand nombre de requêtes.

Les bases de données NoSQL ont été conçues pour privilégier l'évolutivité, les performances et la disponibilité par rapport à la cohérence des bases de données relationnelles. Un élément important des bases de données NoSQL est qu'elles n'appliquent généralement pas de schéma strict. Les données sont réparties sur des partitions qui peuvent être redimensionnées horizontalement et sont récupérées à l'aide de clés de partition.

Les microservices individuels étant conçus pour faire une chose bien, ils disposent généralement d'un modèle de données simplifié qui peut être bien adapté à la persistance NoSQL. Il est important de comprendre que les modèles d'accès aux bases de données NoSQL sont différents de ceux des bases de données relationnelles. Par exemple, il n'est pas possible de joindre des tables. Si cela

est nécessaire, la logique doit être implémentée dans l'application. Vous pouvez utiliser Amazon DynamoDB pour créer une table de base de données capable de stocker et de récupérer n'importe quel volume de données et de traiter n'importe quel niveau de trafic de demandes. DynamoDB fournit des performances à un chiffre en millisecondes, mais certains cas d'utilisation nécessitent des temps de réponse en microsecondes. L'accélérateur DynamoDB (DAX) fournit des fonctionnalités de mise en cache pour accéder aux données.

DynamoDB propose également une fonction de mise à l'échelle automatique pour ajuster dynamiquement la capacité de débit en réponse au trafic réel. Cependant, dans certains cas, la planification des capacités est difficile ou impossible en raison de pics d'activité importants de courte durée dans votre application. Dans de telles situations, DynamoDB propose une option à la demande, qui propose une tarification simple. pay-per-request DynamoDB à la demande est capable de répondre instantanément à des milliers de demandes par seconde, sans planification des capacités.

Pour plus d'informations, reportez-vous à Gestion distribuée des données la section Comment choisir une base de données.

Simplification des opérations

Pour simplifier davantage les efforts opérationnels nécessaires à l'exécution, à la maintenance et à la surveillance des microservices, nous pouvons utiliser une architecture entièrement sans serveur.

Déploiement d'applications basées sur Lambda

Vous pouvez déployer votre code Lambda en chargeant une archive de `zip` fichiers ou en créant et en téléchargeant une image de conteneur via l'interface utilisateur de la console à l'aide d'une URI d'image Amazon ECR valide. Toutefois, lorsqu'une fonction Lambda devient complexe, c'est-à-dire qu'elle comporte des couches, des dépendances et des autorisations, le téléchargement via l'interface utilisateur peut s'avérer fastidieux en cas de modification du code.

L'utilisation de AWS CloudFormation and the AWS Serverless Application Model (AWS SAM) ou de Terraform rationalise le processus de définition des applications sans serveur. AWS Cloud Development Kit (AWS CDK) AWS SAM, pris en charge nativement par CloudFormation, propose une syntaxe simplifiée pour spécifier les ressources sans serveur.AWS Lambda Les couches aident à gérer les bibliothèques partagées entre plusieurs fonctions Lambda, en minimisant l'encombrement des fonctions, en centralisant les bibliothèques adaptées aux locataires et en améliorant l'expérience des développeurs. Lambda SnapStart pour Java améliore les performances de démarrage des applications sensibles à la latence.

Pour déployer, spécifiez les ressources et les politiques d'autorisation dans un CloudFormation modèle, empaquetez les artefacts de déploiement et déployez le modèle. SAM Local, un AWS CLI outil, permet le développement local, le test et l'analyse d'applications sans serveur avant leur téléchargement vers Lambda.

L'intégration avec des outils tels que l' AWS Cloud9 IDE et AWS CodePipeline rationalise la création AWS CodeBuild AWS CodeDeploy, les tests, le débogage et le déploiement d'applications basées sur le SAM.

Le schéma suivant montre le déploiement de AWS Serverless Application Model ressources à l'aide CloudFormation d'outils AWS CI/CD.

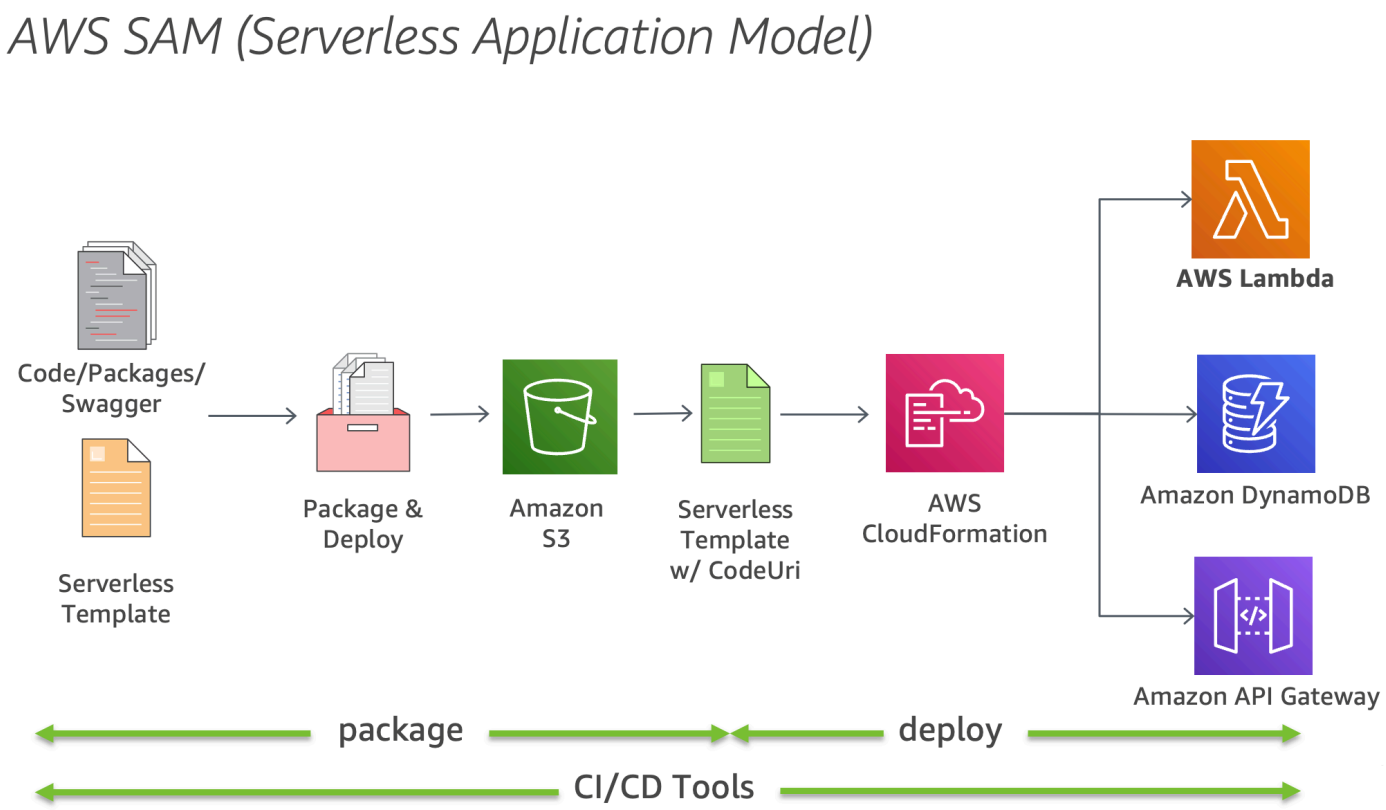

Figure 2 : AWS Serverless Application Model (AWS SAM)

Abstraction des complexités liées à la multilocation

Dans un environnement mutualisé tel que les plateformes SaaS, il est essentiel de rationaliser les subtilités liées à la mutualisation, afin de permettre aux développeurs de se concentrer sur le développement des fonctionnalités. Cela peut être réalisé à l'aide d'outils tels que AWS Lambda Layers, qui proposent des bibliothèques partagées pour répondre à des préoccupations

transversales. La raison d'être de cette approche est que les bibliothèques et les outils partagés, lorsqu'ils sont utilisés correctement, gèrent efficacement le contexte des locataires.

Cependant, ils ne devraient pas s'étendre à l'encapsulation de la logique métier en raison de la complexité et des risques qu'ils peuvent introduire. L'un des problèmes fondamentaux des bibliothèques partagées est la complexité croissante des mises à jour, qui les rend plus difficiles à gérer par rapport à la duplication de code standard. Il est donc essentiel de trouver un équilibre entre l'utilisation de bibliothèques partagées et la duplication dans la recherche de l'abstraction la plus efficace.

Gestion des API

La gestion APIs peut prendre beaucoup de temps, en particulier lorsqu'il s'agit de plusieurs versions, d'étapes du cycle de développement, d'autorisations et d'autres fonctionnalités telles que la limitation et la mise en cache. Outre API Gateway, certains clients utilisent également ALB (Application Load Balancer) ou NLB (Network Load Balancer) pour la gestion des API. Amazon API Gateway permet de réduire la complexité opérationnelle liée à la création et à la maintenance RESTful APIs. Il vous permet de créer APIs par programmation, sert de « porte d'entrée » pour accéder aux données, à la logique métier ou aux fonctionnalités de vos services principaux, à l'autorisation et au contrôle d'accès, à la limitation du débit, à la mise en cache, à la surveillance et à la gestion du trafic et fonctionne APIs sans gérer de serveurs.

La figure 3 montre comment API Gateway gère les appels d'API et interagit avec les autres composants. Les demandes provenant d'appareils mobiles, de sites Web ou d'autres services principaux sont acheminées vers le CloudFront point de présence (PoP) le plus proche afin de réduire la latence et d'offrir une expérience utilisateur optimale.

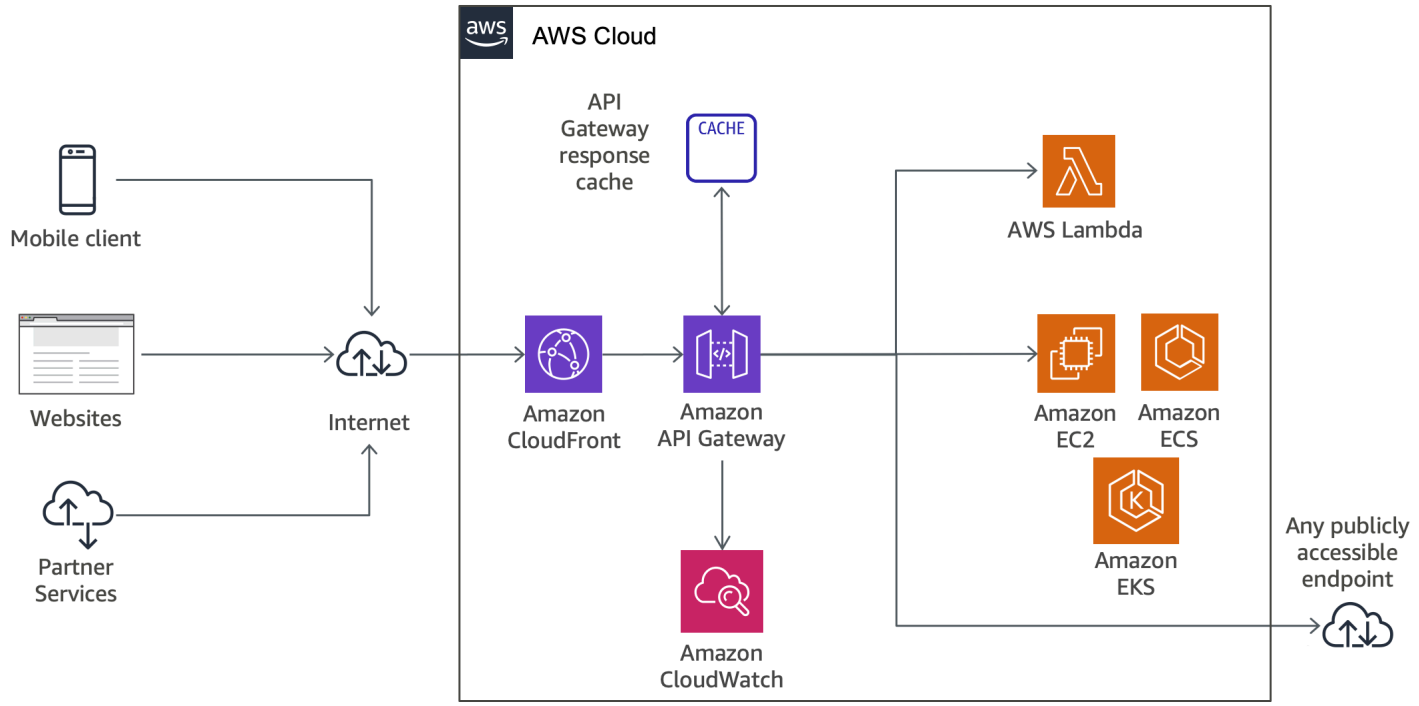

Figure 3 : flux d'appels API Gateway

Microservices sur les technologies sans serveur

L'utilisation de microservices avec des technologies sans serveur peut considérablement réduire la complexité opérationnelle. AWS Lambda et AWS Fargate, intégré à API Gateway, permet la création d'applications entièrement sans serveur. À compter du 7 avril 2023, les fonctions Lambda peuvent progressivement renvoyer les charges utiles de réponse au client, améliorant ainsi les performances des applications Web et mobiles. Auparavant, les applications basées sur Lambda utilisant le modèle traditionnel d'invocation de demande-réponse devaient générer et mettre en mémoire tampon la réponse avant de la renvoyer au client, ce qui pouvait retarder le délai jusqu'au premier octet. Grâce au streaming des réponses, les fonctions peuvent renvoyer des réponses partielles au client dès qu'elles sont prêtes, ce qui améliore considérablement le délai d'obtention du premier octet, auquel les applications Web et mobiles sont particulièrement sensibles.

La figure 4 illustre une architecture de microservices sans serveur utilisant AWS Lambda des services gérés. Cette architecture sans serveur réduit le besoin de concevoir en fonction de l'évolutivité et de la haute disponibilité, et réduit les efforts nécessaires à l'exécution et à la surveillance de l'infrastructure sous-jacente.

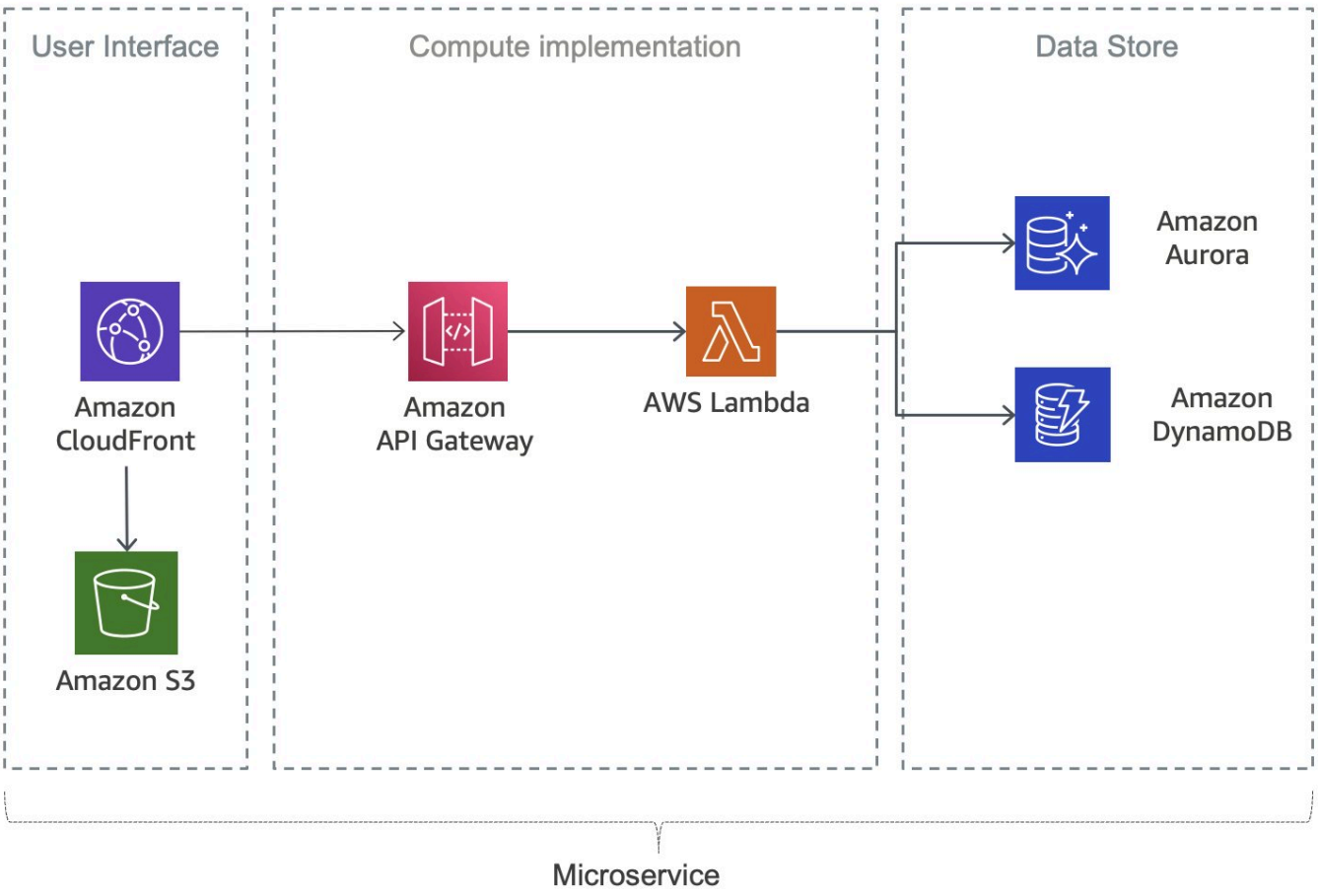

Figure 4 : microservice sans serveur utilisant AWS Lambda

La figure 5 montre une implémentation sans serveur similaire utilisant des conteneurs AWS Fargate, éliminant ainsi les inquiétudes concernant l'infrastructure sous-jacente. Il intègre également Amazon Aurora Serverless, une base de données auto-scalable à la demande qui ajuste automatiquement la capacité en fonction des exigences de votre application.

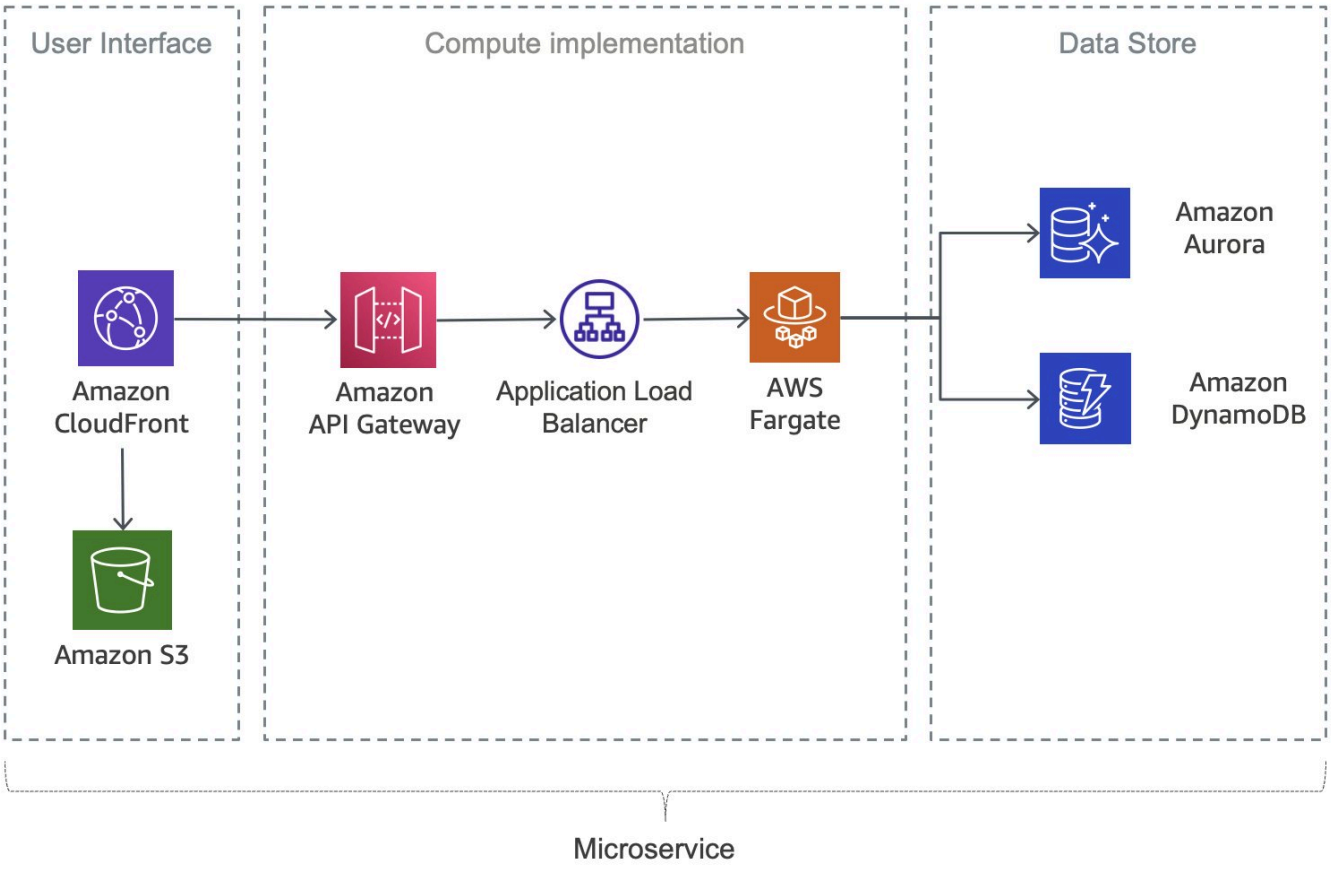

Figure 5 : microservice sans serveur utilisant AWS Fargate

Systèmes résilients et efficaces

Reprise après sinistre (DR)

Les applications de microservices suivent souvent le modèle des applications à douze facteurs, dans lequel les processus sont apatrides et les données persistantes sont stockées dans des services de sauvegarde dynamiques tels que les bases de données. Cela simplifie la reprise après sinistre (DR), car en cas de défaillance d'un service, il est facile de lancer de nouvelles instances pour rétablir les fonctionnalités.

Les stratégies de reprise après sinistre pour les microservices doivent se concentrer sur les services en aval qui maintiennent l'état de l'application, tels que les systèmes de fichiers, les bases de données ou les files d'attente. Organisations doivent planifier un objectif de temps de reprise (RTO) et un objectif de point de reprise (RPO). Le RTO est le délai maximal acceptable entre l'interruption du service et la restauration, tandis que le RPO est le délai maximal écoulé depuis le dernier point de récupération des données.

Pour en savoir plus sur les stratégies de reprise après sinistre, consultez le livre blanc sur la reprise après sinistre des charges de travail sur AWS : la restauration dans le cloud.

Haute disponibilité (HA)

Nous examinerons la haute disponibilité (HA) pour les différents composants d'une architecture de microservices.

Amazon EKS fournit une haute disponibilité en exécutant des instances de contrôle et de plan de données Kubernetes dans plusieurs zones de disponibilité. Il détecte et remplace automatiquement les instances du plan de contrôle défaillantes et fournit des mises à niveau de version et des correctifs automatisés.

Amazon ECR utilise Amazon Simple Storage Service (Amazon S3) pour le stockage afin de rendre vos images de conteneurs hautement disponibles et accessibles. Il fonctionne avec Amazon EKS, Amazon ECS et AWS Lambda simplifie le flux de travail du développement à la production.

Amazon ECS est un service régional qui simplifie la gestion des conteneurs de manière hautement disponible dans plusieurs zones de disponibilité au sein d'une région, en proposant plusieurs stratégies de planification qui placent les conteneurs en fonction des besoins en ressources et des exigences de disponibilité.

AWS Lambda fonctionne dans plusieurs zones de disponibilité, garantissant ainsi la disponibilité pendant les interruptions de service dans une seule zone. Si vous connectez votre fonction à un VPC, spécifiez des sous-réseaux dans plusieurs zones de disponibilité pour une haute disponibilité.

Composants de systèmes distribués

Dans une architecture de microservices, la découverte de services fait référence au processus de localisation et d'identification dynamiques des emplacements réseau (adresses IP et ports) de microservices individuels au sein d'un système distribué.

Lorsque vous choisissez une approche AWS, tenez compte de facteurs tels que :

- Modification du code : pouvez-vous bénéficier des avantages sans modifier le code ?
- Trafic entre VPC ou entre comptes : si nécessaire, votre système a-t-il besoin d'une gestion efficace des communications entre différents opérateurs ? VPCs Comptes AWS
- Stratégies de déploiement : votre système utilise-t-il ou prévoit-il d'utiliser des stratégies de déploiement avancées telles que les déploiements Blue-Green ou Canary ?
- Considérations relatives aux performances : si votre architecture communique fréquemment avec des services externes, quel en sera l'impact sur les performances globales ?

AWS propose plusieurs méthodes pour implémenter la découverte de services dans votre architecture de microservices :

- Amazon ECS Service Discovery : Amazon ECS prend en charge la découverte de services à l'aide de sa méthode basée sur le DNS ou en l'intégrant à AWS Cloud Map (voir ECS Service Discovery). ECS Service Connect améliore encore la gestion des connexions, ce qui peut être particulièrement bénéfique pour les applications de plus grande envergure dotées de plusieurs services en interaction.
- Amazon Route 53 : Route 53 s'intègre à ECS et à d'autres AWS services, tels que EKS, pour faciliter la découverte de services. Dans un contexte ECS, Route 53 peut utiliser la fonctionnalité ECS Service Discovery, qui tire parti de l'API Auto Naming pour enregistrer et désenregistrer automatiquement les services.
- AWS Cloud Map: cette option propose une découverte de services dynamique basée sur une API, qui propage les modifications à l'ensemble de vos services.

Pour les besoins de communication les plus avancés, Amazon VPC Lattice est un service de mise en réseau d'applications qui connecte, surveille et sécurise de manière cohérente les communications entre vos services, contribuant ainsi à améliorer la productivité afin que vos développeurs puissent se concentrer sur la création de fonctionnalités importantes pour votre entreprise. Vous pouvez

définir des politiques de gestion du trafic réseau, d'accès et de surveillance afin de connecter les services informatiques de manière simplifiée et cohérente entre les instances, les conteneurs et les applications sans serveur.

Si vous utilisez déjà des logiciels tiers, tels que HashiCorp Consul ou Netflix Eureka pour la découverte de services, vous préférerez peut-être continuer à les utiliser lors de votre migration AWS, afin de faciliter la transition.

Le choix entre ces options doit correspondre à vos besoins spécifiques. Pour des exigences plus simples, des solutions basées sur le DNS comme Amazon ECS ou Amazon AWS Cloud Map peuvent être suffisantes. Pour les systèmes plus complexes ou plus volumineux, les maillages de service tels qu'Amazon VPC Lattice peuvent être plus adaptés.

En conclusion, la conception d'une architecture de microservices AWS consiste à sélectionner les bons outils pour répondre à vos besoins spécifiques. En gardant à l'esprit les considérations évoquées, vous pouvez vous assurer de prendre des décisions éclairées pour optimiser la découverte des services et la communication entre services de votre système.

Gestion distribuée des données

Dans les applications traditionnelles, tous les composants partagent souvent une seule base de données. En revanche, chaque composant d'une application basée sur des microservices conserve ses propres données, ce qui favorise l'indépendance et la décentralisation. Cette approche, connue sous le nom de gestion distribuée des données, pose de nouveaux défis.

L'un de ces défis réside dans le compromis entre cohérence et performance dans les systèmes distribués. Il est souvent plus pratique d'accepter de légers retards dans les mises à jour des données (cohérence éventuelle) que d'exiger des mises à jour instantanées (cohérence immédiate).

Parfois, les opérations commerciales nécessitent plusieurs microservices pour fonctionner ensemble. Si une partie échoue, vous devrez peut-être annuler certaines tâches terminées. Le modèle Saga permet de gérer cela en coordonnant une série d'actions compensatoires.

Pour aider les microservices à rester synchronisés, un magasin de données centralisé peut être utilisé. Ce magasin, géré à l'aide d'outils tels que AWS Lambda AWS Step Functions,, et Amazon EventBridge, peut aider à nettoyer et à dédupliquer les données.

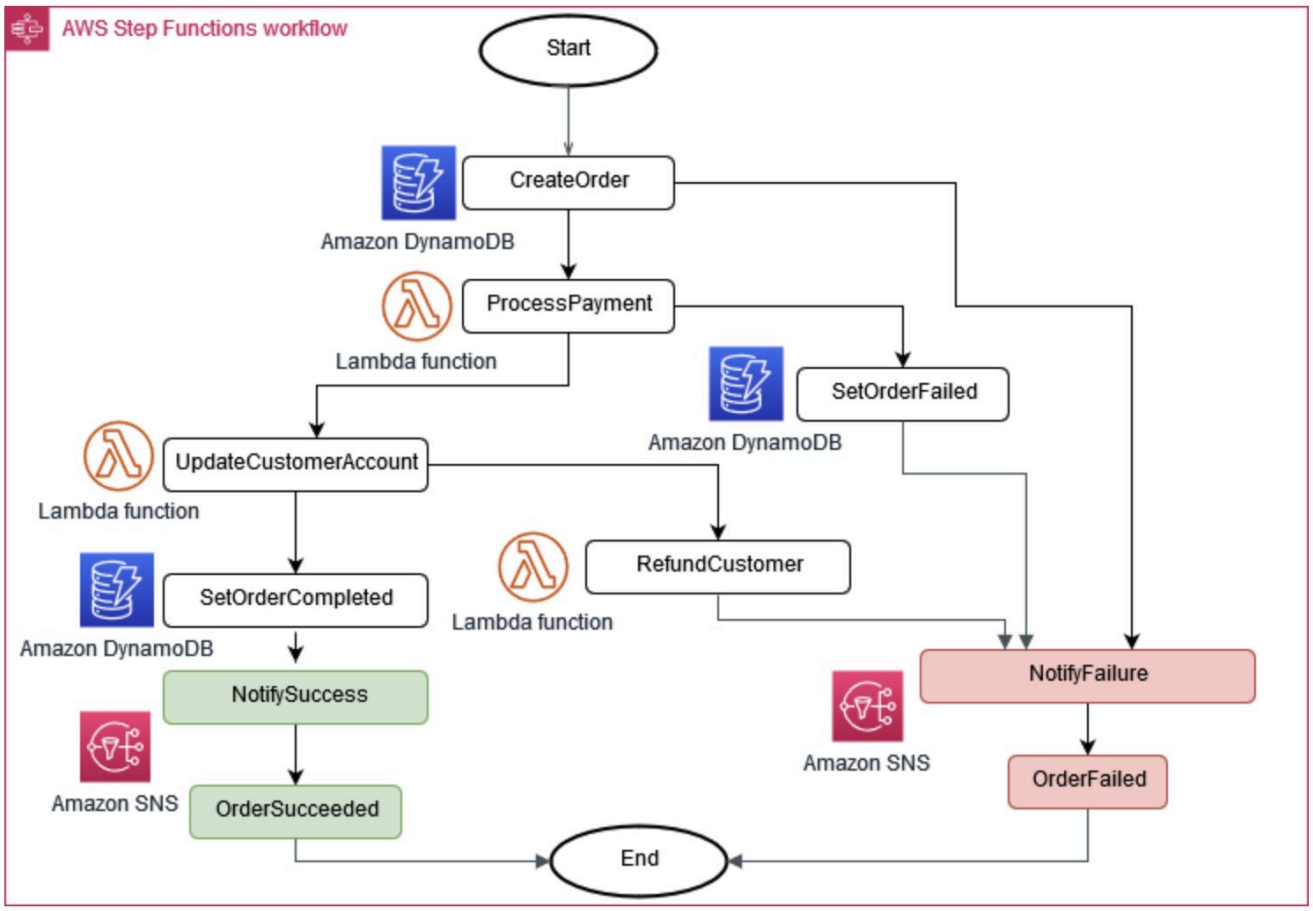

Figure 6 : Coordinateur de l'exécution de Saga

L'approvisionnement en événements est une approche courante pour gérer les changements dans les microservices. Chaque modification apportée à l'application est enregistrée en tant qu'événement, ce qui crée une chronologie de l'état du système. Cette approche permet non seulement le débogage et l'audit, mais permet également aux différentes parties d'une application de réagir aux mêmes événements.

L'approvisionnement en événements fonctionne souvent hand-in-hand avec le modèle CQRS (Command Query Responsibility Segrégation), qui sépare la modification des données de l'interrogation de données en différents modules pour améliorer les performances et la sécurité.

Oui AWS, vous pouvez implémenter ces modèles à l'aide d'une combinaison de services. Comme le montre la Figure 7, Amazon Kinesis Data Streams peut servir de magasin d'événements central, tandis qu'Amazon S3 fournit un stockage durable pour tous les enregistrements d'événements. AWS

Lambda, Amazon DynamoDB et Amazon API Gateway travaillent ensemble pour gérer et traiter ces événements.

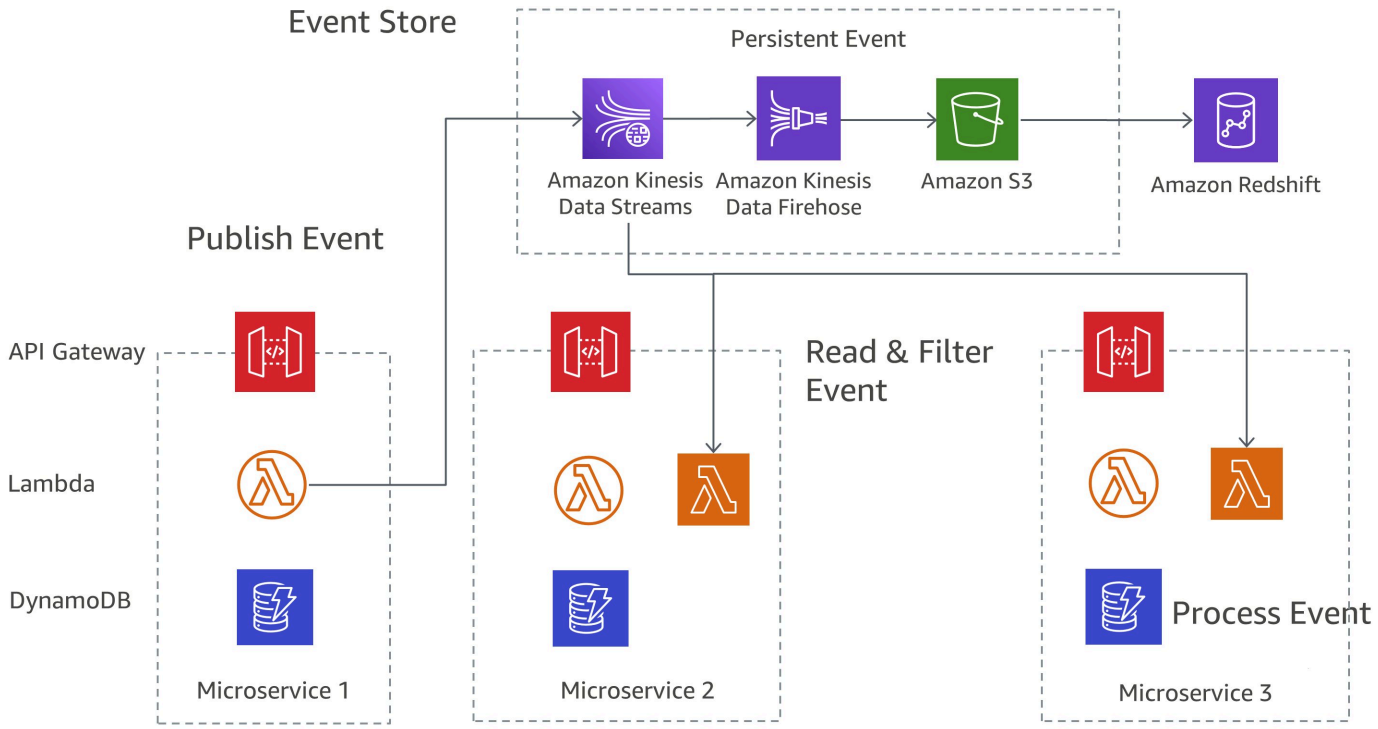

Figure 7 : Modèle d'approvisionnement en événements sur AWS

N'oubliez pas que dans les systèmes distribués, les événements peuvent être transmis plusieurs fois en raison de nouvelles tentatives. Il est donc important de concevoir vos applications de manière à gérer ce problème.

Gestion de la configuration

Dans une architecture de microservices, chaque service interagit avec diverses ressources telles que des bases de données, des files d'attente et d'autres services. Il est essentiel de disposer d'une méthode cohérente pour configurer les connexions et l'environnement d'exploitation de chaque service. Idéalement, une application doit s'adapter aux nouvelles configurations sans avoir besoin de redémarrer. Cette approche fait partie des principes de l'application à douze facteurs, qui recommandent de stocker les configurations dans des variables d'environnement.

Une autre approche consiste à utiliser AWS App Config. Il s'agit d'une fonctionnalité de AWS Systems Manager qui permet aux clients de configurer, de valider et de déployer rapidement et en toute sécurité des indicateurs de fonctionnalités et la configuration des applications. Votre indicateur de fonctionnalité et vos données de configuration peuvent être validés syntaxiquement ou sémantiquement lors de la phase de pré-déploiement, et peuvent être surveillés et automatiquement annulés si une alarme que vous avez configurée est déclenchée. AppConfig peut être intégré à Amazon ECS et Amazon EKS à l'aide de l' AWS AppConfig agent. L'agent fonctionne comme un conteneur annexe exécuté parallèlement à vos applications de conteneur Amazon ECS et Amazon EKS. Si vous utilisez des indicateurs de AWS AppConfig fonctionnalité ou d'autres données de configuration dynamiques dans une fonction Lambda, nous vous recommandons d'ajouter l'extension AWS AppConfig Lambda en tant que couche à votre fonction Lambda.

GitOpsest une approche innovante de la gestion des configurations qui utilise Git comme source fiable pour toutes les modifications de configuration. Cela signifie que toutes les modifications apportées à vos fichiers de configuration sont automatiquement suivies, versionnées et auditées via Git.

Gestion des secrets

La sécurité étant primordiale, les informations d'identification ne doivent pas être transmises en texte brut. AWS propose des services sécurisés à cet effet, tels que AWS Systems Manager Parameter Store et AWS Secrets Manager. Ces outils peuvent envoyer des secrets aux conteneurs d'Amazon EKS sous forme de volumes ou à Amazon ECS sous forme de variables d'environnement. Dans AWS Lambda, les variables d'environnement sont automatiquement mises à la disposition de votre code. Pour les flux de travail Kubernetes, l'opérateur de secrets externes récupère les secrets directement auprès de services AWS Secrets Manager, par exemple en créant les secrets Kubernetes correspondants. Cela permet une intégration fluide avec les configurations natives de Kubernetes.

Optimisation des coûts et durabilité

L'architecture des microservices peut améliorer l'optimisation des coûts et la durabilité. En divisant une application en parties plus petites, vous pouvez étendre uniquement les services qui nécessitent davantage de ressources, réduisant ainsi les coûts et le gaspillage. Cela est particulièrement utile lorsqu'il s'agit de traiter un trafic variable. Les microservices sont développés indépendamment. Les développeurs peuvent ainsi effectuer de petites mises à jour et réduire les ressources consacrées aux tests de bout en bout. Lors de la mise à jour, ils ne devront tester qu'un sous-ensemble des fonctionnalités, par opposition aux monolithes.

Les composants apatrides (services qui stockent l'état dans un magasin de données externe plutôt que dans un magasin de données local) de votre architecture peuvent utiliser les instances Amazon EC2 Spot, qui offrent une EC2 capacité inutilisée dans le AWS cloud. Ces instances sont plus économiques que les instances à la demande et conviennent parfaitement aux charges de travail susceptibles de gérer des interruptions. Cela permet de réduire encore les coûts tout en maintenant une haute disponibilité.

Avec les services isolés, vous pouvez utiliser des options de calcul optimisées en termes de coûts pour chaque groupe d'auto-scaling. Par exemple, AWS Graviton propose des options de calcul économiques et performantes pour les charges de travail adaptées aux instances basées sur ARM.

L'optimisation des coûts et de l'utilisation des ressources permet également de minimiser l'impact environnemental, conformément au pilier du développement durable du Well-Architected Framework. Vous pouvez suivre vos progrès en matière de réduction des émissions de carbone à l'aide de l'outil AWS Customer Carbon Footprint. Cet outil fournit des informations sur l'impact environnemental de votre AWS utilisation.

Mécanismes de communication

Dans le paradigme des microservices, les différents composants d'une application doivent communiquer sur un réseau. Les approches courantes à cet égard incluent la messagerie basée sur REST, GraphQL, GRPC et asynchrone.

Communication basée sur le repos

Le protocole HTTP/S, largement utilisé pour les communications synchrones entre microservices, fonctionne souvent. RESTful APIs API Gateway offre un moyen rationalisé de créer une API qui sert de point d'accès centralisé aux services principaux, gérant des tâches telles que la gestion du trafic, les autorisations, la surveillance et le contrôle des versions.

Communication basée sur GraphQL

De même, GraphQL est une méthode répandue de communication synchrone, utilisant les mêmes protocoles que REST mais limitant l'exposition à un seul point de terminaison. Vous pouvez ainsi créer et publier des applications GraphQL qui interagissent directement avec les AWS services et les banques de données, ou intégrer des fonctions Lambda pour la logique métier. AWS AppSync

Communication basée sur le GRPC

gRPC est un protocole de communication RPC open source synchrone, léger, performant. gRPC améliore ses protocoles sous-jacents en utilisant HTTP/2 et en activant davantage de fonctionnalités telles que la compression et la priorisation des flux. Il utilise le langage de définition d'interface Protobuf (IDL) qui est codé en binaire et tire ainsi parti du cadrage binaire HTTP/2.

Messagerie asynchrone et transmission d'événements

La messagerie asynchrone permet aux services de communiquer en envoyant et en recevant des messages via une file d'attente. Cela permet aux services de rester faiblement couplés et de promouvoir la découverte de services.

La messagerie peut être définie selon les trois types suivants :

- Files d'attente de messages : une file de messages agit comme une mémoire tampon qui dissocie les expéditeurs (producteurs) et les destinataires (consommateurs) des messages. Les

producteurs placent les messages dans la file d'attente, et les consommateurs les retirent et les traitent. Ce modèle est utile pour les communications asynchrones, le nivellement de charge et la gestion des rafales de trafic.

- Publier-Souscrire : dans le modèle de publication-abonnement, un message est publié sur un sujet et plusieurs abonnés intéressés reçoivent le message. Ce modèle permet de diffuser des événements ou des messages à plusieurs consommateurs de manière asynchrone.
- Messagerie axée sur les événements : La messagerie axée sur les événements consiste à capturer les événements qui se produisent dans le système et à y réagir. Les événements sont publiés sur un courtier de messages, et les services intéressés s'abonnent à des types d'événements spécifiques. Ce modèle permet un couplage souple et permet aux services de réagir aux événements sans dépendance directe.

Pour implémenter chacun de ces types de messages, AWS propose divers services gérés tels qu'Amazon SQS, Amazon SNS, Amazon, EventBridge Amazon MQ et Amazon MSK. Ces services présentent des fonctionnalités uniques adaptées à des besoins spécifiques :

- Amazon Simple Queue Service (Amazon SQS) et Amazon Simple Notification Service (Amazon SNS) : comme vous pouvez le voir sur la Figure 8, ces deux services se complètent, Amazon SQS fournissant un espace pour le stockage des messages et Amazon SNS permettant de distribuer des messages à plusieurs abonnés. Ils sont efficaces lorsque le même message doit être transmis à plusieurs destinations.

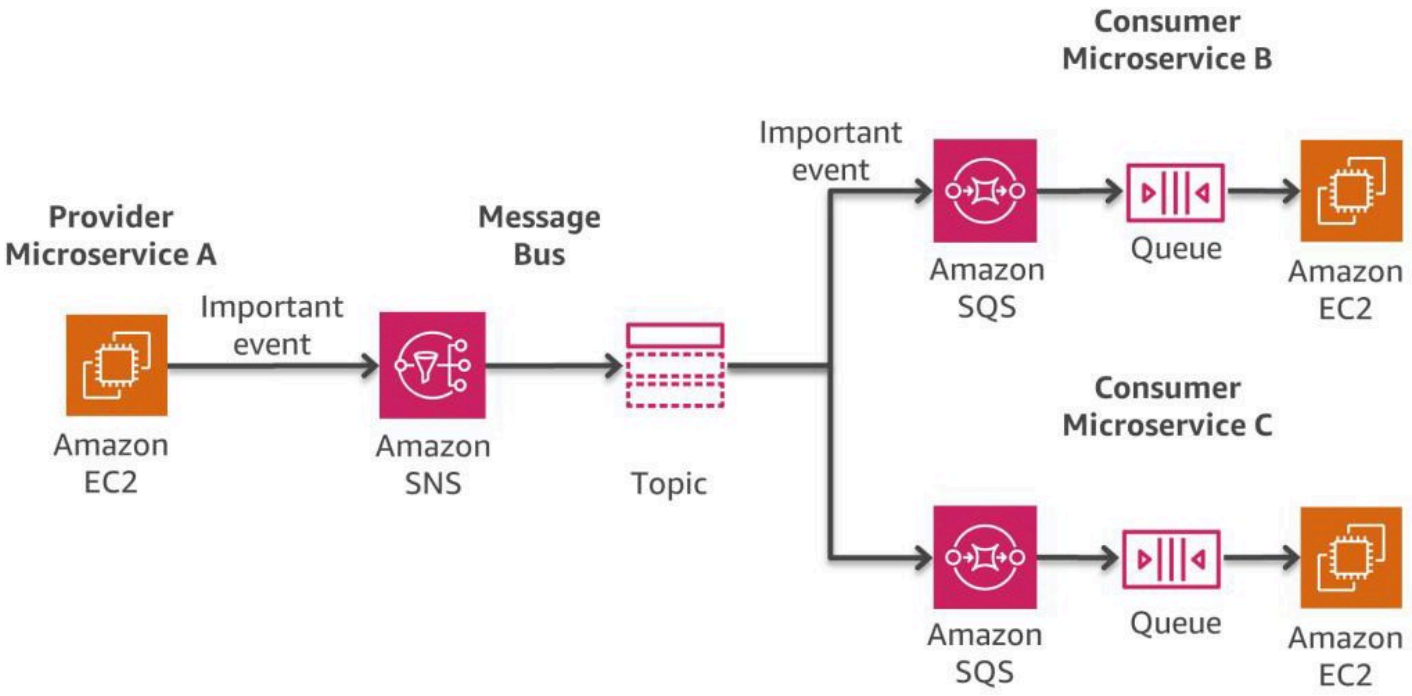

Figure 8 : Schéma du bus de messages activé AWS

- Amazon EventBridge : un service sans serveur qui utilise des événements pour connecter les composants de l'application entre eux, ce qui vous permet de créer plus facilement des applications évolutives pilotées par des événements. Utilisez-le pour acheminer des événements provenant de sources telles que des applications locales, AWS des services et des logiciels tiers vers des applications grand public au sein de votre entreprise. EventBridge fournit un moyen simple et cohérent d'ingérer, de filtrer, de transformer et de diffuser des événements afin que vous puissiez créer rapidement de nouvelles applications. EventBridge les bus d'événements sont parfaitement adaptés au many-to-many routage d'événements entre des services pilotés par des événements.
- Amazon MQ : un bon choix si vous possédez un système de messagerie préexistant qui utilise des protocoles standard tels que JMS, AMQP ou similaires. Ce service géré permet de remplacer votre système sans perturber les opérations.
- Amazon MSK (Managed Kafka) : un système de messagerie pour le stockage et la lecture des messages, utile dans les cas où les messages doivent être traités plusieurs fois. Il prend également en charge le streaming de messages en temps réel.
- Amazon Kinesis : traitement et analyse en temps réel des données de streaming. Cela permet le développement d'applications en temps réel et assure une intégration parfaite avec l' AWS écosystème.

N'oubliez pas que le meilleur service pour vous dépend de vos besoins spécifiques. Il est donc important de comprendre ce que chacun propose et comment il correspond à vos exigences.

Orchestration et gestion des états

L'orchestration des microservices fait référence à une approche centralisée, dans laquelle un composant central, appelé orchestrateur, est chargé de gérer et de coordonner les interactions entre les microservices. L'orchestration des flux de travail sur plusieurs microservices peut s'avérer difficile. Il est déconseillé d'intégrer le code d'orchestration directement dans les services, car cela introduit un couplage plus étroit et empêche le remplacement de services individuels.

Step Functions fournit un moteur de flux de travail pour gérer les complexités de l'orchestration des services, telles que la gestion des erreurs et la sérialisation. Cela vous permet de faire évoluer et de modifier rapidement les applications sans ajouter de code de coordination. Step Functions fait

partie de la plate-forme AWS sans serveur et prend en charge les fonctions Lambda EC2, Amazon, Amazon EKS, Amazon ECS SageMaker , AI AWS Glue, etc.

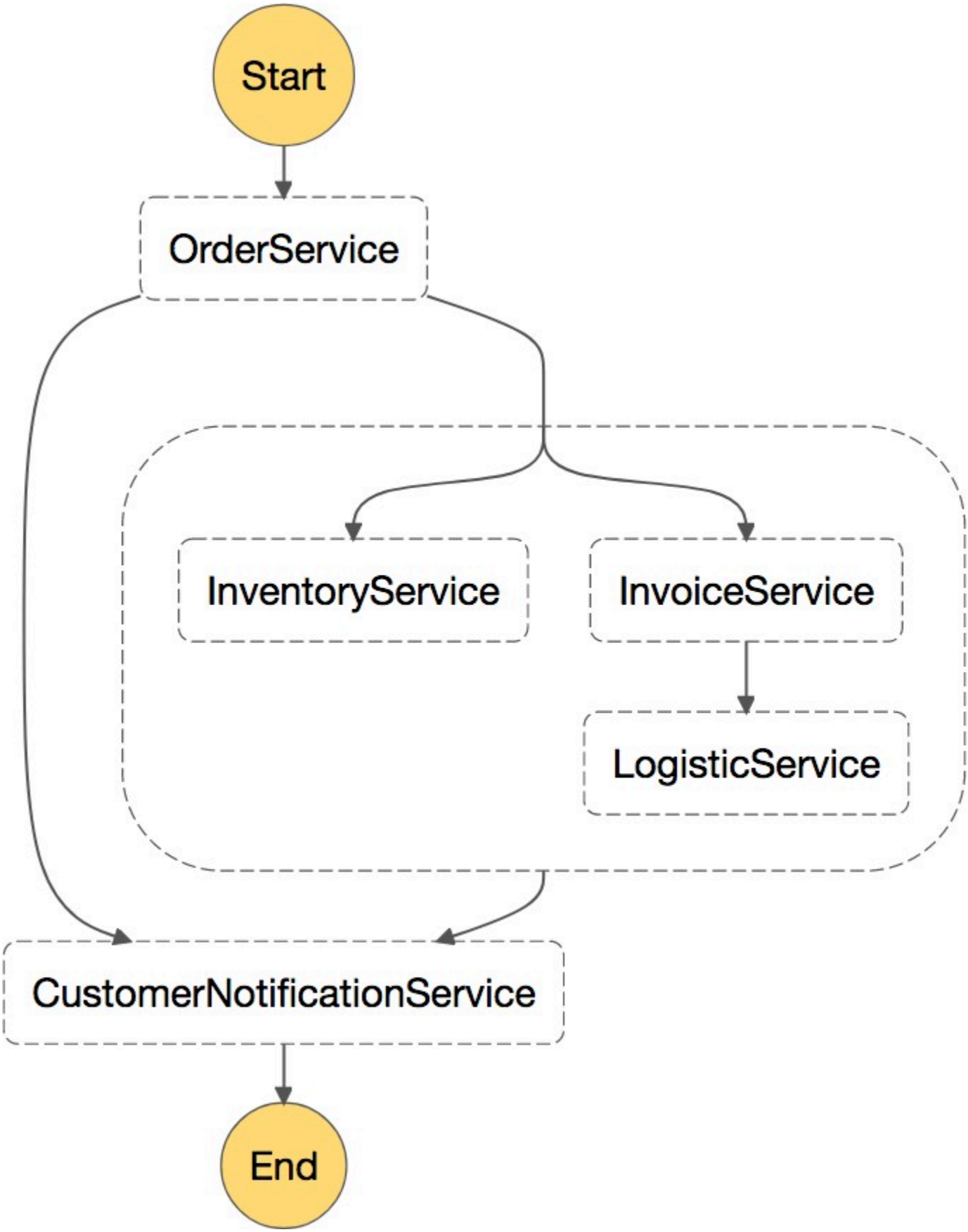
Start
OrderService
InventoryService
InvoiceService
LogisticService
CustomerNotificationService
End

Figure 9 : Exemple de flux de travail de microservices comportant des étapes parallèles et séquentielles invoquées par AWS Step Functions

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) est une alternative à Step Functions. Vous devez utiliser Amazon MWAA si vous privilégiez l'open source et la portabilité. Airflow dispose d'une communauté open source importante et active qui apporte régulièrement de nouvelles fonctionnalités et intégrations.

Observabilité

Les architectures de microservices étant intrinsèquement composées de nombreux composants distribués, l'observabilité de tous ces composants devient essentielle. Amazon CloudWatch permet cela en collectant et en suivant les métriques, en surveillant les fichiers journaux et en réagissant aux modifications de votre AWS environnement. Il peut surveiller AWS les ressources et les mesures personnalisées générées par vos applications et services.

Rubriques

Contrôle

CloudWatch offre une visibilité à l'échelle du système sur l'utilisation des ressources, les performances des applications et la santé opérationnelle. Dans une architecture de microservices, la surveillance des métriques personnalisées CloudWatch est avantageuse, car les développeurs peuvent choisir les métriques à collecter. La mise à l'échelle dynamique peut également être basée sur ces mesures personnalisées.

CloudWatch Container Insights étend cette fonctionnalité en collectant automatiquement des métriques pour de nombreuses ressources telles que le processeur, la mémoire, le disque et le réseau. Il aide à diagnostiquer les problèmes liés aux conteneurs et à rationaliser leur résolution.

Pour Amazon EKS, le choix souvent préféré est Prometheus, une plateforme open source offrant des fonctionnalités complètes de surveillance et d'alerte. Il est généralement associé à Grafana pour une visualisation intuitive des métriques. Amazon Managed Service for Prometheus (AMP) propose un service de surveillance entièrement compatible avec Prometheus, qui vous permet de superviser des applications conteneurisées sans effort. En outre, Amazon Managed Grafana (AMG) simplifie l'analyse et la visualisation de vos indicateurs, éliminant ainsi le besoin de gérer l'infrastructure sous-jacente.

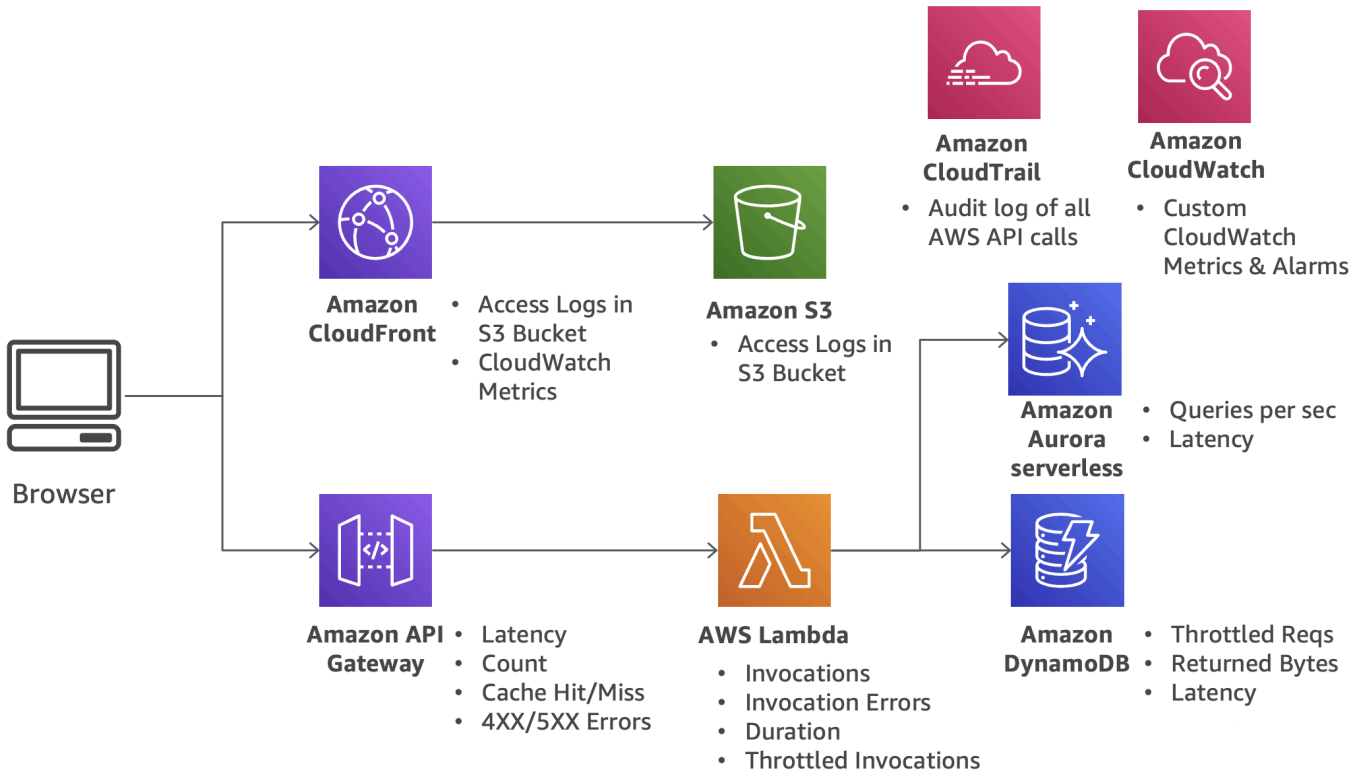

Figure 10 : Architecture sans serveur avec composants de surveillance

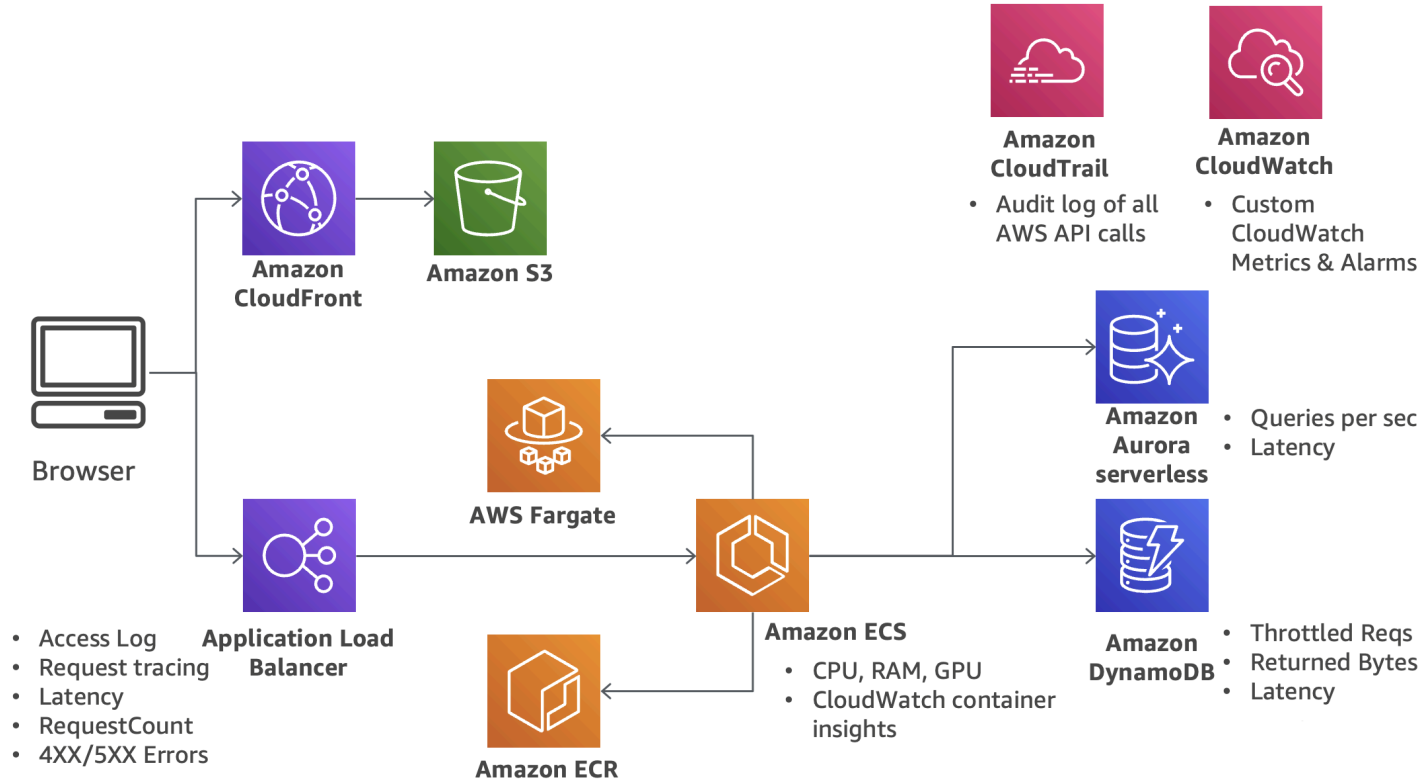

Figure 11 : Architecture basée sur des conteneurs avec des composants de surveillance

Centralisation des journaux

La journalisation est essentielle pour identifier et résoudre les problèmes. Grâce aux microservices, vous pouvez publier plus fréquemment et tester de nouvelles fonctionnalités. AWS fournit des services tels qu'Amazon S3, CloudWatch Logs et Amazon OpenSearch Service pour centraliser les fichiers journaux. Amazon EC2 utilise un démon pour envoyer les journaux à CloudWatch, tandis que Lambda et Amazon ECS y envoient nativement les résultats de leurs journaux. Pour Amazon EKS, Fluent Bit ou Fluentd peuvent être utilisés pour transférer les journaux vers Kibana à des CloudWatch fins de création OpenSearch de rapports. Cependant, en raison de son faible encombrement et de ses avantages en termes de performances, Fluent Bit est recommandé par rapport à Fluentd.

La figure 12 montre comment les journaux de différents AWS services sont dirigés vers Amazon S3 et CloudWatch. Ces journaux centralisés peuvent être analysés plus en détail à l'aide d'Amazon OpenSearch Service, notamment de Kibana pour la visualisation des données. Amazon Athena peut également être utilisé pour des requêtes ad hoc sur les journaux stockés dans Amazon S3.

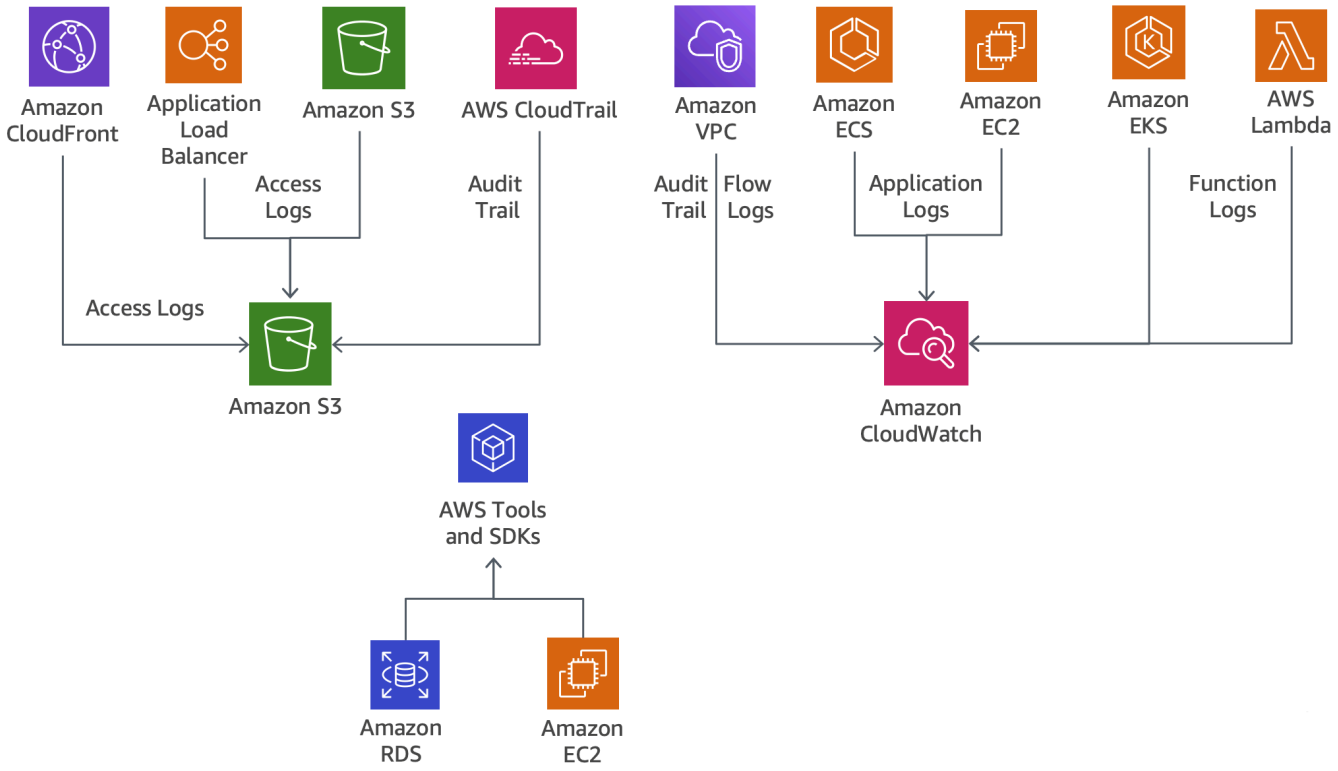

Figure 12 : Capacités de journalisation des AWS services

Traçabilité distribuée

Les microservices travaillent souvent ensemble pour traiter les demandes. AWS X-Ray utilise la corrélation IDs pour suivre les demandes provenant de ces services. X-Ray fonctionne avec Amazon EC2, Amazon ECS, Lambda et Elastic Beanstalk.

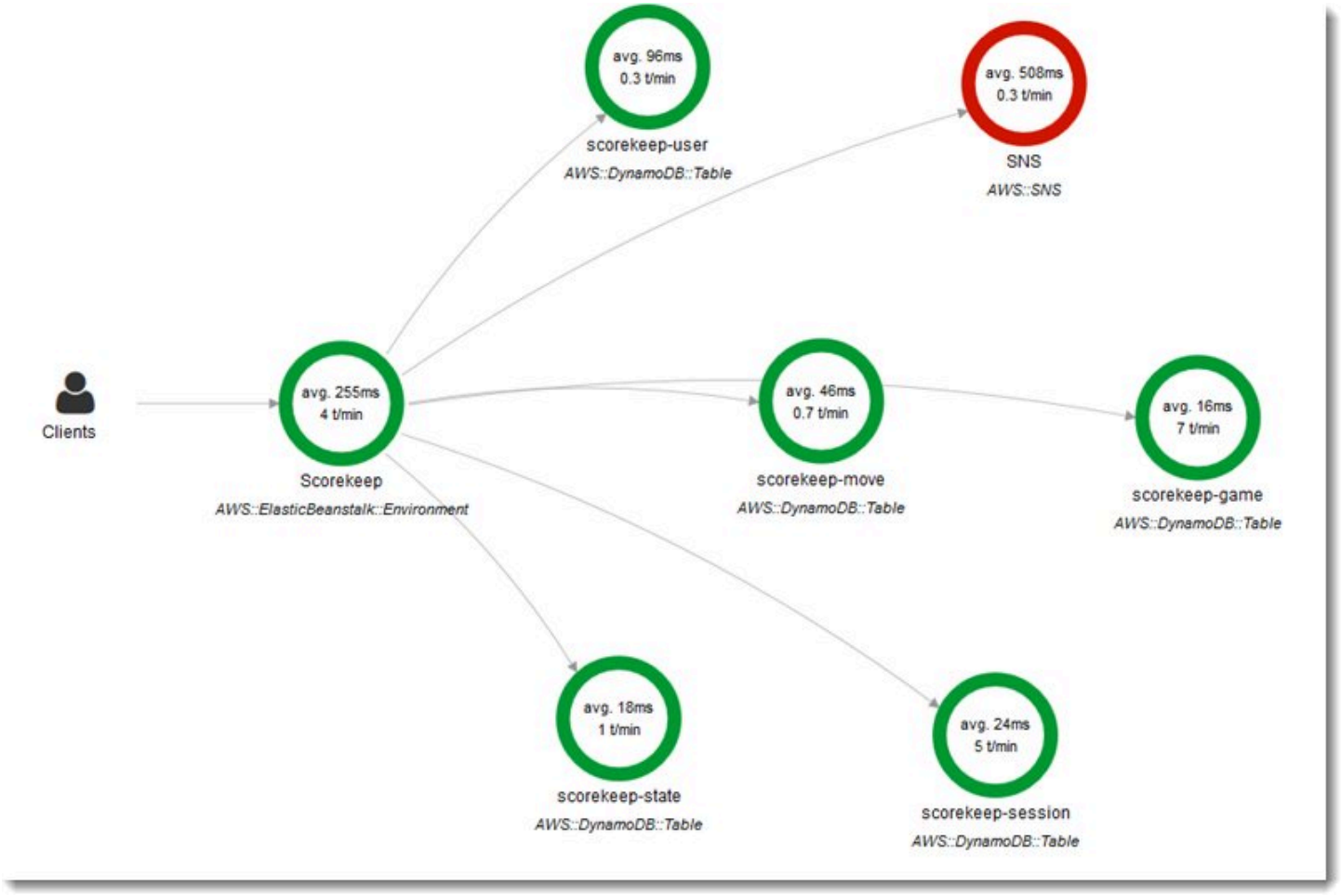

Figure 13 : carte AWS X-Ray des services

AWS Distro for OpenTelemetry fait partie du OpenTelemetry projet et fournit des logiciels libres et des agents pour collecter des traces APIs et des métriques distribuées, améliorant ainsi la surveillance de vos applications. Il envoie des métriques et des traces à de multiples solutions de surveillance AWS et à des solutions de surveillance partenaires. En collectant des métadonnées à partir de vos AWS ressources, il aligne les performances des applications sur les données d'infrastructure sous-jacentes, accélérant ainsi la résolution des problèmes. De plus, il est compatible avec de nombreux AWS services et peut être utilisé sur site.

Analyse du journal sur AWS

Amazon CloudWatch Logs Insights permet d'explorer, d'analyser et de visualiser les journaux en temps réel. Pour une analyse plus approfondie des fichiers journaux, Amazon OpenSearch Service, qui inclut Kibana, est un outil puissant. CloudWatch Les journaux peuvent transmettre les entrées des journaux au OpenSearch Service en temps réel. Kibana, parfaitement intégré OpenSearch, visualise ces données et propose une interface de recherche intuitive.

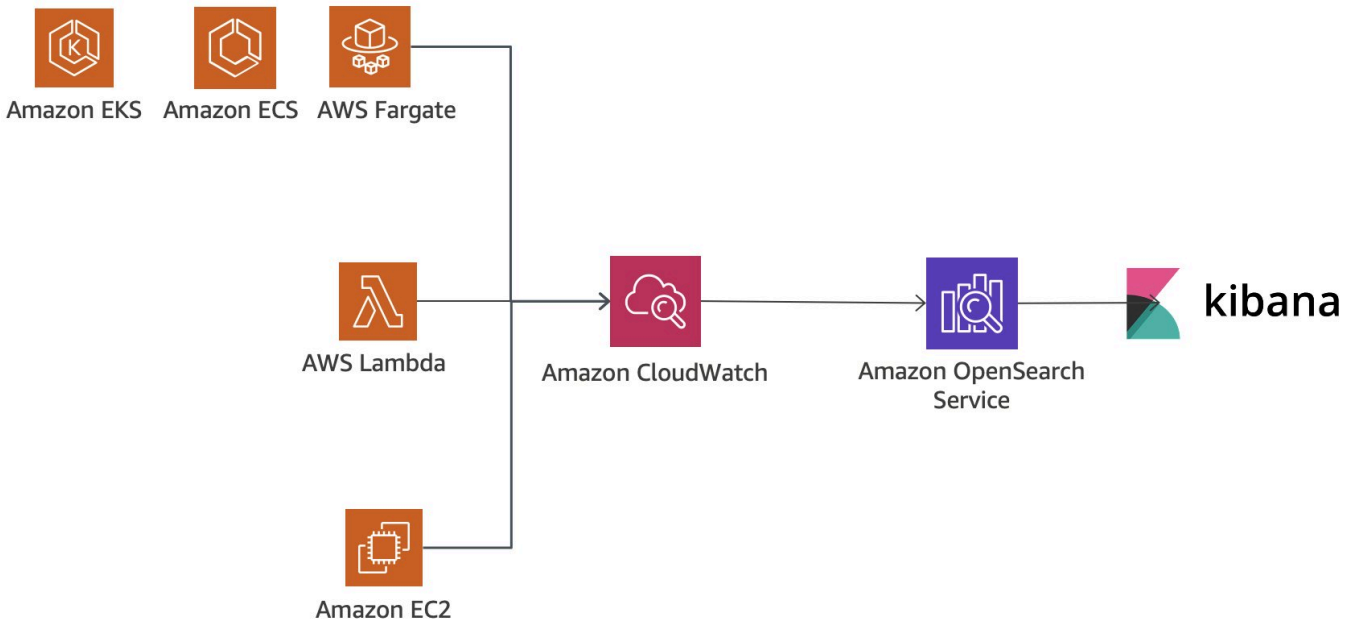

Figure 14 : Analyse des journaux avec Amazon OpenSearch Service

Autres options d'analyse

Pour une analyse plus approfondie des journaux, Amazon Redshift, un service d'entrepôt de données entièrement géré, et Quick Suite, un service de business intelligence évolutif, proposent des solutions efficaces. QuickSight fournit une connectivité facile à divers services de AWS données tels que Redshift, RDS, Aurora, EMR, DynamoDB, Amazon S3 et Kinesis, simplifiant ainsi l'accès aux données.

CloudWatch Les journaux peuvent diffuser des entrées de journal vers Amazon Data Firehose, un service de diffusion de données en temps réel. QuickSight utilise ensuite les données stockées dans Redshift pour une analyse, des rapports et une visualisation complets.

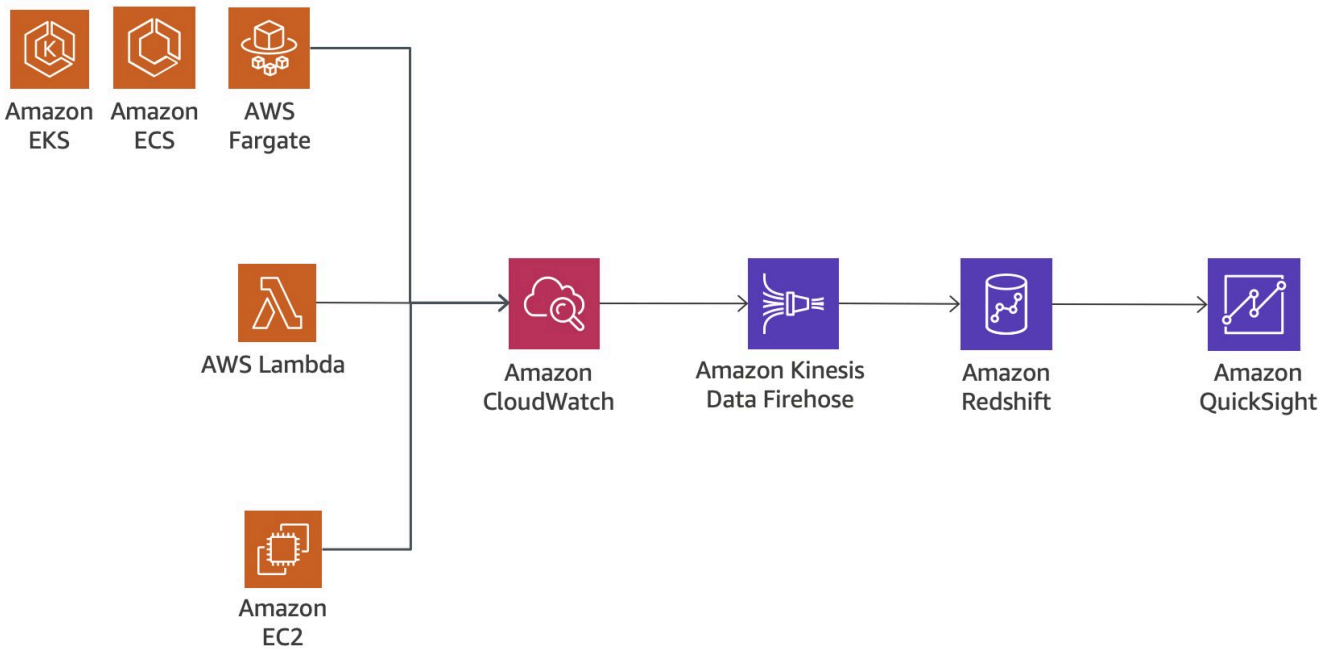

Figure 15 : Analyse des journaux avec Amazon Redshift et Quick Suite

De plus, lorsque les journaux sont stockés dans des buckets S3, un service de stockage d'objets, les données peuvent être chargées dans des services tels que Redshift ou EMR, une plateforme de mégadonnées basée sur le cloud, permettant une analyse approfondie des données de journal stockées.

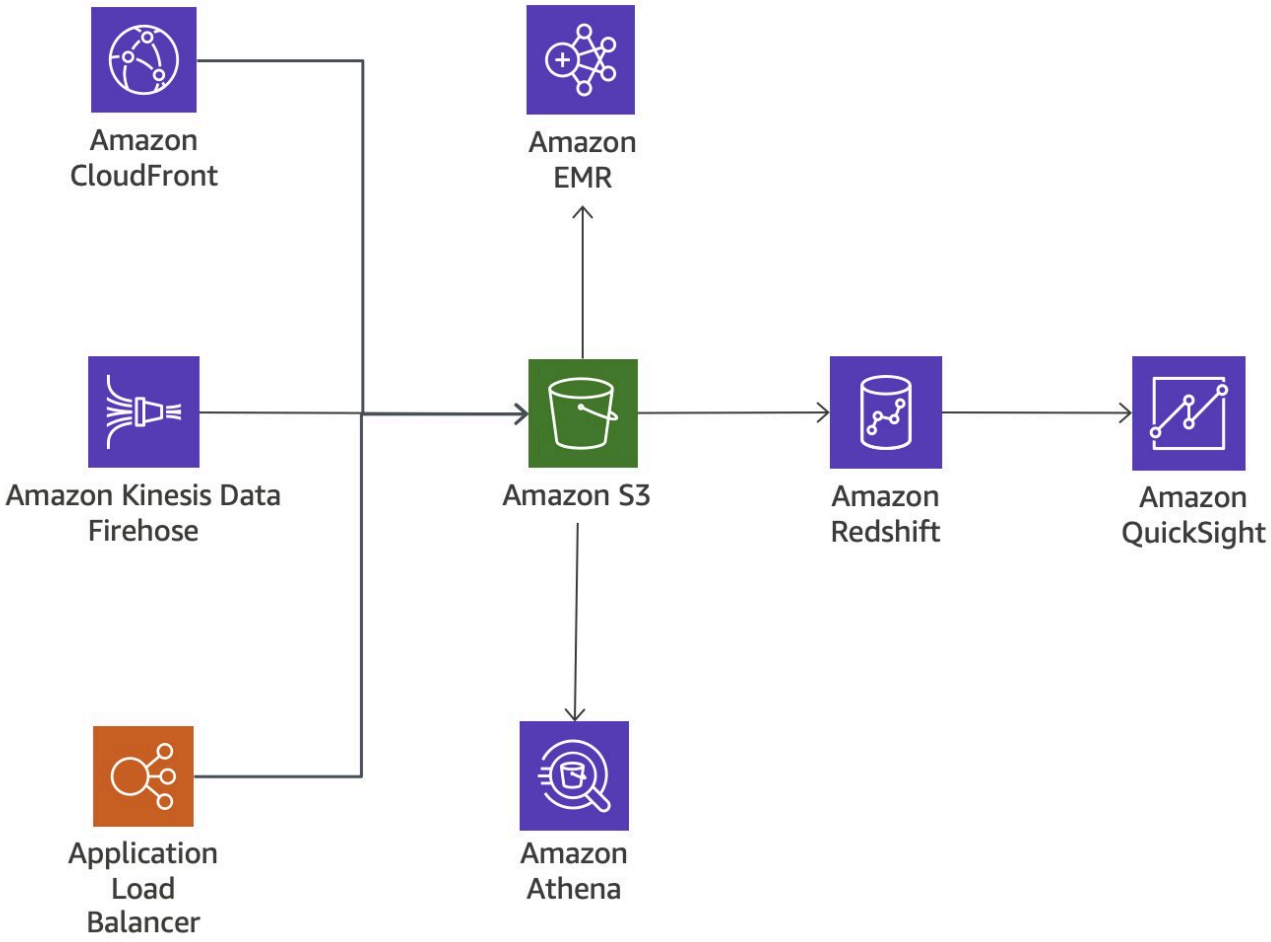

Figure 16 : Rationalisation de l'analyse des journaux : des AWS services à QuickSight

Gérer le bavardage dans les communications par microservices

Le chattiness fait référence à une communication excessive entre les microservices, ce qui peut entraîner une inefficacité en raison de l'augmentation de la latence du réseau. Il est essentiel de gérer efficacement le bavardage pour que le système fonctionne bien.

Certains outils clés pour gérer le chattiness sont REST APIs, HTTP et APIs gRPC. APIs REST APIs propose une gamme de fonctionnalités avancées telles que les clés d'API, la régulation par client, la validation des demandes, AWS WAF l'intégration ou les points de terminaison d'API privés. APIs Les protocoles HTTP sont conçus avec un minimum de fonctionnalités et sont donc proposés à un prix inférieur. Pour plus de détails sur ce sujet et une liste des fonctionnalités principales disponibles dans REST APIs et HTTP APIs, voir Choisir entre REST APIs et HTTP APIs.

Les microservices utilisent souvent REST sur HTTP pour communiquer en raison de son utilisation généralisée. Mais dans les situations de volume élevé, la surcharge de REST peut entraîner des problèmes de performances. C'est parce que la communication utilise le protocole TCP handshake qui est requis pour chaque nouvelle demande. Dans de tels cas, l'API gRPC est un meilleur choix. gRPC réduit la latence car elle autorise plusieurs requêtes via une seule connexion TCP. gRPC prend également en charge le streaming bidirectionnel, permettant aux clients et aux serveurs d'envoyer et de recevoir des messages en même temps. Cela permet une communication plus efficace, en particulier pour les transferts de données importants ou en temps réel.

Si le chat persiste malgré le choix du bon type d'API, il peut être nécessaire de réévaluer votre architecture de microservices. La consolidation des services ou la révision de votre modèle de domaine peuvent réduire les bavardages et améliorer l'efficacité.

Utilisation de protocoles et de mise en cache

Les microservices utilisent souvent des protocoles tels que gRPC et REST pour la communication (voir la section Mécanismes de communication précédente) gRPC utilise HTTP/2 pour le transport, tandis que REST utilise généralement HTTP/1.1. gRPC utilise des tampons de protocole pour la sérialisation, tandis que REST utilise généralement JSON ou XML. Pour réduire la latence et la surcharge de communication, la mise en cache peut être appliquée. Des services tels qu'Amazon ElastiCache ou la couche de mise en cache d'API Gateway peuvent contribuer à réduire le nombre d'appels entre microservices.

Audit

Dans une architecture de microservices, il est essentiel d'avoir une visibilité sur les actions des utilisateurs dans tous les services. AWS fournit des outils tels que AWS CloudTrail, qui enregistre tous les appels d' AWS API effectués et AWS CloudWatch qui est utilisé pour capturer les journaux des applications. Cela vous permet de suivre les modifications et d'analyser le comportement dans l'ensemble de vos microservices. Amazon EventBridge peut réagir rapidement aux modifications du système, en informant les bonnes personnes ou même en démarrant automatiquement des flux de travail pour résoudre les problèmes.

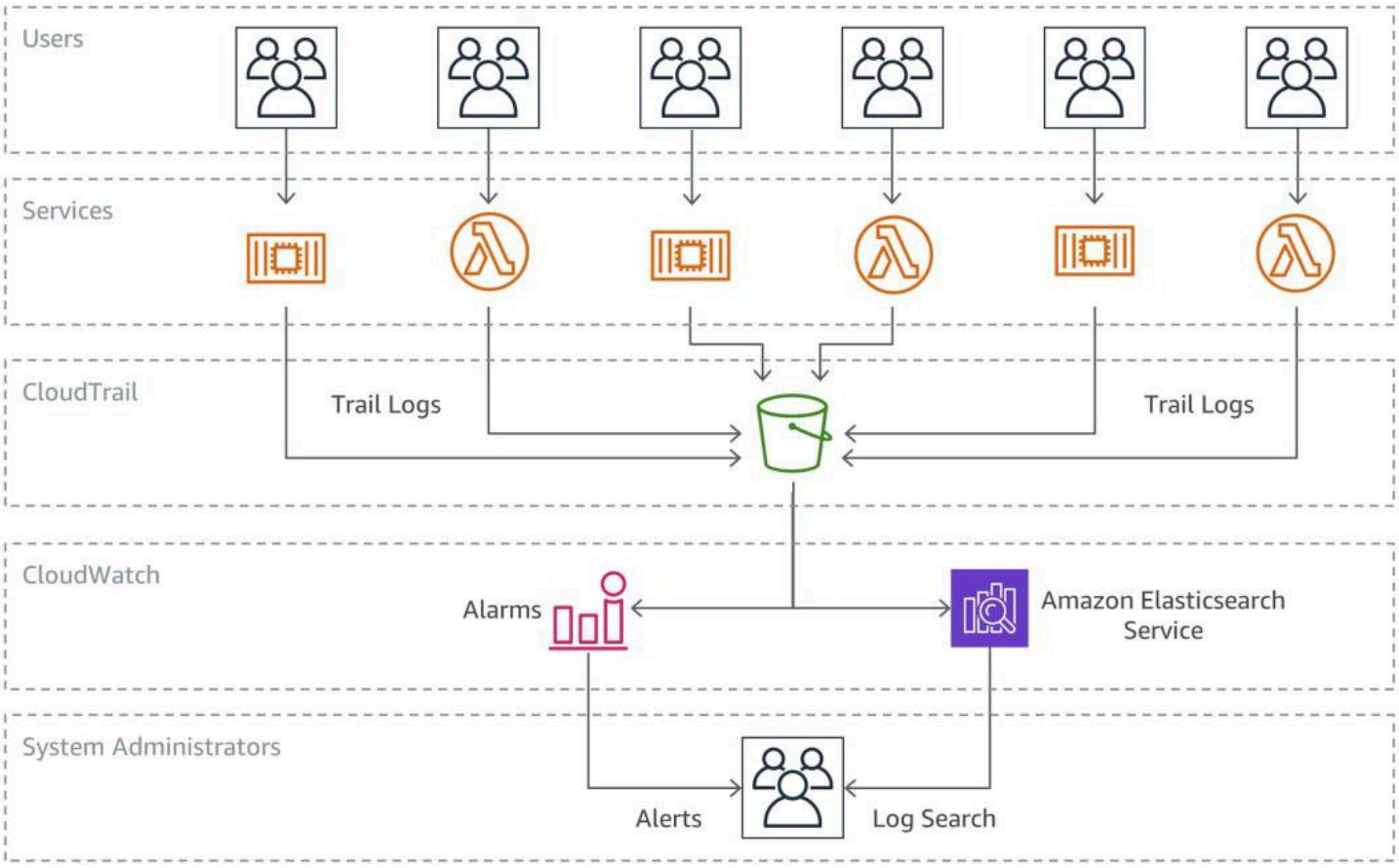

Figure 17 : Audit et correction sur l'ensemble de vos microservices

Inventaire des ressources et gestion du changement

Dans un environnement de développement agile où les configurations d'infrastructure évoluent rapidement, l'audit et le contrôle automatisés sont essentiels. AWS Config Rules fournir une approche gérée pour surveiller ces modifications sur l'ensemble des microservices. Ils permettent de

définir des politiques de sécurité spécifiques qui détectent, suivent et envoient automatiquement des alertes en cas de violation des politiques.

Par exemple, si la configuration d'une API Gateway dans un microservice est modifiée pour accepter le trafic HTTP entrant au lieu des seules requêtes HTTPS, une AWS Config règle prédéfinie peut détecter cette violation de sécurité. Il enregistre la modification à des fins d'audit et déclenche une notification SNS, rétablissant ainsi l'état de conformité.

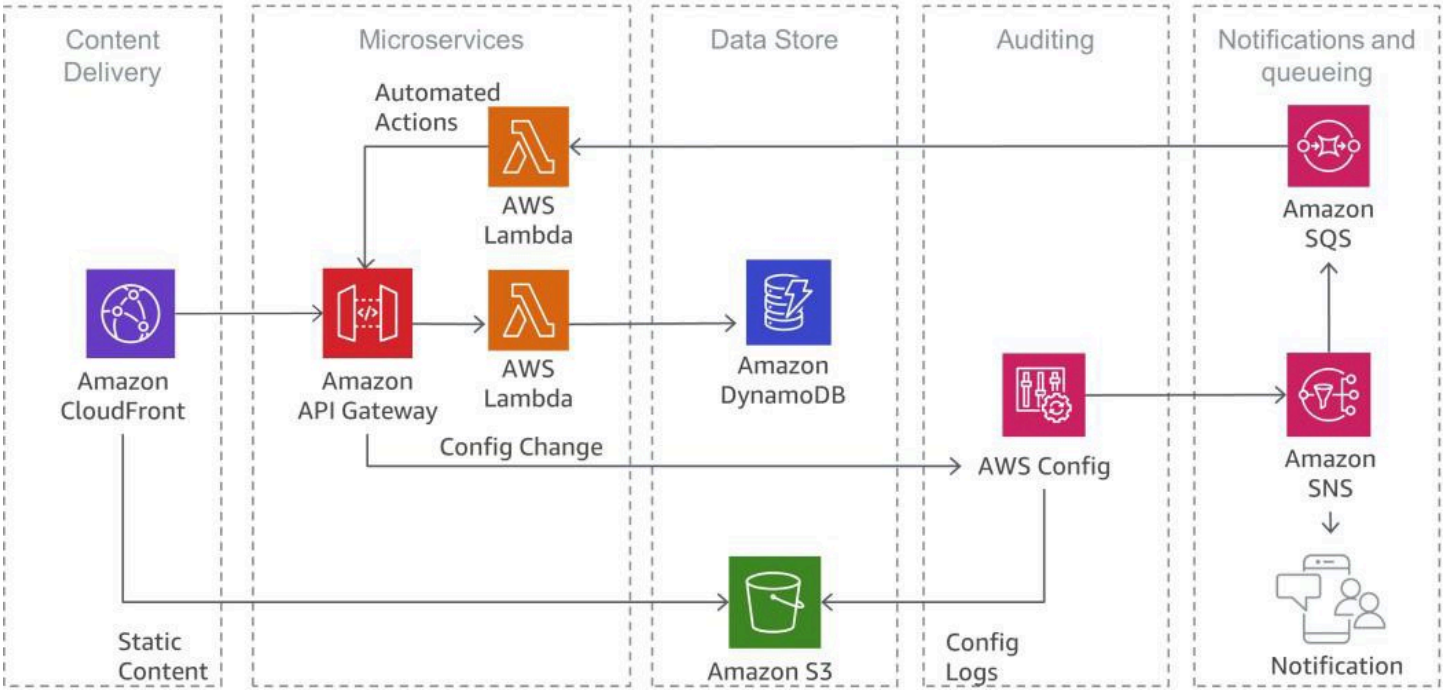

Figure 18 : Détection des violations de sécurité avec AWS Config

Conclusion

L'architecture des microservices, une approche de conception polyvalente qui fournit une alternative aux systèmes monolithiques traditionnels, aide à faire évoluer les applications, à accélérer le développement et à favoriser la croissance organisationnelle. Grâce à son adaptabilité, il peut être mis en œuvre à l'aide de conteneurs, d'approches sans serveur ou d'un mélange des deux, en l'adaptant à des besoins spécifiques.

Cependant, ce n'est pas une one-size-fits-all solution. Chaque cas d'utilisation nécessite une évaluation méticuleuse compte tenu de l'augmentation potentielle de la complexité architecturale et des exigences opérationnelles. Mais lorsqu'ils sont abordés de manière stratégique, les avantages des microservices peuvent largement l'emporter sur ces défis. La clé réside dans la planification proactive, en particulier dans les domaines de l'observabilité, de la sécurité et de la gestion du changement.

Il est également important de noter qu'au-delà des microservices, il existe des cadres architecturaux totalement différents, tels que les architectures d'IA générative telles que Retrieval Augmented Generation (RAG), offrant une gamme d'options répondant au mieux à vos besoins.

AWS, avec sa suite robuste de services gérés, permet aux équipes de créer des architectures de microservices efficaces et de minimiser efficacement la complexité. Ce livre blanc a pour objectif de vous guider à travers les AWS services concernés et la mise en œuvre de modèles clés. L'objectif est de vous fournir les connaissances nécessaires pour exploiter la puissance des microservices AWS, afin de tirer parti de leurs avantages et de transformer votre parcours de développement d'applications.

Collaborateurs

Les personnes et organisations suivantes ont contribué à l'élaboration du présent document :

- Sascha Möllering, architecture des solutions, Amazon Web Services
- Christian Müller, architecture des solutions, Amazon Web Services
- Matthias Jung, architecture des solutions, Amazon Web Services
- Peter Dalbhanjan, architecture des solutions, Amazon Web Services
- Peter Chapman, architecture des solutions, Amazon Web Services
- Christoph Kassen, architecture des solutions, Amazon Web Services
- Umair Ishaq, architecture des solutions, Amazon Web Services
- Rajiv Kumar, architecture des solutions, Amazon Web Services
- Ramesh Dwarakanath, architecture des solutions, Amazon Web Services
- Andrew Watkins, architecture des solutions, Amazon Web Services
- Yann Stoneman, architecture des solutions, Amazon Web Services
- Mainak Chaudhuri, architecture des solutions, Amazon Web Services
- Gaurav Acharya, architecture des solutions, Amazon Web Services

Historique du document

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

Modification	Description	Date
Mise à jour majeure	Ajout d'informations sur AWS Customer Carbon Footprint Tool EventBridge, Amazon AWS AppSync (GraphQL), AWS Lambda Layers, Lambda, Large Language Models () SnapStartLLMs, Amazon Managed Streaming for Apache Kafka (MSK), Amazon Managed Workflows for Apache Airflow (MWAA), Amazon VPC Lattice,. AWS AppConfig Ajout d'une section séparée sur l'optimisation des coûts et la durabilité.	31 juillet 2023
Mises à jour mineures	Well-Architected a été ajouté au résumé.	13 avril 2022
Livre blanc mis à jour	Intégration d'Amazon EventBridge, AWS, AMP OpenTelemetry, AMG, Container Insights, modificat ions mineures du texte.	9 novembre 2021
Mises à jour mineures	Mise en page ajustée	30 avril 2021
Mises à jour mineures	Modifications mineures du texte.	1 août 2019
Livre blanc mis à jour	Intégration d'Amazon EKS, d'AWS Fargate, d'Amazon	1er juin 2019

	MQ, d'AWS, d'AWS PrivateLink App Mesh et d'AWS Cloud Map	
Livre blanc mis à jour	Intégration des flux d'événements AWS Step Functions, AWS X-Ray et ECS.	1 septembre 2017
Publication initiale	Implémentation des microservices sur AWS publiée.	1er décembre 2016

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

Avis

Il incombe aux clients de procéder à une évaluation indépendante des informations contenues dans le présent document. Ce document : (a) est fourni à titre informatif uniquement, (b) représente les offres de AWS produits et les pratiques actuelles, qui sont susceptibles d'être modifiées sans préavis, et (c) ne crée aucun engagement ni aucune assurance de la part de AWS ses filiales, fournisseurs ou concédants de licence. AWS les produits ou services sont fournis « tels quels » sans garanties, déclarations ou conditions d'aucune sorte, qu'elles soient explicites ou implicites. Les responsabilités et obligations AWS de ses clients sont régies par AWS des accords, et ce document ne fait partie d'aucun accord conclu entre AWS et ses clients et ne les modifie pas.

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le AWS glossaire dans la Glossaire AWS référence.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.