Whitepaper da AWS

Implementação de microsserviços na AWS

Implementação de microsserviços na AWS: Whitepaper da AWS

Table of Contents

Implementação de microsserviços na AWS

Data de publicação: 9 de novembro de 2021 (Histórico do documento e colaboradores)

Resumo

Os microsserviços são uma abordagem arquitetônica e organizacional de desenvolvimento de software criada para acelerar ciclos de implantação, promover inovação e propriedade, melhorar a capacidade de manutenção e a escalabilidade de aplicações de software e escalar organizações que fornecem software e serviços usando um método ágil que ajuda as equipes a trabalhar de forma independente umas das outras. Com uma abordagem de microsserviços, o software é composto de pequenos serviços que se comunicam por meio de Interfaces do Programa da Aplicação (APIs) bem definidas que podem ser implantadas de forma independente. Esses serviços são de propriedade de pequenas equipes autônomas. Essa abordagem ágil é essencial para escalar a organização com êxito.

Três padrões comuns foram observados quando os clientes da AWS criam microsserviços: orientado por API, orientado por eventos e transmissão de dados. Este whitepaper apresenta todas as três abordagens e resume as características comuns dos microsserviços, aborda os principais desafios de criação de microsserviços e descreve como as equipes de produto podem utilizar a Amazon Web Services (AWS) para superar esses desafios.

Devido à natureza bastante envolvente de vários tópicos discutidos neste whitepaper, como armazenamento de dados, comunicação assíncrona e descoberta de serviços, é recomendável considerar os requisitos e casos de uso específicos de suas aplicações, bem como as orientações fornecidas, antes de fazer escolhas arquitetônicas.

Introdução

As arquiteturas de microsserviços não são uma abordagem totalmente nova da engenharia de software, mas uma combinação de vários conceitos bem-sucedidos e comprovados, como:

- Desenvolvimento de software ágil
- Arquiteturas orientadas a serviços
- Projeto voltado a APIs
- Integração e entrega contínuas (CI/CD)

Em muitos casos, os padrões de projeto da aplicação de 12 fatores são usados nos microsserviços.

Este whitepaper primeiro descreve os diferentes aspectos de uma arquitetura de microsserviços altamente escalável e tolerante a falhas (interface do usuário, microsserviços, implementação e datastore) e como criá-la na AWS usando tecnologias de contêiner. Em seguida, recomendamos os serviços da AWS para implementar uma arquitetura de microsserviços sem servidor típica e reduzir a complexidade operacional.

A arquitetura sem servidor é definida como um modelo operacional baseado nos seguintes princípios:

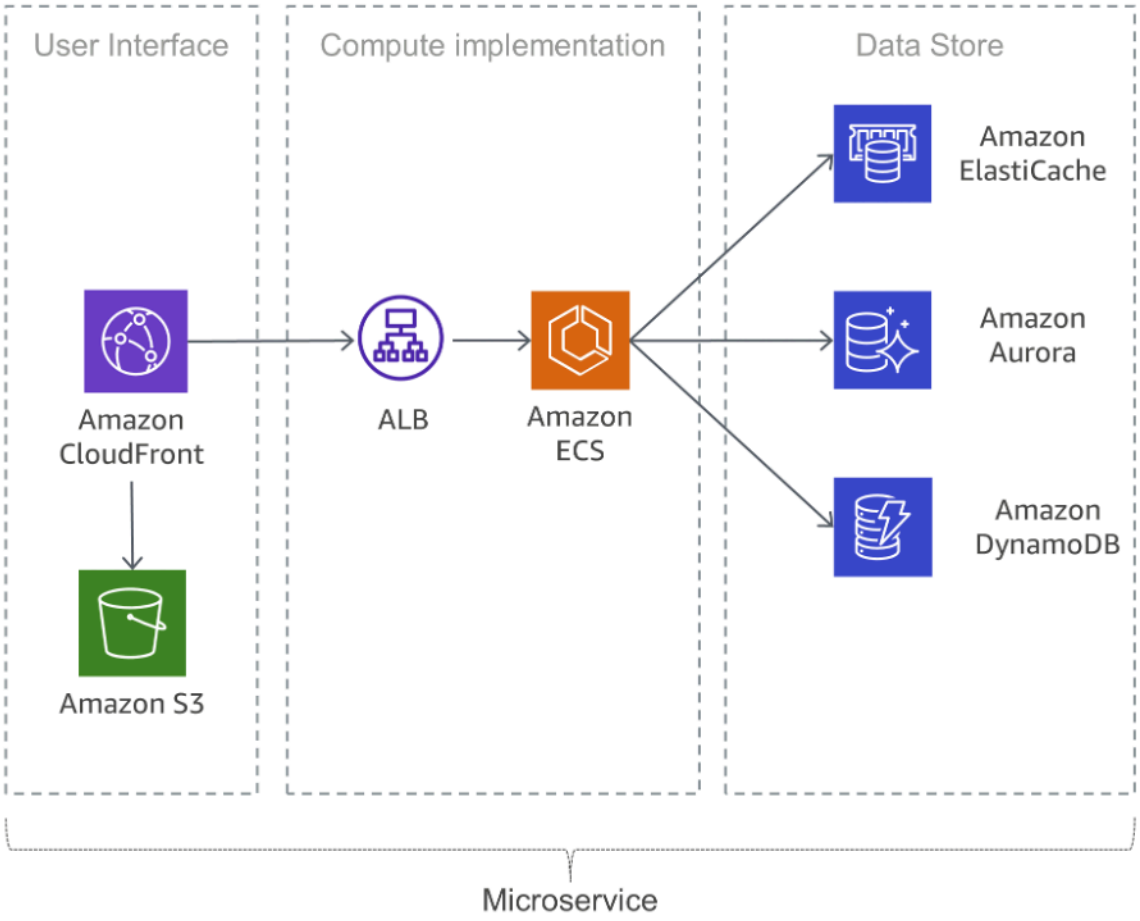
- Nenhuma infraestrutura para provisionar ou gerenciar
- Escalabilidade automática por unidade de consumo
- Modelo de faturamento pague pelo valor
- Recursos incorporados de disponibilidade e tolerância a falhas

Por fim, este whitepaper aborda o sistema geral e discute os aspectos entre serviços de uma arquitetura de microsserviços, como monitoramento e auditoria distribuídos, consistência de dados e comunicação assíncrona.

Este whitepaper se concentra apenas nas workloads executadas na Nuvem AWS. Ele não abrange cenários híbridos ou estratégias de migração. Para obter mais informações sobre migração, consulte o whitepaper Metodologia de migração de contêineres.

Arquitetura de microsserviços na AWS

As aplicações monolíticas típicos eram criadas usando camadas diferentes: uma camada de interface do usuário (IU), uma camada de negócios e uma camada de persistência. A ideia central de uma arquitetura de microsserviços é dividir funcionalidades em verticais coerentes, não por meio de camadas de tecnologia, mas implementando determinado domínio. A figura a seguir mostra uma arquitetura de referência para uma aplicação típica de microsserviços na AWS.



Aplicação típica de microsserviços na AWS

Tópicos

- Interface do usuário
- Microsserviços
- Datastore

Interface do usuário

As aplicações Web modernas costumam usar frameworks JavaScript para implementar uma aplicação de uma única página que se comunica com uma API de transferência de estado representacional (REST) ou RESTful. O conteúdo Web estático pode ser enviado usando o Amazon Simple Storage Service (S3) e o Amazon CloudFront.

Como os clientes de um microsserviço são atendidos pelo local da borda mais próximo e obtêm respostas de um cache ou servidor de proxy com conexões otimizadas para a origem, as latências podem ser consideravelmente reduzidas. No entanto, os microsserviços executados próximos uns dos outros não se beneficiam de uma rede de entrega de conteúdo. Em alguns casos, essa abordagem pode até aumentar a latência. A implementação de outros mecanismos de armazenamento em cache para reduzir interações desnecessárias e minimizar latências é uma melhor prática. Para obter mais informações, consulte o tópico the section called “Interações desnecessárias”.

Microsserviços

As APIs são a porta de entrada dos microsserviços, o que significa que elas servem como ponto de entrada para a lógica de aplicações por trás de um conjunto de interfaces programáticas, normalmente uma API de serviços da Web REST completa. Essa API aceita e processa chamadas de clientes e pode implementar funcionalidades como gerenciamento de tráfego, filtragem de solicitações, roteamento, armazenamento em cache, autenticação e autorização.

Implementação de microsserviços

A AWS integrou componentes básicos que apoiam o desenvolvimento de microsserviços. Duas abordagens comuns são o uso do AWS Lambda e de contêineres do Docker com o AWS Fargate.

Com o AWS Lambda, basta carregar o código que o Lambda se encarrega de tudo o que é necessário para executar e escalar o código a fim de atender à curva de demanda real com alta disponibilidade. Não é necessário administrar a infraestrutura. O Lambda comporta várias linguagens de programação e pode ser acionado de outros serviços da AWS ou ser chamado diretamente de qualquer aplicação Web ou aplicativo móvel. Uma das maiores vantagens do AWS Lambda é a maior agilidade: você pode se concentrar na lógica de negócios porque a segurança e a escalabilidade são gerenciadas pela AWS. A abordagem opinativa do Lambda promove a plataforma escalável.

Uma abordagem comum para reduzir esforços operacionais nas implantações é a implantação baseada em contêineres. Tecnologias de contêiner como o Docker se tornaram mais apreciadas nos últimos anos devido a benefícios como portabilidade, produtividade e eficiência. A curva de aprendizado dos contêineres pode ser íngreme e você precisa pensar em monitoramento e correções de segurança para as imagens do Docker. O Amazon Elastic Container Service (Amazon ECS) e o Amazon Elastic Kubernetes Service (Amazon EKS) eliminam a necessidade de instalar, operar e escalar sua própria infraestrutura de gerenciamento de clusters. Com chamadas de API simples, é possível executar e interromper aplicações compatíveis com o Docker, consultar o estado completo do cluster e acessar diversos recursos familiares, como grupos de segurança, balanceadores de carga, volumes do Amazon Elastic Block Store (Amazon EBS) e funções do AWS Identity and Access Management (IAM).

O AWS Fargate é um mecanismo de computação sem servidor para contêineres que funciona tanto com o Amazon ECS quanto o Amazon EKS. Com o Fargate, você não precisa mais se preocupar em provisionar recursos computacionais suficientes para aplicativos de contêineres. O Fargate pode executar dezenas de milhares de contêineres e escalar facilmente para executar os mais exigentes aplicativos de missão crítica.

O Amazon ECS comporta estratégias e restrições de posicionamento de contêineres para personalizar como o Amazon ECS posiciona e finaliza tarefas. Uma restrição de posicionamento de tarefas é uma regra que é considerada durante o posicionamento da tarefa. Você pode associar atributos, que são basicamente pares de chave-valor, às instâncias de contêiner e usar uma restrição para posicionar tarefas com base nesses atributos. Por exemplo, você pode usar restrições para posicionar microsserviços específicos de acordo com o tipo ou o recurso da instância, como instâncias baseadas em GPU.

O Amazon EKS executa versões atualizadas do software de código aberto Kubernetes, o que permite usar todos os plugins e ferramentas existentes da comunidade Kubernetes. Os aplicativos executados no Amazon EKS são totalmente compatíveis com aplicativos executados em qualquer ambiente Kubernetes padrão em datacenters locais ou em nuvens públicas. O Amazon EKS integra o IAM com o Kubernetes, permitindo que você registre entidades do IAM com o sistema de autenticação nativo do Kubernetes. Não há necessidade de configurar manualmente as credenciais para autenticação com os ambientes de gerenciamento do Kubernetes. A integração do IAM permite que você use o IAM para autenticar diretamente no próprio ambiente de gerenciamento e fornecer diferentes níveis de acesso ao endpoint público de seu ambiente de gerenciamento do Kubernetes.

As imagens do Docker usadas no Amazon ECS e no Amazon EKS podem ser armazenadas no Amazon Elastic Container Registry (Amazon ECR.) O Amazon ECR elimina a necessidade de operar e escalar a infraestrutura necessária para operar o registro do contêiner.

A integração e a entrega contínuas (CI/CD) são práticas recomendadas e parte essencial de uma iniciativa de DevOps que viabiliza mudanças rápidas no software e, ao mesmo tempo, mantém a estabilidade e a segurança do sistema. No entanto, isso está fora do escopo deste whitepaper. Para obter mais informações, consulte o whitepaper Prática de integração e entrega contínuas na AWS.

Links privados

OAWS PrivateLink é uma tecnologia altamente disponível e escalável que permite que você conecte de forma privada sua VPC aos serviços compatíveis da AWS, a serviços hospedados por outras contas da AWS (serviços do endpoint da VPC) e serviços compatíveis de parceiros do AWS Marketplace. Você não precisa de recursos como um gateway da Internet, dispositivo de conversão de endereço de rede, endereço IP público, conexão do AWS Direct Connect ou conexão VPN para se comunicar com o serviço. O tráfego entre sua VPC e o serviço não deixa a rede da Amazon.

Os links privados são uma ótima maneira de aumentar o isolamento e a segurança da arquitetura de microsserviços. Por exemplo, um microsserviço pode ser implantado em uma VPC totalmente separada, coberto por um balanceador de carga e exposto a outros microsserviços por meio de um endpoint do AWS PrivateLink. Com essa configuração, usando o AWS PrivateLink, o tráfego de rede de e para o microsserviço nunca atravessa a Internet pública. Um dos casos de uso para esse tipo de isolamento é a conformidade regulatória para serviços que lidam com dados sigilosos, como PCI, HIPPA e EU/US Privacy Shield. Além disso, o AWS PrivateLink permite conectar microsserviços em diferentes contas e Amazon VPCs, sem a necessidade de regras de firewall, definições de caminho ou tabelas de rotas, o que simplifica o gerenciamento de rede. Utilizando o PrivateLink, os provedores de software como serviço (SaaS) e os ISVs podem oferecer soluções baseadas em microsserviços com isolamento operacional completo, bem como acesso seguro.

Datastore

O datastore é usado para persistir os dados necessários para os microsserviços. Caches na memória, como Memcached ou Redis, são datastores populares para dados de sessão. A AWS oferece ambas as tecnologias como parte do serviço gerenciado Amazon ElastiCache.

O posicionamento de um cache entre servidores de aplicação e um banco de dados é um mecanismo comum para reduzir a carga de leitura do banco de dados. Isso, por sua vez, pode permitir o uso de recursos para comportar mais gravações. Os caches também aprimoram a latência.

Os bancos de dados relacionais ainda são muito apreciados para armazenar dados estruturados e objetos de negócio. A AWS oferece seis mecanismos de banco de dados (Microsoft SQL Server, Oracle, MySQL, MariaDB, PostgreSQL e Amazon Aurora) como serviços gerenciados por meio do Amazon Relational Database Service (Amazon RDS).

No entanto, os bancos de dados relacionais não são projetados para escalar indefinidamente. Por isso, pode ser muito difícil e demorado aplicar técnicas para comportar um alto número de consultas.

Os bancos de dados NoSQL foram projetados para priorizar a escalabilidade, a performance e a disponibilidade em relação à consistência dos bancos de dados relacionais. Um elemento importante é que, normalmente, os bancos de dados NoSQL não aplicam um esquema rigoroso. Os dados são distribuídos por meio de partições, que podem ser escaladas horizontalmente e são recuperadas por meio de chaves de partição.

Como os microsserviços individuais são projetados para executar bem uma única função, eles normalmente têm um modelo de dados simplificado que pode ser bastante adequado para persistência no NoSQL. É importante compreender que bancos de dados NoSQL têm padrões de acesso diferentes dos bancos de dados relacionais. Por exemplo, não é possível associar tabelas. Se a associação for necessária, a lógica deverá ser implementada no aplicativo. Você pode usar o Amazon DynamoDB para criar uma tabela de banco de dados capaz de armazenar e recuperar qualquer quantidade de dados, assim como atender a todos os níveis de solicitação de tráfego. O DynamoDB oferece performance abaixo de dez milissegundos. No entanto, alguns casos de uso demandam tempos de resposta em microssegundos. O Amazon DynamoDB Accelerator (DAX) oferece recursos de armazenamento em cache para acesso aos dados.

O DynamoDB também oferece um recurso de escalabilidade automática para ajustar dinamicamente a capacidade de throughput de acordo com o tráfego real. No entanto, há casos em que o planejamento de capacidade é difícil ou impossível devido a grandes picos de atividade de curta duração no aplicativo. Para tais situações, o DynamoDB disponibiliza uma opção sob demanda com definição de preço simples no modelo de pagamento por solicitação. O DynamoDB sob demanda consegue atender instantaneamente a milhares de solicitações por segundo, sem precisar de planejamento de capacidade.

Redução da complexidade operacional

A arquitetura descrita anteriormente neste whitepaper já está usando serviços gerenciados, mas as instâncias do Amazon Elastic Compute Cloud (Amazon EC2) ainda precisam ser gerenciadas. É possível reduzir ainda mais os esforços operacionais necessários para executar, manter e monitorar microsserviços usando uma arquitetura totalmente sem servidor.

Tópicos

Implementação de APIs

Tarefas como arquitetura, implantação, monitoramento, melhoria contínua e manutenção de uma API podem ser demoradas. Às vezes, versões diferentes de APIs precisam ser executadas a fim de garantir a compatibilidade com versões anteriores de todos os clientes. Os diferentes estágios do ciclo de desenvolvimento (por exemplo, desenvolvimento, teste e produção) multiplicam ainda mais os esforços operacionais.

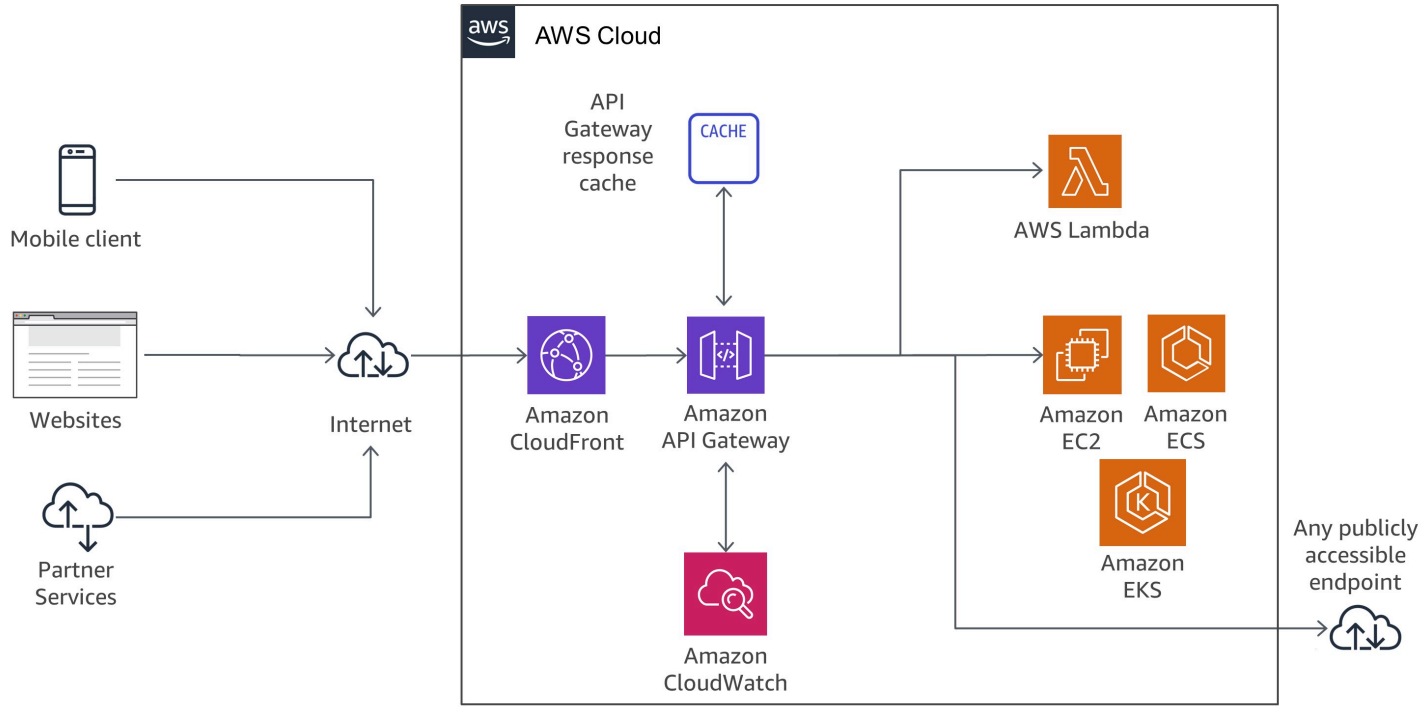
A autorização é um recurso essencial para todas as APIs, mas sua criação costuma ser complexa e envolve trabalho repetitivo. Quando uma API é publicada e se torna bem-sucedida, o próximo desafio é gerenciar, monitorar e monetizar o ecossistema de desenvolvedores de terceiros que usam a API.

Outros recursos e desafios importantes incluem o controle de utilização de solicitações para proteger os serviços de backend, o armazenamento em cache de respostas de API, o processamento da transformação de solicitações e respostas e a geração de definições e documentação de API usando ferramentas como o Swagger.

O Amazon API Gateway lida com esses desafios e reduz a complexidade operacional de criação e manutenção de APIs RESTful. O API Gateway permite que você crie APIs de forma programática importando definições do Swagger por meio da API da AWS ou do Console de Gerenciamento da AWS. O API Gateway funciona como uma porta de entrada para qualquer aplicação Web executada

no Amazon EC2, no Amazon ECS, no AWS Lambda ou em qualquer ambiente on-premises. Basicamente, o API Gateway permite a execução de APIs sem a necessidade de gerenciar servidores.

A figura a seguir mostra como o API Gateway processa as chamadas de API e interage com outros componentes. As solicitações de dispositivos móveis, sites ou outros serviços de backend são roteadas para o ponto de presença (PoP) do CloudFront mais próximo a fim de minimizar a latência e proporcionar uma ótima experiência do usuário.



Fluxo de chamadas do API Gateway

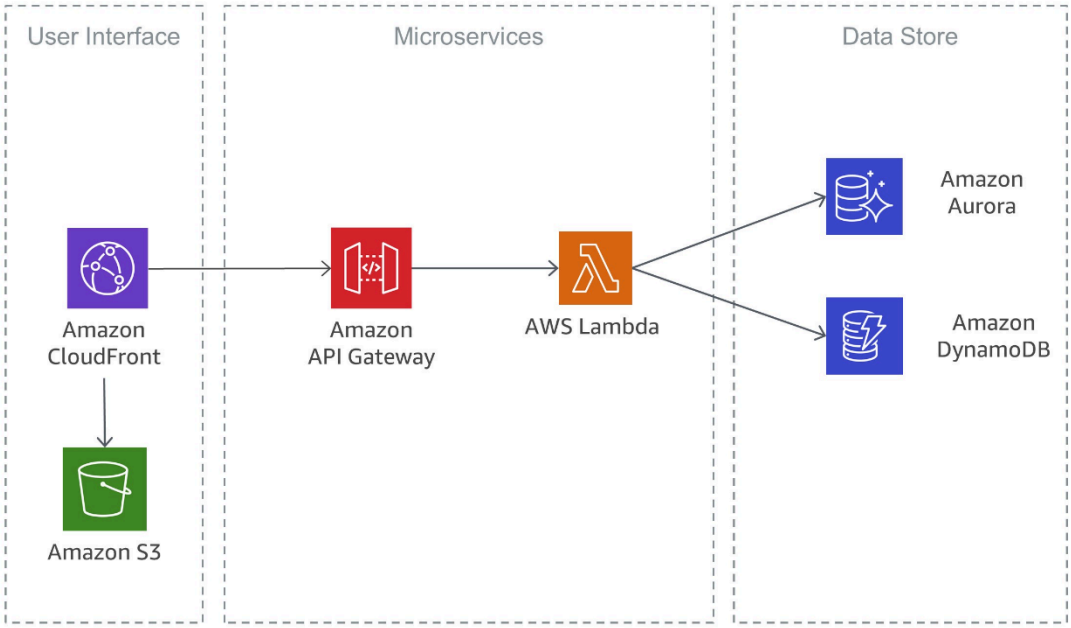
Microsserviços sem servidor

"Nenhum servidor é mais fácil de gerenciar do que nenhum servidor."

Erradicar os servidores é uma ótima forma de eliminar a complexidade operacional.

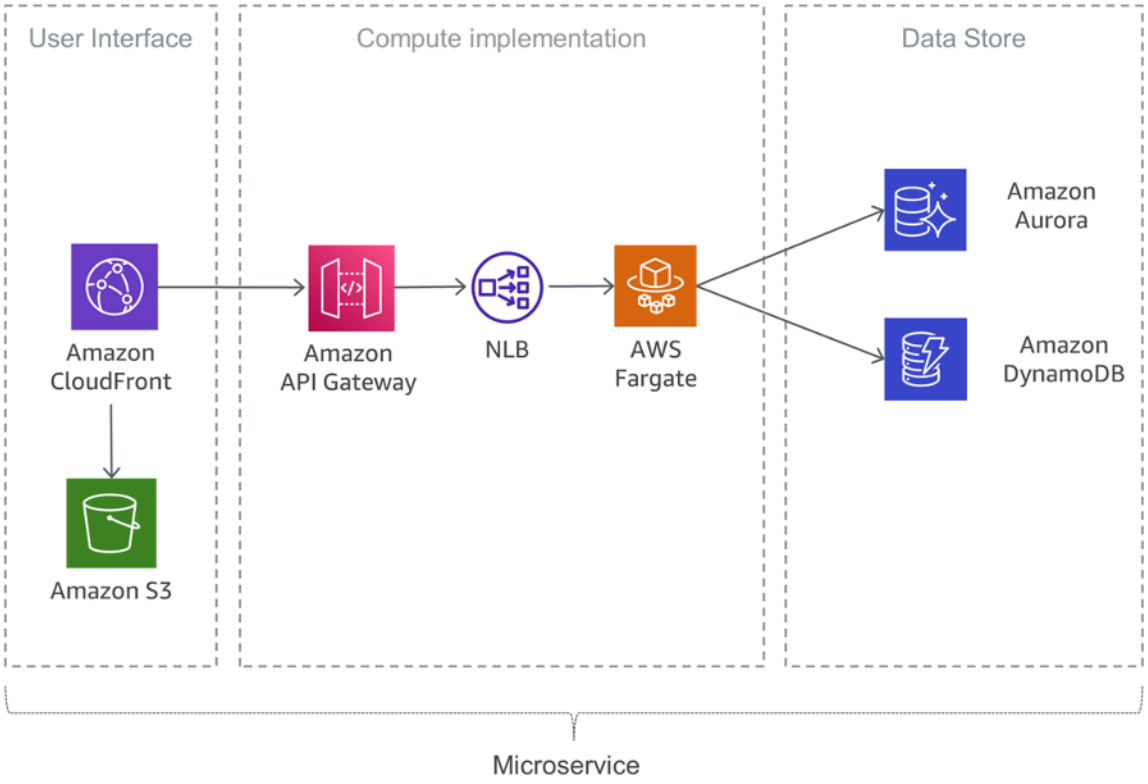
O Lambda é altamente integrado ao API Gateway. A capacidade de fazer chamadas síncronas do API Gateway para o Lambda permite a criação de aplicações totalmente sem servidor e é descrita em detalhes em no Guia do desenvolvedor do Amazon API Gateway.

A figura a seguir mostra a arquitetura de um microsserviço sem servidor com o AWS Lambda. O serviço inteiro é criado com base em serviços gerenciados, o que elimina a sobrecarga de projetar a arquitetura para oferecer escala e alta disponibilidade, bem como os esforços operacionais para executar e monitorar a infraestrutura subjacente do microsserviço.



Arquitetura sem servidor usando o AWS Lambda

Uma implementação semelhante que também é baseada em serviços sem servidor é mostrada na figura a seguir. Nessa arquitetura, contêineres do Docker são usados com o Fargate. Assim, não é preciso se preocupar com a infraestrutura subjacente. Além do DynamoDB, também é usado o Amazon Aurora Serverless, uma configuração sob demanda com escalabilidade automática para o Amazon Aurora (edição compatível com o MySQL). Nessa configuração, o banco de dados inicia, encerra e aumenta ou reduz automaticamente a escala da capacidade na vertical de acordo com as necessidades da aplicação.



Microsserviço sem servidor usando o Fargate

Recuperação de desastres

Conforme mencionado anteriormente na introdução deste whitepaper, aplicações típicas de microsserviços são implementadas usando os padrões da aplicação de 12 fatores. A seção Processos menciona que “Os processos de 12 fatores são sem estado e não compartilham nada. Todos os dados que precisam persistir devem ser armazenados em um serviço de backup com estado, normalmente um banco de dados”.

Para uma arquitetura de microsserviços típica, isso significa que o foco principal da recuperação de desastres deve estar nos serviços posteriores que mantêm o estado da aplicação. Por exemplo, eles podem ser sistemas de arquivos, bancos de dados ou filas. Ao criar uma estratégia de recuperação de desastres, as organizações geralmente se preparam para o objetivo de tempo de recuperação (RTO) e o objetivo de ponto de recuperação (RPO).

O objetivo de tempo de recuperação (RTO) é o atraso máximo aceitável entre a interrupção e a restauração do serviço. Esse objetivo, definido pela organização, determina o que é considerado uma janela de tempo aceitável quando o serviço está indisponível.

O objetivo de ponto de recuperação (RPO) é o tempo máximo aceitável desde o último ponto de recuperação de dados. Esse objetivo, definido pela organização, determina o que é considerado uma perda aceitável de dados entre o último ponto de recuperação e a interrupção do serviço.

Para obter mais informações, consulte o whitepaper Recuperação de desastres de workloads na AWS: recuperação na nuvem.

Alta disponibilidade

Esta seção analisa mais detalhadamente a alta disponibilidade para diferentes opções de computação.

O Amazon EKS executa instâncias de controle e plano de dados do Kubernetes entre várias zonas de disponibilidade para garantir alta disponibilidade. O Amazon EKS detecta e substitui automaticamente instâncias não íntegras do ambiente de gerenciamento, além de fornecer atualizações de versão e correções automatizadas para elas. Esse ambiente de gerenciamento é composto de pelo menos dois nós de servidor de API e três nós etcd que são executados em três zonas de disponibilidade dentro de uma região. O Amazon EKS usa a arquitetura das Regiões da AWS para manter a alta disponibilidade.

O Amazon ECR hospeda imagens em uma arquitetura altamente disponível e de alta performance, permitindo a implantação confiável de imagens para aplicações conteinerizadas em todas as zonas de disponibilidade. O Amazon ECR funciona com o Amazon EKS, o Amazon ECS e o AWS Lambda, o que simplifica o desenvolvimento para o fluxo de trabalho de produção.

O Amazon ECS é um serviço regional que simplifica a execução de contêineres de uma forma altamente disponível em várias zonas de disponibilidade em uma Região da AWS. O Amazon ECS inclui várias estratégias de programação que posicionam os contêineres em clusters com base na necessidade de recursos (por exemplo, CPU ou RAM) e nos requisitos de disponibilidade.

O AWS Lambda executa sua função em várias zonas de disponibilidade a fim de garantir que ela esteja disponível para processar eventos no caso de interrupção do serviço em uma única zona. Se você configurar a função para se conectar a uma nuvem privada virtual (VPC) em sua conta, especifique sub-redes em várias zonas de disponibilidade para garantir alta disponibilidade.

Implantação de aplicações baseadas no Lambda

Você pode usar o AWS CloudFormation para especificar, implantar e configurar aplicações sem servidor.

O AWS Serverless Application Model (AWS SAM) é uma maneira conveniente de definir aplicações sem servidor. O AWS SAM é nativamente compatível com o CloudFormation e define uma sintaxe simplificada para expressar recursos sem servidor. Para implantar sua aplicação, basta especificar os respectivos recursos de que você precisa, bem como as políticas de permissões correspondentes em um modelo do CloudFormation, empacotar os artefatos da implantação e implantar o modelo. Baseado no AWS SAM, o SAM Local é uma ferramenta do AWS Command Line Interface (AWS CLI) que fornece um ambiente para você desenvolver, testar e analisar suas aplicações sem servidor localmente antes de carregá-las para o tempo de execução do Lambda. Você pode usar o AWS SAM Local para criar um ambiente de teste local que simula o ambiente de tempo de execução da AWS.

Componentes de sistemas distribuídos

Depois de examinar como a AWS pode resolver os problemas de microsserviços específicos, o foco passa a ser os desafios da interação entre esses serviços, como descoberta de serviços, consistência de dados, comunicação assíncrona e auditoria e monitoramento distribuídos.

Tópicos

Descoberta de serviços

Um dos principais desafios nas arquiteturas de microsserviços é permitir que os serviços descubram outros serviços e interajam entre si. Além de dificultar a comunicação entre os serviços, as características distribuídas das arquiteturas de microsserviços também apresentam outros desafios, como a verificação da integridade desses sistemas e o anúncio da disponibilidade de novos aplicativos. Além disso, você deve decidir como e onde armazenar informações de metadados, como dados de configuração, que podem ser usados pelos aplicativos. Nesta seção, examinamos várias técnicas para executar a descoberta de serviços na AWS em arquiteturas baseadas em microsserviços.

Descoberta de serviços baseada em DNS

Agora, o Amazon ECS inclui a descoberta de serviços integrados, um recurso que facilita a descoberta de serviços conteinerizados e a conexão entre eles.

Antes, para assegurar a descoberta e a conexão dos serviços entre si, era preciso configurar e executar um sistema de descoberta de serviços próprio, baseado no Amazon Route 53, no AWS Lambda e no ECS Event Stream, ou conectar todos os serviços a um balanceador de carga.

O Amazon ECS cria e gerencia um registro de nomes de serviço usando a API Auto Naming do Route 53. Os nomes são mapeados automaticamente para um conjunto de registros de DNS, o que

permite fazer referência a um serviço por nome no código e criar consultas de DNS para que o nome seja resolvido em tempo de execução para o endpoint do serviço. Você pode especificar condições de verificação de integridade na definição de tarefa de um serviço e o Amazon ECS garante que apenas endpoints de serviço íntegros sejam retornados por uma pesquisa de serviços.

Além disso, você também pode utilizar a descoberta unificada de serviços gerenciados pelo Kubernetes. Para habilitar essa integração, a AWS contribuiu para o External DNS, um projeto de incubadora do Kubernetes.

Outra opção é usar os recursos do AWS Cloud Map. O AWS Cloud Map amplia a capacidade das APIs de nomeação automática fornecendo um registro de serviços para recursos como protocolos de Internet (IPs), localizadores de recursos uniformes (URLs) e nomes de recurso da Amazon (ARNs), bem como um mecanismo de descoberta de serviços baseado em APIs com uma propagação de alterações mais rápida e a possibilidade de usar atributos para restringir o conjunto de recursos descobertos. Os recursos de nomeação automática do Route 53 são atualizados automaticamente para o AWS Cloud Map.

Software de terceiros

Uma abordagem diferente para implementar a descoberta de serviços é a utilização de software de terceiros, como o HashiCorp Consul, o etcd ou o Netflix Eureka. Os três exemplos são armazenamentos de chave-valor distribuídos e confiáveis. Para o HashiCorp Consul, há um Quick Start da AWS que define um ambiente flexível e escalável na Nuvem AWS e inicia automaticamente o HashiCorp Consul com a configuração escolhida.

Malhas de serviços

Em uma arquitetura de microsserviços avançada, a aplicação real pode consistir em centenas ou mesmo milhares de serviços. Muitas vezes, a parte mais complexa da aplicação não são os serviços propriamente ditos, mas a comunicação entre eles. As malhas de serviço são uma camada adicional para processar a comunicação entre serviços, que é responsável por monitorar e controlar o tráfego em arquiteturas de microsserviços. Isso possibilita que tarefas como a descoberta de serviços sejam tratadas inteiramente nessa camada.

Normalmente, uma malha de serviços é dividida em um plano de dados e um plano de controle. O plano de dados consiste em um conjunto de proxies inteligentes que são implantados com o código do aplicativo como um proxy complementar especial que intercepta toda a comunicação de rede entre os microsserviços. O plano de controle é responsável pela comunicação entre os proxies.

As malhas de serviço são transparentes, o que significa que os desenvolvedores de aplicativos não precisam se preocupar com essa camada adicional nem alterar o código dos aplicativos atuais. O AWS App Mesh é uma malha de serviços que fornece redes para aplicações com o objetivo de possibilitar a comunicação entre os serviços por meio de vários tipos de infraestrutura de computação. O App Mesh padroniza a forma de comunicação dos serviços, oferecendo total visibilidade e garantindo alta disponibilidade para as aplicações.

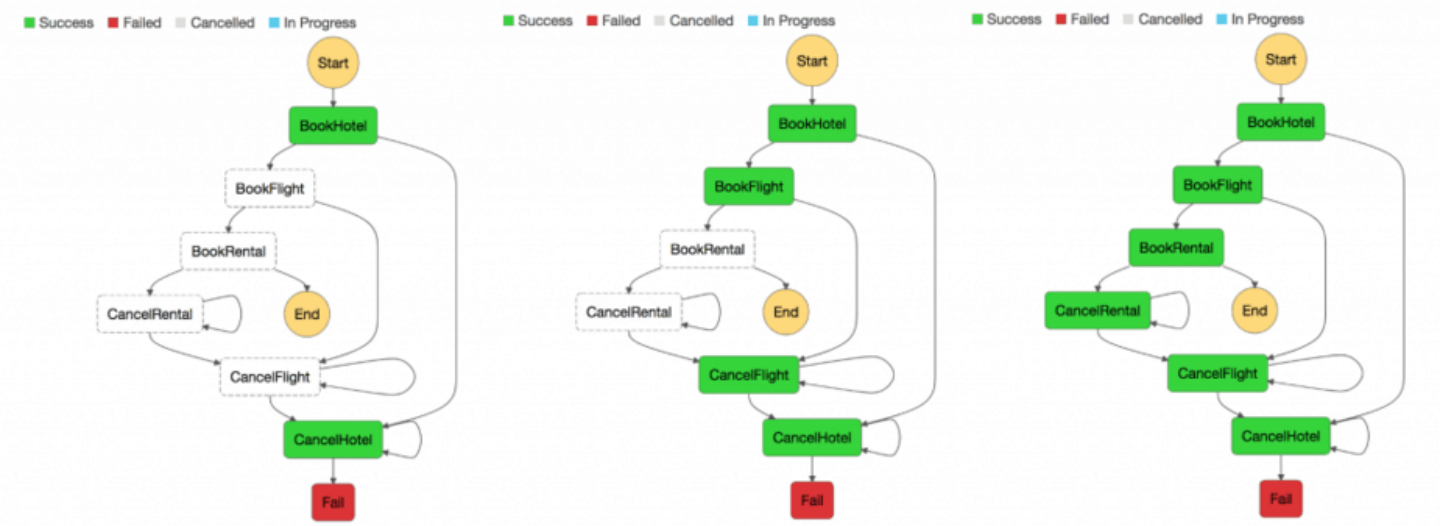
Você pode usar o AWS App Mesh com microsserviços novos ou existentes em execução no AWS Fargate, no Amazon ECS, no Amazon EKS e em Kubernetes autogerenciados na AWS. O App Mesh pode monitorar e controlar comunicações para microsserviços executados em clusters, sistemas de orquestração ou VPCs como uma única aplicação, sem nenhuma alteração de código.

Gerenciamento de dados distribuídos

Normalmente, os aplicativos monolíticos são apoiados por um grande banco de dados relacional, que define um único modelo de dados comum a todos os componentes do aplicativo. Em uma abordagem de microsserviços, esse banco de dados central seria um obstáculo para o objetivo de criar componentes descentralizados e independentes. Cada componente de microsserviços deve ter a sua própria camada de persistência de dados.

No entanto, o gerenciamento de dados distribuído traz novos desafios. Em consequência do teorema CAP, as arquiteturas de microsserviços distribuídos comprometem inerentemente a consistência para ganhar performance e precisam adotar a consistência final.

Em um sistema distribuído, as transações empresariais podem abranger vários microsserviços. Como eles não podem executar uma única transação ACID, você pode ter de lidar com execuções parciais. Nesse caso, precisaríamos de alguma lógica de controle para executar novamente transações já processadas. Normalmente, o padrão Saga distribuído é usado para essa finalidade. No caso de falha de uma transação empresarial, o Saga orquestra uma série de transações de compensação que desfazem as alterações realizadas pelas transações anteriores. O AWS Step Functions facilita a implementação de um coordenador de execução do Saga, como mostrado na próxima figura.



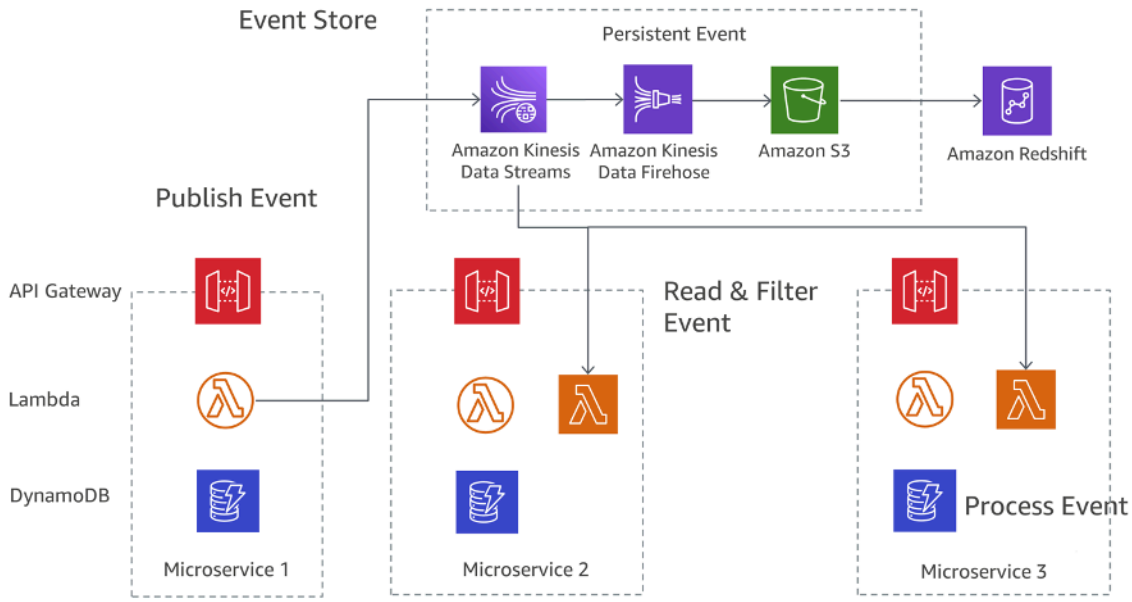
Coordenador de execução do Saga

A criação de um armazenamento centralizado de dados de referência essenciais que é selecionado por meio de ferramentas e procedimentos de gerenciamento de dados principais oferece aos microsserviços uma forma de sincronizar os dados essenciais e, possivelmente, reverter o estado. O uso do Lambda com eventos programados do Amazon CloudWatch Events permite criar um mecanismo simples de limpeza e eliminação de duplicidades.

É muito comum as mudanças de estado afetarem mais de um microsserviço. Nesses casos, a origem de eventos demonstrou ser um padrão útil. O principal conceito da origem de eventos é representar e manter todas as mudanças de aplicativos como um registro de eventos. Em vez de persistir o estado do aplicativo, os dados são armazenados como um stream de eventos. Os sistemas de registro de transações de banco de dados em log e de controle de versões são dois exemplos conhecidos de origem de eventos. A origem de eventos oferece alguns benefícios: o estado de qualquer point-in-time pode ser determinado e reconstruído. Além disso, ela gera naturalmente uma trilha de auditoria persistente e facilita a depuração.

No contexto de arquiteturas de microsserviços, a origem de eventos permite desacoplar partes diferentes de uma aplicação usando um padrão de publicação/assinatura. Além disso, ela alimenta os mesmos dados de eventos em diferentes modelos de dados para microsserviços separados. Muitas vezes, a origem de eventos é usada em conjunto com o padrão Command, Query, Responsibility, Segregation (CQRS – comando, consulta, responsabilidade, segregação) para desacoplar workloads de leitura das de gravação e otimizar a performance, a escalabilidade e a segurança de ambas. Em sistemas de gerenciamento de dados tradicionais, os comandos e consultas são executados usando o mesmo repositório de dados.

A figura a seguir mostra como o padrão de origem de eventos pode ser implementado na AWS. O Amazon Kinesis Data Streams atua como o componente principal do armazenamento de eventos centralizado, que captura alterações nas aplicações como eventos e os conservam no Amazon S3. A figura retrata três microsserviços diferentes constituídos dos serviços Amazon API Gateway, AWS Lambda e Amazon DynamoDB. As setas indicam o fluxo de eventos: quando o Microsserviço 1 experimenta uma alteração de estado de evento, ele publica um evento gravando uma mensagem no Kinesis Data Streams. Todos os microsserviços executam sua própria aplicação do Kinesis Data Streams no AWS Lambda, que lê uma cópia da mensagem, filtra essa mensagem com base na relevância para o microsserviço e possivelmente a encaminha para processamento posterior. Se sua função retornar um erro, o Lambda tentará executar novamente o lote até que o processamento ocorra com êxito ou os dados expirem. Para evitar fragmentos paralisados, você pode configurar o mapeamento da origem do evento para uma nova execução com um tamanho de lote menor, limitar o número de novas tentativas ou descartar os registros muito antigos. Para reter eventos descartados, você pode configurar o mapeamento da origem do evento para enviar detalhes sobre lotes com falha a uma fila do Amazon Simple Queue Service (Amazon SQS) ou a um tópico do Amazon Simple Notification Service (Amazon SNS).



Padrão de origem de eventos na AWS

O Amazon S3 armazena de forma resiliente todos os eventos de todos os microsserviços e é a única fonte de verdade para fins de depuração, recuperação do estado da aplicação ou auditoria de alterações na aplicação. Há dois principais motivos pelos quais os registros podem ser entregues

mais de uma vez à sua aplicação do Kinesis Data Streams: novas tentativas de produtor e novas tentativas de consumidor. Sua aplicação precisa prever e tratar devidamente o processamento de registros individuais várias vezes.

Gerenciamento de configuração

Em uma arquitetura de microsserviços típica com dezenas de serviços diferentes, cada um deles precisa acessar vários serviços e componentes de infraestrutura posteriores que expõem os dados ao serviço. Exemplos podem ser filas de mensagens, bancos de dados e outros microsserviços. Um dos principais desafios é configurar cada serviço de forma consistente para fornecer informações sobre a conexão com serviços e infraestrutura posteriores. Além disso, a configuração também deve conter informações sobre o ambiente em que o serviço está operando, e reiniciar a aplicação para usar novos dados de configuração não deve ser necessário.

O terceiro princípio dos padrões da aplicação de doze fatores aborda este tópico: “A aplicação de 12 fatores armazena a configuração em variáveis de ambiente (geralmente abreviado como env vars ou env)”. Para o Amazon ECS, as variáveis de ambiente podem ser passadas para o contêiner por meio do parâmetro de definição de contêiner de ambiente que mapeia para a opção `--env` para docker run. As variáveis de ambiente podem ser passadas em massa para seus contêineres por meio do parâmetro de definição de contêiner `environmentFiles` para listar um ou mais arquivos contendo as variáveis de ambiente. O arquivo deve ser hospedado no Amazon S3. No AWS Lambda, o tempo de execução disponibiliza variáveis de ambiente para o código e define outras variáveis de ambiente que contêm informações sobre a função e solicitação de chamada. Para o Amazon EKS, você pode definir variáveis de ambiente no campo env do manifesto de configuração do pod correspondente. O uso de um ConfigMap é uma maneira diferente de utilizar variáveis de ambiente.

Comunicação assíncrona e sistemas de mensagem leves

A comunicação em aplicações monolíticas tradicionais é simples: uma parte da aplicação utiliza chamadas de métodos ou um mecanismo interno de distribuição de eventos para se comunicar com as outras partes. Se a mesma aplicação for implementada usando microsserviços desacoplados, a comunicação entre as diferentes partes da aplicação precisará ser implementada usando comunicação de rede.

Comunicação baseada em REST

O protocolo HTTP/S é a maneira mais popular para implementar a comunicação síncrona entre microsserviços. Na maioria dos casos, as APIs RESTful usam o HTTP como camada de transporte.

O estilo de arquitetura REST se baseia em comunicação stateless, interfaces uniformes e métodos padrão.

Com alguns cliques no API Gateway, você pode criar uma API que atua como uma porta de entrada para que as aplicações acessem dados, lógica de negócios ou funcionalidades de seus serviços de backend. Os desenvolvedores de APIs podem criar APIs que acessem serviços da AWS ou outros serviços da Web, bem como dados armazenados na Nuvem AWS. Um objeto de API definido com o serviço API Gateway é um grupo de recursos e métodos.

Um recurso é um objeto tipificado dentro do domínio de uma API e pode ter associado um modelo de dados ou relacionamentos a outros recursos. Cada recurso pode ser configurado para responder a um ou mais métodos, ou seja, verbos HTTP padrão como GET, POST ou PUT. As APIs REST podem ser implantadas em diferentes estágios, bem como versionadas e também clonadas para novas versões.

O API Gateway administra todas as tarefas envolvidas na aceitação e no processamento de até centenas de milhares de chamadas de API simultâneas, como gerenciamento de tráfego, controle de autorização e acesso, monitoramento e gerenciamento de versões de API.

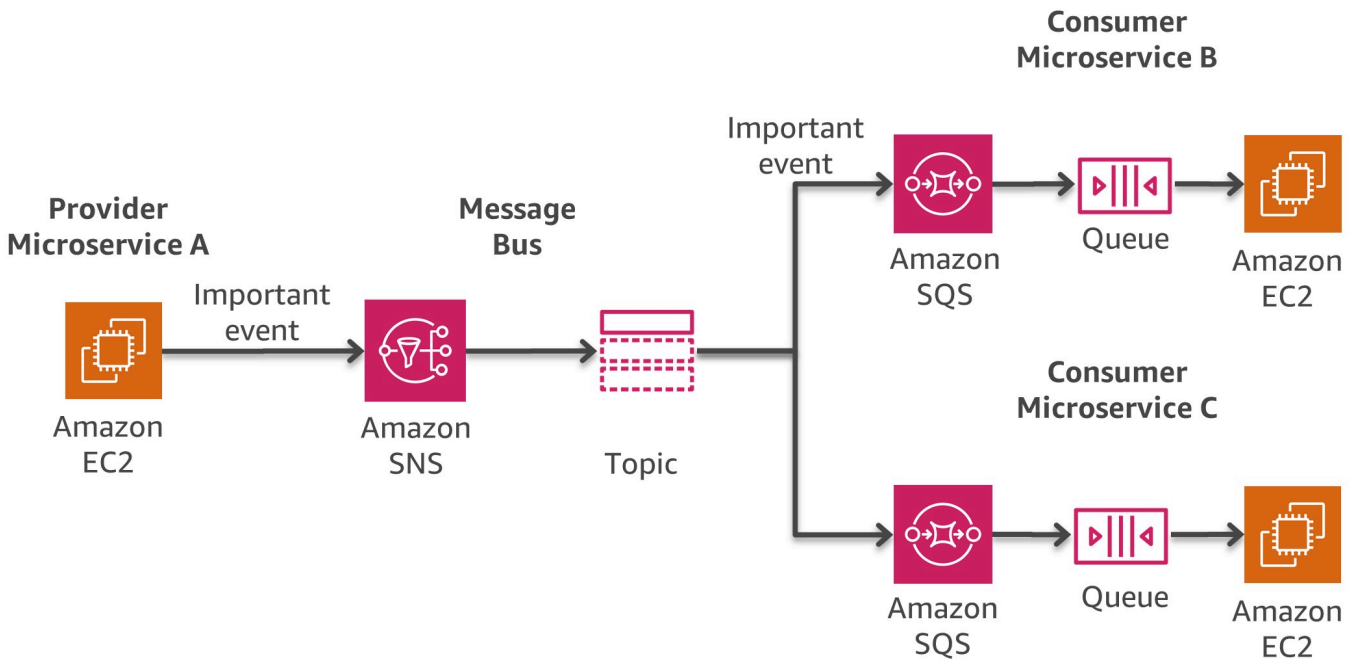
Sistemas de mensagens assíncronas e passagem de eventos

Outro padrão para implementar a comunicação entre microsserviços é a passagem de mensagens. Os serviços se comunicam pela troca de mensagens por meio de uma fila. Uma das principais vantagens desse estilo de comunicação é que não é necessário ter uma descoberta de serviços e os serviços têm baixo acoplamento.

Os sistemas síncronos têm alto acoplamento. Ou seja, um problema em uma dependência síncrona em uma aplicação posterior afeta imediatamente os chamadores anteriores. As novas tentativas de execução dos chamadores anteriores podem rapidamente se distribuir e ampliar os problemas.

Dependendo de requisitos específicos, como protocolos, a AWS oferece diferentes serviços que ajudam a implementar esse padrão. Uma possível implementação usa filas do Amazon Simple Queue Service (Amazon SQS) ou o Amazon Simple Notification Service (Amazon SNS).

Os dois serviços trabalham estreitamente em conjunto: o Amazon SNS permite que as aplicações enviem mensagens a vários assinantes por meio de um mecanismo de push. Usando o Amazon SNS e o Amazon SQS em conjunto, uma mensagem pode ser entregue a vários consumidores. A figura a seguir demonstra a integração do Amazon SNS e do Amazon SQS.



Padrão de barramento de mensagens na AWS

Ao assinar uma fila do Amazon SQS para um tópico do SNS, basta publicar uma mensagem no tópico e o Amazon SNS envia uma mensagem à fila do Amazon SQS assinada. A mensagem contém o assunto e a mensagem publicados no tópico, bem como as informações de metadados em formato JSON.

Outra opção para criar arquiteturas orientadas por eventos com origens de eventos abrangendo aplicações internas, aplicações SaaS de terceiros e serviços da AWS em escala é o Amazon EventBridge. O EventBridge, um serviço de barramento de eventos totalmente gerenciado, recebe eventos de fontes diferentes, identifica um destino com base em uma regra de roteamento e entrega dados quase em tempo real a esse destino, inclusive ao AWS Lambda, Amazon SNS e Amazon Kinesis Streams, entre outros. Um evento de entrada também pode ser personalizado, por transformador de entrada, antes da entrega.

Para desenvolver aplicações orientadas por eventos de uma maneira significativamente mais rápida, os registros de esquema do EventBridge coletam e organizam esquemas, inclusive para todos os eventos gerados pelos serviços da AWS. Os clientes também podem definir esquemas personalizados ou usar uma opção de inferência de esquema para descobrir esquemas automaticamente. No entanto, em suma, uma possível compensação para todos esses recursos

é um valor de latência relativamente maior para a entrega do EventBridge. Além disso, a taxa de transferência e as cotas padrão do EventBridge podem exigir um aumento, por meio de uma solicitação de suporte, com base no caso de uso.

Uma estratégia de implementação diferente, baseada no Amazon MQ, poderá ser usada se o software existente utilizar APIs e protocolos de sistema de mensagens abertos padrão, como JMS, NMS, AMQP, STOMP, MQTT e WebSocket. O Amazon SQS expõe uma API personalizada, ou seja, se tiver uma aplicação existente que você pretende migrar, por exemplo, de um ambiente on-premises para a AWS, será necessário alterar o código. Com o Amazon MQ, isso não é necessário em muitos casos.

O Amazon MQ gerencia a administração e a manutenção do ActiveMQ, um agente de mensagens conhecido de código aberto. A infraestrutura subjacente é provisionada automaticamente para oferecer alta disponibilidade e resiliência de mensagens e favorecer a confiabilidade das aplicações.

Orquestração e gerenciamento de estados

A natureza distribuída dos microsserviços faz com que a orquestração de fluxos de trabalho envolvendo vários microsserviços seja um desafio. Os desenvolvedores podem querer adicionar código de orquestração diretamente aos serviços deles. Essa abordagem deve ser evitada porque introduz um acoplamento mais alto e dificulta a rápida substituição de serviços específicos.

Você pode usar o AWS Step Functions para criar aplicações com componentes individuais que executam uma função distinta. O Step Functions fornece uma máquina de estado que oculta a complexidade da orquestração de serviços, como tratamento de erros, serialização e paralelização. Isso permite escalar e alterar rapidamente os aplicativos, evitando código de coordenação adicional dentro dos serviços.

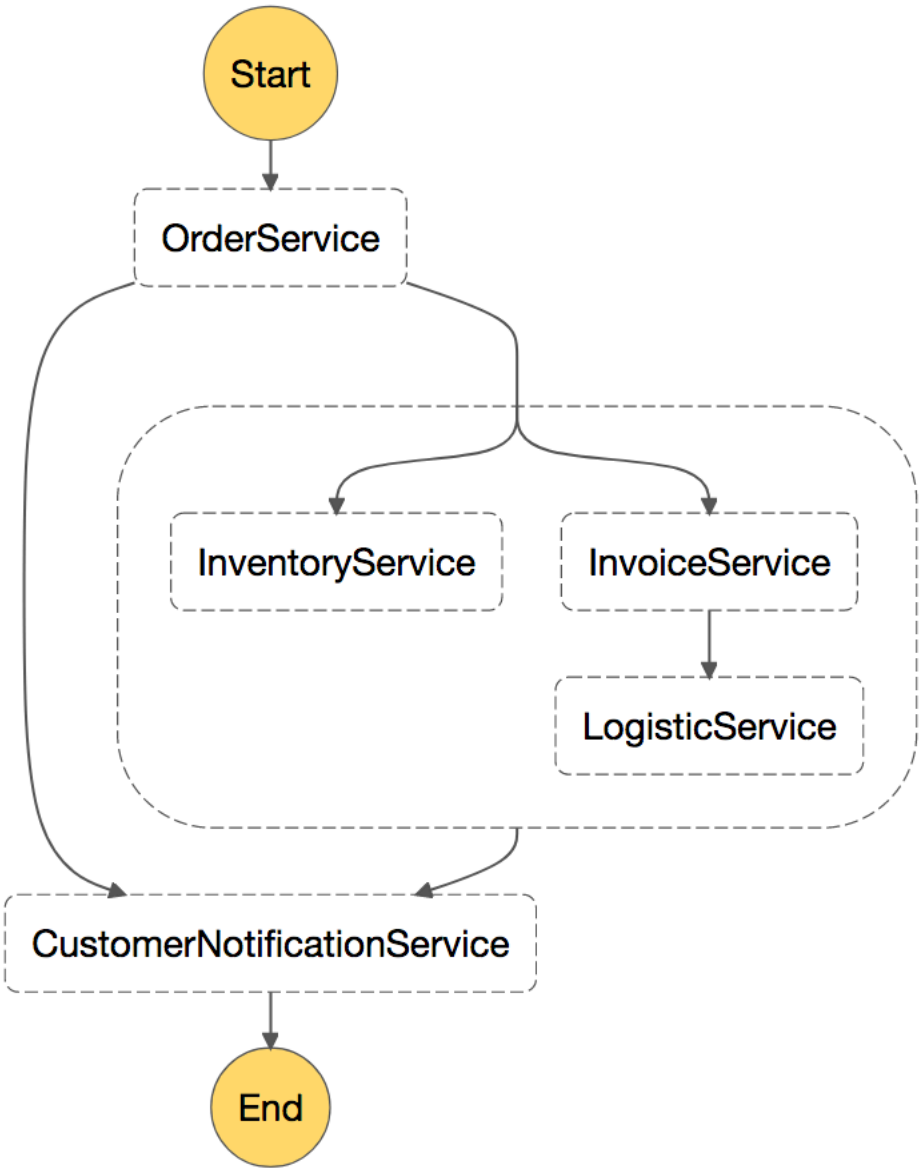
O Step Functions é uma forma confiável de coordenar componentes e percorrer as funções do aplicativo. O Step Functions oferece um console gráfico para organizar e visualizar os componentes do aplicativo como uma série de etapas. Isso facilita a criação e a execução de serviços distribuídos.

O Step Functions inicia e rastreia automaticamente todas as etapas e tenta executar novamente aquelas que apresentaram falha para que a aplicação seja executada na ordem e da forma esperadas. O Step Functions registra em log o estado de cada etapa. Quando ocorre algum erro, você pode diagnosticar e depurar rapidamente os problemas. Você pode alterar e adicionar etapas sem escrever código, o que ajuda na evolução do aplicativo e na agilidade da inovação.

O Step Functions faz parte da plataforma sem servidor da AWS e comporta a orquestração de funções do Lambda, bem como aplicações baseadas em recursos de computação, como o Amazon EC2, o Amazon EKS e o Amazon ECS, e serviços adicionais como o Amazon SageMaker e o AWS Glue. O Step Functions gerencia as operações e a infraestrutura subjacente para ajudar a garantir a disponibilidade da aplicação em qualquer escala.

Para criar fluxos de trabalho, o Step Functions usa a Amazon States Language. Os fluxos de trabalho podem conter etapas paralelas ou sequenciais, bem como de ramificação.

A figura a seguir mostra um exemplo de fluxo de trabalho para uma arquitetura de microsserviços combinando etapas sequenciais e paralelas. A invocação desse fluxo de trabalho pode ser feita usando a API do Step Functions ou o API Gateway.



Um exemplo de fluxo de trabalho de microsserviços invocado pelo Step Functions

Monitoramento distribuído

Uma arquitetura de microsserviços é composta de várias partes distribuídas diferentes que precisam ser monitoradas. Você pode usar o Amazon CloudWatch para coletar e rastrear métricas, centralizar e monitorar arquivos de log, definir alarmes e reagir automaticamente a alterações em seu ambiente da AWS. O CloudWatch pode monitorar recursos da AWS, como instâncias do Amazon EC2, tabelas do DynamoDB e instâncias de banco de dados do Amazon RDS, além de métricas personalizadas geradas por seus serviços e aplicações, bem como quaisquer arquivos de log que suas aplicações gerarem.

Monitoramento

Você pode usar o CloudWatch para obter visibilidade sobre a utilização de recursos, a performance de aplicativos e a integridade operacional em todo o sistema. O CloudWatch oferece uma solução de monitoramento confiável, escalável e flexível que você pode começar a usar em minutos. Não é mais necessário configurar, gerenciar e escalar sua própria infraestrutura e sistemas de monitoramento. Em uma arquitetura de microsserviços, o recurso do monitoramento de métricas personalizadas usando o CloudWatch é um benefício adicional, pois os desenvolvedores podem decidir quais métricas devem ser coletadas para cada serviço. Além disso, a escalabilidade dinâmica pode ser implementada com base em métricas personalizadas.

Além do Amazon CloudWatch, você também pode usar o CloudWatch Container Insights para coletar, agregar e resumir métricas e logs de seus microsserviços e aplicações conteinerizados. O CloudWatch Container Insights coleta automaticamente métricas para muitos recursos, como CPU, memória, disco e rede, e as agrega como métricas do CloudWatch em nível de cluster, nó, pod, tarefa e serviço. Com o CloudWatch Container Insights, você pode obter acesso às métricas do painel do CloudWatch Container Insights. Ele também fornece informações de diagnóstico, como falhas de reinicialização de contêiner, para ajudar você a isolar problemas e resolvê-los rapidamente. Você também pode definir alarmes do CloudWatch em métricas que o Container Insights coleta.

O Container Insights está disponível para as plataformas Amazon ECS, Amazon EKS e Kubernetes no Amazon EC2. O suporte do Amazon ECS inclui suporte para o Fargate.

Outra opção apreciada, particularmente para o Amazon EKS, é o Prometheus. O Prometheus é um toolkit de monitoramento e alertas de código aberto usado frequentemente com o Grafana para visualizar as métricas coletadas. Muitos componentes do Kubernetes armazenam métricas em `/metrics`, e o Prometheus pode acessá-las em intervalos regulares.

O Amazon Managed Service for Prometheus (AMP) é um serviço de monitoramento compatível com o Prometheus que permite o monitoramento de aplicações conteinerizadas em qualquer escala. Com o AMP, você pode usar a linguagem de consulta do Prometheus (PromQL) de código aberto para monitorar a performance de workloads conteinerizadas sem ter de controlar a infraestrutura subjacente necessária para gerenciar a ingestão, o armazenamento e a consulta de métricas operacionais. Você pode coletar métricas do Prometheus de ambientes do Amazon EKS e do Amazon ECS usando o AWS Distro para servidores OpenTelemetry ou Prometheus como agentes de coleta.

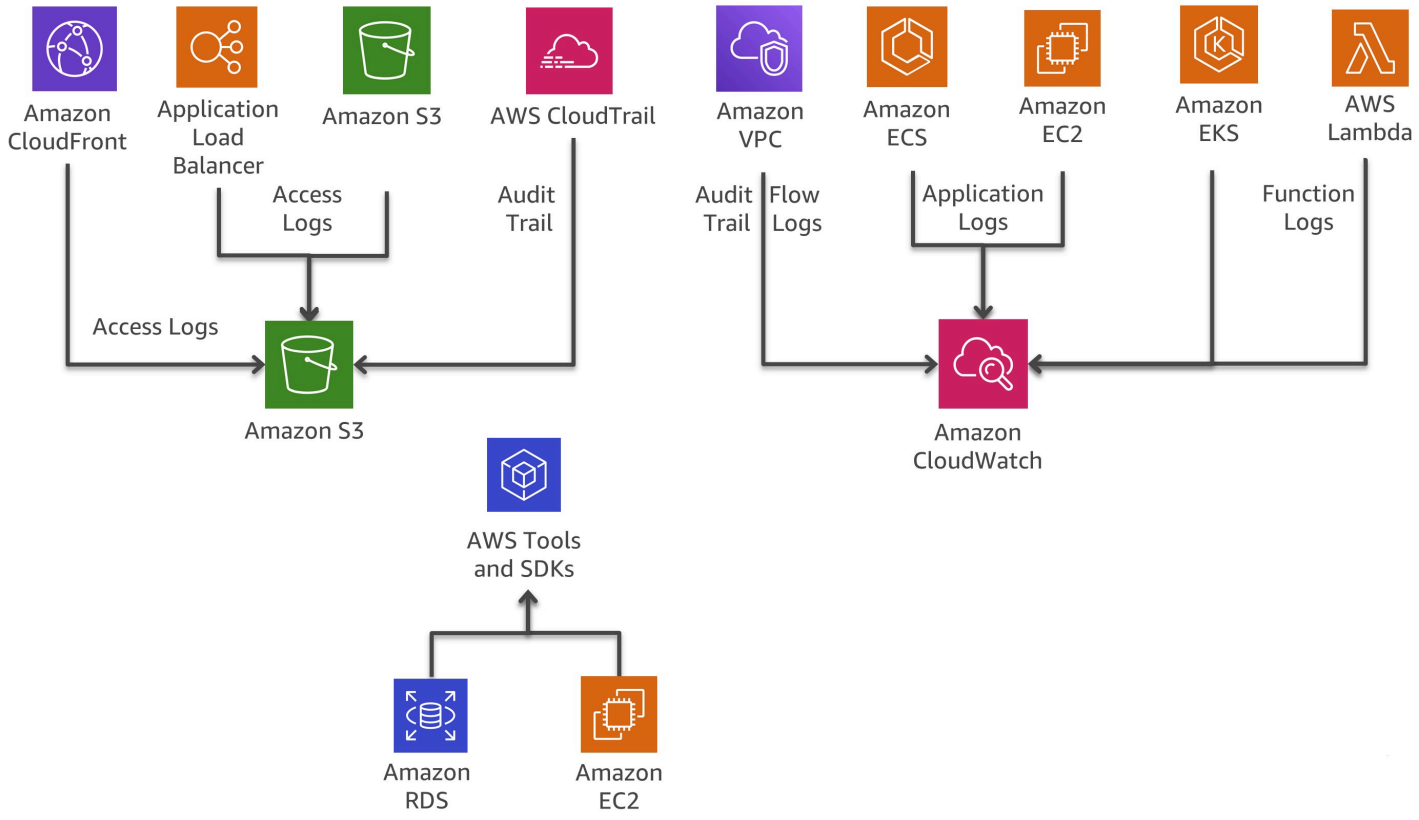
O AMP é frequentemente usado em conjunto com o Amazon Managed Service for Grafana (AMG). Com o AMG, é fácil consultar, visualizar, alertar e entender suas métricas, não importa onde elas estejam armazenadas. Com o AMG, você pode analisar métricas, logs e rastreamentos sem ter de provisionar servidores, configurar e atualizar software ou fazer o trabalho pesado envolvido na segurança e escalabilidade do Grafana na produção.

Centralização de logs

Um registro em log consistente é essencial para identificar e solucionar problemas. Os microsserviços possibilitam que as equipes entreguem muito mais versões do que antes e incentivam as equipes de segurança a experimentar novos recursos em produção. Compreender o impacto sobre o cliente é essencial para aprimorar gradativamente uma aplicação.

Por padrão, a maioria dos serviços da AWS centraliza os arquivos de log. Os principais destinos para arquivos de log na AWS são o Amazon S3 e o Amazon CloudWatch Logs. Para aplicações executadas em instâncias do Amazon EC2, há um daemon disponível para enviar arquivos de log ao CloudWatch Logs. As funções do Lambda enviam nativamente a saída dos logs ao CloudWatch Logs e o Amazon ECS inclui suporte para o driver de logs awslogs que habilita a centralização dos logs de contêiner no CloudWatch Logs. Para o Amazon EKS, tanto o Fluent Bit quanto o Fluentd podem encaminhar logs das instâncias individuais no cluster a um CloudWatch Logs centralizado, no qual eles são combinados para gerar relatórios mais gerais usando o Amazon OpenSearch Service e o Kibana. Devido ao menor espaço e às vantagens de performance, recomenda-se o Fluent Bit, em vez de o FluentD.

A figura a seguir mostra os recursos de registro em log de alguns dos serviços. As equipes podem pesquisar e analisar esses logs usando ferramentas como o Amazon OpenSearch Service e o Kibana. O Amazon Athena pode ser usado para executar consultas únicas em arquivos de log centralizados no Amazon S3.



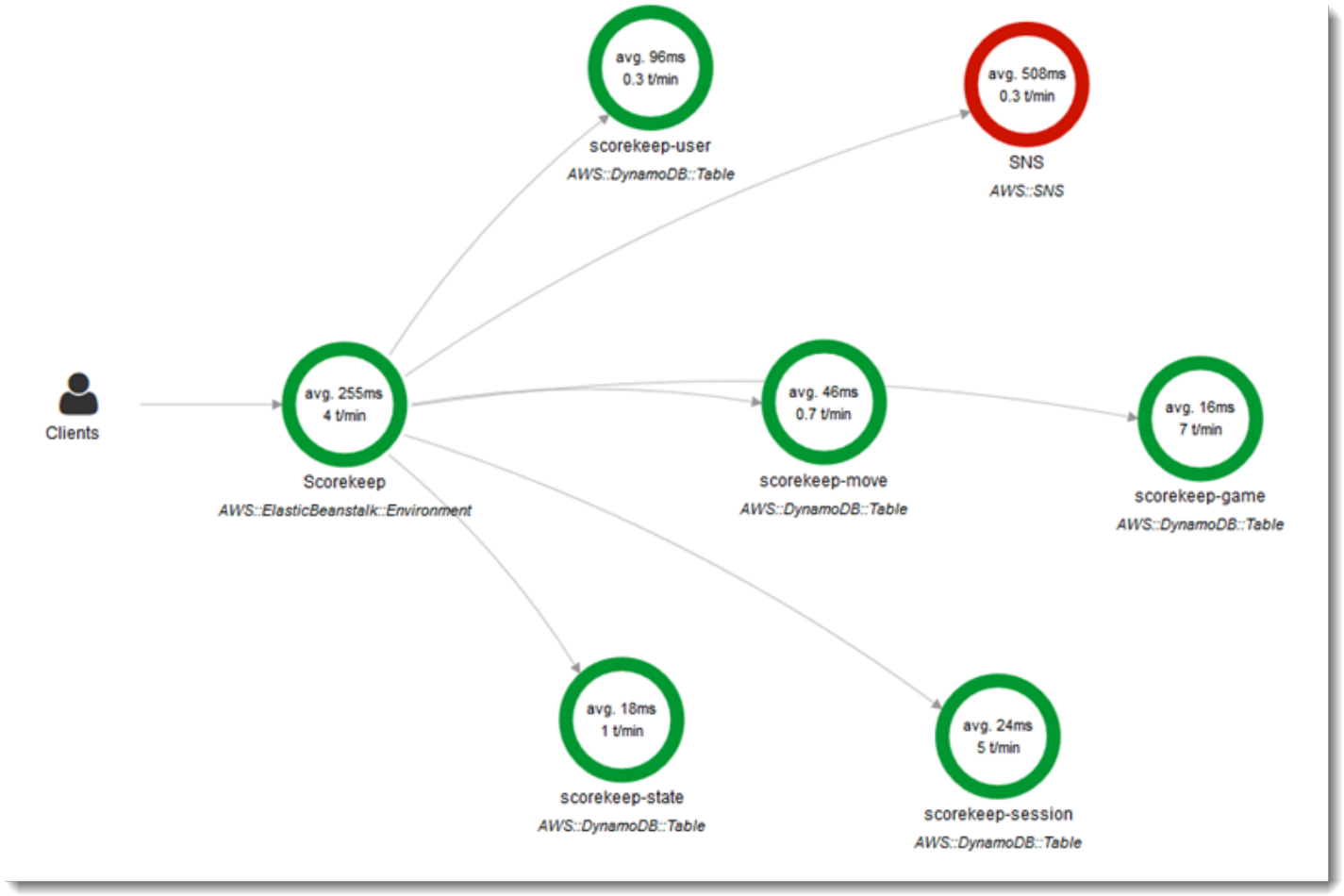
Recursos de registro em log dos serviços da AWS

Rastreamento distribuído

Em muitos casos, um conjunto de microsserviços funciona em conjunto para atender a uma solicitação. Imagine um sistema complexo que consiste em dezenas de microsserviços, e que ocorre um erro em um dos serviços na cadeia de chamadas. Mesmo que cada microsserviço gere corretamente os registros em log e esses logs sejam consolidados em um sistema central, poderá ser difícil encontrar todas as mensagens de log relevantes.

A ideia central do AWS X-Ray é o uso de IDs de correlação, que são identificadores exclusivos anexados a todas as solicitações e mensagens relacionadas a uma cadeia de eventos específica. O ID de rastreamento é adicionado a solicitações HTTP em cabeçalhos de rastreamento específicos, denominados `X-Amzn-Trace-Id`, quando a solicitação chega ao primeiro serviço integrado ao X-Ray (por exemplo, um Application Load Balancer ou um API Gateway), e incluído na resposta. Com o X-Ray SDK, qualquer microsserviço pode ler, adicionar ou atualizar esse cabeçalho.

O X-Ray funciona com o Amazon EC2, o Amazon ECS, o Lambda e o AWS Elastic Beanstalk. Você pode usar o X-Ray com aplicações escritas em Java, Node.js e .NET que são implantadas nesses serviços.



Mapa de serviços do AWS X-Ray

O Epsagon é um SaaS totalmente gerenciado que inclui rastreamento para todos os serviços da AWS, APIs de terceiros (por meio de chamadas HTTP) e outros serviços comuns, como Redis, Kafka e Elastic. O serviço Epsagon inclui recursos de monitoramento, alertas para os serviços mais comuns e visibilidade da carga útil em cada chamada que seu código estiver fazendo.

O AWS Distro for OpenTelemetry é uma distribuição do projeto OpenTelemetry segura, pronta para produção e apoiada pela AWS. Como parte do projeto Cloud Native Computing Foundation, o AWS Distro for OpenTelemetry fornece APIs, bibliotecas e agentes de código aberto para coletar métricas e rastreamentos distribuídos para monitoramento de aplicações. Com o AWS Distro for OpenTelemetry, você pode instrumentalizar suas aplicações apenas uma vez para enviar métricas e rastreamentos correlacionados a diversas soluções de monitoramento de parceiros e da AWS.

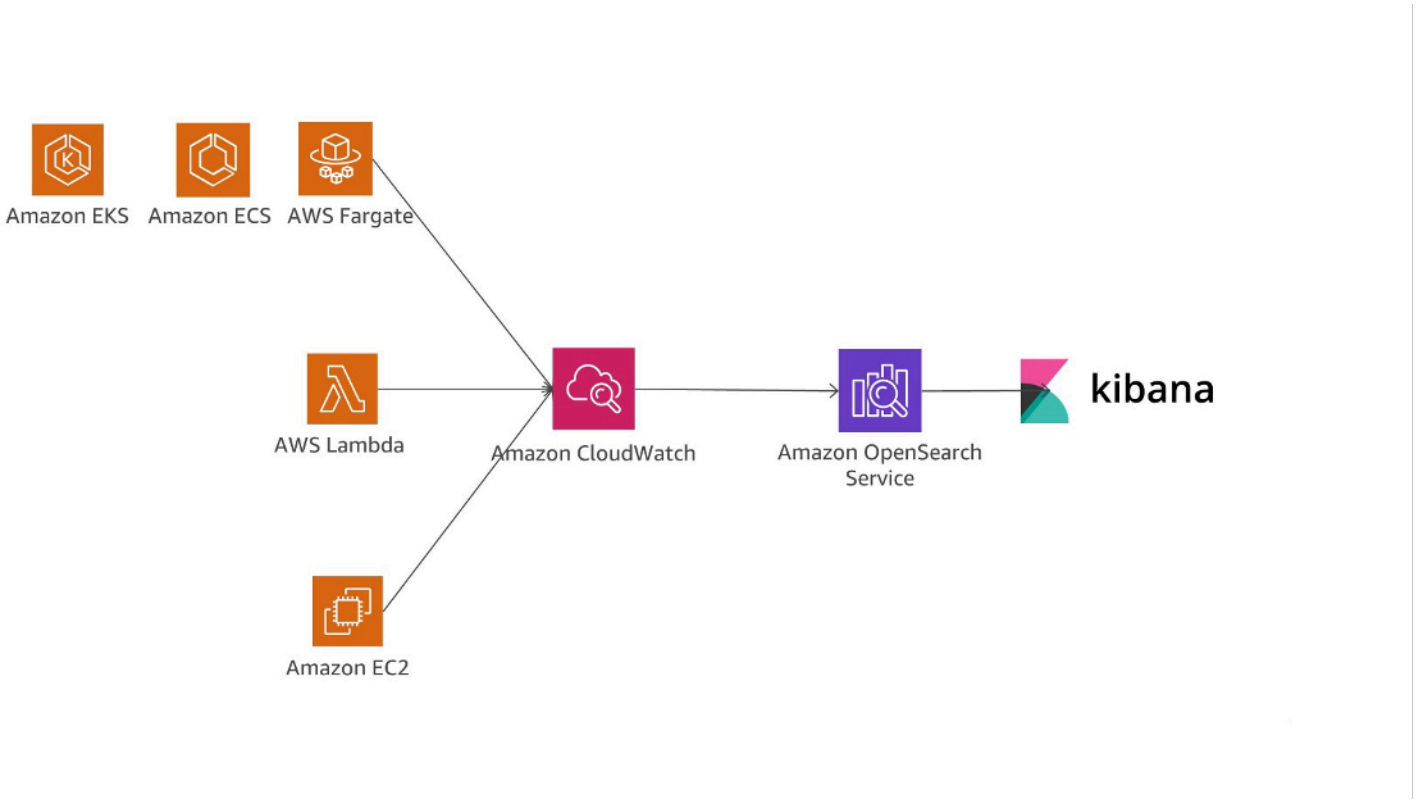
Use agentes de instrumentalização automática para coletar rastreamentos sem alterar seu código. O AWS Distro for OpenTelemetry também coleta metadados de seus recursos e serviços gerenciados da AWS para que você possa correlacionar dados de performance de aplicações com dados de infraestrutura subjacentes, o que reduz o tempo médio para a resolução de problemas. Use o AWS Distro for OpenTelemetry para instrumentar suas aplicações executadas no Amazon EC2, Amazon ECS, Amazon EKS no Amazon EC2, Fargate e AWS Lambda, bem como em ambientes on-premises.

Opções para análise de logs na AWS

A pesquisa, análise e visualização de dados de log são aspectos importantes para compreender os sistemas distribuídos. O Amazon CloudWatch Logs Insights possibilita que você explore, analise e visualize instantaneamente os logs. Isso permite que você resolva problemas operacionais. Outra opção para analisar arquivos de log é usar o Amazon OpenSearch Service com o Kibana.

O Amazon OpenSearch Service pode ser usado para pesquisa de texto completo, pesquisa estruturada, análises e todas as três atividades combinadas. O Kibana é um plugin de visualização de dados de código aberto que se integra perfeitamente ao Amazon OpenSearch Service.

A figura a seguir demonstra a análise de logs com o Amazon OpenSearch Service e o Kibana. O CloudWatch Logs pode ser configurado para fazer uma transmissão de entradas de log no Amazon OpenSearch Service praticamente em tempo real por meio de uma assinatura do CloudWatch Logs. O Kibana visualiza os dados e apresenta uma interface de pesquisa conveniente para datastores no Amazon OpenSearch Service. Essa solução pode ser usada com programas de software como o ElastAlert a fim de implementar um sistema de alertas para enviar notificações e e-mails do SNS, criar tíquetes do JIRA e outras funcionalidades quando anomalias, picos ou outros padrões de interesse são detectados nos dados.



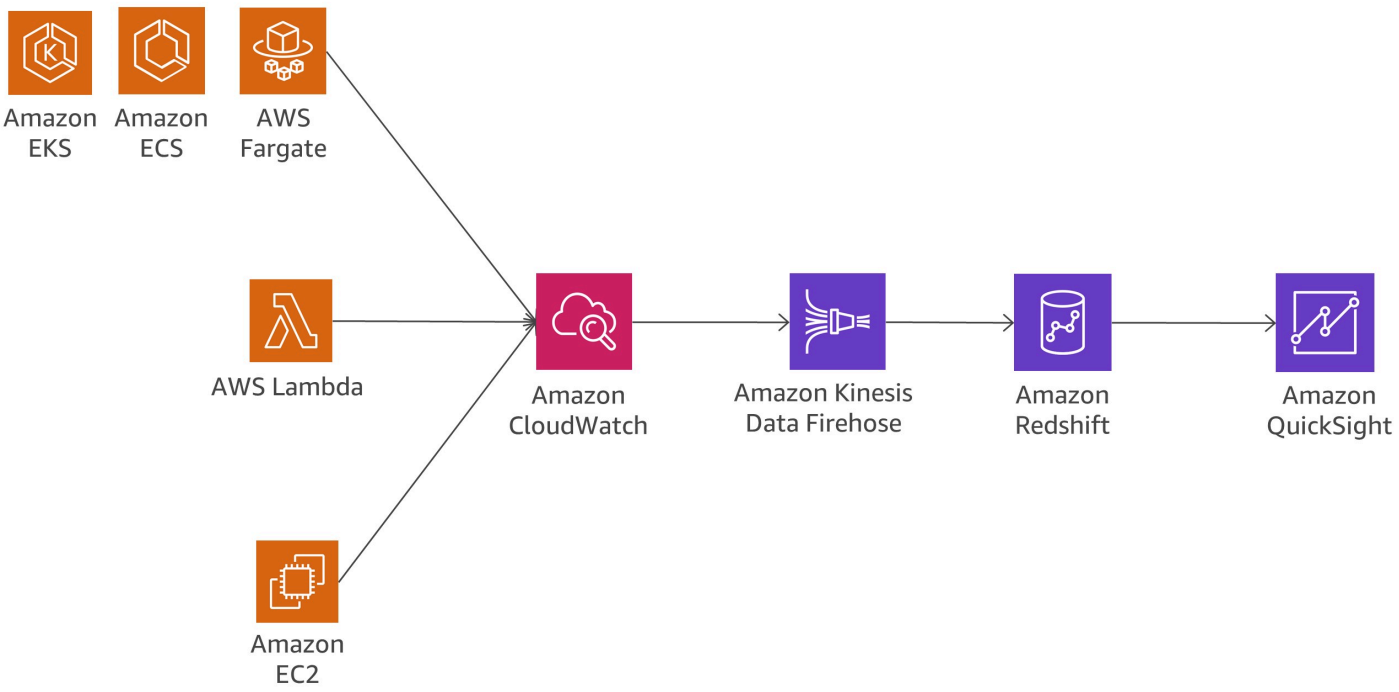
Análise de logs com o Amazon OpenSearch Service e o Kibana

Outra opção para analisar arquivos de log é usar o Amazon Redshift com o Amazon QuickSight.

O QuickSight pode ser facilmente conectado a serviços de dados da AWS, como Amazon Redshift, Amazon RDS, Amazon Aurora, Amazon EMR, DynamoDB, Amazon S3 e Amazon Kinesis.

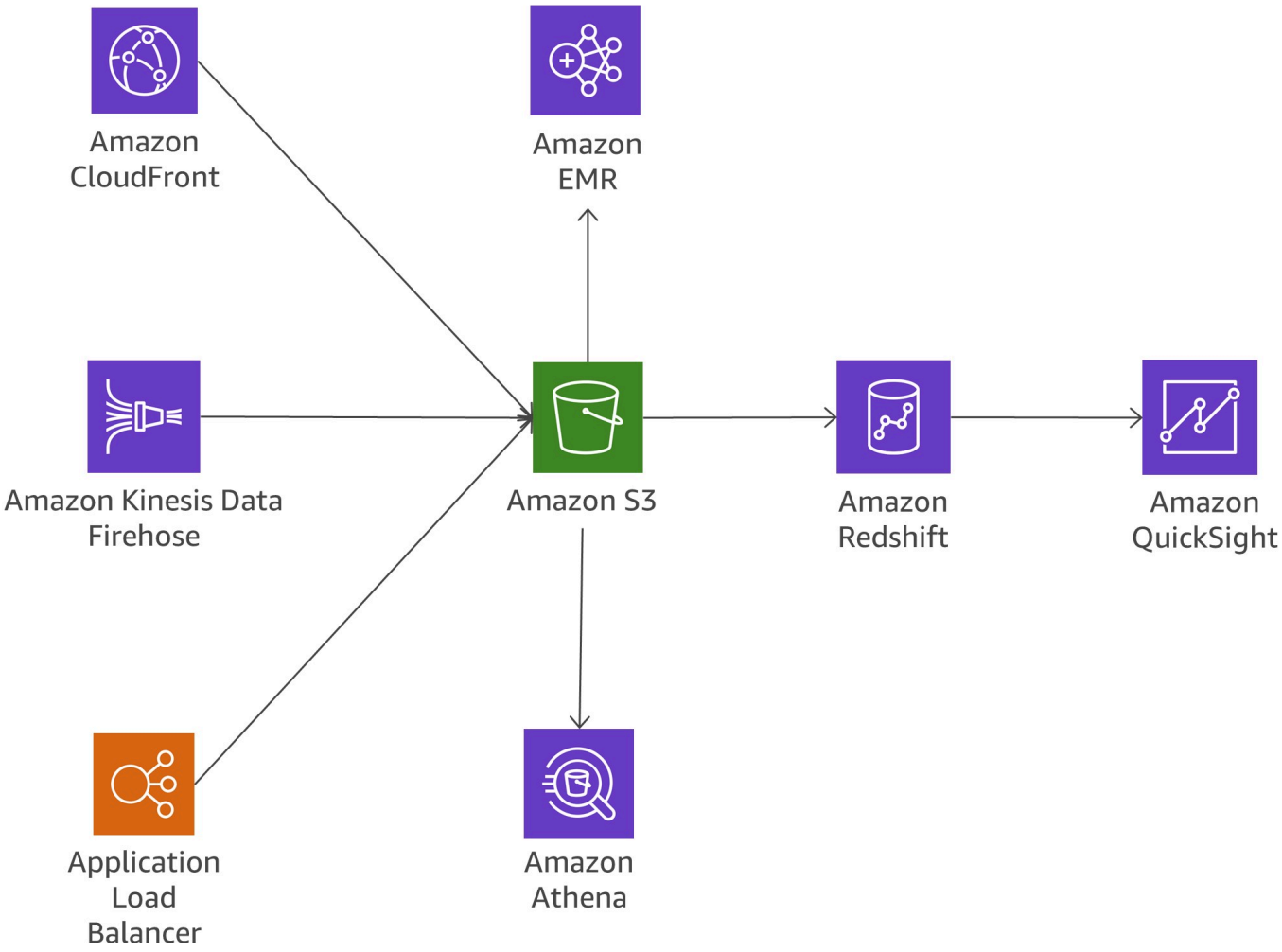
O CloudWatch Logs pode funcionar como um armazenamento centralizado para dados de log. Além de apenas armazenar os dados, é possível fazer a transmissão de entradas de log para o Amazon Kinesis Data Firehose.

A figura a seguir retrata um cenário em que entradas de log de diferentes fontes são transmitidas para o Amazon Redshift usando o CloudWatch Logs e o Kinesis Data Firehose. O Amazon QuickSight usa os dados armazenados no Amazon Redshift para análise, geração de relatórios e visualização.



Análise de logs com o Amazon Redshift e o Amazon QuickSight

A figura a seguir mostra um cenário de análise de logs no Amazon S3. Quando os logs são armazenados em buckets do Amazon S3, os dados de log podem ser carregados em diferentes serviços de dados da AWS, como Amazon Redshift ou Amazon EMR, para analisar os dados armazenados no fluxo de logs e encontrar anomalias.



Análise de logs no Amazon S3

Interações desnecessárias

A divisão de aplicativos monolíticos em pequenos microsserviços aumenta a sobrecarga de comunicação, pois os microsserviços precisam conversar entre si. Muitas implementações usam o REST com HTTP por ser um protocolo de comunicação leve. No entanto, grandes volumes de mensagens podem causar problemas. Em alguns casos, você pode pensar na possibilidade de consolidar os serviços que enviam e recebem grande quantidade de mensagens. Se estiver em uma situação em que consolida um número maior de serviços apenas para reduzir as interações desnecessárias, você precisará revisar os domínios com problemas e o modelo de domínios.

Protocolos

Em uma seção anterior deste whitepaper, the section called “Comunicação assíncrona e sistemas de mensagem leves”, diferentes protocolos possíveis são discutidos. Para microsserviços, é comum usar protocolos simples como HTTP. As mensagens trocadas pelos serviços podem ser codificadas de diferentes maneiras, como formatos legíveis, como JSON ou YAML, ou formatos binários eficientes, como Avro ou Protocol Buffers.

Armazenamento em cache

Os caches são uma ótima maneira de reduzir a latência e as interações desnecessárias de arquiteturas de microsserviços. Várias camadas de cache são possíveis, dependendo do caso de uso real e dos gargalos. Muitas aplicações de microsserviços executadas na AWS usam o ElastiCache para reduzir o volume de chamadas para outros microsserviços armazenando resultados em caches locais. O API Gateway fornece uma camada de armazenamento em cache incorporada para reduzir a carga em servidores de back-end. Além disso, o armazenamento em cache também é útil para reduzir a carga da camada de persistência de dados. O desafio de qualquer mecanismo de armazenamento em cache é encontrar o equilíbrio certo entre uma boa taxa de acertos do cache e a prontidão e consistência dos dados.

Auditoria

Outro desafio a ser resolvido nas arquiteturas de microsserviços, que podem ter até centenas de serviços distribuídos, é garantir a visibilidade das ações dos usuários em todos os serviços e conseguir obter uma boa visão geral em nível organizacional. Para ajudar a impor políticas de segurança, é importante auditar tanto o acesso a recursos quanto as atividades que causam alterações no sistema.

As alterações devem ser rastreadas por serviços individuais e, mais amplamente, nos serviços executados no sistema. Normalmente, as mudanças são frequentes em arquiteturas de microsserviços, o que aumenta ainda mais a importância da auditoria de mudanças. Esta seção examina os principais serviços e recursos da AWS que podem ajudar a auditar uma arquitetura de microsserviços.

Trilha de auditoria

O AWS CloudTrail é uma ferramenta útil para rastrear alterações em microsserviços porque permite que todas as chamadas de API executadas na Nuvem AWS sejam registradas em log e enviadas ao CloudWatch Logs em tempo real ou ao Amazon S3 em alguns minutos.

Todas as ações de usuário e ações de sistema automatizadas podem ser pesquisadas e analisadas para detectar comportamentos inesperados, violações de políticas da empresa ou depurações. As informações registradas incluem um carimbo de data/hora, informações de usuário/conta, o serviço que foi chamado, a ação de serviço solicitada, o endereço IP do chamador, bem como parâmetros de solicitação e os elementos de resposta.

O CloudTrail habilita a definição de várias trilhas para a mesma conta, o que permite que diferentes partes interessadas, como administradores de segurança, desenvolvedores de software ou auditores de TI, criem e gerenciem sua própria trilha. Se as equipes de microsserviços tiverem contas da AWS diferentes, será possível agregar as trilhas em um único bucket do S3.

As vantagens de armazenar as trilhas de auditoria no CloudWatch são que os dados de trilha de auditoria são capturados em tempo real e fica fácil redirecionar as informações para o Amazon OpenSearch Service para pesquisa e visualização. Você pode configurar o CloudTrail para fazer login no Amazon S3 e no CloudWatch Logs.

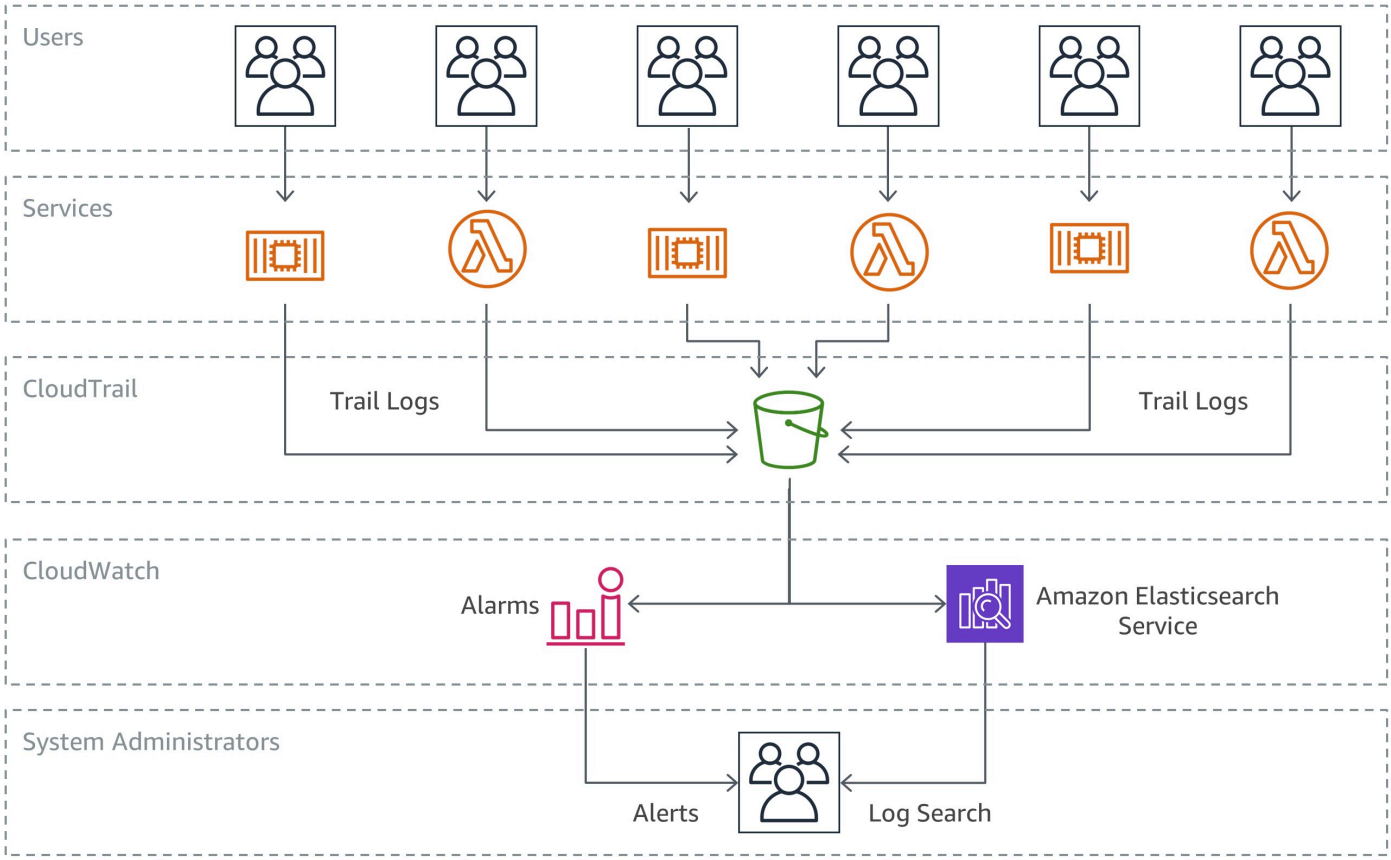
Eventos e ações em tempo real

Algumas alterações nas arquiteturas de sistema devem ser respondidas rapidamente por meio de ações a fim de corrigir a situação ou procedimentos de governança específicos para autorizar a mudança. A integração do Amazon CloudWatch Events com o CloudTrail permite que o CloudWatch Events gere eventos para todas as chamadas de API modificadas em todos os serviços da AWS. Também é possível definir eventos personalizados ou gerar eventos com base em uma programação fixa.

Quando um evento é acionado e corresponde a uma regra definida, um grupo predefinido de pessoas em sua organização pode ser notificado imediatamente, para que seja possível tomar a medida apropriada. Se a ação necessária puder ser automatizada, a regra poderá acionar automaticamente um fluxo de trabalho incorporado ou invocar uma função do Lambda para resolver o problema.

A figura a seguir mostra um ambiente em que o CloudTrail e o CloudWatch Events funcionam em conjunto para cumprir requisitos de auditoria e correção em uma arquitetura de microsserviços. Todos os microsserviços estão sendo rastreados pelo CloudTrail e a trilha de auditoria é armazenada em um bucket do Amazon S3. O CloudWatch Events toma conhecimento das mudanças operacionais à medida que elas ocorrem. O CloudWatch Events responde a essas alterações operacionais e executa a ação corretiva conforme necessário, enviando mensagens para responder ao ambiente, ativando funções, fazendo alterações e capturando informações de estado. O

CloudWatch Events é baseado no CloudTrail e aciona alertas quando uma alteração específica é feita na arquitetura.



Auditoria e correção

Inventário de recursos e gerenciamento de alterações

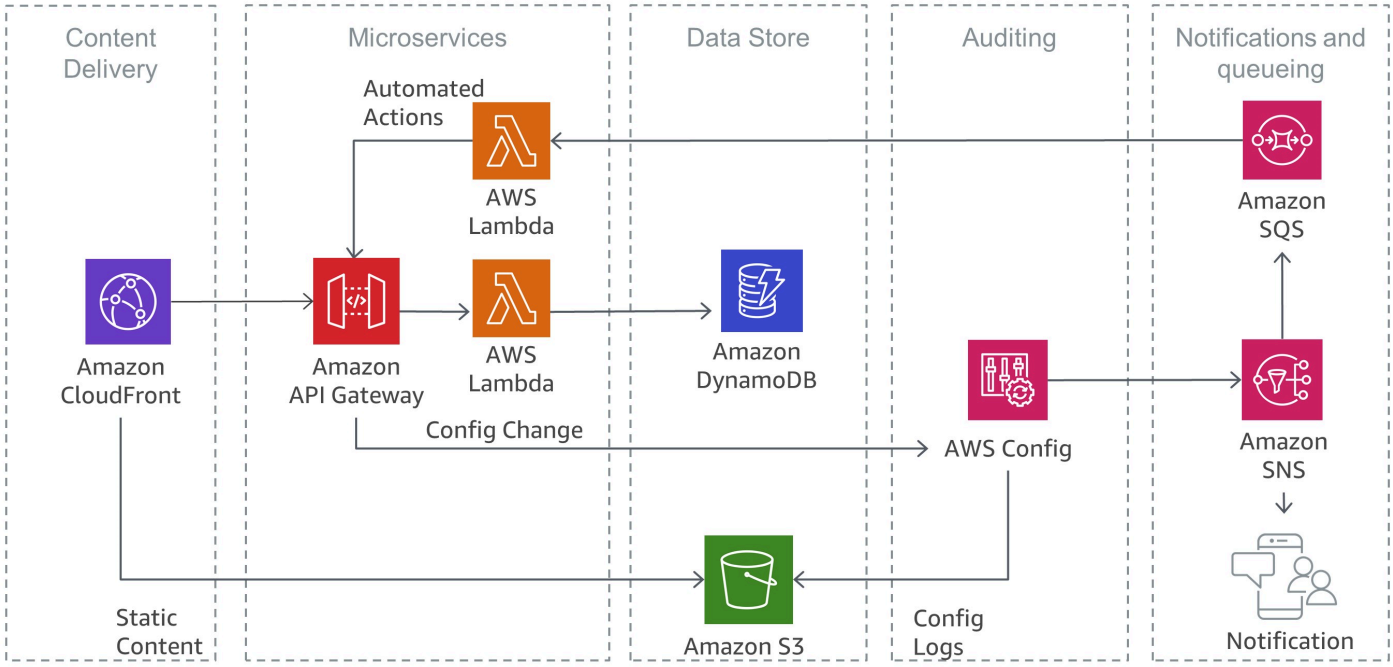
Para manter o controle sobre a rápida evolução de configurações de infraestrutura em um ambiente de desenvolvimento ágil, é essencial adotar uma abordagem mais automatizada e gerenciada para auditoria e controle da arquitetura.

Embora o CloudTrail e o CloudWatch Events sejam componentes básicos importantes para controlar as alterações de infraestrutura nos microsserviços e respondam a essas alterações, as regras do AWS Config permitem que uma empresa defina políticas de segurança com regras específicas para detectar, rastrear e notificar automaticamente violações dessas políticas.

O próximo exemplo demonstra como é possível detectar, informar e reagir automaticamente a alterações de configuração não compatíveis em uma arquitetura de microsserviços. Um membro da

equipe de desenvolvimento alterou o API Gateway para um microsserviço, permitindo que o endpoint aceite o tráfego HTTP de entrada, em vez de permitir somente solicitações HTTPS.

Como essa situação foi previamente identificada pela organização como uma preocupação de conformidade de segurança, uma regra do AWS Config já está monitorando essa condição. A regra identifica a alteração como uma violação de segurança e executa duas ações: cria um log da alteração detectada em um bucket do Amazon S3 para auditoria e cria uma notificação no SNS. O Amazon SNS é usado para duas finalidades em nosso cenário: enviar um e-mail a um grupo especificado para notificar a violação de segurança e adicionar uma mensagem a uma fila do SQS. Em seguida, a mensagem é recebida e o estado compatível é restaurado por meio da alteração da configuração do API Gateway.



Detecção de violações de segurança com o AWS Config

Conclusão

A arquitetura de microsserviços é uma abordagem de projetos distribuída, projetada para superar as limitações de arquiteturas monolíticas tradicionais. Os microsserviços ajudam a escalar aplicativos e organizações e, ao mesmo tempo, reduzir os tempos de ciclo. No entanto, eles também trazem alguns desafios que podem acrescentar complexidade de arquitetura e sobrecarga operacional.

A AWS oferece um amplo portfólio de serviços gerenciados que podem ajudar equipes de produtos a criar arquiteturas de microsserviços e reduzir a complexidade da arquitetura e das operações. Este whitepaper mostra os serviços relevantes da AWS e como implementar nativamente padrões comuns, como descoberta de serviços ou origem de eventos, usando os serviços da AWS.

Recursos

- Central de arquitetura da AWS
- Whitepapers da AWS
- Arquitetura mensal da AWS
- Blog de arquitetura da AWS
- Vídeos “This is my Architecture”
- AWS Answers
- Documentação da AWS

Histórico do documento e colaboradores

Histórico do documento

Para ser notificado sobre atualizações deste whitepaper, inscreva-se no RSS feed.

update-history-change	update-history-description	update-history-date
Whitepaper atualizado	Integração do Amazon EventBridge, AWS OpenTelemetry, AMP, AMG, Container Insights: pequenas alterações no texto.	9 de novembro de 2021
Pequenas atualizações	Layout de página ajustado	30 de abril de 2021
Pequenas atualizações	Pequenas mudanças no texto.	1.º de agosto de 2019
Whitepaper atualizado	Integração do Amazon EKS, AWS Fargate, Amazon MQ, AWS PrivateLink, AWS App Mesh, AWS Cloud Map	1.º de junho de 2019
Whitepaper atualizado	Integração do AWS Step Functions, do AWS X-Ray e das sequências de eventos do ECS.	1.º de setembro de 2017
Publicação inicial	Implementação de microsser viços na AWS publicada.	1.º de dezembro de 2016

> **Note**
>
> Para assinar atualizações RSS, você deve ter um plugin RSS habilitado para o navegador que está usando.

Colaboradores

Os indivíduos e empresas a seguir contribuíram para este documento:

- Sascha Möllering, arquitetura de soluções da AWS
- Christian Müller, arquitetura de soluções da AWS
- Matthias Jung, arquitetura de soluções da AWS
- Peter Dalbhanjan, arquitetura de soluções da AWS
- Peter Chapman, arquitetura de soluções da AWS
- Christoph Kassen, arquitetura de soluções da AWS
- Umair Ishaq, arquitetura de soluções da AWS
- Rajiv Kumar, arquitetura de soluções da AWS

Avisos

Os clientes são responsáveis por fazer sua própria avaliação independente das informações neste documento. Este documento é: (a) fornecido apenas para fins informativos, (b) representa as ofertas e práticas de produtos atuais da AWS, que estão sujeitas a alterações sem aviso prévio e (c) não cria nenhum compromisso ou garantia da AWS e suas afiliadas, fornecedores ou licenciadores. Os produtos ou serviços da AWS são fornecidos no “estado em que se encontram”, sem garantias, declarações ou condições de qualquer tipo, explícitas ou implícitas. As responsabilidades e obrigações da AWS com seus clientes são regidas por contratos da AWS, e este documento não modifica nem faz parte de nenhum contrato entre a AWS e seus clientes.