AWS Livro branco

# Implementando microsserviços em AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

# Implementando microsserviços em AWS: AWS Livro branco

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

# Table of Contents

Resumo e introdução ..... i
Introdução ..... 1
Você é Well-Architected? ..... 2
Modernização para microsserviços ..... 3
Arquitetura de microsserviços em AWS ..... 4
Interface do usuário ..... 5
Microsserviços ..... 5
Implementações de microsserviços ..... 5
CI/CD ..... 6
Redes privadas ..... 7
Datastore ..... 7
Simplificando as operações ..... 8
Implantação de aplicativos baseados em Lambda ..... 8
Abstraindo as complexidades da multilocação ..... 9
Gerenciamento de APIs ..... 10
Arquitetura de microsserviços sem servidor ..... 12
Sistemas resilientes e eficientes ..... 15
Recuperação de desastres (DR) ..... 15
Alta disponibilidade (HA) ..... 15
Componentes de sistemas distribuídos ..... 17
Gerenciamento distribuído de dados ..... 19
Gerenciamento de configuração ..... 22
Gerenciamento de segredos ..... 22
Otimização de custos e sustentabilidade ..... 23
Mecanismos de comunicação ..... 24
Comunicação baseada em REST ..... 24
Comunicação baseada em GraphQL ..... 24
Comunicação baseada em gRPC ..... 24
Mensagens assíncronas e transmissão de eventos ..... 24
Orquestração e gerenciamento de estado ..... 26
Observabilidade ..... 29
Monitoramento ..... 29
Centralizando registros ..... 31
Rastreamento distribuído ..... 32

Análise de log em AWS ..... 33
Outras opções para análise ..... 33
Gerenciando a comunicação de microsserviços ..... 36
Usando protocolos e armazenamento em cache ..... 36
Auditoria ..... 37
Inventário de recursos e gerenciamento de mudanças ..... 37
Conclusão ..... 39
Colaboradores ..... 40
Histórico do documento ..... 41
Avisos ..... 43
AWS Glossário ..... 44
..... xlv

# Implementando microsserviços em AWS

Data de publicação: 31 de julho de 2023 (Histórico do documento)

Os microsserviços oferecem uma abordagem simplificada para o desenvolvimento de software que acelera a implantação, incentiva a inovação, melhora a capacidade de manutenção e aumenta a escalabilidade. Esse método se baseia em serviços pequenos e pouco acoplados que se comunicam por meio de serviços bem definidos APIs, gerenciados por equipes autônomas. A adoção de microsserviços oferece benefícios, como maior escalabilidade, resiliência, flexibilidade e ciclos de desenvolvimento mais rápidos.

Este whitepaper explora três padrões populares de microsserviços: orientado por API, orientado por eventos e streaming de dados. Fornecemos uma visão geral de cada abordagem, descrevemos os principais recursos dos microsserviços, abordamos os desafios em seu desenvolvimento e ilustramos como a Amazon Web Services (AWS) pode ajudar as equipes de aplicativos a enfrentar esses obstáculos.

Considerando a natureza complexa de tópicos como armazenamento de dados, comunicação assíncrona e descoberta de serviços, recomendamos que você avalie as necessidades específicas e os casos de uso do seu aplicativo junto com a orientação fornecida ao tomar decisões de arquitetura.

## Introdução

As arquiteturas de microsserviços combinam conceitos bem-sucedidos e comprovados de vários campos, como:

- Desenvolvimento ágil de software
- Arquiteturas orientadas a serviços
- Design que prioriza a API
- ContínuoIntegration/Continuous Delivery (CI/CD)

Frequentemente, os microsserviços incorporam padrões de design do aplicativo Twelve-Factor.

Embora os microsserviços ofereçam muitos benefícios, é vital avaliar os requisitos exclusivos do seu caso de uso e os custos associados. Arquitetura monolítica ou abordagens alternativas podem ser mais apropriadas em alguns casos. A decisão entre microsserviços ou monólitos deve ser feita case-by-case com base em fatores como escala, complexidade e casos de uso específicos.

Primeiro, exploramos uma arquitetura de microsserviços altamente escalável e tolerante a falhas (interface de usuário, implementação de microsserviços, armazenamento de dados) e demonstramos como desenvolvê-la usando tecnologias de contêiner. AWS Em seguida, sugerimos AWS serviços para implementar uma arquitetura típica de microsserviços sem servidor, reduzindo a complexidade operacional.

A tecnologia sem servidor é caracterizada pelos seguintes princípios:

- Sem infraestrutura para provisionar ou gerenciar
- Dimensionamento automático por unidade de consumo
- Modelo de cobrança “Pague pelo valor”
- Disponibilidade e tolerância a falhas integradas
- Arquitetura orientada a eventos (EDA)

Por fim, examinamos o sistema geral e discutimos os aspectos entre serviços de uma arquitetura de microsserviços, como monitoramento distribuído, registro, rastreamento, auditoria, consistência de dados e comunicação assíncrona.

Este documento se concentra nas cargas de trabalho em execução no Nuvem AWS, excluindo cenários híbridos e estratégias de migração. Para obter informações sobre estratégias de migração, consulte o whitepaper da Metodologia de Migração de Contêineres.

## Você é Well-Architected?

A AWS Well-Architected Framework ajuda você a entender os prós e os contras das decisões que você toma ao criar sistemas na nuvem. Os seis pilares do framework permitem a você conhecer as melhores práticas de arquitetura para criar e operar sistemas confiáveis, seguros, econômicos e sustentáveis na nuvem. Usando o AWS Well-Architected Tool, disponível gratuitamente no Console de gerenciamento da AWS, você pode analisar suas workloads em relação a essas práticas recomendadas respondendo a um conjunto de perguntas para cada pilar.

No Serverless Application Lens, nos concentramos nas melhores práticas para arquitetar seus aplicativos sem servidor em. AWS

Para obter orientações especializadas e melhores práticas adicionais para a arquitetura de sua nuvem (implantações de arquitetura de referência, diagramas e whitepapers), consulte o Centro de Arquitetura da AWS.

# Modernização para microsserviços

Os microsserviços são essencialmente unidades pequenas e independentes que compõem um aplicativo. A transição de estruturas monolíticas tradicionais para microsserviços pode seguir várias estratégias.

Essa transição também afeta a forma como sua organização opera:

- Ela incentiva o desenvolvimento ágil, em que as equipes trabalham em ciclos rápidos.
- As equipes geralmente são pequenas, às vezes descritas como duas equipes de pizza — pequenas o suficiente para que duas pizzas possam alimentar toda a equipe.
- As equipes assumem total responsabilidade por seus serviços, desde a criação até a implantação e manutenção.

# Arquitetura simples de microsserviços em AWS

Os aplicativos monolíticos típicos consistem em diferentes camadas: uma camada de apresentação, uma camada de aplicativo e uma camada de dados. As arquiteturas de microsserviços, por outro lado, separam as funcionalidades em verticais coesas de acordo com domínios específicos, em vez de camadas tecnológicas. A Figura 1 ilustra uma arquitetura de referência para um aplicativo típico de microsserviços em. AWS

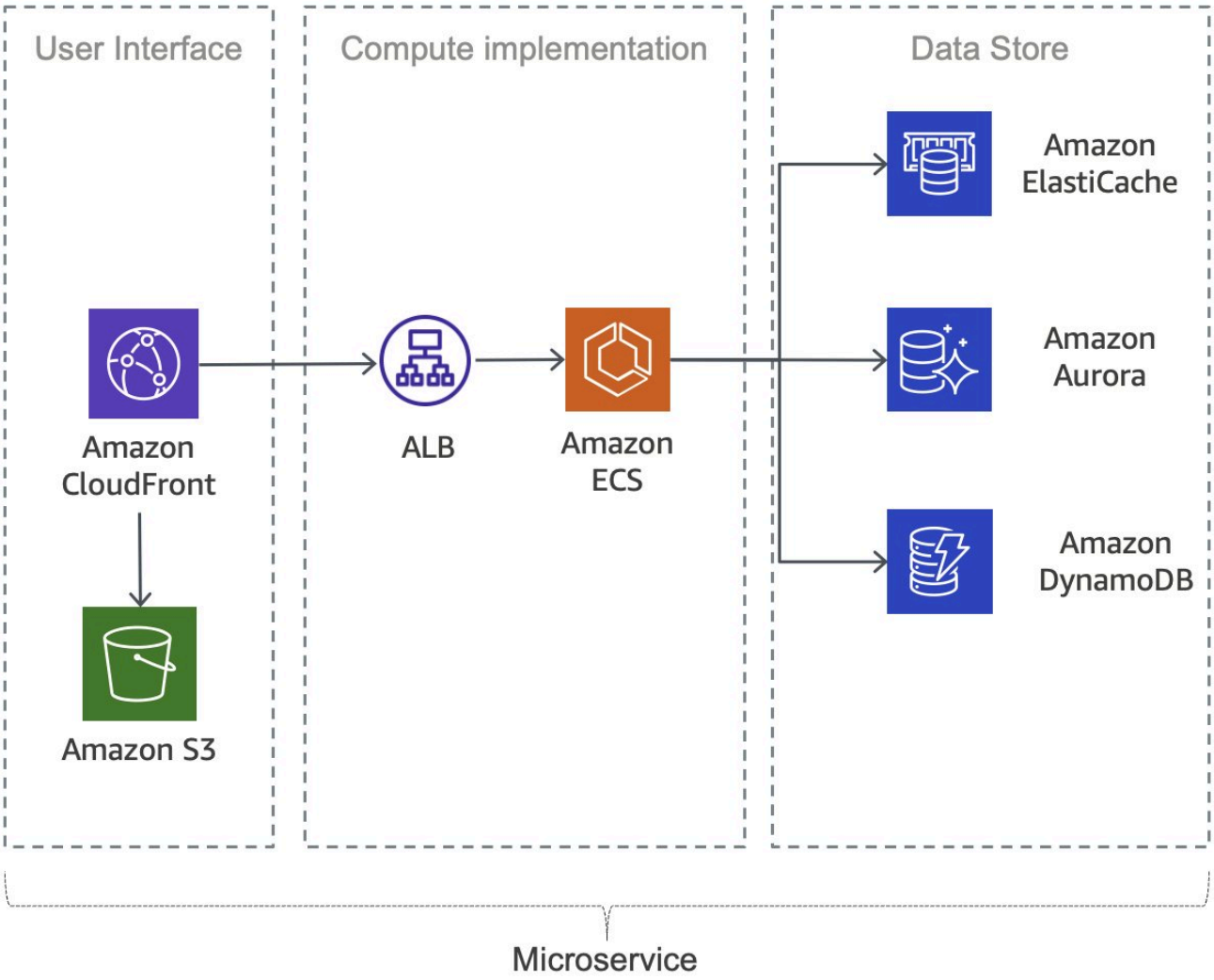


Figura 1: Aplicativo típico de microsserviços em AWS

# Interface do usuário

Os aplicativos web modernos geralmente usam JavaScript estruturas para desenvolver aplicativos de página única que se comunicam com o back-end. APIs Normalmente, eles APIs são criados usando Representational State Transfer (REST) ou RESTful APIs GraphQL APIs. O conteúdo estático da web pode ser servido usando o Amazon Simple Storage Service (Amazon S3) e a Amazon. CloudFront

# Microsserviços

APIs são considerados a porta de entrada dos microsserviços, pois são o ponto de entrada para a lógica do aplicativo. Normalmente, a API de serviços RESTful web ou o GraphQL APIs são usados. Eles APIs gerenciam e processam chamadas de clientes, gerenciando funções como gerenciamento de tráfego, filtragem de solicitações, roteamento, armazenamento em cache, autenticação e autorização.

## Implementações de microsserviços

AWS oferece elementos básicos para desenvolver microsserviços, incluindo Amazon ECS e Amazon EKS como opções para mecanismos de orquestração de contêineres AWS Fargate e EC2 como opções de hospedagem. AWS Lambda é outra forma sem servidor de criar microsserviços. AWS A escolha entre essas opções de hospedagem depende dos requisitos do cliente para gerenciar a infraestrutura subjacente.

AWS Lambda permite que você carregue seu código, escalando e gerenciando automaticamente sua execução com alta disponibilidade. Isso elimina a necessidade de gerenciamento de infraestrutura, para que você possa agir rapidamente e se concentrar em sua lógica de negócios. O Lambda oferece suporte a várias linguagens de programação e pode ser acionado por outros AWS serviços ou chamado diretamente de aplicativos móveis ou da web.

Os aplicativos baseados em contêineres ganharam popularidade devido à portabilidade, produtividade e eficiência.AWS oferece vários serviços para criar, implantar e gerenciar contêineres.

- App2Container, uma ferramenta de linha de comando para migrar e modernizar aplicativos web Java e.NET em formato de contêiner. AWS O A2C analisa e cria um inventário de aplicativos executados em máquinas virtuais, bare metal, instâncias do Amazon Elastic Compute Cloud (EC2) ou na nuvem.

- O Amazon Elastic Container Service (Amazon ECS) e o Amazon Elastic Kubernetes Service (Amazon EKS) gerenciam sua infraestrutura de contêineres, facilitando o lançamento e a manutenção de aplicativos em contêineres.
  - O Amazon EKS é um serviço gerenciado de Kubernetes para executar o Kubernetes na AWS nuvem e em datacenters locais (Amazon EKS Anywhere). Isso estende os serviços em nuvem para ambientes locais para processamento de dados local de baixa latência, altos custos de transferência de dados ou requisitos de residência de dados (consulte o whitepaper sobre "Executando cargas de trabalho de contêineres híbridos com o Amazon EKS Anywhere"). Você pode usar todos os plug-ins e ferramentas existentes da comunidade Kubernetes com o EKS.
  - O Amazon Elastic Container Service (Amazon ECS) é um serviço de orquestração de contêineres totalmente gerenciado que simplifica sua implantação, gerenciamento e escalabilidade de aplicativos em contêineres. Os clientes escolhem o ECS pela simplicidade e pela profunda integração com AWS os serviços.

Para ler mais, consulte o blog Amazon ECS vs Amazon EKS: compreendendo os serviços de AWS contêineres.

- AWS App Runneré um serviço de aplicativo de contêiner totalmente gerenciado que permite criar, implantar e executar aplicativos web em contêineres e serviços de API sem experiência prévia em infraestrutura ou contêiner.
- AWS Fargate, um mecanismo de computação sem servidor, trabalha com o Amazon ECS e o Amazon EKS para gerenciar automaticamente os recursos computacionais para aplicativos de contêineres.
- O Amazon ECR é um registro de contêineres totalmente gerenciado que oferece hospedagem de alto desempenho, para que você possa implantar imagens e artefatos de aplicativos de forma confiável em qualquer lugar.

## Integração contínua e implantação contínua (CI/CD)

A integração contínua e a entrega contínua (CI/CD) são uma parte crucial de uma DevOps iniciativa para mudanças rápidas no software. AWS oferece serviços CI/CD para implementação de microsserviços, mas uma discussão detalhada está além do escopo deste documento. Para obter mais informações, consulte o AWS whitepaper Praticando a integração contínua e a entrega contínua.

## Redes privadas

AWS PrivateLink é uma tecnologia que aprimora a segurança dos microsserviços ao permitir conexões privadas entre sua Virtual Private Cloud (VPC) e os serviços compatíveis. AWS Ele ajuda a isolar e proteger o tráfego de microsserviços, garantindo que ele nunca atravesse a Internet pública. Isso é particularmente útil para cumprir regulamentações como PCI ou HIPAA.

## Datastore

O armazenamento de dados é usado para manter os dados necessários aos microsserviços. Armazenamentos populares para dados de sessão são caches na memória, como Memcached ou Redis. AWS oferece as duas tecnologias como parte do ElastiCache serviço gerenciado da Amazon.

Colocar um cache entre os servidores de aplicativos e um banco de dados é um mecanismo comum para reduzir a carga de leitura no banco de dados, o que, por sua vez, pode permitir que recursos sejam usados para suportar mais gravações. Os caches também podem melhorar a latência.

Os bancos de dados relacionais ainda são muito populares para armazenar dados estruturados e objetos de negócios. 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).

Os bancos de dados relacionais, no entanto, não são projetados para uma escala infinita, o que pode tornar difícil e demorada a aplicação de técnicas para suportar um grande número de consultas.

Os bancos de dados NoSQL foram projetados para favorecer a escalabilidade, o desempenho e a disponibilidade em relação à consistência dos bancos de dados relacionais. Um elemento importante dos bancos de dados NoSQL é que eles normalmente não impõem um esquema estrito. Os dados são distribuídos em partições que podem ser escaladas horizontalmente e recuperados usando chaves de partição.

Como os microsserviços individuais são projetados para fazer algo bem, eles normalmente têm um modelo de dados simplificado que pode ser adequado à persistência do NoSQL. É importante compreender que os bancos de dados NoSQL têm padrões de acesso diferentes dos bancos de dados relacionais. Por exemplo, não é possível unir tabelas. Se isso for necessário, a lógica deve ser implementada no aplicativo. Você pode usar o Amazon DynamoDB para criar uma tabela de banco de dados que pode armazenar e recuperar qualquer quantidade de dados e atender a qualquer nível de tráfego de solicitações. O DynamoDB oferece desempenho de um dígito em milissegundos, no

entanto, há certos casos de uso que exigem tempos de resposta em microssegundos. O DynamoDB Accelerator (DAX) fornece recursos de cache para acessar dados.

O DynamoDB também oferece um recurso de escalabilidade automática para ajustar dinamicamente a capacidade de transferência em resposta ao tráfego real. No entanto, há casos em que o planejamento da capacidade é difícil ou não é possível devido aos grandes picos de atividade de curta duração em seu aplicativo. Para tais situações, o DynamoDB oferece uma opção sob demanda, que oferece preços simples. pay-per-request O DynamoDB on-demand é capaz de atender milhares de solicitações por segundo instantaneamente sem planejamento de capacidade.

Para obter mais informações, consulte Gerenciamento distribuído de dados Como escolher um banco de dados.

# Simplificando as operações

Para simplificar ainda mais os esforços operacionais necessários para executar, manter e monitorar microsserviços, podemos usar uma arquitetura totalmente sem servidor.

## Implantação de aplicativos baseados em Lambda

Você pode implantar seu código Lambda fazendo o upload de um `zip` arquivo arquivado ou criando e carregando uma imagem de contêiner por meio da interface do usuário do console usando um URI de imagem válido do Amazon ECR. No entanto, quando uma função Lambda se torna complexa, o que significa que ela tem camadas, dependências e permissões, o upload por meio da interface do usuário pode se tornar complicado para alterações no código.

Usar AWS CloudFormation and the AWS Serverless Application Model (AWS SAM) AWS Cloud Development Kit (AWS CDK), ou Terraform simplifica o processo de definição de aplicativos sem servidor. AWS O SAM, suportado nativamente pelo CloudFormation, oferece uma sintaxe simplificada para especificar recursos sem servidor.AWS Lambda As camadas ajudam a gerenciar bibliotecas compartilhadas em várias funções do Lambda, minimizando o espaço ocupado pelas funções, centralizando bibliotecas com reconhecimento de inquilinos e melhorando a experiência do desenvolvedor. O Lambda SnapStart for Java aprimora o desempenho de inicialização de aplicativos sensíveis à latência.

Para implantar, especifique políticas de recursos e permissões em um CloudFormation modelo, empacote artefatos de implantação e implante o modelo. O SAM Local, uma AWS CLI ferramenta, permite o desenvolvimento local, o teste e a análise de aplicativos sem servidor antes do upload para o Lambda.

A integração com ferramentas como AWS Cloud9 IDE, AWS CodeBuild, e AWS CodePipeline simplifica a criação AWS CodeDeploy, o teste, a depuração e a implantação de aplicativos baseados em SAM.

O diagrama a seguir mostra a implantação de AWS Serverless Application Model recursos usando ferramentas CloudFormation de AWS CI/CD.

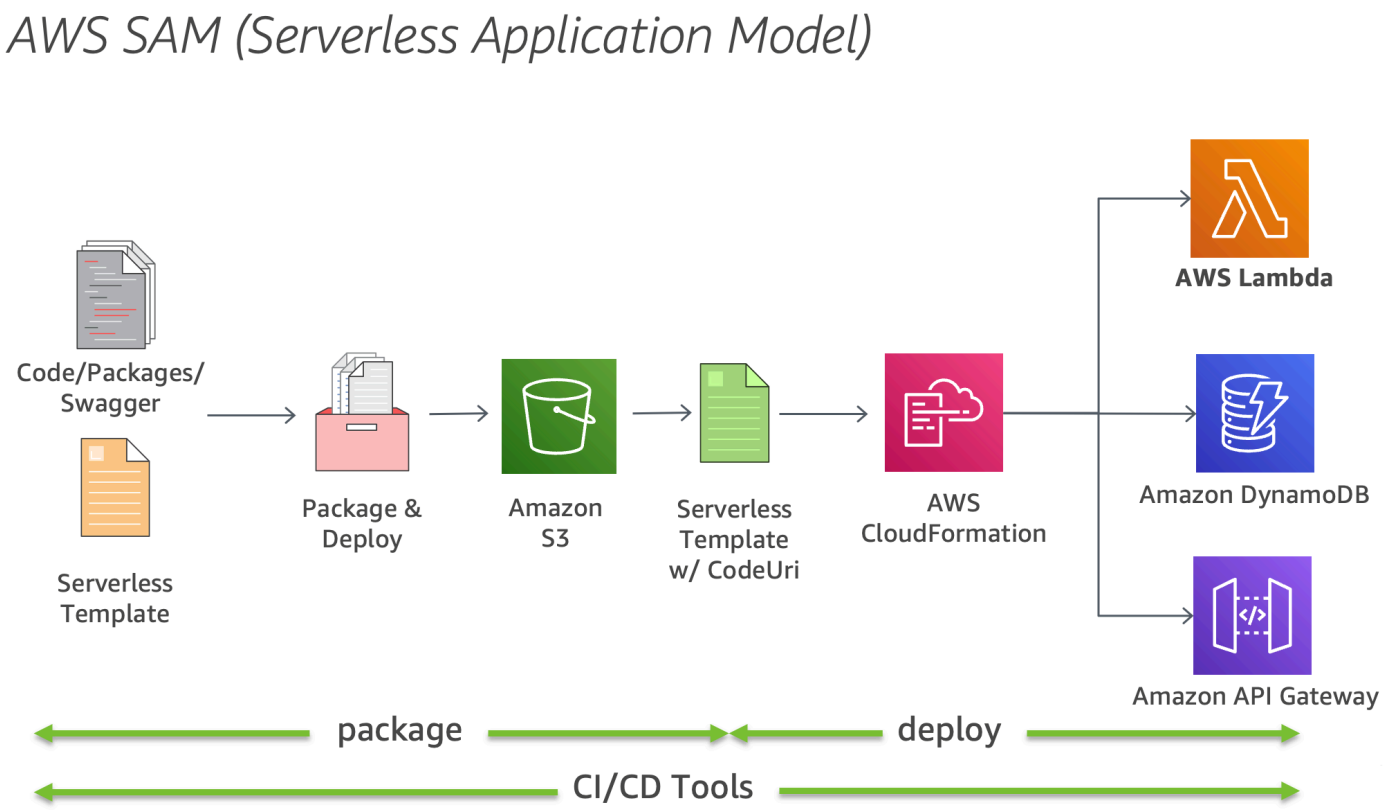


Figura 2: AWS Serverless Application Model (AWS SAM)

## Abstraindo as complexidades da multilocação

Em um ambiente multilocatário, como plataformas SaaS, é crucial simplificar as complexidades relacionadas à multilocação, permitindo que os desenvolvedores se concentrem no desenvolvimento de recursos e funcionalidades. Isso pode ser feito usando ferramentas como AWS Lambda Layers, que oferecem bibliotecas compartilhadas para abordar questões transversais. A lógica por trás dessa abordagem é que bibliotecas e ferramentas compartilhadas, quando usadas corretamente, gerenciam com eficiência o contexto do inquilino.

No entanto, eles não devem se estender ao encapsulamento da lógica de negócios devido à complexidade e ao risco que podem apresentar. Um problema fundamental com as bibliotecas

compartilhadas é o aumento da complexidade das atualizações, tornando-as mais difíceis de gerenciar em comparação com a duplicação de código padrão. Portanto, é essencial encontrar um equilíbrio entre o uso de bibliotecas compartilhadas e a duplicação na busca pela abstração mais eficaz.

## Gerenciamento de APIs

O gerenciamento APIs pode ser demorado, especialmente quando se considera várias versões, estágios do ciclo de desenvolvimento, autorização e outros recursos, como limitação e armazenamento em cache. Além do API Gateway, alguns clientes também usam o ALB (Application Load Balancer) ou o NLB (Network Load Balancer) para gerenciamento de API. O Amazon API Gateway ajuda a reduzir a complexidade operacional de criação e manutenção RESTful APIs. Ele permite que você crie APIs programaticamente, serve como uma “porta de entrada” para acessar dados, lógica de negócios ou funcionalidade de seus serviços de back-end, autorização e controle de acesso, limitação de taxa, armazenamento em cache, monitoramento e gerenciamento de tráfego e é executado sem gerenciar servidores. APIs

A Figura 3 ilustra como o API Gateway lida com chamadas de API e interage com outros componentes. Solicitações de dispositivos móveis, sites ou outros serviços de back-end são encaminhadas para o CloudFront Ponto de Presença (PoP) mais próximo para reduzir a latência e fornecer uma experiência de usuário ideal.

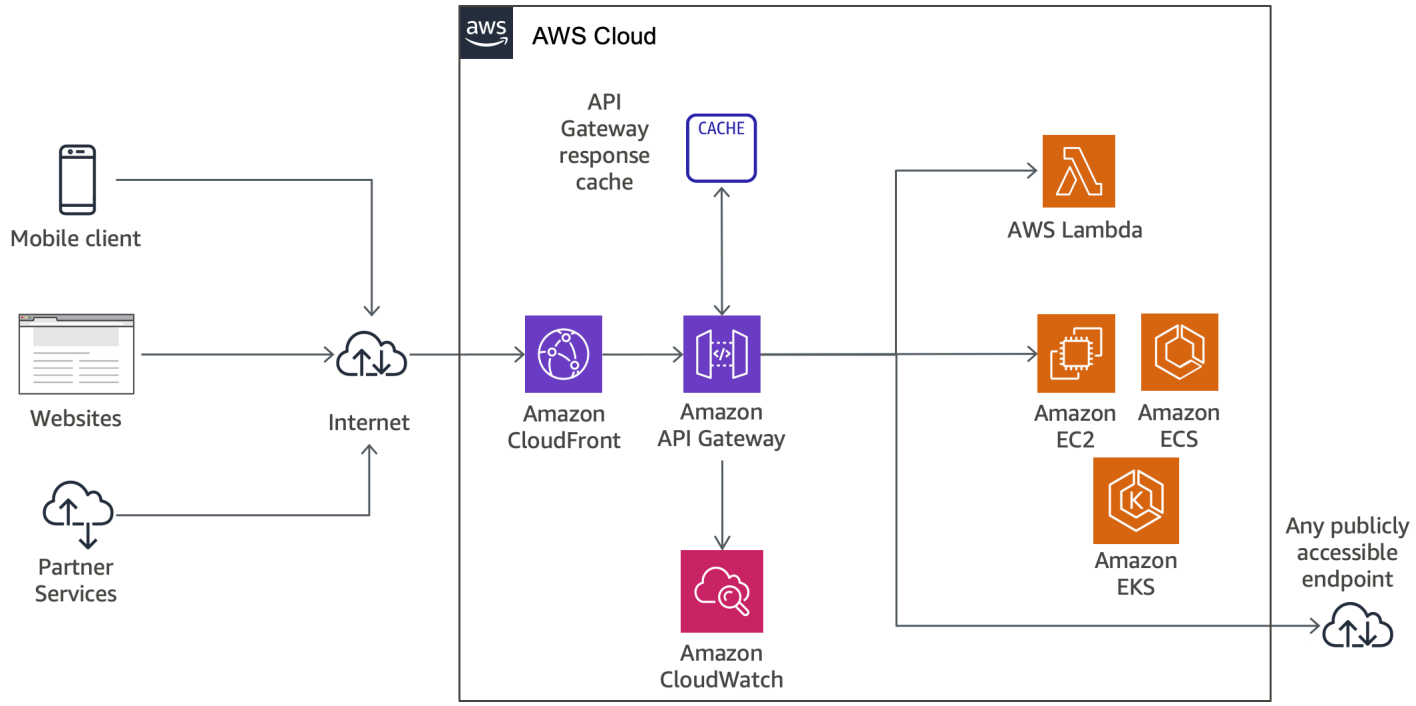

Figura 3: Fluxo de chamadas do API Gateway

# Microsserviços em tecnologias sem servidor

Usar microsserviços com tecnologias sem servidor pode diminuir consideravelmente a complexidade operacional. AWS Lambda e AWS Fargate, integrado ao API Gateway, permite a criação de aplicativos totalmente sem servidor. A partir de 7 de abril de 2023, as funções do Lambda podem transmitir progressivamente as cargas de resposta de volta ao cliente, aprimorando o desempenho de aplicativos móveis e da web. Antes disso, os aplicativos baseados em Lambda que usavam o modelo tradicional de invocação de solicitação-resposta precisavam gerar e armazenar a resposta em buffer antes de devolvê-la ao cliente, o que poderia atrasar o tempo até o primeiro byte. Com o streaming de respostas, as funções podem enviar respostas parciais de volta ao cliente assim que estiverem prontas, melhorando significativamente o tempo até o primeiro byte, ao qual os aplicativos móveis e da Web são especialmente sensíveis.

A Figura 4 demonstra uma arquitetura de microsserviços sem servidor usando AWS Lambda e gerenciando serviços. Essa arquitetura sem servidor atenua a necessidade de projetar para escalabilidade e alta disponibilidade e reduz o esforço necessário para executar e monitorar a infraestrutura subjacente.

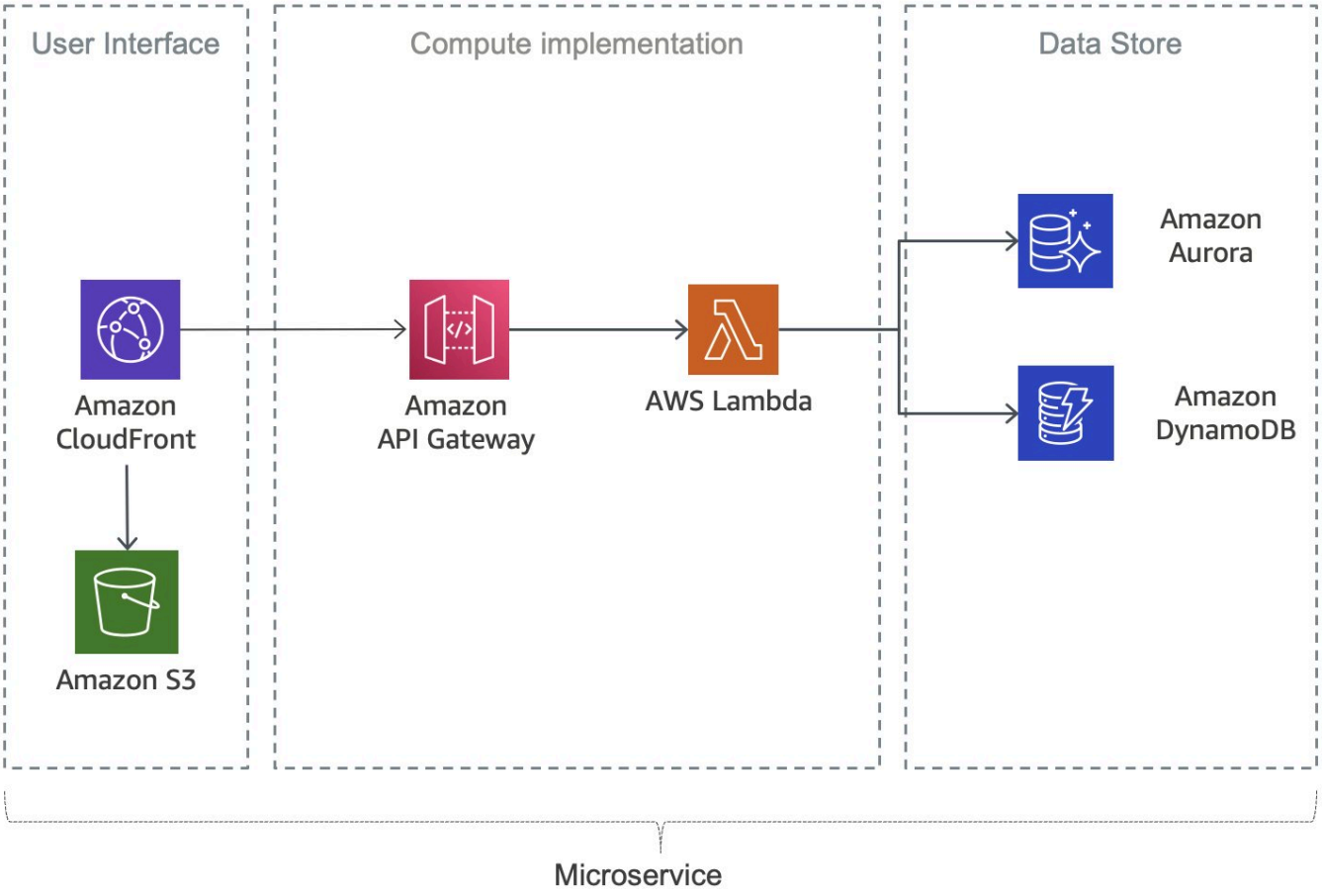


Figura 4: Microsserviço sem servidor usando AWS Lambda

A Figura 5 mostra uma implementação sem servidor semelhante usando contêineres com AWS Fargate, eliminando as preocupações com a infraestrutura subjacente. Ele também conta com o Amazon Aurora Serverless, um banco de dados sob demanda e com escalabilidade automática que ajusta automaticamente a capacidade com base nos requisitos do seu aplicativo.

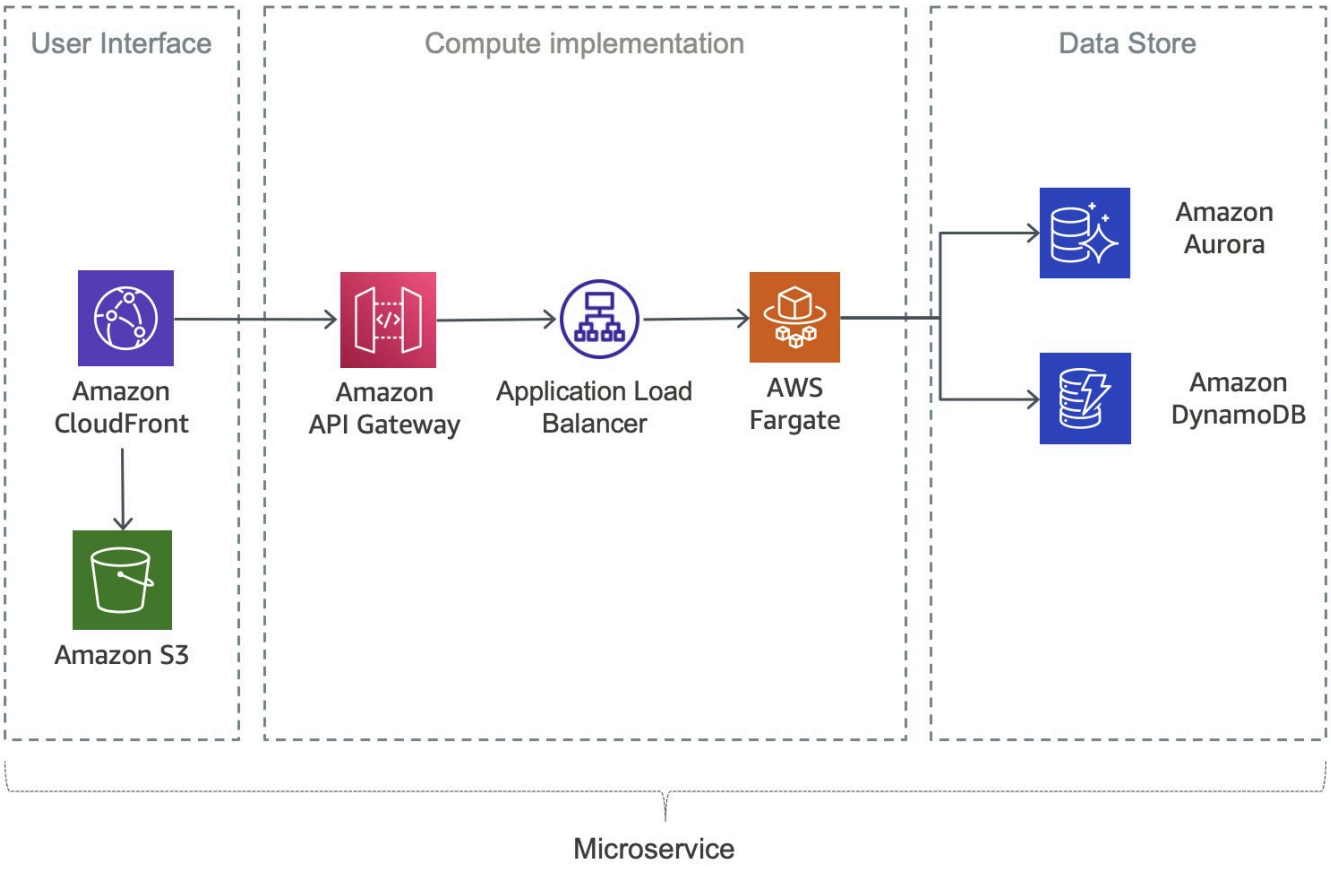


Figura 5: Microsserviço sem servidor usando AWS Fargate

# Sistemas resilientes e eficientes

## Recuperação de desastres (DR)

Os aplicativos de microsserviços geralmente seguem os padrões de aplicativos de doze fatores, em que os processos não têm estado e os dados persistentes são armazenados em serviços de apoio com estado, como bancos de dados. Isso simplifica a recuperação de desastres (DR) porque, se um serviço falhar, é fácil iniciar novas instâncias para restaurar a funcionalidade.

As estratégias de recuperação de desastres para microsserviços devem se concentrar em serviços downstream que mantenham o estado do aplicativo, como sistemas de arquivos, bancos de dados ou filas. As organizações devem planejar o objetivo de tempo de recuperação (RTO) e o objetivo de ponto de recuperação (RPO). O RTO é o atraso máximo aceitável entre a interrupção e a restauração do serviço, enquanto o RPO é o tempo máximo desde o último ponto de recuperação de dados.

Para saber mais sobre estratégias de recuperação de desastres, consulte o whitepaper Recuperação de Desastres de Cargas de Trabalho em AWS: Recuperação na Nuvem.

## Alta disponibilidade (HA)

Examinaremos a alta disponibilidade (HA) de vários componentes de uma arquitetura de microsserviços.

O Amazon EKS fornece alta disponibilidade executando instâncias de controle e plano de dados do Kubernetes em várias zonas de disponibilidade. Ele detecta e substitui automaticamente instâncias insalubres do plano de controle e fornece atualizações e patches de versão automatizados.

O Amazon ECR usa o Amazon Simple Storage Service (Amazon S3) para armazenamento a fim de tornar suas imagens de contêiner altamente disponíveis e acessíveis. Ele funciona com o Amazon EKS, o Amazon ECS e AWS Lambda, simplificando o fluxo de trabalho do desenvolvimento para a produção.

O Amazon ECS é um serviço regional que simplifica a execução de contêineres de forma altamente disponível em várias zonas de disponibilidade em uma região, oferecendo várias estratégias de agendamento que colocam contêineres para necessidades de recursos e requisitos de disponibilidade.

AWS Lambda opera em várias zonas de disponibilidade, garantindo disponibilidade durante interrupções de serviço em uma única zona. Se você conectar sua função a uma VPC, especifique sub-redes em várias zonas de disponibilidade para obter alta disponibilidade.

# Componentes de sistemas distribuídos

Em uma arquitetura de microsserviços, a descoberta de serviços se refere ao processo de localizar e identificar dinamicamente os locais de rede (endereços IP e portas) de microsserviços individuais em um sistema distribuído.

Ao escolher uma abordagem AWS, considere fatores como:

- Modificação do código: você pode obter os benefícios sem modificar o código?
- Tráfego entre VPCs ou entre contas: se necessário, seu sistema precisa de um gerenciamento eficiente da comunicação entre diferentes ou? VPCs Contas da AWS
- Estratégias de implantação: seu sistema usa ou planeja usar estratégias avançadas de implantação, como implantações azul-esverdeadas ou canárias?
- Considerações de desempenho: se sua arquitetura se comunica frequentemente com serviços externos, qual será o impacto no desempenho geral?

AWS oferece vários métodos para implementar a descoberta de serviços em sua arquitetura de microsserviços:

- Amazon ECS Service Discovery: O Amazon ECS oferece suporte à descoberta de serviços usando seu método baseado em DNS ou por meio da integração com AWS Cloud Map (consulte Descoberta de serviços do ECS). O ECS Service Connect melhora ainda mais o gerenciamento de conexões, o que pode ser especialmente benéfico para aplicativos maiores com vários serviços interativos.
- Amazon Route 53: o Route 53 se integra ao ECS e a outros AWS serviços, como o EKS, para facilitar a descoberta de serviços. Em um contexto de ECS, o Route 53 pode usar o recurso ECS Service Discovery, que aproveita a API de nomeação automática para registrar e cancelar o registro automático de serviços.
- AWS Cloud Map: essa opção oferece uma descoberta dinâmica de serviços baseada em API, que propaga as alterações em seus serviços.

Para necessidades de comunicação mais avançadas, o Amazon VPC Lattice é um serviço de rede de aplicativos que conecta, monitora e protege consistentemente as comunicações entre seus serviços, ajudando a melhorar a produtividade para que seus desenvolvedores possam se concentrar na criação de recursos importantes para sua empresa. Você pode definir políticas para

gerenciamento, acesso e monitoramento de tráfego de rede para conectar serviços de computação de forma simplificada e consistente entre instâncias, contêineres e aplicativos sem servidor.

Caso você já esteja usando software de terceiros, como HashiCorp Consul ou Netflix Eureka para descobrir serviços, talvez prefira continuar usando-os durante a migração AWS, permitindo uma transição mais suave.

A escolha entre essas opções deve estar alinhada às suas necessidades específicas. Para requisitos mais simples, soluções baseadas em DNS, como Amazon ECS ou AWS Cloud Map Amazon, podem ser suficientes. Para sistemas mais complexos ou maiores, malhas de serviços como o Amazon VPC Lattice podem ser mais adequadas.

Concluindo, projetar uma arquitetura de microsserviços significa selecionar as ferramentas certas para atender às suas necessidades específicas. AWS Ao ter em mente as considerações discutidas, você pode ter certeza de que está tomando decisões informadas para otimizar a descoberta de serviços e a comunicação entre serviços do seu sistema.

# Gerenciamento distribuído de dados

Em aplicativos tradicionais, todos os componentes geralmente compartilham um único banco de dados. Por outro lado, cada componente de um aplicativo baseado em microsserviços mantém seus próprios dados, promovendo independência e descentralização. Essa abordagem, conhecida como gerenciamento distribuído de dados, traz novos desafios.

Um desses desafios surge da troca entre consistência e desempenho em sistemas distribuídos. Geralmente, é mais prático aceitar pequenos atrasos nas atualizações de dados (consistência eventual) do que insistir em atualizações instantâneas (consistência imediata).

Às vezes, as operações comerciais exigem que vários microsserviços funcionem juntos. Se uma peça falhar, talvez seja necessário desfazer algumas tarefas concluídas. O padrão Saga ajuda a gerenciar isso coordenando uma série de ações compensatórias.

Para ajudar os microsserviços a permanecerem sincronizados, um armazenamento de dados centralizado pode ser usado. Essa loja, gerenciada com ferramentas como AWS Lambda, AWS Step Functions, e Amazon EventBridge, pode ajudar na limpeza e desduplicação de dados.

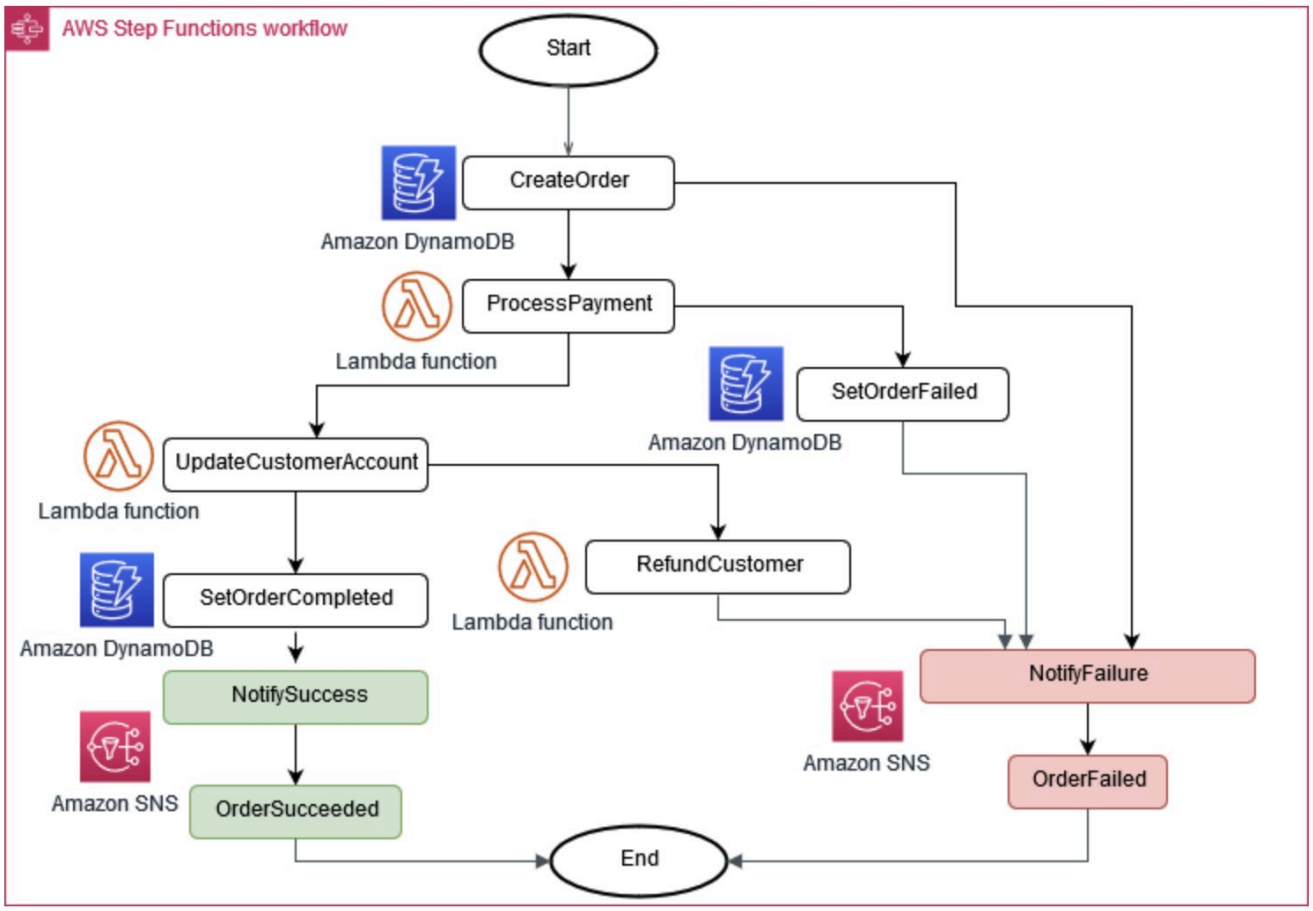


Figura 6: Coordenador de execução da Saga

Uma abordagem comum no gerenciamento de mudanças em microsserviços é o fornecimento de eventos. Cada alteração no aplicativo é registrada como um evento, criando uma linha do tempo do estado do sistema. Essa abordagem não só ajuda a depurar e auditar, mas também permite que diferentes partes de um aplicativo reajam aos mesmos eventos.

O fornecimento de eventos geralmente funciona hand-in-hand com o padrão Command Query Responsibility Segregation (CQRS), que separa a modificação e a consulta de dados em módulos diferentes para melhor desempenho e segurança.

Ativado AWS, você pode implementar esses padrões usando uma combinação de serviços. Como você pode ver na Figura 7, o Amazon Kinesis Data Streams pode servir como sua loja central de eventos, enquanto o Amazon S3 fornece um armazenamento durável para todos os registros de eventos. AWS Lambda, o Amazon DynamoDB e o Amazon API Gateway trabalham juntos para lidar e processar esses eventos.

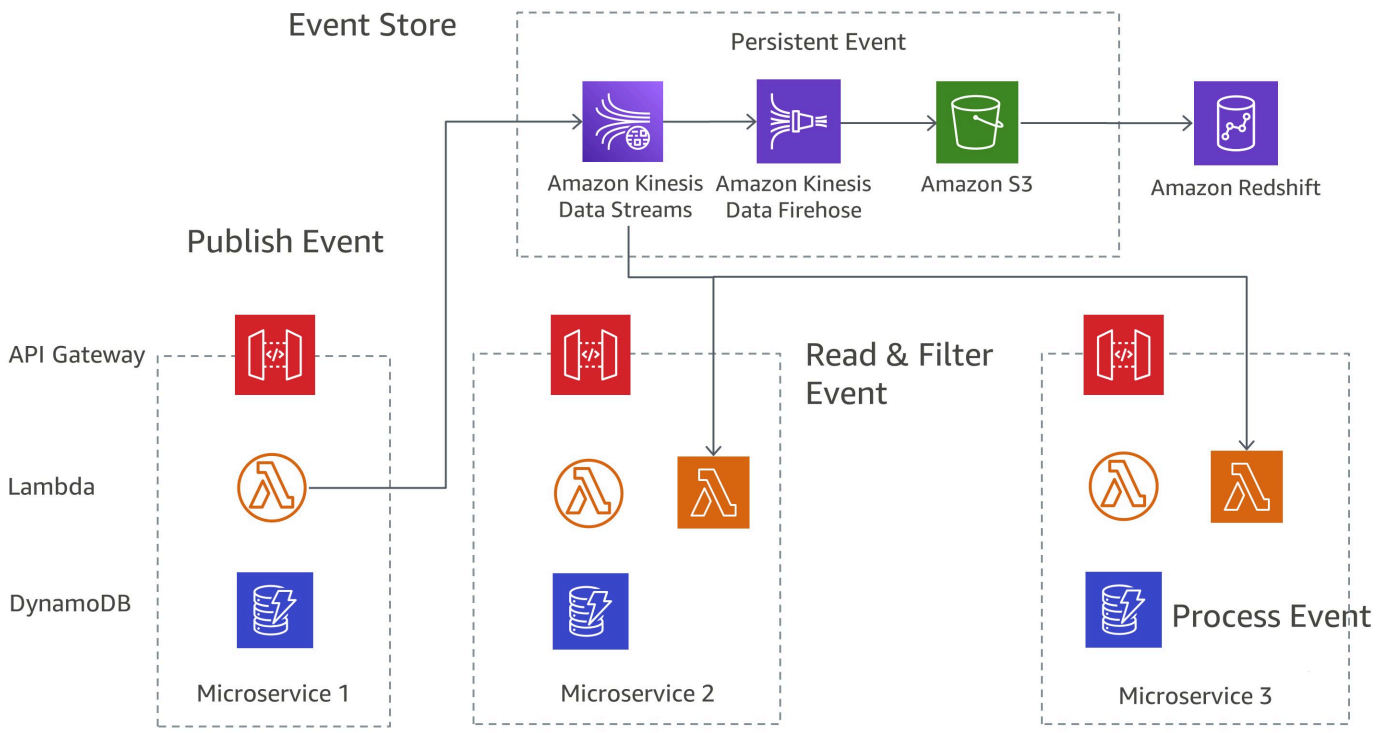


Figura 7: Padrão de fornecimento de eventos em AWS

Lembre-se de que, em sistemas distribuídos, os eventos podem ser entregues várias vezes devido a novas tentativas, por isso é importante projetar seus aplicativos para lidar com isso.

# Gerenciamento de configuração

Em uma arquitetura de microsserviços, cada serviço interage com vários recursos, como bancos de dados, filas e outros serviços. Uma forma consistente de configurar as conexões e o ambiente operacional de cada serviço é vital. Idealmente, um aplicativo deve se adaptar às novas configurações sem precisar ser reiniciado. Essa abordagem faz parte dos princípios do Twelve-Factor App, que recomendam armazenar configurações em variáveis de ambiente.

Uma abordagem diferente é usar o AWS App Config. É um recurso do AWS Systems Manager que facilita aos clientes configurar, validar e implantar sinalizadores de recursos e configuração de aplicativos com rapidez e segurança. Seu sinalizador de recursos e dados de configuração podem ser validados sintática ou semanticamente na fase de pré-implantação e podem ser monitorados e revertidos automaticamente se um alarme que você configurou for acionado. AppConfig pode ser integrado ao Amazon ECS e ao Amazon EKS usando o AWS AppConfig agente. O agente funciona como um contêiner auxiliar executado junto com seus aplicativos de contêiner Amazon ECS e Amazon EKS. Se você usa sinalizadores de AWS AppConfig recursos ou outros dados de configuração dinâmica em uma função Lambda, recomendamos que você adicione a extensão AWS AppConfig Lambda como uma camada à sua função Lambda.

GitOpsé uma abordagem inovadora para o gerenciamento de configurações que usa o Git como fonte confiável para todas as alterações de configuração. Isso significa que todas as alterações feitas em seus arquivos de configuração são automaticamente rastreadas, versionadas e auditadas por meio do Git.

# Gerenciamento de segredos

A segurança é fundamental, portanto, as credenciais não devem ser passadas em texto simples. AWS oferece serviços seguros para isso, como AWS Systems Manager Parameter Store AWS Secrets Manager e. Essas ferramentas podem enviar segredos para contêineres no Amazon EKS como volumes ou para o Amazon ECS como variáveis de ambiente. Em AWS Lambda, as variáveis de ambiente são disponibilizadas automaticamente para seu código. Para fluxos de trabalho do Kubernetes, o operador externo de segredos busca segredos diretamente de serviços, como AWS Secrets Manager criar segredos do Kubernetes correspondentes. Isso permite uma integração perfeita com as configurações nativas do Kubernetes.

# Otimização de custos e sustentabilidade

A arquitetura de microsserviços pode aprimorar a otimização de custos e a sustentabilidade. Ao dividir um aplicativo em partes menores, você pode ampliar somente os serviços que precisam de mais recursos, reduzindo custos e desperdícios. Isso é particularmente útil ao lidar com tráfego variável. Os microsserviços são desenvolvidos de forma independente. Assim, os desenvolvedores podem fazer atualizações menores e reduzir os recursos gastos em testes de ponta a ponta. Durante a atualização, eles precisarão testar apenas um subconjunto dos recursos, em vez dos monólitos.

Os componentes sem estado (serviços que armazenam o estado em um armazenamento de dados externo em vez de em um armazenamento de dados local) em sua arquitetura podem usar as Amazon EC2 Spot Instances, que oferecem EC2 capacidade não utilizada na AWS nuvem. Essas instâncias são mais econômicas do que as instâncias sob demanda e são perfeitas para cargas de trabalho que podem lidar com interrupções. Isso pode reduzir ainda mais os custos e, ao mesmo tempo, manter a alta disponibilidade.

Com serviços isolados, você pode usar opções de computação com custo otimizado para cada grupo de auto-scaling. Por exemplo, o AWS Graviton oferece opções de computação econômicas e de alto desempenho para cargas de trabalho adequadas às instâncias baseadas em ARM.

A otimização dos custos e do uso de recursos também ajuda a minimizar o impacto ambiental, alinhando-se ao pilar de sustentabilidade do Well-Architected Framework. Você pode monitorar seu progresso na redução das emissões de carbono usando a Ferramenta de Pegada de Carbono AWS do Cliente. Essa ferramenta fornece informações sobre o impacto ambiental de seu AWS uso.

# Mecanismos de comunicação

No paradigma de microsserviços, vários componentes de um aplicativo devem se comunicar por meio de uma rede. As abordagens comuns para isso incluem mensagens baseadas em REST, GraphQL, gRPC e assíncronas.

## Comunicação baseada em REST

O HTTP/S protocolo, usado amplamente para comunicação síncrona entre microsserviços, geralmente opera por meio de. RESTful APIs O API Gateway oferece uma maneira simplificada de criar uma API que serve como um ponto de acesso centralizado para serviços de back-end, lidando com tarefas como gerenciamento de tráfego, autorização, monitoramento e controle de versão.

## Comunicação baseada em GraphQL

Da mesma forma, o GraphQL é um método difundido para comunicação síncrona, usando os mesmos protocolos do REST, mas limitando a exposição a um único endpoint. Com AWS AppSync, você pode criar e publicar aplicativos GraphQL que interagem diretamente com AWS serviços e datastores ou incorporar funções Lambda para lógica de negócios.

## Comunicação baseada em gRPC

O gRPC é um protocolo de comunicação RPC síncrono, leve, de alto desempenho e de código aberto. O gRPC aprimora seus protocolos subjacentes usando HTTP/2 e habilitando mais recursos, como compressão e priorização de fluxo. Ele usa o Protobuf Interface Definition Language (IDL), que é codificado em binário e, portanto, tira proveito do enquadramento binário HTTP/2.

## Mensagens assíncronas e transmissão de eventos

As mensagens assíncronas permitem que os serviços se comuniquem enviando e recebendo mensagens por meio de uma fila. Isso permite que os serviços permaneçam frouxamente acoplados e promovam a descoberta de serviços.

As mensagens podem ser definidas dos três tipos a seguir:

- Filas de mensagens: uma fila de mensagens atua como um buffer que separa remetentes (produtores) e destinatários (consumidores) de mensagens. Os produtores colocam as mensagens

na fila e os consumidores as retiram da fila e as processam. Esse padrão é útil para comunicação assíncrona, nivelamento de carga e tratamento de picos de tráfego.

- Publicar-Assinar: no padrão publicar-assinar, uma mensagem é publicada em um tópico e vários assinantes interessados recebem a mensagem. Esse padrão permite a transmissão de eventos ou mensagens para vários consumidores de forma assíncrona.
- Mensagens orientadas por eventos: as mensagens orientadas por eventos envolvem capturar e reagir a eventos que ocorrem no sistema. Os eventos são publicados em um agente de mensagens e os serviços interessados se inscrevem em tipos específicos de eventos. Esse padrão permite um acoplamento frouxo e permite que os serviços reajam aos eventos sem dependências diretas.

Para implementar cada um desses tipos de mensagem, AWS oferece vários serviços gerenciados, como Amazon SQS, Amazon SNS, Amazon, EventBridge Amazon MQ e Amazon MSK. Esses serviços têm recursos exclusivos adaptados às necessidades específicas:

- Amazon Simple Queue Service (Amazon SQS) e Amazon Simple Notification Service (Amazon SNS): Como você pode ver na Figura 8, esses dois serviços se complementam, com o Amazon SQS fornecendo um espaço para armazenar mensagens e o Amazon SNS permitindo a entrega de mensagens para vários assinantes. Eles são eficazes quando a mesma mensagem precisa ser entregue a vários destinos.

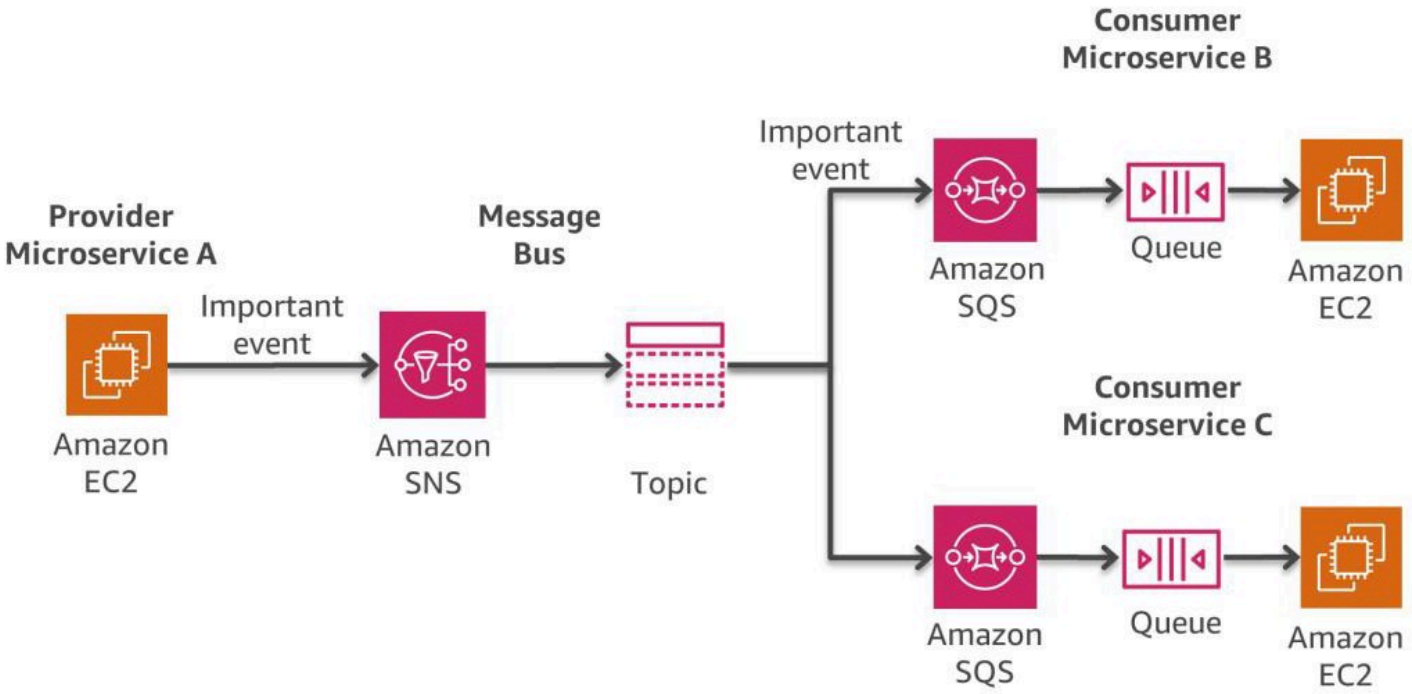


Figura 8: Padrão de barramento de mensagens em AWS

- Amazon EventBridge: um serviço sem servidor que usa eventos para conectar componentes do aplicativo, facilitando a criação de aplicativos escaláveis orientados por eventos. Use-o para rotear eventos de fontes como aplicativos, AWS serviços e software de terceiros desenvolvidos internamente para aplicativos de consumo em toda a sua organização. EventBridge fornece uma maneira simples e consistente de ingerir, filtrar, transformar e entregar eventos para que você possa criar novos aplicativos rapidamente. EventBridge os ônibus de eventos são adequados para o many-to-many roteamento de eventos entre serviços orientados a eventos.
- Amazon MQ: uma boa opção se você tiver um sistema de mensagens preexistente que usa protocolos padrão como JMS, AMQP ou similares. Esse serviço gerenciado substitui seu sistema sem interromper as operações.
- Amazon MSK (Managed Kafka): um sistema de mensagens para armazenar e ler mensagens, útil para casos em que as mensagens precisam ser processadas várias vezes. Ele também suporta streaming de mensagens em tempo real.
- Amazon Kinesis: processamento e análise em tempo real de dados de streaming. Isso permite o desenvolvimento de aplicativos em tempo real e fornece uma integração perfeita com o AWS ecossistema.

Lembre-se de que o melhor serviço para você depende de suas necessidades específicas, por isso é importante entender o que cada um oferece e como eles se alinham às suas necessidades.

## Orquestração e gerenciamento de estado

A orquestração de microsserviços se refere a uma abordagem centralizada, em que um componente central, conhecido como orquestrador, é responsável por gerenciar e coordenar as interações entre microsserviços. Orquestrar fluxos de trabalho em vários microsserviços pode ser um desafio. A incorporação de código de orquestração diretamente nos serviços é desencorajada, pois introduz um acoplamento mais estreito e impede a substituição de serviços individuais.

O Step Functions fornece um mecanismo de fluxo de trabalho para gerenciar as complexidades da orquestração de serviços, como tratamento de erros e serialização. Isso permite que você escale e altere aplicativos rapidamente sem adicionar código de coordenação. O Step Functions faz parte da plataforma AWS sem servidor e oferece suporte às funções Lambda, Amazon EC2, Amazon EKS, Amazon ECS, AI e muito mais. SageMaker AWS Glue

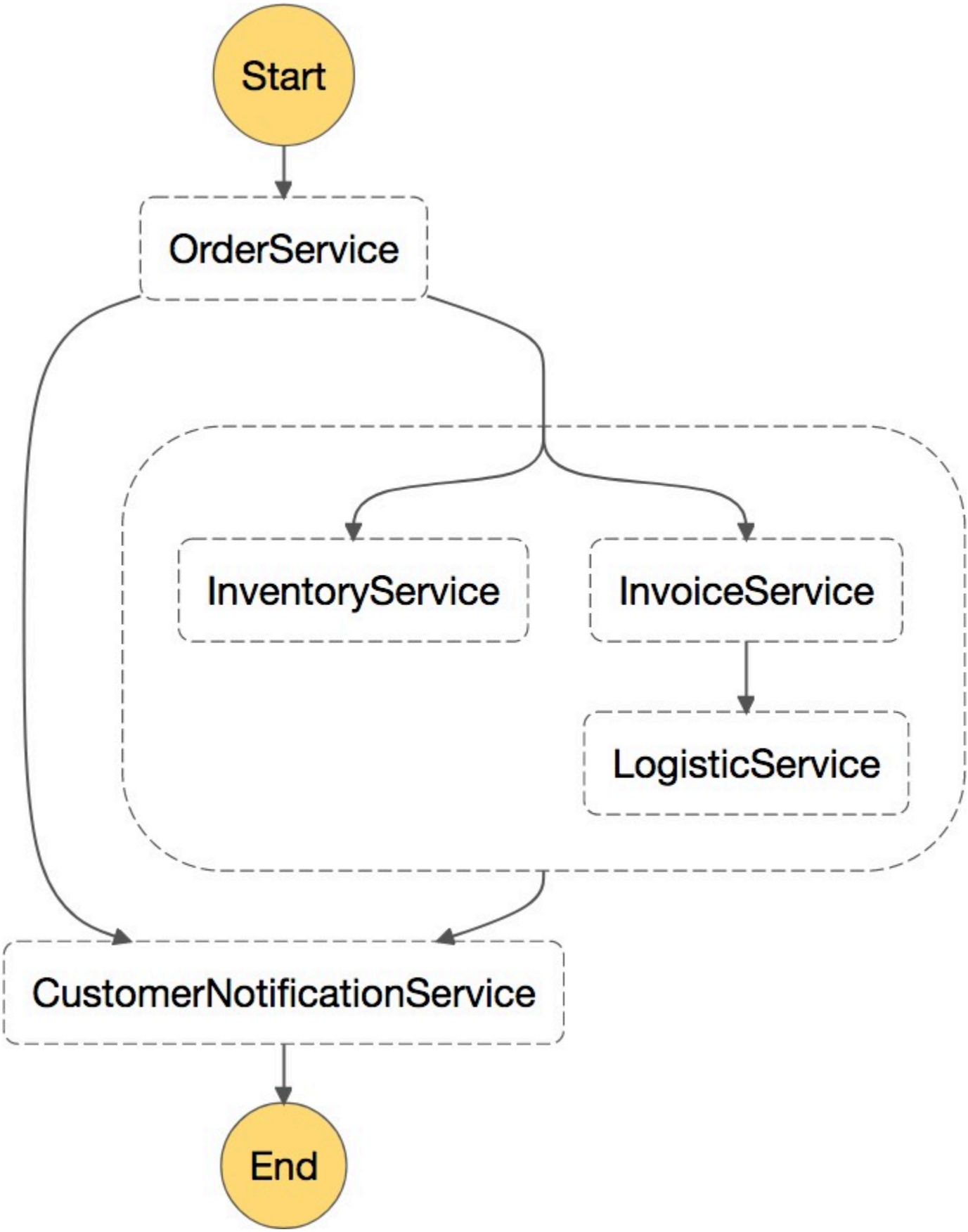
Start
OrderService
InventoryService
InvoiceService
LogisticService
CustomerNotificationService
End

Figura 9: Um exemplo de um fluxo de trabalho de microsserviços com etapas paralelas e sequenciais invocadas por AWS Step Functions

O Amazon Managed Workflows para Apache Airflow (Amazon MWAA) é uma alternativa ao Step Functions. Você deve usar o Amazon MWAA se priorizar o código aberto e a portabilidade. O Airflow tem uma grande e ativa comunidade de código aberto que contribui regularmente com novas funcionalidades e integrações.

# Observabilidade

Como as arquiteturas de microsserviços são inerentemente compostas por muitos componentes distribuídos, a observabilidade em todos esses componentes se torna crítica. A Amazon CloudWatch permite isso, coletando e rastreando métricas, monitorando arquivos de log e reagindo às mudanças em seu AWS ambiente. Ele pode monitorar AWS recursos e métricas personalizadas geradas por seus aplicativos e serviços.

Tópicos


- Monitoramento
- Centralizando registros
- Rastreamento distribuído
- Análise de log em AWS
- Outras opções para análise


## Monitoramento

CloudWatch oferece visibilidade de todo o sistema sobre a utilização de recursos, desempenho de aplicativos e integridade operacional. Em uma arquitetura de microsserviços, o monitoramento de métricas personalizadas CloudWatch é benéfico, pois os desenvolvedores podem escolher quais métricas coletar. O escalonamento dinâmico também pode ser baseado nessas métricas personalizadas.

CloudWatch O Container Insights amplia essa funcionalidade, coletando automaticamente métricas para vários recursos, como CPU, memória, disco e rede. Ele ajuda a diagnosticar problemas relacionados ao contêiner, simplificando a resolução.

Para o Amazon EKS, uma opção geralmente preferida é o Prometheus, uma plataforma de código aberto que fornece recursos abrangentes de monitoramento e alerta. Normalmente é acoplado ao Grafana para visualização intuitiva de métricas. O Amazon Managed Service for Prometheus (AMP) oferece um serviço de monitoramento totalmente compatível com o Prometheus, permitindo que você supervisione aplicativos em contêineres sem esforço. Além disso, o Amazon Managed Grafana (AMG) simplifica a análise e a visualização de suas métricas, eliminando a necessidade de gerenciar a infraestrutura subjacente.

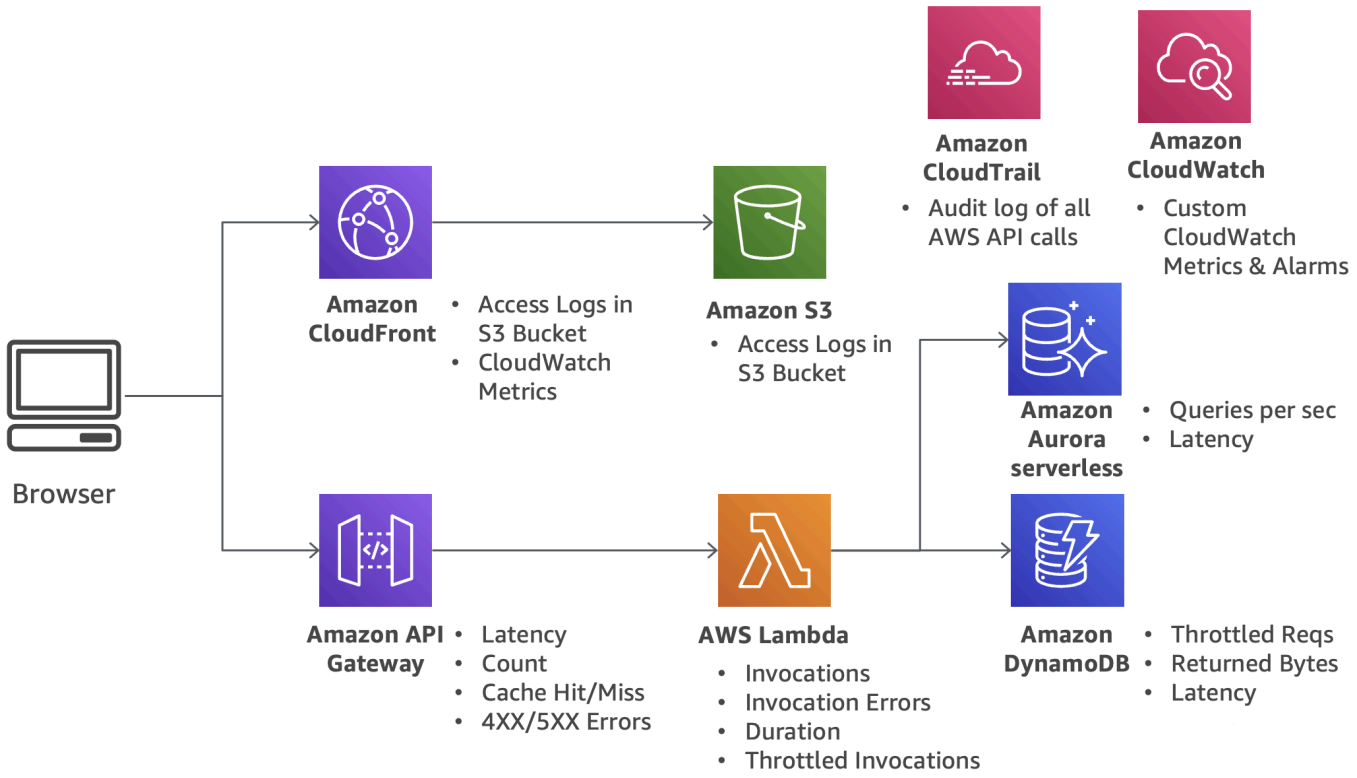


Figura 10: Uma arquitetura sem servidor com componentes de monitoramento

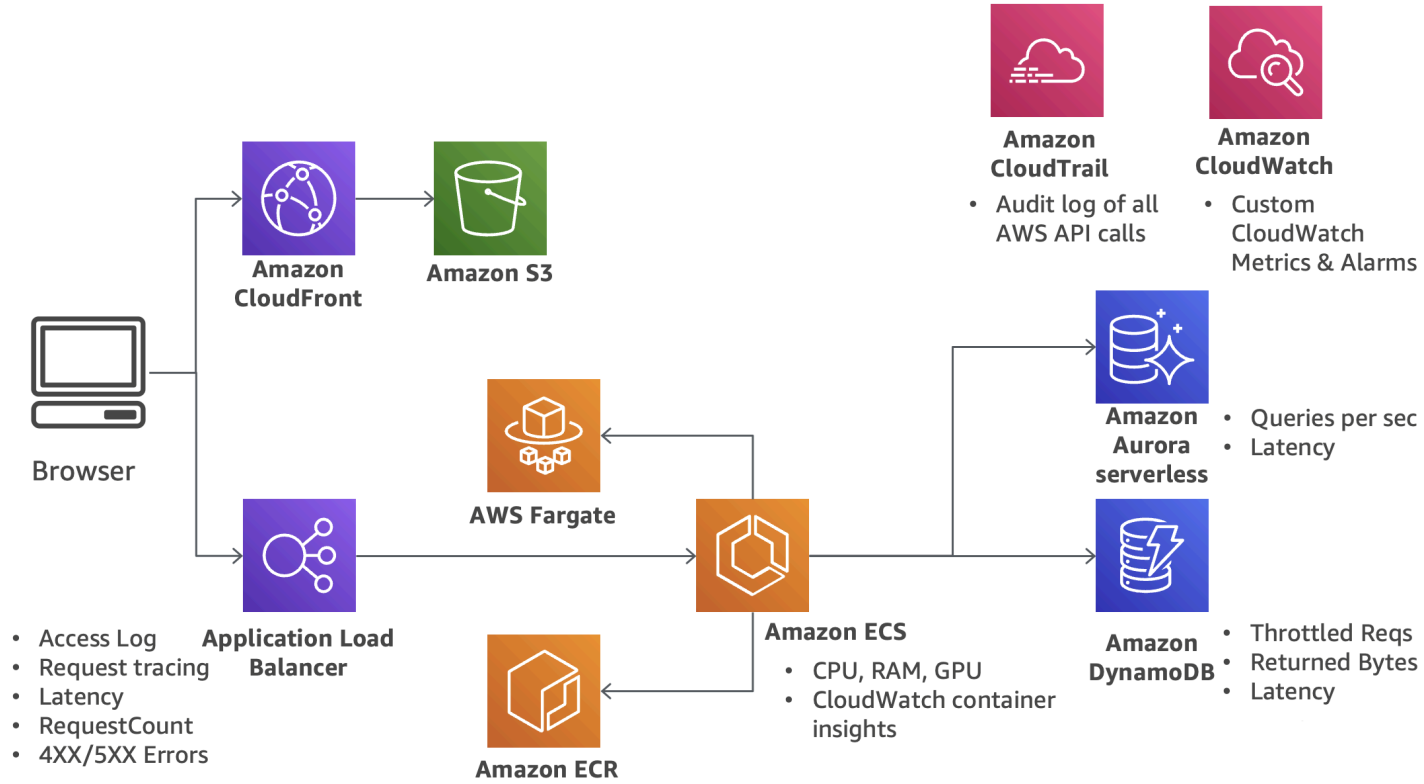

Figura 11: Uma arquitetura baseada em contêiner com componentes de monitoramento

## Centralizando registros

O registro é fundamental para identificar e resolver problemas. Com microsserviços, você pode lançar com mais frequência e experimentar novos recursos. AWS fornece serviços como Amazon S3, CloudWatch Logs e Amazon OpenSearch Service para centralizar arquivos de log. O Amazon EC2 usa um daemon para enviar registros para, CloudWatch enquanto o Lambda e o Amazon ECS enviam nativamente sua saída de log para lá. Para o Amazon EKS, o Fluent Bit ou o Fluentd podem ser usados para encaminhar registros CloudWatch para emissão de relatórios usando OpenSearch o Kibana. No entanto, devido à menor pegada e às vantagens de desempenho, o FluentBit é recomendado em vez do Fluentd.

A Figura 12 ilustra como os logs de vários AWS serviços são direcionados para o Amazon S3 e. CloudWatch Esses registros centralizados podem ser analisados posteriormente usando o Amazon OpenSearch Service, inclusive o Kibana, para visualização de dados. Além disso, o Amazon Athena pode ser empregado para consultas ad hoc em relação aos registros armazenados no Amazon S3.

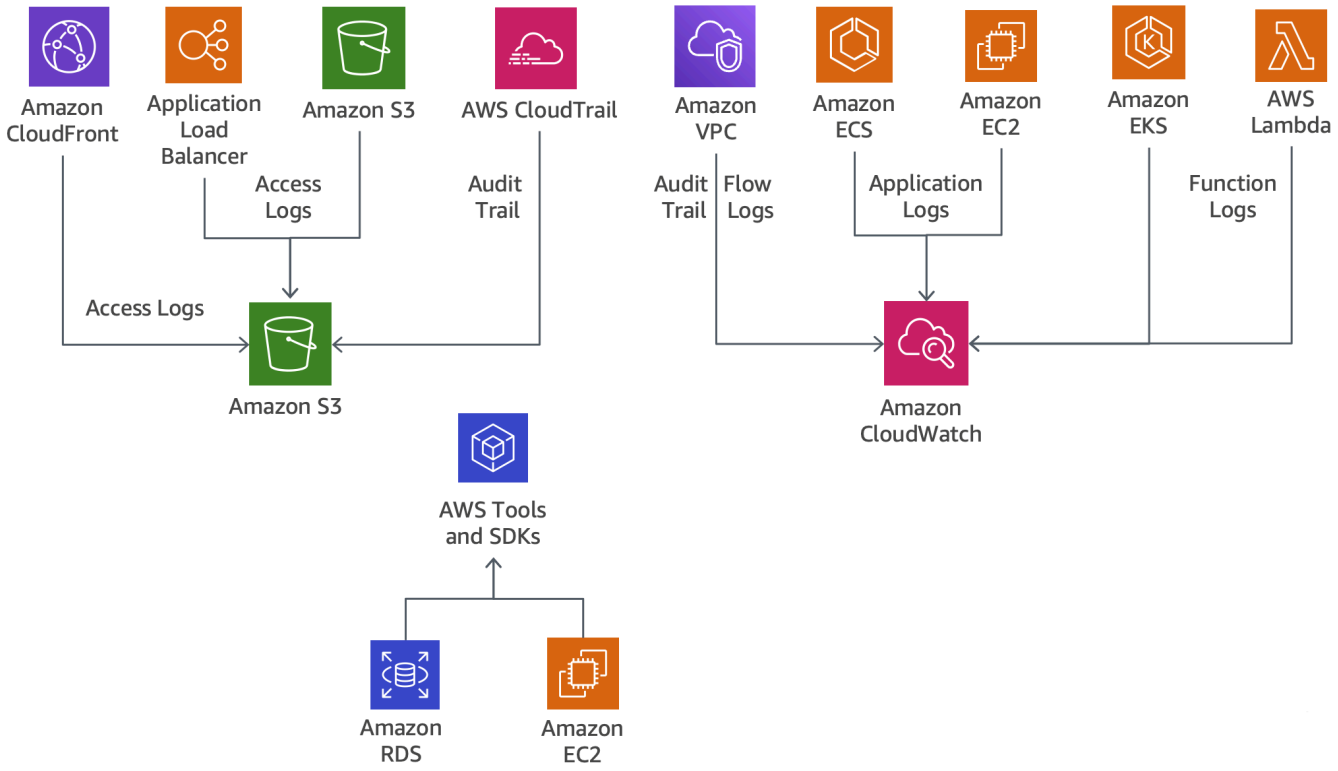


Figura 12: Capacidades de registro dos AWS serviços

# Rastreamento distribuído

Os microsserviços geralmente trabalham juntos para lidar com solicitações. AWS X-Ray usa IDs de correlação para rastrear solicitações nesses serviços. O X-Ray funciona com Amazon EC2, Amazon ECS, Lambda e Elastic Beanstalk.

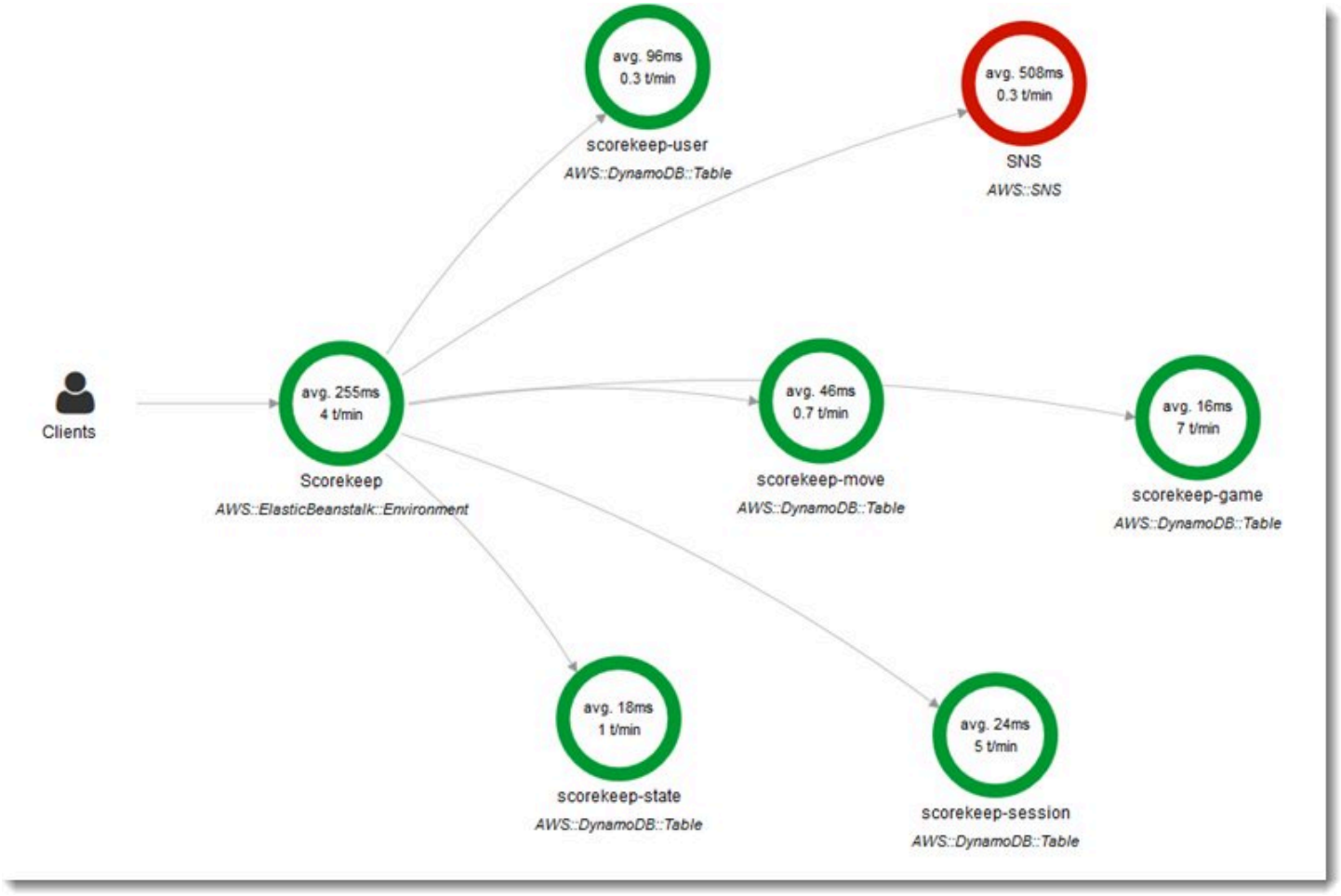


Figura 13: mapa AWS X-Ray de serviços

AWS O Distro OpenTelemetry for faz parte do OpenTelemetry projeto e fornece código aberto e agentes para coletar rastreamentos APIs e métricas distribuídos, melhorando o monitoramento de seus aplicativos. Ele envia métricas e rastreamentos para várias soluções de monitoramento AWS e de parceiros. Ao coletar metadados de seus AWS recursos, ele alinha o desempenho do aplicativo com os dados da infraestrutura subjacente, acelerando a solução de problemas. Além disso, é compatível com uma variedade de AWS serviços e pode ser usado localmente.

# Análise de log em AWS

O Amazon CloudWatch Logs Insights permite a exploração, análise e visualização de registros em tempo real. Para uma análise mais aprofundada dos arquivos de log, o Amazon OpenSearch Service, que inclui o Kibana, é uma ferramenta poderosa. CloudWatch Os registros podem transmitir entradas de registro para o OpenSearch Serviço em tempo real. O Kibana, perfeitamente integrado OpenSearch, visualiza esses dados e oferece uma interface de pesquisa intuitiva.

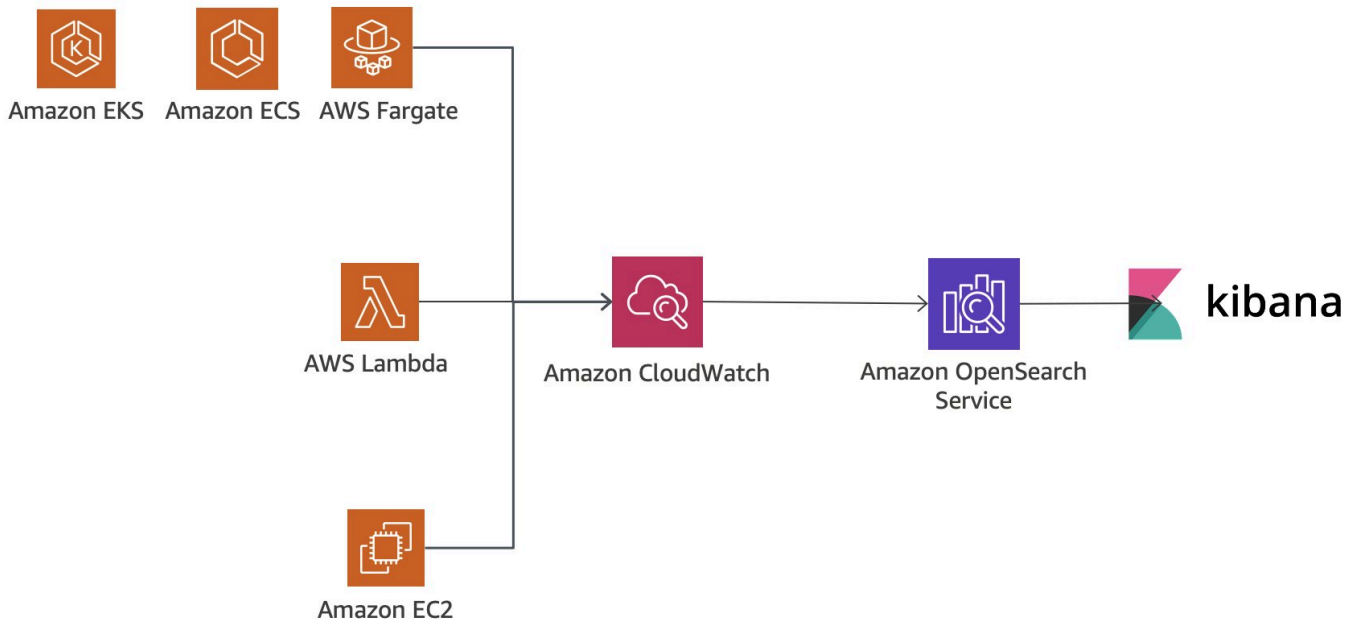


Figura 14: Análise de registros com o Amazon OpenSearch Service

# Outras opções para análise

Para uma análise mais aprofundada dos registros, o Amazon Redshift, um serviço de armazém de dados totalmente gerenciado, e o Quick, um serviço de inteligência comercial escalável, oferecem soluções eficazes. QuickSight fornece conectividade fácil a vários serviços de AWS dados, como Redshift, RDS, Aurora, EMR, DynamoDB, Amazon S3 e Kinesis, simplificando o acesso aos dados.

CloudWatch Os registros podem transmitir entradas de registro para o Amazon Data Firehose, um serviço para fornecer dados de streaming em tempo real. QuickSight em seguida, usa os dados armazenados no Redshift para análise, geração de relatórios e visualização abrangentes.

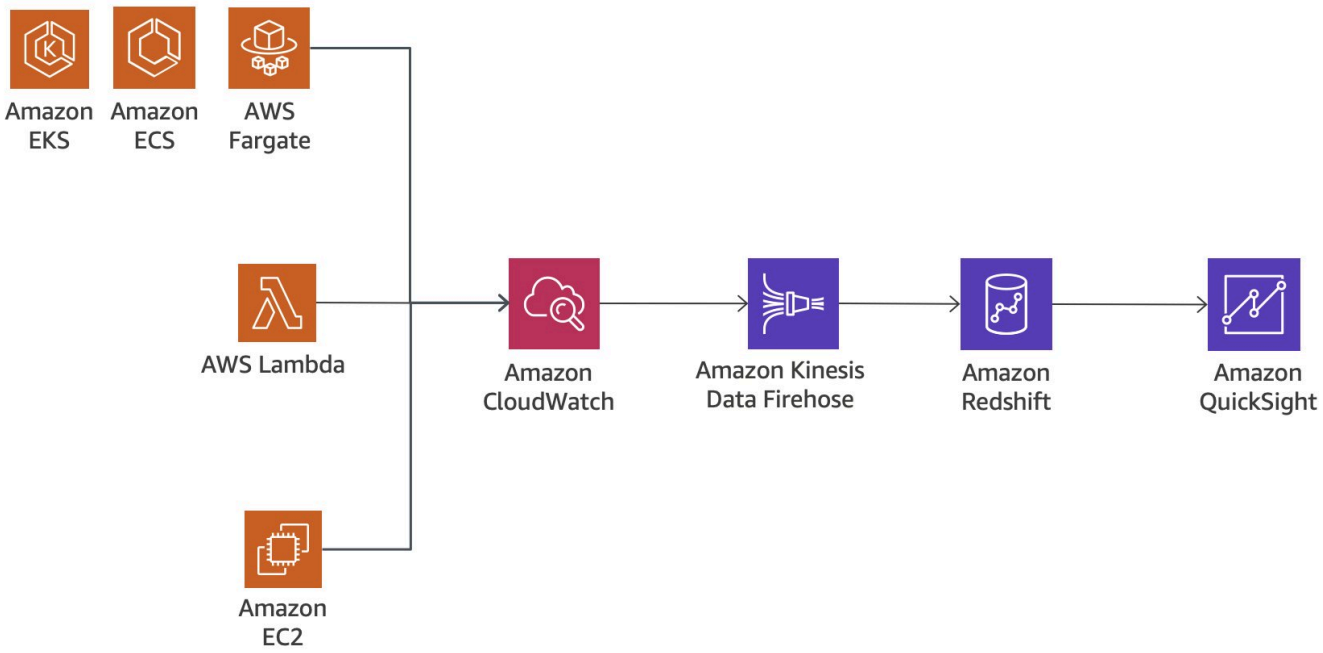


Figura 15: Análise de log com Amazon Redshift e Quick

Além disso, quando os registros são armazenados em buckets do S3, um serviço de armazenamento de objetos, os dados podem ser carregados em serviços como o Redshift ou o EMR, uma plataforma de big data baseada em nuvem, permitindo uma análise completa dos dados de log armazenados.

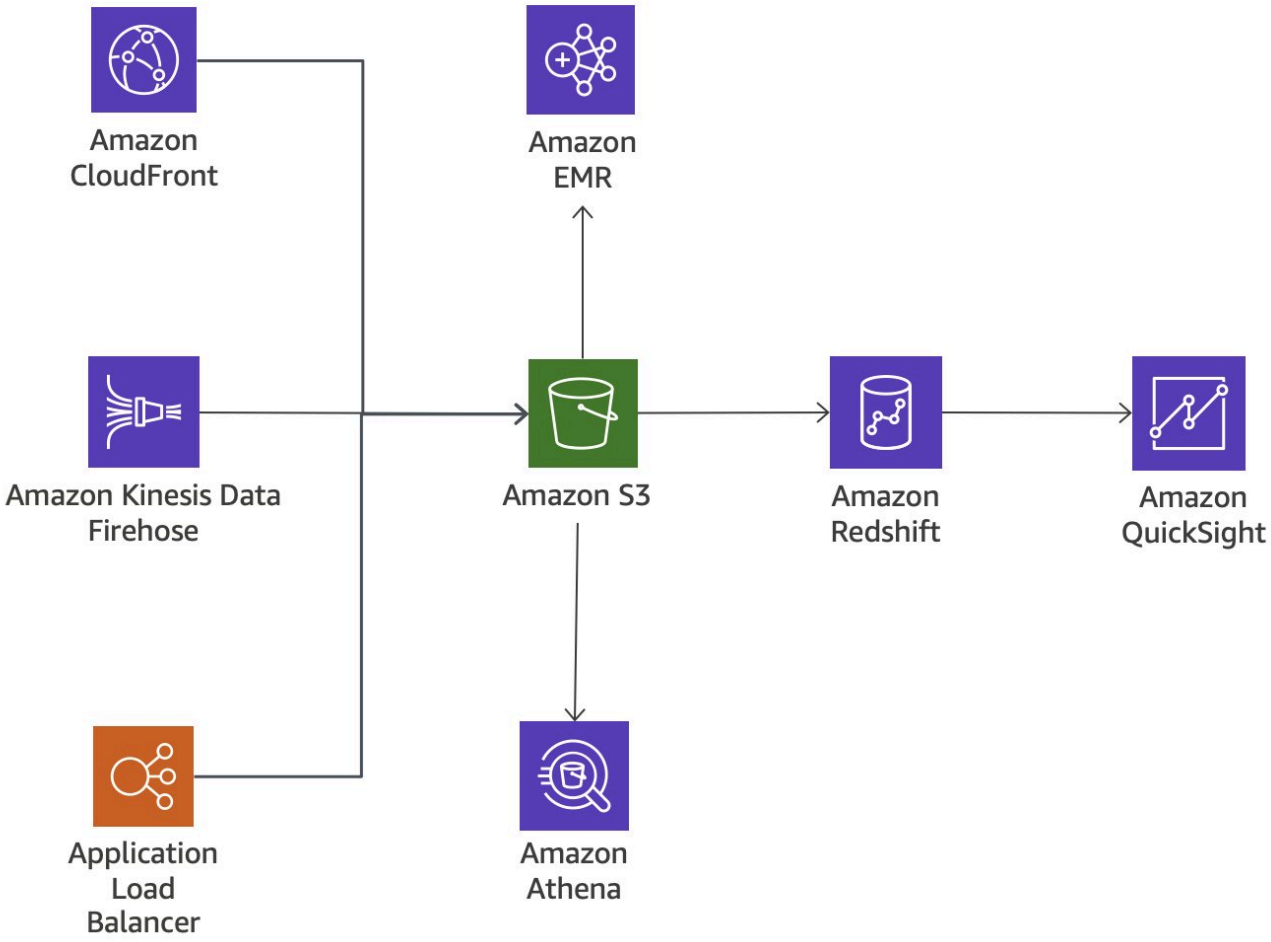


Figura 16: Simplificando a análise de registros: dos AWS serviços aos QuickSight

# Gerenciando a conversação na comunicação de microsserviços

A conversa se refere à comunicação excessiva entre microsserviços, o que pode causar ineficiência devido ao aumento da latência da rede. É essencial gerenciar o bate-papo de forma eficaz para que um sistema funcione bem.

Algumas ferramentas importantes para gerenciar o bate-papo são REST APIs, HTTP e APIs gRPC. APIs O REST APIs oferece uma variedade de recursos avançados, como chaves de API, limitação por cliente, validação de solicitações, AWS WAF integração ou endpoints de API privados. O HTTP APIs é projetado com recursos mínimos e, portanto, tem um preço mais baixo. Para obter mais detalhes sobre esse tópico e uma lista dos principais recursos disponíveis em REST APIs e HTTP APIs, consulte Escolha entre REST APIs e HTTP APIs.

Freqüentemente, os microsserviços usam REST sobre HTTP para comunicação devido ao seu uso generalizado. Mas em situações de alto volume, a sobrecarga do REST pode causar problemas de desempenho. É porque a comunicação usa o handshake TCP, que é necessário para cada nova solicitação. Nesses casos, a API gRPC é a melhor opção. O gRPC reduz a latência, pois permite várias solicitações em uma única conexão TCP. O gRPC também suporta streaming bidirecional, permitindo que clientes e servidores enviem e recebam mensagens ao mesmo tempo. Isso leva a uma comunicação mais eficiente, especialmente para transferências de dados grandes ou em tempo real.

Se a conversa persistir apesar de escolher o tipo certo de API, talvez seja necessário reavaliar sua arquitetura de microsserviços. A consolidação de serviços ou a revisão de seu modelo de domínio pode reduzir o tempo de conversação e melhorar a eficiência.

## Usando protocolos e armazenamento em cache

Os microsserviços geralmente usam protocolos como gRPC e REST para comunicação (consulte a seção anterior Mecanismos de comunicação sobre.) O gRPC usa HTTP/2 para transporte, enquanto o REST normalmente usa HTTP/1.1. O gRPC emprega buffers de protocolo para serialização, enquanto o REST geralmente usa JSON ou XML. Para reduzir a latência e a sobrecarga de comunicação, o armazenamento em cache pode ser aplicado. Serviços como a Amazon ElastiCache ou a camada de cache no API Gateway podem ajudar a reduzir o número de chamadas entre microsserviços.

# Auditoria

Em uma arquitetura de microsserviços, é crucial ter visibilidade das ações do usuário em todos os serviços. AWS fornece ferramentas como AWS CloudTrail, que registra todas as chamadas de API feitas e AWS AWS CloudWatch, que é usada para capturar registros de aplicativos. Isso permite que você acompanhe as mudanças e analise o comportamento em seus microsserviços. A Amazon EventBridge pode reagir rapidamente às mudanças do sistema, notificando as pessoas certas ou até mesmo iniciando fluxos de trabalho automaticamente para resolver problemas.

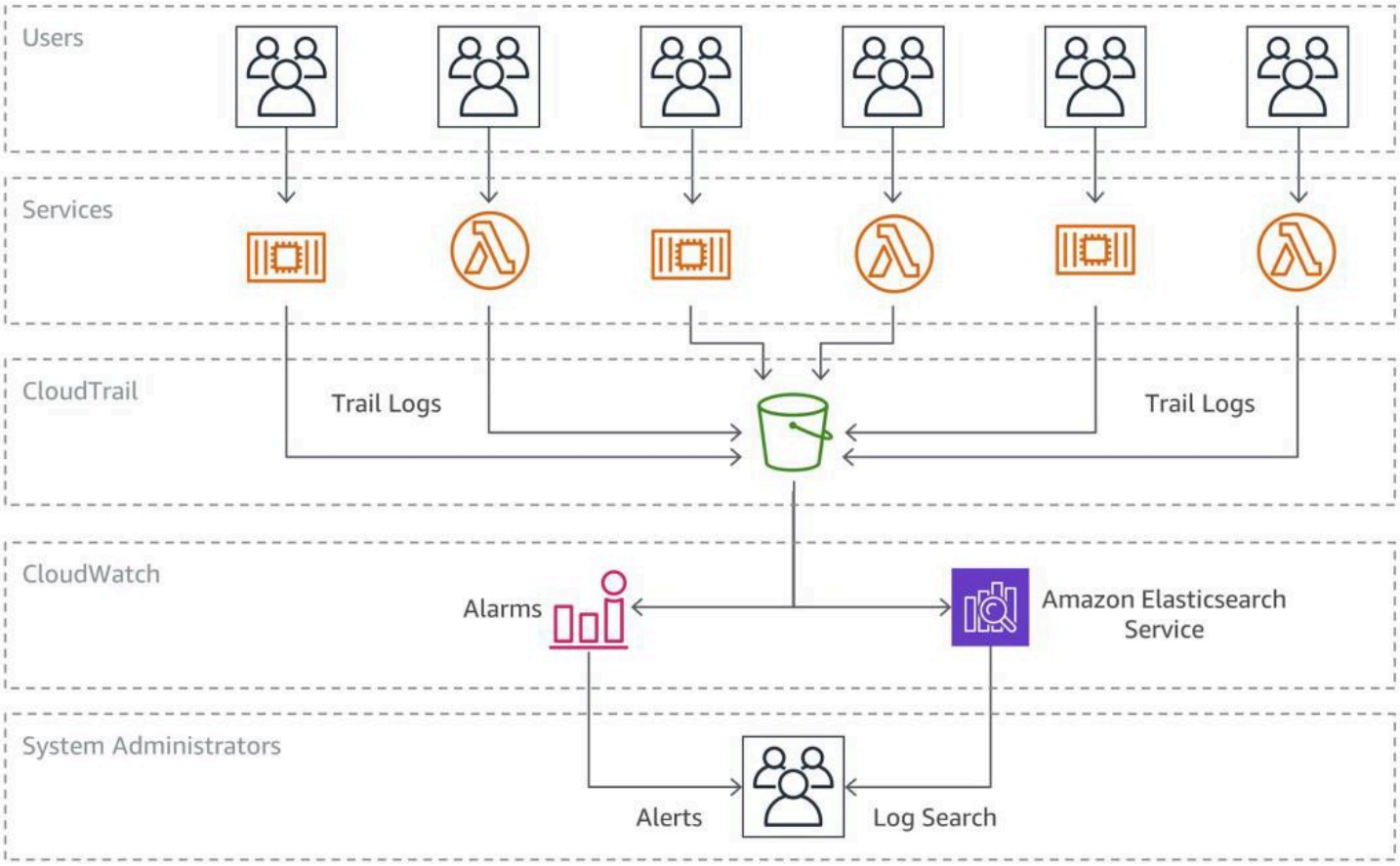


Figura 17: Auditoria e remediação em seus microsserviços

## Inventário de recursos e gerenciamento de mudanças

Em um ambiente de desenvolvimento ágil com configurações de infraestrutura em rápida evolução, a auditoria e o controle automatizados são vitais. Regras do AWS Config fornecem uma abordagem gerenciada para monitorar essas mudanças em microsserviços. Eles permitem a definição de políticas de segurança específicas que detectam, rastreiam e enviam alertas automaticamente sobre violações de políticas.

Por exemplo, se uma configuração do API Gateway em um microsserviço for alterada para aceitar tráfego HTTP de entrada em vez de somente solicitações HTTPS, uma AWS Config regra predefinida poderá detectar essa violação de segurança. Ele registra a alteração para auditoria e aciona uma notificação do SNS, restaurando o estado de conformidade.

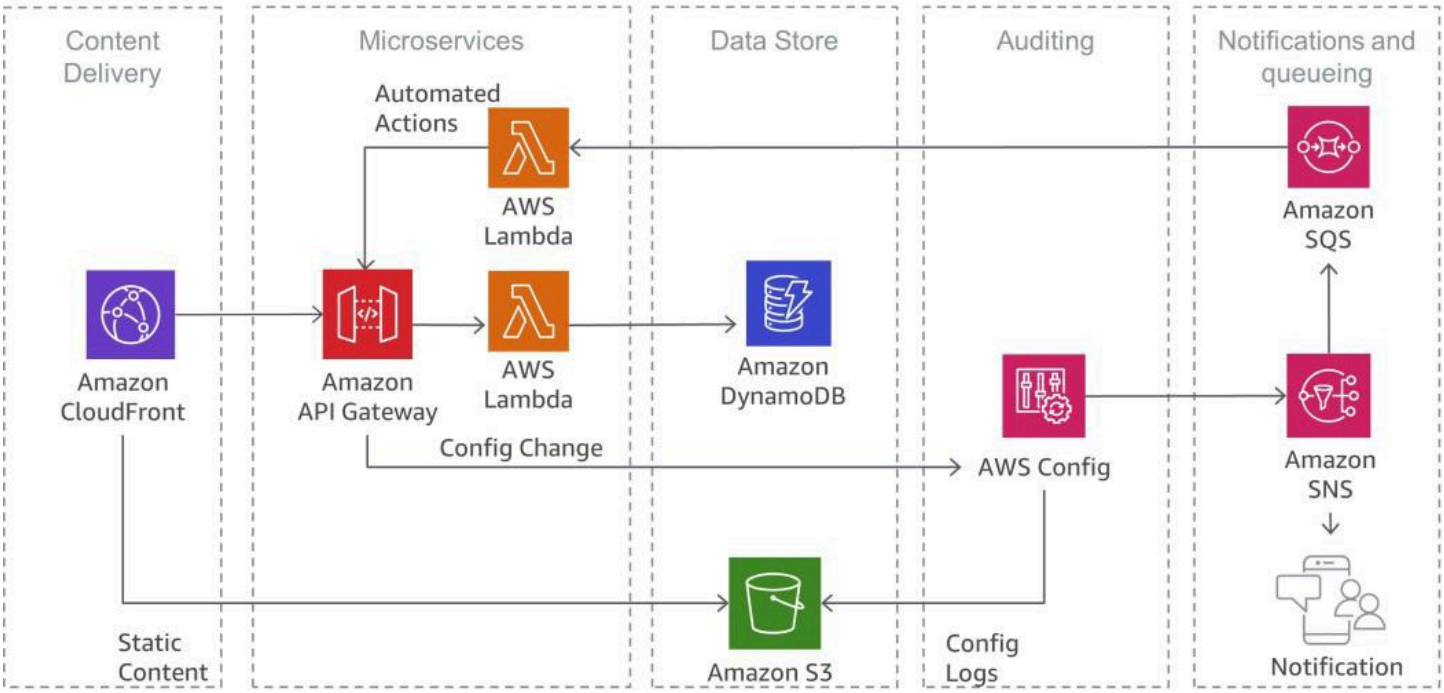


Figura 18: Detectando violações de segurança com AWS Config

# Conclusão

A arquitetura de microsserviços, uma abordagem de design versátil que fornece uma alternativa aos sistemas monolíticos tradicionais, auxilia na escalabilidade de aplicativos, impulsionando a velocidade de desenvolvimento e promovendo o crescimento organizacional. Com sua adaptabilidade, ele pode ser implementado usando contêineres, abordagens sem servidor ou uma combinação dos dois, adaptando-se às necessidades específicas.

No entanto, não é uma one-size-fits-all solução. Cada caso de uso requer uma avaliação meticulosa, dado o aumento potencial na complexidade arquitetônica e nas demandas operacionais. Mas, quando abordados estrategicamente, os benefícios dos microsserviços podem superar significativamente esses desafios. A chave está no planejamento proativo, especialmente nas áreas de observabilidade, segurança e gerenciamento de mudanças.

Também é importante observar que, além dos microsserviços, existem estruturas arquitetônicas totalmente diferentes, como arquiteturas de IA generativa, como Retrieval Augmented Generation (RAG), fornecendo uma variedade de opções para melhor atender às suas necessidades.

AWS, com seu pacote robusto de serviços gerenciados, capacita as equipes a criar arquiteturas de microsserviços eficientes e minimizar a complexidade de forma eficaz. Este whitepaper tem como objetivo orientá-lo nos AWS serviços relevantes e na implementação dos principais padrões. O objetivo é equipá-lo com o conhecimento necessário para aproveitar o poder dos microsserviços AWS, permitindo que você aproveite seus benefícios e transforme sua jornada de desenvolvimento de aplicativos.

# Colaboradores

Os seguintes indivíduos e organizações contribuíram para este documento:

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

# Histórico do documento

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

| Alteração | Descrição | Data |
| --- | --- | --- |
| Atualização principal | Foram adicionadas informaçõ es sobre AWS Customer Carbon Footprint Tool EventBridge, Amazon AWS AppSync (GraphQL) AWS Lambda , Layers, SnapStart Lambda, Large Language Models (LLMs), Amazon Managed Streaming for Apache Kafka (MSK), Amazon Managed Workflows for Apache Airflow (MWAA), Amazon VPC Lattice,. AWS AppConfig Foi adicionada uma seção separada sobre otimização de custos e sustentabilidade. | 31 de julho de 2023 |
| Atualizações menores | Adicionou Well-Architected ao resumo. | 13 de abril de 2022 |
| Whitepaper atualizado | Integração da Amazon EventBridge, AWS OpenTelemetry, AMP, AMG, Container Insights, pequenas alterações de texto. | 9 de novembro de 2021 |
| Atualizações menores | Layout de página ajustado | 30 de abril de 2021 |
| Atualizações menores | Pequenas alterações no texto. | 1 de agosto de 2019 |

| | | |
|---|---|---|
| Whitepaper atualizado | Integração do Amazon EKS, AWS Fargate, Amazon MQ, PrivateLink AWS, AWS App Mesh, AWS Cloud Map | 1.º de junho de 2019 |
| Whitepaper atualizado | Integração dos fluxos de eventos do AWS Step Functions, do AWS X-Ray e do ECS. | 1 de setembro de 2017 |
| Publicação inicial | Implementação de microsserviços na AWS publicada. | 1° de dezembro de 2016 |

> **Note**
>
> Para se inscrever nas atualizações de RSS, você precisa ter um plug-in de RSS habilitado no navegador.

# Avisos

Os clientes são responsáveis por fazer a própria avaliação independente das informações contidas neste documento. Este documento: (a) é apenas para fins informativos, (b) representa ofertas e práticas atuais de AWS produtos, que estão sujeitas a alterações sem aviso prévio, e (c) não cria nenhum compromisso ou garantia de AWS suas afiliadas, fornecedores ou licenciadores. AWS os produtos ou serviços são fornecidos “como estão” sem garantias, representações ou condições de qualquer tipo, expressas ou implícitas. As responsabilidades e obrigações de AWS seus clientes são controladas por AWS contratos, e este documento não faz parte nem modifica nenhum contrato entre AWS e seus clientes.

Copyright © 2023 Amazon Web Services, Inc. ou suas afiliadas.

# AWS Glossário

Para obter a AWS terminologia mais recente, consulte o AWS glossário na Glossário da AWS Referência.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.