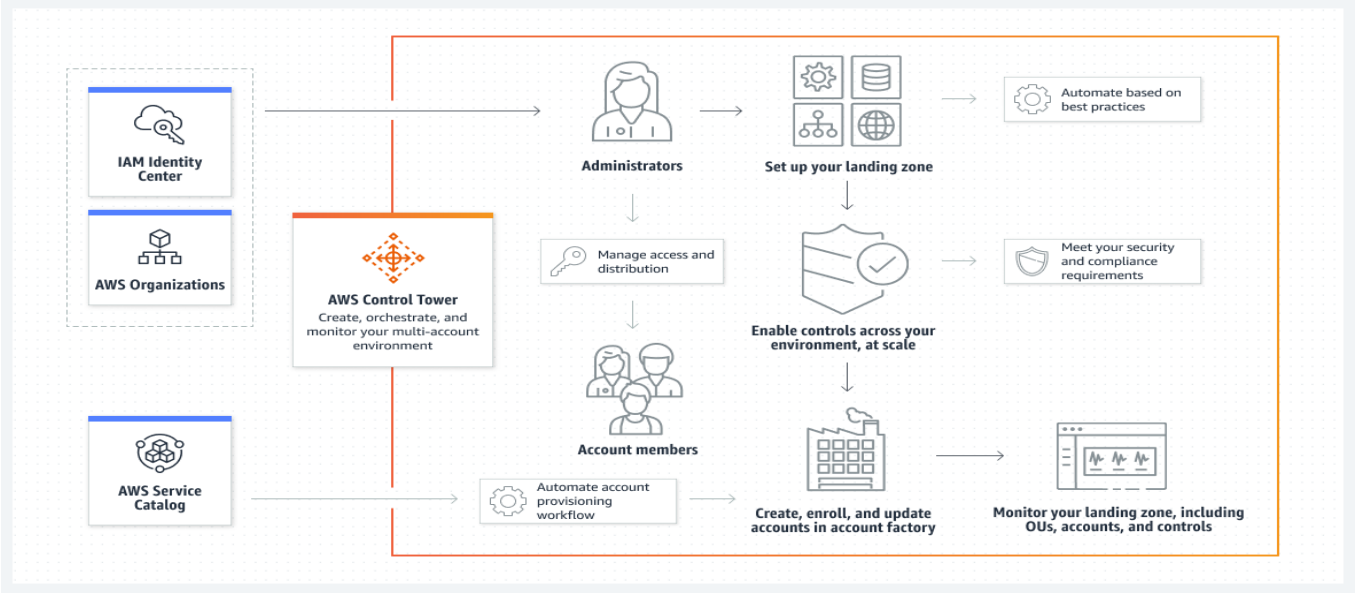
**1. Pergunta**

Categoria: CSAA – Design de Arquiteturas Seguras

Uma empresa precisa limitar a visibilidade e a capacidade de descoberta de cargas de trabalho em contas da AWS com diferentes requisitos administrativos. Usando o AWS Organizations, um arquiteto de soluções e as partes interessadas em contas agruparam as contas em unidades organizacionais (UOs).
É necessária uma solução que monitore alterações na hierarquia da UO e permita que as partes interessadas assinem alertas relacionados.
Qual opção atende a esses requisitos com MENOR sobrecarga administrativa?
**a AWS Control Tower para provisionar as contas da AWS. Habilite notificações de desvio de conta.**

**O AWS Organizations** é um serviço de gerenciamento de contas que permite consolidar várias contas da AWS em uma organização que você cria e gerencia centralmente. O AWS Organizations inclui recursos de gerenciamento de contas e faturamento consolidado que permitem atender melhor às necessidades orçamentárias, de segurança e de conformidade da sua empresa.




**O AWS Control Tower** é um serviço de alto nível que oferece uma maneira simples de configurar e governar um ambiente de múltiplas contas da AWS, seguindo as melhores práticas prescritivas. O AWS Control Tower orquestra os recursos de vários outros serviços da AWS, incluindo AWS Organizations, AWS Service Catalog e AWS IAM Identity Center, para criar uma landing zone em menos de uma hora.

Nesse cenário, os mecanismos de recursos que permitem o gerenciamento do desvio de contas já estão incorporados à Torre de Controle. Isso dispensa configurações adicionais de engenharia e evita a fragmentação da administração para outro serviço. Isso oferece suporte direto ao monitoramento de alterações na hierarquia da UO, como a adição ou remoção de contas, permitindo que as partes interessadas assinem facilmente alertas sobre esses eventos.

Portanto, a resposta correta é: **usar a AWS Control Tower para provisionar as contas da AWS. Habilite notificações de desvio de conta.**

A opção que diz: **Usar a AWS Control Tower para provisionar as contas da AWS. Configurar regras agregadas do AWS Config** está incorreta. Embora as regras do AWS Config possam abranger várias contas, elas foram projetadas principalmente para avaliar configurações de recursos entre contas, não para alterações na hierarquia de uma UO.

A opção que diz: **Usar o AWS Service Catalog para criar contas em organizações. Configurar uma trilha organizacional do AWS CloudTrail para consolidar alterações entre contas** está incorreta, pois as trilhas organizacionais registram todos os eventos da conta de gerenciamento e de todas as contas da organização. Simplesmente entender esse log para uma preocupação muito específica, como alterações na hierarquia de UOs, exigiria ferramentas adicionais.

A opção que diz: **Usar conjuntos de pilhas do AWS CloudFormation para criar recursos em todas as organizações. Iniciar a operação de detecção de desvios em uma pilha para identificar alterações** está incorreta. Embora isso possa detectar alterações em recursos, o foco está apenas na infraestrutura como código e nas configurações de recursos, não especificamente no monitoramento de alterações na hierarquia de Uos.

✅ **Justificativa:**

A pergunta exige uma solução que:

- **Monitore mudanças** na estrutura das UOs
- **Envie alertas** para as partes interessadas
- Ofereça **baixa sobrecarga administrativa**

✔ O AWS Control Tower:

- É uma solução **gerenciada** pela AWS para **governança multi-conta**
- Detecta e alerta sobre **desvios de configuração**, como mudanças nas UOs
- Oferece **notificações automáticas** via Amazon SNS para partes interessadas
- Requer **mínima configuração manual**, reduzindo a sobrecarga administrativa
- Integra-se diretamente com o **AWS Organizations** e fornece visibilidade centralizada

✅ **Resumo Final:**

Para monitorar mudanças na hierarquia de contas em UOs e notificar as partes interessadas com **o menor esforço operacional possível**, a solução mais adequada é usar o **AWS Control Tower** com **notificações de desvio habilitadas**. Essa abordagem aproveita recursos nativos da AWS para **governança e automação**, promovendo controle eficiente com **baixa complexidade de gerenciamento**.

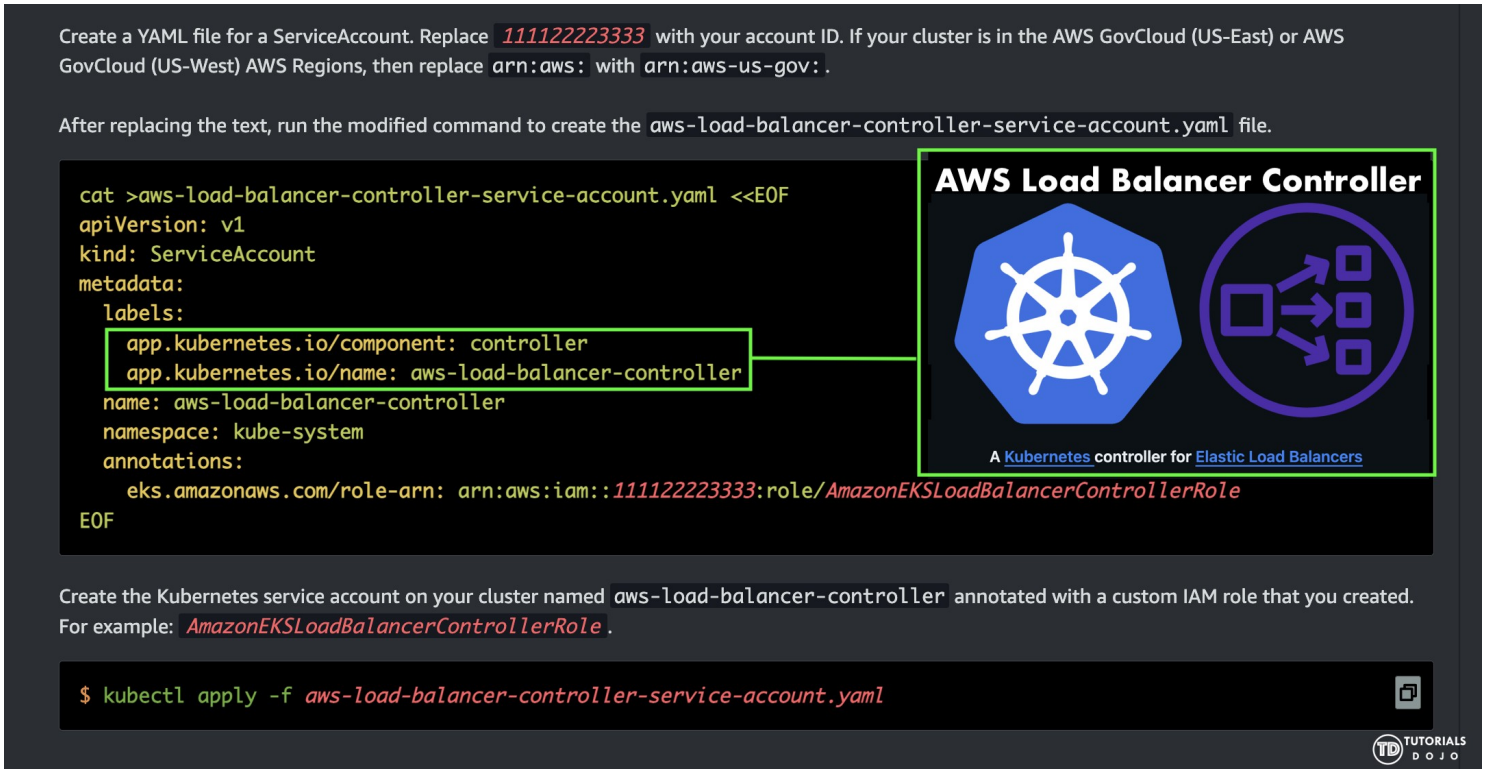
**2. Pergunta**

Categoria: CSAA – Design de Arquiteturas Resilientes

Uma empresa de bens de consumo administra seu site, processa pedidos on-line de clientes e oferece suporte a microsserviços em um cluster do Amazon Elastic Kubernetes Service (Amazon EKS).
É necessária uma solução para rotear solicitações aos serviços hospedados com base em caminhos de URL.
Qual opção atenderá a esse requisito com a MENOR quantidade de configuração?
**Provisione um Application Load Balancer (ALB) usando o AWS Load Balancer Controller.**

**O Amazon Elastic Kubernetes Service (Amazon EKS)** é um serviço gerenciado que elimina a necessidade de instalar, operar e manter seu próprio plano de controle do Kubernetes na Amazon Web Services (AWS).

Um **complemento do Amazon EKS** é um software que fornece recursos operacionais de suporte a aplicativos Kubernetes, mas não é específico para o aplicativo. Isso inclui softwares como agentes de observabilidade ou drivers do Kubernetes que permitem que o cluster interaja com recursos subjacentes da AWS para rede, computação e armazenamento. Por exemplo, o **AWS Load Balancer Controller,** iniciado pela comunidade , simplifica o gerenciamento e o provisionamento de recursos de balanceamento de carga na AWS, inicialmente concebido para balanceamento de carga relacionado ao Ingress e posteriormente expandido para incluir balanceamento de carga relacionado ao Serviço, ou seja, L4 e NLB.




**O AWS Elastic Load Balancing** distribui automaticamente o tráfego de entrada entre vários destinos, como instâncias do EC2, contêineres e endereços IP, em uma ou mais Zonas de Disponibilidade. Ele monitora a integridade dos destinos registrados e encaminha o tráfego apenas para os destinos íntegros. O Elastic Load Balancing dimensiona seu balanceador de carga conforme o tráfego de entrada muda ao longo do tempo. Ele pode ser dimensionado automaticamente para a grande maioria das cargas de trabalho.

A família de produtos ELB inclui:

- Balanceador de carga de aplicação
- Balanceador de carga de rede
- Balanceador de carga de gateway

Nesse cenário, usar o complemento AWS Load Balancer Controller deve ajudar a provisionar o ALB e outros recursos da AWS.

Portanto, a resposta correta é: **Provisionar um Application Load Balancer (ALB) usando o AWS Load Balancer Controller.**

A opção que diz: **Provisionar um Controlador de Entrada NGINX** está incorreta. Ela resultará apenas na mesma funcionalidade de um Ingress provisionado via Controlador do Balanceador de Carga da AWS, mas com uma configuração um pouco mais complexa.

A opção que diz: **Usar uma função do AWS Lambda para enviar solicitações ao EKS** está incorreta, pois um recurso de entrada ainda seria necessário apenas para a conexão da função do Lambda. Implementar o balanceamento de carga no Lambda exigiria um esforço de engenharia nada trivial.

A opção que diz: **Provisionar um balanceador de carga de rede (NLB) usando o controlador do balanceador de carga da AWS** ocorre porque os NLBs operam na camada de transporte (camada 4) e roteiam principalmente o tráfego com base no endereço IP e na porta, não em dados da camada de aplicativo, como caminhos de URL.

✔ **Justificativa:**

A pergunta exige uma solução que:

- Roteie solicitações com base em **caminhos de URL**
- Seja integrada ao **Amazon EKS**
- Requeira a **menor quantidade de configuração possível**

✔ O AWS Load Balancer Controller com ALB:

- É uma **solução nativa da AWS para EKS**, facilitando a integração com Kubernetes
- Permite **roteamento baseado em URL** de forma automática com annotations no Ingress
- **Reduz a complexidade** ao evitar a necessidade de configurar proxies manuais ou serviços externos
- Gera e gerencia automaticamente recursos do ALB com base nas definições de Ingress do Kubernetes
- É **altamente escalável e gerenciado pela AWS**, com menor esforço de manutenção

**✔ Resumo Final:**

Para rotear requisições por caminho de URL em um cluster Amazon EKS com **mínima configuração**, a melhor opção é usar o **AWS Load Balancer Controller** com um **Application Load Balancer**. Essa abordagem oferece **integração direta com os manifests Kubernetes**, reduzindo esforço operacional e oferecendo uma solução eficiente, escalável e compatível com microsserviços modernos.

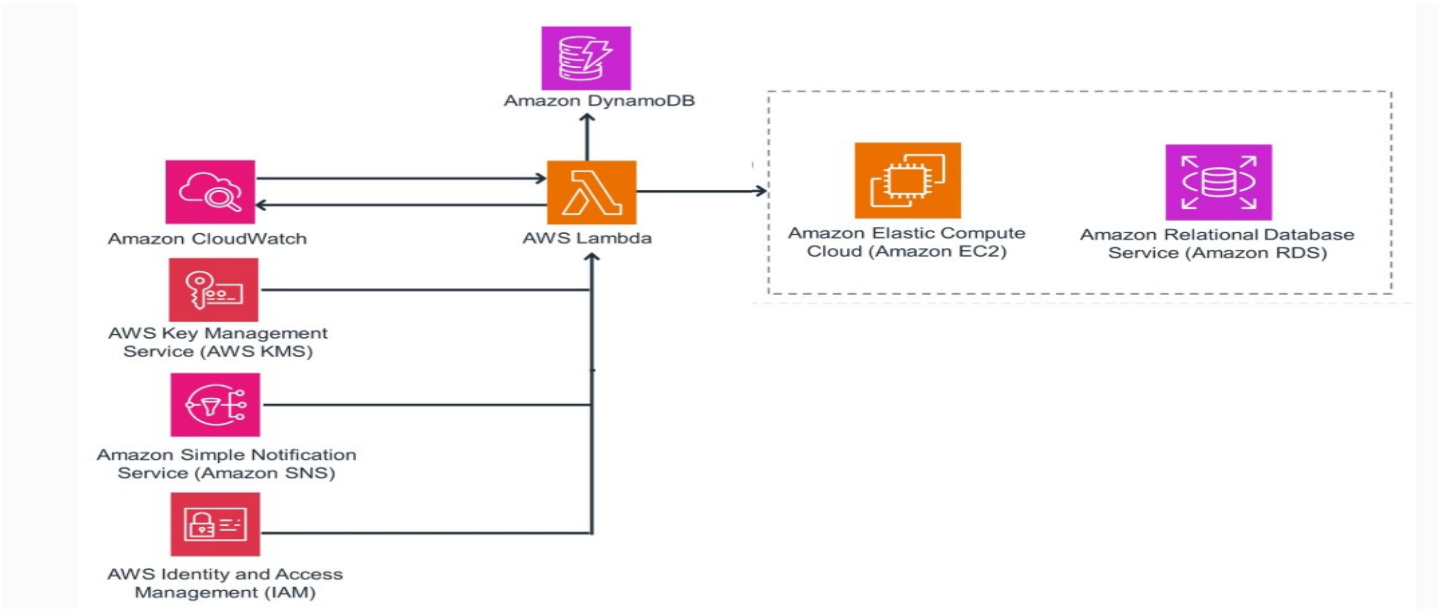
## 3. Pergunta

Categoria: CSAA – Projeto de Arquiteturas Otimizadas em Custo

Uma empresa executa uma aplicação interna na AWS que utiliza instâncias do Amazon EC2 para computação e o Amazon RDS para PostgreSQL para seu armazenamento de dados. Considerando que a aplicação só funciona durante o horário comercial em dias úteis, é necessária uma solução que otimize custos com o mínimo de sobrecarga operacional.
Qual solução atenderia a esses requisitos?
**Implante o modelo do AWS CloudFormation Instance Scheduler on AWS. Configure os agendamentos de início e término da instância do EC2 e da instância do RDS DB.**

A solução **Instance Scheduler na AWS** automatiza a inicialização e a interrupção de instâncias do Amazon Elastic Compute Cloud (Amazon EC2) e do Amazon Relational Database Service (Amazon RDS).

Esta solução ajuda a reduzir custos operacionais, interrompendo recursos que não estão em uso e iniciando-os quando sua capacidade é necessária. Por exemplo, uma empresa pode usar um Agendador de Instâncias na AWS em um ambiente de produção para interromper automaticamente instâncias fora do horário comercial todos os dias. Se você deixar todas as suas instâncias em execução com utilização máxima, esta solução pode resultar em uma economia de custos de até 70% para aquelas instâncias que são necessárias apenas durante o horário comercial normal (utilização semanal reduzida de 168 horas para 50 horas).



*Solução CloudFormation do Agendador de Instâncias da AWS*

O aspecto importante neste cenário é o padrão de uso, que não se enquadra no modelo de uso contínuo assumido por assinaturas de Instâncias Reservadas ou planos de economia de computação. É essencial entender que o Agendador de Instâncias não é um serviço ou recurso da AWS em si, mas um modelo do CloudFormation fornecido pela AWS. Ao implantar este modelo, você pode simplesmente definir os horários de início e término desejados para suas instâncias EC2 e RDS de acordo com o horário de operação do seu aplicativo.

Portanto, a resposta correta é: **Implante o modelo do AWS CloudFormation Instance Scheduler on AWS. Configure os agendamentos de início e término da instância do EC2 e da instância do RDS DB.**
A opção que diz: **Adquirir um plano de economia de computação para EC2 e RDS** está incorreta, pois os planos de economia de computação são aplicáveis apenas ao uso consistente por hora de instâncias de computação. Isso significa que, para períodos em que as instâncias podem estar no status "Interrompidas", elas ainda são cobradas, embora com desconto. Além disso, não há um plano de economia de computação aplicável a instâncias do tipo banco de dados RDS.

A opção que diz: **Criar um alarme do Amazon CloudWatch que acione uma função do AWS Lambda quando a utilização da CPU cair abaixo de um limite de ociosidade. Na função, implementar a lógica para interromper a instância do EC2 e o banco de dados RDS** está incorreta. Essa abordagem apenas fornece uma maneira dinâmica de interromper instâncias com base no uso, em vez de um cronograma fixo. Além disso, interromper instâncias do RDS com base na utilização da CPU pode levar a um tempo de inatividade não intencional durante o horário de trabalho se a utilização cair temporariamente.

A opção que diz: **Comprar assinaturas de instâncias reservadas para EC2 e RDS** está incorreta porque, assim como nos planos de economia de computação, as assinaturas de instâncias reservadas se aplicam apenas ao uso consistente de computação por hora. Portanto, a premissa de disponibilidade constante anularia, se não anularia, o benefício do desconto.

**✅ Justificativa:**

A pergunta exige uma solução que:

- Otimize custos de forma eficiente
- Funcione apenas em horários pré-definidos (comercial, dias úteis)
- Requeira **baixa manutenção e mínima intervenção manual**

✔ O **AWS Instance Scheduler on AWS**:

- É uma **solução oficial da AWS** disponível via modelo CloudFormation
- Permite **automatizar o start/stop** de recursos como EC2 e RDS com base em agendamentos personalizados
- É **altamente personalizável**, com suporte para múltiplos fusos horários e diferentes padrões de uso
- **Reduz significativamente os custos** ao desligar recursos fora do expediente
- Tem **baixa sobrecarga operacional**, sendo fácil de implementar e manter

**✅ Resumo Final:**

Para cargas de trabalho previsíveis e intermitentes (como uso em horário comercial), desligar recursos fora do expediente é a forma mais eficaz de economizar. O **Instance Scheduler on AWS** automatiza esse processo de forma segura, confiável e com **mínimo esforço operacional**, atendendo com precisão os requisitos de **otimização de custo** e **simplicidade de gerenciamento**.

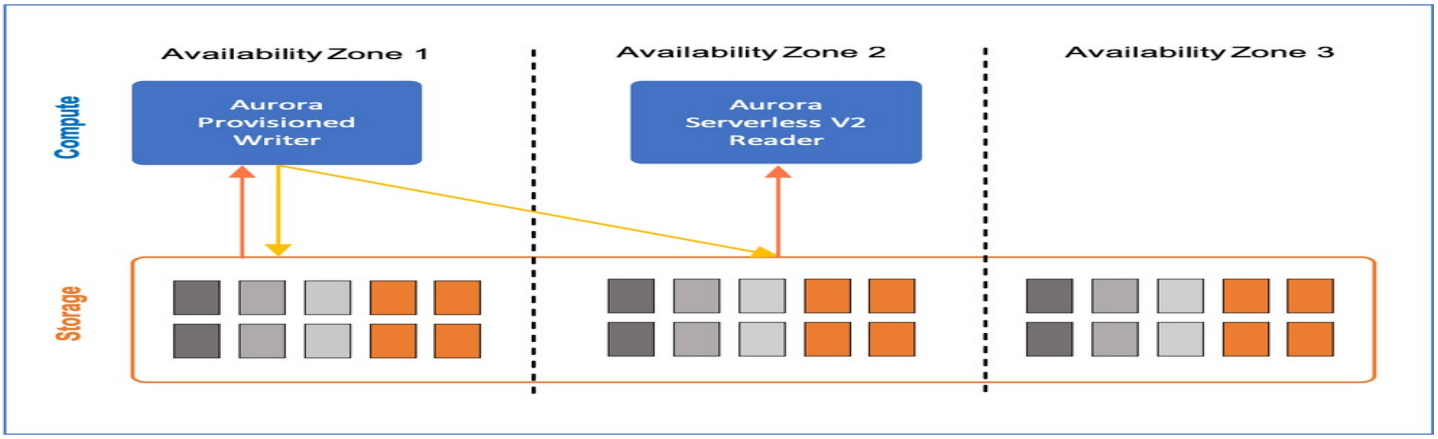
**4. Pergunta**

Categoria: CSAA – Design de Arquiteturas de Alto Desempenho

Uma empresa opera uma aplicação de três camadas com todos os seus componentes hospedados na AWS, exceto a camada de dados. Essa camada de dados consiste em um banco de dados compatível com MySQL, localizado localmente e conectado à AWS por meio de uma VPN Site-to-Site. O consumo de memória do banco de dados varia de 2 a 16 GiB.
Por outro lado, o aplicativo apresenta padrões de tráfego imprevisíveis, incluindo picos e períodos de atividade zero, o que resulta em carga irregular no banco de dados. A empresa deseja substituir

o banco de dados local por um serviço gerenciado que possa se adaptar automaticamente a diferentes demandas.
Qual opção atenderá a esses requisitos?
**Configure um banco de dados Amazon Aurora Serverless v2 com capacidade mínima de 1 e máxima de 8 unidades de capacidade Aurora (ACUs).**

**O Amazon Aurora** é um serviço de banco de dados relacional que combina a velocidade e a disponibilidade de bancos de dados comerciais de ponta com a simplicidade e a relação custo-benefício de bancos de dados de código aberto. O Aurora é totalmente compatível com MySQL e PostgreSQL, permitindo que aplicativos e ferramentas existentes sejam executados sem necessidade de modificações.




Com **o Aurora Serverless V2** , você pode combinar instâncias de leitor/gravador com capacidade provisionada com gravador/leitor sem servidor que correspondam às suas cargas de trabalho. Tomando como exemplo uma carga de trabalho com leitura intensa, mas consistente, e gravações erráticas, o cluster pode ser configurado para incluir instância(s) de leitor/gravador provisionado(s) e um gravador sem servidor.

Nesse cenário, devido ao padrão de tráfego imprevisível e ao fato de que apenas o intervalo de consumo de memória é conhecido, é melhor simplesmente alocar o número correspondente de ACUs e deixar o Aurora Serverless cuidar do escalonamento para cima/baixo. A unidade de medida do Aurora Serverless v2 é a unidade de capacidade do Aurora (ACU). Cada ACU inclui cerca de 2 gibibytes (GiB) de memória, juntamente com a CPU e a rede necessárias. Ao definir um intervalo com um mínimo de 1 ACU e um máximo de 8 ACUs, você pode garantir que o banco de dados seja escalonado de forma eficiente para atender à demanda, do uso mínimo ao pico.

Portanto, a resposta correta é: **Configurar um banco de dados Amazon Aurora Serverless v2 com capacidade mínima de 1 e máxima de 8 unidades de capacidade Aurora (ACUs).**

A opção que diz: **Configurar uma tabela do Amazon DynamoDB com escalonamento automático habilitado. Configurar a tabela com um mínimo de 2 e um máximo de 16 unidades de capacidade** está incorreta. Embora o DynamoDB atenda ao requisito de uma solução sem servidor que possa ser escalonada junto com seu aplicativo, ele não é um banco de dados compatível com MySQL. É um serviço de banco de dados NoSQL, o que exigiria alterações significativas no aplicativo se ele já estivesse projetado para um banco de dados compatível com MySQL.

A opção que diz: **Configurar um banco de dados Amazon Aurora com uma classe de instância de banco de dados otimizada para memória** está incorreta. Embora o Amazon Aurora seja um banco de dados compatível com MySQL que pode aumentar os custos de armazenamento com o uso, uma instância de banco de dados provisionada é fixa e não aumentará/diminuirá com o uso. Isso simplesmente levará ao mesmo problema de superprovisionamento durante períodos de baixa atividade ou subprovisionamento durante picos.

A opção que diz: **Configurar um Amazon RDS para o banco de dados MySQL com 4 gigabytes de memória** está incorreta. Com o RDS para MySQL, a capacidade de computação e memória é fixa. Embora atenda ao requisito de um serviço gerenciado, não possui a capacidade de escalar automaticamente em resposta ao uso do aplicativo, o que o torna menos adequado para lidar com padrões de tráfego imprevisíveis de forma econômica.

✔ **Justificativa:**

A pergunta exige uma solução que:

- Substitua o banco local por um serviço **gerenciado**

- Seja **compatível com MySQL**
- Suporte **carga variável** e tráfego imprevisível
- **Escalone automaticamente** de acordo com a demanda
- Reduza a complexidade operacional

✔ O Amazon Aurora Serverless v2:

- É **compatível com MySQL**, permitindo migração com poucas alterações
- Oferece **escalabilidade automática em incrementos granulares de ACUs**
- Suporta **picos e períodos de inatividade**, cobrando apenas pelo uso real
- Elimina a necessidade de **dimensionamento manual**
- É totalmente gerenciado pela AWS, garantindo **alta disponibilidade e backups automáticos**

✔ **Resumo Final:**

O **Amazon Aurora Serverless v2** é a melhor escolha para cargas de trabalho com tráfego variável e requisitos imprevisíveis de desempenho. Com escalabilidade automática e compatibilidade com MySQL, ele substitui de forma eficiente o banco de dados local, **reduzindo custos operacionais**, **melhorando a disponibilidade** e **ajustando recursos conforme necessário**, sem intervenção manual.

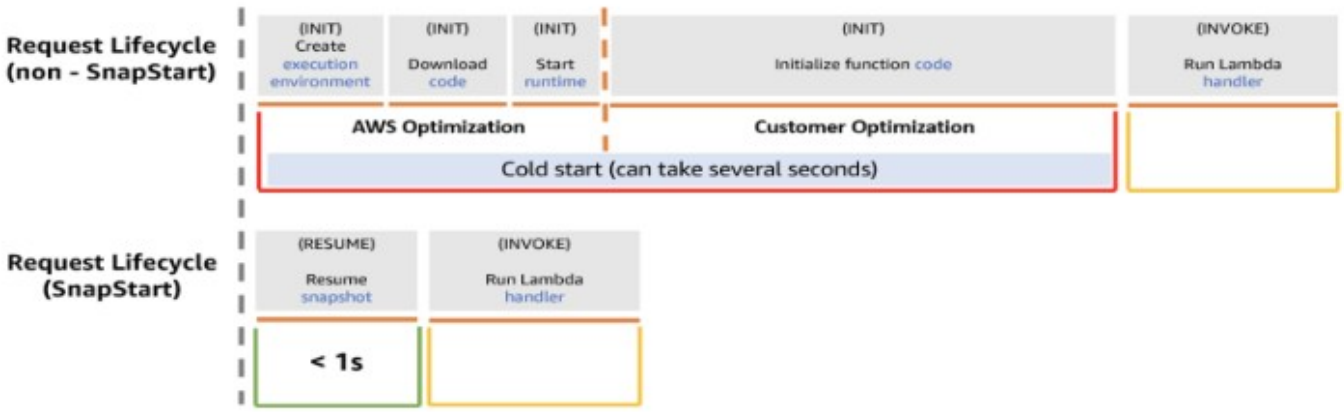
**5. Pergunta**

Categoria: CSAA – Projeto de Arquiteturas Otimizadas em Custo

Uma empresa busca reestruturar seu subsistema para um design orientado a eventos usando o AWS Lambda. No entanto, a empresa tem algumas hesitações, visto que suas cargas de trabalho são todas serviços Java baseados em contêineres. A empresa deseja minimizar inicializações a frio e latências discrepantes ao atender solicitações da maneira mais econômica possível.
Qual atenderia aos requisitos?
**Habilitar o Lambda SnapStart.**

**O AWS Lambda** é um serviço de computação que permite executar código sem provisionar ou gerenciar servidores.

O Lambda executa seu código em uma infraestrutura de computação de alta disponibilidade e realiza toda a administração dos recursos de computação, incluindo manutenção do servidor e do sistema operacional, provisionamento de capacidade e dimensionamento e registro automáticos. Com o Lambda, tudo o que você precisa fazer é fornecer seu código em um dos tempos de execução de linguagem suportados pelo Lambda.




Com **o Lambda SnapStart** para Java, o Lambda inicializa funções conforme novas versões são publicadas. O Lambda então tira um snapshot da microVM Firecracker do estado da memória e do disco

do ambiente de execução inicializado, criptografa o snapshot e o armazena em cache para acesso de baixa latência.

Nesse cenário, aproveitar o ambiente inicializado em cache no Lambda SnapStart é, de certa forma, trapacear nas inicializações a frio.

Portanto, a resposta correta é: **Habilitar Lambda SnapStart** .

A opção que diz: **Habilitar simultaneidade provisionada do Lambda** está incorreta. Considerando apenas os tempos de inicialização a frio, esta pode ser uma opção viável. No entanto, executar com simultaneidade provisionada incorre em custos indiretos que podem não ser justificados durante períodos em que a função sofre invocações mínimas ou nulas.

A opção que diz: **Configurar streaming de resposta para funções Lambda** está incorreta. O streaming de respostas deve melhorar as latências gerais do tempo até o primeiro byte. No entanto, em casos em que o tempo de inicialização anula todas as outras latências, como no caso de tempos de execução Java, esta solução é insuficiente.

A opção que diz: **Configurar camadas Lambda para dependências** está incorreta. Usar camadas Lambda pode resultar em alguma redução no tempo de inicialização, mas é mais comumente utilizado para otimização de build/espaço ou reutilização de dependências.

✅ **Justificativa:**

A pergunta requer uma solução que:

- Minimize **inicializações a frio**, especialmente para funções Java, que têm maior latência nesse cenário
- Reduza latências discrepantes para melhorar a experiência do usuário
- Seja econômica, evitando custos extras com provisionamento ou recursos dedicados

✔ O Lambda SnapStart:

- **Pré-inicializa** as funções Lambda baseadas em Java
- Cria um snapshot do ambiente pronto para execução, eliminando o overhead da inicialização a frio
- Reduz significativamente a latência inicial sem necessidade de manter funções provisionadas (reduzindo custos)
- É uma funcionalidade integrada e simples de ativar, sem necessidade de arquiteturas adicionais

✅ **Resumo Final:**

Para workloads Java baseados em Lambda, o SnapStart é a melhor solução para reduzir inicializações a frio e latências variáveis de forma econômica. Ele proporciona desempenho consistente e resposta rápida, alinhando-se perfeitamente com o objetivo da empresa de ter um design orientado a eventos eficiente e com custo otimizado.

**6. Pergunta**

Categoria: CSAA – Projeto de Arquiteturas Otimizadas em Custo

Uma empresa utiliza instâncias do Amazon EC2, Amazon RDS e Amazon S3 para executar seu aplicativo. Durante uma revisão recente de seus custos de infraestrutura, a empresa notou padrões de gastos incomuns.
A empresa quer monitorar os custos de uso e enviar alertas aos departamentos apropriados quando houver gastos incomuns em sua carga de trabalho.
Qual opção atenderá a esses requisitos?
**No console do AWS Billing and Cost Management, crie um monitor de custos usando o AWS Cost Anomaly Detection.**

**A Detecção de Anomalias de Custo da AWS** é um recurso de Gerenciamento de Custos da AWS. Este recurso utiliza modelos de aprendizado de máquina para detectar e alertar sobre padrões de gastos anômalos nos seus serviços da AWS implantados.


uso do AWS Cost Anomaly Detection inclui os seguintes benefícios:

– receber alertas individualmente em relatórios agregados, seja em uma mensagem de e-mail ou em um tópico do Amazon SNS.
– avaliar padrões de gastos usando métodos de aprendizado de máquina para minimizar alertas de falsos positivos. Por exemplo, você pode avaliar a sazonalidade semanal ou mensal e o crescimento natural.
– investigar a causa raiz da anomalia, como a conta, o serviço, a região ou o tipo de uso da AWS que está impulsionando o aumento de custos.
– configurar como avaliar seus custos. Escolha se deseja analisar todos os seus serviços da AWS de forma independente ou analisar contas de membros específicas, tags de alocação de custos ou categorias de custos.

Portanto, a resposta correta é: **No console do AWS Billing and Cost Management, crie um monitor de custos usando o AWS Cost Anomaly Detection.**

A opção que diz: **Criar um modelo de orçamento de gasto zero no AWS Budgets** está incorreta porque o modelo de orçamento de gasto zero apenas emite um alerta quando seus gastos excedem os limites do Nível Gratuito da AWS. No entanto, ele fornece apenas alertas básicos e não oferece monitoramento avançado nem detecção de anomalias.

A opção que diz: **Usar o AWS Cost Explorer e habilitar dados plurianuais com granularidade mensal** está incorreta. Essa solução ajudaria na análise de gastos ao longo do tempo, mas, principalmente, não alertará ativamente sobre anomalias ou padrões de gastos incomuns.

A opção que diz: **Habilitar o Amazon CloudWatch para monitorar custos e detectar gastos incomuns** está incorreta. Embora o Amazon CloudWatch possa ser configurado para emitir alertas com base na `EstimateCharges` métrica, ele não tem capacidade de determinar se as violações de um limite definido são devidas a gastos incomuns ou a aumentos normais/esperados.

✅ **Justificativa:**

A pergunta exige uma solução que:

- Monitore os custos de forma contínua
- Identifique padrões incomuns ou anômalos de gastos
- Envie alertas automáticos para os departamentos responsáveis
- Seja integrada e fácil de configurar

✔ O AWS Cost Anomaly Detection:

- É um serviço gerenciado que utiliza machine learning para detectar anomalias de custo

- Permite configurar alertas personalizados e grupos responsáveis para notificações
- Oferece monitoramento automatizado e em tempo real dos gastos
- Facilita o controle financeiro e a resposta rápida a desvios inesperados

✅ **Resumo Final:**

Para monitorar gastos incomuns e alertar departamentos automaticamente, a melhor solução é usar o AWS Cost Anomaly Detection. Essa ferramenta nativa da AWS permite detectar rapidamente anomalias de custo e enviar notificações, ajudando a empresa a controlar despesas e agir preventivamente com mínima intervenção manual.

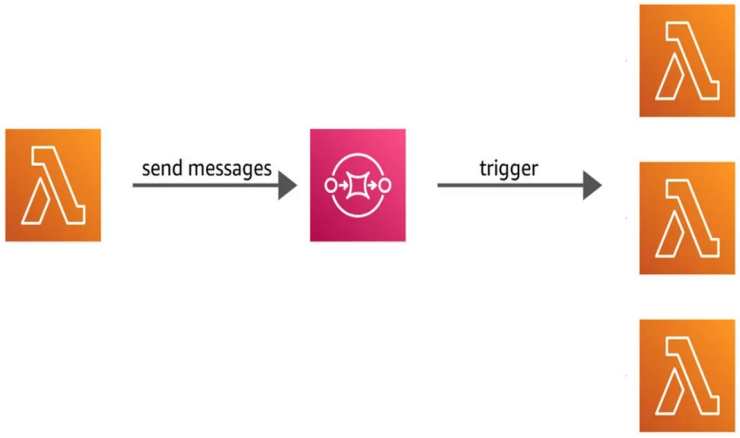
## 7. Pergunta

Categoria: CSAA – Design de Arquiteturas de Alto Desempenho

Uma empresa está refatorando a arquitetura de um aplicativo para aproveitar uma função do AWS Lambda e o Amazon API Gateway. O aplicativo recebe blobs JSON de 5 KB, os processa e armazena o resultado em um banco de dados do Amazon Aurora. O aplicativo só precisa confirmar quando aceita uma solicitação, não quando a processa.
Inicialmente, o aplicativo não atendia aos critérios de aceitação devido a inúmeros erros de limitação durante o processamento de dados. Para resolver esse problema, a empresa teve que aumentar o número padrão de execuções simultâneas do Lambda diversas vezes.
Como um arquiteto de soluções pode melhorar a escalabilidade da infraestrutura para reduzir erros de limitação?
**Reescreva o código da função Lambda em duas funções: uma para receber as informações e outra para persistir as informações no banco de dados. Use o Amazon Simple Queue Service (Amazon SQS) para sinalizar o trabalho disponível para a segunda função Lambda.**

**O Amazon Simple Queue Service (Amazon SQS)** oferece uma fila hospedada segura, durável e disponível que permite integrar e desacoplar sistemas e componentes de software distribuídos. O Amazon SQS oferece estruturas comuns, como filas de mensagens mortas e tags de alocação de custos.

O Amazon SQS fornece uma API genérica de serviços web que você pode acessar usando qualquer linguagem de programação suportada pelo AWS SDK. Um único assinante normalmente processa mensagens na fila. Isso não significa necessariamente que um único consumidor consuma toda a fila em série. Por exemplo, pode haver execuções simultâneas de uma função do AWS Lambda, cada uma consumindo um item de fila diferente. O importante é que os itens da fila geralmente sejam consumidos apenas uma vez. Em alguns casos de uso, vários assinantes precisam atuar no mesmo item; o Amazon SQS e o Amazon SNS são frequentemente usados juntos para criar um aplicativo de mensagens fanout nesses cenários.




**O Amazon SNS** é um serviço de publicação e assinatura que fornece entrega de mensagens de publicadores (também conhecidos como produtores) para vários endpoints de assinantes (também conhecidos como consumidores). Os publicadores se comunicam de forma assíncrona com os assinantes enviando mensagens para um tópico, que é um ponto de acesso lógico e canal de comunicação. Os assinantes podem assinar um tópico do Amazon SNS e receber mensagens publicadas usando um tipo de endpoint compatível, como Amazon Data Firehose, Amazon SQS, Lambda, HTTP, e-mail, notificações push

móveis e mensagens de texto móveis (SMS). O Amazon SNS atua como um roteador de mensagens e entrega mensagens aos assinantes em tempo real. Se um assinante não estiver disponível no momento da publicação da mensagem, ela não será armazenada para recuperação posterior.

Neste cenário, podemos dividir a lógica da aplicação em duas funções Lambda: uma para receber solicitações e outra para processá-las, com o Amazon SQS servindo como uma ponte entre elas. Essa configuração permite que o SQS atue como um buffer, armazenando os dados recebidos quando a função de processamento estiver ocupada. Ele gerencia picos de tráfego de forma eficaz, garantindo que nenhum dado seja perdido durante períodos de alta demanda. Você pode criar um mapeamento de origem de eventos para a fila do SQS para invocar a função Lambda de processamento. Se a função falhar devido à limitação, o Lambda implementará uma estratégia de back-off para tentar executar a função novamente, reduzindo a ocorrência de erros de limitação.

Portanto, a resposta correta é: **reescreva o código da função Lambda em duas funções: uma para receber as informações e outra para persistir as informações no banco de dados. Use o Amazon Simple Queue Service (Amazon SQS) para sinalizar o trabalho disponível para a segunda função Lambda.**

A opção que diz: **Reescreva o código da função Lambda em um aplicativo Go executado em uma instância do Amazon EC2. Usar drivers Go nativos para conexão com o banco de dados** está incorreta. Essa abordagem apenas mudaria o aplicativo de uma arquitetura sem servidor para uma baseada em servidor, o que introduz um conjunto diferente de desafios de escalabilidade e gerenciamento. Embora Go seja conhecido por seu desempenho eficiente e suporte à simultaneidade, a mera portabilidade da lógica do aplicativo para Go envolve esforços de desenvolvimento significativos com retorno mínimo.

A opção que diz: **Migrar o armazenamento de dados do Aurora para o Amazon DynamoDB. Configurar um cluster do DynamoDB Accelerator (DAX) entre o aplicativo e o banco de dados DynamoDB. Configurar o aplicativo para rotear solicitações do DynamoDB para o cluster DAX usando o SDK do cliente DAX** está incorreta. Embora o DynamoDB e o DAX sejam soluções sem servidor, eles não são adequados para o cenário de uma carga de trabalho com uso intenso de gravação. O DAX é um cache na memória desenvolvido especificamente para o Dynamo, a fim de atender a cargas de trabalho mais balanceadas ou com uso intenso de leitura.

A opção que diz: **Reescreva o código da função Lambda em duas funções – uma para receber as informações e outra para persistir as informações no banco de dados. Usar o Amazon Simple Notification Service (Amazon SNS) para sinalizar trabalho disponível para a segunda função Lambda** está incorreta, pois o SNS geralmente é usado para comunicação baseada em fanout/push um-para-muitos. Enquanto isso, as informações a serem recebidas pela segunda função precisam ser processadas apenas uma vez e recebidas somente quando o receptor puder processá-las.

**✅ Justificativa:**

A pergunta exige uma solução que:

- Melhore a escalabilidade do processamento
- Reduza erros de limitação do Lambda
- Confirme a solicitação rapidamente, mesmo sem processar os dados imediatamente

✔ A separação em duas funções Lambda usando SQS:

- Permite desacoplar o recebimento da solicitação do processamento de dados
- SQS funciona como buffer, gerenciando a fila de trabalho e controlando a taxa de processamento
- Reduz a pressão sobre a função Lambda que grava no banco, minimizando erros de limitação
- Garante resposta rápida ao cliente, confirmando a aceitação da solicitação imediatamente
- Proporciona melhor controle da escalabilidade e resiliência da aplicação

**✅ Resumo Final:**

Para melhorar a escalabilidade e evitar erros de limitação do AWS Lambda, a melhor prática é **desacoplar o processamento** usando uma fila gerenciada pelo Amazon SQS. Essa abordagem permite respostas rápidas ao cliente e um processamento controlado dos dados, garantindo eficiência, confiabilidade e menor custo operacional.