**Introdução**

**Regiões:**

Conjunto de Data Centers

Cada região está separa umas das outras

Projetado para funcionar independentemente de outra região

Existe diferença do valor cobrado para o mesmo serviço entre as regiões

A região Norte da Virginia (us-east-1) é onde os serviços chegam primeiro e normalmente mais barato

No console da AWS é disponibilizado somente os recursos que estão vinculados à região especificada

Os recursos não são replicados entre regiões da AWS automaticamente

Os custos são disponibilizados por região, caso exista mais de uma região sendo usada

Cada região possui pelo menos 2 Zonas de Disponibilidade

**Zonas de Disponibilidade:**

Uma AZ é um conjunto de data centers, sendo redundante e seguro

Projetado para trabalhar de forma isolada das demais AZs

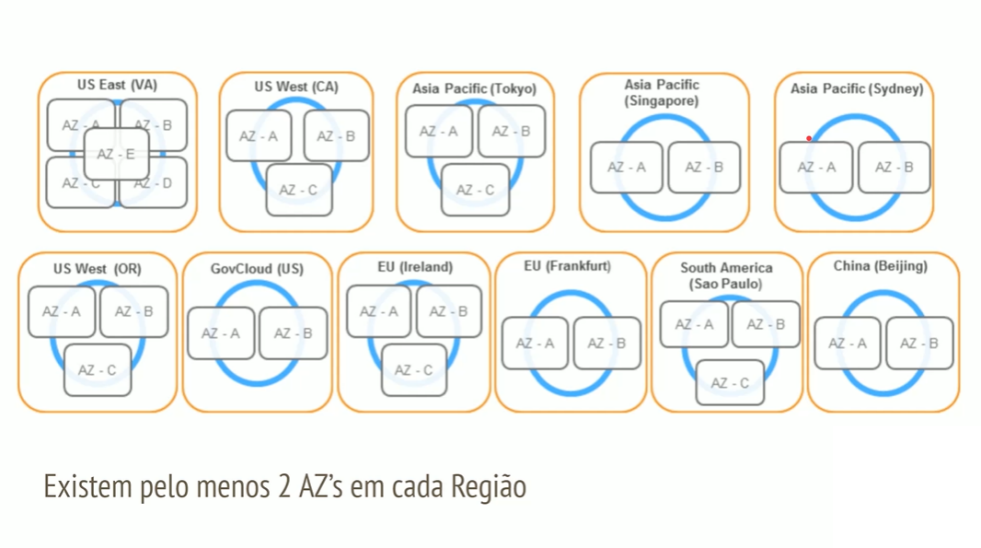
Cada região possui pelo menos 2 Zonas de Disponibilidade

A região física é AZ e o conjunto delas é uma região, ex: região de São Paulo com 3 AZs

As AZ ficam geograficamente distantes umas das outras para que, caso ocorra uma catástrofe, todas não sejam afetadas

Elas são construídas a uma distância que possibilite uma interconexão de rede de baixa latência

Exemplo de Regiões e Zonas de Disponibilidade:



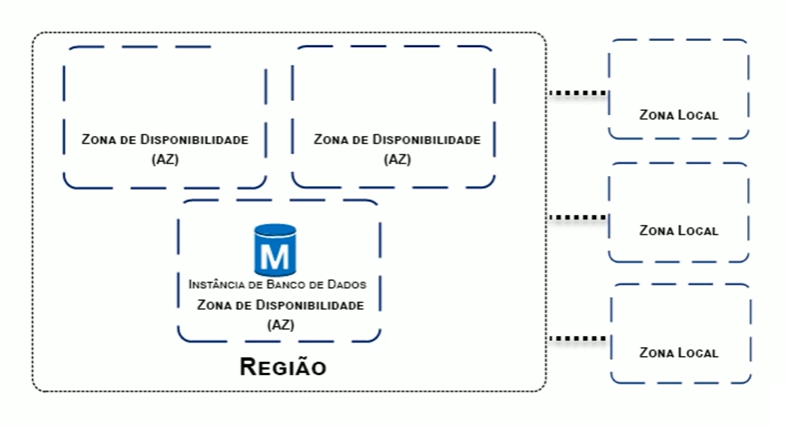
**Zonas Locais:**

Permite que se utilize serviços seletos, como os de computação e armazenamento, mais pertos do usuário

Os usuários têm acesso de baixíssima latência aos aplicativos executados no local

São conectadas a região principal por meio de uma rede privada de banda larga muito alta e redundante

Garante confiabilidade de execução de serviços de uma Zona Local dentro da Região



Nível Gratuito da AWS

O nível gratuito da AWS oferece a capacidade de explorar e testar gratuitamente serviços da AWS até os limites especificados para cada serviço

**Acessar o link:** https://aws.amazon.com/pt/free/

**Existem 3 tipos de gratuidade:**



**O que é Serverless?**

Serverless (sem servidor) é uma **arquitetura nativa da nuvem** **orientada a eventos** e que permite transferir mais das suas responsabilidades operacionais à AWS

Ela permite criar e executar aplicativos e serviços sem se preocupar com a infraestrutura em que esses aplicativos estão rodando e elimina as tarefas de gerenciamento de servidores

Deixa de ser necessário tarefas como o provisionamento de servidores ou de clusters, patches, manutenção do sistema operacional e provisionamento de capacidade

Podemos criar servidores para praticamente qualquer tipo de aplicativo ou serviço de back-end, e a AWS cuida de tudo o que for necessário para executar e escalar aplicativos com alta disponibilidade, reduzindo custos

Permite focar mais no desenvolvimento da aplicação e menos em configuração ou gerenciamento de servidores

**Como funciona de fato?**

**1. Primeiro vamos relembrar a Arquitetura Padrão (tradicional):**

Servidor 24h ligado mesmo sem usuários utilizando

Servidor está sempre ouvindo as requisições dos diferentes clients

Para escalar temos que subir mais máquinas

Mas sempre temos ao menos 1 máquina online para caso algum usuário acesse

Além destes pontos, precisamos estimar o fluxo de usuários e provisionar capacidade da máquina para receber esta demanda estimada

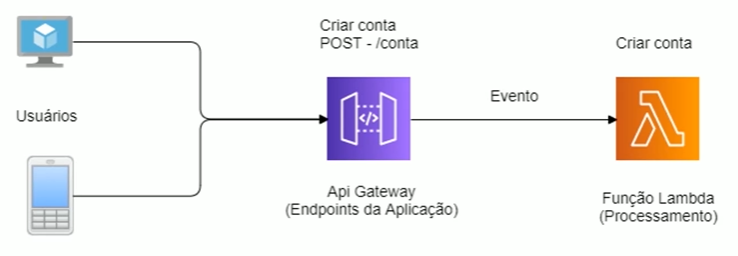
**2. Já na Arquitetura Serverless (sem servidor):**

Ainda existem servidores, porém eles funcionam de uma maneira um pouco diferente

Não existe um servidor online “ouvindo” as requisições, quando chega uma requisição a AWS sobe uma máquina para processá-la e depois a encerra

Ou seja, a partir de um Evento, uma Função Lambda é ativada para processar uma tarefa

Só paga pelo tempo em que a Função Lambda subiu, processou e devolveu a resposta, ou seja, só paga pelo tempo de processamento, ou seja, quanto mais tempo uma Função Lambda ficar processando mais caro fica



Exemplo 2: um arquivo chega no S3 Bucket, é possível programar eventos para serem disparados a acionar Funções Lambdas que importa o arquivo para Dynamo DB



Exemplo 3: suponha que todo dia ao meio dia aconteça um processamento, podemos usar o Cloud Watch para disparar um evento que executa uma Função Lambda



É claro que conforme a necessidade, como por exemplo várias requisições, diversas Funções Lambdas podem ficar ativas ao mesmo tempo

Ou seja, existem várias formas de se invocar uma Função Lambda, e ela faz tudo o que um servidor comum faz

Apenas devemos ter em mente que a máquina é mais limitada, e que a inicialização do código deve ser o mais rápido possível

Ou seja, as Funções Lambdas devem ser concisas e bem definidas, para que sua inicialização e processamento seja rápido e valha a pena

**Entendendo AWS Lambda**

O AWS Lambda permite que se execute código sem provisionar ou gerenciar servidores. Paga-se apenas pelo tempo de computação consumido

Basta carregar o código e o Lambda se encarrega de todos os itens necessários para executar e permitir que o código seja escalável e com alta disponibilidade

O AWS Lambda executa automaticamente o código sem exigir que você provisione ou gerencie servidores. Basta escrever o código e fazer upload para o Lambda

A Escalabilidade é sempre. O código é executado em paralelo e processa cada acionamento individualmente, escalando precisamente continua. O Lambda escala automaticamente as aplicações executando código em resposta a cada acionamento de acordo com o tamanho da carga de trabalho, ou seja, se chegar 5 requisições Lambdas, cada uma será processada individualmente em paralelo

Possui um **medidor de fração de segundo**. Com o AWS Lambda, você é **cobrado a cada 100ms de execução** de código **e pelo número de vezes que o código é acionado**. Pagando apenas pelo tempo de computação consumido

Performance uniforme. Com o AWS Lambda, é possível otimizar o tempo de execução do código escolhendo o tamanho de memória ideal para a função

É possível habilitar a **Simultaneidade Provisionada** para manter as funções inicializadas e prontas para responder em questão de poucos milissegundos. O processamento da função acaba sendo um pouco mais caro porém, a cobrança não é realizada pelo tempo todo da função ativa pois se não, não faria sentido

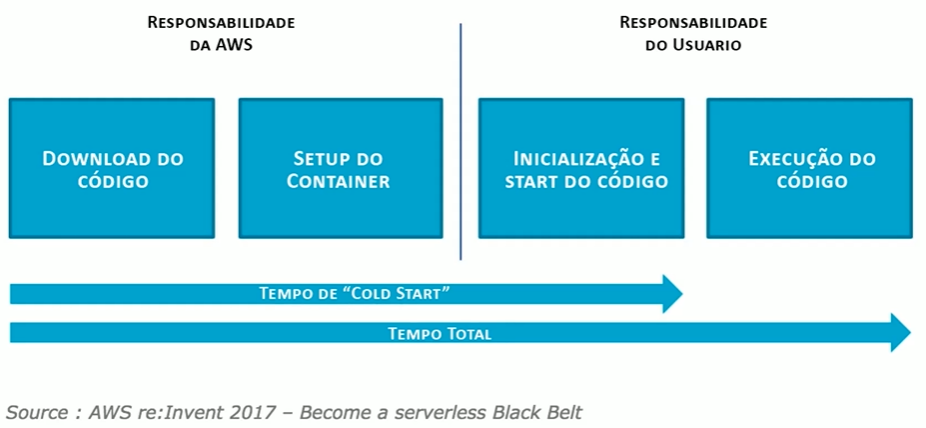


**Start do Lambda:**

Quando o Lambda é acionado, acontece o download do código, a uma altíssima velocidade, e faz o setup do container, ou seja, a AWS está montando o ambiente que o código vai ser executado

A execução do código não entra no tempo de “Cold Start”, que é o tempo de aquecer o lambda

Quanto mais conciso for o código e mais rápido ele ser executado, menor será o tempo de execução



Exemplos:





A cobrança é realizada pelo número de solicitações de suas funções e pela duração, o tempo que leva para que seu código seja executado

O Lambda conta uma solicitação cada vez que começa a executar em resposta a uma notificação de evento ou chamada de invocação, incluindo invocações de teste do console

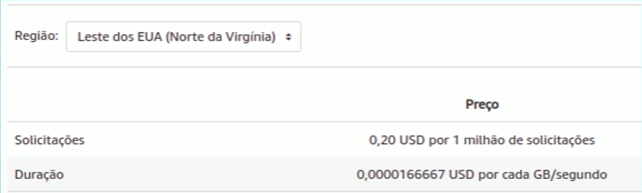
A duração é calculada a partir do momento em que seu código começa a ser executado até ele retornar ou encerrar, arredondando para os **1 ms** mais próximos

O preço depende da quantidade de memória que você alocar para sua função. A capacidade de CPU e outros recursos são alocados de forma proporcional

Um aumento no tamanho da memória aciona um aumento equivalente na CPU disponível para a função

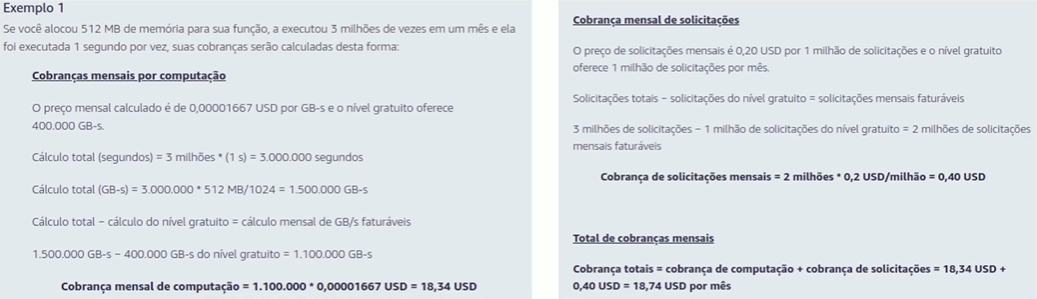
Todas as chamadas feitas para o AWS Lambda devem completar a execução dentro de **15 min**. **O timeout default é de 15 segundos**, mas é possível setar o timeout para qualquer valor entre 1 segundo e 15 min

O nível de uso gratuito da AWS Lambda inclui 1 milhão de solicitações gratuitas e 400.000 GB/segundos de tempo de computação por mês (400.000 segundos utilizando 1 GB de memória por mês)

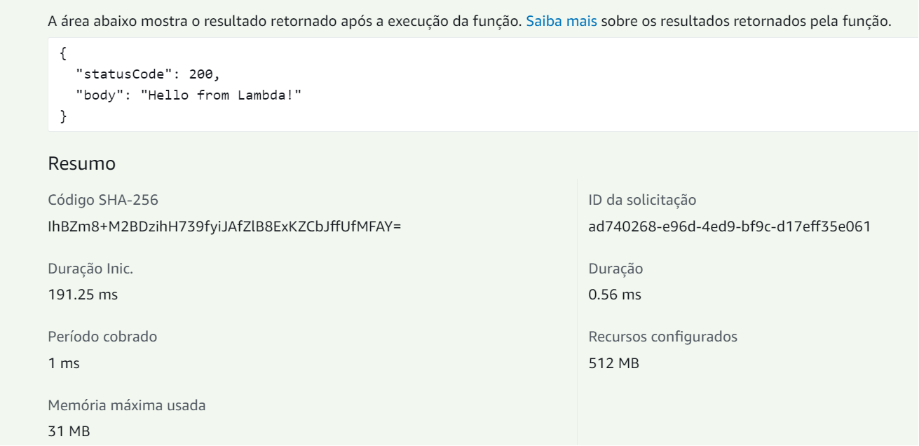




Exemplo de Cobrança:



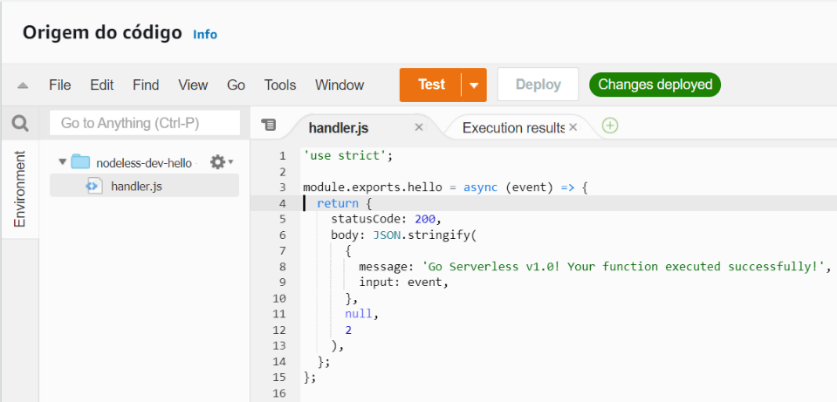
Testando uma Função Lambda:



A primeira execução do teste mostra o campo “Duração Inic.” que é apresentada toda vez que uma função é executada pela primeira vez, ou quando ela passa um tempo sem ser executada, caso uma nova requisição ocorra em seguida esse campo não será apresentado pois a função ainda está ativa em background, ficando inativa depois de alguns segundos

O campo “Memória máxima usada” de 31 MB demonstra que o recurso configurado de 512 MB pode ser baixado para 128 MB

O nome do manipulador deve ser constituído pelo nome do arquivo + o nome da function:





**Entendendo AWS API Gateway**

O AWS API Gateway é um serviço gerenciado que permite que desenvolvedores criem, publiquem, mantenham, monitorem e protejam APIs em qualquer escala com facilidade

Usando o API Gateway, é possível criar APIs RestFul e APIs WebSockets que habilitam aplicativos de comunicação bidirecionais em tempo real

O API Gateway dá suporte a cargas de trabalho conteinerizadas (ex: EC2) e sem servidor, além de aplicativos da web

Ele administra todas as tarefas envolvidas no recebimento e processamento de até centenas de milhares de chamadas de API simultâneas

Inclusive gerenciamento de tráfego, suporte de CORS, controle de autorização e acesso, com fluxo controlado, monitoramento e gerenciamento de versões de API



Além disso, com o modelo de definição de preço em camadas do API Gateway, é possível reduzir custos à medida que seu uso da API é escalado (quanto mais é usado, mais barato fica)

**Existem 2 tipos de APIs:**

1. **APIs RESTful:** Existem dois tipos de APIs RESTful, o primeiro é o proxy HTTP que é um serviço de proxy simples para repassar as chamadas aos nossos serviços. O segundo sendo a API REST, que garante funcionalidade de proxy de API e recursos de gerenciamento de API em uma única solução
2. **APIs WebSockets:** Crie aplicativos de comunicação bidirecionais em tempo real, como aplicativos de bate-papo e painéis de streaming, com APIs WebSocket

**Cache do API Gateway:** O API Gateway também oferece a opção de armazenar dados em cache mediante uma taxa por hora que varia de acordo com o tamanho de cache selecionado. Ou seja, ao invés da API Gateway executar um lambda que busca o mesmo dado no S3, estes dados ficam cacheados, reduzindo chamadas a serviços de forma desnecessária

O API Gateway não tem taxas mínimas ou custos antecipados. Você paga apenas pelas chamadas de API recebidas e pela quantidade transferida de dados de saída

Para APIs HTTP e APIs Rest, o pagamento ocorre apenas pelas chamadas de API recebidas e pela quantidade de dados transferida para fora

Para APIs do WebSocket, o pagamento ocorre somente quando as APIs estão em uso com base no número de mensagens enviadas e recebidas e pelos minutos de conexão

O nível gratuito do API Gateway inclui 1 milhão de chamadas de API HTTP, 1 milhão de chamadas de API Rest, 1 milhão de mensagens e 750.000 minutos de conexão por mês durante até 12

Com um preço de solicitações de API de apenas 0,90 USD a cada milhão de solicitações na camada mais alta, você pode reduzir seus custos conforme seu uso de API aumenta por região nas suas contas da AWS.

Link de definição de preços API Gateway: <https://aws.amazon.com/pt/api-gateway/pricing/>

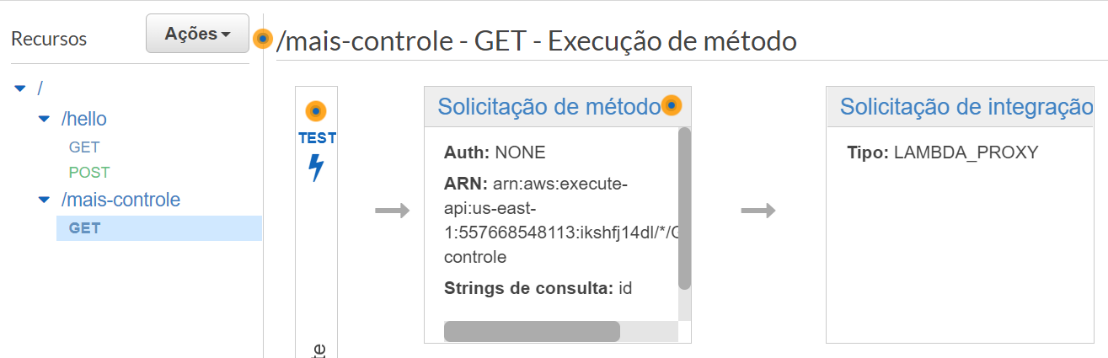
Não esquecer dos custos adicionais de outros serviços que são usados em conjunto como o AWS Lambda e AWS CloudWatch

**Compreendendo API Gateway Lambda Proxy Integration:**

<https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html>

<https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format>

Para acessar a opção Lambda Proxy Integration, clicar em **Solicitação de Integração**



A ideia é que a integração com o proxy lambda transforme o formato da solicitação, ou seja, em vez de a resposta vir no formato de String como um único body, os campos do body são retornados como uma requisição. Exemplo:

{

"isBase64Encoded": true|false,

"statusCode": httpStatusCode,

"headers": {“headerName": "headerValue", ...},

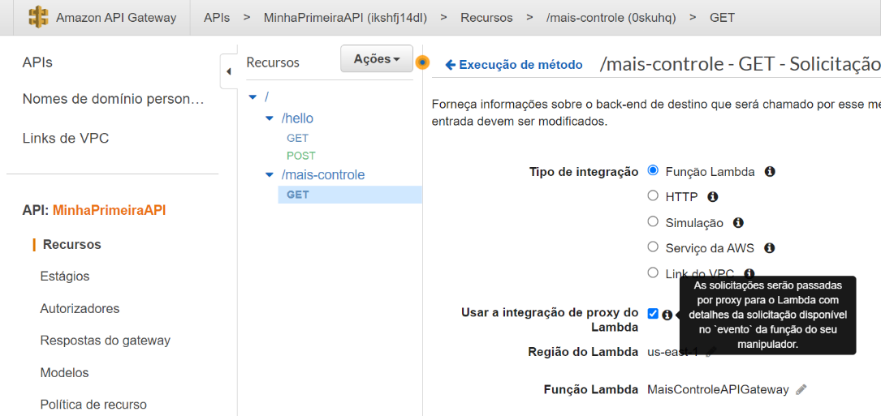
"multiValueHeaders": {“headerName": ["headerValue", "headerValue2", ...], ...},

"body": "..."

}

Dessa maneira, é transferido ao Lambda a responsabilidade de formatar a resposta da solicitação, dando mais controle sobre os endpoints criados quando integrados ao lambda

Na opção “Solicitação de Integração”, selecionar a opção de integração de proxy do lambda para capturar as informações da solicitação disponível

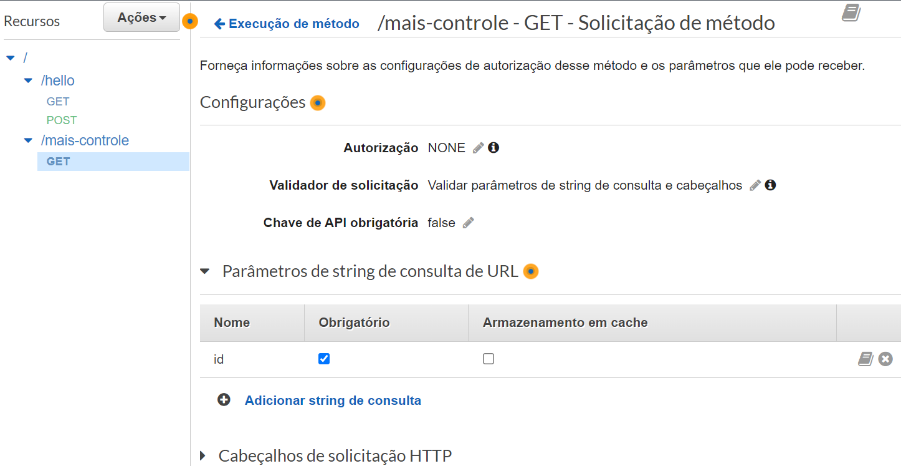


**Autorizadores**

Os autorizadores permitem que você controle o acesso às APIs usando os grupos de usuários do Amazon Cognito ou uma função Lambda.

**Solicitação de Método**

Na solicitação de método, é possível validar parâmetros de String, cabeçalhos de solicitação Http e o corpo da solicitação, por exemplo criar um parâmetro de String obrigatório, caso não seja passado na requisição, a própria API Gateway, e não o Lambda, irá retornar mensagem e status erro.



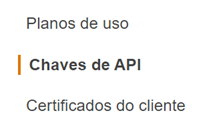
**Chave de API**

A opção **“Chave de API obrigatória”** tem uma grande importância, e serve como por exemplo para obrigar o usuário solicitante informar uma chave de API e a partir disso limitar a quantidade de requisições/segundo e a quantidade total mensal, além é claro, de garantir uma segurança adicional na chamada da API

Após mudar para true a opção “Chave de API obrigatória”, clicar na opção “Chaves de API”, após criar a chave, adicionar a um plano de uso, porém deve ir à opção “Planos de Uso”

Os planos de uso ajudam a medir o uso de APIs. Com eles, é possível aplicar um limite de cota e de controle de utilização em cada chave de API. Os limites de controle de utilização definem o número máximo de solicitações por segundo disponíveis para cada chave. Os limites de cota definem o número de solicitações que cada chave de API pode fazer em um período

Observação: não esquecer de implantar a API novamente para que as funcionalidades criadas tenham efeito e no header da request passar a chave como **x-api-key** e valor sendo a Chave de API





**Entendendo AWS SAM (Serverless Application Model)**

<https://aws.amazon.com/pt/serverless/sam/>

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-getting-started-hello-world.html>

O AWS SAM ou Serverless Application Model é um framework open-source para construir aplicações Serverless na AWS. Fornece sintaxe abreviada para expressar funções, APIs, databases e mapeamentos de origens de eventos

É concorrente do Serverless framework, é considerado o framework oficial da AWS, ambos tendem a facilitar o desenvolvimento, criar o ambiente de forma simplificada gerando as aplicações através de template e deploiando de forma muito mais fácil

Basicamente, através de template e linha de comando, conseguimos criar nossas aplicações Serverless, testar localmente simulando o ambiente da AWS e fazer deploy de forma simplificada

Totalmente gratuito, é integrado com ferramentas de desenvolvimento (ex: plugin no vscode AWS Toolkit), podendo executar o ambiente em containers do Docker, baixando as imagens direto da AWS. O único trabalho então é instala-lo, executável pode ser baixado pelo link acima, após instalar o **aws\_sam\_cli**, executar o comando **sam --version**

#Step 1 - Download a sample application

**sam init**

#Step 2 - Build your application

**cd sam-app**

**sam build**

#Step 3 - Deploy your application

**sam deploy –guided**

Invocando função localmente:

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-using-invoke.html>

# Invoking function with event file

$ sam local invoke "HelloWorldFunction" -e events/event.json

# Invoking api gateway local

$ sam local start-api

**Entendendo AWS IAM**

O AWS IAM – Identity and Access Management permite gerenciar com segurança o acesso aos serviços e recursos da AWS. Usando o IAM, é possível criar e gerenciar usuários e grupos da AWS e usar permissões para conceder e negar acesso a recursos da AWS, sendo totalmente gratuito.

Controle de **acesso minucioso** para seus recursos da AWS: O IAM permite que os usuários controlem o acesso às APIs de serviço e a recursos específicos da AWS. O IAM também permite que seja adicionado **condições específicas**, como hora certa para controlar como um usuário pode usar a AWS, seu endereço IP de origem, se estão usando SSL ou se fizeram a autenticação com um dispositivo de autenticação multifator.

**Identidades do IAM**

Usuário Raiz: Ao criar uma conta da AWS, você começa com uma única identidade de login que tem acesso total a todos os serviços e recursos da AWS na conta. Essa identidade é chamada de usuário raiz da conta e é acessada pelo login com o endereço de e-mail e a senha que você usou para criar a conta. A própria AWS recomenda não usar o usuário raiz para as tarefas diárias nem mesmo para as tarefas administrativas

**Políticas do IAM**

Uma política é um objeto na AWS que, quando associado a uma identidade ou a um recurso, define suas permissões

**Função do IAM**

Uma função do IAM é uma identidade do IAM que se pode criar na conta com permissões especificas. Você pode usar funções para delegar acesso a usuários, aplicativos ou serviços que normalmente não tem acesso aos seus recursos da AWS. Em um breve resumo, uma Função é um conjunto das políticas, ou seja, várias políticas formam uma Função

**Usuário do IAM**

O usuário do IAM representa a pessoa ou o serviço que usa o usuário do IAM para interagir com a AWS. Seu uso principal é oferecer às pessoas a capacidade de entrar no Console de gerenciamento da AWS ou solicitações programáticas aos servidores da AWS, usando a API ou a CLI

Concede-se a ele permissões tornando-o um membro de um grupo com **políticas de permissão** apropriadas anexadas (recomendadas) ou anexando diretamente políticas ao usuário

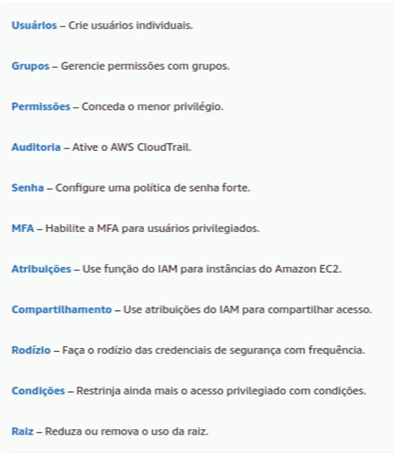
**Grupos do IAM**

Um grupo do IAM é um conjunto de usuários do IAM. Pode-se usar grupos para especificar permissões para um conjunto de usuários, o que pode facilitar o gerenciamento dessas permissões para esses usuários. Lembrando que é possível setar as permissões para cada usuário individualmente

**Credenciais Temporárias**

O IAM também permite que se conceda aos usuários credencias temporárias de segurança com uma expiração definida para dar acesso aos seus recursos da AWS

**Melhores Práticas do IAM**



**Quando criar um usuário do IAM?**

Quando uma pessoa precisa de acesso para acessar os recursos da AWS

**Quando criar uma função do IAM?**

Quando um serviço, aplicativo ou recurso precisa acessar os recursos da AWS (inclusive outros serviços da AWS)

Ou quando usuários da empresa estão autenticados na rede corporativa e desejam utilizar a AWS sem a necessidade de realizar login novamente

**Quando criar uma política personalizada?**

Quando se não há uma política pronta com os acessos e recursos desejados para atribuir a um ou mais usuário, grupo ou função da conta

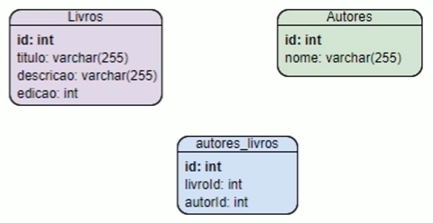
**Entendendo AWS DynamoDB**

**NoSQL x SQL**

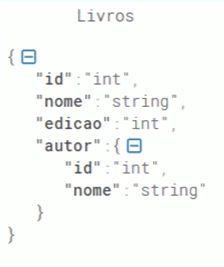
NoSQL é um tipo de banco de dados que possui esquemas flexíveis para a criação de aplicativos modernos. Eles são amplamente reconhecidos por sua facilidade de desenvolvimento, funcionalidade e performance em escala

Eles são otimizados especificamente para aplicativos que exigem modelos de grande volume de dados, baixa latência e flexibilidade. Esses requisitos são atendidos mediante o relaxamento de algumas restrições de consistência de dados dos outros bancos

Banco de dados SQL (Relacional) o modelo é projetado para permitir que o banco de dados imponha a integridade referencial entre as tabelas no banco de dados, normalizadas para reduzir a redundância e geralmente otimizadas para armazenamento



Banco de dados NoSQL (Não Relacional), neste modelo os dados são otimizados para desenvolvimento intuitivo e escalabilidade horizontal



Os bancos de dados NoSQL são ideias para muitos aplicativos modernos, como dispositivos móveis, Web e jogos que exigem banco de dados flexíveis, escaláveis, de alta performance e altamente funcionais para proporcionar ótimas experiências aos usuários

Fornecem esquemas flexíveis que permitem um desenvolvimento mais rápido e iterativo. O modelo de dados flexível torna os bancos de dados NoSQL ideias para dados semiestruturados e não estruturados

Escalabilidade: São projetados para serem escalados horizontalmente usando clusters distribuídos de hardware, em vez de escalá-los verticalmente adicionando servidores caros e robustos. Alguns provedores de nuvem lidam com essas operações nos bastidores como um serviço totalmente gerenciado

Alta Performance: Otimizados para modelos de dados específicos e padrões de acesso que permitem maior performance do que quando se tenta realizar uma funcionalidade semelhante com bancos de dados relacionais

Altamente Funcionais: Fornecem APIs e tipos de dados altamente funcionais criados especificamente para cada um de seus respectivos modelos de dados

Eles são altamente particionáveis e permitem escalabilidade horizontal em escalas que outros tipos de banco de dados não conseguem alcançar

O Amazon DynamoDB foi projetado para proporcionar uma latência consistente de um dígito de milissegundo para qualquer escala de cargas de trabalho

**AWS DynamoDB**

Além de ser NoSQL e os dados serem acessados através de **chave/valor**, é um banco de dados durável, totalmente gerenciado com segurança, backup e restauração integrados, além de armazenamento em cache

O DynamoDB pode processar mais de 10 trilhões de solicitações por dia e comportar picos de mais de 20 milhões de solicitações por segundo. Empresas como Lyft, Airbnb, Samsung, Toyota, Netflix e muitas outras o utilizam

Performance em grande escala: Ele garante tempos de resposta consistente abaixo de **10 milissegundos**. As tabelas globais do DynamoDB replicam seus dados em várias regiões da AWS para oferecer acesso rápido e local a dados para seus aplicativos distribuídos globalmente

Ainda mais performance: Para casos de uso que exigem acesso ainda mais rápido com latência de microssegundos, o **DynamoDB Accelerator (DAX)** oferece um cache de memória totalmente gerenciado

Não há servidores para gerenciar. O DynamoDB é **sem servidor**, e é capas de expandir e reduzir tabelas automaticamente para ajustar de acordo com a capacidade e manter o desempenho. A disponibilidade e a tolerância a falhas são incorporadas, eliminando a necessidade de projetar esses recursos em seus aplicativos

Modos de Capacidade: O DynamoDB oferece modos de **capacidade provisionada** e **sob demanda** para que se possa otimizar custos especificando a capacidade por carga de trabalho ou pagamento somente pelos recursos que consumir, lembrando que a capacidade sob demanda é um pouco mais cara que a capacidade provisionada

Pronto para uso empresarial: O DynamoDB **criptografa todos os dados por padrão** e oferece controle refinado de acesso e identidade em todas as suas tabelas. É possível criar **backups completos** de centenas de terabytes de dados instantaneamente, sem impacto no desempenho de suas tabelas, e recuperar qualquer momento dos 35 dias anteriores sem tempo de inatividade

**Mais Sobre AWS DynamoDB**

Unidade de Capacidade de Leitura (RCU): Unidade responsável por ler registros do banco de dados

Unidade de Capacidade de Gravação (WCU): Unidade responsável por gravar registros do banco de dados

Cada **chamada de API** para ler dados de uma tabela é uma solicitação de leitura

As unidades de leitura do DynamoDB podem ser **eventualmente consistentes**, **fortemente consistentes** ou **transacionais**

Eventualmente Consistente: Quando se lê dados de uma tabela do DynamoDB, a resposta pode não refletir os resultados de uma operação de gravação recentemente concluída. A resposta pode incluir alguns dados obsoletos. Se você repetir sua solicitação de leitura após um curto período, a resposta deverá retornar os dados mais recentes

Fortemente Consistente: Quando é solicitada uma leitura fortemente consistente, o DynamoDB retorna uma resposta com os dados mais atualizados, refletindo as atualizações de todas as operações de gravação anteriores que foram bem-sucedidas. No entanto, essa consistência traz algumas desvantagens, tais como:

Uma leitura fortemente consistente **poderá não estar disponíve**l se houver um **atraso ou uma interrupção de rede**. Retornando um erro de servidor (HTTP 500). Podem também ter **maior latência** do que as eventualmente consistente. E por fim, elas usam **maior capacidade de taxa de transferência** do que as leituras eventualmente consistentes e mais RCU

**Unidade de Capacidade de Leitura (RCU)**

Para itens com tamanho de até 4KB, uma RCU pode executar uma solicitação de leitura fortemente consistente por segundo, itens maiores que 4KB exigem mais RCUs. Para itens com tamanho de até 4KB, uma RCU pode executar duas solicitações de leitura eventualmente consistente por segundo. As solicitações de leitura transacional exigem duas RCUs para executar uma leitura por segundo para itens de até 4KB. É possível escalar um número específico ou de acordo com a demanda

**Unidade de Capacidade de Gravação (WCU)**

Cada chamada de API para gravar dados de uma tabela é uma solicitação de gravação. Uma unidade de solicitação de gravação padrão pode gravar um item de até 1KB/segundo. Para itens maiores que 1KB, são necessárias unidades de solicitação de gravação adicionais. Uma gravação transacional exige duas unidades de solicitação de gravação. É possível escalar um número específico ou de acordo com a demanda

**Solicitação de Leitura/Gravação Transacionais**

No DynamoDB, uma leitura ou gravação transacional é diferente de uma leitura ou gravação padrão porque garante que **todas as operações de um único conjunto** de transações executem com sucesso ou falhem em conjunto

**Tabelas Globais e rWCU**

As tabelas globais replicam automaticamente as tabelas do DynamoDB nas regiões da AWS à sua escolha. Cada gravação ocorre na **região local** e nas **regiões replicadas**. Unidade de Capacidade de Gravação replicadas (**rWCU**) é a unidade responsável por gravar registros do banco de dados nas tabelas replicadas, lembrando que a replicação ocorre quase que em tempo real, ou seja, gravou um dado localmente o registro é automaticamente replicado

**Streams do AWS DynamoDB**

Um stream do DynamoDB é um fluxo ordenado de informações sobre alterações em itens de uma tabela do DynamoDB. Quando é ativado um stream em uma tabela, o DynamoDB captura informações sobre todas as modificações em itens de dados na tabela e pode por exemplo invocar um lambda, ou seja, as inserções, alterações e remoções na tabela são interpretados como eventos que podem disparar um lambda

**Definição de Preço para Capacidade Provisionada**

Com o modo de capacidade provisionada, você especifica o número de leituras e gravações de dados por segundo exigidas pelo aplicativo. Você pode usar o **Auto Scaling** para ajustar automaticamente a capacidade da tabela de acordo com a taxa de utilização do aplicativo e reduzir os custos. Com o Auto Scaling, é possível configurar o mínimo e máximo de unidades de leitura e gravação e o DynamoDB scala conforme a necessidade e quantidade de requisições

**Definição de Preço para Capacidade Sob Demanda**

Com o modo de capacidade sob demanda, você paga pelas leituras e gravações de dados efetuadas pelo aplicativo nas tabelas. Não é preciso especificar a **quantidade de throughput** (taxa de transferência) de leitura e gravação que espera que o aplicativo execute, pois o DynamoDB acomoda instantaneamente o aumento e a redução das cargas de trabalho, lembrando que o preço pode ficar mais elevado de acordo com a necessidade

**Integrando DynamoDB e Lambda**

<https://docs.aws.amazon.com/pt_br/sdk-for-javascript/v2/developer-guide/dynamodb-example-table-read-write.html>

<https://docs.aws.amazon.com/pt_br/sdk-for-javascript/v2/developer-guide/dynamodb-example-document-client.html>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.NodeJs.04.html>

<https://docs.aws.amazon.com/lambda/latest/dg/kinesis-tutorial-spec.html>

**Entendendo AWS SQS**

O Amazon Simple Queue Service (SQS) é um serviço de filas de mensagens gerenciado que permite o desacoplamento e a escalabilidade de microsserviços, sistemas distribuídos e aplicativos sem servidor

O SQS elimina a complexidade e a sobrecarga associadas ao gerenciamento e à operação de middleware orientado a mensagens

Use o SQS para enviar, armazenar e receber mensagens entre componentes de software em qualquer volume, sem perder mensagens ou precisar que outros serviços estejam disponíveis. O SQS oferece dois tipos de filas de mensagens, que são elas:

**Fila Padrão**

Taxa de transferência ilimitada: as filas padrão comportam um número quase ilimitado de transações por segundo (TPS) por ação de API

Entrega pelo menos uma vez: uma mensagem é entregue pelo menos uma vez, mas, às vezes, mais de uma cópia da mensagem é entregue

Melhor ordenação possível: às vezes, as mensagens podem ser entregues em uma ordem diferente da qual elas foram enviadas

**Fila Padrão – Exemplo de aplicação**

Desacoplar solicitações dinâmicas do usuário de trabalhos intensos em segundo plano: permite aos usuários enviar mídia, além de redimensioná-la e codifica-la

Alocar tarefas para nós com diversos operadores: processa uma alta quantidade de solicitações de validação de cartão de crédito

Mensagens em lote para processamento futuro: agende diversas entradas para adicioná-las ao banco de dados

**Fila FIFO (First In – First Out)**

Entrega FIFO: a ordem em que as mensagens são enviadas e recebidas é preservada com rigor, ou seja, a primeira a entrar será a primeira a sair

Processamento exatamente uma vez: uma mensagem é entregue uma vez e permanece disponível até que um consumidor a processe e exclua. Duplicatas não são inseridas na fila

Alta taxa de transferência: por padrão, as filas do tipo FIFO comportam até 300 mensagens por segundo (300 operações de envio, recebimento ou exclusão por segundo)

Ao agrupar em lote 10 mensagens por operação (no máximo), as filas FIFO podem dar suporte a até 3000 mensagens por segundo.

**Fila FIFO – Exemplo de aplicação**

Verifique se os comandos inseridos por usuários são executados na ordem correta

Visualize o preço do produto correto ao enviar modificações de preço na ordem correta

Garanta que usuários farão ações que foram cumpridos os pré-requisitos

**Pontos importantes**

O número de filas que podemos criar é ilimitado

As mensagens podem ter até 256 KB, porém é cobrado 1 solicitação a cada 64 KB

É possível agrupar mensagens em lotes de até 10 mensagens ou 256 KB (cada lote custa o mesmo que mensagens únicas).

Tempo limite de visibilidade: Enquanto a mensagem está sendo processada ela fica como “bloqueada” e não será processada por outro consumer, quando o tempo limite de visibilidade se esgota, a mensagem volta para a fila. O tempo limite é importante para caso caia o serviço consumidor, a mensagem não ficará presa e sendo processada por uma grande quantidade de tempo

Atraso de entrega: é o tempo que a mensagem vai levar para ficar disponível na fila para outros serviços consumirem

Tempo de espera do recebimento da mensagem: é o tempo que o consumer tem para ficar ouvindo a fila, ou seja, caso a mensagem já esteja disponível automaticamente ela será processada, caso não o consumir tem um tempo de espera para o recebimento da mensagem

Período de retenção da mensagem: retenha as mensagens nas filas por até 14 dias

Criptografia no lado do servidor (SSE)

Dead Letter Queues (DLQ), suponha que uma mensagem de problema e volte pra fila, para que ela não fique em loop infinito sendo processada e voltando pra fila as filas DLQs, ou filas de mensagens mortas, podem ser configuradas para que quando ocorrer um número máximo de processamento para a mesma mensagem, as DLQs possam alertar o usuário do problema

**Integrado com outros serviços da AWS**

As filas de mensagens do Amazon SQS podem ser usadas com outros serviços da AWS, como Redshift, DynamoDB, RDS, EC2, ECS, Lambda e S3 para aumentar a escalabilidade e a confiabilidade de aplicativos distribuídos

**Aplicação de integração comuns**

Filas de trabalho: desconecte componentes de um aplicativo distribuído que podem não processar o mesmo volume de trabalho simultaneamente

Operações de buffer e lote: acrescente escalabilidade e confiabilidade à sua arquitetura e suavize picos de volume temporários sem perder mensagens ou aumentar a latência

Descarregamento de solicitações: remova operações lentas de caminhos de solicitação interativos enfileirando a solicitação

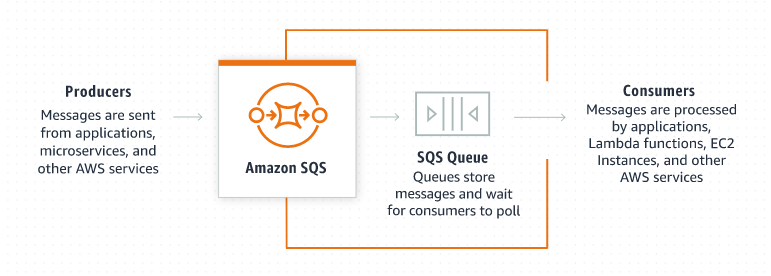
Prioridade: use filas separadas para disponibilizar a priorização do trabalho

Escalabilidade: como as filas de mensagens dissociam seus processos, torna-se fácil expandir a taxa de envio e recebimento de mensagens (basta adicionar outro processo)

**Definição de Preço**

O primeiro 1 milhão de solicitações mensais é gratuito, acima disso a fila padrão e a fila FIFO tem preços diferentes, sendo a fila FIFO um pouco mais cara. Lembrando que cada ação do Amazon SQS conta como uma solicitação, ou seja, cada bloco de 64KB de carga útil é cobrado como 1 solicitação

A Amazon SQS também é cobrada pela transferência de dados, sendo as transferências de dados para DENTRO do SQS totalmente gratuitos e as transferências para FORA tendo diferentes faixas de preço por volume em GB/mês transferido



Links Uteis

<https://docs.aws.amazon.com/pt_br/sdk-for-javascript/v2/developer-guide/sqs-examples-send-receive-messages.html>

**Entendendo AWS SNS**

O Amazon Simple Notification Service (SNS) é um serviço de mensagens totalmente gerenciado para comunicação de sistema para sistema e de aplicativo para pessoa (A2P)

Ele permite que você se comunique entre sistemas por meio de padrões de publicação/assinatura (pub/sub)

A principal diferença do SNS para o SQS é que no SQS existe uma fila onde apenas uma vez a mensagem será processada por um consumer e já no SNS todos os consumers que estiverem inscritos em um ou mais tópicos receberão a mensagem (um emissor e um consumer: SQS, vários emissores e vários consumers: SNS)

Ele permite mensagens entre aplicativos de microsserviços dissociados ou para se comunicar diretamente com os usuários via SMS, push móvel e e-mail

A funcionalidade de pub/sub de sistema para sistema fornece tópicos para mensagens de alto rendimento, baseadas em push e de muitos-para-muitos

Usando tópicos do Amazon SNS, seus sistemas editores podem espalhar mensagens para um grande número de sistemas de assinantes ou endpoints de clientes, incluindo filas do Amazon SQS, funções do AWS Lambda e HTTP/S, para processamento paralelo

**Pontos Importantes**

É possível filtrar mensagens

As mensagens são replicadas em múltiplas AZs para garantir durabilidade

Caso o subscriber não esteja disponível, é possível enviar a/as mensagens para DLQ

Suporta criptografia

O SNS envia notificações mobile, além do SMS e E-mail

**Definição de Preço**

O primeiro 1 milhão de publicações mensais é gratuito. Tamanho máximo de 256KB para mensagens publicadas. Cada bloco de 64KB de dados publicados é cobrado como uma solicitação. Não só o envio, mas também a entrega é cobrada.

Com o Amazon SNS, não há taxa mínima e você paga somente por aquilo que usar. Os usuários pagam 0,50 USD por 1 milhão de solicitações do Amazon SNS, 0,06 USD por 100.000 entregas de notificação via HTTP e 2,00 USD por 100.000 entregas de notificação via e-mail.

A Amazon SNS também é cobrada pela transferência de dados, sendo as transferências de dados para DENTRO do SNS totalmente gratuitos e as transferências para FORA tendo diferentes faixas de preço por volume em GB/mês transferido

**Aplicação para aplicação (A2A)**

****

**Aplicação para pessoa (A2P)**

****

**Links Úteis**

<https://docs.aws.amazon.com/pt_br/lambda/latest/dg/with-sns-example.html>

**Entendendo AWS S3**

O Amazon Simple Storage Service (Amazon S3) é um serviço de armazenamento de objetos que oferece escalabilidade líder do setor, disponibilidade de dados, segurança e performance

Isso significa que clientes de todos os tamanhos e setores podem usá-lo para armazenar qualquer volume de dados em uma grande variedade de casos de uso

Tais como sites, aplicações para dispositivos moveis, backup e restauração, arquivamento, aplicações empresariais, dispositivos IoT e análises de big data

O Amazon S3 foi projetado para 99,999999999% (11 9s) de durabilidade e armazena dados para milhões de aplicativos para empresas de todo o mundo

O Amazon S3 é altamente redundante pois os arquivos são copiados para diversas AZs justamente para garantir a segurança necessária de que os arquivos estarão disponíveis quando for preciso

Projeta os dados contra acesso não autorizado com recursos de criptografia e ferramentas de gerenciamento de acesso

O S3 mantém diversos padrões de conformidade de segurança (PCI-DSS, HIPAA/HITECH, FedRAMP, EU Data Protection Directive e FISMA) para ajudar a cumprir requisitos normativos, como por exemplo uma aplicação financeira que tem compliances de segurança a seguir

Execute análises de big data nos objetos do S3 (e em outros conjuntos de dados na AWS) com os serviços de consulta local

Use o **Amazon Athena** para consultar dados do S3 com expressões SQL padrão e o **Amazon Redshift Spectrum** para analisar dados armazenados nos data warehouses e recursos do S3 na AWS

**Categorias de Armazenamento do S3**

Como funciona a configuração de armazenamento do S3

As classes de armazenamento S3 podem ser configuradas no nível do objeto, em um único Bucket pode conter objetos armazenados no S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA e no S3 One Zone-IA

Também podemos usar as políticas de ciclo de vida do S3 para migrar automaticamente objetos entre classes de armazenamento sem nenhuma alteração nos aplicativos

Amazon S3 Standard: Oferece um armazenamento de objetos com altos níveis de resiliência, disponibilidade e performance **para dados acessados com frequência**. Como fornece baixa latência e alto throughput (taxa de transferência), o S3 Standard é adequado para uma grande variedade de casos de uso, como aplicativos na nuvem, sites dinâmicos, distribuição de conteúdo, aplicativos moveis e de jogos analíticos de big data

Amazon S3 Standard-Infrequent Access (S3 Starndard-IA): É indicado para dados acessados com **menos frequência**, mas que exige acesso rápido quando necessários. Oferece altos níveis de resiliência e throughput e a baixa latência do S3 Standard. A combinação de baixo custo e alta performance a tornaram ideal para armazenamento de longa duração, backups e datastores para arquivos de recuperação de desastres

Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA): Ao contrário de outras classes de armazenamento do S3, que armazena dados em no mínimo três Zonas de Disponibilidade (AZs), armazena dados em uma única AZ. É uma opção de menor custo para dados acessados com pouca frequência, mas não precisam de disponibilidade e da resiliência como no S3 Standard ou S3 Standard-IA

Amazon S3 Intelligent-Tiering (S3 Intelligent-Tiering): O S3 Inteligente-Hierarquização foi projetado para otimizar os custos movendo automaticamente os dados para o nível de acesso mais econômico, sem impacto na performance ou sobrecarga operacional. Ela funciona a partir do armazenamento de objetos em dois níveis de acesso: um nível é otimizado para acesso frequente e o outro nível de custo mais baixo, otimizado para acesso infrequente

**Amazon S3 Glacier (Arquivamento)**

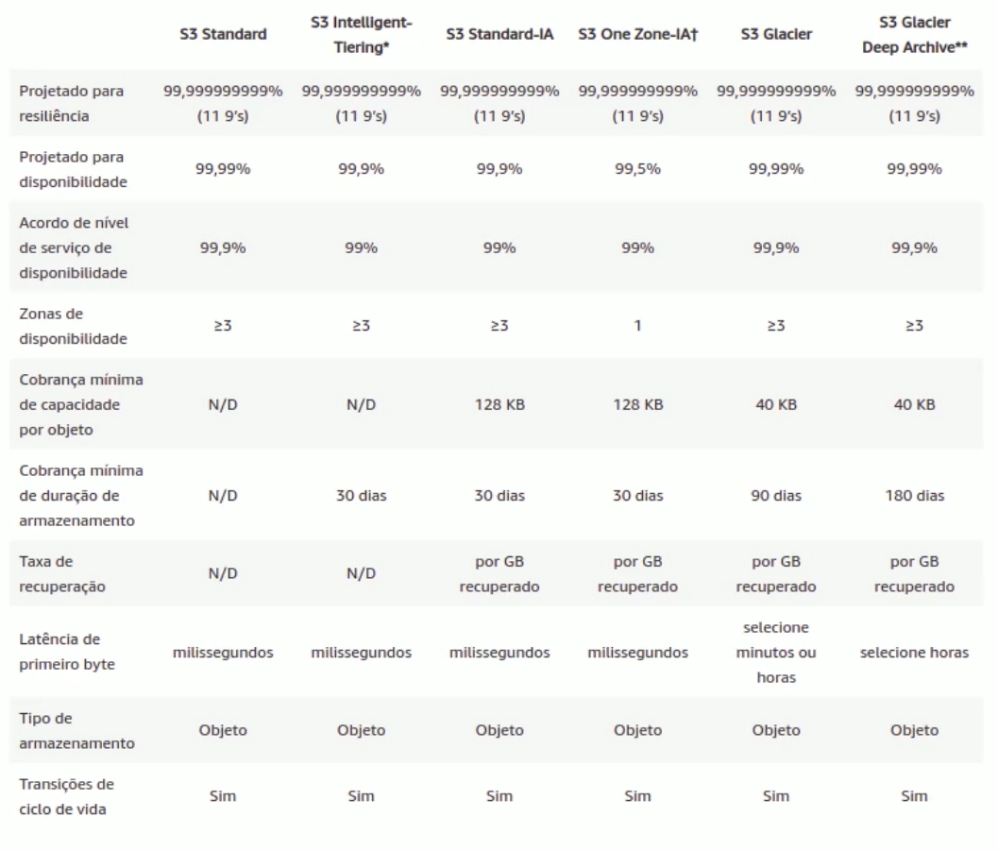
É uma classe de armazenamento segura, durável e de baixo custo para arquivamento de dados. Pode-se armazenar com confiabilidade qualquer volume de dados a um custo competitivo ou inferior ao custo de soluções no local. Para manter os custos baixos, mas com condições de suprir necessidades variáveis, o S3 Glacier disponibiliza três opções de recuperação, que podem levar de alguns minutos a várias horas

Amazon S3 Glacier Deep Archive (Arquivamento): É a classe de armazenamento mais barata do Amazon S3 e oferece suporte à retenção e preservação digitais de longo prazo para dados que podem ser acessados **uma ou duas vezes** por ano. Essa classe é projetada para clientes que mantêm conjuntos de dados por 7 a 10 anos ou mais para cumprir requisitos de conformidade normativa, e podem ser restaurados em até 12 horas

**Resumo das Classes de Armazenamento:**

****

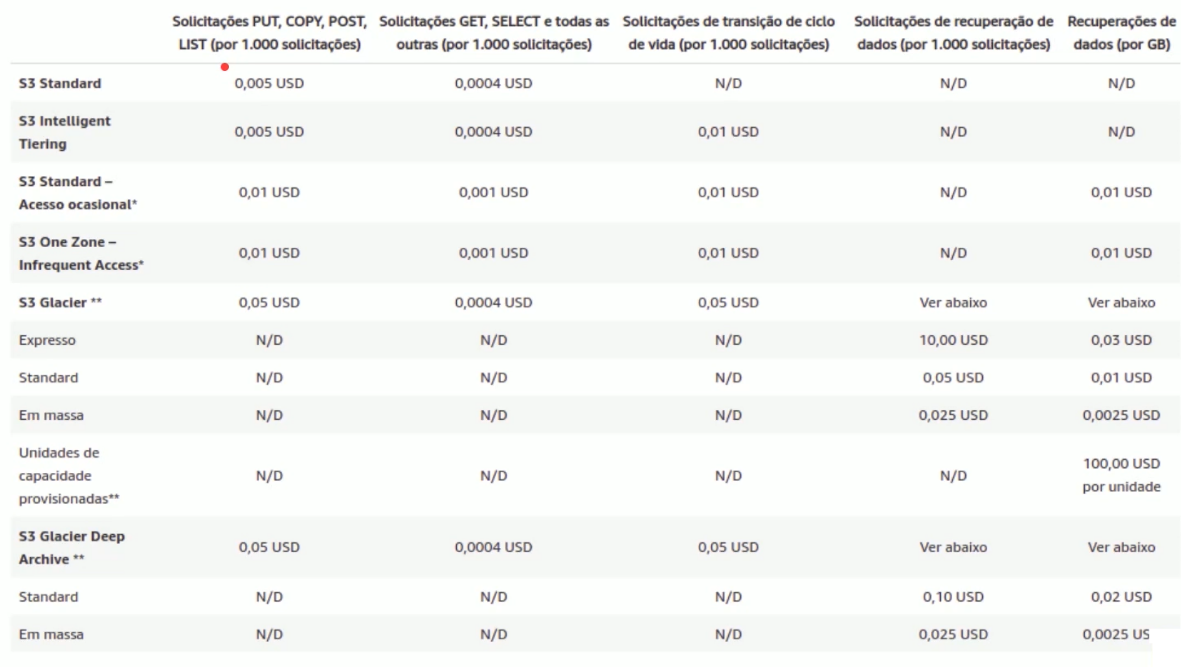
**Classes de Armazenamento e Suas Características:**



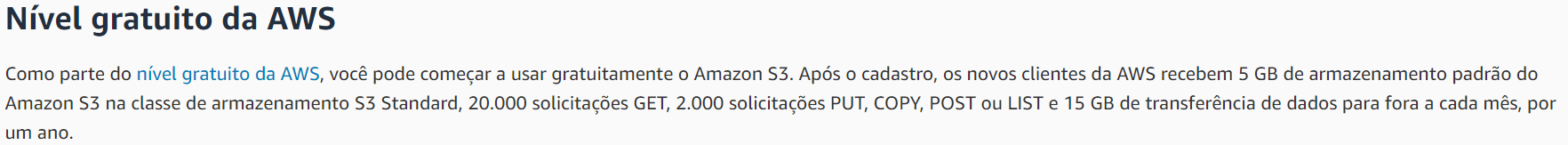
**Definição de Preço de Armazenamento por GB:**



**Definição de Preço de Solicitação e Recuperação de Dados:**



**Observações:** As categorias **Infrequent Access (Acesso Ocasional)** são mais baratas no armazenamento porem nas solicitações de recuperação de dados são mais caros. Reparar também que as solicitações do tipo POST são mais caras do que as solicitações do tipo GET



**Outros Pontos Importantes:**

Registros de Log de Acesso ao Servidor: O registro de acesso ao servidor fornece registros detalhados para as solicitações feitas a um intervalo. Os logs de acesso ao servidor são úteis para muitos aplicativos. Por exemplo, as informações do log de acesso podem ser úteis em auditorias de segurança e acesso. Também pode ajudá-lo a aprender sobre sua base de clientes e entender sua fatura do Amazon S3. Quando os logs são habilitados o console do S3 atualiza automaticamente a lista de controle de acesso (ACL) do bucket. É recomendado que ao se habilitar o registro de logs, um outro bucket que não o atual seja selecionado como destino dos logs, diminuindo o faturamento de armazenamento e melhorando a localização de logs

Eventos de Dados do AWS CloudTrail: O Amazon S3 é integrado ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou serviço AWS no Amazon S3. O CloudTrail captura um subconjunto de chamadas de API para Amazon S3 como eventos, incluindo chamadas do console do Amazon S3 e chamadas de código para as APIs do Amazon S3. Se você criar uma trilha, poderá habilitar a entrega contínua de eventos CloudTrail para um bucket do Amazon S3, incluindo eventos para Amazon S3. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no console do CloudTrail em Histórico de eventos. Usando as informações coletadas pelo CloudTrail, você pode determinar a solicitação feita ao Amazon S3, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando foi feita e detalhes adicionais. Para saber mais sobre o CloudTrail, incluindo como configurá-lo e habilitá-lo, consulte o AWS CloudTrail User Guide.

Amazon S3 Transfer Acceleration: é um recurso de nível de bucket que permite transferências rápidas, fáceis e seguras de arquivos em longas distâncias entre seu cliente e um bucket S3. O Transfer Acceleration tira proveito dos pontos de presença globalmente distribuídos no Amazon CloudFront. Conforme os dados chegam em um ponto de presença, eles são roteados para o Amazon S3 por um caminho de rede otimizado.

Pagamento pelo Solicitante: Em geral, os proprietários de bucket pagam por todos os custos de armazenamento e transferência de dados do Amazon S3 que estão associados a seu bucket. No entanto, você pode configurar um bucket para ser um bucket do Requester Pays. Com os buckets de Requester Pays, o solicitante, em vez do proprietário do bucket, paga o custo da solicitação e o download dos dados do bucket. O proprietário do intervalo sempre paga o custo de armazenamento de dados.

Hospedagem de Site Estático: Pode-se usar o Amazon S3 para hospedar um site estático. Em um site estático, as páginas da web individuais incluem conteúdo estático. Eles também podem conter scripts do lado do cliente. Por outro lado, um site dinâmico depende do processamento do lado do servidor, incluindo scripts do lado do servidor, como PHP, JSP ou ASP.NET. O Amazon S3 não oferece suporte a scripts do lado do servidor, mas a AWS tem outros recursos para hospedar sites dinâmicos. Para saber mais sobre hospedagem de sites na AWS, consulte Web Hosting

Lista de Controle de Acesso (ACL): As listas de controle de acesso (ACLs) do Amazon S3 permitem que você gerencie o acesso a **buckets e objetos**. Cada bloco e objeto tem uma ACL anexada a ele como um sub-recurso. Ele define quais contas ou grupos da AWS têm acesso concedido e o tipo de acesso. Quando uma solicitação é recebida em um recurso, o Amazon S3 verifica a ACL correspondente para verificar se o solicitante tem as permissões de acesso necessárias. Quando você cria um bucket ou um objeto, o Amazon S3 cria uma ACL padrão que concede ao proprietário do recurso controle total sobre o recurso

Compartilhamento de Recursos de Origem Cruzada (CORS): O compartilhamento de recursos de origem cruzada (CORS) define uma maneira para os aplicativos da Web cliente que são carregados em um domínio interagirem com os recursos em um domínio diferente. Com o suporte do CORS, você pode construir aplicativos da web ricos no lado do cliente com o Amazon S3 e permitir seletivamente o acesso de origem cruzada aos recursos do Amazon S3. Esta seção mostra como habilitar o CORS usando o console do Amazon S3, a API REST do Amazon S3 e os SDKs da AWS. Para configurar seu bucket para permitir solicitações de origem cruzada, você adiciona uma configuração CORS ao bucket. Uma configuração CORS é um documento que define regras que identificam as origens que você permitirá que você acesse seu bucket, as operações (métodos HTTP) com suporte para cada origem e outras informações específicas da operação. No console S3, a configuração do CORS deve ser um documento JSON.

Regras de Ciclo de Vida: Para gerenciar seus objetos de forma que sejam armazenados de maneira econômica durante todo o seu ciclo de vida, configure o Amazon S3 Lifecycle. Uma configuração de ciclo de vida do S3 é um conjunto de regras que definem ações que o Amazon S3 aplica a um grupo de objetos. Existem dois tipos de ações: **Ações de transição** - defina quando os objetos fazem a transição para outro usando classes de armazenamento do Amazon S3. Por exemplo, você pode optar por fazer a transição de objetos para a classe de armazenamento S3 Standard-IA 30 dias depois de criá-los ou arquivar objetos para a classe de armazenamento S3 Glacier um ano após criá-los. **Ações de expiração** - defina quando os objetos expiram. O Amazon S3 exclui objetos expirados em seu nome.

Criar Regra de Replicação: A replicação permite a cópia automática e assíncrona de objetos em buckets do Amazon S3. Buckets configurados para replicação de objeto podem pertencer à mesma conta da AWS ou a contas diferentes. Os objetos podem ser replicados para um único intervalo de destino ou vários intervalos de destino. Os buckets de destino podem estar em diferentes regiões da AWS ou na mesma região do bucket de origem. **Regras:** A replicação requer que o versionamento seja habilitado para o bucket de origem. É possível criar várias replicas para buckets de destino distintos para um único bucket de origem. Uma função do IAM fará a gestão das políticas. É possível alterar a classe de armazenamento do bucket de destino dos objetos replicados

Versionamento do Bucket: O controle de versão no Amazon S3 é um meio de manter várias variantes de um objeto no mesmo intervalo. Você pode usar o recurso S3 Versioning para preservar, recuperar e restaurar todas as versões de todos os objetos armazenados em seus depósitos. Com o controle de versão, você pode se recuperar mais facilmente de ações não intencionais do usuário e de falhas do aplicativo. Depois que o controle de versão é habilitado para um bucket, se o Amazon S3 receber várias solicitações de gravação para o mesmo objeto simultaneamente, ele armazena todos esses objetos. Ao trabalhar com o controle de versão S3 em buckets do Amazon S3, você pode opcionalmente adicionar outra camada de segurança configurando um bucket para habilitar a exclusão de MFA (autenticação multifator). Ao fazer isso, o proprietário do intervalo deve incluir duas formas de autenticação em qualquer solicitação para excluir uma versão ou alterar o estado de controle de versão do intervalo.

Criptografia Padrão: Com a criptografia padrão do Amazon S3, você pode definir o comportamento de criptografia padrão para um bucket do S3 para que todos os novos objetos sejam criptografados quando são armazenados no bucket. Os objetos são criptografados usando criptografia do lado do servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3) ou chaves AWS KMS armazenadas no AWS Key Management Service (AWS KMS) (SSE-KMS). Ao configurar seu bucket para usar criptografia padrão com SSE-KMS, você também pode habilitar S3 Bucket Keys para diminuir o tráfego de solicitação do Amazon S3 para o AWS Key Management Service (AWS KMS) e reduzir o custo da criptografia. Quando você usa criptografia do lado do servidor, o Amazon S3 criptografa um objeto antes de salvá-lo no disco e o descriptografa quando você baixa os objetos.

Bloqueio de Objetos (Object Lock): Com o S3 Object Lock, você pode armazenar objetos usando um modelo write-once-read-many (WORM). O Bloqueio de Objeto pode ajudar a evitar que objetos sejam excluídos ou sobrescritos por um período de tempo fixo ou indefinidamente. Você pode usar o Object Lock para ajudar a atender aos requisitos regulamentares que exigem armazenamento WORM ou simplesmente adicionar outra camada de proteção contra alterações e exclusão de objetos. O Bloqueio de objeto oferece duas maneiras de gerenciar a retenção de objetos: períodos de retenção e retenções legais. **Período de retenção** - especifica um período fixo de tempo durante o qual um objeto permanece bloqueado. Durante esse período, seu objeto fica protegido por WORM e não pode ser sobrescrito ou excluído. **Retenção legal** - oferece a mesma proteção que um período de retenção, mas não tem data de expiração. Em vez disso, uma retenção legal permanece em vigor até que você a remova explicitamente. As retenções legais são independentes dos períodos de retenção.