ANÁLISE DA PLATAFORMA AWS LAMBDA EM AMBIENTES DE NUVENS COMPUTACIONAIS

ANALYSIS OF THE AWS LAMBDA PLATFORM IN COMPUTATIONAL CLOUD ENVIRONMENTS

Yasmin Oliveira PISONI

yasmin.pisoni@hotmail.com

Aluna do curso de Bacharelado em Ciência da Computação

Centro Universitário Padre Anchieta - Jundiaí - SP

Carlos Eduardo CÂMARA

dinhocamara@gmail.com

Pesquisador na Área de Ciência da Computação

Doutorado e Mestrado em Engenharia Elétrica - FEEC/UNICAMP – Campinas/SP

Resumo

O conhecimento e o desenvolvimento de novos modelos de computação em nuvem podem levar a uma grande economia, nesse contexto, o modelo serverless computing apresenta-se como uma das possíveis soluções para disponibilização de aplicações e serviços online, tem como uma de suas principais características a disponibilidade de recursos por demanda. Esse estudo busca preencher a lacuna em relação a estudos que abordem a serverless computing, focando principalmente na plataforma AWS Lambda. Com esse estudo foi visto que o AWS Lambda elimina a complexidade de lidar com servidores, além de apresentar um modelo de faturamento de pagamento mediante solicitação, em que são eliminados os custos da capacidade computacional considerada ociosa.

Palavras-Chave

armazenamento; nuvem; serveless computing; AWS lambda, serveless, AWS

Abstract

The knowledge and development of new cloud emergency models can lead to great savings, in this context, the serverless computing model presents itself as one of the solutions for the provision of services and services online, with availability as one of its main characteristics of resources on demand. This study seeks to fill a gap in relation to studies that address serverless computing, focusing primarily on the AWS Lambda platform. With this study, it was seen that AWS Lambda eliminates the complexity of dealing with servers, in addition to presenting a pay-on-demand billing model, in which the costs of computing capacity considered idle are eliminated.

Keywords

storage; a cloud; serveless computing; AWS lambda, serveless, AWS

1.INTRODUÇÃO

O armazenamento em nuvem vem apresentando um crescimento exponencial e adquirindo mais usuários nos últimos anos. Esse tipo de armazenamento inovou a maneira como as informações são gerenciadas, acessadas e utilizadas. As nuvens computacionais disponibilizam diferentes recursos que são oferecidos como serviços pelo provedor. Os recursos desse tipo de armazenamento estão disponíveis de acordo com a capacidade e a quantidade que o usuário necessita, onde futuramente realizará o pagamento a respeito da quantidade de armazenamento adquirida (VAQUERO et al. 2009).

Desse modo, diante do alto crescimento de usuários de Internet observado nos últimos anos (IBGE, 2018), o volume de acessos para serviços e aplicações online tende a seguir aumentando de forma contínua. Portanto, o conhecimento e a descoberta de novos modelos de infraestruturas em nuvem podem gerar impactos diretos em custos computacionais.

Com o rápido avanço da tecnologia, surgem novos modelos de computação em nuvem e um dos considerados mais importantes é o denominado Serverless Computing, esse sistema é baseado na transparência ao cliente das informações do sistema operacional. A principal diferença desse sistema é a ausência dos servidores nas aplicações. A lógica da aplicação é definida pelo usuário, dessa forma quando é requisitada a execução, uma cópia da função é executada pelo provedor, que envia o resultado da execução de volta para o usuário (ADZIC; CHATLEY, 2017).

O sistema Serverless Computing é considerado um grande avanço em relação ao sistema de armazenamento em nuvem pois fornece toda a responsabilidade de gestão dos recursos para o provedor. Seu funcionamento é baseado no escalonamento de recursos de computação para executar funções *stateless* fornecidas por programadores e acionadas mediante eventos. O escalonamento e gestão dos recursos são de responsabilidade do provedor do serviço, e o preço é calculado com base na utilização durante o tempo de execução das funções em milissegundos. Exemplos de aplicações que utilizam o ambiente *serverless* são aplicações web, mobile, IoT, Big Data e *chatbots* (SREEKANTI et al. 2020).

Um dos principais modelos de computação que utiliza *serverless* é o de Function-as-a-Service (FaaS), que permite desenvolver, executar e gerenciar funcionalidades de forma mais prática. São *stateless* e temporárias, podendo permanecer ativas durante a execução de uma única requisição, e gerenciadas inteiramente pelo provedor, que fica responsável por garantir sua escalabilidade e disponibilidade. O usuário não é cobrado por recursos inativos. Dessa forma, esse modelo é econômico quando comparado aos mais tradicionais (ADZIC; CHATLEY, 2017).

As plataformas FaaS são capazes de escalonar centenas ou até milhares de funções em segundos ou minutos através da implementação da separação entre computação e armazenamento, esse tipo de designer está cada vez mais comum no armazenamento em nuvem. No entanto, a execução das funções utiliza o modelo *stateless*, isto é, sem persistência de dados, exigindo o uso de serviços de armazenamento externo para a troca de estado em aplicações *stateful*. Também há um *trade-off* entre a latência e o custo que deve ser considerado no uso das plataformas *serverless* (KLIMOVIC et al. 2018).

Para as empresas, com a nuvem não existe a necessidade da utilização de um hardware próprio, qualquer arquitetura baseada em servidor ainda exige que a escalabilidade e a confiabilidade sejam estruturadas (AWS, 2017). Elas serão responsáveis por escalar as possíveis frotas de servidores, levando em consideração a redução dos picos de cargas, quando possível para assim reduzir os custos, levando sempre em consideração a integridade dos usuários e dos sistemas (AWS, 2017).

O *serverless computing* foi desenvolvido para enfrentar os desafios que o mercado procura, oferecendo às empresas uma redução específica nos custos e no tempo de entrada no mercado (AWS, 2017). Nesse contexto, o modelo *serverless computing* apresenta-se como uma das possíveis soluções para disponibilização

de aplicações e serviços online, pois tem como uma de suas principais características a disponibilidade de recursos por demanda (AWS, 2017).

A AWS Lambda permite que as empresas executem código sem a necessidade de manterem seus servidores. Os desenvolvedores tornam-se responsáveis apenas pela execução do código e os valores cobrados são referentes ao número de instâncias que foram utilizadas após a execução dos mesmos. Ele possui uma grande gama de linguagens já pré-definidas que são aceitas, o AWS Lambda permite que eles se concentrem em suas tarefas sem ter que gerenciar servidores ou até mesmo que sejam responsáveis pelas atualizações do sistema operacional e todas as aplicações instaladas estejam atualizados (AWS, 2020).

A AWS garante que os ambientes de execução usados nas funções do Lambda estejam sempre em constante evolução. Isso significa que os ambientes de execução estão constantemente sendo substituídos por novos. Devido a isso, os dados salvos em um evento permanecerão lá apenas por um curto período, a não ser que os mesmos sejam solicitados novamente dentro da função (AWS, 2020).

A utilização do modelo *serverless* permite às empresas utilizarem um provedor de serviços de nuvem para reduzir despesas relacionadas às operações e manutenções dos servidores e aos demais processos associados, como gerenciamento de *patches* e escalonamento (OLIVEIRA, 2020). Dessa maneira, elas tornam-se responsáveis apenas pelo desenvolvimento de códigos, ao invés de usar recursos humanos para manter e proteger a infraestrutura do servidor. Isso significa que as empresas que optam por trabalhar com o *serverless* acabam se beneficiando devido a maior flexibilidade, automação, economia, agilidade e segurança de dados que o *serverless* oferece (OLIVEIRA, 2020).

Desse modo, a justificativa desse estudo é auxiliar na compreensão do s*erverless computing* com foco na plataforma AWS, assim como na análise de seus impactos em termos de eficiência, redução de custos e tempo de execução, com foco direto na AWS Lambda.

A divisão deste artigo está organizada da seguinte maneira: a seção 2 apresenta os modelos de computação em nuvem existentes e os conceitos de AWS Lambda. Na seção 3, a análise sobre a plataforma AWS Lambda. Na seção 4, apresenta o algoritmo simples de AWS Lambda sendo o “Hello Word”. E a seção 5 conclui o trabalho abordando os possíveis impactos da AWS Lambda em relação às empresas.

2.COMPUTAÇÃO EM NUVEM

O desenvolvimento da computação em nuvem, também denominada como Cloud Computing, está na frente para a centralização e gerenciamento de sites e aplicativos, vem causando muitas transformações digitais e já tem garantido o seu lugar dentro das grandes empresas atualmente (SOFTLINE, 2017). Ela considera a computação um serviço ao invés de um produto, sendo assim os desenvolvedores usam suas máquinas como um determinado serviço, facilitando a aplicação e os acessos aos dados em qualquer lugar que estiverem. Com isso as empresas não têm a necessidade de possuírem uma plataforma de desenvolvimento ou um servidor e através disso torna-se possível que as execuções das tarefas sejam realizadas de maneira mais simples e de forma remota (SOFTLINE, 2017). Alguns dos serviços prestados pela computação em nuvem são: servidores virtuais, armazenamento, softwares e desenvolvimento de softwares (SOFTLINE, 2017).

Com o grande leque de possibilidades quando se fala da computação em nuvem elas acabaram sendo divididas em três categorias. De acordo com (MELL, 2011), os três modelos principais de computação em nuvem são a Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS), e o Software como Serviço (SaaS).

No modelo SaaS o usuário é o responsável pelo gerenciamento da capacidade do sistema operacional. Nesse modelo as aplicações estão armazenadas em data centers de nuvem. Na maioria dos casos o usuário não é um desenvolvedor, embora também seja possível oferecer serviços que podem ser incorporados em outras aplicações. Nesse modelo é oferecido um produto completo, ou seja, o gerenciamento é executado pelo próprio

provedor, fazendo com que o usuário não se preocupe com a manutenção ou gerenciamento da infraestrutura (MELL, 2011)

Já o modelo PaaS é voltado para usuários que necessitam de maior controle sobre os recursos computacionais utilizados, permitindo o desenvolvimento de aplicações na nuvem, nesse modelo o usuário não possui a necessidade de gerenciar os sistemas operacionais e o hardware tendo a possibilidade mais eficiência e foco maior nas implantações e gerenciamento já que não realizará as tarefas repetitivas necessárias (MELL, 2011)

No modelo IaaS o gerenciamento é realizado pelo provedor de serviços e são oferecidos recursos, como processamento, rede e armazenamento, que são configurados e gerenciados pelo usuário (MELL, 2011). O IaaS possui os componentes básicos da infraestrutura na nuvem e apesar de os serviços serem realizados pelo provedor, ele possui flexibilidade e controle de gerenciamento. Esse modelo é o mais conhecido dentro dos departamentos de TI por possuir recursos semelhantes aos já existentes (SOFTLINE, 2017).

Além dessas categorias de nuvens que existem, a IaaS e a PaaS dividem outros três tipos de classificações que são: pública, privada e híbrida. Na nuvem pública todas as informações ficam disponíveis na internet e são compartilhadas entre diversos usuários. No modelo de nuvem privada os dados são de acesso próprio do servidor da empresa, mantendo os arquivos privados e mais seguros. Na híbrida ocorre a junção das duas anteriores (SOFTLINE, 2017).

A necessidade de constante crescimento de redes compartilhadas, com taxas de respostas mais rápidas para os serviços solicitados, com velocidade para armazenar recursos e com o tempo de execuções acelerados (MULLER et al. 2020) fez com que houvesse a necessidade de avanços na computação em nuvem, com isso as vantagens e competitividade dentro das empresas cresceram bastante com o decorrer dos anos, e dentre elas estão: a redução dos custos já que a Cloud Computing permite que as empresas tenham serviços fornecidos de acordo com as suas necessidades sem realizar grandes investimentos, a praticidade pois os procedimentos como instalação, manutenção e atualização ficam por conta dos fornecedores. Os modelos de computação em nuvem conseguem atender todos os tipos de demandas e vieram para auxiliar as empresas através de inovações independentes. A partir disso deu-se início a FaaS que é considerado também um modelo de computação em nuvem, já que com ele não é necessário que o usuário se preocupe com a manutenção da sua infraestrutura (CLEAN CLOUD, 2018).

Com isso, a computação em nuvem sofreu diversas evoluções com o decorrer dos anos e assim surgiu a *Serverless Computing* ou também conhecido como, computação sem servidores. É composta por uma arquitetura em que a execução dos códigos são gerenciados por provedores em nuvens diferente dos métodos tradicionais onde os desenvolvimentos são feitos através de instalações diretas nos servidores (IPSENSE, 2020), porém essas aplicações não estão completamente sem os servidores como o próprio nome já diz, a origem de "sem servidor" é em relação a experiência do cliente com os servidores já que eles acabam sendo invisíveis para os usuários que não possuem acessos ou qualquer tipo interação com os servidores (IBM, 2021).

O Serverless Computing transfere as responsabilidades de gerenciamento das tarefas operacionais e a infraestrutura da nuvem em back-end para o provedor da nuvem. (IBM, 2021).

Dentre as vantagens da *Serverless Computing* podemos listar as principais que são:

- A redução de custos: uma vez que só será realizada a cobrança caso o seu código esteja em execução (IPSENSE, 2020).
- Possui poucos riscos de ameaças ao sistema: os provedores possuem equipes especializadas em segurança que são responsáveis pela atualização, monitoramentos aos possíveis ofensores do sistema (IPSENSE, 2020).
- Ampliação Global: com a utilização das nuvens é possível ampliar as atividades de

determinadas regiões e realizar a implementação em questão de minutos, de onde estiver (IPSENSE, 2020).

Para o usuário conseguir obter os sucessos desejados é necessário possuir uma boa conexão com a internet, garantindo velocidade e escalabilidade necessárias para a aplicação da *Serverless Computing* e também garantir que as ferramentas utilizadas no dia-a-dia estejam alocadas em nuvem e, caso não estejam, o usuário precisará garantir que os seus recursos sejam escaláveis. Atualmente diversos provedores como Azure, Google e IBM oferecem a plataforma "sem servidor", porém, esse formato de plataforma ainda está em expansão no território nacional, com algumas empresas como Nubank, SemParar e MaxMilhas apostando na *Serverless* Computing um cenário diferente do que encontramos em muitos países do exterior (KOHN, 2018).

Uma das empresas que deu início a *Serverless Computing* foi, Amazon, uma empresa norte-americana que começou com a computação sem servidor no ano de 2014, com a plataforma Amazon Web Service, que fornece diversos serviços para os usuários. A AWS (Amazon Web Service), possui diversos recursos fornecidos, porém o foco deste estudo será o AWS Lambda.

2.1. Conceitos do Lambda

O AWS Lambda possui termos que são muito utilizados quando se faz referência às suas execuções. Algumas denominações para as chamadas de execução são:

- Função: "Recurso" onde após ser chamado, ocorre a execução do código. Utilizada para processar eventos, ou para realizar outros serviços solicitados por outras partes da AWS (AWS, 2021).
- Evento: Trata-se de uma documentação em formato de JSON que contém os dados para serem processados dentro da função Lambda (AWS, 2021).
- Ambiente de execução: Fornece um ambiente seguro e isolado para assim conseguir gerenciar e executar as funções chamadas (AWS, 2021).
- Arquiteturas de conjuntos de instruções: É onde é definido o tipo de processador no qual o Lambda irá executar as tarefas. As arquiteturas aceitas por ele são: arm64 (ARM de 64 bits) e x86_64 (x86 de 64 bits). Após definir a arquitetura não é possível alterá-la posteriormente (AWS, 2021).
- Pacote de implantação: O Lambda aceita dois modelos de pacote de implantação: o arquivo .zip com o código do usuário já inserido dentro do arquivo para assim o Lambda fornecer o seu sistema para realizar determinada função. E também o modelo imagem de contêiner, porém a mesma deve ser compatível com as especificações desejadas pela AWS (AWS, 2021).
- Runtime: O tempo de execução fornece um ambiente específico de código, onde ele transmite eventos com informações de resposta entre o Lambda e outras funções. O tempo escolhido pode ser criado pelo usuário ou escolher os disponíveis dentro da AWS, os tempos de execução devem ser definidos de acordo com os seus objetivos dentro da plataforma (AWS, 2021).
- Extensão: A extensão tem como objetivo permitir que o usuário aumente o seu número de funções, podendo criar várias extensões já existentes nas ferramentas parceiras do AWS ou até mesmo criar as suas próprias (AWS, 2021).
- Concurrency: ou simultaneidade representa o número de solicitações que uma determinada função recebe a cada momento, as funções só conseguem ser invocadas quando a anterior já estiver terminado (AWS, 2021).

Os termos acima serão citados no decorrer do artigo, e podem ser levados em consideração apenas quando forem abordados os temas referente a AWS Lambda.

3.AWS LAMBDA

O Amazon Web Services Lambda é um serviço de computação serverless computing orientado a eventos que possibilita o usuário a criar pequenas funções com funcionalidades únicas, onde cada uma é

responsável por desenvolver determinadas tarefas necessárias (MACHADO, 2020). Sua utilização tem similaridade com algumas das categorias mais comuns, como:

- Aplicativos Web: Automatização da implantação e hospedagem utilizando os recursos fornecidos pela AWS (AWS, 2021).
- Back-End Web e de aplicativos: Integração entre o backend e o frontend utilizando os recursos fornecidos pela AWS (AWS, 2021).
- Processamento de dados: Processamento de execução de dados, após serem acionados para realizar o armazenamento de dados (AWS, 2021).
- Atuações paralelas: divisão entre as tarefas de curta e longa duração, para conseguir executar os dados em paralelo, levando em consideração o tempo de duração de cada um (AWS, 2021).
- Carga de trabalho para internet das coisas: processamento de dados para os dispositivos ligados à internet das coisas (AWS, 2021).

O Lambda é o responsável por disponibilizar os recursos necessários para a execução das funções com base na solicitação ou eventos de entrada, para qualquer demanda, é responsável também por executar automaticamente os códigos dos eventos, por meio do Amazon API Gateway, Amazon S3, Amazon DynamoDB e, também, AWS Step Functions (AWS,2021).

O modelo seguido pelo Lambda é o de execução orientada a eventos, que é quando as funções são executadas através de *containers*. O Lambda é um dos recursos mais importantes que a AWS oferece, pois pode ser considerado o "futuro" da interação desenvolvedor x infraestrutura. Ele permite que os desenvolvedores usem a infraestrutura da AWS sem ter preocupações, já que o mesmo é responsável pela implantação, gerenciamento, manutenção do servidor e do sistema operacional, escalabilidade automática e a implantação do código, registro das execuções do código, monitoramento da integridade do servidor e disponibilização das estatísticas detalhadas além de toda a infraestrutura necessária, tudo que o usuário precisará fornecer é apenas o código em uma linguagem que o sistema suporte. (AWS, 2021)

Com os recursos oferecidos pela AWS o usuário é capaz de acionar mais de 200 serviços e aplicações (AWS, 2021). E através do exemplo abaixo é possível notar que para criar determinadas aplicações é necessário apenas combinar o Lambda com outros serviços como os citados anteriormente que são: Amazon S3, Amazon API e Amazon DynamoDB.

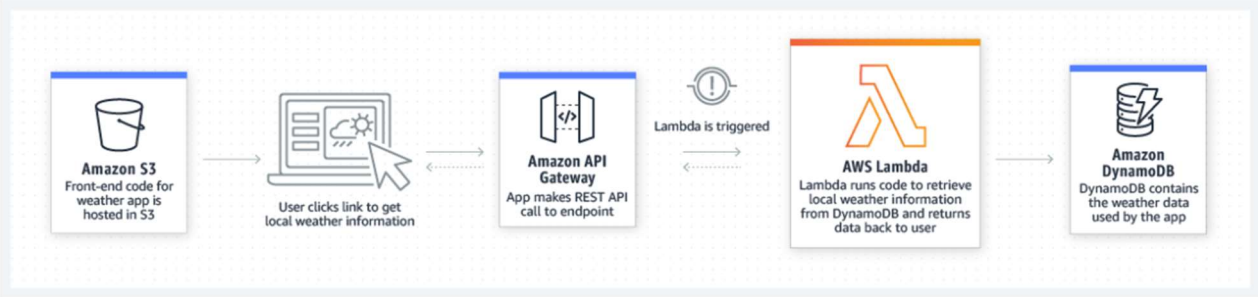


Figura 1. Um exemplo de aplicação Web (AWS, 2021)

O código responsável pela execução no AWS Lambda é chamado de "função Lambda". Após criar a sua função ela ficará disponível para futuras execuções e sempre que forem acionadas pelo servidor. As funções não possuem afinidades com a infraestrutura, pois apenas assim elas podem ser escaladas mais rapidamente de acordo com o número de solicitações que forem recebidas (AWS, 2021). Além disso, as funções são executadas apenas quando são chamadas automaticamente, podendo ser algumas por dia até milhares por segundo, sendo o Lambda um serviço altamente disponível (AWS, 2021). Ele é configurado para

dimensionar instantaneamente uma determinada demanda para um processo de execução de código em paralelo. Ele possui também o *downscale* que permite que as funções desnecessárias parem de funcionar automaticamente a partir da execução do código (AWS, 2021).

O Lambda aceita diferentes configurações para as funções utilizando códigos em formato ZIP ou de imagens de contêiner. As linguagens aceitas no Lambda são: Java, Python e Node.JS e outras linguagens, que possuem frameworks de código aberto. Há também ferramentas para *deploys* ainda mais convenientes e rápidos, como Apex, *Serverless* e Sparta que podem ser utilizadas (AWS, 2021).

3.1. Segurança da Nuvem AWS Lambda

A segurança da nuvem AWS Lambda é responsável pela infraestrutura que executa os produtos dos usuários, eles contam com auditorias de terceiros que são responsáveis pela realização dos testes de verificação da eficácia da segurança e os seus serviços são protegidos por procedimentos de segurança da rede global da AWS. (AWS, 2021)

Entretanto as responsabilidades de segurança são compartilhadas com os usuários, eles têm como responsabilidade manter o controle dos conteúdos hospedados dentro da AWS, além de garantir que seus dados de acesso na plataforma sejam confidenciais e caso sejam compartilhados para terceiros ambos os lados devem ter ciência. (AWS, 2021).

Em casos em que o AWS Lambda seja usado junto com outro serviço, as empresas devem estar cientes das permissões concedidas às demais. O usuário responsável pelas permissões deve conceder à empresa terceira apenas o privilégio mínimo, já que em casos como esse as permissões para usuários terceiros precisam ser concedidas de forma manual (OLIVEIRA, 2020).

A segurança de Lambda baseia-se na mecânica de que usuário e plataforma possuam responsabilidades em relação às informações confidenciais pertencentes dentro da plataforma, como exemplo, temos usuários e senhas para acessos, codificações, informações compartilhadas entre Lambda e o usuário que variam de suas necessidades (AWS, 2021).

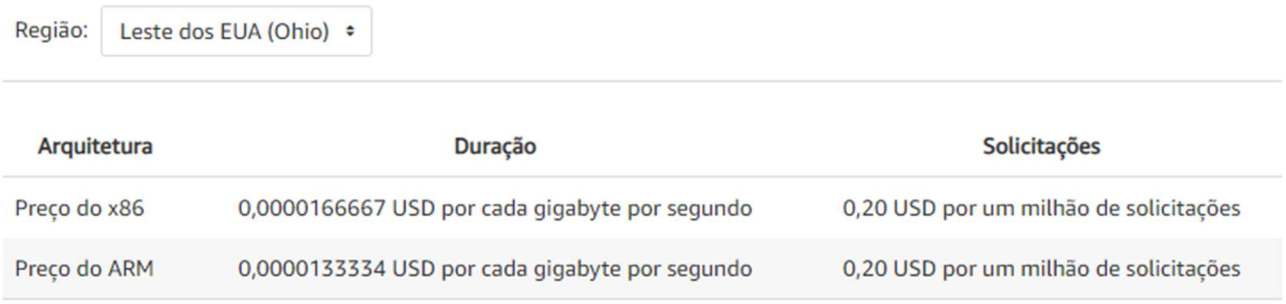
O Lambda atua com a responsabilidade compartilhada favorecendo ambos os lados, já que cada lado torna-se responsável apenas pelas partes que são suas responsabilidades.

3.2. Valores dos serviços

Seu serviço é considerado muito econômico, pois o seu modelo de precificação é o "pay-as-you-go", ou seja, os clientes pagam apenas pelos recursos utilizados, sendo eles, número de solicitações para as funções configuradas e a duração necessária para realizar a execução do código. Os valores começam a ser calculados a partir do momento em que a função começa a ser executada e só para de calcular a partir do momento em que ela se encerra, o valor é sempre arredondado para um milissegundo mais próximo (AWS, 2021). O usuário ao configurar os recursos que deseja utilizar deverá escolher a quantidade de memória que será necessária para cada função, o que irá determinar a capacidade da CPU e deve também definir o tempo de execução.

As funções podem ser executadas a partir de processadores com arquiteturas x86 ou ARM, e de acordo os valores obtidos na AWS Lambda, “Usando uma arquitetura de processador baseada em ARM projetada pela AWS, oferecem performance de preço até 34% menor em comparação com as funções em execução em processadores x86. Isso se aplica a uma variedade de workloads sem servidor, como processamento de backends da Web e móveis, de dados e de mídia.” (AWS, 2021).

A partir desses dados é possível obter uma amostra dos valores com relação à arquitetura escolhida pelo usuário, como localização foi utilizado “Leste dos EUA (Ohio)”, podendo ocorrer alterações dos valores de acordo com a região.



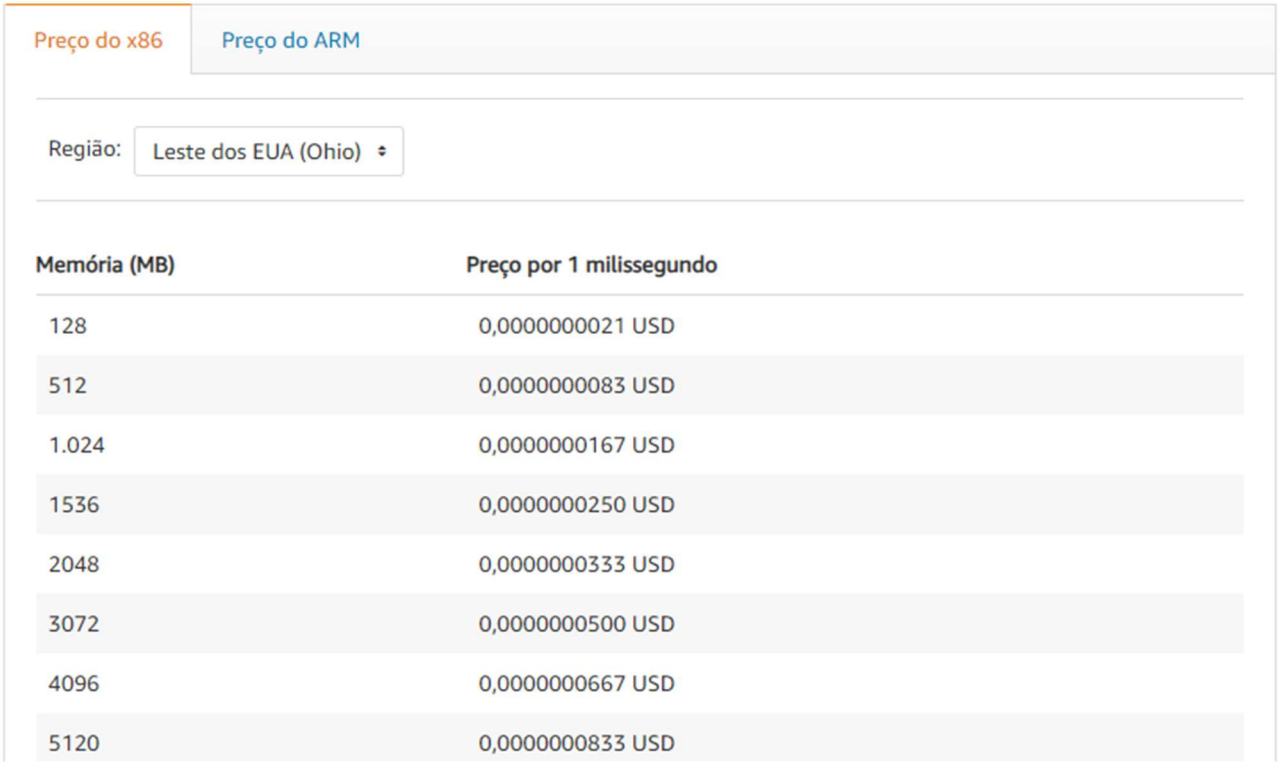
Região: Leste dos EUA (Ohio)

Arquitetura	Duração	Solicitações
Preço do x86	0,0000166667 USD por cada gigabyte por segundo	0,20 USD por um milhão de solicitações
Preço do ARM	0,0000133334 USD por cada gigabyte por segundo	0,20 USD por um milhão de solicitações

Figura 2. Relação dos valores do AWS Lambda em relação à arquitetura e estão baseados em preços no Leste dos EUA. (AWS, 2021)

E, também, em relação à capacidade de memória que é responsável por definir a tempo de execução para cada função, neste exemplo foi utilizado “Leste dos EUA (Ohio)”.

Os valores disponíveis são referentes à arquitetura x capacidade de memória, as quais serão selecionadas através da necessidade de cada usuário. Nas imagens não foram disponibilizados todos os tipos de memória que o AWS Lambda oferece para os usuários.



Preço do x86 | Preço do ARM

Região: Leste dos EUA (Ohio)

Memória (MB)	Preço por 1 milissegundo
128	0,0000000021 USD
512	0,0000000083 USD
1.024	0,0000000167 USD
1536	0,0000000250 USD
2048	0,0000000333 USD
3072	0,0000000500 USD
4096	0,0000000667 USD
5120	0,0000000833 USD

Figura 3. Relação dos valores do AWS Lambda em relação à capacidade de memória desejada pelo usuário. A arquitetura utilizada é a x86. Os valores estão baseados em preços no Leste dos EUA (AWS, 2021).

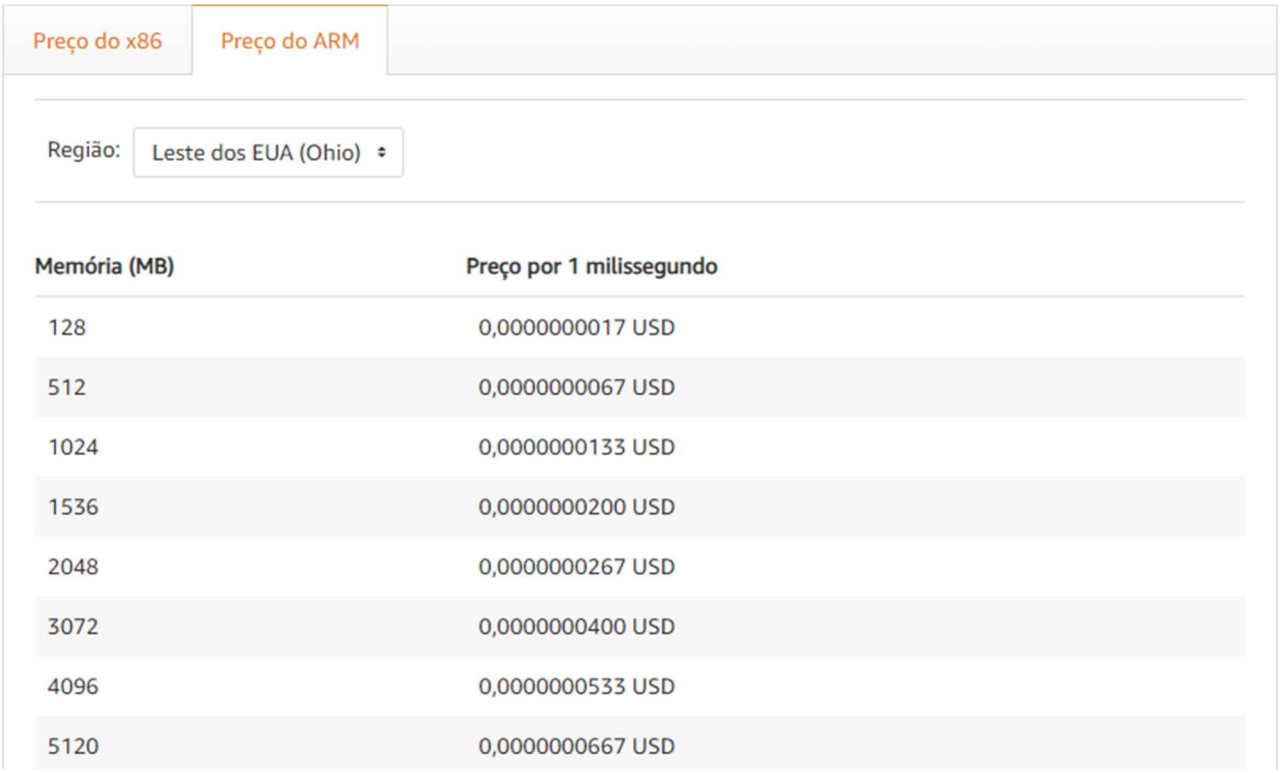


Figura 4. Relação dos valores do AWS Lambda em relação à capacidade de memória desejada pelo usuário. A arquitetura utilizada ARM. Os valores estão baseados em preços no Leste dos EUA. (AWS, 2021)

Além desses custos citados acima existe também a 'simultaneidade provisionada', isso significa que o usuário paga apenas pela quantidade de simultaneidade solicitada e pelo seu tempo de configuração e os valores acima entram apenas quando o limite de execução é ultrapassado. E em casos que ocorra transferência de dados será cobrado os valores por demanda de tarefas.(AWS, 2021)

Quando se fala sobre os valores de uma determinada plataforma nova no mercado, pode-se acabar gerando alguma insegurança para muitos dos usuários, já que para muitas plataformas é necessário contratá-las realizando o seu pagamento, sem ao menos ter a possibilidade de realizar alguns testes, para realmente entender como funciona a plataforma e se ela irá se encaixar dentro das atividades que uma determinada empresa irá precisar. Porém, com AWS Lambda o usuário pode realizar os testes desejados pagando apenas por um valor baixo, ou seja, ele irá pagar apenas o que utilizará sem a necessidade de contratar o serviço. Em alguns casos é possível realizar alguns testes de forma gratuita, para isso é necessário verificar na documentação se as funções que precisará executar se encaixam nos requisitos.

Os valores acima foram apresentados para tranquilizar e informar aos usuários que ainda possuem algum receio, devido a algumas experiências de que a plataforma possui valores consideravelmente baixos, apesar dos valores apresentados estarem em dólar. Com a grande necessidade de economia de muitos usuários e empresas a computação em nuvem chegou para realizar essa tarefa, pois além de possuírem valores extremamente baixos não necessitam de servidores mantendo apenas os em nuvem, sendo assim também acaba não tendo a necessidade de manutenção dos mesmos, diminuindo ainda mais os custos.

3.3. Tempo de execução

O tempo de execução do Lambda pode ser escolhido pelo próprio usuário de acordo com a função selecionada. Ele é gerado junto com a Amazon Linux ou Amazon Linux 2. Para realizar o cálculo do tempo de execução o Lambda envia a função para o denominado "ambiente de execução", que é considerado seguro e, a partir de lá, consegue gerenciar os recursos necessários para executar a função. O ambiente de execução é responsável por administrar os recursos que são responsáveis por realizar uma determinada função, fornecendo suporte para o tempo de execução e determinadas extensões externas (AWS, 2021).

Na imagem a seguir é possível visualizar de maneira clara, como funcionam as comunicações dentro de Lambda para determinar o tempo de execução de uma função. As extensões utilizam o API extensões para se comunicarem, o Lambda utiliza a API Runtime, e as extensões recebem todas as mensagens que contenham logs da função. (AWS, 2021).

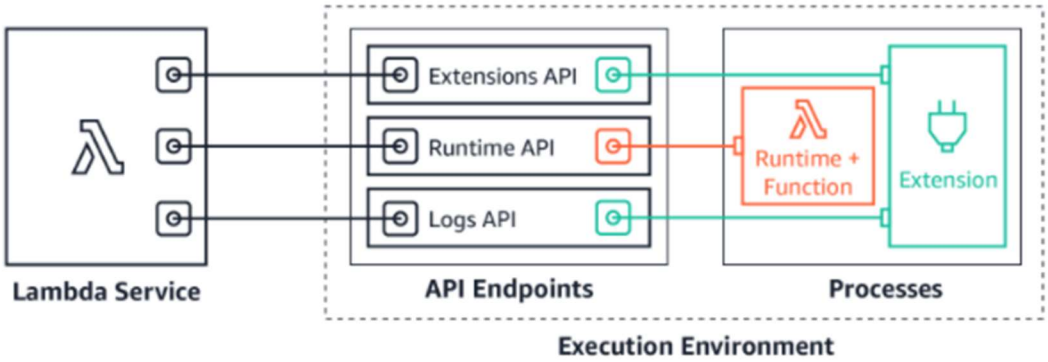


Figura 5. O processo entre as comunicações dentro do Lambda (AWS, 2021).

O tempo de execução pode rodar por várias vezes e só parar quando atingir a sua vida útil. As funções possuem um ciclo de vida que contém três etapas: inicial (*Init*), invocação (*Invoke*) e a final (*Shutdown*).

- Init: Ocorre criação do ambiente de execução ou descongelamento, as seguintes tarefas que são executadas nessa fase são: Inicialização das extensões (*Extension init*); Tempo de execução da inicialização (*Runtime init*) e a execução da função de código estático (*Function init*). Essas fases garantem que todas as extensões concluam as suas tarefas já configuradas antes que o código seja executado (AWS, 2021).
- Invoke:O Lambda envia os eventos para dentro de cada extensão e seu tempo de execução, fazendo com que a função seja executada dentro do tempo limite, sendo esse, já pré definido pelo usuário (AWS, 2021).
- Shutdown: Na fase final o Lambda envia eventos para as extensões informando que serão encerradas (AWS, 2021).

E em casos em que o usuário deseje utilizar outras linguagens que não estão nas configurações automáticas do AWS Lambda, podem ser utilizados tempo de execução personalizados (AWS, 2021).

O tempo de execução ilustrado abaixo é referente a utilização do python, a mesma linguagem que será utilizada para desenvolvimento do “Hello World” no decorrer do artigo.

Python runtimes

Nome	Identificar	SDK AWS para Python	Sistema operacional	Arquiteturas
Python 3.9	`python3.9`	boto3-1.18.55 botocore-1.21.55	Amazon Linux 2	x86_64, arm64
Python 3.8	`python3.8`	boto3-1.18.55 botocore-1.21.55	Amazon Linux 2	x86_64, arm64
Python 3.7	`python3.7`	boto3-1.18.55 botocore-1.21.55	Amazon Linux	x86_64
Python 3.6	`python3.6`	boto3-1.18.55 botocore-1.21.55	Amazon Linux	x86_64
Python 2.7	`python2.7`	boto3-1.17.100 botocore-1.20.100	Amazon Linux	x86_64

Figura 6. Tempo de execução para a Linguagem de Phyton, listando variações de sistema operacional x arquitetura (AWS, 2021).

De modo geral o tempo de execução do Lambda é consideravelmente rápido podendo executar determinadas funções em questão de milissegundos, utilizando diferentes linguagens de programação e sistemas operacionais. O Lamba consegue diferenciar uma função que demora mais tempo do que outras que demoram menos, e a partir disso ele é capaz de fazer com que as funções sejam executadas de forma paralela, para que não ocorra uma fila na execução impedindo que tarefas rápidas, levam um grande tempo para serem executadas devido às demandas (AWS, 2021).

A utilização do Lamba não é recomendada para usuários que desejam realizar grandes funções, visto que o tempo limite de execução do Lambda é de 15 minutos, em casos que o tempo ultrapasse o usuário terá que repartir em partições menores executando apenas uma por função, ao invés de executar todas apenas uma vez, sendo que algumas dessas partições podem rodar simultaneamente, porém isso dependerá do Lambda (MACHADO, 2020).

4.Desenvolvimento de um programa simples - "Hello Word em AWS LAMBDA"

Nessa seção será abordado como criar uma função Lambda sendo ela, “Hello World” através do console. Os exemplos a seguir foram retirados da página *‘Execute um aplicativo “Hello, World!” sem servidor’* (LAMBDA, 2021), fornecida pela AWS como tutorial para inicialização dos conceitos básicos sem a necessidade de gerenciamento de servidores.

Antes de conseguir acessar a plataforma Lambda foi necessário criar uma conta gratuita dentro da AWS. É necessário verificar na plataforma quais serviços são fornecidos gratuitamente.

O algoritmo apresentado a seguir mostra o passo a passo que é necessário realizar antes de chegar na parte de testes do Lambda.

Configurações Iniciais da AWS Lambda

Passo 01: Criar uma conta no AWS Lambda.

Passo 02: Acessar o console da AWS.

Passo 03: Selecionar AWS Lambda.

Passo 04: Criar uma função.

Passo 05: Em "criar uma função" selecionar "Usar um esquema".

Passo 06: Na barra de pesquisa procurar por "hello-world-python".

Passo 07: Após selecionar "hello-world-python" clicar em configurações.

Passo 08: Configurando as funções: Inserir o nome da função que desejar.

Passo 09: Selecionar papel de execução "Criar uma execução a partir da política de AWS templates".

Passo 10: Salvar as informações inseridas na etapa anterior.

Passo 11: Configurar o tempo de execução da função que será definido a partir da linguagem que irá ser utilizada. Foi utilizado o "Python 3.7".

Passo 12: Definir também o manipulador. Foi utilizado "lambda function.lambda handler".

Passo 13: Selecionar a arquitetura. Foi utilizado o x86_64.

Passo 14: Salvar as informações inseridas na etapa anterior.

Passo 15: Editar as configurações básicas.

Passo 16: Inserir uma descrição, essa é uma etapa opcional.

Passo 17: Inserir a quantidade de memória que será necessária para realizar a função. Foi utilizado 128 MB.

Passo 18: Definir qual será o tempo limite para a execução da função. Foi utilizado 3 segundos.

Passo 19: Salvar as informações inseridas na etapa anterior.

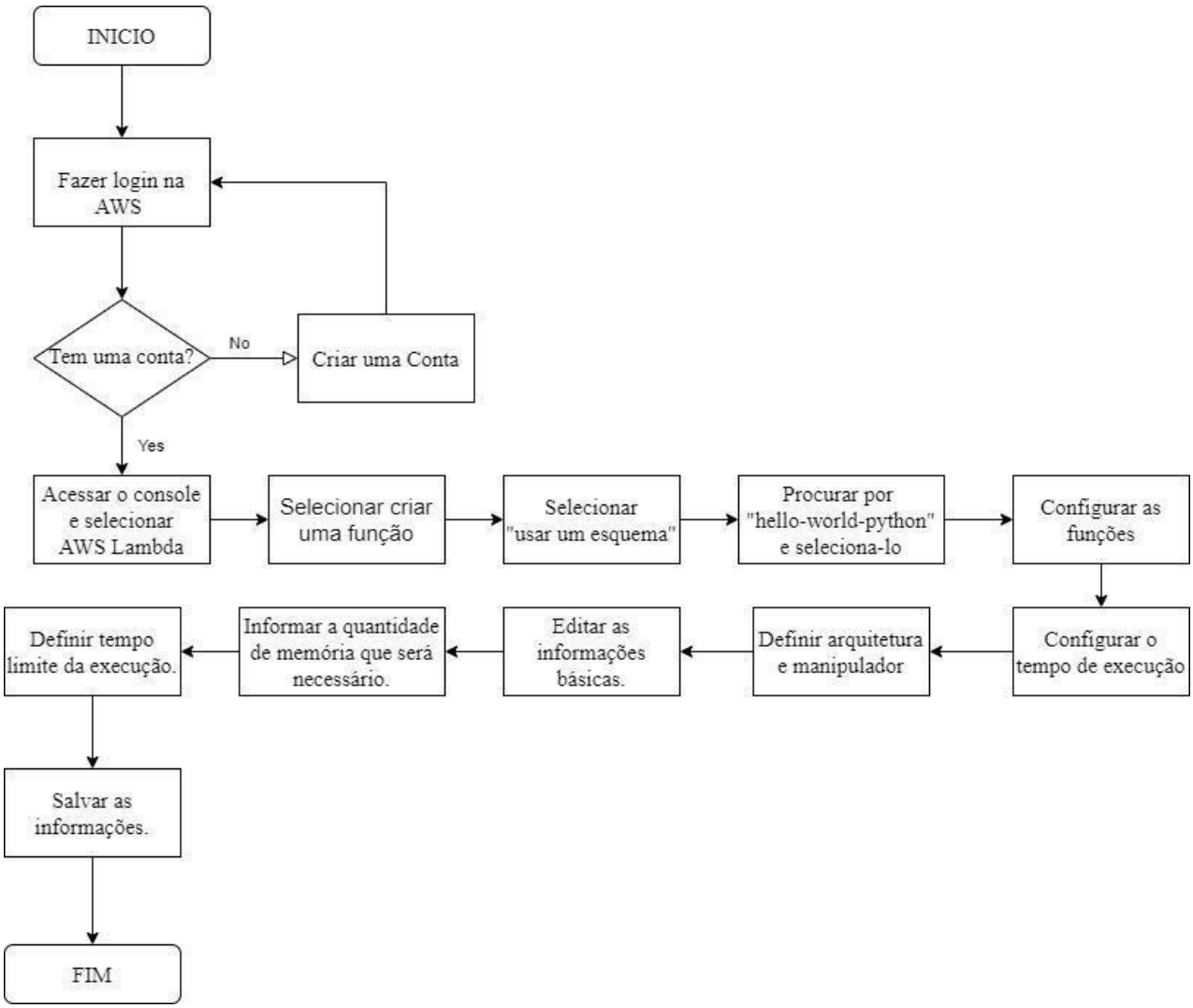


Figura 7. Fluxograma com o passo a passo para configurar uma função dentro da AWS Lambda (Elaborado pela autora, 2021).

Após realizar o registro na plataforma, e as devidas configurações o usuário deverá seguir o seguinte algoritmo e fluxograma para realização dos testes:

Início da realização dos testes:

Passo 01: Abrir a origem do código. O código já estará preenchido com as configurações da "hello-world-python".

Passo 02: Abrir “configurar evento de teste”.

Passo 03: Selecionar "Criar novo evento de teste".

Passo 04: Definir Modelo de evento. Foi selecionado "hello world".

Passo 05: Definir nome do evento. Pode ser definido pelo usuário.

Passo 06: Inserir os valores que deseja que apareça no console.

Passo 07: Salvar as informações definidas na etapa anterior.

Passo 08: Clicar em realizar o teste.

Passo 09: Após isso, será exibida a informação de sucesso, junto com a resposta desejada "Hello, world".

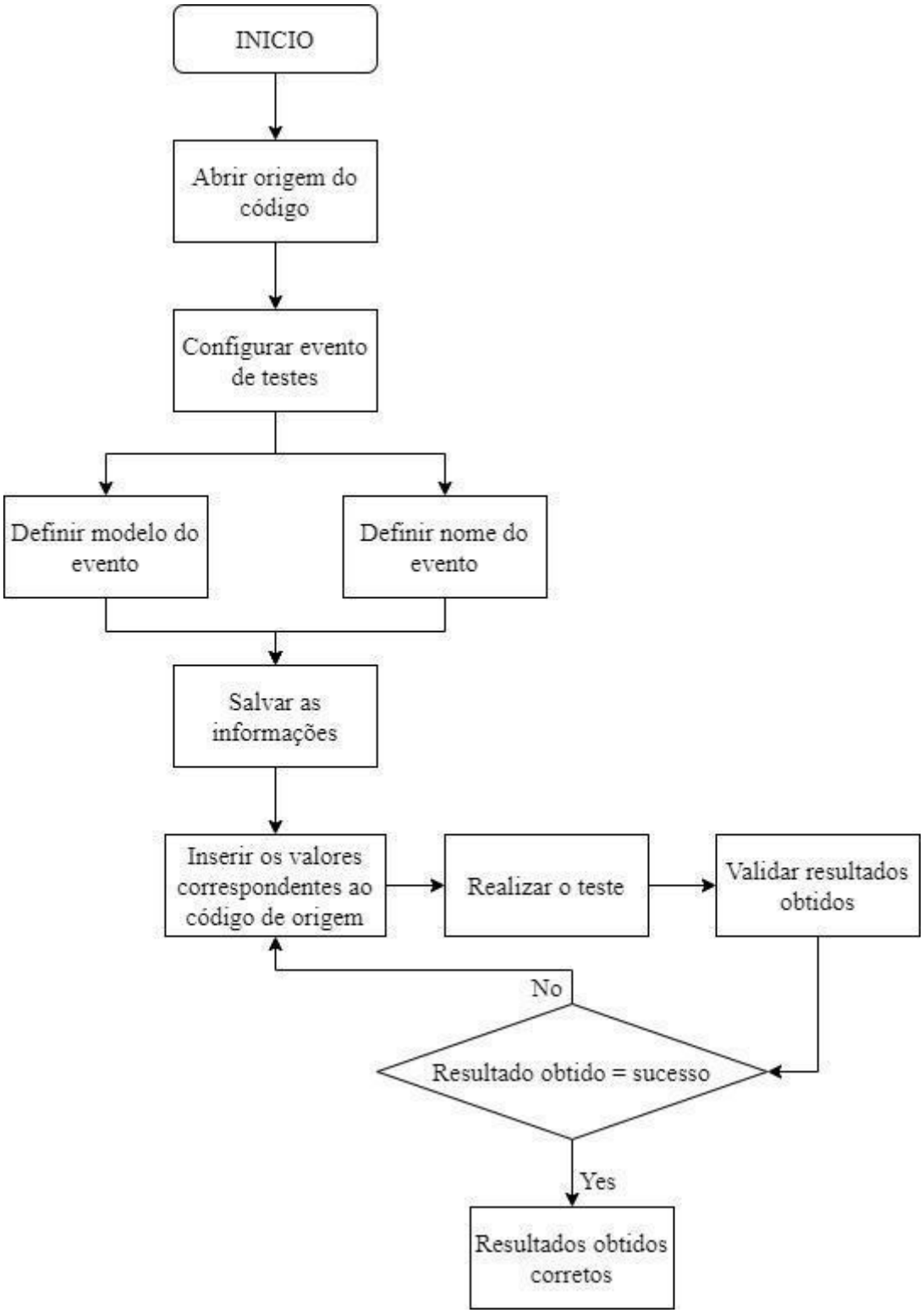


Figura 8. Fluxograma com o passo a passo para a realização dos testes dentro da AWS Lambda (Elaborado pela autora, 2021).

Todos os passos listados acima foram seguidos e realizados com auxílio do tutorial, as imagens abaixo foram realizadas dentro da plataforma AWS Lambda, durante a realização da função.

Segue de forma detalhada todo o processo listado acima:

Ao acessar AWS Service, selecionar: Todos os serviços> Computação -> Lambda e assim abrir o console do Lambda.

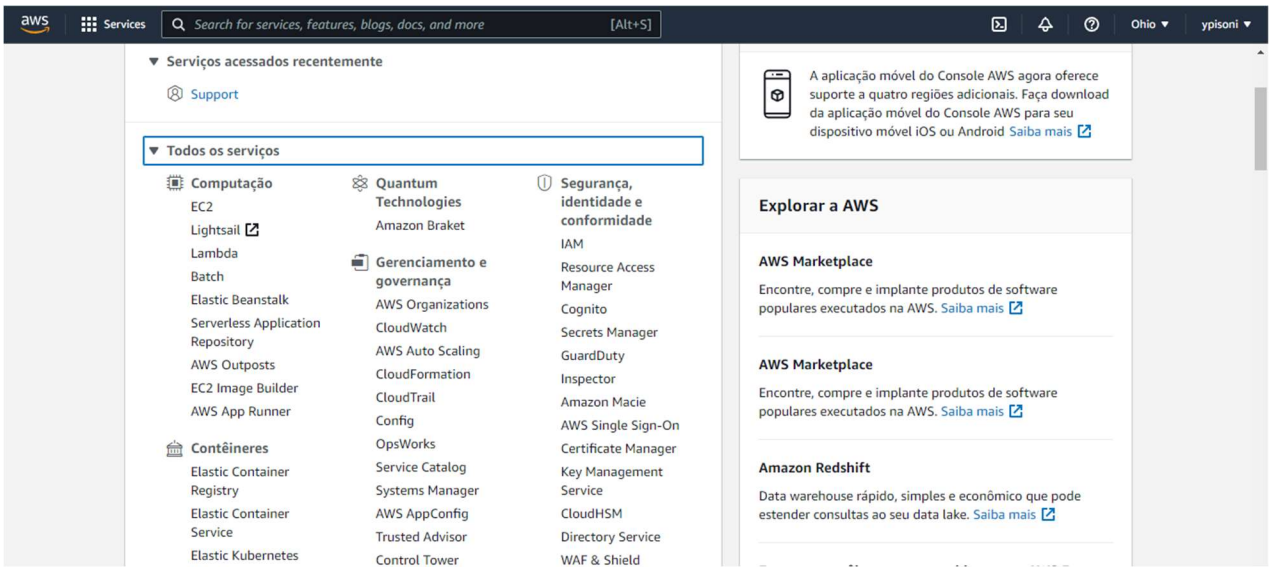


Figura 9. Console AWS, onde existem os serviços oferecidos pela AWS (Desenvolvido pela autora, 2021).

Após clicar em "Lambda", o usuário será redirecionado para a página responsável pelo desenvolvimento das funções do Lambda, a página aparece com as funções zeradas. Porém, como foram realizados alguns testes é possível visualizar algumas funções já criadas.

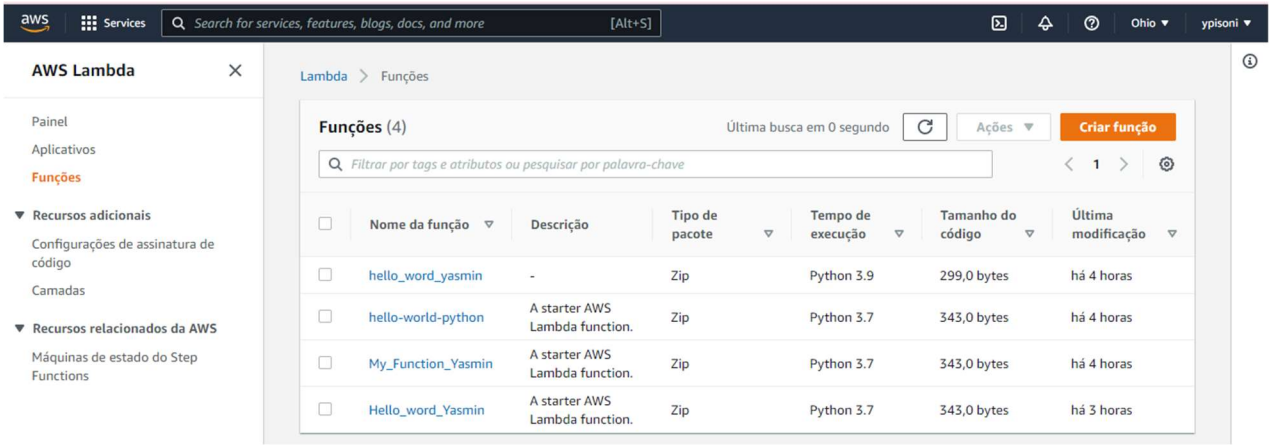


Figura 10. Exibe as funções criadas dentro da plataforma AWS Lambda (Elaborado pela autora, 2021).

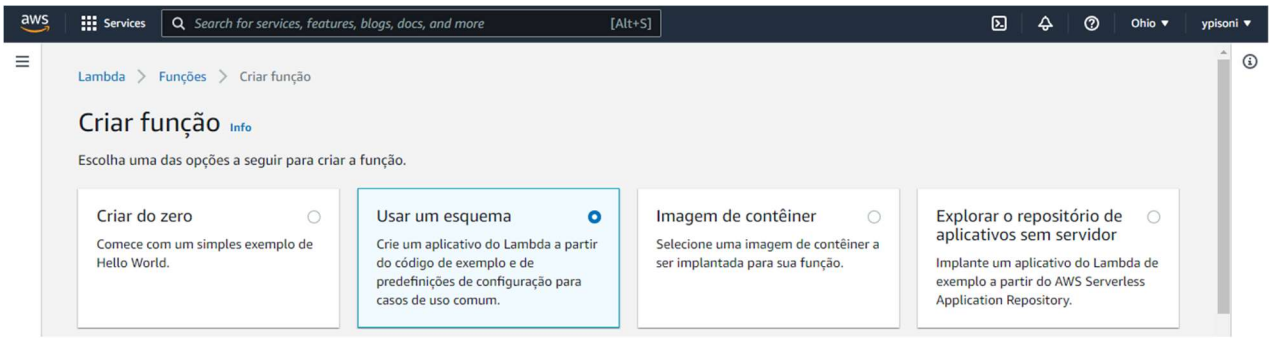


Figura 11. Exibe as opções para criação de uma função no AWS Lambda (Desenvolvido pela autora, 2021).

Após a página Lambda> Funções ser aberta o usuário deverá clicar em criar a função. Ao abrir a página de Lambda> Funções -> "Criar função", deverá ser selecionado a seguinte ação "Usar um esquema", foram sugeridos alguns esquemas já pré-definidos pelo Lambda, o esquema que iremos utilizar é o "*hello-world-python*".

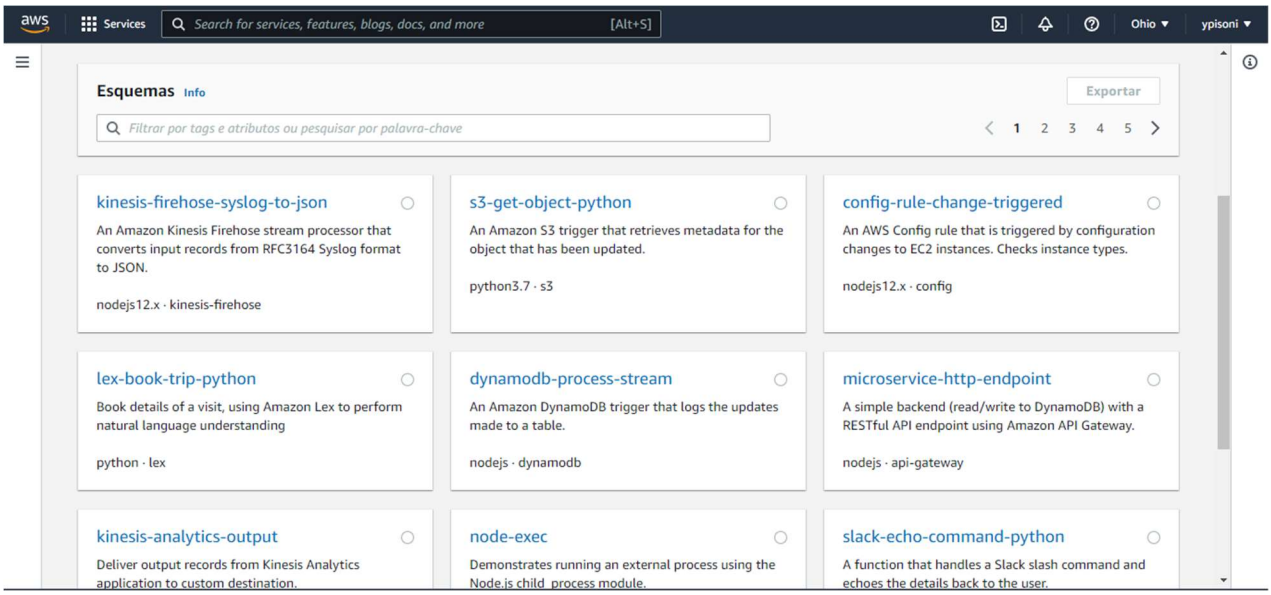


Figura 12. Exibe as opções já pré-definidas de esquema já configurados pela AWS (Desenvolvido pela autora, 2021).

Ao selecionar o usuário deverá escolher "configuração" e será direcionado para a página onde é possível realizar as configurações básicas, como: nome da função e papel da execução, nessa etapa o usuário também criará uma função do IAM escolhendo uma que definirá as permissões para a sua função que o AWS Lambda pode estar assumindo para invocar a função lambda.

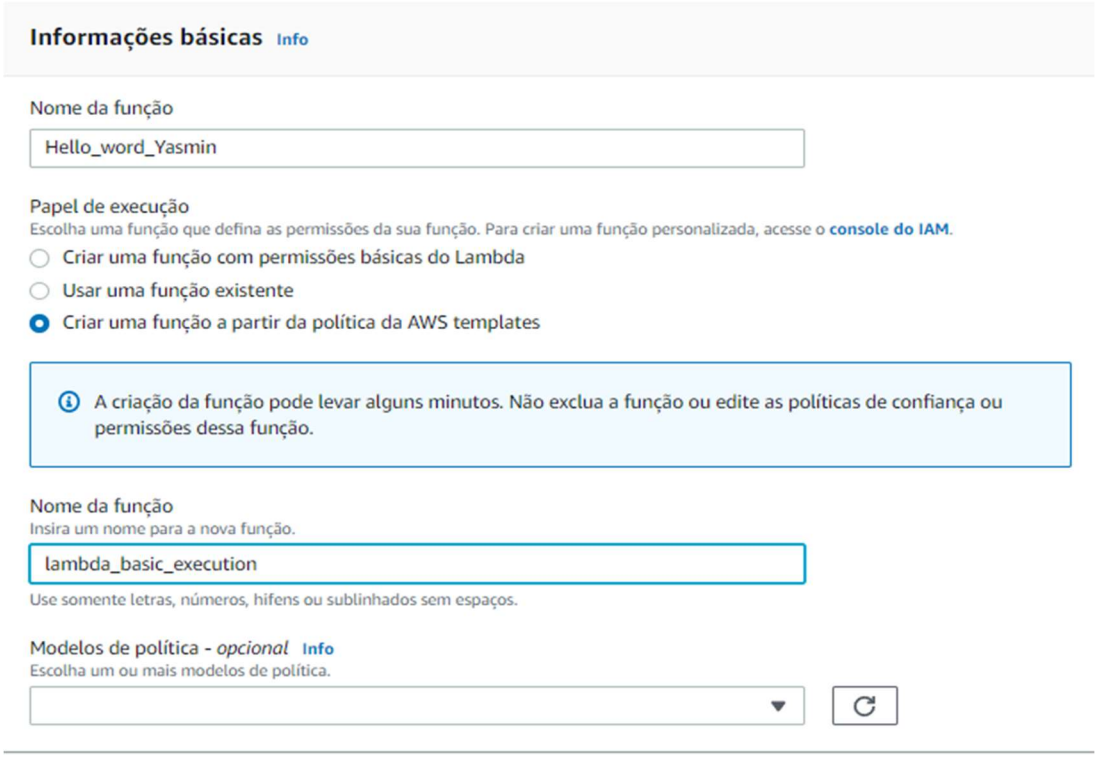


Figura 13. Exibe as opções para as configurações básicas da função Lambda (Desenvolvido pela autora, 2021).

O próximo passo do usuário é definir o tempo de execução, onde deve ser definido o código de função, e nesse caso foi utilizado a linguagem Python 3.7. O usuário também pode especificar um manipulador em que o AWS Lambda possa começar a executar seu código, para o nosso exemplo foi utilizado o "lambda_function.lambda_handler"

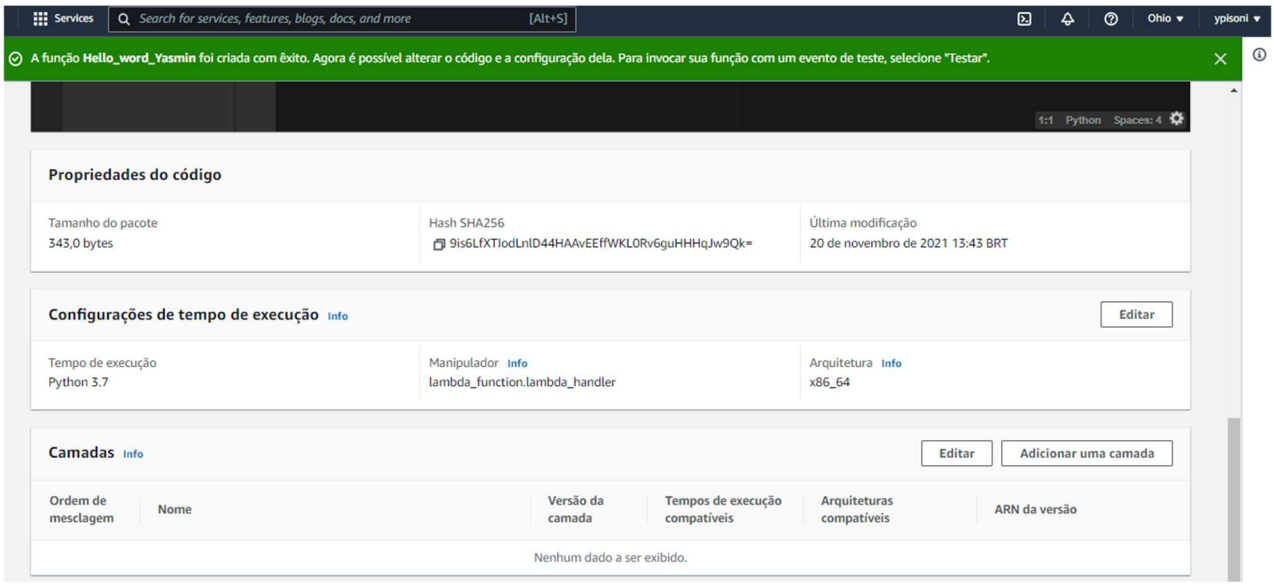


Figura 14. Exibição das configurações básicas da função Lambda (Desenvolvido pela autora, 2021).

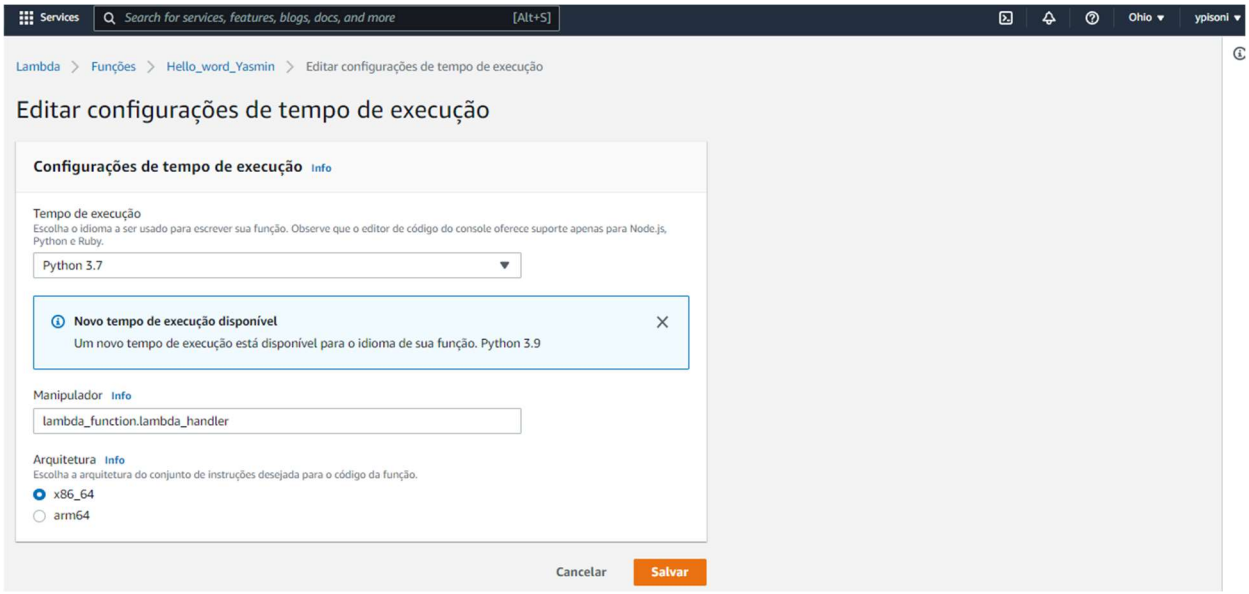


Figura 15. Edição da configuração do tempo de execução da função lambda (Desenvolvido pela autora, 2021).

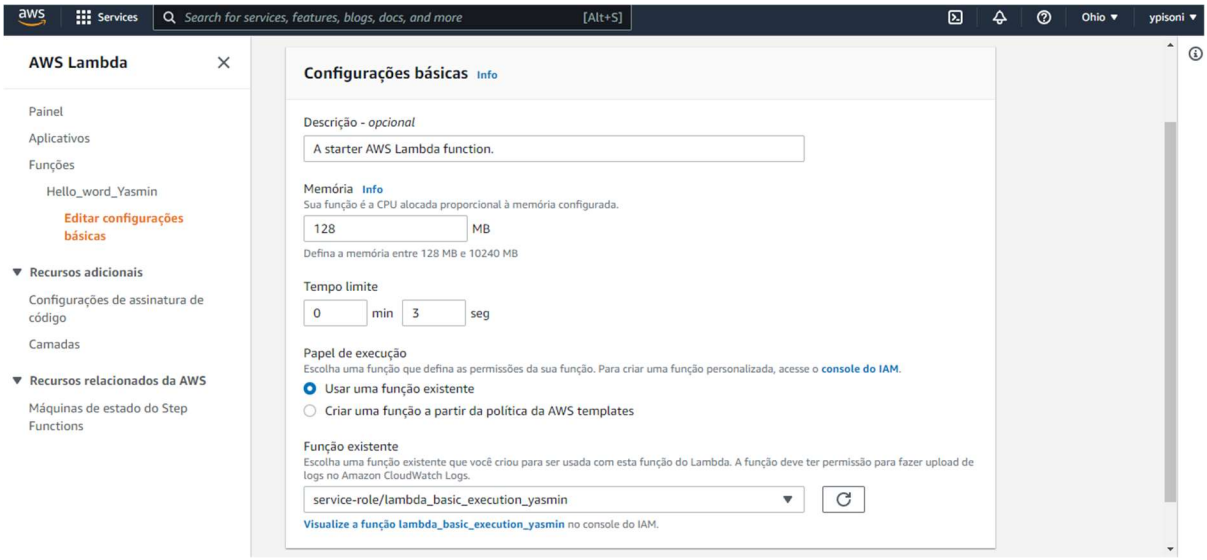


Figura 16. Edição da configuração do tempo de execução da função Lambda (Desenvolvido pela autora, 2021).

Após a finalização das configurações básicas o usuário pode passar a realizar testes. O código retirado através da “AWS Lambda” utilizando JSON é:

```
import json

print('Loading function')
```

```
def lambda_handler(event, context):
    print("value1 = " + event['key1'])
    print("value2 = " + event['key2'])
    print("value3 = " + event['key3'])
    return event['key1']
```

Os valores utilizados foram:

```
{
  "key1": "hello, word!",
  "key2": "Yasmin Pisoni",
  "key3": "value3"
}
```

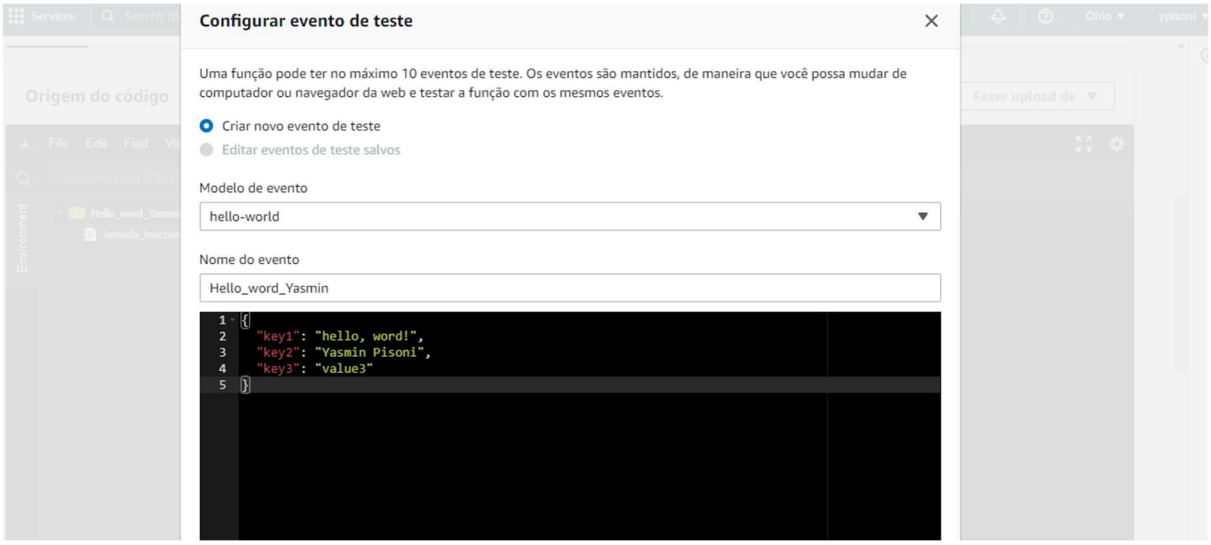


Figura 17. Configuração do evento de teste (Desenvolvido pela autora, 2021).

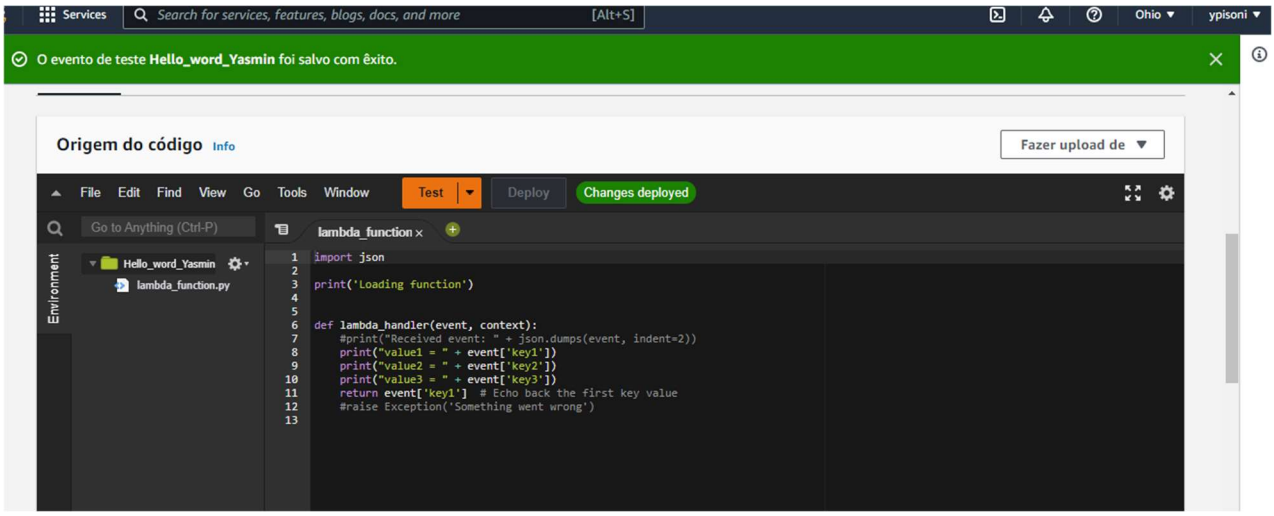


Figura 18. Realização do evento de teste da função lambda (Desenvolvido pela autora, 2021).

Após a realização do teste foi obtido o seguinte resultado:

Nome do evento:

Hello_word_Yasmin

Resposta:

"hello, word!"

Função

START RequestId: f401c829-c5ce-440f-9504-837ad5e4a342 Version: $LATEST

value1 = hello, word!

value2 = Yasmin Pisoni

value3 = value3

FIM RequestId: f401c829-c5ce-440f-9504-837ad5e4a342

REPORT RequestId: f401c829-c5ce-440f-9504-837ad5e4a342

Duração: 2.05 ms

Duração faturada: 3 ms

Tamanho da memória: 128 MB

Memória máxima utilizada: 37 MB

Request ID

f401c829-c5ce-440f-9504-837ad5e4a342

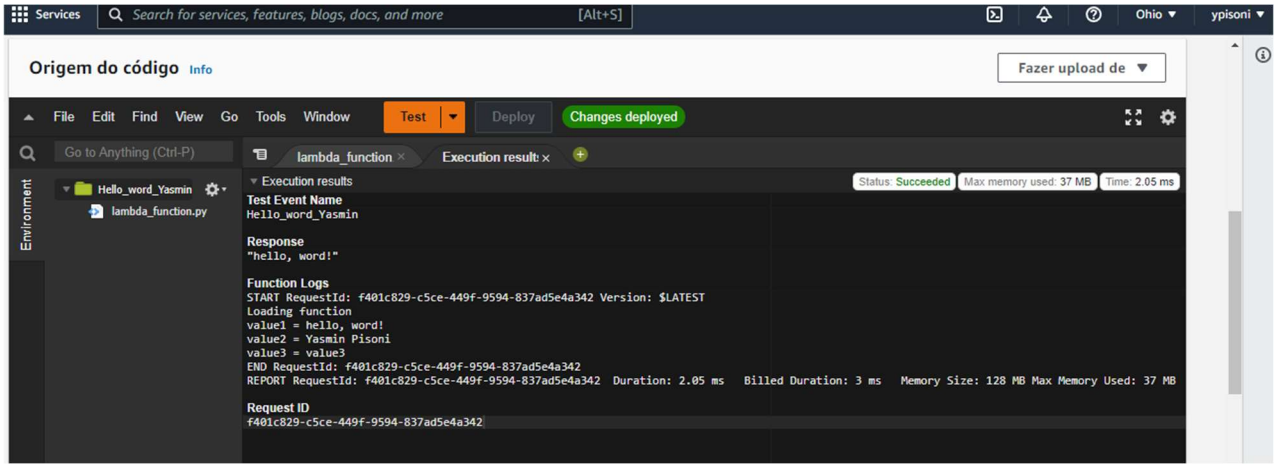


Figura 19. Execução bem sucedida conforme observado na tela do console (Desenvolvido pela autora, 2021).

Os resultados obtidos acima representam que a função foi executada corretamente. Ao finalizar a criação da função é possível verificar através do painel do AWS Lambda quantas funções o usuário possui, sendo elas já criadas anteriormente e qual a quantidade de memória que está sendo utilizada pelas funções.

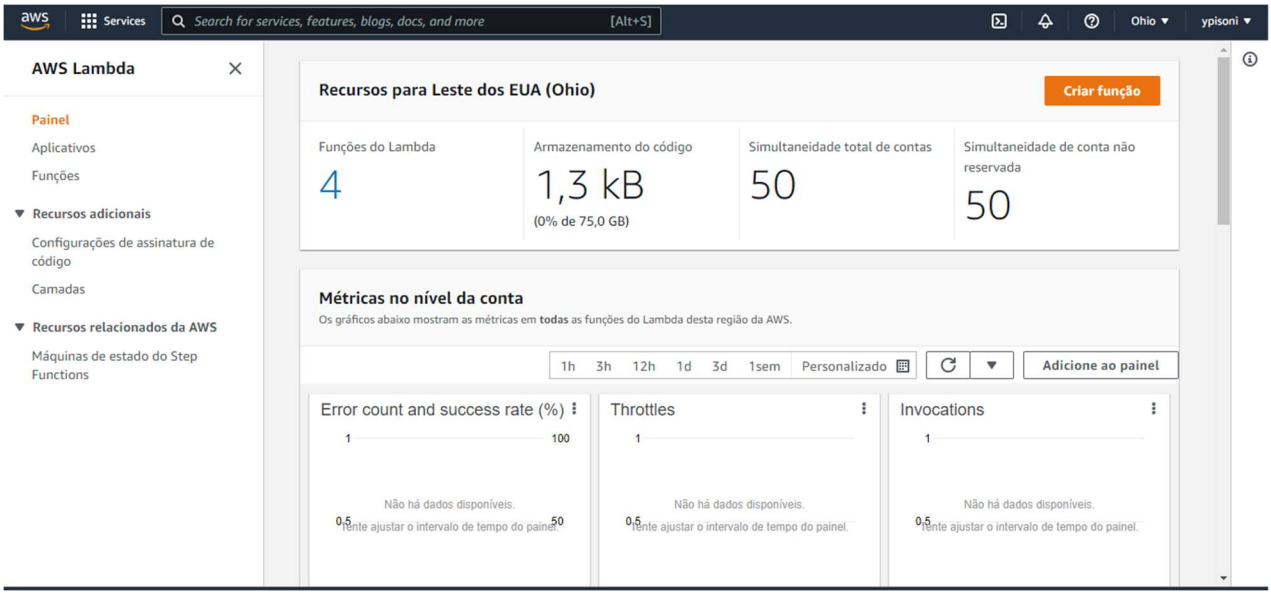


Figura 20. Painel do AWS Lambda (Visão do painel obtida pela autora, 2021)

O tempo utilizado entre a criação de uma conta até o desenvolvimento teve um total de 40 minutos, para cada função desenvolvida. Após estar com o login realizado, o tempo de duração foi em torno de 10 minutos; isso sendo o tempo de desenvolvimento e de execução.

Através do desenvolvimento realizado foi possível notar o quão rápida é a ferramenta a partir do início da criação da função até o momento da execução sendo que nesse caso a execução da função foi de 2.05 milissegundos.

5.CONSIDERAÇÕES FINAIS

Com a chegada da era digital, o mundo corporativo sofreu uma série de mudanças, e a sociedade como um todo. Diversas novidades tecnológicas começaram a fazer parte do cotidiano das pessoas e empresas. É importante reafirmar, que dentre esses avanços, o investimento em infraestrutura em nuvem chegou para criar um diferencial no âmbito empresarial, as soluções de armazenamento sem servidor se mostram cada vez mais estratégicas e possuem resultados mais eficientes. A partir dessas análises foi possível notar que o grande desafio das plataformas, nesse sentido, é alcançar o máximo possível de empresas, pois, a tecnologia se renova constantemente e o que ontem era novidade, hoje se torna obsoleto. Com isso fica claro, quando se faz uma avaliação sobre as empresas renomadas, como a Amazon, que está no 'ranking' das mais rentáveis e pioneira em armazenamento em nuvem.

Os recursos oferecidos pela Amazon Web Services Lambda, são capazes de atingir uma grande gama de empresas, pois além de ser mais rentável, devido a sua forma de pagamento que é realizada através do '*Pay-Per-Use'* , ou seja, as empresas pagam apenas pelas tarefas executadas, sendo esses valores relativamente baixos em relação a empresas que fazem as cobranças a partir de pacotes pré-estabelecidos além de possuir vários tutoriais e micro serviços gratuitos, sendo o formato gratuito o utilizado para realizar o desenvolvimento de uma função.

Com o Lambda as empresas não necessitam de diversos profissionais focados em hardware, uma vez que os servidores digitais não estarão mais presentes. As empresas conseguirão executar as suas funções de maneira rápida, visto que o tempo limite de execução do Lambda é de quinze minutos, porém a maioria das execuções levam apenas milissegundos para serem concluídas.

Sendo notável que através das análises, o uso desses recursos torna-se bastante eficaz, pode-se gerar grandes impactos positivos para as grandes e pequenas empresas, pois ao aderirem aos recursos tecnológicos da AWS Lambda, estarão obtendo a melhor forma de minimizar tarefas e custos e de maximizar os lucros.

REFERÊNCIAS BIBLIOGRÁFICAS

ADZIC, Gojko; CHATLEY, Robert. Serverless Computing: Economic and Architectural Impact. p. 01-06, 2017. Disponível em: https://www.doc.ic.ac.uk/~rbc/papers/fse-serverless-17.pdf. Acesso em: 6 set. 2021.

AWS. Otimização da economia empresarial com arquiteturas sem servidor. 2017. Disponível em: https://d1.awsstatic.com/whitepapers/pt_BR/optimizing-enterprise-economics-serverless-architectures.pdf. Acesso em: 26 nov. 2021.

AWS, AWS Lambda, 2021. Disponível em: https://aws.amazon.com/pt/lambda/. Acesso em: 27 out. 2021.

AWS, LAMBDA. Execute um aplicativo "Hello, World!" sem servidor. (2021) Disponível em: https://aws.amazon.com/pt/getting-started/hands-on/run-serverless-code/. Acesso em: 16=5 nov. 2021.

AWS, LAMBDA. Introdução AWS Lambda. (2021) Disponível em: https://docs.aws.amazon.com/lambda/latest/operatorguide/intro.html. Acesso em: 10 ago. 2021.

AWS, LAMBDA. AWS Lambda execution environment. (2021) Disponível em: https://docs.aws.amazon.com/lambda/latest/dg/runtimes-context.html. Acesso em: 12 nov. 2021.

CHOUDHARY, Brijesh et al. Case Study: Use of AWS Lambda for Building a Serverless Chat Application. O que é Serverless Computing? Entenda mais! Disponível em: https://www.researchgate.net/publication/338391132_Case_Study_Use_of_AWS_Lambda_for_Building_a_Serverless_Chat_Application. Acesso em: 06 nov. 2021.

CLEANCLOUND, FaaS: Um novo conceito de arquitetura sem servidor. 2018. Disponível em: https://cleancloud.io/faas-um-novo-conceito-de-arquitetura-sem-servidor/. Acesso em: 29 nov. 2021.

FEDAK, Vladmir. Serverless Applications with AWS Lambda: 5 Use Cases. 13 ago. 2018. Disponível em: https://dzone.com/articles/serverless-applications-with-aws-lambda-5-use-case. Acesso em: 8 out. 2021.

HOSSEINI, MEHRSHAD et al. AWS Lambda Language Performance. 12 nov. 2019. Disponível em: https://gupea.ub.gu.se/handle/2077/62454. Acesso em: 26 nov. 2021.

IBGE, Uso de internet, televisão e celular no Brasil. Disponível em: https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html. Acesso em: 23 nov. 2021.

IBM, Cloud Education. What is Serverless Computing? (2021). Disponível em: https://www.ibm.com/cloud/learn/serverless. Acesso em: 10 nov. 2021.

IPENSE, O que é Serverless Computing? Entenda mais! (2020). Disponível em: https://www.ipsense.com.br/blog/o-que-e-serverless-computing-entenda-mais/. Acesso em: 16 nov. 2021.

KLIMOVIC, Ana et al. Pocket: Elastic Ephemeral Storage for Serverless Analytics. In: Pocket: Elastic Ephemeral Storage for Serverless Analytics. (2020). Disponível em: https://www.usenix.org/conference/osdi18/presentation/klimovic. Acesso em: 21 out. 2021.

KOHN , Stephanie. O que é e para quem é indicado a Serverless Computing. [S. l.], 8 ago. 2018. Disponível em: https://canaltech.com.br/infra/o-que-e-e-para-quem-e-indicado-a-serverless-computing-119782/. Acesso em: 19 nov. 2021.

LAMBDA CONSOLE, AWS. Console AWS Lambda. In: LAMBDA CONSOLE. (2021) Disponível em: https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/discover. Acesso em: 16 nov. 2021.

MACHADO, Gabriel. O que é AWS Lambda?. 2020. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-aws-lambda. Acesso em: 27 nov. 2021.

MELL, P. and GRACE, T. (2011) The NIST Definition of Cloud Computing. National Institute of Standards and Technology Special Publication, 53, 1-7. Disponível em: https://www.scirp.org/(S(i43dyn45teexjx455qlt3d2q))/reference/ReferencesPapers.aspx?ReferenceID=1788248. Acesso em: 27 de agosto de 2021.

MULLER , Lisa et al. A break in theA Traffic Analysis on Serverless Computing Based on the Example of a File Upload Stream on AWS Lambda clouds: towards a cloud definition. MDPI, p. 01-29, 10 dez. 2020. Disponível em: https://www.mdpi.com/2504-2289/4/4/38. Acesso em: 29 out. 2021.

OLIVEIRA, Alfredo. Protegendo Pontos Fracos em Arquiteturas Serverless: Riscos e Recomendações. Trend Micro Research, [S. l.], p. 1-32, 8 ago. 2020. Disponível em: https://canaltech.com.br/infra/o-que-e-e-para-quem-e-indicado-a-serverless-computing-119782/. Acesso em: 19 nov. 2021.

SREEKANTI, Vikram et al. Cloudburst: Stateful Functions-as-a-Service. 14 jan. 2020. Disponível em: https://arxiv.org/abs/2001.04592. Acesso em: 20 nov. 2021.

SOFTLINE, IaaS, PaaS e SaaS: entenda os modelos de nuvem e suas finalidades. In: SOFTLINE, Softline. IaaS, PaaS e SaaS: entenda os modelos de nuvem e suas finalidades. 31 out. 2017. Disponível em: https://brasil.softlinegroup.com/sobre-a-empresa/blog/iaas-paas-saas-nuvem. Acesso em: 15 out. 2021.

SRAVAN KUMAR, Nunna et al. Monitoring and Handling the Errors in Serverless Applications Using AWS Lambda. A JOURNAL OF COMPOSITION THEORY, p. 1-8, 1 jan. 2020. Disponível em: http://www.jctjournal.com/gallery/24-april-2020.pdf. Acesso em: 20 out. 2021.

VAQUERO, Luis M et al. A break in the clouds: towards a cloud definition. ACM DIGITAL LIBRARY, p. 01-06, 31 dez. 2008. Disponível em: https://dl.acm.org/doi/abs/10.1145/1496091.1496100. Acesso em: 27 ago. 2021.

AGRADECIMENTOS

Em primeiro lugar, a Deus. Ao meu orientador do trabalho de conclusão de curso: Prof. Dr. Carlos Eduardo Câmara, por compartilhar os seus conhecimentos. E aos meus familiares e amigos, que sempre me apoiaram no decorrer do curso e me mantiveram determinada.