

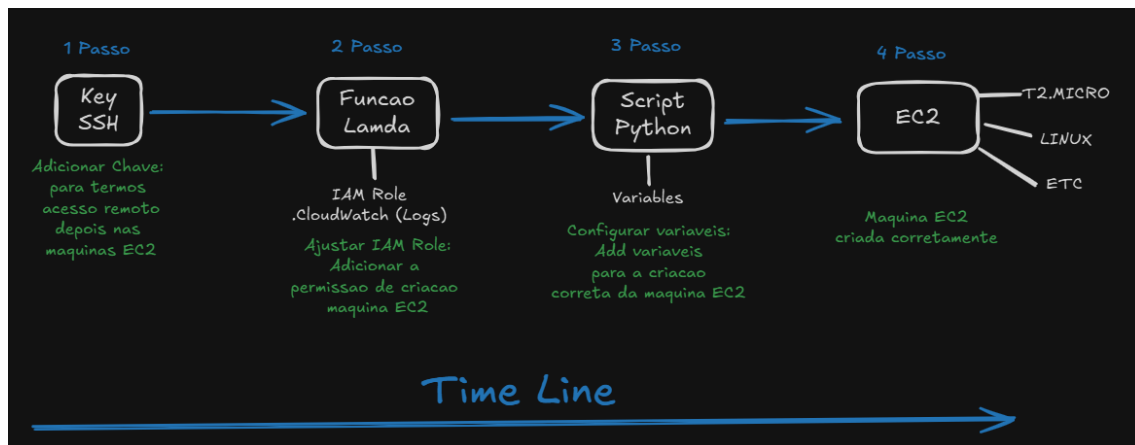
# Criação Máquina EC2

## Apresentação Projeto

Nesse projeto, iremos automatizar a criação de uma máquina (servidor) **EC2**!! Pois em alguns momentos, a criação dessas máquinas pode ser bem demorada e cansativa!!

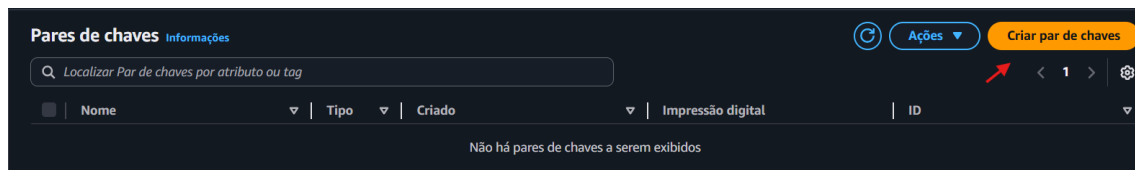
Dito isso, vamos criar um **script Python**, onde através de um botão será criado toda a **infraestrutura** de uma máquina **EC2**!!

## Time-Line Projeto



## Criacao Par Chave (SSH)

Seguindo a nosso **time line** do projeto, vamos iniciar criando um **par de chave SSH**, para que possamos nos conectar com as máquinas **EC2** que forem criadas!



**Criar par de chaves** Informações

**Par de chaves**  
Um par de chaves, que consiste em uma chave privada e uma chave pública, é um conjunto de credenciais de segurança que você usa para provar sua identidade.

**Nome**  
ec2-script  
O nome pode incluir até 255 caracteres ASCII. Ele não pode incluir espaços iniciais ou finais.

**Tipo de par de chaves** Informações  
☒ RSA ☐ ED25519

**Formato de arquivo de chave privada**  
☐ .pem Para uso com OpenSSH  
☒ .ppk Para uso com PuTTY

**Tags — opcional**  
Nenhuma tag associada ao recurso.  
[Adicionar nova tag](#)  
Você pode adicionar até mais 50 etiquetas.

## Criacao da Funcao Lambda

Após de criar um par de chave **SSH** para nos conectar remotamente com as máquinas **EC2**, vamos criar a nossa **função lambda**, onde será responsável em executar um **script Python!!**

E no cadastro da criação da nova **função lambda**, segue as informações básicas nele:

**Criar função** Informações

Escolha uma das opções a seguir para criar a função.

- ☒ **Criar do zero**  
Comece com um simples exemplo de Hello World.
- ☐ **Usar um esquema**  
Crie um aplicativo do Lambda a partir do código de exemplo e de predefinições de configuração para casos de uso comum.
- ☐ **Usar uma função existente**  
Crie um aplicativo do Lambda a partir de uma função existente.

**Informações básicas**

**Nome da função**  
Insira um nome que descreva o propósito da função.  
  
O nome da função deve ter de 1 a 64 caracteres, deve ser exclusivo para a região e não pode incluir espaços. Os caracteres válidos são a-z, A-Z, 0-9, hifens (-) e sublinhados (\_).

**Tempo de execução** Informações  
Escolha o idioma a ser usado para escrever sua função. Observe que o editor de código do console suporta apenas node.js, python e ruby.

**Arquitetura** Informações  
Escolha a arquitetura do conjunto de instruções desejada para o código da função.  
☐ arm64  
☒ x86\_64

- **Nome da função: Create-EC2-Linux**
  - Refere-se a criação de máquinas EC2 Linux.
- **Tempo de execução: Python 3.13**
  - Linguagem de programação que a função lambda irá executar em seu script.

E antes de criar a função lambda em si, veja um alerta importante referente as permissões quando criamos **funções Lambda**, dizendo que por padrão teremos apenas as permissões de **logs** do serviço **CloudWatch!**

**Permissões** Informações

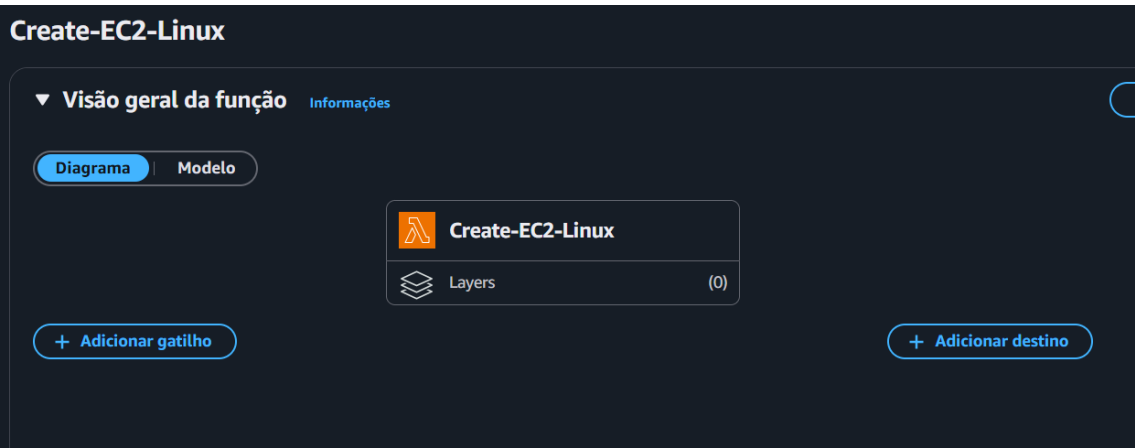
Por padrão, o Lambda cria uma função de execução com permissões para fazer upload de logs para o Amazon CloudWatch Logs. Você pode personalizar essa função padrão posteriormente ao adicionar triggers.

▼ **Alterar a função de execução padrão**

**Papel de execução**  
Escolha uma função que defina as permissões da sua função. Para criar uma função personalizada, acesse o [console do IAM](#).

- ☒ Criar uma função com permissões básicas do Lambda
- ☐ Usar uma função existente
- ☐ Criar uma função a partir da política da AWS templates

Sabendo disso, vamos enfim confirmar a criação da nossa função lambda **Create-EC2-Linux**!



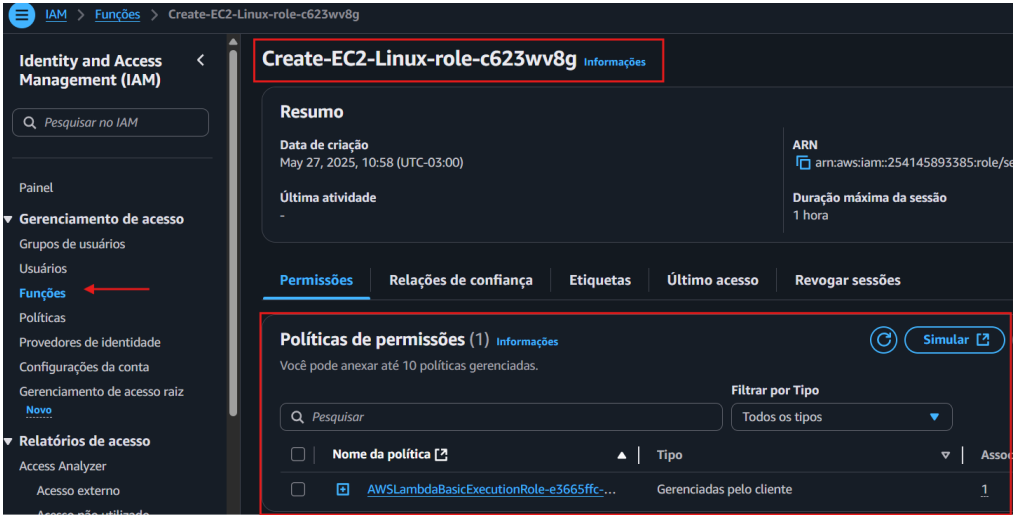
### Ajustando role de permissões da funcao lambda

E como foi dito anteriormente, a nossa função lambda só tem permissão em realizar upload de logs no **CloudWatch**!

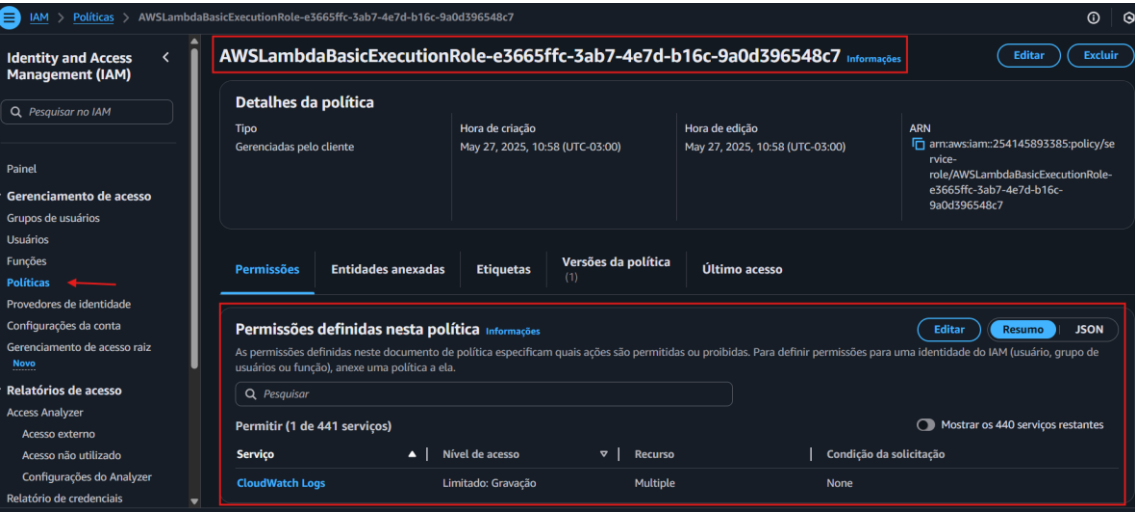


Logo precisamos ajustar sua **IAM Role** das permissões dessa **função lambda**, para que seja permitido também a criação de máquinas **EC2**!

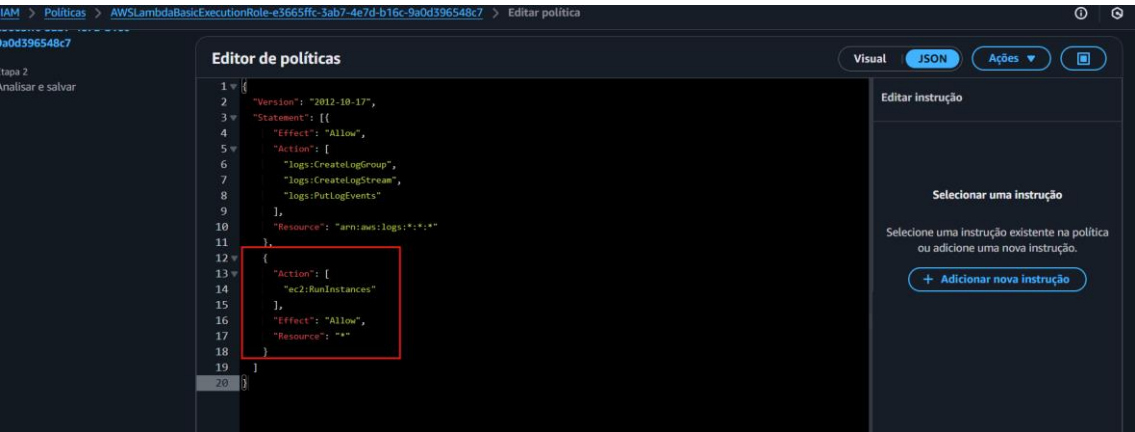
Sabendo disso, vamos até o serviço de **IAM** referente as nossas funções, e veremos que temos uma **role** criada com o mesmo **nome da função** que criamos concatenado com um **ID**.



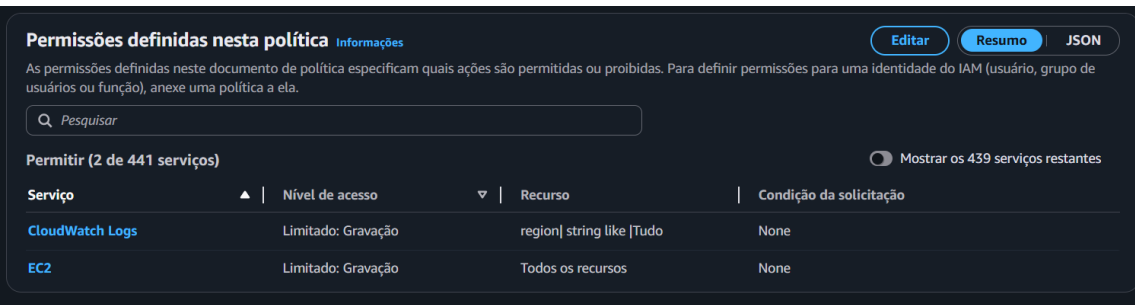
Ao entrar no **IAM Role** da função veremos também as políticas associadas a ela, inclusive ao entrar nessa política, veremos que no momento, só temos permissão ao serviço do **CloudWatch**!



Sabendo disso, vamos enfim alterar essa política, para que seja possível além da permissão do serviço do **CloudWatch**, que também tenha **permissão** na utilização do serviço de **EC2**!!



Pronto, agora podemos ver que a Role da nossa função lambda terá acesso aos serviços do **CloudWatch** e **EC2**!



## Criação script função lambda

Enfim vamos criar nosso script **Python** para que possamos automatizar todo o processo de criação de uma máquina **EC2**!!

Primeiramente precisamos importar essas bibliotecas:

```
lambda_function.py X
lambda_function.py
1 import os
2 import boto3
```

- **os**: biblioteca onde iremos recuperar as **variáveis de ambiente**, referente as informações necessárias que serão passadas para a criação de uma máquina **EC2**.
- **boto3**: biblioteca **SDK oficial da AWS**, onde temos acesso total nas integrações aos serviços da própria **AWS**.

Após a importação das bibliotecas, vamos recuperar os dados referente as **variáveis de ambiente** que serão necessários para a criação da máquina **EC2**.

```
AMI = os.environ['AMI']
INSTANCE_TYPE = os.environ['INSTANCE_TYPE']
KEY_NAME = os.environ['KEY_NAME']
SUBNET_ID = os.environ['SUBNET_ID']
```

Após a recuperação dos dados referente as variáveis de ambiente, precisamos criar o **objeto** referente ao serviço do **EC2** pela biblioteca **boto3**, que nesse caso usamos a função **resource()**, passando o nome do serviço (**'ec2'**) como **parâmetro**!

```
ec2 = boto3.resource('ec2')
```

E para finalizar, definiremos a função em si onde será executado o nosso algoritmo de criar a máquina **EC2**, no momento do **evento** da **função lambda**!! E quando esse evento da função lambda for invocado, será executado a função **lambda\_handler(event, context)**

```
def lambda_handler(event, context):
    instance_ec2 = ec2.create_instances(
        ImageId=AMI,
        InstanceType=INSTANCE_TYPE,
        KeyName=KEY_NAME,
        SubnetId=SUBNET_ID,
        MaxCount=1,
        MinCount=1
    )

    print("New instance created: ", instance_ec2[0].id)
```

E pronto!! Nosso **script python** está finalizado!!

```
lambda_function.py X
lambda_function.py
1  import os
2  import boto3
3
4  AMI = os.environ['AMI']
5  INSTANCE_TYPE = os.environ['INSTANCE_TYPE']
6  KEY_NAME = os.environ['KEY_NAME']
7  SUBNET_ID = os.environ['SUBNET_ID']
8
9  ec2 = boto3.resource('ec2')
10
11 def lambda_handler(event, context):
12     instance_ec2 = ec2.create_instances(
13         ImageId=AMI,
14         InstanceType=INSTANCE_TYPE,
15         KeyName=KEY_NAME,
16         SubnetId=SUBNET_ID,
17         MaxCount=1,
18         MinCount=1
19     )
20
21     print("New instance created: ", instance_ec2[0].id)
```

## Configurando variáveis de ambiente

Vamos agora configurar os valores das **variáveis de ambiente** citado no **script** da **função lambda** citado acima!



**Editar variáveis de ambiente**

**Variáveis de ambiente**  
Você pode definir as variáveis de ambiente como pares de chave-valor acessíveis a partir do código de função. Elas são úteis para armazenar parâmetros de configuração sem precisar alterar o código de função. Saiba mais

Chave	Valor	
AMI	ami-0953476d60561c955	Remove
INSTANCE_TYPE	t2.micro	Remove
KEY_NAME	ec2-script	Remove
SUBNET_ID	subnet-0455f3d668097a2a5	Remove

Adicionar variáveis de ambiente

Configuração da criptografia

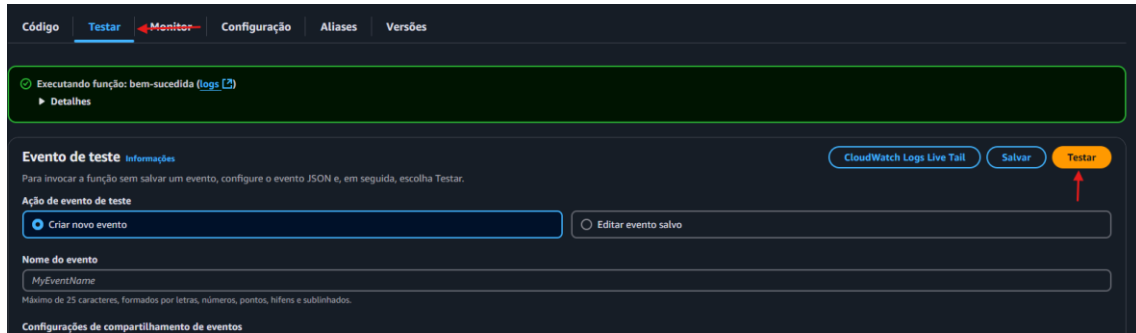
Cancelar Salvar

- **AMI:** Imagem do sistema operacional da máquina virtual, que nesse caso será um **sistema operacional Linux**.
- **INSTANCE\_TYPE:** Tipo da instancia EC2, que nesse caso será **t2.micro**, por ser mais barato (gratuito).
- **KEY\_NAME:** Nome da **chave SSH** que estamos vinculando com a máquina **EC2**, para que possamos ter conexão remotamente após a criação dela.
- **SUBNET\_ID:** Sub-rede virtual onde será criada a nossa máquina **EC2**.

## Criando máquina EC2

Enfim vamos testar a criação da nossa máquina **EC2** através da execução da nossa **função lambda**!!

E para realizar esse teste é muito simples, só precisamos ir ao menu **Testar**, e no **evento de teste** clicar em testar.



E pronto!! Podemos ver nas nossas máquinas **EC2** que já temos uma máquina que foi criada corretamente!!

