

EC2 Backup

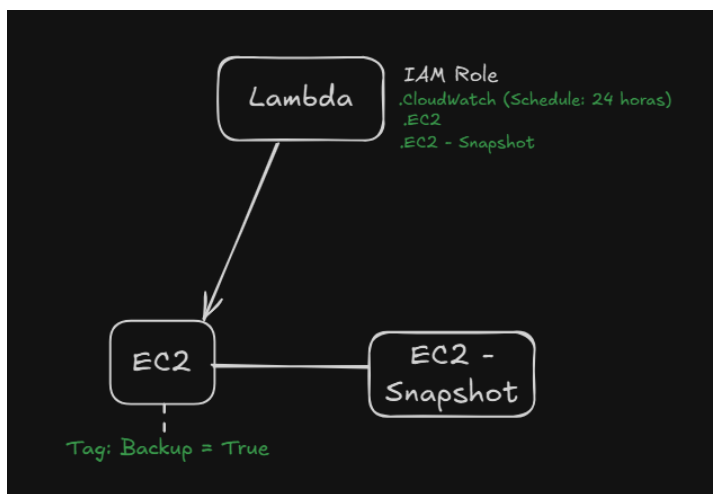
Apresentação Projeto

Nesse projeto, iremos automatizar os **backups** referente as nossas maquinas **EC2**!!

Pois em alguns momentos, precisamos realizar **backups** em nossas maquinas **EC2** para que seja possviel **restaurar-las** quando der algum tipo de problema!!

Dito isso, vamos automatizar de forma que a cada **24 horas** seja realziado **backups**, ou conhecido como **snapshot**, em nossas maquinas **EC2**!

Time line Projeto



Maquinas EC2

Primeiramente, temos que ter maquinas **EC2** em nosso **ambiente AWS**!! Sabendo disso, foi criado uma maquina **EC2** para teste, como podemos ver!!

A captura de tela mostra a interface de gerenciamento de instâncias EC2 na AWS. No topo, há uma barra de ferramentas com botões como 'Conectar', 'Estado da instância', 'Ações' e 'Executar instâncias'. Abaixo, há uma tabela com as seguintes colunas: 'Name', 'ID da instância', 'Estado da inst...', 'Tipo de inst...', 'Verificação de stat', 'Status do alarm', 'Zona de dispon...', 'DNS IPv4 público', 'Endereço IP...', 'IP elástico'. A única instância listada é 'EC2_Test' com ID 'i-015107d9a843ca052', estado 'Executando', tipo 't2.micro', status de alarm 'Inicializando', zona de disponibilidade 'us-east-1d', DNS IPv4 público 'ec2-3-95-184-81.comp...', endereço IP '3.95.184.81' e IP elástico '-'. Abaixo da tabela, há uma barra de status com o texto 'Executando' e um ícone de lupa.

E um detalhe importante, precisamos adicionar uma tag para a nossa maquina **EC2**!! Pois a nossa **automatizacao** so sera feita para essas maquinas **EC2** que tiverem essa **tag**, para que nao seja necessario realizar o **backup** em todas as maquinas, mas sim so nas mais importantes!!

A captura de tela mostra a interface de gerenciamento de tags na AWS. No topo, há uma barra de ferramentas com o botão 'Adicionar nova tag'. Abaixo, há uma seção 'Gerenciar tags' com o texto 'Uma tag é um rótulo personalizado que você atribui a um recurso da AWS. Você pode usar tags para organizar e identificar suas instâncias.' A interface é dividida em duas colunas: 'Chave' e 'Valor - opcional'. Na coluna 'Chave', há um campo de busca com o texto 'Q Name' e um botão 'Remover'. Na coluna 'Valor - opcional', há um campo de busca com o texto 'Q EC2_Test' e um botão 'Remover'. Abaixo, há um campo de busca com o texto 'Q Backup' e um botão 'Remover'. Um botão 'Adicionar nova tag' está na base da interface. No rodapé, há o texto 'Você pode adicionar até mais 48 etiquetas.'

Criando funcao lambda

Vamos agora criar a nossa **função lambda**!! Logo segue a criação de forma simples e direta, lembrando que sera uma **funcao lambda** utilizando **script Python**!!

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 de Lambda a partir do código de exemplo e de definições de configuração para casos de uso comum.
- ☐ **Imagem de contêiner**
Selecione uma imagem de contêiner a ser implantada para sua função.

Informações básicas

Nome da função
Insira um nome que descreva o propósito da função.
EC2-Snapshot
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, hífen (-) e sublinhados (_).

Tempo de execução Informações
Escolha o idioma a ser usado para escrever sua função. Observe que o editor de código da console suporta apenas node.js, python e ruby.
Python 3.13

Após a criação da **função lambda**, como já sabemos, o **IAM Role** padrão dará permissão apenas para o serviço do **CloudWatch**, onde sera necessario para a nossa automatizacao, pois utilizaremos o **job** de **Schedule**, que sera configurado nele para que a nossa **funcao lambda** seja executada a cada **24 horas**!!

Alem do **CloudWatch**, precisaremos editar a política dessa role, adicionando as permissões referente ao serviço do **EC2** e **EC2 Snapshot** também!

Editor de políticas

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [{
4     "Effect": "Allow",
5     "Action": [
6       "logs:CreateLogGroup",
7       "logs:CreateLogStream",
8       "logs:PutLogEvents"
9     ],
10    "Resource": "arn:aws:logs:*:*:*"
11  },
12  {
13    "Effect": "Allow",
14    "Action": [
15      "ec2:CreateSnapshot",
16      "ec2:CreateTags",
17      "ec2>DeleteSnapshot",
18      "ec2:Describe*",
19      "ec2:ModifySnapshotAttribute",
20      "ec2:ResetSnapshotAttribute"
21    ],
22    "Resource": "*"
23  }
24 ]
25 }
```

Criacao script python

Emfim vamos criar o **script python** da nossa **funcao lambda**!! E Primeiramente vamos importar as bibliotecas necessarias, que serao o **boto3** e **datetime**.

```
lambda_function.py
1  from datetime import datetime
2  import boto3
```

Após a importacao, vamos criar a funcao principal `lambda_handler()`, e de cara já vamos criar o objeto **ec2** pela funcao `boto3.client('ec2')`!

```
5  def lambda_handler(event, context):
6      ec2_client = boto3.client('ec2')
```

Em seguida vamos realizar um **List Comprehension**, para que seja retornado a lista de todas as regioes existentes da **AWS** referente ao servico do **EC2** que estao **habilitadas** na sua **conta**, atraves da funcao `describe_regions()`, e conseqüentemente iterar nessa lista atraves de um **for loop**.

```
regions = [region['RegionName']] for region in ec2_client.describe_regions()['Regions']
for region in regions:
```

Agora buscar todas as instancias das maquinas **EC2**!!

Dito isso, vamos buscar todas as maquiains **EC2** atraves da funcao `resource()` definindo o parametro **region_name**. E depois realizaremos um filtro para trabalharmos apenas com as instancias com a **tag backup = True**.

```
for region in regions:
    print(f'Instances in EC2 Region {region}:')

    ec2 = boto3.resource('ec2', region_name=region)

    instances = ec2.instances.filter(
        Filters=[
            {'Name': 'Backup', 'Values': ['True']}
        ]
    )
```

Agora que temos a lista das instancias da regioao especifica, vamos criar um **for loop** atraves da funcao `all()` do objeto **instances**, para que possamos iterar por cada **instancia**, e de fato realizar o **snapshot** da **instancia especifica**!!

Mas antes de fato realizar o **snapshot** de cada **instancia** da regioao, vamos realizar um `print` informando a **instancia** e todos os **volume** que serao realizados no **snapshot**!!

```
timestamp = datetime.utcnow().replace(microsecond=0).isoformat()

for i in instances.all():
    for v in i.volumes.all():
        desc_snapshot = 'Backup of instance {i.id}, volume {v.id}, created {timestamp}'
        print(desc_snapshot)]
```

Veja que alem do **for loop** das **instancias**, foi realizado tambem um **for loop** de cada **volume** daquela instancia especifica!! Pois o **backup** do **snapshot** eh feito por cada **volume** de uma **instancia do EC2**!!

Dito isso, chegou a hora em realizar o snapshot do volume si!! E para isso utilizaremos a funcao `create_snapshot()` do objeto **volume (v)** que estamos iterando no **for loop**!

```
for v in i.volumes.all():
    desc_snapshot = 'Backup of instance {i.id}, volume {v.id}, created {timestamp}'
    print(desc_snapshot)

    snapshot = v.create_snapshot(Description=desc_snapshot)
    print("Created snapshot:", snapshot.id)
```

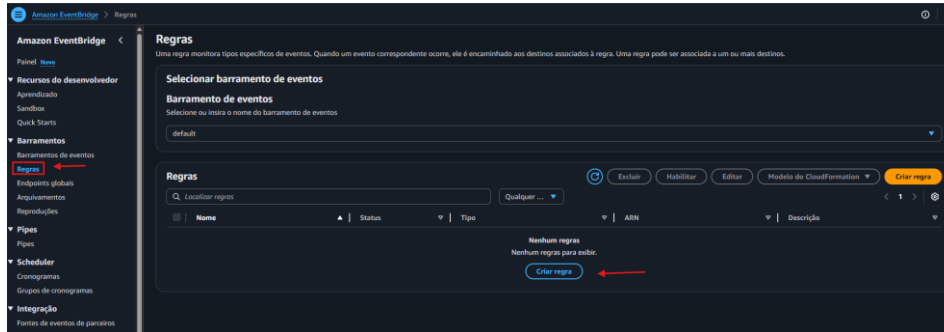
E pronto!! Nosso **script python** para automatizar o **snapshot** de todas as **instancias EC2** em todas as **regioes** da nossa conta **AWS** esta feito!!

```
lambda_function.py
1  from datetime import datetime
2  import boto3
3
4
5  def lambda_handler(event, context):
6      ec2_client = boto3.client('ec2')
7      regions = [region['RegionName']] for region in ec2_client.describe_regions()['Regions']
8
9      for region in regions:
10         print(f'Instances in EC2 Region {region}:')
11
12         ec2 = boto3.resource('ec2', region_name=region)
13
14         instances = ec2.instances.filter(
15             Filters=[
16                 {'Name': 'Backup', 'Values': ['True']}
17             ]
18         )
19
20         timestamp = datetime.utcnow().replace(microsecond=0).isoformat()
21
22         for i in instances.all():
23             for v in i.volumes.all():
24                 desc_snapshot = 'Backup of instance {i.id}, volume {v.id}, created {timestamp}'
25                 print(desc_snapshot)
26
27                 snapshot = v.create_snapshot(Description=desc_snapshot)
28                 print("Created snapshot:", snapshot.id)
```

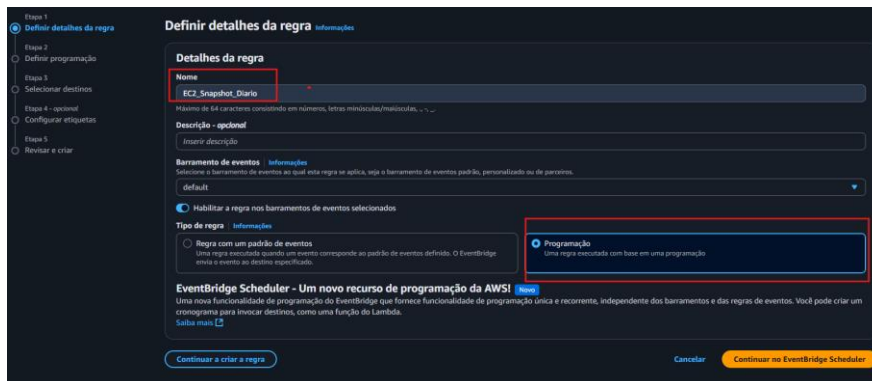
Configurando Schedule pelo Amazon EventBridge

E para finalizar, vamos agora configurar o **schedule** (agendamento) da execucao da nossa **funcao lambda**!! E para isso precisamos ir no servico do **EventBridge**!!

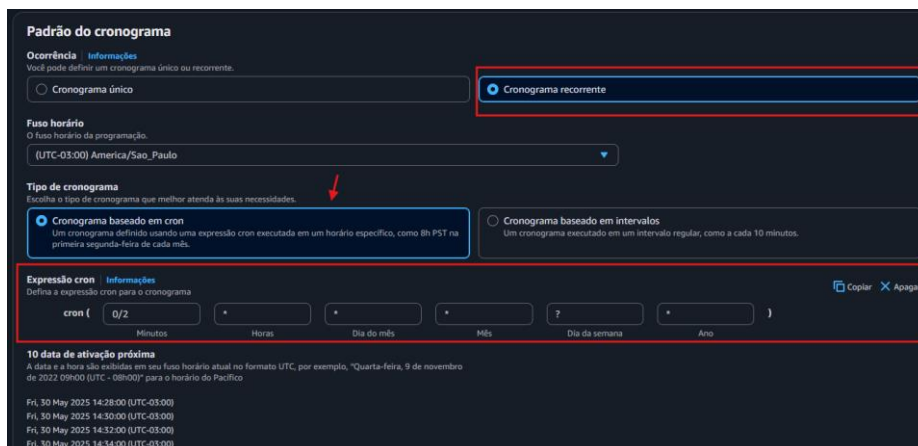
Ao abrir o servico, vamos no menu lateral em barramentos e solicitar para criar uma regra!! Sera nele que iremos criar nossa regra de **schedule**(agendamento) para executar a nossa funcao lambda!!



E para criar uma **regra**, devemos seguir uma serie de passos, e na **primeira etapa** basicamente definimos o **nome da regra** e o **tipo da regra**, que nesse caso o tipo sera **programacao**, já que devemos debir um **agendamento** para ser executado essa regra!!

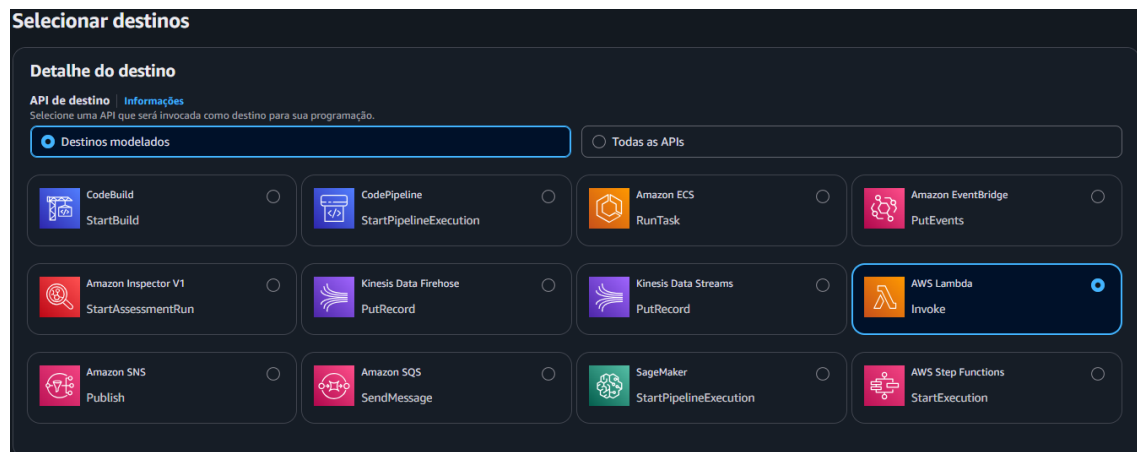


Já na proxima etapa basicamete sera onde definimos nossa expressao de agendamento **schedule**, conhecido como **cron**!!



Veja que para ajudar no nosso teste, foi utilizado uma expressão **cron**, onde será executado a cada **2 minutos!!** Porém a versão final correta seria a cada **24 horas!!**

Na próxima etapa enfim definimos qual **tipo de serviço** que será executado nesse **schedule!!** Como se trata de uma **função lambda**, será então selecionado a **função lambda** que **criamos anteriormente** para esse **vinculo** do **schedule!!**



E pronto!! Nossa **configuração do schedule** está realizada!!

Criação dos snapshots

E para finalizar, logo em seguida veremos que a execução da **função lambda** será feita pelo agendamento que realizamos do **schedule!!** Tanto que se formos no serviço do **EC2**, e irmos na **área de snapshots** das nossas instâncias, veremos que temos um novo **snapshot** realizado pelo nosso **script python** do **lambda!!**

