Simulando AWS com LocalStack e TerraForm

SIMULANDO AWS COM LOCALSTACK E TERRAFORM: GUIA COMPLETO
https://www.youtube.com/watch?v=0nU9yvqg2Rw&t=917s

Repo com os exemplo Terraform:
https://github.com/badtuxx/salvando-dinheiro

terraform-101
https://github.com/badtuxx/terraform-101

Docker Compose básico para subir container do LocalStack:

```
services:
  localstack:
    image: localstack/localstack:latest
    container_name: awslocal
    hostname: awslocal
    environment:
      - SERVICES=iam,lambda,apigateway,dynamodb
      - DEBUG=${DEBUG:-0}
      - LOCALSTACK_HOST=awslocal
    ports:
      - '4566-4597:4566-4597'
    networks:
      - my-network
    volumes:
      - "./.localstack:/var/lib/localstack"
      - "/var/run/docker.sock:/var/run/docker.sock"
networks:
  my-network:
    driver: bridge
```

Comando para executar docker compose
docker-compose up

O path aonde o docker-compose.yml foi salvo não pode ter " ","-" e "_" para rodar no Podman

Comandos terraform

- inicializa o terraform: `terraform init`
- exibe as alterações que serão realizadas: `terraform plan`
- aplica as alterações planejadas: `terraform apply -auto-approve`
- destrói toda a infraestrutura criada: `terraform destroy`

Comandos AWS

 # Testar LocalStack, exibindo informações do usuário

```
aws --endpoint http://localhost:4566 sts get-caller-identity
```

 # Resultado

```
{
   "UserId": "AKIAIOSFODNN7EXAMPLE",
   "Account": "000000000000",
   "Arn": "arn:aws:iam::000000000000:root"
}
```

 # Verificação da Infraestrutura - Liste os endpoints

```
aws apigateway get-rest-apis --endpoint-url=http://localhost:4566
```

 # Resultado

```
{
    "items": [
        {
            "id": "00jgzn4ohp",
            "name": "StreamLambdaHandlerBank",
            "description": "API Microsservice Bank",
            "createdDate": "2025-02-25T13:50:02-03:00",
            "apiKeySource": "HEADER",
```

```
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "disableExecuteApiEndpoint": false
        },
        {
            "id": "6h2lt41s4k",
            "name": "StreamLambdaHandlerFeedback",
            "description": "API Microsservice Buyfeedback",
            "createdDate": "2025-02-25T13:50:02-03:00",
            "apiKeySource": "HEADER",
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "disableExecuteApiEndpoint": false
        },
        {
            "id": "f6zak4vehc",
            "name": "StreamLambdaHandlerTrip",
            "description": "API Microsservice Buytrip",
            "createdDate": "2025-02-25T13:50:02-03:00",
            "apiKeySource": "HEADER",
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "disableExecuteApiEndpoint": false
        }
    ]
}
```


 # Confirme se o método da API Gateway está vinculado à Lambda

```
aws apigateway get-resources --rest-api-id 6h2lt41s4k --endpoint-url=http://localhost:4566
```

 # Resultado


```
{
"items": [
        {
            "id": "apfwojmbry",
            "path": "/"
        },
        {
            "id": "ictnkbg7np",
            "parentId": "apfwojmbry",
            "pathPart": "feedback",
            "path": "/feedback",
            "resourceMethods": {
                "GET": {
                    "httpMethod": "GET",
                    "authorizationType": "NONE",
                    "apiKeyRequired": false,
                    "methodResponses": {},
                    "methodIntegration": {
                        "type": "AWS_PROXY",
                        "httpMethod": "POST",
                        "uri":
"arn:aws:apigateway:sa-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:sa-east-
1:000000000000:function:StreamLambdaHandlerFeedback/invocations",
                        "connectionType": "INTERNET",
                        "passthroughBehavior": "WHEN_NO_MATCH",
                        "timeoutInMillis": 29000,
                        "cacheNamespace": "ictnkbg7np",
                        "cacheKeyParameters": []
```

```
                    }
                }
            }
        },
        {
            "id": "teenveqmyu",
            "parentId": "ictnkbg7np",
            "pathPart": "meunome",
            "path": "/feedback/meunome",
            "resourceMethods": {
                "POST": {
                    "httpMethod": "POST",
                    "authorizationType": "NONE",
                    "apiKeyRequired": false,
                    "methodResponses": {},
                    "methodIntegration": {
                        "type": "AWS_PROXY",
                        "httpMethod": "POST",
                        "uri":
"arn:aws:apigateway:sa-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:sa-east-
1:000000000000:function:StreamLambdaHandlerFeedback/invocations",
                        "connectionType": "INTERNET",
                        "passthroughBehavior": "WHEN_NO_MATCH",
                        "timeoutInMillis": 29000,
                        "cacheNamespace": "teenveqmyu",
                        "cacheKeyParameters": []
                    }
                }
            }
        },
        {
            "id": "4kv82wtljo",
            "parentId": "ictnkbg7np",
            "pathPart": "{id}",
            "path": "/feedback/{id}",
            "resourceMethods": {
                "DELETE": {
                    "httpMethod": "DELETE",
                    "authorizationType": "NONE",
                    "apiKeyRequired": false,
                    "methodResponses": {},
                    "methodIntegration": {
                        "type": "AWS_PROXY",
                        "httpMethod": "POST",
                        "uri":
"arn:aws:apigateway:sa-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:sa-east-
1:000000000000:function:StreamLambdaHandlerFeedback/invocations",
                        "connectionType": "INTERNET",
                        "passthroughBehavior": "WHEN_NO_MATCH",
                        "timeoutInMillis": 29000,
                        "cacheNamespace": "4kv82wtljo",
                        "cacheKeyParameters": []
                    }
                },
                "GET": {
                    "httpMethod": "GET",
                    "authorizationType": "NONE",
                    "apiKeyRequired": false,
                    "methodResponses": {},
                    "methodIntegration": {
                        "type": "AWS_PROXY",
                        "httpMethod": "POST",
                        "uri":
"arn:aws:apigateway:sa-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:sa-east-
1:000000000000:function:StreamLambdaHandlerFeedback/invocations",
                        "connectionType": "INTERNET",
                        "passthroughBehavior": "WHEN_NO_MATCH",
```

```
                        "timeoutInMillis": 29000,
                        "cacheNamespace": "4kv82wtljo",
                        "cacheKeyParameters": []
                    }
                }
            }
        }
    ]
}
```

 # Listar buckets aws
```
aws s3 ls --profile localstack
```

 # Criar um bucket
```
aws s3 mb s3://lduran-bucket --profile localstack
```

 # Copia um arquivo para o seu bucket
```
aws s3 cp testes.txt s3://lduran-bucket --profile localstack
```

 # Listar o conteúdo de um bucket
```
aws s3 ls s3://lduran-bucket/ --profile localstack
```

 # Listar funções Lambda
```
aws lambda list-functions --endpoint-url=http://localhost:4566
```

 # Listar APIs no API Gateway
```
aws apigateway get-rest-apis --endpoint-url=http://localhost:4566
```

Comandos para testar o DynamoDB:

Testar a Tabela DynamoDB

 # Cria Tabela Books
```
aws --endpoint-url=http://localhost:4566 --profile localstack dynamodb create-table --table-name Books --attribute-definitions AttributeName=id,AttributeType=S --key-schema AttributeName=id,KeyType=HASH --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5
```

 # Resultado
```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "id",
                "AttributeType": "S"
            }
        ],
        "TableName": "Books",
        "KeySchema": [
            {
                "AttributeName": "id",
                "KeyType": "HASH"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": "2025-03-08T09:07:54.642000-03:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:sa-east-1:000000000000:table/Books",
        "TableId": "2cca172b-1248-43f3-b182-4706cecceb73"
    }
}
```

 # Listar tabelas do DynamoDB / Verifica a Tabela Criada
```
aws --endpoint-url=http://localhost:4566 --profile localstack dynamodb list-tables
```

 # Resultado

```
{
    "TableNames": [
        "Books"
    ]
}
```

 # Adicionar um item ao DynamoDB (Deve ser executado no Git Bash)

```
aws dynamodb put-item \
  --table-name Books \
  --item '{"id": {"S": "1"}, "title": {"S": "Test Book"}, "author": {"S": "Author Name"}}' \
  --endpoint-url=http://localhost:4566
```

 # Listar os itens do DynamoDB

```
aws dynamodb scan --table-name Books --endpoint-url=http://localhost:4566
```

 # Resultado

```
{
    "Items": [
        {
            "title": {
                "S": "Test Book"
            },
            "author": {
                "S": "Author Name"
            },
            "id": {
                "S": "1"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

 # Recuperar um item específico do DynamoDB (Deve ser executado no Git Bash)

```
aws dynamodb get-item \
  --table-name Books \
  --key '{"id": {"S": "1"}}' \
  --endpoint-url=http://localhost:4566
```

 # Resultado

```
{
    "Item": {
        "title": {
            "S": "Test Book"
        },
        "author": {
            "S": "Author Name"
        },
        "id": {
            "S": "1"
        }
    }
}
```

 # Excluir um item do DynamoDB (Deve ser executado no Git Bash)

```
aws --endpoint-url=http://localhost:4566 --profile localstack dynamodb delete-item --table-
name Books --key '{"id": {"S": "1"}}'
```

 # Apaga tabela Books

```
aws --endpoint-url=http://localhost:4566 --profile localstack dynamodb delete-table --
table-name  Books
```

Testar as Funções Lambda

 # Invocar a função Lambda para listar livros

```
aws lambda invoke \
  --function-name BookFunction \
```

```
  --payload '{"httpMethod": "GET", "path": "/book"}' \
  --cli-binary-format raw-in-base64-out \
  response.json \
  --endpoint-url=http://localhost:4566
```

 # Invocar a função Lambda para criar um livro

```
aws lambda invoke \
  --function-name BookFunction \
  --payload '{"httpMethod": "POST", "path": "/book", "body":
"{\"id\": \"2\", \"title\": \"New Book\", \"author\": \"New Author\"}"}' \
  --cli-binary-format raw-in-base64-out \
  response.json \
  --endpoint-url=http://localhost:4566
```

Testar o API Gateway

Para testar o API Gateway, você pode usar uma ferramenta como curl, Postman ou até mesmo a AWS CLI para enviar requisições HTTP para os endpoints criados.

Substitua {api-id} pelo ID da API retornado no passo de "# Verificação da Infraestrutura".

O api-id deve ser pego desta linha da saída do script do TerraForm: aws_api_gateway_resource.book_id: Creation complete after 0s [id=oty6yjwvyw]

 # Listar todos os livros

```
curl -X GET "http://localhost:4566/restapis/{api-id}/prod/_user_request_/book"
```

 # Buscar um livro específico por ID

```
curl -X GET "http://localhost:4566/restapis/{api-id}/prod/_user_request_/book/{id}"
```

Testar SQS e SNS

 # Criando Queue(Standard) no SQS do LocalStack...

```
aws --endpoint http://localhost:4566 --profile localstack sqs create-queue --queue-name sqsPassagens
```

 # Resultado

```
{
    "QueueUrl": "http://sqs.sa-east-1.awslocal:4566/000000000000/sqsPassagens"
}
```

 # Criando Queue(Standard) no SNS do LocalStack...

```
aws --endpoint http://localhost:4566 --profile localstack sns create-topic --name
snsFilaComprasFinalizado
```

 # Subscreve o topico snsFilaComprasFinalizado a fila sqsPassagens do LocalStack...

```
aws --endpoint http://localhost:4566 --profile localstack sns subscribe --topic-arn
arn:aws:sns:sa-east-1:000000000000:snsFilaComprasFinalizado --protocol sqs --notification-
endpoint arn:aws:sqs:sa-east-1:000000000000:sqsPassagens
```

 # Envia Mensagem para a Fila

```
aws --endpoint-url=http://localhost:4566 --profile localstack sqs send-message --queue-
url=http://localhost:4566/000000000000/sqsPassagens --message-body "{'id': '123', 'content':
'Test message'}"
```

 # Resultado

```
{
    "MD5OfMessageBody": "c590f6ae4f1213a35bf0674907bc31ba",
    "MessageId": "72d7c13d-8f0e-4e05-b884-3b7f6fac961f"
}
```

 # Verifica Sucesso no Envio da Mensagem

```
aws --endpoint-url=http://localhost:4566 --profile localstack sqs receive-message --queue-
url=http://sqs.sa-east-1.awslocal:4566/000000000000/sqsPassagens
```

 # Resultado

```
{
    "Messages": [
        {
            "MessageId": "72d7c13d-8f0e-4e05-b884-3b7f6fac961f",
            "ReceiptHandle":
"ODg0MWE3NjQtNjBjOC00YTc1LWEwMDYtNWQwMTkzMDg4OThiIGFybjphd3M6c3FzOnNhLWVhc3QtMTowMDAwMDAwMDAwMDA6c3FzU
GFzc2FnZW5zIDcyZDdjMTNkLThmMGUtNGUwNS1iODg0LTNiN2Y2ZmFjOTYxZiAxNzQxNDQ1MzU5LjMzNTk4MTY=",
```

```
            "MD5OfBody": "c590f6ae4f1213a35bf0674907bc31ba",
            "Body": "{'id': '123', 'content': 'Test message'}"
        }
    ]
}
```