Trabalho final infra como código

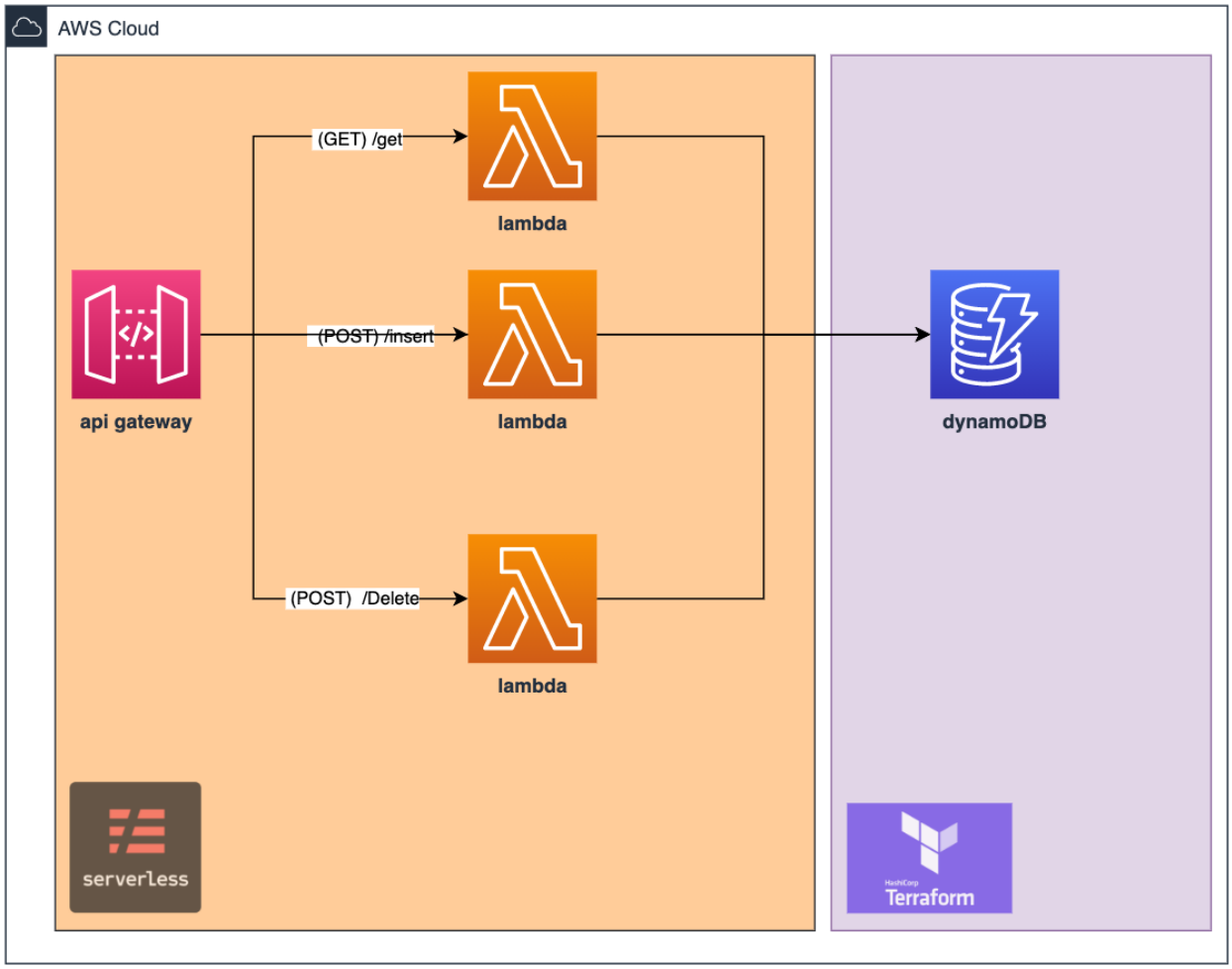
Este exemplo implementa uma API com 3 endpoints:
um get(Utilizando verbo GET)
um insert (Utilizando verbo POST)
um delete (Utilizando verbo POST)
O banco de dados usado é dynamoDB

A API lambdas será criada utilizando o serverless framework e o dynamoDB utilizando Terrafrom

Estrutura

Está separada em dois diretório:
o 'terraform-dynamoDB' para criação do Banco de dados
o 'api-gateway-lambda-serverless' para criação das apis

```
ps. A instância e acesso à AWS usando o Cloud9 ou acessando EC2 deve está configurada.
```



Caso de uso

- •API para Aplicação Web
- •API para Aplicação Mobile
- •Implementa um cadastro simples com inserção , busca de um item e exclusão de um item

Setup

Instalação do Terraform

```
sudo apt-get install unzip -y

curl https://releases.hashicorp.com/terraform/1.0.4/terraform_1.0.4_linux_amd64.zip

unzip terraform*

sudo mv terraform /usr/local/bin/
```

Instalação do serveless framework

```
npm install -g serverless
```

Deploy

Ordem para o deploy:
1 - Execute o comando `git clone https://github.com/fredux/trabalho-api-lambda.git`

2 - Execute o comando `cd trabalho-api-lambda/` para entrar na pasta criada pelo git
3 - Execute o comando `cd terraform-dynamoDB/` para entrar na pasta com os scripts de Configuração do dynamoDB.
4 -Execute o comando `terraform init`
5 - Execute um `terraform plan` para ver o que será executado.
6 - Execute um `terraform apply -auto-approve` para que sejam criados os recursos na AWS
7 - Execute o comando `cd ..` para entrar na pasta
8 - Execute o comando `cd api-gateway-lambda-serverless/` para entrar na pasta
9 - Execute o comando `sls deploy` para deploy

```

```

```
O resultado no meu caso foi:

Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Service files not changed. Skipping deployment...
Service Information
service: rest-api-python-dynamodb
stage: dev
region: us-east-1
stack: rest-api-python-dynamodb-dev
resources: 24
api keys:
  None
endpoints:
  POST - https://ng9qw7apg3.execute-api.us-east-1.amazonaws.com/dev/cliente
  GET - https://ng9qw7apg3.execute-api.us-east-1.amazonaws.com/dev/cliente/{codigo}
  POST - https://ng9qw7apg3.execute-api.us-east-1.amazonaws.com/dev/cliente/{codigo}
functions:
  create: rest-api-python-dynamodb-dev-create
  get: rest-api-python-dynamodb-dev-get
  delete: rest-api-python-dynamodb-dev-delete
layers:
```

Uso

Podemos criar, deletar e encontrar um item por codigo. Segue os commandos:

Criar

```
curl -X POST https://XXXXXXX.execute-api.us-east-1.amazonaws.com/dev/cliente --data
'{ "nome": "Jose Silva" }'
```

Retornará algo mais ou menos assim:

```
{"codigo": "e297ba10-099c-11ec-bf26-95a168b39df5", "nome": "Jose Silva", "createdAt":
"1630332996.2597847", "updatedAt": "1630332996.2597847"}
```

Pesquisa um item

```
curl  https://XXXXXXX.execute-api.us-east-1.amazonaws.com/dev/cliente/<codigo>
```

Retorno esperado:

```
{"codigo": "e297ba10-099c-11ec-bf26-95a168b39df5", "nome": "Jose Silva", "createdAt": "1630332996.2597847", "updatedAt": "1630332996.2597847"}
```

Excluir um item

```
curl -X POST https://XXXXXXX.execute-api.us-east-1.amazonaws.com/dev/cliente/<codigo>
```

10 - Destruindo

Execute o comando `cd ..` para entrar na pasta
Execute o comando `cd terraform-dynamoDB/` para entrar na pasta
Execute um `terraform destroy -auto-approve` para que sejam removidos os recursos na AWS

Execute o comando `cd ..` para entrar na pasta
Execute o comando `cd api-gateway-lambda-serverless/` para entrar na
Execute o comando `sls remove` para que sejam removidos os recursos na AWS