🤖 Implementación Local de Infraestructura con LocalStack: Despliegue y Configuración de Recursos, Incluyendo Creación de Tabla DynamoDB con Terraform.

Paso 1: Configuración Inicial

a. Instalar Terraform

Descarga e instala Terraform:

```
bash
```

```
wget https://releases.hashicorp.com/terraform/0.15.5/terraform_0.15.5_linux_amd64.zip
unzip terraform_0.15.5_linux_amd64.zip
sudo mv terraform /usr/local/bin/

```

b. Instalar LocalStack

Instala LocalStack siguiendo las instrucciones del sitio web oficial de LocalStack, o puedes instalar LocalStack utilizando el administrador de paquetes de Python, `pip`

Ejecuta el siguiente comando:

```
bash
```

```
pip install localstack
```

c. Instalar Bibliotecas de Python

Crea y activa un entorno virtual:

```
bash
```

```
python3 -m venv ./venv
source ./venv/bin/activate

```

Instala las bibliotecas de Python necesarias:

```
bash
```

```
python3 -m pip install --upgrade localstack
pip install awscli awscli-local terraform-local boto3

```

Paso 2: Iniciar LocalStack

Inicia LocalStack ejecutando el siguiente comando:

```
bash
```

```
localstack start

```

Paso 3: Crear 2 SQS con Terraform

a. Ubicación y Configuración

Asegúrate de estar en el directorio que contiene tus archivos de Terraform.

b. Contenido de `provider.tf`

```
hcl
```

```hcl
provider "aws" {
  access_key = "test"
  secret_key = "test"
  region     = "us-east-1"
}

```

c. Contenido de `main.tf`

```
hcl
```

```hcl
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

resource "aws_security_group" "my_security_group" {
  name        = "my-security-group"
  description = "Security group for my EC2 instances"

  ingress {
    from_port   = 8000
    to_port     = 8000
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "app_server" {
  ami           = "ami-ff0fea8310f3"
  instance_type = "t3.nano"
  vpc_security_group_ids  = [aws_security_group.my_security_group.id]
  user_data       = file("user_script.sh")

  tags = {
    Name = "ExampleAppServerInstance"
  }
  count = 2
}

resource "aws_sqs_queue" "tf_queue_one" {
  name                      = "queue-one"
  delay_seconds             = 10
  max_message_size          = 2048
  message_retention_seconds = 86400
```

```
  receive_wait_time_seconds = 10

  tags = {
    Environment = "production"
  }
}

resource "aws_sqs_queue" "tf_queue_two" {
  name                      = "queue-two"
  delay_seconds             = 10
  max_message_size          = 2048
  message_retention_seconds = 86400
  receive_wait_time_seconds = 10

  tags = {
    Environment = "production"
  }
}

```

d. Ejecutar Terraform

```
bash
```

```
terraform init
terraform apply -auto-approve

```

Esto inicializará Terraform y creará las colas SQS definidas en el archivo `main.tf`.

Paso 4: Crear Mensaje Manualmente en una Cola SQS

a. Obtener URL de Cola

Después de crear las colas con Terraform, ejecuta el siguiente comando para obtener la URL de la cola:

```
bash
```

```
awslocal sqs get-queue-url --queue-name queue-one

```

b. Enviar Mensaje

Utiliza la URL de la cola obtenida para enviar un mensaje:

```
bash
```

```
awslocal sqs send-message --queue-url <URL_obtenida_anteriormente> --message-body "Mensaje de prueba"

```

Paso 5: Crear Tabla DynamoDB con Terraform

a. Ubicación y Configuración

Asegúrate de estar en el directorio que contiene tus archivos de Terraform.

b. Contenido de `main.tf`

```
hcl
```

```
resource "aws_dynamodb_table" "primera_tabla" {
  name           = "PrimeraTabla"
  billing_mode   = "PROVISIONED"
  read_capacity  = 5
  write_capacity = 5
  hash_key       = "Id"

  attribute {
    name = "Id"
    type = "S"
  }
}

```

c. Ejecutar Terraform

```
bash
```

```
terraform init
terraform apply -auto-approve

```

Esto inicializará Terraform y creará la tabla DynamoDB definida en el archivo `main.tf`.

Paso 6: Crear Código de Python para Levantar los Servicios

a. Crear un nuevo archivo llamado `deploy_services.py`

```
python
```

```
import boto3
import time

localstack_endpoint = 'http://localhost.localstack.cloud:4566'
aws_access_key_id = "test"
aws_secret_access_key = "test"
aws_region = "us-east-1"

def create_dynamodb_table():
    dynamodb = boto3.client('dynamodb', aws_access_key_id=aws_access_key_id, aws_secret_access_key=aws_secret_ac

    # Resto del código para crear la tabla DynamoDB

def create_sqs_queue():
    sqs = boto3.client('sqs', aws_access_key_id=aws_access_key_id, aws_secret_access_key=aws_secret_access_key,

    # Resto del código para crear las colas SQS

def launch_ec2_instance():
    ec2 = boto3.client('ec2', aws_access_key_id=aws_access_key_id, aws_secret_access_key=aws_secret_access_key,

```

Continuación - Paso 6: Crear Código de Python para Levantar los Servicios

b. Completar el código de `deploy_services.py`

```
python
```

```
    ec2 = boto3.client('ec2', aws_access_key_id=aws_access_key_id, aws_secret_access_key=aws_secret_access_key,

    ami_id = 'ami-ff0fea8310f3'  # Reemplaza con un AMI válido
    instance_type = 't3.nano'

    response = ec2.run_instances(
        ImageId=ami_id,
        InstanceType=instance_type,
        MinCount=1,
        MaxCount=1
    )

    instance_id = response['Instances'][0]['InstanceId']
    print(f'Instancia EC2 lanzada en LocalStack: {instance_id}')

# Llamar a las funciones para crear los recursos en LocalStack
create_dynamodb_table()
create_sqs_queue()
launch_ec2_instance()

```

Paso 7: Ejecutar el Código de Python

Ejecuta el script Python recién creado para interactuar con LocalStack y crear los recursos definidos:

```
bash
```

```
python3 deploy_services.py

```

Este script ejecutará las funciones que crean la tabla DynamoDB, las colas SQS y lanzarán una instancia EC2 en LocalStack.

Paso 8: Destruir Recursos con Terraform

Si deseas destruir los recursos creados en LocalStack, sigue estos pasos:

a. Ir al Directorio de Terraform

Asegúrate de estar en el directorio que contiene tus archivos de Terraform.

b. Ejecutar Terraform para Destruir Recursos

Ejecuta el siguiente comando para destruir los recursos:

```
bash
```

```
terraform destroy -auto-approve

```

Este comando eliminará todas las instancias EC2, colas SQS y la tabla DynamoDB creadas anteriormente.

Paso 9: Desconectar LocalStack

Cuando hayas terminado de usar LocalStack, puedes desconectarlo ejecutando el siguiente comando:

```
bash
```

```
localstack stop

```

Este comando detendrá LocalStack y liberará los recursos locales.

Recuerda que la destrucción de recursos es irreversible. Asegúrate de confirmar que deseas destruir los recursos antes de ejecutar el comando `terraform destroy`.