# Creacion y muestra de servicios DynamoDB aws con localstack

Comandos de creación de tablas:

aws --endpoint-url=http://localhost:4566 --profile default dynamodb create-table --table-name SegundaTabla --attribute-definitions AttributeName=Id,AttributeType=S --key-schema AttributeName=Id,KeyType=HASH --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5

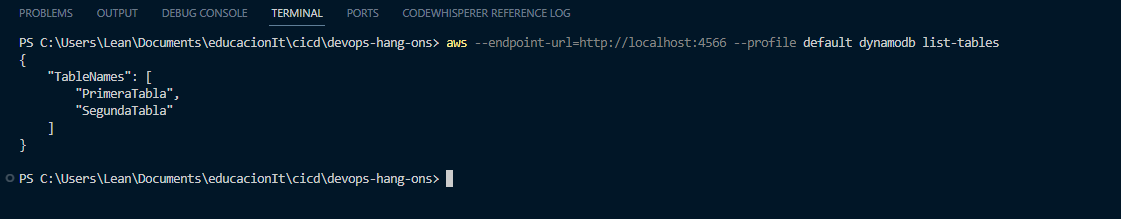
aws --endpoint-url=http://localhost:4566 --profile default dynamodb create-table --table-name SegundaTabla --attribute-definitions AttributeName=Id,AttributeType=S --key-schema AttributeName=Id,KeyType=HASH --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5

Listo las dos tablas creadas:

aws --endpoint-url=http://localhost:4566 --profile default dynamodb list-tables

// Revisa la informacion de las tablas

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



# Crear modulo terraform para levantar tablas DynamoDB en localstack

Generamos el archivo main.tf:

provider "aws" {

region = "us-east-1"

access\_key = "1234"

secret\_key = "1234"

skip\_credentials\_validation = true

skip\_metadata\_api\_check = true

skip\_requesting\_account\_id = true

endpoints {

dynamodb = "http://localhost:4566"

}

}

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"

}

}

resource "aws\_dynamodb\_table" "segunda\_tabla" {

name = "SegundaTabla"

billing\_mode = "PROVISIONED"

read\_capacity = 5

write\_capacity = 5

hash\_key = "UserId"

attribute {

name = "UserId"

type = "S"

}

}

Ejecutamos los siguientes commandos:

terraform init

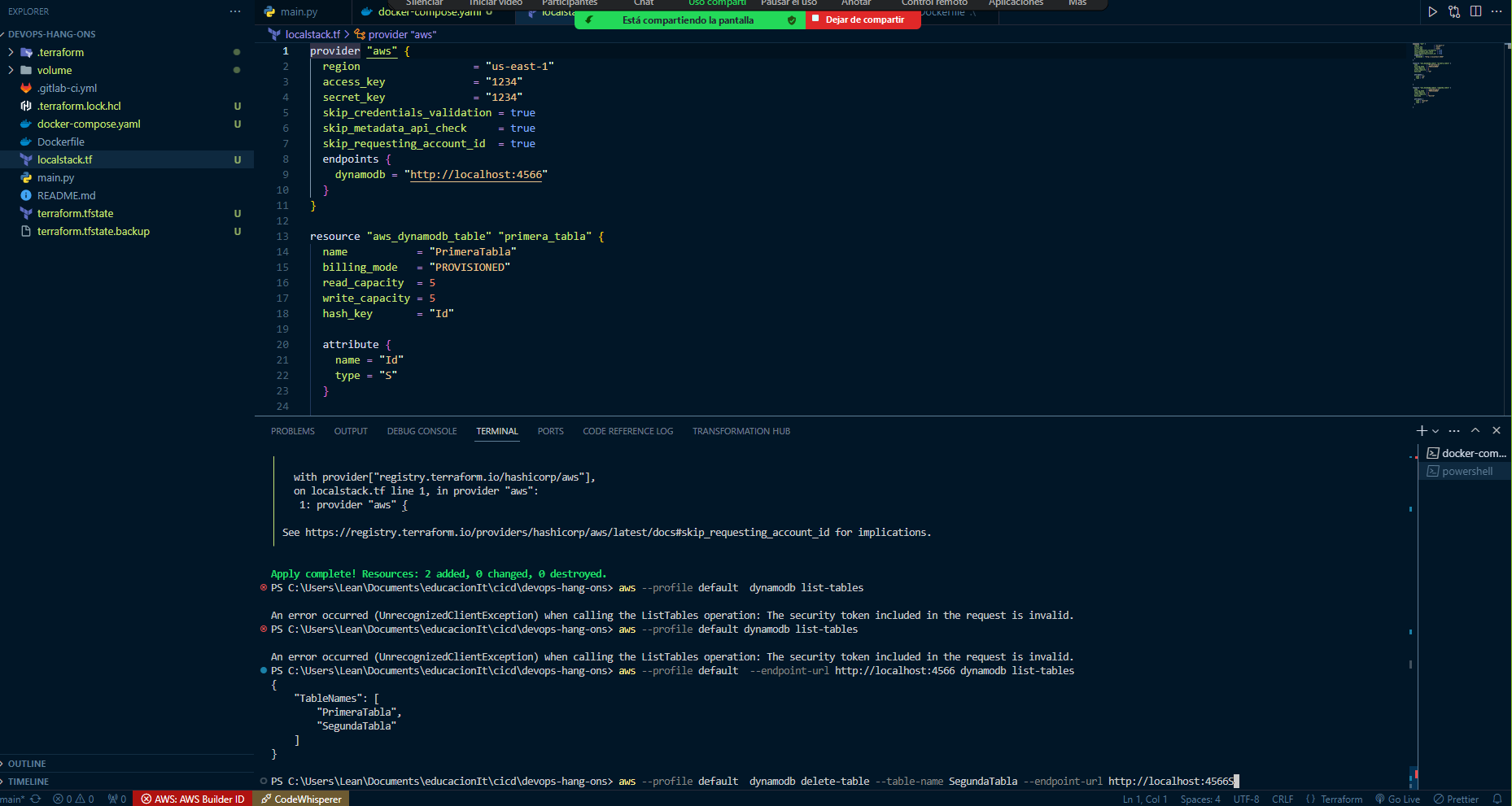
terraform plan

terraform apply

En caso de ser necesario, se deben eliminar las tablas pre-existentes

aws --profile default --endpoint-url http://localhost:4566 dynamodb list-tables

Adjuntamos captura:



Crear servicio S3 en localstack

Se genera el archivo de configuracion .tf para crear los buckets y subir el recurso:

resource "aws\_s3\_bucket" "bucket1" {

bucket = "bucket1"

}

resource "aws\_s3\_bucket" "bucket2" {

bucket = "bucket2"

}

resource "aws\_s3\_object" "bucket1" {

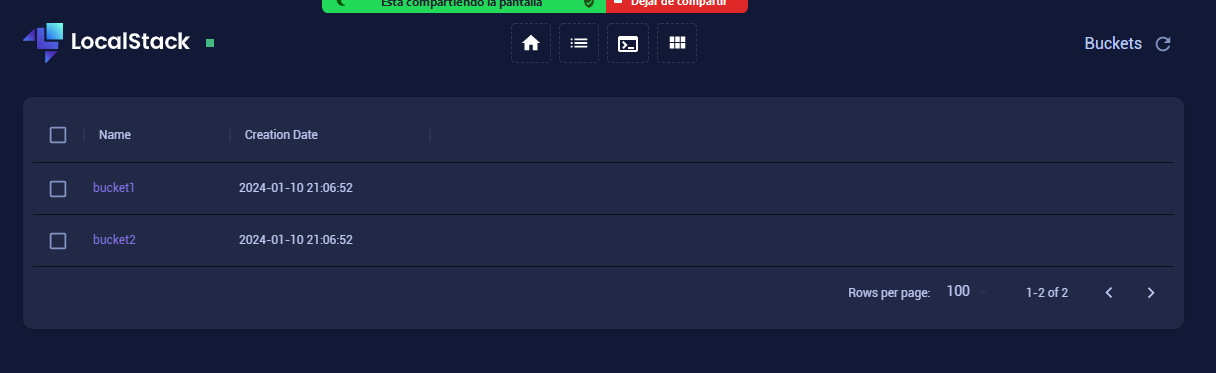
bucket = aws\_s3\_bucket.bucket1.bucket

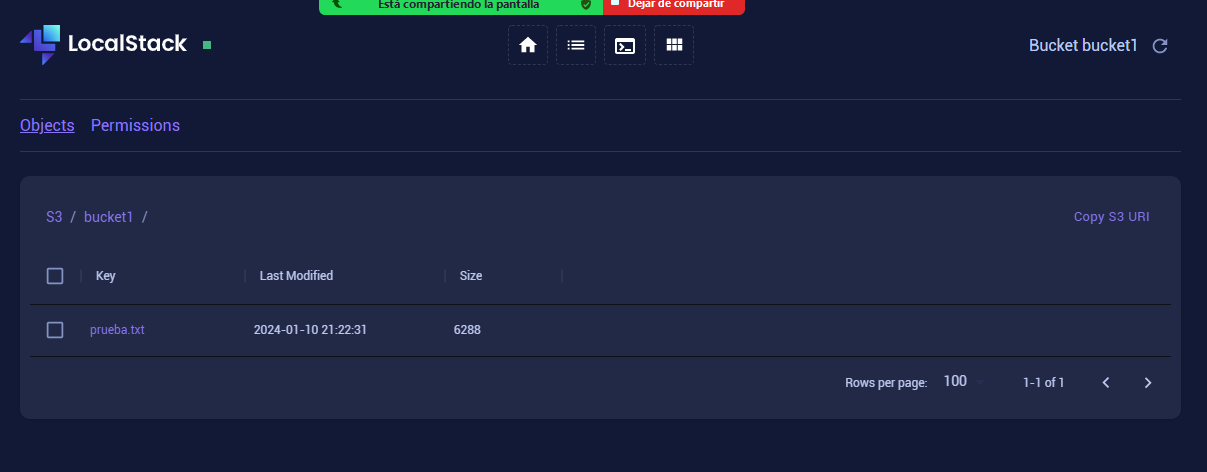
key = "prueba.txt" *# Nombre del archivo en el bucket*

acl = "private"

source = "./README.md" *# Ruta local del archivo que deseas subir al bucket*

}





# Crear servicio EC2 en localstack

Se genera las EC2 y SQS según la documentacion del grupo 1:

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 = 1

}

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

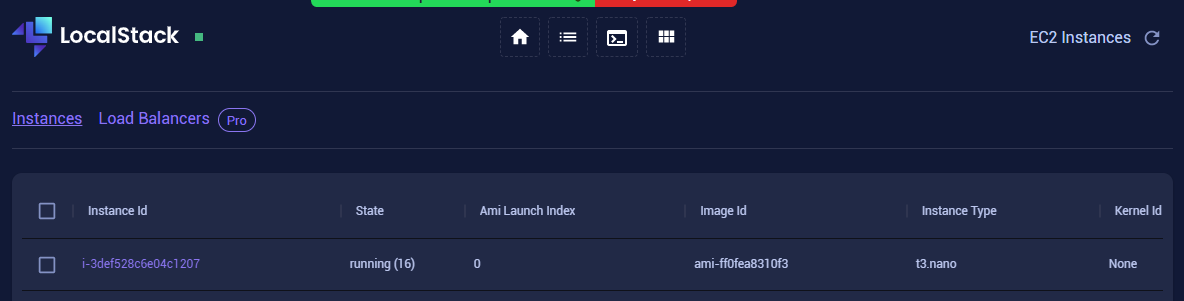
receive\_wait\_time\_seconds = 10

tags = {

Environment = "production"

}

}



# Automation LocalStack

Se genera el siguiente codigo para levantar los servicios:

**import** boto3

**import** time

*# Configura el cliente de DynamoDB*

dynamodb\_client = boto3.client('dynamodb', endpoint\_url='http://localhost:4566')

*# Configura el cliente de EC2*

ec2\_client = boto3.client('ec2', endpoint\_url='http://localhost:4566')

*# Configura el cliente de S3*

s3\_client = boto3.client('s3', endpoint\_url='http://s3.localhost.localstack.cloud:4566')

**def** create\_dynamodb\_table():

table\_name = 'tabla1'

attribute\_definitions = [

{'AttributeName': 'id', 'AttributeType': 'S'},

]

key\_schema = [

{'AttributeName': 'id', 'KeyType': 'HASH'},

]

dynamodb\_client.create\_table(

TableName=table\_name,

AttributeDefinitions=attribute\_definitions,

KeySchema=key\_schema,

ProvisionedThroughput={

'ReadCapacityUnits': 1,

'WriteCapacityUnits': 1,

}

)

**print**(f'DynamoDB table {table\_name} created.')

**def** create\_ec2\_instance():

instance\_type = 't3.nano'

ami\_id = 'ami-ff0fea8310f3' *# Replace with a valid AMI ID*

response = ec2\_client.run\_instances(

ImageId=ami\_id,

InstanceType=instance\_type,

MinCount=1,

MaxCount=1,

)

instance\_id = response['Instances'][0]['InstanceId']

**print**(f'EC2 instance {instance\_id} created.')

**def** create\_s3\_bucket():

bucket\_name = 'bucket1'

s3\_client.create\_bucket(Bucket=bucket\_name)

**print**(f'S3 bucket {bucket\_name} created.')

**if** \_\_name\_\_ == "\_\_main\_\_":

*# Crea la tabla de DynamoDB*

create\_dynamodb\_table()

*# Crea la instancia de EC2*

create\_ec2\_instance()

*# Crea el bucket de S3*

create\_s3\_bucket()

*# Espera unos segundos para que los servicios estén completamente disponibles*

time.sleep(5)

**print**("Servicios creados exitosamente.")

