

2. Creación de subredes públicas y privadas en la VPC simulada en LocalStack.

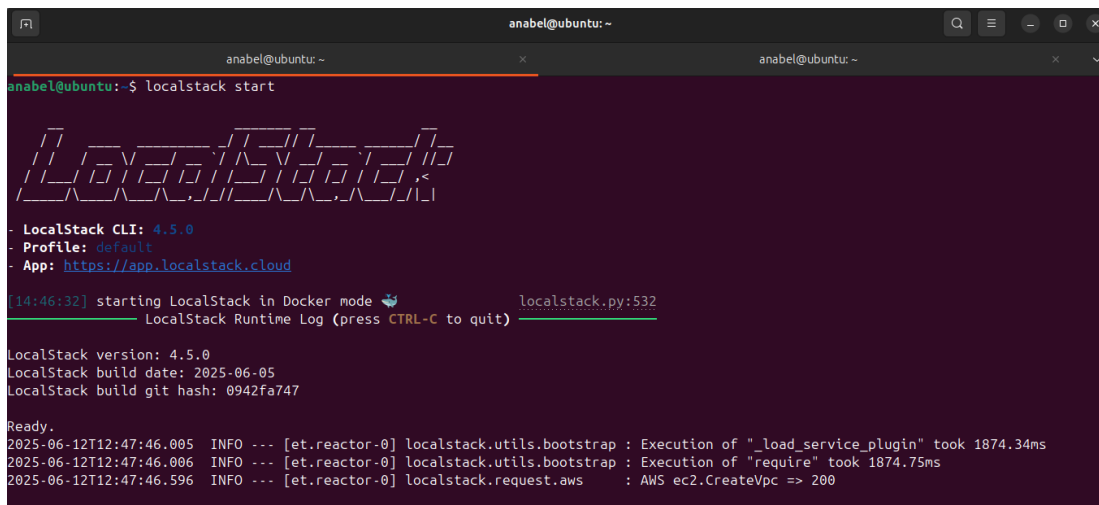
Para diseñar la arquitectura de red dentro de la VPC simulada, vpc-sprint1, vamos a realizar los siguientes pasos:

1. Iniciamos LocalStack

En una terminal, ejecutamos el comando:

```
localstack start
```

Esto lanza LocalStack en modo Docker. Esta terminal tiene que estar abierta mientras ejecutamos todo el proceso.

A screenshot of a terminal window titled 'anabel@ubuntu: ~'. The terminal shows the command 'localstack start' being executed. The output includes a stylized ASCII art logo for LocalStack, followed by CLI version 4.5.0, profile 'default', and the app URL 'https://app.localstack.cloud'. It then shows 'starting LocalStack in Docker mode' with a progress bar and 'LocalStack Runtime Log (press CTRL-C to quit)'. Below this, it displays 'LocalStack version: 4.5.0', 'LocalStack build date: 2025-06-05', and 'LocalStack build git hash: 0942fa747'. The terminal ends with 'Ready.' and three log entries from 'et.reactor-0' showing the execution of bootstrap and require functions, and an AWS 'ec2.CreateVpc' request returning '200'.

```
anabel@ubuntu: ~  
anabel@ubuntu:~$ localstack start  
  
LocalStack  
- LocalStack CLI: 4.5.0  
- Profile: default  
- App: https://app.localstack.cloud  
  
[14:46:32] starting LocalStack in Docker mode localstack.py:532  
LocalStack Runtime Log (press CTRL-C to quit)  
  
LocalStack version: 4.5.0  
LocalStack build date: 2025-06-05  
LocalStack build git hash: 0942fa747  
  
Ready.  
2025-06-12T12:47:46.005 INFO --- [et.reactor-0] localstack.utils.bootstrap : Execution of "_load_service_plugin" took 1874.34ms  
2025-06-12T12:47:46.006 INFO --- [et.reactor-0] localstack.utils.bootstrap : Execution of "require" took 1874.75ms  
2025-06-12T12:47:46.596 INFO --- [et.reactor-0] localstack.request.aws : AWS ec2.CreateVpc => 200
```

2. Obtenemos el ID de la VPC creada en el ejercicio 1

En otra ventana del terminal, mientras dejamos correr LocalStack, vamos a ver el ID de la VPC que hemos creado con el siguiente comando:

```
aws ec2 describe-vpcs --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```


- Creamos la subred pública en el rango 10.0.1.0./24
- Asignamos la etiqueta 'SubredPublica' para identificarla

4. Creamos y asociamos el Internet Gateway a nuestra subred pública

Después de crear la subred pública, creamos el Internet Gateway para simular la conexión a Internet desde esta con el siguiente comando:

```
aws ec2 create-internet-gateway --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```

Con el anterior comando, conoceremos el ID del IG, el cuál es: igw-e550ea409b9265755. Este ID lo asociamos con la VPC ejecutando el comando:

```
aws ec2 attach-internet-gateway --internet-gateway-id igw-e550ea409b9265755 --vpc-id vpc-b555a072263e40c04 --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```



```
anabel@ubuntu: ~  
anabel@ubuntu: ~  
anabel@ubuntu: ~  
anabel@ubuntu:~$ aws ec2 create-internet-gateway --profile vpc-sprint1 --endpoint-url=http://localhost:4566  
{  
  "InternetGateway": {  
    "Attachments": [],  
    "InternetGatewayId": "igw-e590ea409b9265755",  
    "OwnerId": "000000000000",  
    "Tags": []  
  }  
}  
anabel@ubuntu:~$ aws ec2 attach-internet-gateway --internet-gateway-id igw-e590ea409b9265755 --vpc-id vpc-b555a072263e40c04 --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```

Con esto, tenemos el ID del IG asociado a nuestra subred pública.

5. Creamos y asociamos la tabla de rutas a nuestra subred pública

Ahora, creamos una tabla de rutas con el comando:

```
aws ec2 create-route-table --vpc-id vpc-b555a072263e40c04 --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```

Tras crear la tabla de rutas, creamos una ruta hacia internet usando el Internet Gateway, convirtiendo nuestra tabla de rutas en una tabla pública, pudiendo conectarnos a Internet con ella. Esto lo hacemos con el comando:

```
aws ec2 create-route --route-table-id rtb-484d9868ef5883672 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-e590ea409b9265755 --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```

Finalmente, asociamos la tabla pública con nuestra subred pública con el comando:

```
aws ec2 associate-route-table --subnet-id subnet-c8a91c2099372fb88 --route-table-id rtb-484d9868ef5883672 --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```


- Creamos la subred privada en el rango 10.0.2.0/24
- Asignamos la etiqueta 'SubredPrivada' para identificarla
- Subred creada para servicios internos que no necesitan Internet

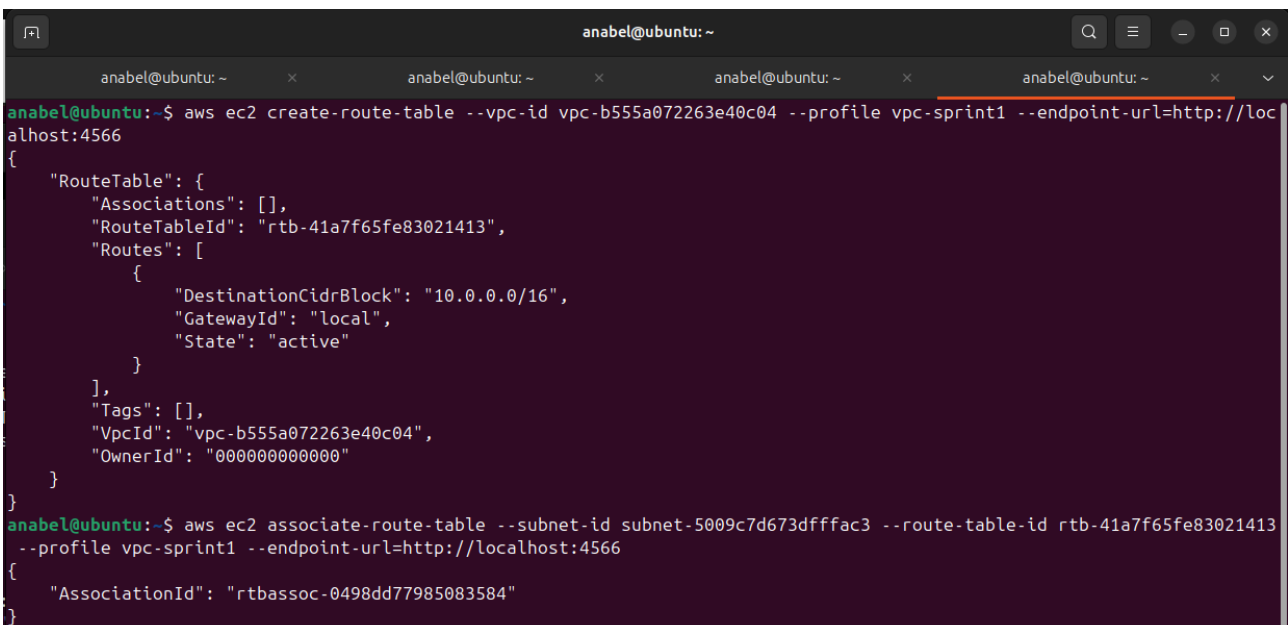
7. Creamos y asociamos la tabla de rutas a nuestra subred privada

Creamos una tabla de rutas con el comando:

```
aws ec2 create-route-table --vpc-id vpc-b555a072263e40c04 --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```

Tras esto, asociamos la tabla creada con nuestra subred privada con el comando:

```
aws ec2 associate-route-table --subnet-id subnet-5009c7d673dfffac3 --route-table-id rtb-41a7f65fe83021413 --profile vpc-sprint1 --endpoint-url=http://localhost:4566
```



```
anabel@ubuntu: ~  
anabel@ubuntu:~$ aws ec2 create-route-table --vpc-id vpc-b555a072263e40c04 --profile vpc-sprint1 --endpoint-url=http://localhost:4566  
{  
  "RouteTable": {  
    "Associations": [],  
    "RouteTableId": "rtb-41a7f65fe83021413",  
    "Routes": [  
      {  
        "DestinationCidrBlock": "10.0.0.0/16",  
        "GatewayId": "local",  
        "State": "active"  
      }  
    ],  
    "Tags": [],  
    "VpcId": "vpc-b555a072263e40c04",  
    "OwnerId": "000000000000"  
  }  
}  
anabel@ubuntu:~$ aws ec2 associate-route-table --subnet-id subnet-5009c7d673dfffac3 --route-table-id rtb-41a7f65fe83021413 --profile vpc-sprint1 --endpoint-url=http://localhost:4566  
{  
  "AssociationId": "rtbassoc-0498dd77985083584"  
}
```

Finalmente, ya tenemos en nuestra red virtual privada vpc-sprint1 una subred pública con acceso a Internet y una subred privada.