Pasos para subir aplicación en Serverless/SAM

1. Instalar CLI de AWS SAM de: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/install-sam-cli.html
2. Crear un bucket S3 en AWS.
    a. Desactiva la opción "Block all public access" si vas a permitir archivos públicos.
3. Crear una política para permitir acceso público a archivos específicos.
    a. Ve a tu bucket > pestaña "Permissions" > sección "Bucket policy".
    b. Pega esta política (reemplaza purchase-receipts-group27
    c. por el nombre de tu bucket):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource":
"arn:aws:s3:::purchase-receipts-group27/*"
    }
  ]
}
```

4. Crea un nuevo directorio para el proyecto SAM, y marca las siguientes opciones:

```
sam init --runtime python3.11 --name {generate-receipt-pdf}
```

    a. Which template source would you like to use? 1 AWS Quick Start Templates
    b. Choose an AWS Quick Start application template 1 Hello World Example
    c. Would you like to enable X-Ray tracing on the function(s) in your application? N
    d. Would you like to enable monitoring using CloudWatch Application Insights?

    For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch-application-insights.html [y/N]: N

e. Would you like to set Structured Logging in JSON format on your Lambda functions? [y/N]: N

5. Ingresar al directorio creado.
6. Renombrar el directorio con nombre hello-world a generate_receipt_pdf.
7. Modificar el archivo app.py del directorio generate_receipt_pdf para generar la boleta de pago con los datos correspondientes.

```python
import json
import boto3
import os
import tempfile
from datetime import datetime
from reportlab.lib.pagesizes import LETTER
from reportlab.pdfgen import canvas

s3 = boto3.client('s3')
BUCKET_NAME = os.environ.get("BUCKET_NAME", "purchase-receipts-group27")
GROUP_ID = "27"

def lambda_handler(event, context):
    body = json.loads(event['body'])

    user_email = body['user_email']
    stock_name = body['stock_name']
    quantity = body['quantity']
    total = body['total']

    file_name = f"receipt_{user_email}_{datetime.utcnow().timestamp()}.pdf"

    # Crear PDF temporal
    with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as tmp_file:

    # Subir a S3 después de cerrar el archivo
    s3.upload_file(
        tmp_file.name,
        BUCKET_NAME,
        file_name,
    )
    receipt_url = s3.generate_presigned_url(
        ClientMethod='get_object',
        Params={
            'Bucket': BUCKET_NAME,
            'Key': file_name
        },
        ExpiresIn=3600  # 1 hora
    )
    os.remove(tmp_file.name)  # Limpia el archivo temporal

    return {
        "statusCode": 200,
        "body": json.dumps({"receipt_url": receipt_url})
    }
```

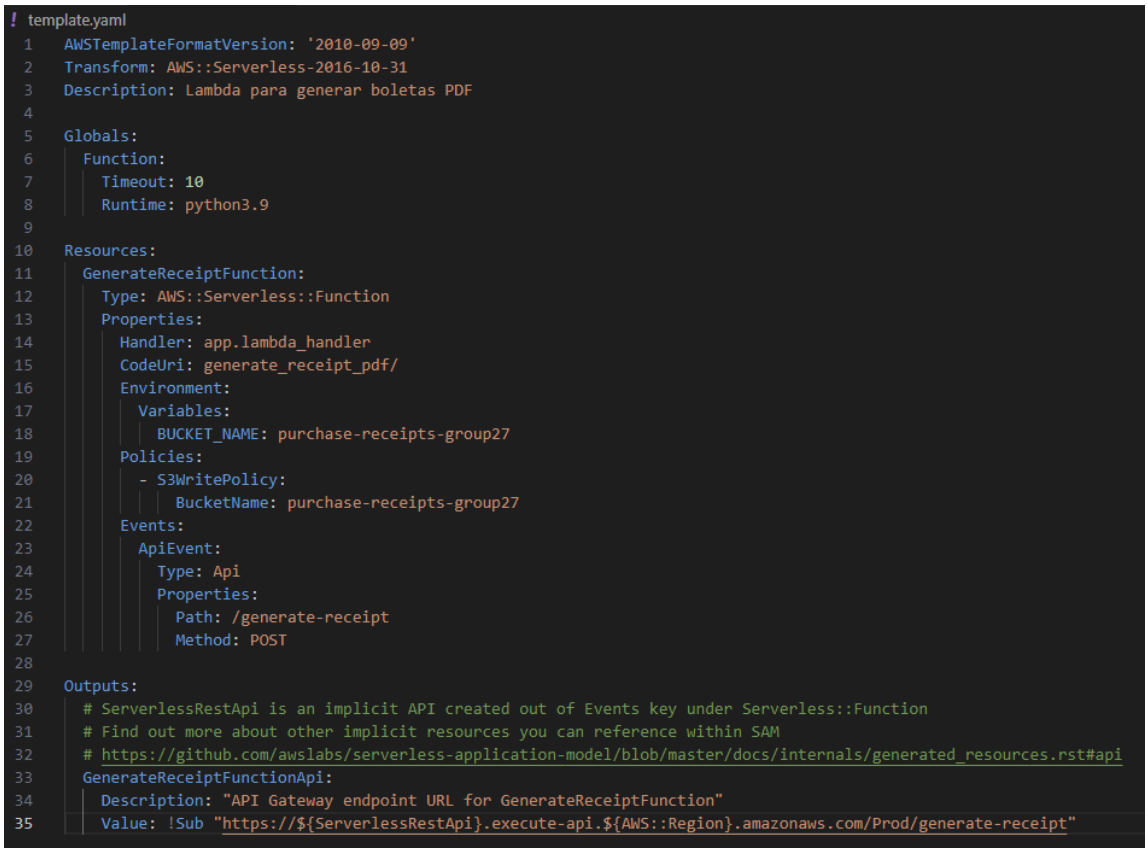
8. Modificar el archivo requirements.txt del directorio generate_receipt_pdf, añadiendo las siguientes librerías:

```
reportlab
```

```
boto3
```

9. Modificar el archivo *template.yaml*.

Nota: Asegúrate de tener la misma versión de python o modificarla.



```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Lambda para generar boletas PDF

Globals:
  Function:
    Timeout: 10
    Runtime: python3.9

Resources:
  GenerateReceiptFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: app.lambda_handler
      CodeUri: generate_receipt_pdf/
      Environment:
        Variables:
          BUCKET_NAME: purchase-receipts-group27
      Policies:
        - S3WritePolicy:
            BucketName: purchase-receipts-group27
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /generate-receipt
            Method: POST

Outputs:
  # ServerlessRestApi is an implicit API created out of Events key under Serverless::Function
  # Find out more about other implicit resources you can reference within SAM
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/generated_resources.rst#api
  GenerateReceiptFunctionApi:
    Description: "API Gateway endpoint URL for GenerateReceiptFunction"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/generate-receipt"
```

10. Despliegue con SAM con los comandos:

```
sam build
```

```
sam deploy --guided
```

Respondiendo las preguntas de la siguiente manera:

- Stack name: generate-receipt-stack
- Region: us-east-1 (tu región)
- Confirm changeset: Y
- Allow SAM to create roles: Y

Guardando el URL generado (ej: https://oi9ys5pgnd.execute-api.us-east-1.amazonaws.com/Prod/generate-receipt)

11. Modificación en el backend

Se realiza una petición POST a la Lambda (mediante el URL generado anteriormente) desde el backend cuando la transacción es exitosa.

```python
@app.post("/webpay/commit")
async def commit_transaction(request: Request, user=Depends(verify_token)):
    body = await request.json()
    token_ws = body.get("token_ws")
    transaction = transactions_collection.find_one({"request_id": body.get("request_id")})

    # TRANSACCIÓN ANULADA POR EL USUARIO
    if not token_ws or token_ws == "": ···

    try:
        response = tx.get_tx().commit(token_ws)

        response_status = "OK" if response["response_code"] == 0 else "REJECTED"
        mqtt_manager.publish_validation(body.get("request_id"), response_status, transaction

        # TRANSACCIÓN RECHAZADA
        if response["response_code"] != 0: ···

        # TRANSACCIÓN ACEPTADA
        #Generación de boleta
        receipt_url = purchase_receip.generate_receipt(
            user_data={"email": user["sub"]},
            stock_data={
                "name": transaction["symbol"],
                "quantity": transaction["quantity"],
                "total": response["amount"]
            }
        )
        transactions_collection.update_one(···

        return {"status": "OK",···

    except Exception as e:···
```

Y se devuelve la URL generada por la función lambda para descargar la boleta desde S3.

```python
return {"status": "OK",
        "message": "Transacción ha sido autorizada.",
        "transaction_id": response["buy_order"],
        "receipt_url": receipt_url}
```

```python
import requests

def generate_receipt(user_data, stock_data):
    payload = {
        "user_email": user_data["email"],
        "stock_name": stock_data["name"],
        "quantity": stock_data["quantity"],
        "total": stock_data["total"]
    }

    response = requests.post("https://oi9ys5pgnd.execute-api.us-east-1.amazonaws.com/Prod/generate-receipt", json=payload)
    if response.ok:
        return response.json()["receipt_url"]
    else:
        raise Exception("Error generando la boleta")
```

12. Modificación en el frontend

Con la URL para descargar la boleta devuelta por el backend se añade la opción desde el frontend.

```jsx
{stock?.receipt_url && (
    <div className="flex justify-center mt-4">
        <a
            href={stock.receipt_url}
            target="_blank"
            rel="noopener noreferrer"
            className="w-full max-w-xs bg-blue-600 text-white font-medium py-3 px-4 rounded-md hover:bg-blue-700 disabled:opaci
        >
            Descargar Boleta PDF
        </a>
    </div>
)}
```

13. Obteniendo el siguiente resultado final: