class TestCreateThumbnail(unittest.TestCase):

@mock\_s3

@mock\_events

@patch("boto3.client")

def test\_create\_thumbnail(self, boto\_mock):

# Create a mock S3 bucket

s3 = boto3.client("s3", region\_name="us-east-1")

s3.create\_bucket(Bucket="photos-yma")

s3.put\_object(

Bucket="photos-yma", Key="uploads/test\_image.jpg", Body=b"test data"

)

# Create a mock EventBridge event

event = {

"Records": [

{

"s3": {

"bucket": {"name": "photos-yma"},

"object": {"key": "uploads/test\_image.jpg"},

}

}

]

}

# Mock the EventBridge client

events = boto\_mock.return\_value

events.put\_events.return\_value = {

"FailedEntryCount": 0,

"Entries": [{"EventId": "1"}],

}

# Call the Lambda handler

create\_thumbnail(event, None)

# Check that an event was put to EventBridge

events.put\_events.assert\_called\_once()

# Check the content of the EventBridge event

call\_args = events.put\_events.call\_args[0]

self.assertEqual(call\_args[0]['Source'], 'custom.myApp')

self.assertEqual(call\_args[0]['DetailType'], 'ImageProcessing')

self.assertEqual(json.loads(call\_args[0]['Detail'])['status'], 'SUCCEEDED')

# Check that the image was processed and uploaded to S3

processed\_image = s3.get\_object(Bucket="photos-yma", Key="processed/test\_image.jpg")

self.assertEqual(processed\_image['Body'].read(), b"test data") # replace with the expected content of the processed image