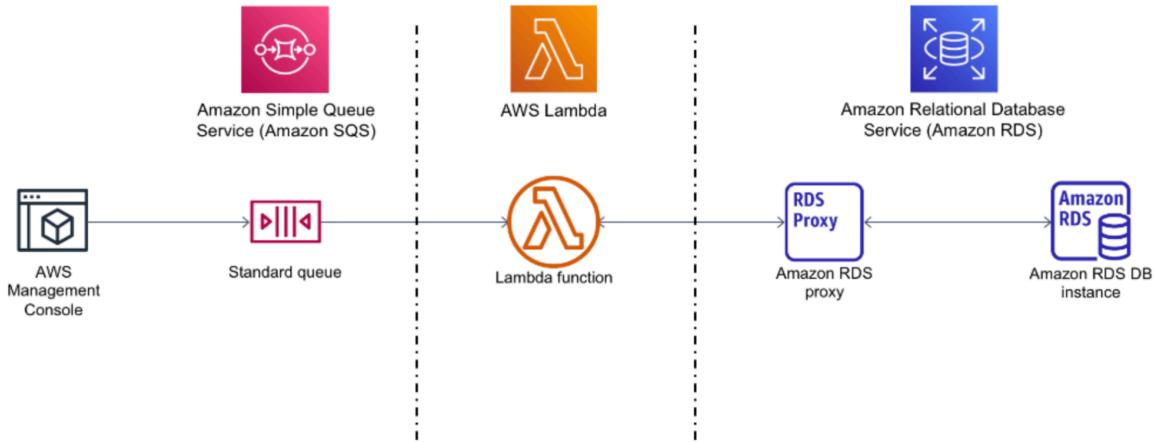


AWS Laboratorio II - SQS/Lambda/RDS



1- Crear RDS

Entrando desde el panel web a la sección RDS desde el buscador y luego click en **Create database** -> seleccionar tipo Standar -> MySQL -> Marcar si o si “Free Tier” -> en Settings en **DB identifier** escribir un nombre, por ej: “MySQLForLambdaMundose” -> en credentials settings usar **Master username y master password**

User: admin

pass: qwerty.1234

Dejar seleccionado la opción “Don’t connect to an EC2...”

Seleccionar una VPC sí o sí con más de una zona de disponibilidad y con más de una subnet (puede ser la default) -> public access **No**

Additional configuration -> database options -> initial database name “ExampleDB”

Muestra un estimado de 14USD pero recordar que es mensual, a fines del laboratorio no debería generar costos elevados.

Una vez creada la RDS la seleccionamos y vamos al desplegable de **actions** y seleccionamos “**set up Lambda connection**” -> crear nueva función -> name “LambdaFunctionRDSmundose” -> bajar y seleccionar en **rds proxy** -> crear nuevo proxy -> seleccionar en database credentials “**Database user and pass**” y colocar debajo las credenciales recién creadas -> por último **set up**.

2- Crear rol/es para Lambdas.

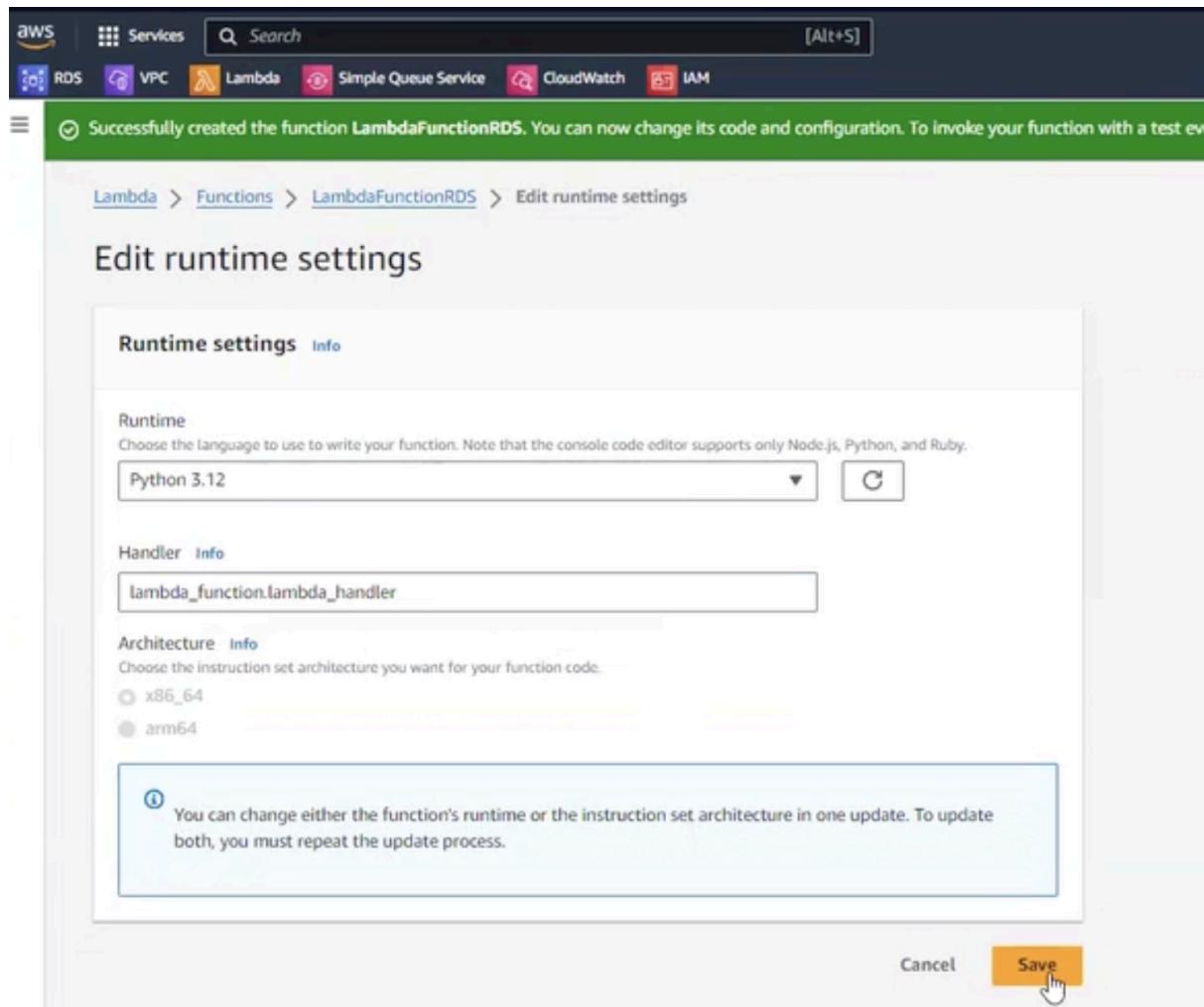
Ir a IAM por el buscador, después en el panel de la izquierda buscar **Roles**

A la derecha de la pantalla click en **Create role** -> seleccionar en **Trusted entity type** “AWS Service” -> en **Use Case** en el desplegable select “Lambda” -> **Next** -> en **Permissions policies** buscar “AWSLambdaSQSQueueExecutionRole” y “AWSLambdaVPCAccessExecutionRole” -> **Next** -> dar nombre al rol “lambda-vpc-sqs-role-mundose” -> **create role**

3- Crear lambda

Técnicamente ya debería estar creada, si no aparece dar a refresh de la consola. Si el paso anterior (desde la RDS) está bien, una Lambda fue creada con el nombre que le dimos. Si no aparece dar click en el botón refresh.

Una vez que aparece la seleccionamos y bajamos hasta “**Runtime Settings**” y seleccionamos “**Editar**” (el que es solo editar) y del desplegable “**Runtime**” seleccionamos cualquiera de “Python 3.12” y en “**Handler**” escribimos “lambda_function.lambda_handler”
-> **save**



Ahora debemos de subir un .zip con el nombre “lambda_function” (respetando el nombre), este archivo está alojado en el github compartido con el nombre lambda_funtion.py lo transformamos en zip localmente y volvemos al panel web de AWS.

En la pestaña “**Code**” a la misma línea que “**Code source**” en el margen derecho el botón “upload from” y seleccionamos lambda_function.zip (nos deberia quedar asi)

```

1 import sys
2 import logging
3 import pymysql
4 import json
5 import os
6
7 # rds settings
8 user_name = os.environ['USER_NAME']
9 password = os.environ['PASSWORD']
10 rds_proxy_host = os.environ['RDS_PROXY_HOST']
11 db_name = os.environ['DB_NAME']
12
13 logger = logging.getLogger()
14 logger.setLevel(logging.INFO)
15
16 # create the database connection outside of the handler to allow connections to be
17 # re-used by subsequent function invocations.
18 try:
19     conn = pymysql.connect(host=rds_proxy_host, user=user_name, passwd=password, db=db_name, connect_timeout=5)
20 except pymysql.MySQLError as e:
21     logger.error("ERROR: Unexpected error: could not connect to MySQL instance.")
22     logger.error(e)
23     sys.exit(1)
24

```

Vamos a probar el test (click en **test**) y configuraremos event name random por ej. "TestLambada", y pegamos el primer bloque de código del archivo queries.json que está en el repo de github en "Event JSON" -> **Save** y despues Test (va a fallar)

```

1 [
2     "Records": [
3         {
4             "messageId": "059f36b4-87a3-44ab-83d2-661975830a7d",
5             "receiptHandle": "AQEBwJnKyrHigUMZj6rYigCgxlaS3SLy0a...",
6             "body": "{\n                 \"CustID\": 1021,\n                 \"Name\": \"Martha Rivera\"\n             }",
7             "attributes": {
8                 "ApproximateReceiveCount": "1",
9                 "SentTimestamp": "1545082649183",
10                "SenderId": "AIDAENQZJOL023YV34V0",
11                "ApproximateFirstReceiveTimestamp": "1545082649185"
12            },
13            "messageAttributes": {},
14            "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
15            "eventSource": "aws:sqs",
16            "eventSourceARN": "arn:aws:sqs:us-west-2:123456789012:my-queue",
17            "awsRegion": "us-west-2"
18        }
19    ]
20 ]

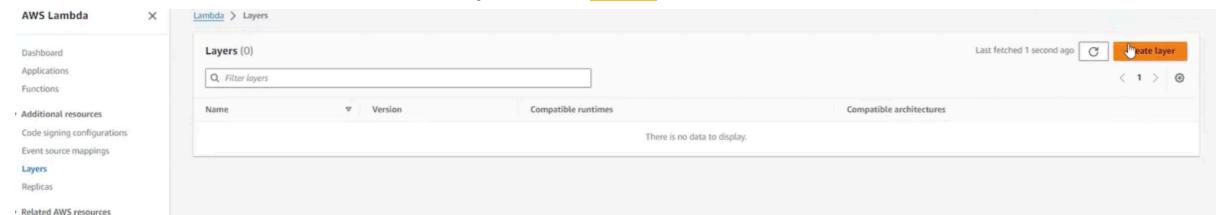
```

En general funciona de una, pero acá falla por un tema python y lambda. Se soluciona agregando un paso extra, que sería básicamente una layer en nuestra lambda, esto se hace así:

Antes de crear la layer se debe de respetar esta estructura de carpetas que está en el .zip del repositorio básicamente se debe de tener una carpeta "python" y dentro de ella lo necesario para las layer, o sea las dependencias. Sino se quiere usar el zip del repositorio lo hacemos de la siguiente manera "mkdir python -> pip install - -target python pymysql" (tener

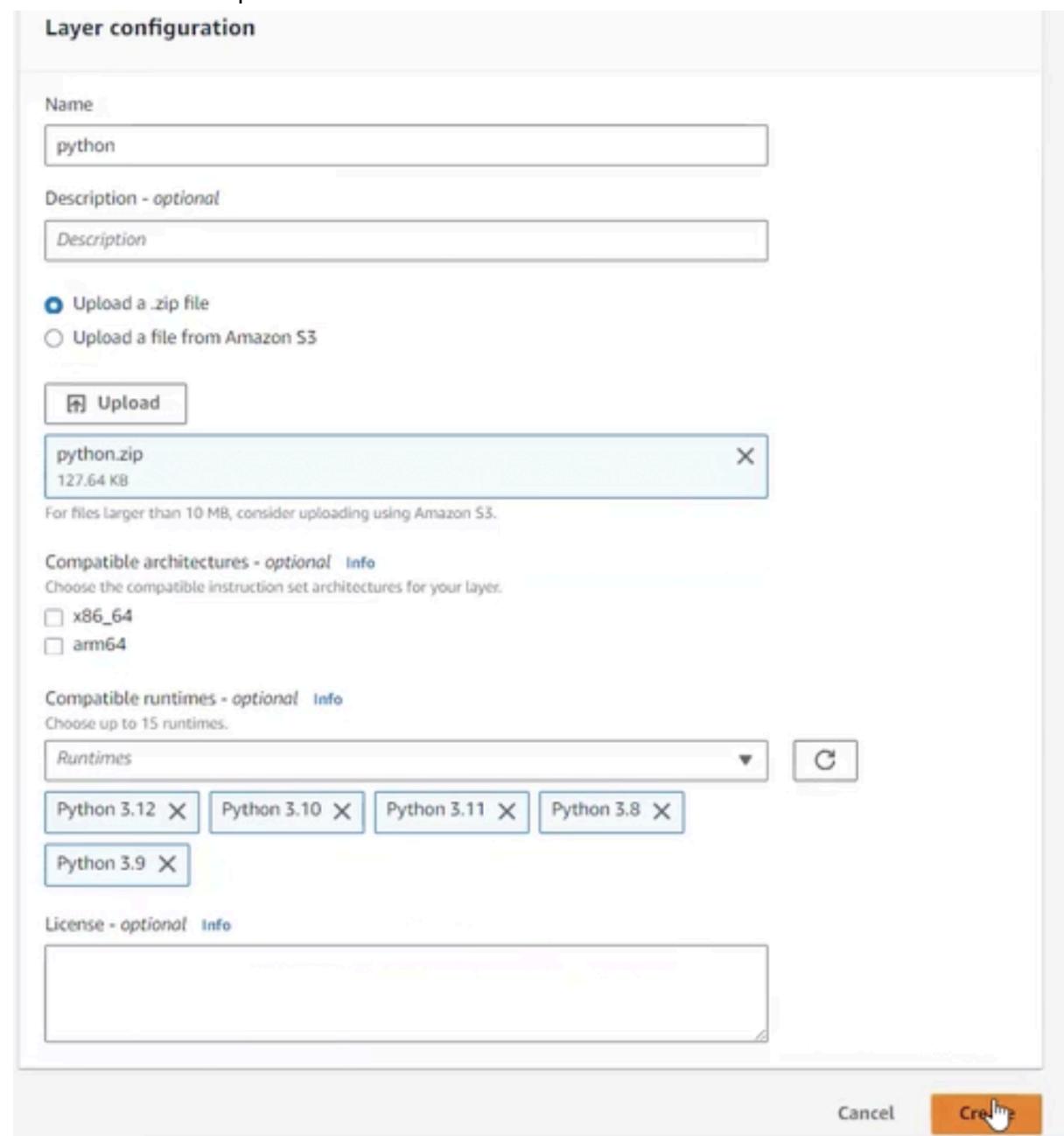
en cuenta la versión de pip que posean instalada en mi caso el comando es pip3 install --target python pymysql; para luego hacer esa carpeta python en un .zip; para simplificar esta subido al git el .zip listo para subir a la layer.

Volviendo a la web de AWS en el panel de las izquierda del servicio de Lambda buscar la opción “**layers**” -> **create layer** -> name “python” -> upload zip recién creado -> y seleccionamos todos los runtimes python -> **crear**



The screenshot shows the AWS Lambda Layers interface. On the left, there's a sidebar with options like Dashboard, Applications, Functions, Additional resources, Code signing configurations, Event source mappings, Layers (which is selected), Replicas, and Related AWS resources. The main area is titled "Layers (0)" and contains a table with columns for Name, Version, Compatible runtimes, and Compatible architectures. A message at the bottom says "There is no data to display." At the top right, there are buttons for "Create layer" and "Import layer".

así nos debería de quedar todo



The screenshot shows the "Layer configuration" dialog for creating a new layer named "python". The "Name" field is filled with "python". The "Description - optional" field has "Description" in it. Under "Upload", the "Upload a .zip file" radio button is selected, and a file named "python.zip" (127.64 KB) is chosen. Below this, a note says "For files larger than 10 MB, consider uploading using Amazon S3." Under "Compatible architectures - optional", there are checkboxes for "x86_64" and "arm64", neither of which is checked. Under "Compatible runtimes - optional", there is a dropdown menu set to "Runtimes" with a "C" icon. A list of runtimes is shown: Python 3.12, Python 3.10, Python 3.11, Python 3.8, and Python 3.9. Below this, there's a "License - optional" section with a text input field and a note about MIT License. At the bottom right, there are "Cancel" and "Create" buttons, with "Create" being highlighted.

Volvemos a Lambda en el panel de la izquierda y vamos a agregar la layer, scroll hasta el final de la pagina que esta el bloque de **Layer** -> seleccionar “add layer” -> de tipo “custom layer” en las opciones -> y en el desplegable seleccionar la layer creada y la versión correspondiente (se verifica en la sección Layers).

Vamos a la tab “**Configuration**” -> y a la izquierda esta la opcion de variables de entorno -> editar -> add -> vamos agregar las key value:

USER_NAME : admin

PASSWORD : qwerty.1234

RDS_PROXY_HOST : (copiar el del proxy creado)

DB_NAME : ExampleDB

The screenshot shows the AWS Lambda Configuration page. The left sidebar has a 'Environment variables' section selected. The main area displays a table for environment variables. A single row is present with the key 'No environment variables' and the value 'No environment variables associated with this function.' An 'Edit' button is located at the bottom right of the table.

Asi debería verse después de cargar las variables

The screenshot shows the AWS Lambda Configuration page with four environment variables added: DB_NAME (Value: ExampleDB), PASSWORD (Value: test1234), RDS_PROXY_HOST (Value: proxy-1709594549973-mysqlforlambda.proxy-chyww6aylh32.us-east-1.rds.amazonaws.com), and USER_NAME (Value: admin). The table structure is identical to the previous screenshot, but now it lists the four new variables.

Vamos a darle a TEST y debería correr.

```

Function Log
[INFO] 2024-03-04T23:40:59.02Z SUCCESS: Connection to RDS for MySQL instance succeeded!
START RequestId: 869d3fb8-c220-47c8-97ff-d9fc40d39c00 Version: $LATEST
[INFO] 2024-03-04T23:40:59.078Z 869d3fb8-c220-47c8-97ff-d9fc40d39c00 The following items have been added to the database:
[INFO] 2024-03-04T23:40:59.078Z 869d3fb8-c220-47c8-97ff-d9fc40d39c00 (2021, "Martha Rivera")
END RequestId: 869d3fb8-c220-47c8-97ff-d9fc40d39c00 Duration: 52.38 ms Billed Duration: 53 ms Memory Size: 128 MB Max Memory Used: 44 MB Init Duration: 345.38 ms
REPORT RequestId: 869d3fb8-c220-47c8-97ff-d9fc40d39c00

```

En la imagen se ve el “SUCCESS” y dos líneas debajo los datos agregados a la base.

4 Crear SQS

Buscamos en la barra de búsqueda “SQS o Simple Queue Service”.

Una vez dentro del servicio le damos a “Create Queue” -> le damos un nombre “LambdaRdsQueueMundose” -> lo demás todo por defecto, vamos al final de la página y damos a **Create Queue**.

Una vez creada la SQS deberemos de volver a ingresar al servicio de Lambda -> en la solapa de “Configuration” (misma solapa donde agregamos las variables de entorno) -> seleccionamos ahora “Permissions” -> hacemos clic en el nombre del “Role name” (hipervínculo en azul de nombre [LambdaFunctionRDSmundose-1718924159642](#) en mi caso)

General configuration	Execution role
Triggers	Role name LambdaFunctionRDSmundose-1718924159642
Permissions	Resource summary
Destinations	To view the resources and actions that your function has permission to access, choose a service.
Function URL	

Esto nos llevará directamente al panel de IAM, dentro de los permisos de la Lambda creada

Aquí seleccionamos “Add permissions” en -> Attach policies

en el buscador ponemos “awslambdasqs” -> seleccionamos la unica que aparecerá -> **Add permissions**

ahora nos deberia quedar asi

IAM > Roles > LambdaFunctionRDSmundose-1718924159642

LambdaFunctionRDSmundose-1718924159642 [Info](#)

[Delete](#)

Summary		Edit
Creation date	June 20, 2024, 19:56 (UTC-03:00)	ARN
Last activity	1 hour ago	arn:aws:iam::083911806224:role/service-role/LambdaFunctionRDSmundose-1718924159642
		Maximum session duration
		1 hour

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

Filter by Type		
<input type="text"/> Search	All types	C Simulate Remove Add permissions
<input type="checkbox"/> Policy name AWSLambdaSQSQueueExecutionRole	Type	1
<input type="checkbox"/> AWSLambdaVPCAccessExecutionRole	AWS managed	2

Luego de configurar el role damos click en “Add trigger”

▼ Function overview [Info](#)

[Diagram](#) [Template](#)

 **LambdaFunctionRDSmundose**

 Layers (1)

[+ Add trigger](#) [+ Add destination](#)

Dentro del trigger seleccionamos SQS como servicio de trigger -> En “SQS queue” seleccionamos la queue que acabamos de crear -> Tildamos “Activate trigger” -> en “Batch size - optional” colocamos 1 como valor (esto es solo para la concurrencia, si deseamos activarla dejamos un valor mayor) -> **Add**

Debería quedarnos tal como la imagen de abajo

SQS  aws event-source-mapping polling queue

SQS queue
Choose or enter the ARN of an SQS queue.

X C

Activate trigger
Select to activate the trigger now. Keep unchecked to create the trigger in a deactivated state for testing (recommended).

Batch size - optional
The number of records in each batch to send to the function.

The maximum is 10,000 for standard queues and 10 for FIFO queues.

Batch window - optional
The maximum amount of time to gather records before invoking the function, in seconds.

When the batch size is greater than 10, set the batch window to at least 1 second.

Maximum concurrency - optional
The maximum number of concurrent function instances that the SQS event source can invoke.

Specify a value between 2 and 1000. To deactivate, leave the box empty. 

Report batch item failures - optional
Allow your function to return a partial successful response for a batch of records.

Filter criteria - optional
Define the filtering criteria to determine whether or not to process an event. Each filter must be in a valid JSON format in filter rule syntax. Lambda processes an event if any one of the filters are met. Otherwise, Lambda discards the event. [Learn more](#) 

During trigger creation, Lambda translates your filter(s) into a single JSON structure containing all your filtering criteria.

In order to read from the SQS trigger, your execution role must have proper permissions.

Cancel Add

Luego de esto deberíamos quedar con una pantalla similar a esta

LambdaFunctionRDSmundose

Description
-

Last modified
2 hours ago

Function ARN
[arn:aws:lambda:us-east-1:083911806224:function:LambdaFunctionRDSmundose](#)

Function URL [Info](#)
-

Code | Test | Monitor | **Configuration** | Aliases | Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

Triggers (1) [Info](#)

Find triggers

Trigger

SQSLambdaMundose
arnaws:sqscus-east-1:083911806224:SQSLambdaMundose
state: Enabled

+ Add trigger

Una vez que completamos esto en el panel de Lambdas podemos hacer clic en el hipervínculo que figura en la imagen como “SQS: **SQSLambdaMundose**” o buscar el servicio de SQS y entrar a la queue creada.

Una vez dentro de la queue debemos seleccionar “**Send and receive messages**”

Amazon SQS > Queues > SQSLambdaMundose

SQSLambdaMundose

Edit | Delete | Purge | **Send and receive messages** | Start DLQ redrive

Details [Info](#)

Se nos abrirá una pantalla como la de la imagen allí lo único queharemos será cargar en formato json un body de lo que deseamos agregar a la RDS, como se ve en la imagen, luego **Send message**

Amazon SQS > Queues > SQSLambdaMundose > Send and receive messages

Send and receive messages

Send messages to and receive messages from a queue.

Send message Info

Clear content **Send message**

Message body
Enter the message to send to the queue.

```
{
    "CustID": 1058,
    "Name": "Vin Diesel"
}
```

Delivery delay Info

0 ▾
Should be between 0 seconds and 15 minutes.

► **Message attributes - Optional** Info

Para verificar la corrida de la SQS deberemos ingresar al servicio de “CloudWatch” (por el buscador) -> en el panel de la izquierda seleccionar “**Log groups**” -> abrir el log group de nombre /aws/lambda/LambdaFunctionRDSmundose (o el nombre que le hayan colocado) -> entrar al “**Log streams**” que figura arriba de todos -> allí se ven las corridas hacia la RDS por parte de la Lambda.

En este caso se ven todas las que hicimos manualmente en la clase más el ejemplo con la queue al final de la imagen.

CloudWatch

Favorites and recents

Dashboards

Alarms

Logs

Log groups

Log Anomalies

Live Tail

Logs Insights

Contributor Insights

Metrics

X-Ray traces

Events

Application Signals New

Network monitoring

CloudWatch > Log groups > /aws/lambda/LambdaFunctionRDSmundose > 2024/06/21/[\${LATEST}]764f92d25a1a45e294b9d4cd560ff2eb

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Filter events - press enter to search

Timestamp	Message
2024-06-21T03:08:06.835Z	INIT_START Runtime Version: python:3.12.v28 Runtime Version ARN: arn:aws:lambda:us-east-1:runtime:7776dee4c90ffff4d4bf833b78d94e5b4148d14d044b867a14756f301ecfe1e28
2024-06-21T03:08:07.019Z	[INFO] 2024-06-21T03:08:07.019Z SUCCESS: Connection to RDS for MySQL instance succeeded
2024-06-21T03:08:07.022Z	START RequestId: cbd06cc08-ac00-5e03-8cc0-32465cd0f005 Version: \$LATEST
2024-06-21T03:08:07.045Z	[INFO] 2024-06-21T03:08:07.044Z cbd06cc08-ac00-5e03-8cc0-32465cd0f005 The following items have been added to the database:
2024-06-21T03:08:07.045Z	[INFO] 2024-06-21T03:08:07.045Z cbd06cc08-ac00-5e03-8cc0-32465cd0f005 (1021, 'Martha Rivera')
2024-06-21T03:08:07.045Z	[INFO] 2024-06-21T03:08:07.045Z cbd06cc08-ac00-5e03-8cc0-32465cd0f005 (1054, 'Richard Roe')
2024-06-21T03:08:07.045Z	[INFO] 2024-06-21T03:08:07.045Z cbd06cc08-ac00-5e03-8cc0-32465cd0f005 (1055, 'Silvester Stallone')
2024-06-21T03:08:07.045Z	[INFO] 2024-06-21T03:08:07.045Z cbd06cc08-ac00-5e03-8cc0-32465cd0f005 (1058, 'Vin Diesel')

Actions ▾ Start tailing Create metric filter

Con esto se completa el práctico de RDS/Lambda/SQS.

Para cerrar no olvidar de **BORRAR TODO..!!!!**