Contenido

[DATA INGESTION PIPELINE 1](#_Toc132134711)

[Python Client 1](#_Toc132134712)

[Lambda function that writes into Kinesis 1](#_Toc132134713)

[Create the API Gateway 6](#_Toc132134714)

[Setup Kinesis 10](#_Toc132134715)

[STREAM TO RAW STORAGE PIPELINE 11](#_Toc132134716)

[Setup S3 11](#_Toc132134717)

[Lambda function that writes into S3 12](#_Toc132134718)

[STREAM TO DYNAMODB PIPELINE 15](#_Toc132134719)

[Setup DynamoDB 15](#_Toc132134720)

[Customer Table 15](#_Toc132134721)

[Invoice Table 17](#_Toc132134722)

[Lambda function that writes into DynamoDB 18](#_Toc132134723)

[API PIPELINE TO QUERY DATA FROM DYNAMODB 21](#_Toc132134724)

# DATA INGESTION PIPELINE

Texto

Descripción generada automáticamente con confianza media

In this first step I’m going to send the data through a Python client, that will simulate the streaming of data. The client will send the data into the API Gateway and from there, a lambda function will be in charge of sending the body into Kinesis Stream.

## Python Client

The Python client will make use of pandas, to manage the data of a csv file (downloaded from Kaggle), and the requests library, to make HTTP requests. It will take each row of data from the csv file, convert it into a JSON object, and send it into the API Gateway.

The URL variable will hold the path to the API stage that I will create in a following step.

## Lambda function that writes into Kinesis

Prior to creating the API Gateway, I am going to create the Lambda function, since its necessary to define where the API Gateway will send the data into.

To create a new Lambda function, simply look for Lambda in the search pane in AWS and click on “Create function”:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

I will specify the name of the function and select Python as the language it will use:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

I will leave AWS to create a new role for this function, and later I will assign the permissions it needs to write into Kinesis:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente

The execution role will be named “*WriteIntoKinesis-role-gxrc5tbs*”.

After that, I just click on “Create function” at the bottom right side, and the lambda function would be created.

The next step is to ensure that the Lambda function can effectively write data into Kinesis. For that, I would assign the permissions needed for that in the Identity and Access Management (IAM). The easiest way to do this is navigating through the Lambda function, in the “Configuration” tab and clicking in the “Permissions” section and after that on the role name that has been assigned to this lambda function.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

This will take us directly to the IAM console, where we can attach the policies we need. In this case I will click on “Policies” on the left hand side to create a new policy for this function:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

In the menu, I click on the upper hand side button “Create policy”:Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Once there, I will write the following:

* Service: Kinesis.
* Actions: Write.
  + PutRecord.
  + PutRecords.
* Resources: All resources (here I should specify the ARN of the Kinesis Data Stream, but since it hasn´t been created yet I will leave it like this).
* Request conditions: (default).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

In this case, I´m only allowing this policy to write into Kinesis and it can only perform two functions: write a single record or writing various records.

Next step is to add the tags, but since this is optional and not needed I won´t add anything here.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Last step is to assign a name and a description to the policy. I will assign the same name as for the Lambda function and a short description:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

After creating the policy, I will assign it to the role that was created for the Lambda function. For that, I come back to “Roles” in the left hand side menu and select the “WriteIntoKinesis-role-gxrc5tbs”:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Once there, click on “Add permissions” and “Attach policies”:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Select the one already created and attach it to the role:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

(The code for this function is in the Lambda-WriteIntoKinesis.py file)

## Create the API Gateway

I look for API Gateway and I create a new REST API:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

I select the following configuration and give it a name and a description and click on Create:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

The first thing to do now is to create the resource:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

I will just give it a name and AWS will automatically assign a path:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Next step is to create the methods that we will need, in this case I will create a GET, POST and PUT methods:

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Integration type would be Lambda function and after that simply write the name of the Lambda function that I created before and click on Save:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

After that, AWS will pop up a message with the ARN that has been assigned to this API.

One thing that is important, since we are dealing with json objects, we need to go to the Integration Request, and in the last part, where it says “Mapping Templates”, write the content that we are expecting (in this case application/json):

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

After clicking on the create button, AWS will pop up a message to secure this integration:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

The last thing here is to generate the template, for that simply choose from the list the option “Method Request passthrough”:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

AWS will generate the code and the only thing left is to click on “Save”.

Now, in order to be able to communicate with the API, it is needed to deploy it. For that, I’ll click on “Deploy API” in the “Actions” menu:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

I´ll just give it a name and click on “Deploy”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

When the stage have been created, it is possible to send and receive requests from the API through the invoke URL:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

This URL is used in the code in the lambda function.

## Setup Kinesis

Look up for “Kinesis” in AWS and create a Kinesis data stream:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Here, I just simply give it a name, “APIDataStream”:

Escala de tiempo

Descripción generada automáticamente con confianza media

And select the capacity for this stream as provisioned with one shard, since I am not expecting the data load to increase and hence, there is no need to write and read faster:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# STREAM TO RAW STORAGE PIPELINE

The idea behind this pipeline is that, each and every time new data arrives to the Kinesis data stream, a Lambda function is going to be triggered and it will store the data in the S3 bucket. In order not to create a lot of files in the S3 bucket, instead of immediately triggering the Lambda function each time there is new data in the data stream, I will make the Lambda function to wait for some time, so it waits for some more data and insert all of them at once into the S3 bucket.

Texto

Descripción generada automáticamente con confianza baja

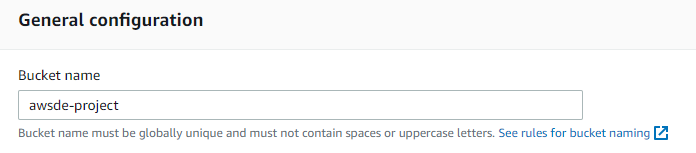
## Setup S3

To store the information that is coming from Kinesis I will create a new bucket in S3. Again, I look for S3 in the search bar and after selecting S3, I click on “Create bucket”:

Diagrama

Descripción generada automáticamente con confianza baja

Assign it a name, leave the rest of the settings as default and create the bucket:



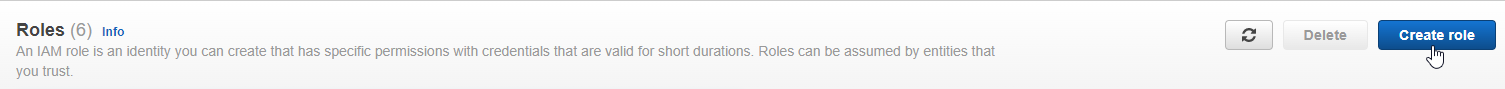
## Lambda function that writes into S3

The lambda function needs to be able to send data to the bucket that I already created and for that, it will need to have the permissions required for that task. So I begin by creating a specific role for this purpose in the Identity and Access Management (IAM) console. For that I navigate to the IAM and I click on “Roles” under the “Access management” section:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

And click on “Create role” in the top right of the window:



Once there I select “Lambda” as the use case:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

And click on Next.

In the following window I will add the permissions. I will grant the following:

* AmazonS3FullAccess (to keep things simple).
* Permission to read from Kinesis: creating a new policy for this purpose (MyKinesisRead), with the following specifications:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Lastly, I will name the role “KinesisToS3”.

Now, I can create the new Lambda function that will be in charge to read from Kinesis and write into S3. In this case, instead of writing all from scratch, I will make use of a blueprint to generate some lines of code:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

I will name this function “WriteIntoS3” and assign it the role I just created:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Last thing in here is to select the trigger, that in this case will be the Kinesis Data Stream that I have just created before:

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

I will leave the rest of the configuration as default.

(The code for this function is in the Lambda-WriteIntoS3.py file)

# STREAM TO DYNAMODB PIPELINE

The next pipeline is the one in charge of writing the data into DynamoDB. As with the previous one, each time we have new data into Kinesis, a lambda function will be triggered. This lambda function will process the data to reformat it according to the way we are storing and querying it later in DynamoDB.

Diagrama

Descripción generada automáticamente

## Setup DynamoDB

To prepare the environment in DynamoDB first I´m going to create the two tables that will be needed in this project. For that, I search for DynamoDB in AWS and, in the main page, I click on the “Create table” button:

Texto

Descripción generada automáticamente con confianza media

### Customer Table

First, I will create the Customer table. I am going to give it a name and assign the partition key:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Another important thing is to change the Capacity mode from “On-demand” to “Provisioned” and change the “Read capacity” to set the “Auto scaling” to off, since we are not expecting our incoming data stream to increase.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

### Invoice Table

Same as before, I will create the Invoice table, this time the partition key will be of type string:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

I will select the same configuration as before and click on “Create Table”.

## Lambda function that writes into DynamoDB

In this case I will create the function first and let AWS create a role for it and, after that, assign the policies needed to write into DynamoDB to it. So first, I go to Lambda and create a new function, the same way I did before, since I want this function to be triggered also when there is new data coming from Kinesis. So I select the same blueprint as before:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

I will name this function and its role “WriteIntoDynamoDB” and create the lambda function.

(The code for this function is in the Lambda-WriteIntoDynamoDB.py file)

After creating the lambda function, I go to the IAM to assign the permissions to this function for it to be able to listen from Kinesis and write into DynamoDB. In the IAM, I click on Roles and then into the role I created for this function.

I create a new policy to allow the following access levels:

* List:
  + ListTables.
* Write:
  + PutItem (to insert a new customer).
  + UpdateItem (to update the row if a customer is already present).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

As with the other cases I will keep it simple allowing all resources. Lastly, I will name this policy “DynamoDBAccess” and click on “Create Policy”. Once this is created, I will attach this new policy and the one that allows reading from Kinesis to the role for this lambda function:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

After that, it is possible to add the trigger and let this function listen to Kinesis:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# API PIPELINE TO QUERY DATA FROM DYNAMODB

Now that the data resides in DynamoDB, I will create a new pipeline that will be used to query that data. The objective is getting all the items of an invoice number, providing this invoice through a client, or all the invoices that a customer has bought, providing the customer number.

The pipeline would look as follows:

Texto, Pizarra

Descripción generada automáticamente

In this case, the client would be requesting all the items for an invoice number and a lambda function is being triggered and queries DynamoDB with this InvoiceNo.

## API Gateway and Lambda function

I am going to re-use the API and Lambda function that I created for the ingestion pipeline in the first step. Now is where the “GET” method comes into play. But, in order for the pipeline to be able to read data from DynamoDB, I need to assign it the policies needed for thar purpose.

For that, I go to Identity and Access Management and I create a new policy called “ReadDynamoDB” with the following settings:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

This time I added the specific ARNs so this policy can only read from the tables I specified.

Once the policy is created, I attach it to the role of the lambda function that I used in the ingestion pipeline (“WriteIntoKinesis-role-gxrc5tbs”) and I´m going to test its behavior using postman.

## Testing the pipeline with Postman

Before using the client to ask for the data stored in DynamoDB, I am going to test that everything is correct using postman. For this, I need to go to API Gateway service and take the invoke URL that it´s under “Stages” and then clicking in one of the methods that are there:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Paste that URL in the requested URL in postman and fill in the key and value fields:

Captura de pantalla de un celular

Descripción generada automáticamente

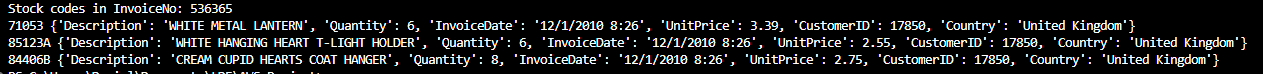
Now, I will check the Invoices, given a CustomerID:

Captura de pantalla de computadora

Descripción generada automáticamente

## Retrieve the data with the client

Now that I know that the pipeline works, I will use the client to introduce the Invoice number I´m interested in getting and retrieve its elements:



The code for this function is on the file “ReadDatafromDynamoDB.py”.