Research Project SEIS 776/777

# Xavier Porter

# Prof. Chiang

# April 26, 2022

The emphasis on this project was to use AWS’s infrastructure to explore Amazon Textract and Amazon Rekognition and their effectiveness in reading text. Amazon Rekognition can detect text in images and videos. It can then convert the detected text into machine-readable text. To detect text in mages in JPEG or PNG format, the DetectText operation and GetTextDetection operations can detect words and lines.

Amazon Textract is a Machine learning service that automatically extracts text, handwriting, and data from scanned documents. Textract uses ML to read and process any type of document, accurately extracting text, handwriting, tables, and other data fairly easily. that uses a Text Detection API to detect a variety of documents including, medical records, tax forms and financial reports. Textract also provides synchronous and asynchronous operations for processing small, single-page documents at real time or a batch process that consists of larger, multipage documents. Below I will provide detailed instructions on how to implement AWS Textract with a pipeline architecture.

AWS Textract Pipeline Architecture Implementation

## Set up

* AWS Account
* Region Selection - US - East 1
* Provision AWS Resources via cloud formation: [Reinvent Stack](https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create/template?stackName=ReinventStack&templateURL=https://s3.amazonaws.com/files.reinvent.awsdemo.leodrak/reinvent.yaml)
* Reference github repo: <https://github.com/Porte221/research-project>

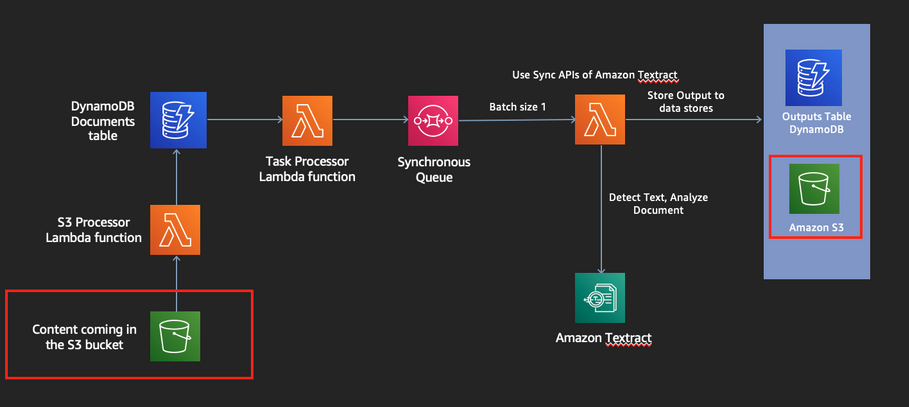
*Note: The Reinvent Stack will deploy the following resources:*

* Two IAM Roles that will be used for Synchronous processing
  + Task processor
  + SQS processor
* A Cloud9 environment which can be used to familiarize yourself with Amazon Textracts API and the deployment of resources for an asynchronous processing.Image

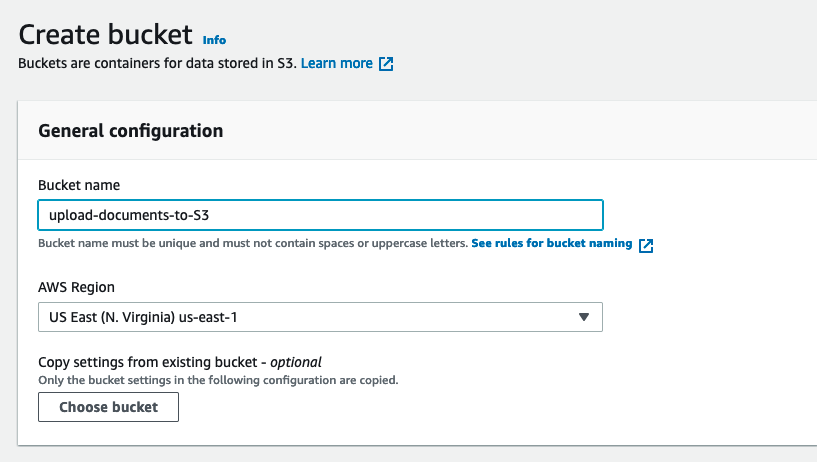
For synchronous document processing our pipeline will consist of 7 steps:

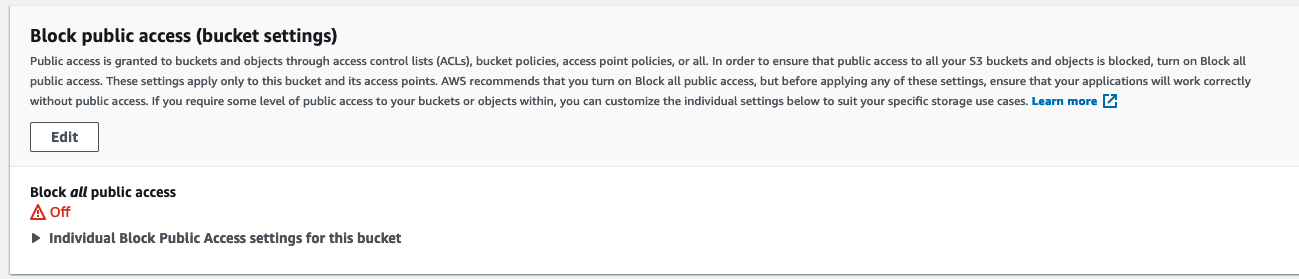
1. Create an S3 bucket to upload documents and store the output of the process.
2. Create a synchronous processing SQS queue. When a document is uploaded to the S3 bucket, we’ll put a message in our queue.
3. Create 2 DynamoDB tables. These will be used to store metadata for the documents processed.
4. Create a lambda function that will be triggered when a document is uploaded in the S3 bucket and create tasks in the documents DynamoDB table.
5. Create a lambda function that will be triggered as soon as a task is created in DynamoDB documents table and will place a message in our synchronous queue.
6. Create a lambda function that will process the uploaded document by picking up messages from the queue and will call the synchronous Textract API’s.
7. Lastly, you can test the pipeline by uploading an image to the S3 bucket and review the outputs.

What is an S3 Bucket?

S3 bucket is an object storage service that allows for scalability, data availability, security and performance. **<https://aws.amazon.com/s3/>**

## Creating an S3 Bucket

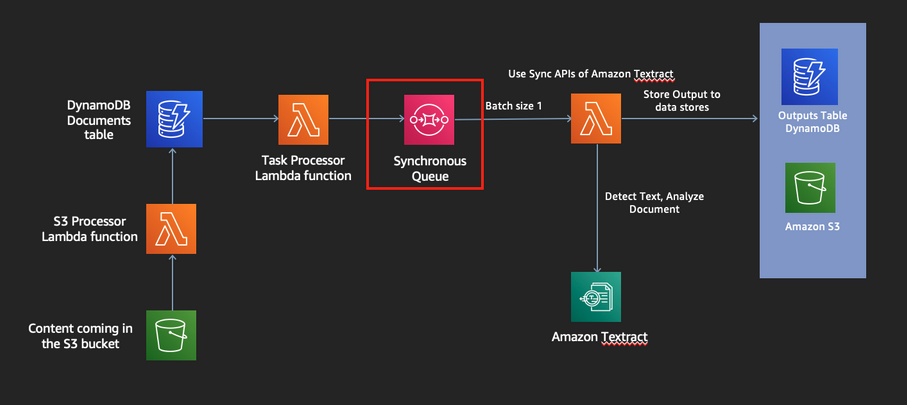
1. Log into your AWS service account.
2. Look up S3 under the finding services tab.
3. Select create the bucket button to start the wizard to create an S3 bucket.
4. Provide a name for your bucket along with a region (east-1) to align with the reinvent-stack we created previously.



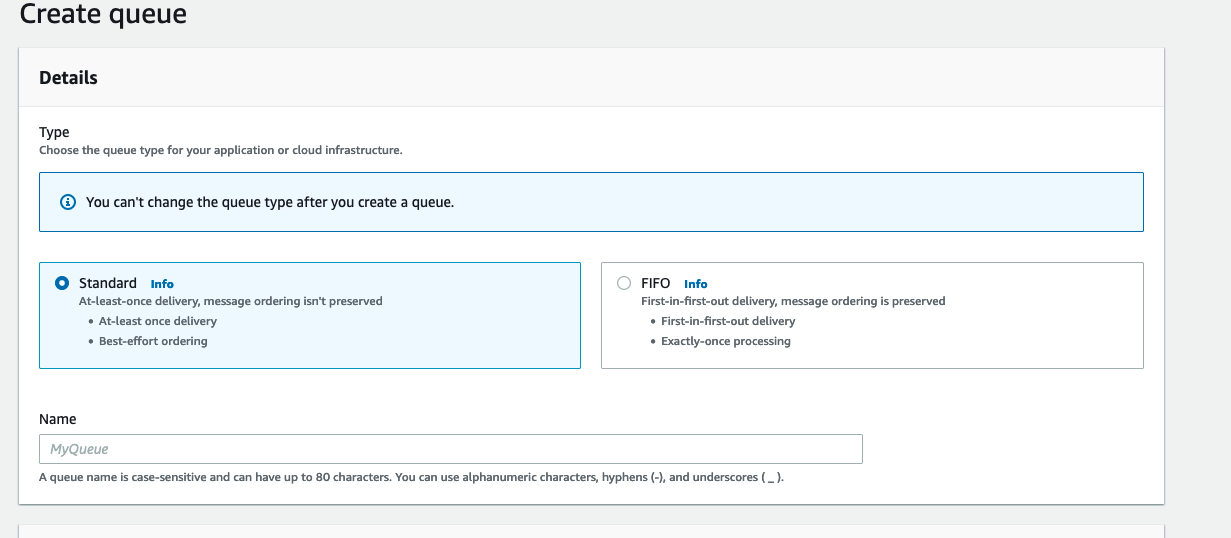
5. You can leave the block public access to off.

6. The other settings can be left to default. Select Create when finished.

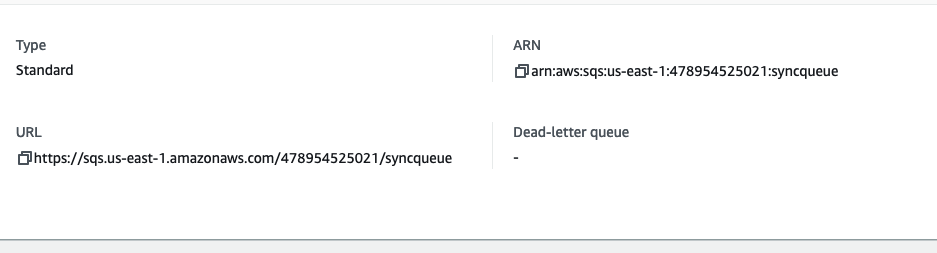
What is an SQS?

The SQS is a fully managed message queue service that enables you to decouple and scale micro services, distributed systems, and server-less applications. <https://aws.amazon.com/sqs/>.

## Creating an SQS

1. Make sure you logged into the appropriate region.
2. Search SQS in the AWS management console.
3. Select create the queue button to start the wizard to create an SQS queue.
4. Provide a name for your bucket along with a region (east-1) to align with the reinvent-stack we created previously.

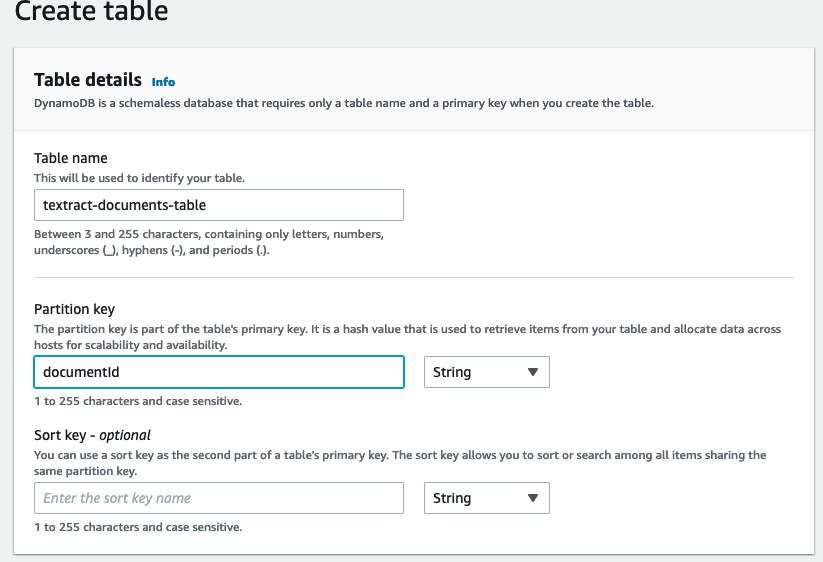
5. Once created you can leave the other settings to default. Make note of the Amazon Resource Names (ARNS) and URL, we’ll need these to specify our lambda trigger.



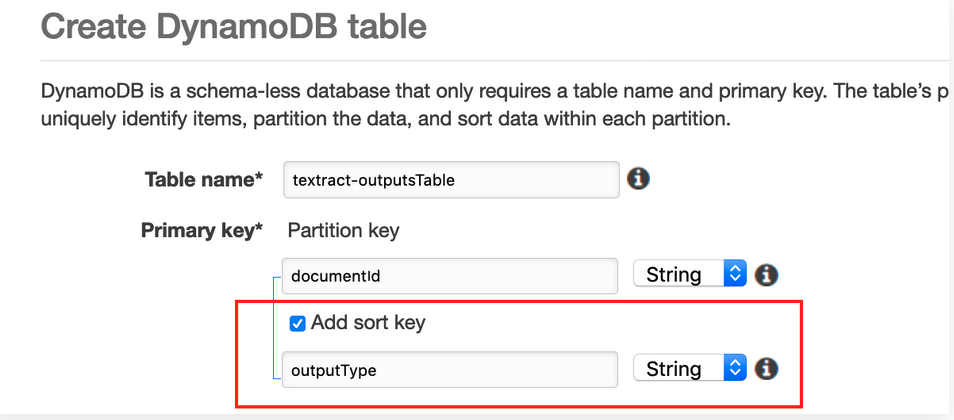
What is DynamoDB?

Amazon DynamoDB is a key-value and document database that delivers performance at any scale. <https://aws.amazon.com/dynamodb/>.

## Create the Amazon DynamoDB Tables

1. Make sure you are in the appropriate region.
2. Search for DynamoDB in the management console.
3. We will create 2 DynamoDB tables:
   1. Documents table: This table will store metadata on the documents that we upload to S3.
   2. Outputs table: This table will store metadata of the processed documents and outputs that Textract delivers.

For the document’s table, provide a partition or primary key to retrieve items in the table. Next select create table at the bottom of the page.

Repeat this same step to create the Outputs table.

Notes: Take a note of the ARN’s for both of these tables.

What is an AWS Lambda?

AWS Lambda lets you run code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. <https://aws.amazon.com/lambda/>.

## What is a Lambda Layer?

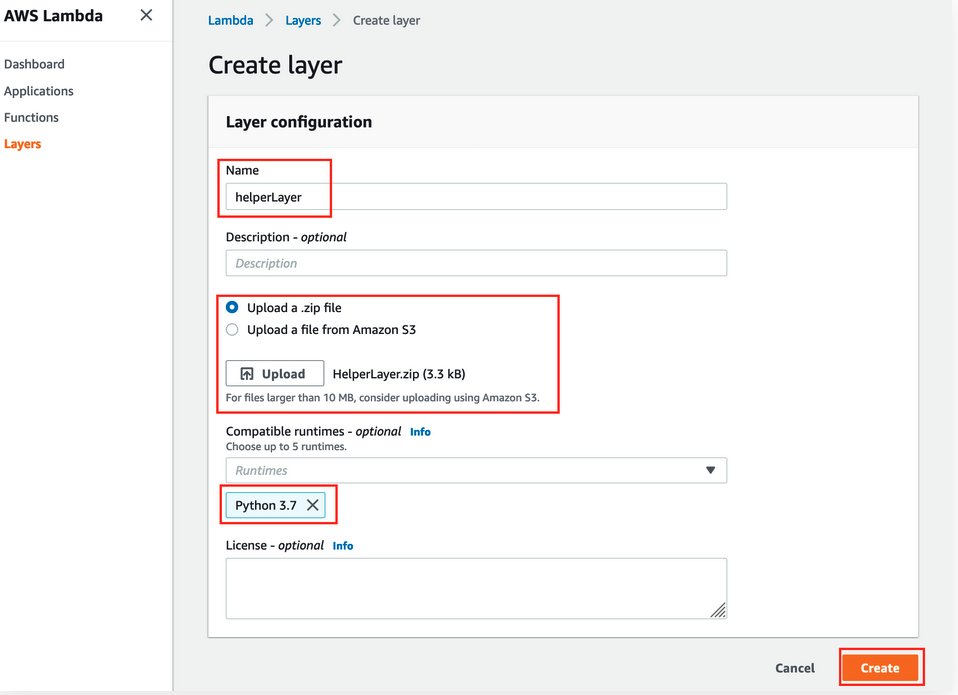
You can configure your lambda function to pull in additional code and content in the form of layers. A layer is a zip archive that contains other dependencies. By keeping these in smaller packages, it makes the deployment easier. <https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html>.

## Creating the Lambda Layers

To begin you will need these two layers to configure within your lambda functions, download these:

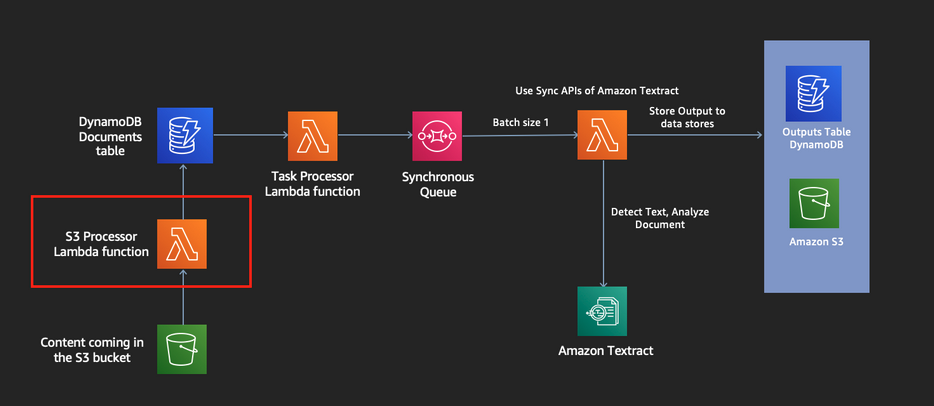
[HelperLayer.zip](https://master.d16gy3p1a3jqtn.amplifyapp.com/en/start-of-the-workshop/synchronous-processing/create-lambda-layers/HelperLayer.zip)

[Textractor.zip](https://master.d16gy3p1a3jqtn.amplifyapp.com/en/start-of-the-workshop/synchronous-processing/create-lambda-layers/Textractor.zip)

1. Make sure you are logged into the same region as your AWS resources.
2. Search for Lambda service in the AWS management console.
3. Click on layers and select create layers
4. Enter a name for your layer and upload the HelperLayer.zip and choose Python 3.7. Then choose create.
5. Repeat the same process to create another layer for the Textractor.zip. You will have both layers defined in the layer dashboard.

## Create a lambda function (S3 bucket trigger)

To create the lambda function that triggers when a document is uploaded to S3 there are 4 steps:

1. Create the Lambda execution IAM Role.
2. Create the Lambda that will trigger off the S3 Bucket
3. Configure an S3 trigger so that when a new file is uploaded to the S3 bucket the lambda function will trigger.
4. Add a lambda layer to the lambda function in order to include the appropriate dependencies.

What is an IAM Role?

IAM provides fine-grained access control across all of AWS. With IAM, you can control access to services and resources under specific conditions. Use IAM policies to manage permissions for your workforce and systems to ensure [least privilege](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege). <https://aws.amazon.com/iam/faqs/>

## Create an IAM Role

1. Make sure you are logged into the same region as your AWS resources.
2. Search for IAM in the AWS management console.
3. Once accessed select, Create Role and allow Lambda functions to call AWS. Select Next.

4. Under the permissions policy, search the **AWSLambdaBasicExecutionRole** and select Create Policy at the top right.

5. Copy the IAM Policy below and add your SQS and DynamoDB ARNs below

\*\*\* (REPLACE WITH YOUR OWN DOCUMENTS TABLE ARN, REPLACE WITH YOUR OWN SQS ARN) \*\*

{

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"dynamodb:BatchGetItem",

"dynamodb:GetRecords",

"dynamodb:GetShardIterator",

"dynamodb:Query",

"dynamodb:GetItem",

"dynamodb:Scan",

"dynamodb:BatchWriteItem",

"dynamodb:PutItem",

"dynamodb:UpdateItem",

"dynamodb:DeleteItem"

],

"Resource": [

"REPLACE WITH YOUR OWN DOCUMENTS TABLE ARN"

],

"Effect": "Allow"

},

{

"Action": [

"sqs:SendMessage",

"sqs:SendMessageBatch",

"sqs:GetQueueAttributes",

"sqs:GetQueueUrl"

],

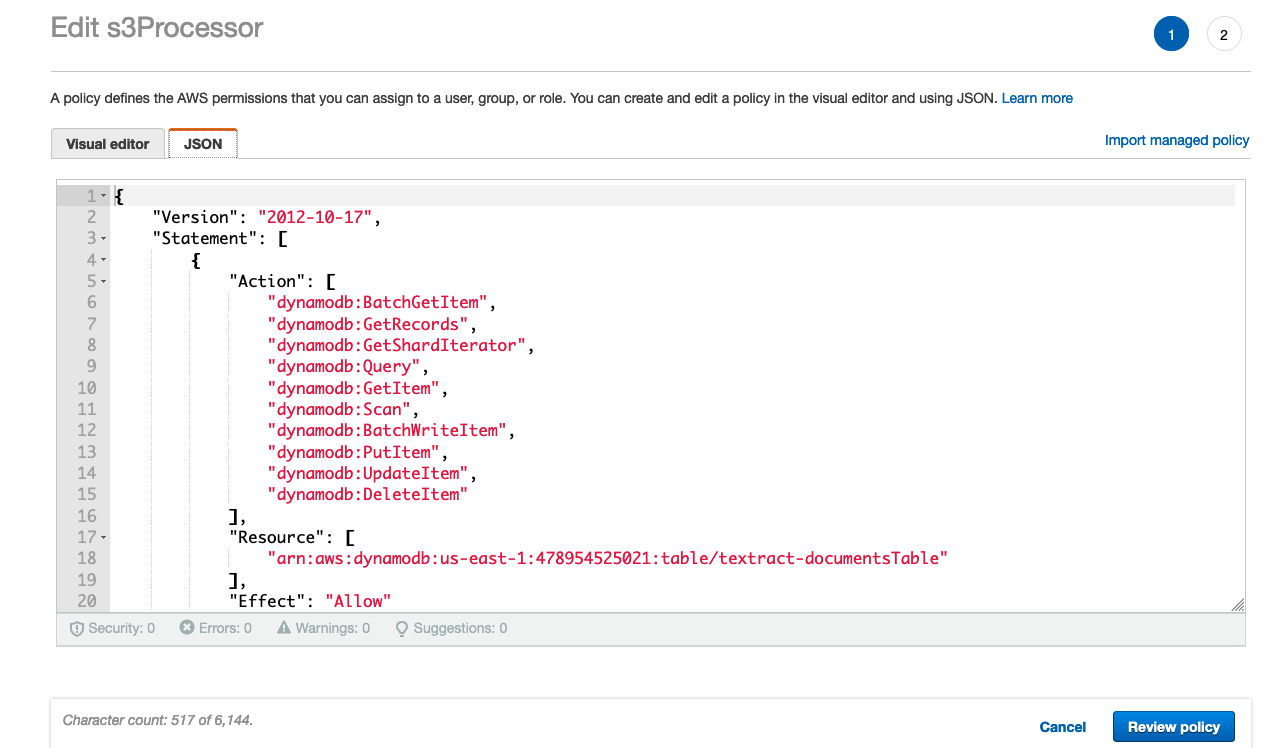
"Resource": "REPLACE WITH YOUR OWN SQS ARN",

"Effect": "Allow"

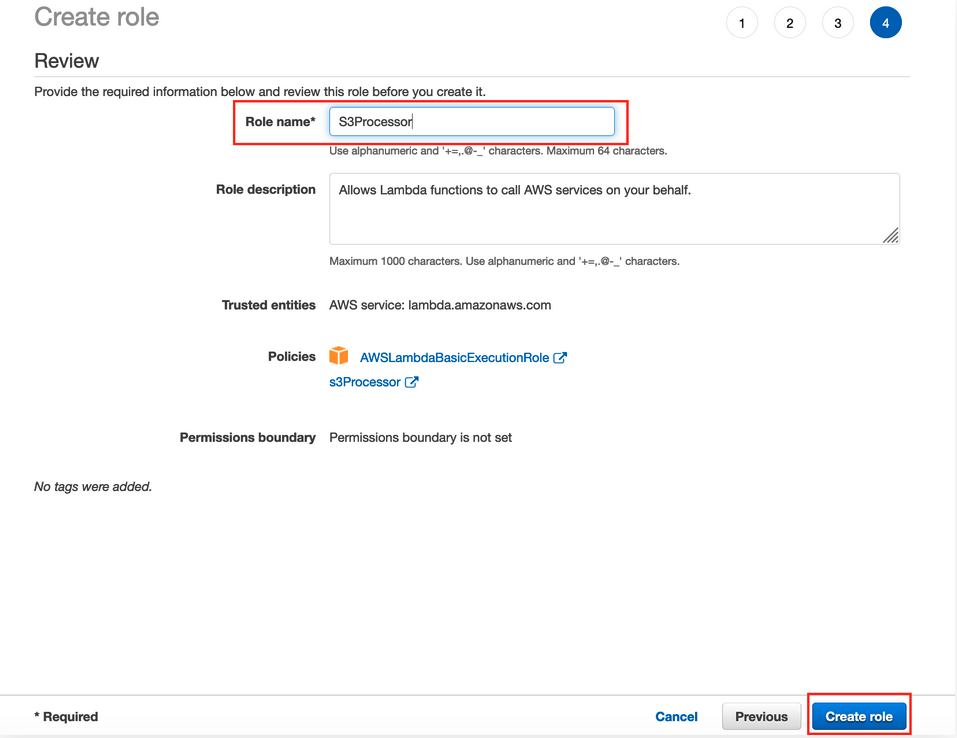
}

]

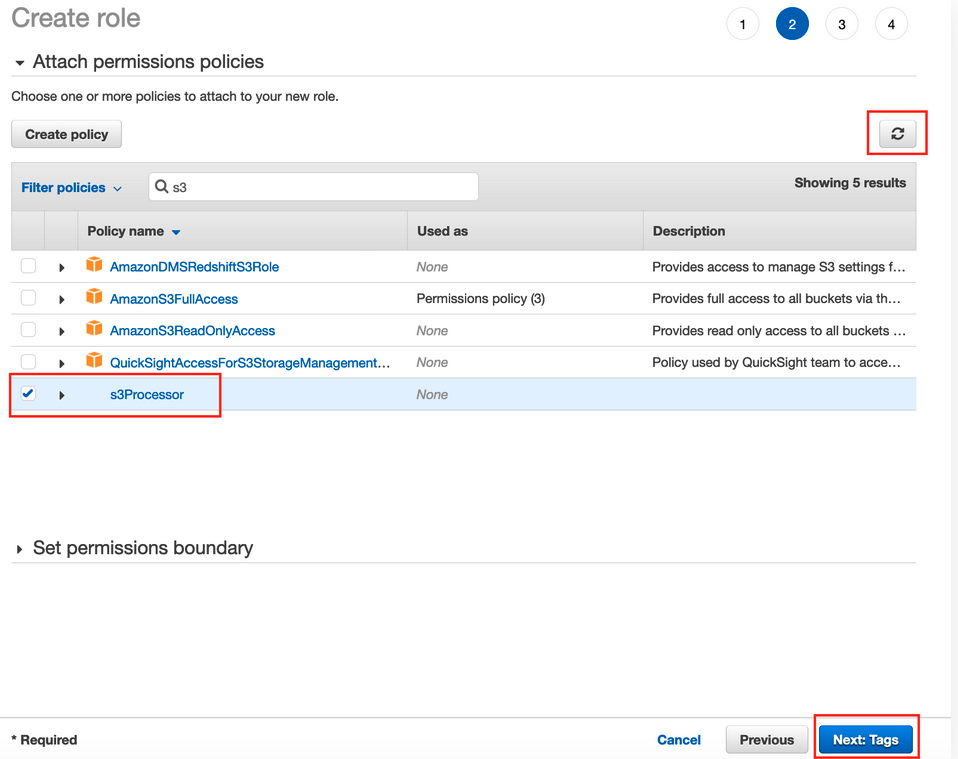
}

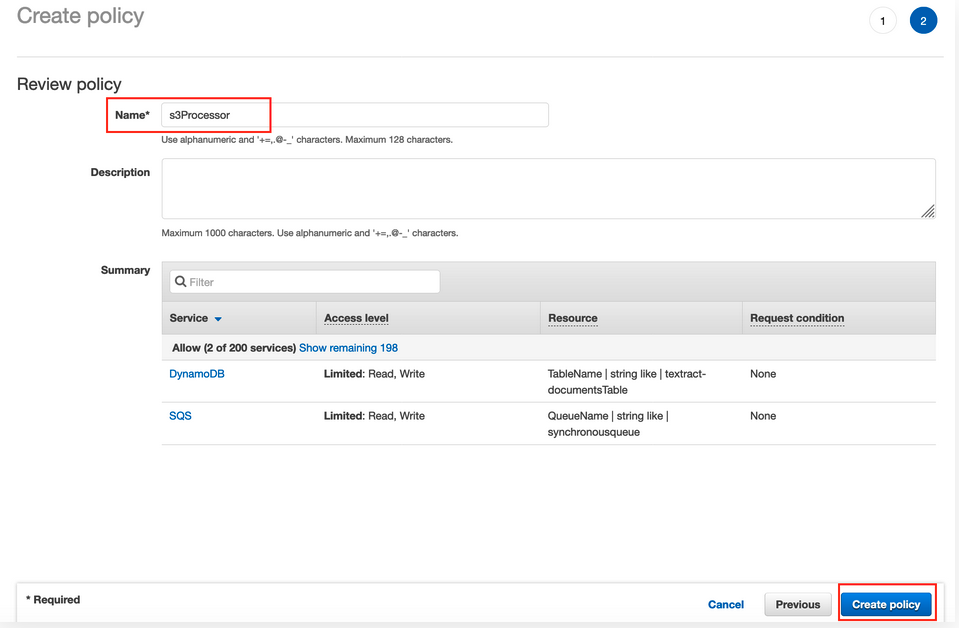


6. Paste the JSON similar to the right section.



7. Select Review Policy and provide the policy a name and select create.

8. As soon as you create the IAM policy go back and finish creating the IAM Role.



9. Find the new policy created and add the policy to the role. Lastly provide a new to the role and click create role.

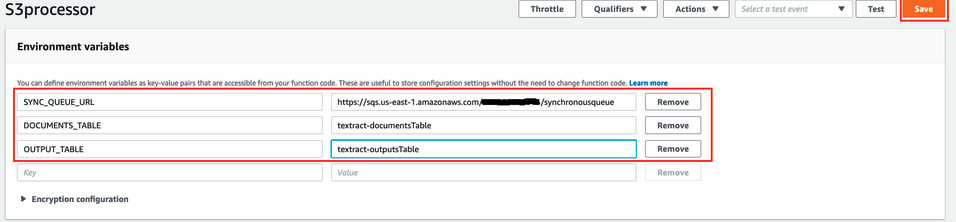
## Create the Lambda Function (S3 bucket trigger)

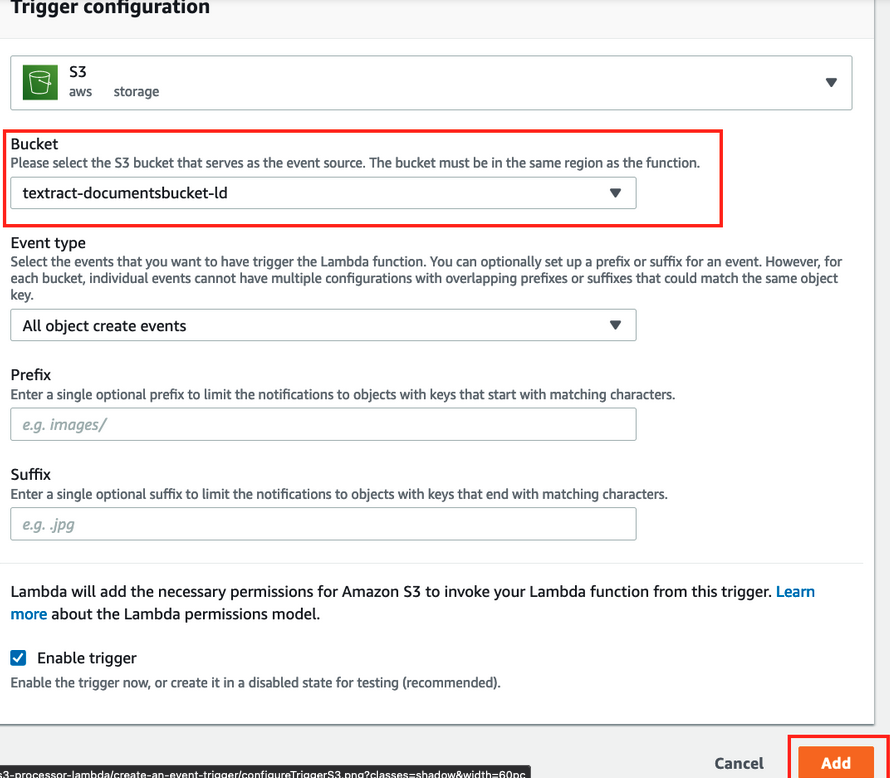
To create the lambda function that triggers when a document is uploaded to S3.

1. Make sure your in the appropriate region. Look for Lambda service under the AWS Management console.
2. Click Create function the screen.
3. Select the **3.7 python** runtime and configure the execution role to be the IAM Role you defined previously.
4. Select Create Function.

5. Copy the code from the [S3Processor lambda\_function.py](https://github.com/Porte221/research-project/blob/main/S3Processor/lambda_function.py) into the code section of the lambda function in AWS. The code will update the DynamoDB Documents table as soon as a document is uploaded to our S3 bucket.

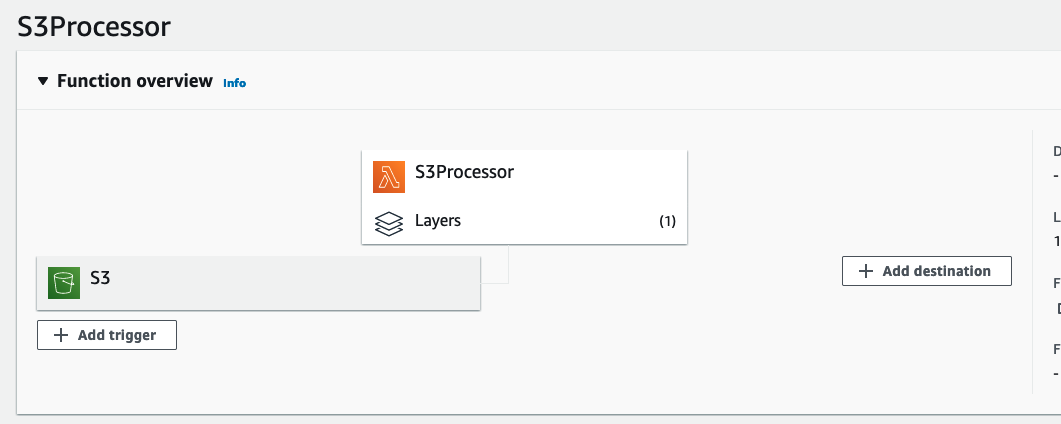
6. Now that we’ve added code to our lambda, we need to configure some environment variables for our lambda to identify our SQS endpoint and DynamoDB table. The environment variables we are going to configure are:

7. Configure those as shown below and click save.



## Create S3 Event Trigger

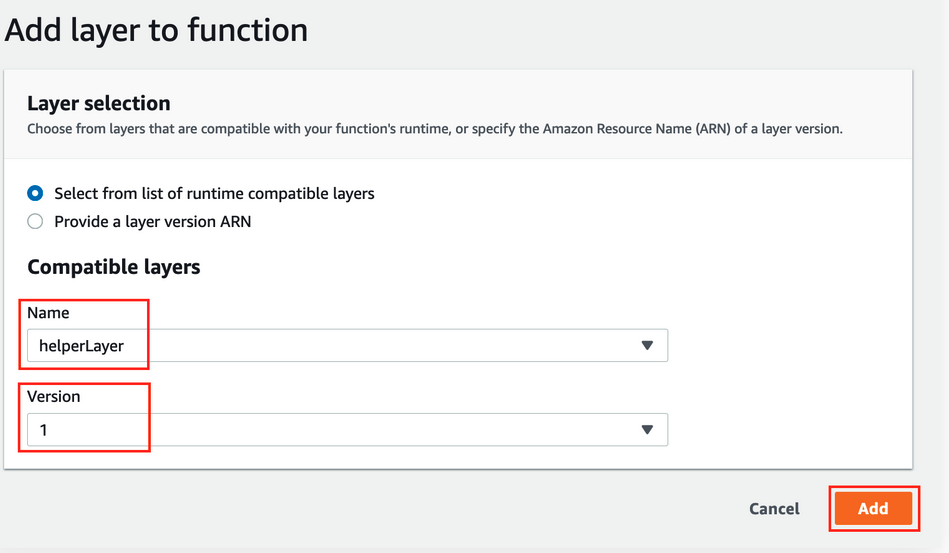
1. In the lambda service, select add trigger.
2. Configure your trigger to be the s3 bucket you created previously.



## Add a Lambda Layer

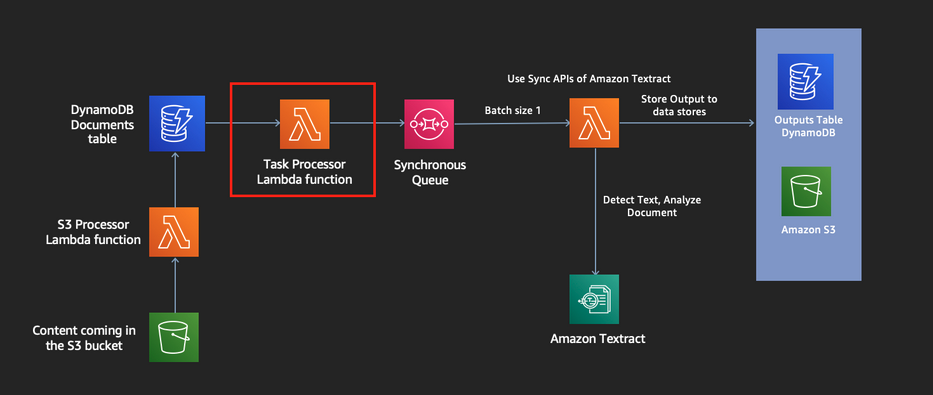
1. In the function overview, click add a layer.

2. Choose the HelperLayer then v1 and click add.



Create the Lambda Function (DynamoDB trigger)

To create the task processor lambda function that triggers as soon as a new task/row is inserted in the Documents DynamoDB table. There are 4 steps for this functionality.

1. You need an IAM role, generated from the Reinvent Stack that was deployed earlier.
2. The Lambda Function.
3. Configure a DynamoDB trigger so that when a task is created in the DynamoDB table the lambda function will trigger.
4. Add a lambda layer to include appropriate dependencies.

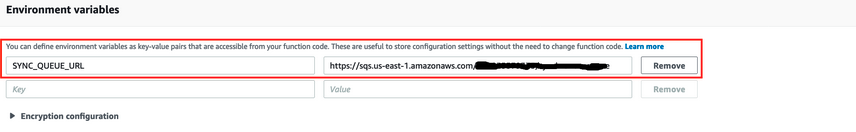
## Create the lambda function

1. Make sure your in the appropriate region. Look for Lambda service under the AWS Management console.
2. Click Create function the screen.
3. Select the **3.7 python** runtime and configure the execution role to be the TaskProcessor role defined in the Reinvent Stack.
4. Select Create Function.

5. Copy the code from the [TaskProcessor lambda\_function.py](https://github.com/Porte221/research-project/blob/main/TaskProcessor/lambda_function.py) into the code section of the lambda function in AWS. The code will be triggered as soon as a new row is added to the Documents DynamoDB table and will place a message in the synchronous SQS queue.

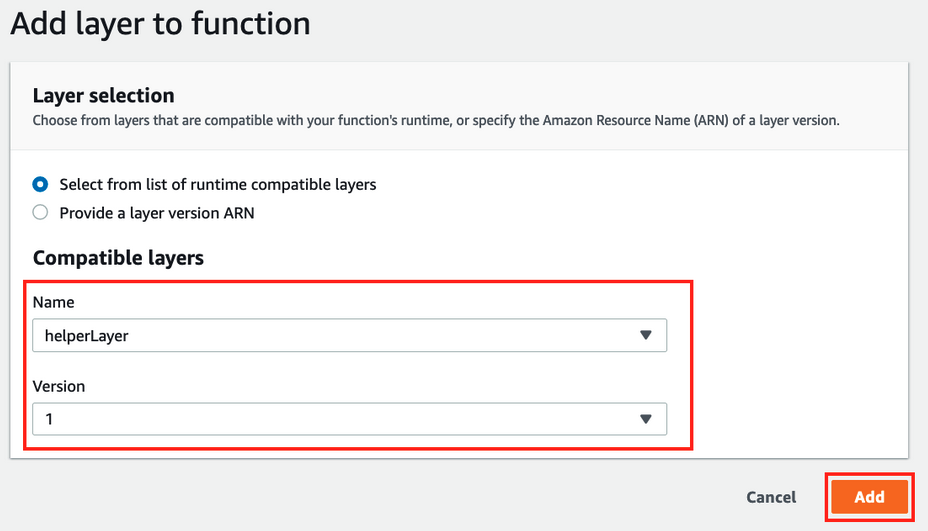
6. Now that we’ve added code to our lambda, we need to configure an environment variables in order for our Lambda function to be able to identify the appropriate SQS Queue to drop the message to. The environment variable we are going to configure is:Image

7. Configure the environment variable as shown below:



## Create the DynamoDBImage Trigger

1. In the lambda service, select add trigger.
2. Configure your trigger to be the DynamoDB you created previously.
3. Select add.

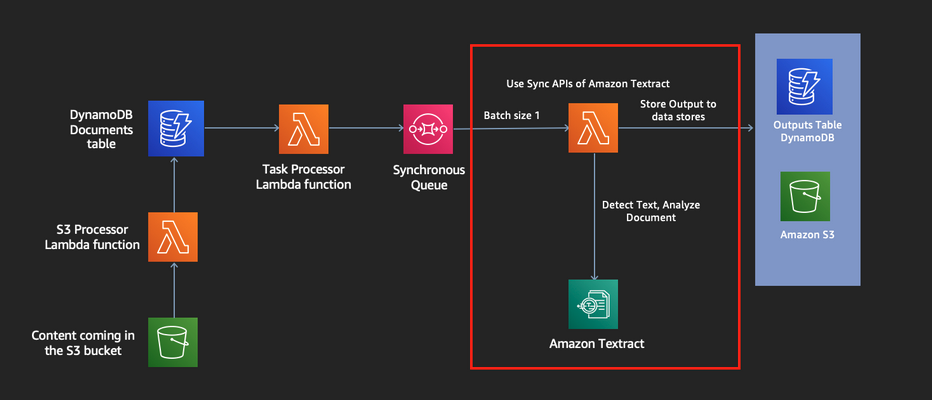


## Add the Lambda Layer

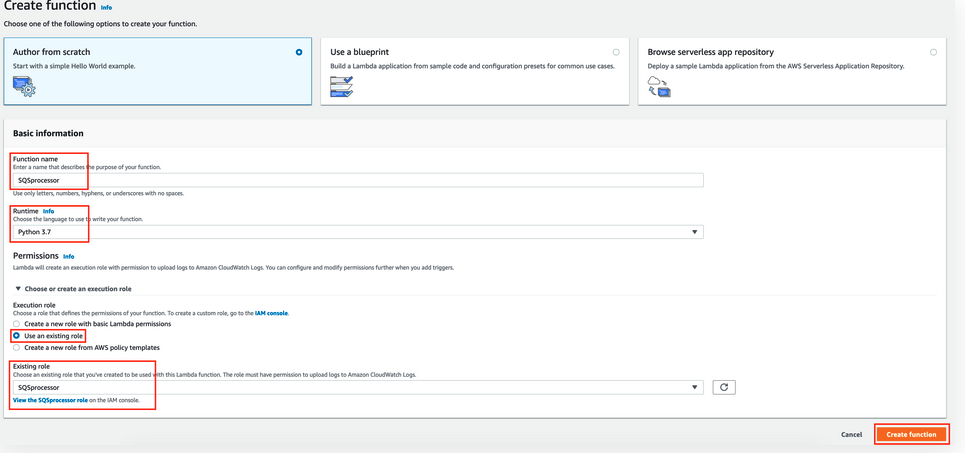
1. Select add a layer.
2. Choose the **HelperLayer** then **v1** and click Add.

Create SQS Processor Lambda

To create the lambda function that will be triggered as soon as a new message is placed in the Synchronous SQS queue.

1. You need an IAM role, generated from the Reinvent Stack that was deployed earlier.
2. The SQS processor Lambda Function.
3. Configure a SQS trigger so that when a message is placed in the SQS the lambda function will trigger.
4. Add a lambda layer to include appropriate dependencies.

## Create the lambda function

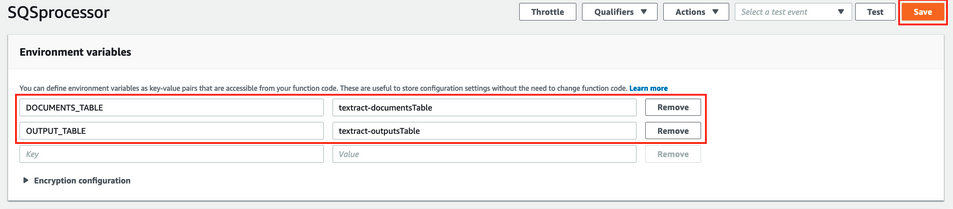
1. Make sure your in the appropriate region. Look for Lambda service under the AWS Management console.
2. Click Create function the screen.
3. Select the **3.7 python** runtime and configure the execution role to be the **SQS Processor** role defined in the Reinvent Stack.
4. Select Create Function.

5. Copy the code from the [SQSProcessor lambda\_function.py](https://github.com/Porte221/research-project/blob/main/SQSProcessor/lambda_function.py) into the code section of the lambda function in AWS. The code will pick up a message in the Sync SQS queue, call Textract to process a document and then update the DynamoDB Documents and Outputs table as soon as a document has been processed.

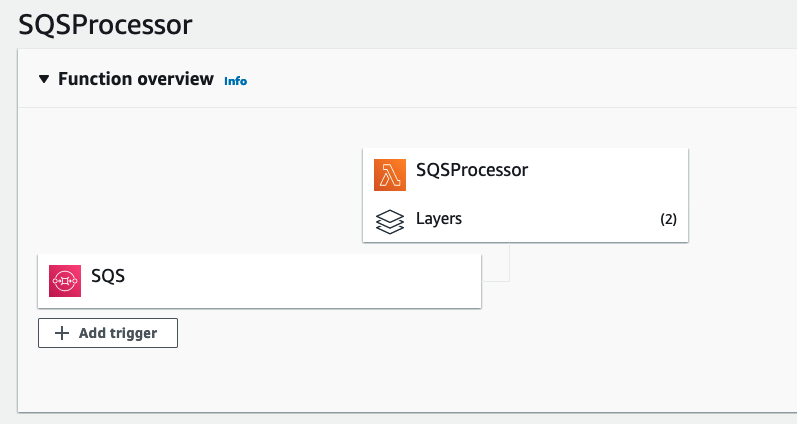
6. Now that we’ve added code to our lambda, need to configure some environment variables in order for our Lambda function to be able to identify the DynamoDB tables. The environment variables we are going to configure are:

Image

7. Configure as shown below:

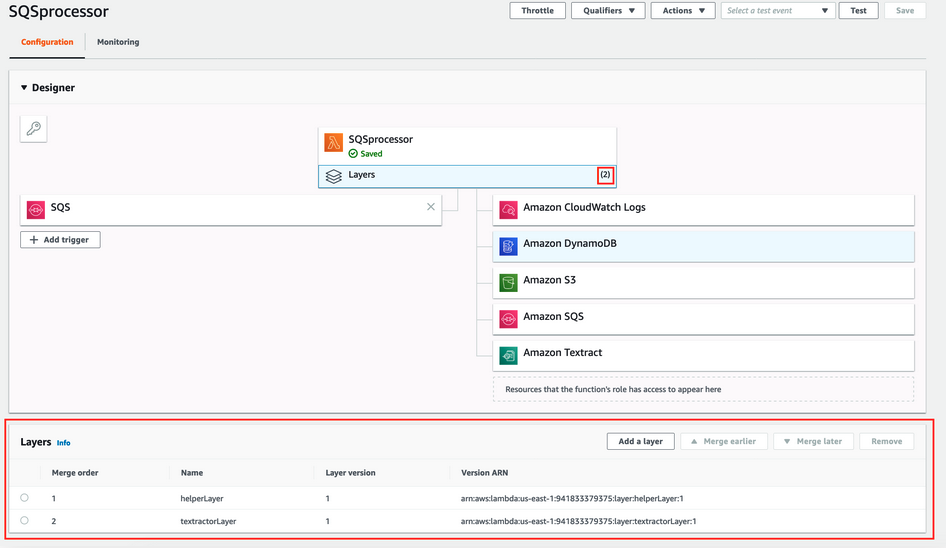


8. Increase the time out of the lambda function (30 seconds) under basic settings. This ensures the lambda has enough time to finish processing before it times out.



## Create the SQS Event Trigger

1. In the lambda service, select add trigger.
2. Configure your trigger to be the SQS you created previously.
3. Alter the Batch Size to 1.
4. Select add.



## Add the Lambda Layer

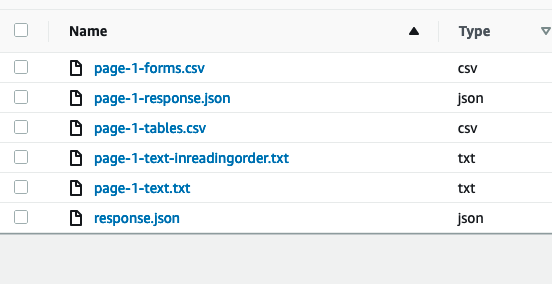
1. Select add a layer.
2. Configure 2 layers to this function. **HelperLayer** and **textractorLayer** both at v1.
3. Select add.

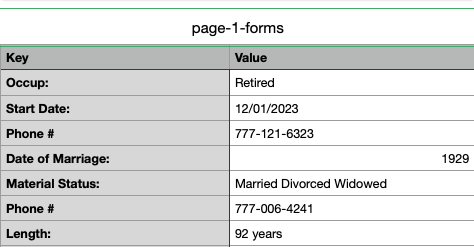
Test the Synchronous pipeline

Here we will upload an image with text into the S3 Bucket and examine the results.

1. Download any PDF, JPEG, or PNG file. Click upload into S3.
2. A few seconds later you should see a folder in the S3 bucket named after the file. Here’s an example of the file and folder structure created.Image

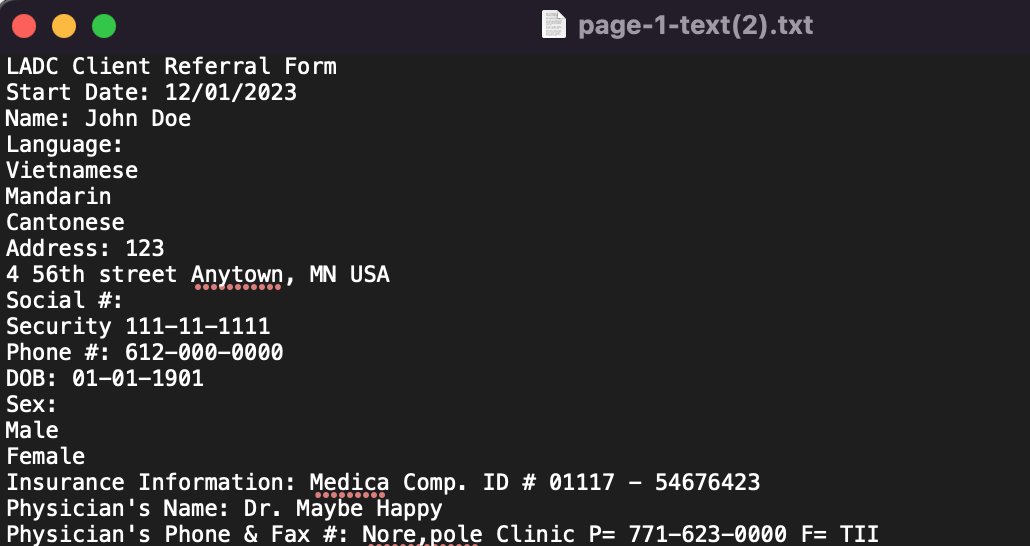
3. Here you will see the following file structures.



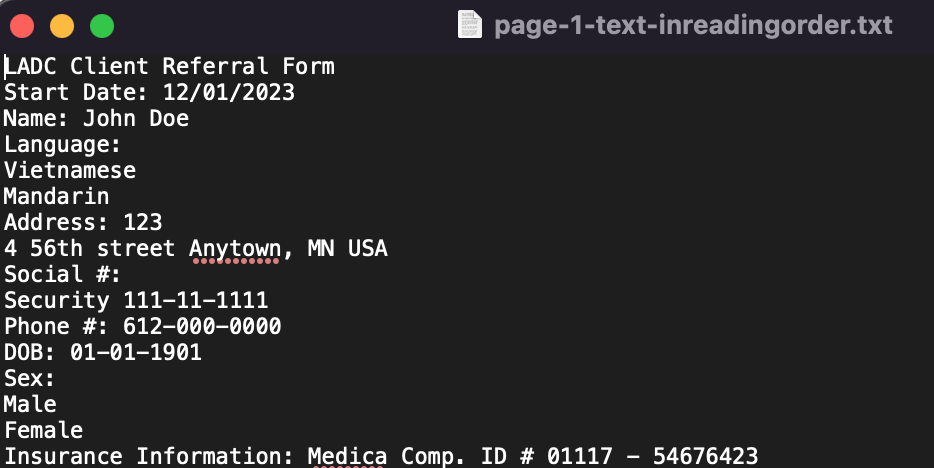
-forms: Creates a key-value pair of each of the mappings textract identified.

-response: Returns the response in a JSON format.

-tables: Returns any key value pairs identified in a table.

-text: Returns the response in a text format.

-text-inreading-order: Returns the text in the order that textract read the values.



Takeaways

Using AWS Textract is much more effective in reading hand-written documents than Rekognition. I tested Textract with over 25 different forms. Each with unique hand styles. Out of those Textract was able to pick up differences. Rekognition on the other hand struggled. It had difficulty reading simply JPEG, PDF, and PNG files. Where in some cases Rekognition was able to read some PNG files but not all. Further analysis can be done to see the effectiveness of Textract with reading text from billboards, videos, or graphic images. As suggested by AWS, Rekognition would be far superior but further tests can be done to verify.

**References:**

AWS S3: <https://aws.amazon.com/s3/>

AWS SQS: <https://aws.amazon.com/sqs/>

AWS Dynamo DB: <https://aws.amazon.com/dynamodb/>

AWS IAM: <https://aws.amazon.com/iam/faqs/>

AWS Lambda <https://aws.amazon.com/lambda/>

AWS Lambda Layer: <https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html>.