**Building Serverless Workflow to Process Files Uploaded to Amazon S3**

Created by[in28Minutes Official](https://deloittedevelopment.udemy.com/user/in28minutes/)

**AWS Workspace**

**90-180 minutes**

Your system will take a new JSON file that is uploaded to an S3 bucket and store details from the JSON file (as an item) with a unique id to a DynamoDB table. You want to build a serverless auto scaling and highly resilient workflow involving Amazon S3 Notifications, Amazon SQS and AWS Lambda to store data to a DynamoDB table.

We would need to build a new workflow. We are receiving millions of JSON files (of different formats), first uploaded to an Amazon S3 Bucket. As soon as we receive a file, we need to store the content of the file as an item to a DynamoDB table. It is important to ensure that every file is successfully processed. We would need to have a mechanism that would automatically retry failed files. In case processing of a file did not succeed after multiple retries, you would need to send them to a failed message queue. We are leaning towards a serverless approach using Amazon SQS and AWS Lambda.

In this lab, you will be an engineer at a company using AWS as their cloud platform. You need to process millions of files being uploaded to Amazon S3 and store the data to DynamoDB. Your architecture team wants to try out a message based, event-driven serverless approach using Amazon SQS and Amazon Lambda. Your boss will want you to build a proof-of-concept.

**How you'll work**

Your project has been broken into a set of tasks. To complete these tasks, use the provided workspace. You can launch your workspace by clicking below or using the button in the top right of the screen.

Task

1-Create an S3 Bucket for storing file

2-Create a DynamoDB table to store data

3-Create a Role for Lambda Function with the right permission

4-Create the Lambda Function using the role LambdaRoleForProcessingItems

5-Create a Dead letter SQS Queue to store failed messages

6-Create an SQS Queue with a Lambda Trigger

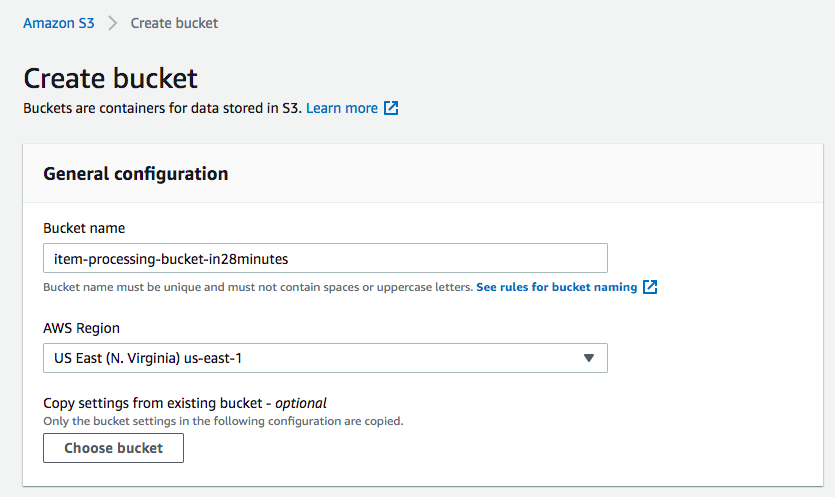
7-Update S3 to trigger notification to SQS

8-Test the serverless workflow with a sample json

Create an S3 Bucket for storing file

Let's create an S3 bucket where our files would be uploaded. We will name the bucket as item-processing-bucket-UNIQUE-ID. Replace `UNIQUE-ID` with something to make your bucket name unique. Note down the bucket name for use in subsequent tasks.

1. In the **AWS Console**, from **Services** menu - choose **S3**
2. Click **Create bucket** and then set the bucket name to item-processing-bucket-UNIQUE-ID. Replace `UNIQUE-ID` with something to make your bucket name unique. (In the screenshot, we use `in28minutes` instead of `UNIQUE-ID`) Your screen should look like :



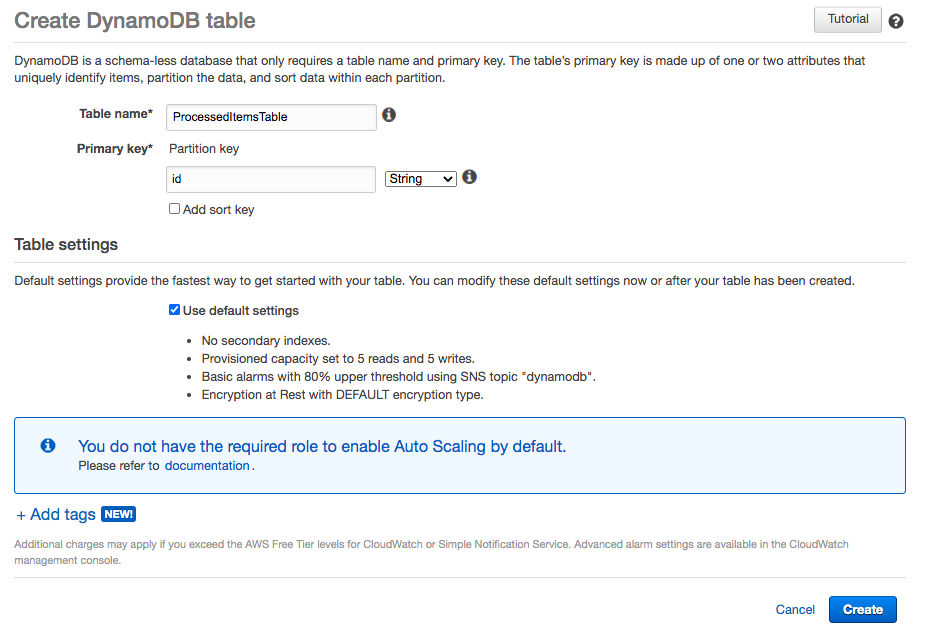
1. Take the defaults for all other settings and click **Create bucket**

**Ref sample.json**

Create a DynamoDB table to store data

We would want to eventually create items in DynamoDB when objects are uploaded to the S3 buckets. Let's create the DynamoDB table with the name `ProcessedItemsTable`.

1. In the **AWS Console**, from **Services** menu - choose **DynamoDB**
2. Click **Create table** and then set the Table name to `ProcessedItemsTable`. Enter `id` as **Primary key** with type as `String`. Your screen should look like:



1. Take the defaults for all other settings and click **Create**
2. Your DynamoDB table is now ready!

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html>

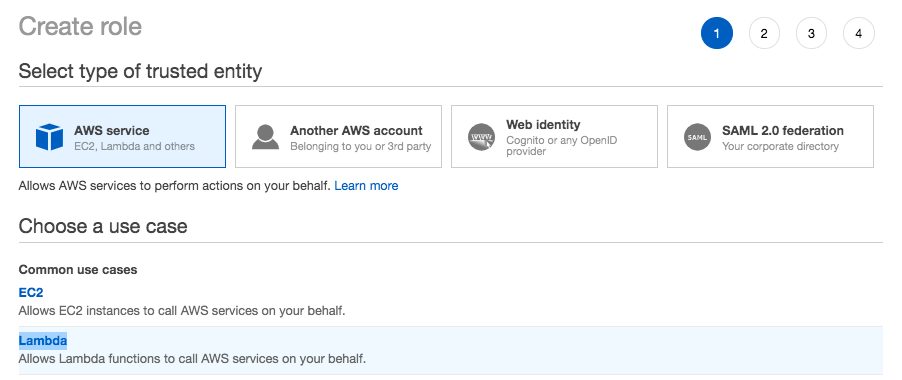
Create a Role for Lambda Function with the right permission

Lambda function needs to have access to:

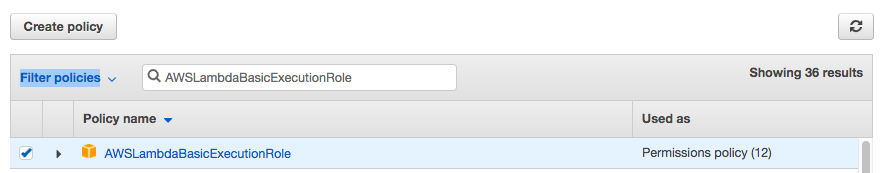
* Read from S3 Bucket (`AmazonS3ReadOnlyAccess`)
* Receive message from SQS Queue (`AWSLambdaSQSQueueExecutionRole`)
* Write to DynamoDB table (`AmazonDynamoDBFullAccess`)
* Write logs to CloudWatch (`AWSLambdaBasicExecutionRole`)

Let's create a role with the name `LambdaRoleForProcessingItems`.

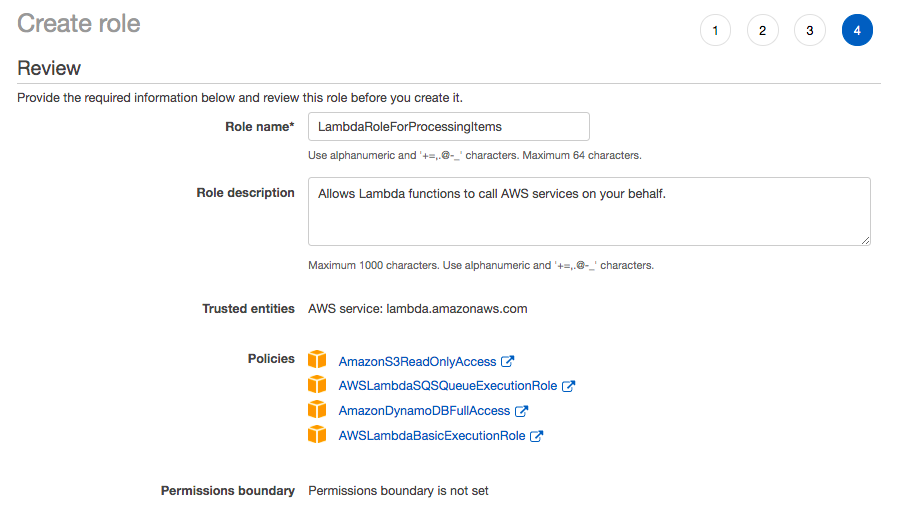
1. In the **AWS Console**, from **Services** menu - choose **IAM**
2. Choose **Roles** from the left hand side menu
3. Click **Create role**. Choose type of trusted entity as **AWS service**. Choose a use case **Lambda**. Your screen should look like:



1. Click **Next: Permissions**
2. Choose 4 Roles one by one: `AmazonS3ReadOnlyAccess`, `AWSLambdaSQSQueueExecutionRole`, `AmazonDynamoDBFullAccess`, `AWSLambdaBasicExecutionRole`. (Enter Role name beside **Filter policies** and check the checkbox beside the role for all four roles) Example for `AmazonS3ReadOnlyAccess` is shown in the screenshot below:



1. Click **Next: Tags**
2. Click **Next: Review**
3. Enter **Role name** as `LambdaRoleForProcessingItems`. Your screen should look like:



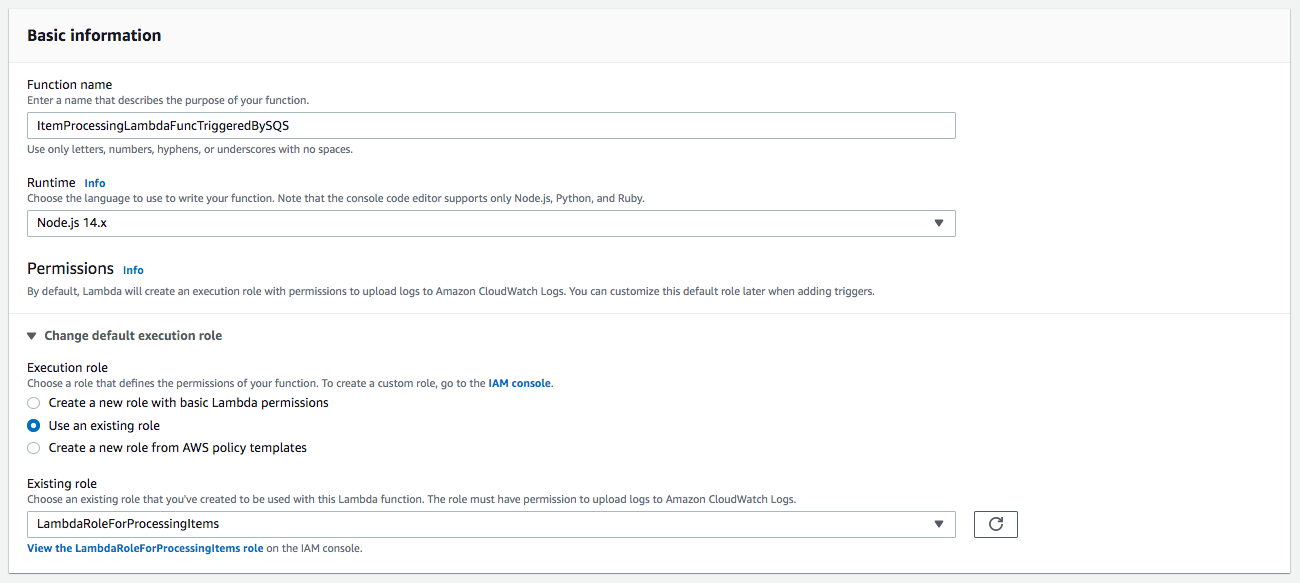
1. Click **Create role**
2. You should see a message saying **The role LambdaRoleForProcessingItems has been created**

<https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.html>

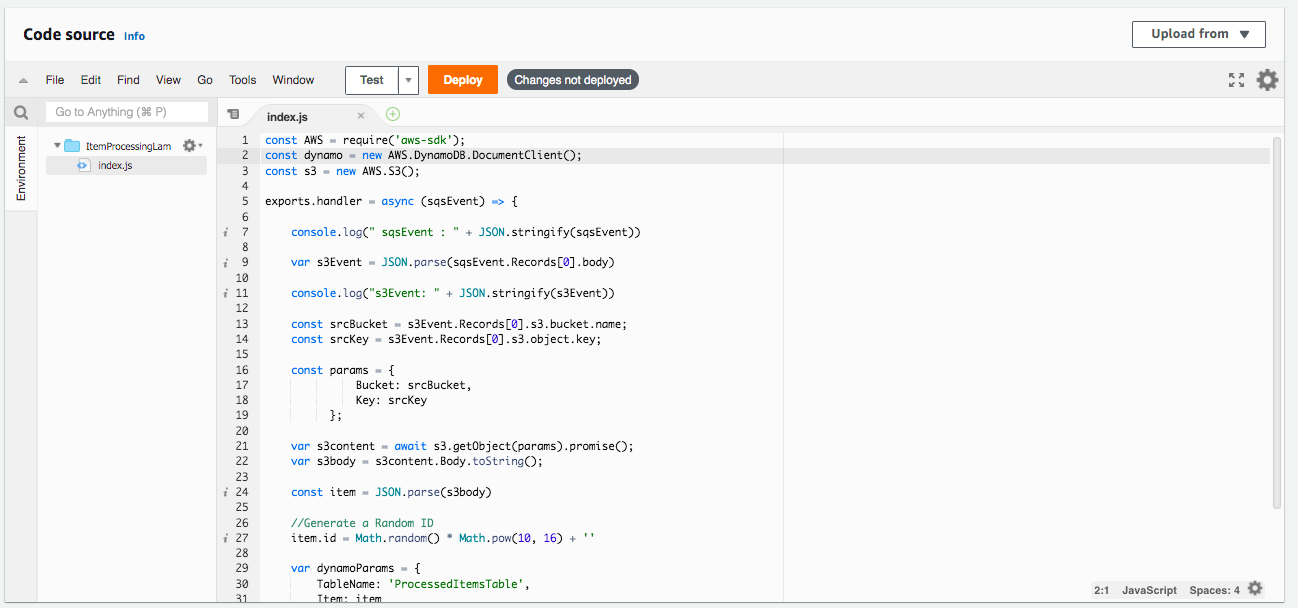
Create the Lambda Function using the role LambdaRoleForProcessingItems

Let's create a Lambda Function and assign it a role of LambdaRoleForProcessingItems. The Lambda Function will listen for messages on the SQS queue (we will configure the trigger in the next task). When a new message arrives, we will ready the object from Amazon S3, process it and insert an item into the DynamoDB table.

1. In the **AWS Console**, from **Services** menu - choose **Lambda**
2. Choose to **Author from scratch**. Enter **Function name** as `ItemProcessingLambdaFuncTriggeredBySQS`. Choose **Runtime** as Node.js 14.x. In Permissions expand **Change default execution role**. Enter **Existing role** as LambdaRoleForProcessingItems. Your screen should look like:



1. Click **Create function**
2. On the Lambda Source Code screen, Under **Code**> **Code Source** > Select the file **index.js** under the folder **ItemProcessingLambdaFuncTriggeredBySQS**in the **Assets** below. Copy the code from the file `lambda-code.js`. Your screen should look like:



1. Click **Deploy**
2. Your Lambda Function is ready

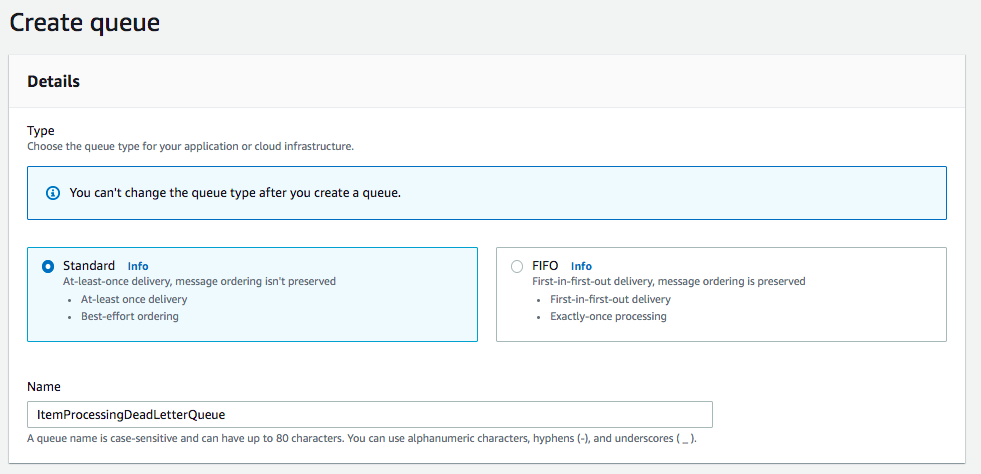
## Ref lambda-code.js, zz-dynamodb-item.json,zz-S3-event-structure.json,zz-sqs-event-structure.json

<https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html>

Create a Dead letter SQS Queue to store failed messages

We want a highly resilient solution. We will create a dead letter queue where messages which failed processing even with repeated attempts will be stored.

1. In the **AWS Console**, from **Services** menu - choose **Simple Queuing Service**
2. Enter **Name** as `ItemProcessingDeadLetterQueue`. In **Configuration**, Enter **Message retention period** as `14 days`. Take the defaults for all other settings. Your screen should look like:



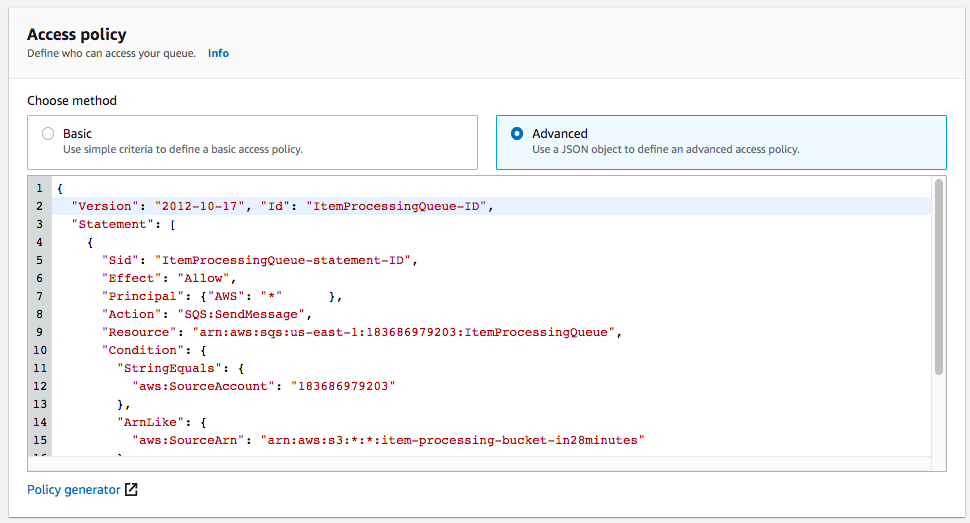
1. Click **Create queue**

<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-dead-letter-queues.html>

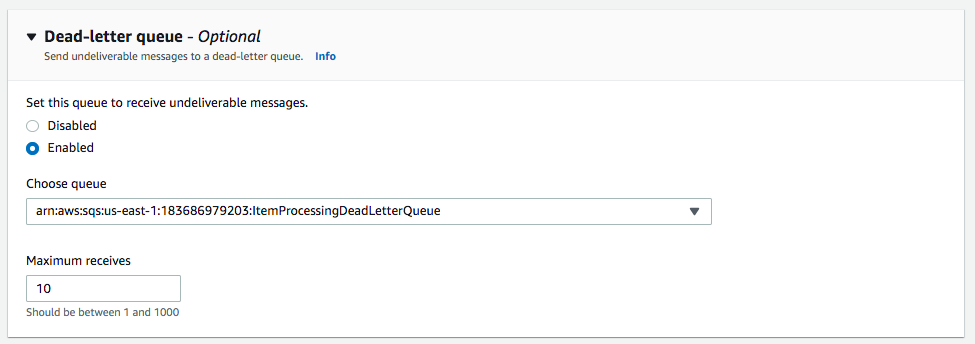
Create an SQS Queue with a Lambda Trigger

We want to put a message on the SQS queue as soon as an object is uploaded to S3 bucket. Let's create an SQS queue. We will also set up a Lambda trigger to enable processing of the message by the Lambda function. We will also configure an Access Policy to allow S3 bucket notification to place a message on the SQS Queue. Make sure that you replace your AWS Account Id, Bucket Id and SQS queue name with the right names in the policy.

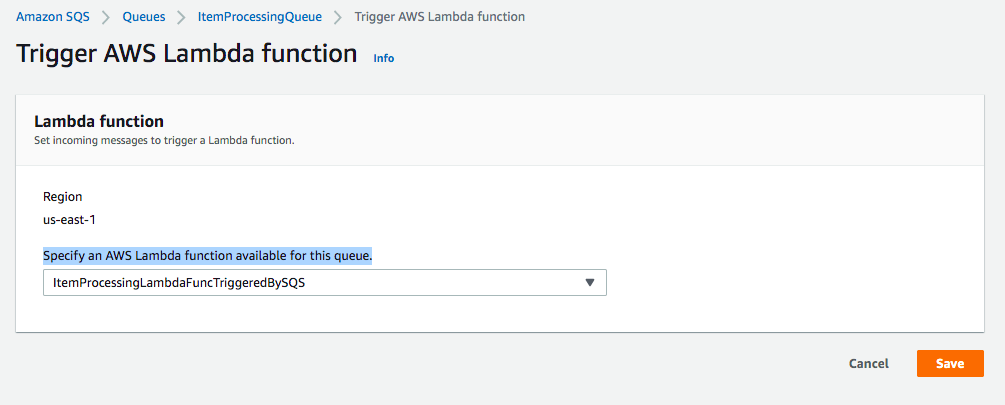
1. In the **AWS Console**, from **Services** menu - choose **Simple Queuing Service**
2. Enter **Name** as `ItemProcessingQueue`
3. Copy to **Access Policy** content from the file `access-policy.json` in the **Assets** below. Make sure you replace the variable ACCOUNT-ID with your AWS account number twice (You can find your AWS account number by going to Support > Support Center > Account number in left hand side menu).. Also replace the S3 bucket name with the name of S3 bucket created earlier(replace `UNIQUE-ID`). Make sure that SQS queue name matches the queue you created earlier. Your screen should look like:



1. Under **Dead-letter queue** > Set **Set this queue to receive undeliverable messages** to Enabled. And set **Choose queue** as `ItemProcessingDeadLetterQueue`. Your screen should look like:



1. Take the defaults for all other settings. Click **Create queue**.
2. On the `ItemProcessingQueue` Details screen, goto **Lambda triggers** tab. Click **Configure Lambda function trigger**. Set **Specify an AWS Lambda function available for this queue** as `ItemProcessingLambdaFuncTriggeredBySQS`. Your screen should look like:



1. Click Save

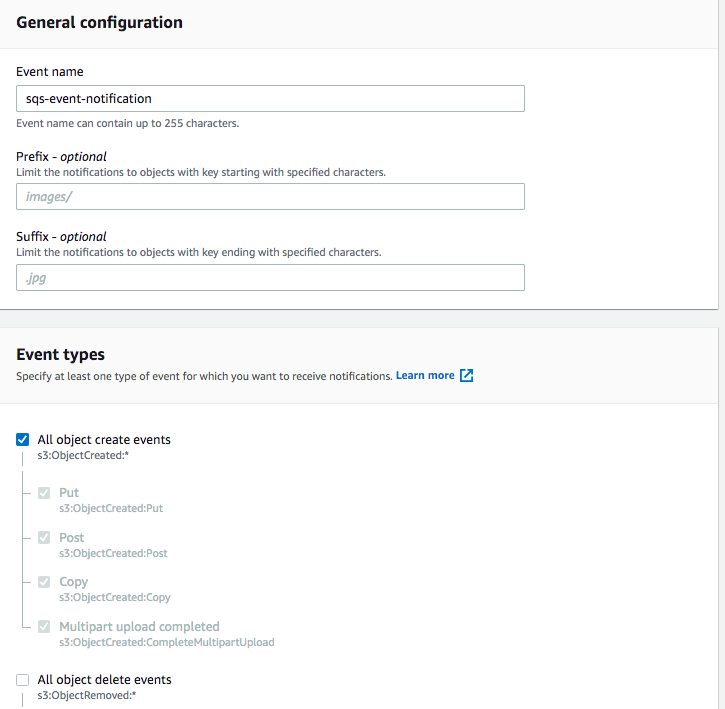
Ref access-policy.json

<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-examples-of-sqs-policies.html>

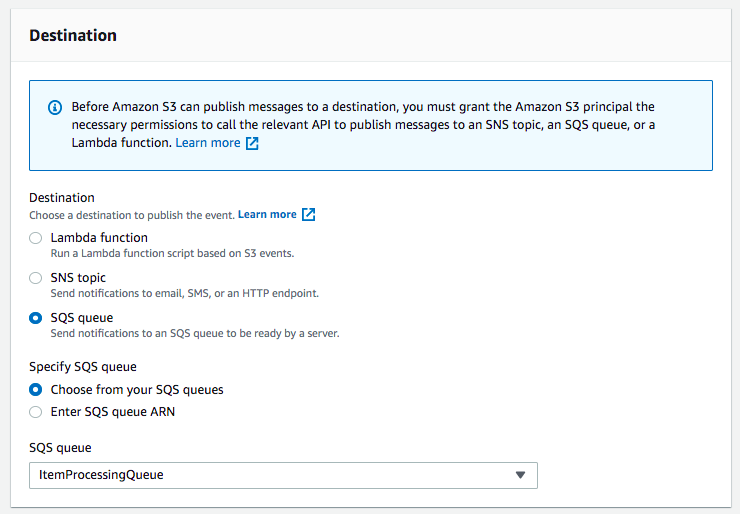
Update S3 to trigger notification to SQS

Now that the SQS queue and the Lambda function to process messages on the queue are ready, let's update our S3 bucket to trigger notifications when objects are uploaded. Let's trigger notifications to an SQS queue.

1. In the **AWS Console**, from **Services** menu - choose **S3**
2. Search for **item-processing-bucket** and choose the S3 bucket (`item-processing-bucket-UNIQUE-ID`) that we created earlier
3. Go to **Properties** tab. Scroll down to **Event notifications**. Click **Create event notification**.
4. On the **Create event notification** screen, Enter **Event name** as sqs-event-notification. Under **Event types**, Enable Checkbox for **All object create events**. Your screen should look like:



1. Scroll down to **Destination** section. Choose **Destination** as **SQS queue**. Under **Specify SQS queue**, Select **Choose from your SQS queues** and choose **SQS queue** as `ItemProcessingQueue`. Your screen should look like:



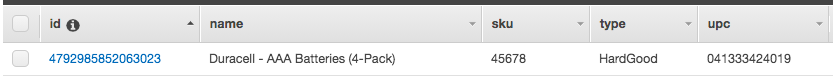
1. Go ahead and click **Save changes**

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/NotificationHowTo.html>

Test the serverless workflow with a sample json

Let's test the workflow by uploading an object (sample.json) to S3 bucket. We can check if an item is created in DynamoDB.

1. In the **AWS Console**, from **Services** menu - choose **S3**
2. Search for **item-processing-bucket** and choose the S3 bucket with a unique id (`item-processing-bucket-UNIQUE-ID`) that we created earlier. Go to **Objects** tab. Click **Upload**. On the next screen, click **Add files** and select the `sample.json` file (in the **Assets** below) to upload. Click **upload**.
3. In the **AWS Console**, from **Services** menu - choose **DynamoDB**. In the left hand side menu, choose **Tables**. Select table `ProcessedItemsTable`. Go to **Items** tab. You should see an item on the screen. Your screen should look like:



1. You can create more test files and upload

Ref sample.json,zz-dynamodb-item.json,zz-s3-event-structure.json,zz-sqs-event-structure.json