CLOUD ENGINEER TEST DOCUMENTATION

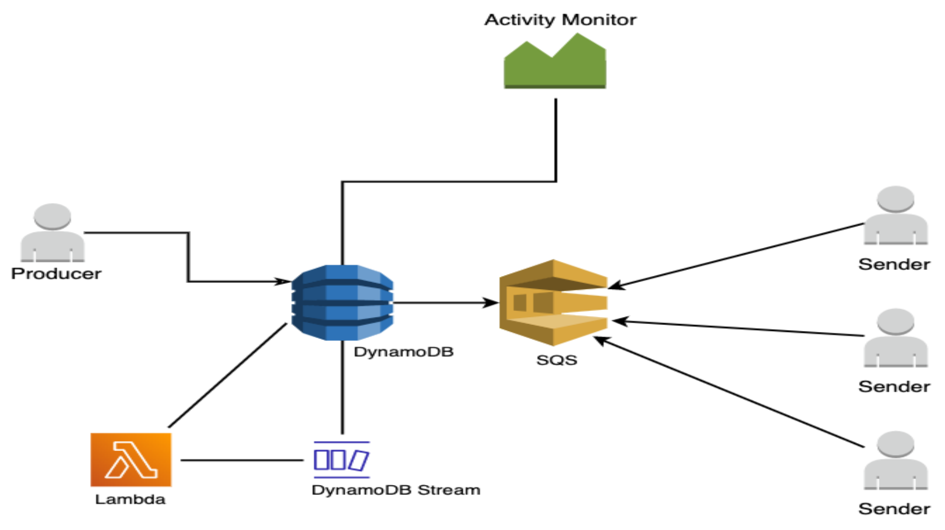
SUBMITTED BY: CAROL NAVYA PAGOLU

1. PROBLEM STATEMENT

The objective is to simulate sending a large number of SMS alerts, like for an emergency alert service. The simulation consists of three parts:

1. A ***Producer*** that generates a configurable number of messages (default 1000) to a random phone number. Each message contains up to 100 random characters.
2. A ***Sender*** who pickups up messages from the producer and simulates sending messages by waiting a random period of time distributed around a configurable mean. The sender also has a configurable failure rate.
3. A ***Progress Monitor*** that displays the following and updates it every N seconds (configurable):
    a. Number of messages sent so far
    b. Number of messages failed so far
    c. Average time per message so far

2. SYSTEM OVERVIEW



The producer continuously generates data and adds it to the dynamodb (for backup) and the sqs queue. The messages will wait in the queue for the senders. The sender instances will pick up the messages in the queue and simulate sending SMS until they hit their failure rate. If there are no messages in the queue, then all the sender instances die.

2.1 Producer

A producer is an actor that generates Phone numbers and SMS pairs and adds the pair to the SQS queue and DynamoDB. Only 1 instance of Producer is generated.

2.2 AWS DynamoDB

DynamoDB is a serverless, fully managed, key-value NoSQL database that can support high-performance software of any size. It is a database service that is quick, adaptable, affordable, highly scalable, fault-tolerant, and secure. DynamoDB stores the messages produced by the producer and can be used to gather progress monitor metrics. DynamoDB can automatically scale by monitoring how close your usage is to the limits. This feature makes it easier for your system to adapt to changes in data traffic, which enhances application performance and lowers costs. Fine-grained access control is a feature of DynamoDB that gives the table owner more control over the data in the table.

2.3 AWS SQS

Amazon Simple Queue Service (SQS) enables users to send, store, and receive messages between software components at any volume. One can use it to handle messages at a large scale while keeping the message in order, which enables message deduplication. The producer adds the messages generated into the SQS queue. The messages wait in the queue until a Sender fetches them and simulates sending of SMS.

2.4 Sender

A sender is an actor that fetches messages generated by the producer from the SQS queue and simulates sending SMS. This actor has a configurable failure rate and the amount of time it waits before sending the next message. This sender dies once it hits the failure rate.

2.4 AWS Lambda

AWS Lambda is a serverless, event-driven computing service, that is triggered whenever the database is modified. This is used to calculate the progress monitor metrics on the fly by collecting processing data from the dynamoDB data streams. This can be scaled automatically to meet any data volume.

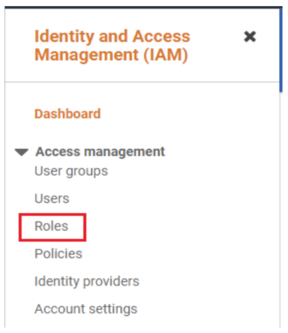
2.5 Activity Monitor

This actor outputs the number of messages that failed, the number of messages sent so far, and the average time to send each message for every 'N' seconds that is configured by the user.

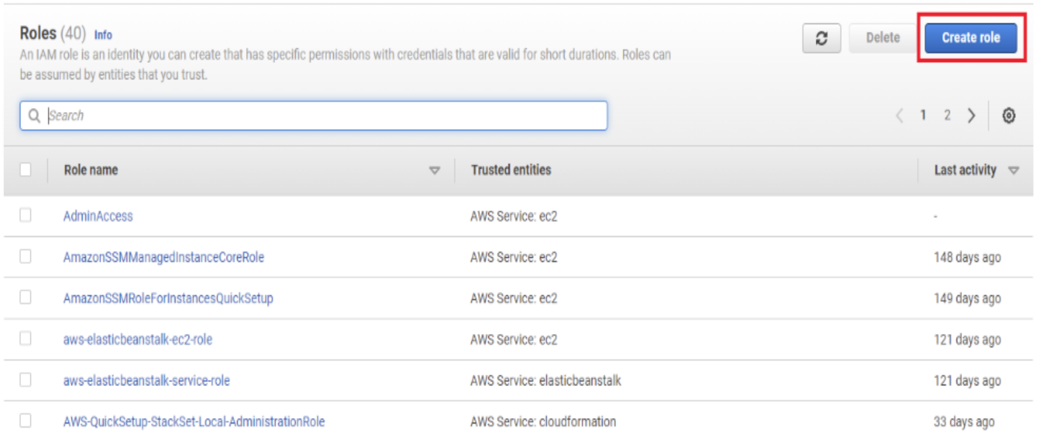
3. SETUP INSTRUCTIONS

STEP 1: Set up a virtual environment and install libraries in requirement.txt
STEP 2: Configure boto3 with your AWS credentials
STEP 3: Clone the GitHub repo
STEP 4: Run setup.py to initialize tables `python3 setup.py`
STEP 5: Create an IAM role for full dynamo DB access.

5.1 Go to IAM in AWS Console and click on ***'Roles'***.



5.2 Click on ***'Create role'***.

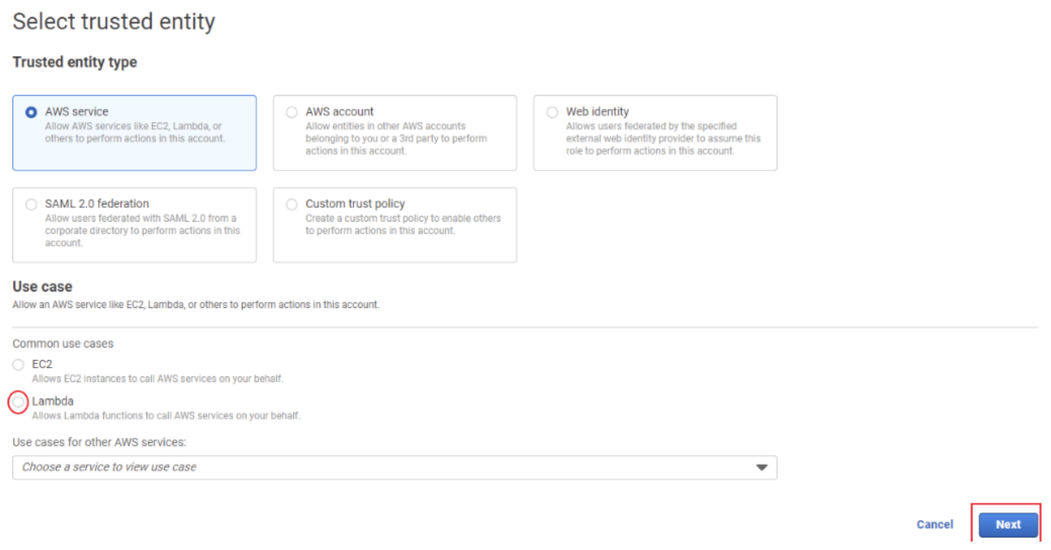


Roles (40) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AdminAccess	AWS Service: ec2	-
AmazonSSMManagedInstanceCoreRole	AWS Service: ec2	148 days ago
AmazonSSMRoleForInstancesQuickSetup	AWS Service: ec2	149 days ago
aws-elasticbeanstalk-ec2-role	AWS Service: ec2	121 days ago
aws-elasticbeanstalk-service-role	AWS Service: elasticbeanstalk	121 days ago
AWS-QuickSetup-StackSet-Local-AdministrationRole	AWS Service: cloudformation	33 days ago

5.3 Select ***AWS service*** as your trusted entity and ***lambda*** for the common use cases, click ***'Next'***



Select trusted entity

Trusted entity type

[✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account — Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity — Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation — Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy — Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

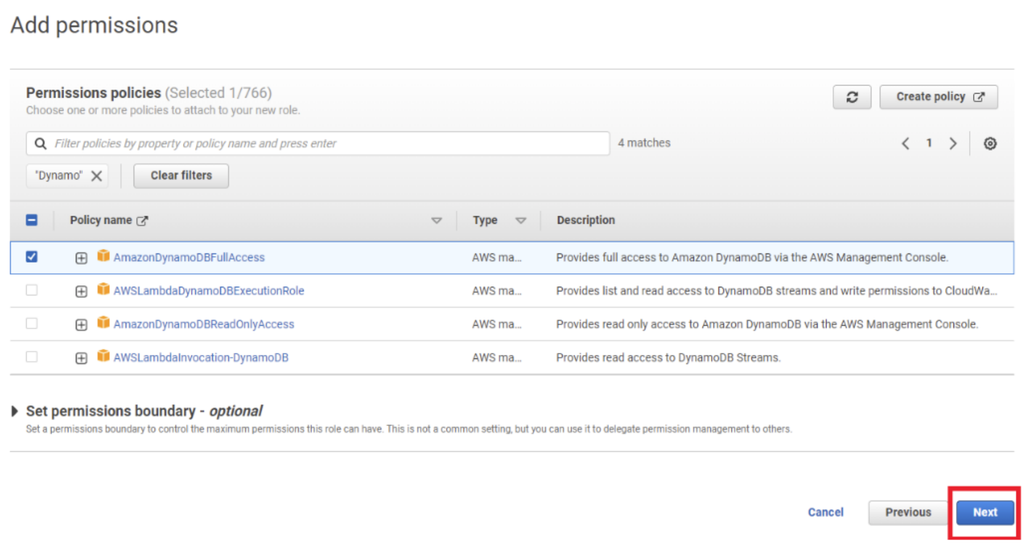
Common use cases

EC2 — Allows EC2 instances to call AWS services on your behalf.

[✓ Lambda] Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

5.4 Enable permission policy by choosing ***'AmazonDynamoDBFullAccess'***



Add permissions

Permissions policies (Selected 1/766)

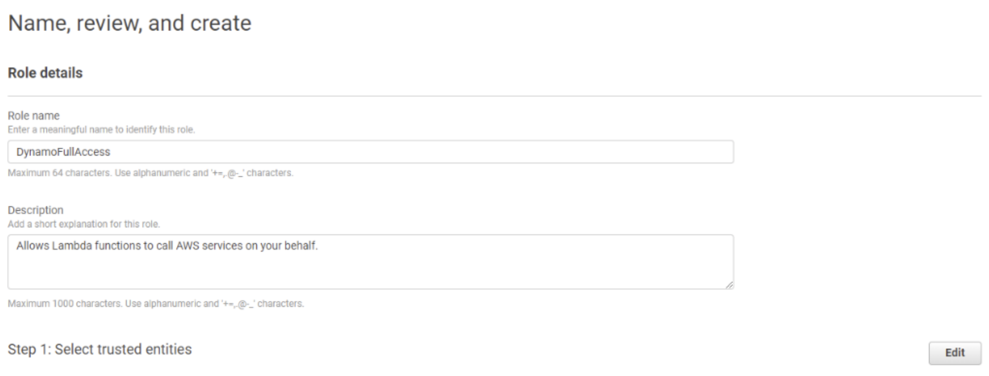
Choose one or more policies to attach to your new role.

	Policy name	Type	Description
✓	AmazonDynamoDBFullAccess	AWS ma...	Provides full access to Amazon DynamoDB via the AWS Management Console.
	AWSLambdaDynamoDBExecutionRole	AWS ma...	Provides list and read access to DynamoDB streams and write permissions to CloudWa...
	AmazonDynamoDBReadOnlyAccess	AWS ma...	Provides read only access to Amazon DynamoDB via the AWS Management Console.
	AWSLambdaInvocation-DynamoDB	AWS ma...	Provides read access to DynamoDB Streams.

Set permissions boundary - *optional*

Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.

5.5 Specify a ***Role name*** and ***Description***. Then select ***'Create role'.***



Name, review, and create

Role details

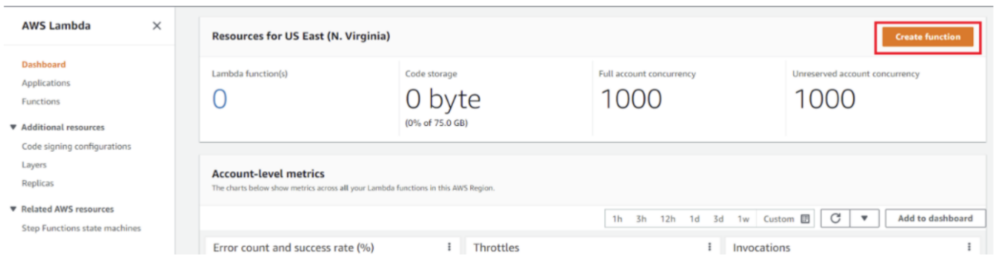
Role name
Enter a meaningful name to identify this role.
DynamoFullAccess
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Step 1: Select trusted entities

STEP 6: Create a Lambda function that will read and write into the database.

6.1 Navigate to AWS Lambda on the console and click on '***Create function***'



Resources for US East (N. Virginia)

Lambda function(s)
0

Code storage
0 byte
(0% of 75.0 GB)

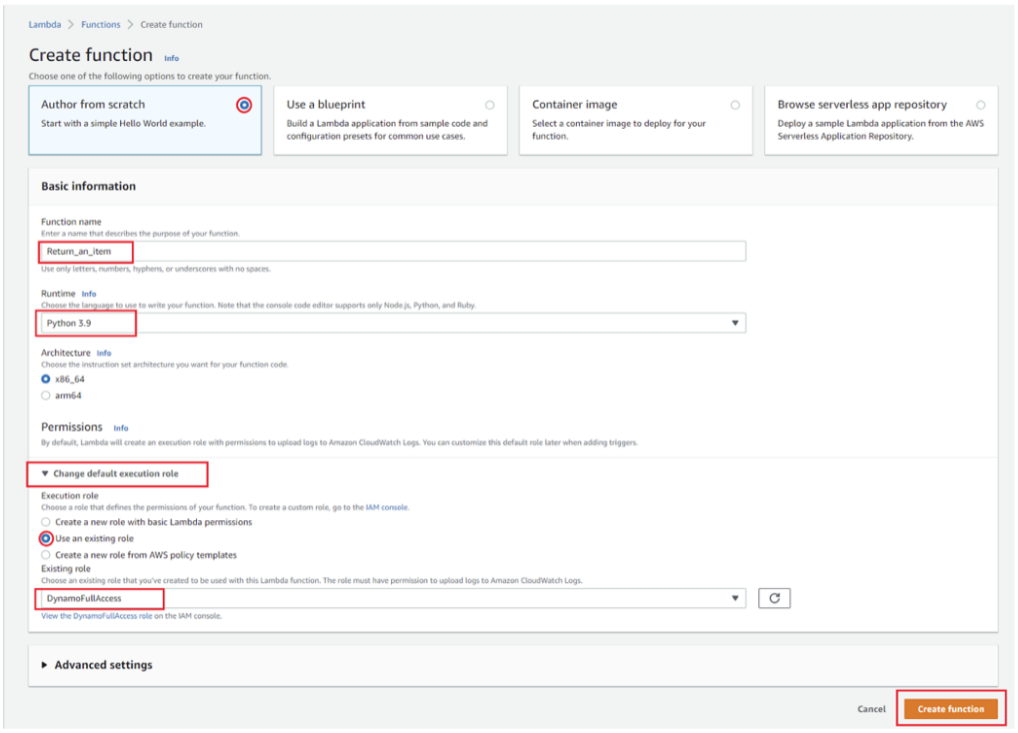
Full account concurrency
1000

Unreserved account concurrency
1000

Account-level metrics
The charts below show metrics across all your Lambda functions in this AWS Region.

Error count and success rate (%)
Throttles
Invocations

6.2 Choose **Author from scratch** and fill out the basic information specified by the red boxes below. Then, click on '***Create function***'



Create function Info
Choose one of the following options to create your function.

[✓ Author from scratch] — Start with a simple Hello World example.
Use a blueprint — Build a Lambda application from sample code and configuration presets for common use cases.
Container image — Select a container image to deploy for your function.
Browse serverless app repository — Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
Return_an_item
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64]
arm64

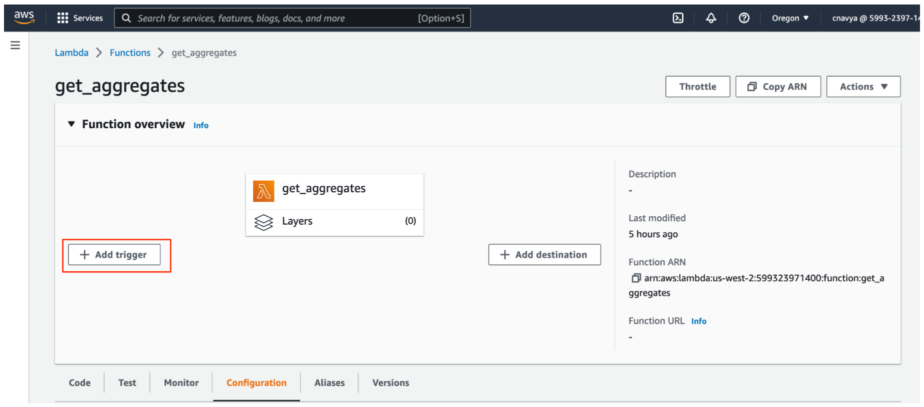
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
[✓ Use an existing role]
Create a new role from AWS policy templates

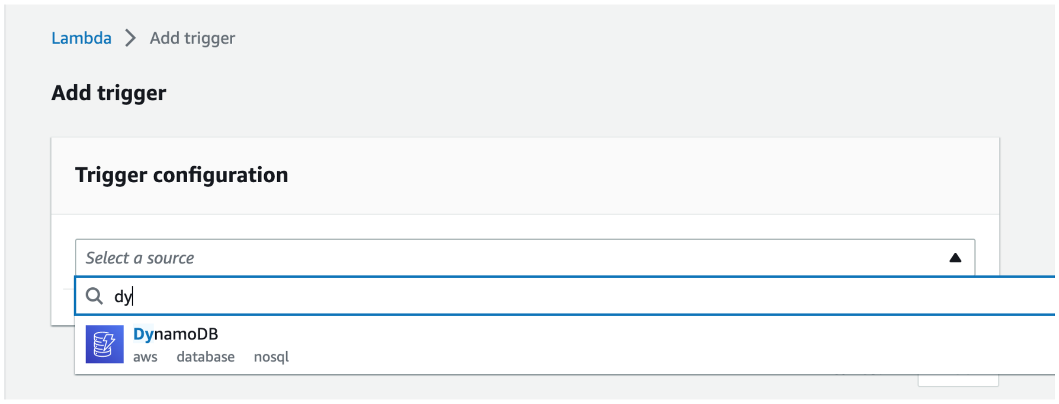
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
DynamoFullAccess
View the DynamoFullAccess role on the IAM console.

6.3 Now, add a trigger to invoke lambda on the message table stream

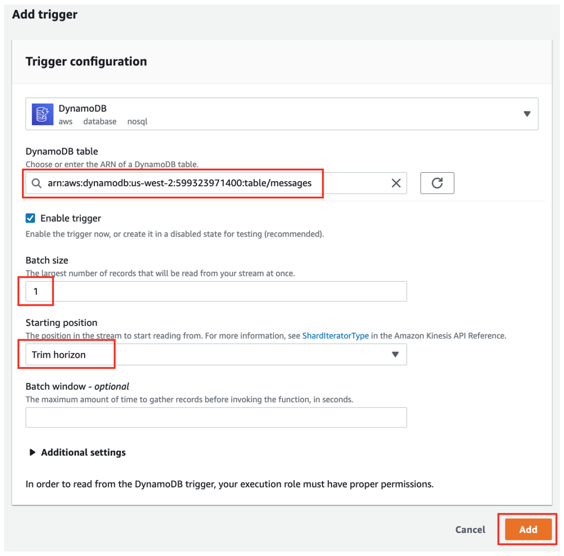


Description
-
Last modified
5 hours ago
Function ARN
arn:aws:lambda:us-west-2:599323971400:function:get_aggregates
Function URL
-

6.4 Select **DynamoDB** as the source.



6.5 Select the **ARN of the *message*** table and specify **batch size=1** and start the position as **Trim horizon** and click on **'*Add*'**



DynamoDB table
Choose or enter the ARN of a DynamoDB table.
arn:aws:dynamodb:us-west-2:599323971400:table/messages

Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

Batch size
The largest number of records that will be read from your stream at once.
1

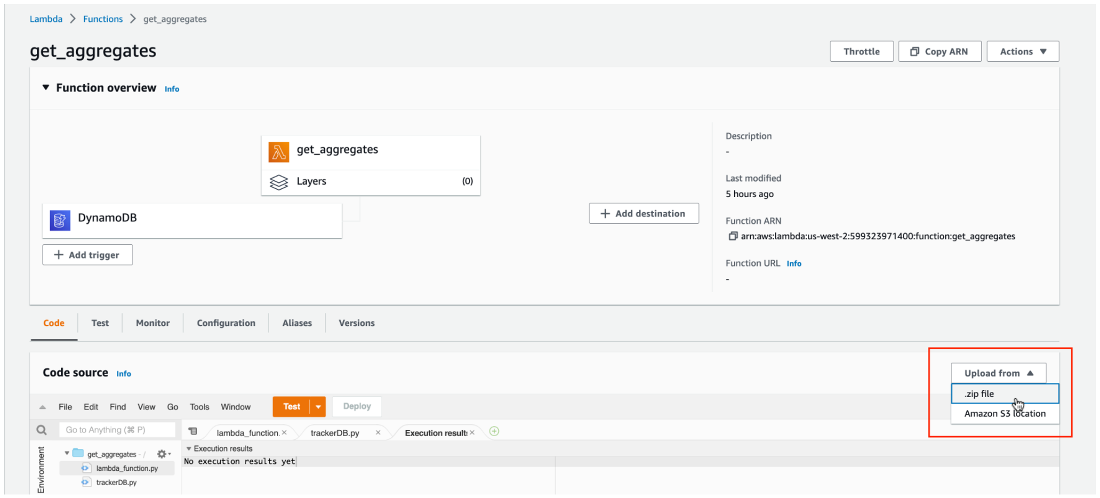
Starting position
The position in the stream to start reading from. For more information, see ShardIteratorType in the Amazon Kinesis API Reference.
Trim horizon

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

Additional settings

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

6.6 Now, upload the code from github’s lambda folder



```
Execution results
No execution results yet
```

4. HOW TO RUN

1. Now, run prouder.py to generate <phno, sms> pairs
   `python3 producer.py -m=<max-messages-generated> -l=<length-of-each-message>`
2. To run sender.py to instantiate the sender and simulate sending an SMS
   `python3 sender.py -w=<wait-time> -f=<failure-rate>`
3. For progress monitoring, run activityMonitor.py.
   `python3 activityMonitor.py -t=<time(secs)-to-casually-overlook-stats>`