

AWS Solution Architect Course Project

Infrastructure Deployment for Real-time Data Management Requirements
on the AWS Cloud

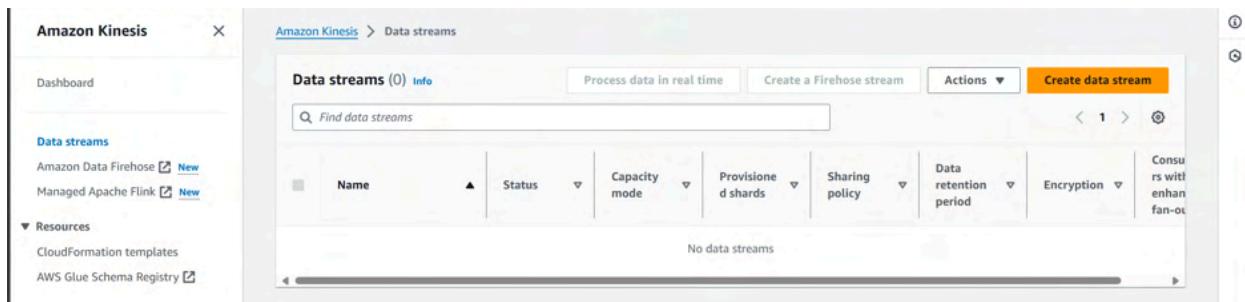
Objective

Create data in a Kinesis stream that can be copied to the DynamoDB database.

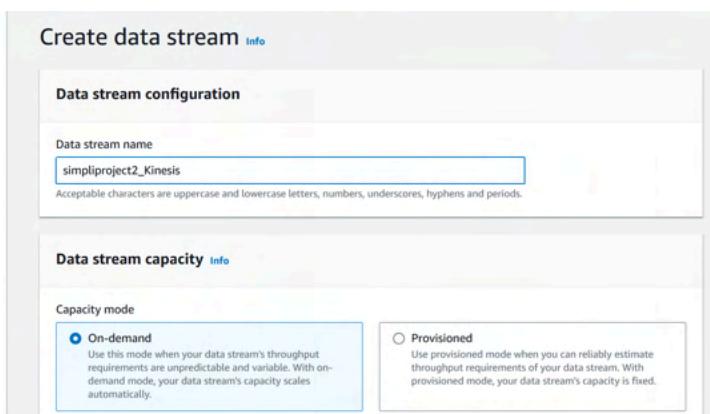
Tasks:

1. Create an AWS Kinesis Data Streams
2. Create an AWS Lambda Function
3. Create an AWS DynamoDB Database
4. Perform Scan Operation

Step 1: Create a Kinesis Data Stream



Go to Kinesis. Go to Data Streams and click on “Create Data Stream”.



In Create data stream, create a name for the data stream. In this case “simpliproject2_Kinesis”.

Data stream capacity [Info](#)

Capacity mode

On-demand
Use this mode when your data stream's throughput requirements are unpredictable and variable. With on-demand mode, your data stream's capacity scales automatically.

Provisioned
Use provisioned mode when you can reliably estimate throughput requirements of your data stream. With provisioned mode, your data stream's capacity is fixed.

Total data stream capacity
By default, data streams with on-demand mode scale throughput automatically to accommodate traffic of up to 200 MiB per second and 200,000 records per second for the write capacity. If traffic exceeds capacity, your data stream will throttle. To request capacity increase up to 2GB per second write and 4GB per second read, [submit a support ticket](#).

Write capacity	Read capacity
Maximum 200 MiB/second and 200,000 records/second	Maximum (per consumer) 400 MiB/second Up to 2 default consumers. Use Enhanced Fan-Out (EFO) for more consumers. EFO supports adding up to 20 consumers, each having a dedicated throughput.

ⓘ On-demand mode has a pay-per-throughput pricing model. See [Kinesis pricing for on-demand mode](#).

Data stream settings
You can edit the settings after the data stream has been created and is in the active status.

Setting	Value	Editable after creation
Capacity mode	On-demand	<input checked="" type="radio"/> Yes
Data retention period	1 day	<input checked="" type="radio"/> Yes
Server-side encryption	Disabled	<input checked="" type="radio"/> Yes
Monitoring enhanced metrics	Disabled	<input checked="" type="radio"/> Yes
Tags	-	<input checked="" type="radio"/> Yes
Data stream sharing policy	No policy	<input checked="" type="radio"/> Yes

You give two options for Dat stream capacity: On-Demand or Provisioned.

In this case, we will use On-Demand. You can add and edit data streams after creation.

ⓘ On-demand mode has a pay-per-throughput pricing model. See [Kinesis pricing for on-demand mode](#).

Data stream settings
You can edit the settings after the data stream has been created and is in the active status.

Setting	Value	Editable after creation
Capacity mode	On-demand	<input checked="" type="radio"/> Yes
Data retention period	1 day	<input checked="" type="radio"/> Yes
Server-side encryption	Disabled	<input checked="" type="radio"/> Yes
Monitoring enhanced metrics	Disabled	<input checked="" type="radio"/> Yes
Tags	-	<input checked="" type="radio"/> Yes
Data stream sharing policy	No policy	<input checked="" type="radio"/> Yes

[Cancel](#) [Create data stream](#)

Click on Create data stream.

The screenshot shows the 'Amazon Kinesis' service in the AWS Management Console. A success message at the top states: 'Data stream simpliproject2_Kinesis successfully created.' The main area displays the details for the newly created data stream 'simpliproject2_Kinesis'. The 'Data stream summary' table includes the following information:

Status	Capacity mode	ARN	Creation time
Active	On-demand	arn:aws:kinesis:us-east-1:501614081035:stream/simpliproject2_Kinesis	August 03, 2024 at 12:47 PDT
	Data retention period		1 day

Below the summary, there are tabs for 'Applications', 'Monitoring', 'Configuration', 'Enhanced fan-out (0)', 'Data viewer', 'Data analytics - new', and 'Data stream sharing'. The 'Applications' tab is selected. Under the 'Producers' section, links are provided for 'Amazon Kinesis Agent', 'AWS SDK', and 'Amazon Kinesis Producer Library (KPL)'. Each link has a 'View in GitHub' button.

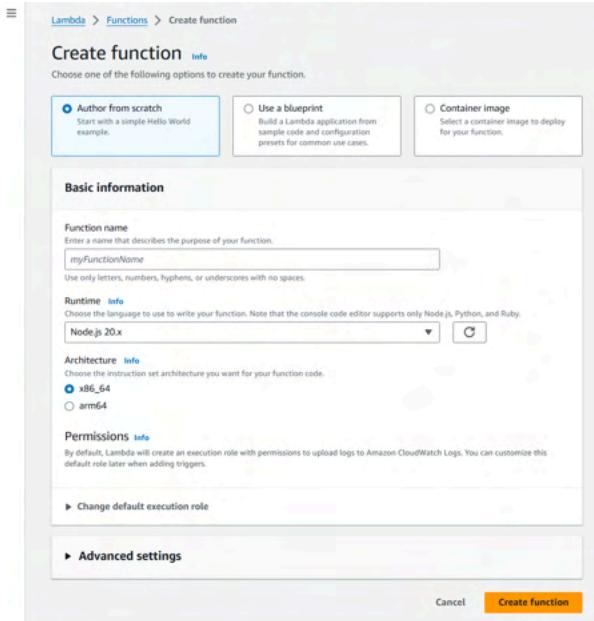
Kinesis data stream created.

Step 2: Create an AWS Lambda Function

2.1: Lambda Function Producer

The screenshot shows the 'Lambda' service in the AWS Management Console. The 'Functions' list page is displayed, showing 0 functions. The table headers are 'Function name', 'Description', 'Package type', 'Runtime', and 'Last modified'. A message at the bottom of the table says 'There is no data to display.'

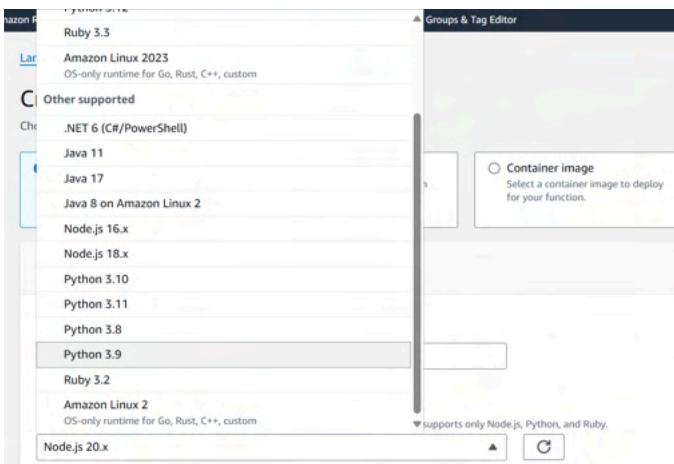
Go to Lambda function. Under Functions and click on 'Create function'.



Under Create Function, enter the name you want for your function.

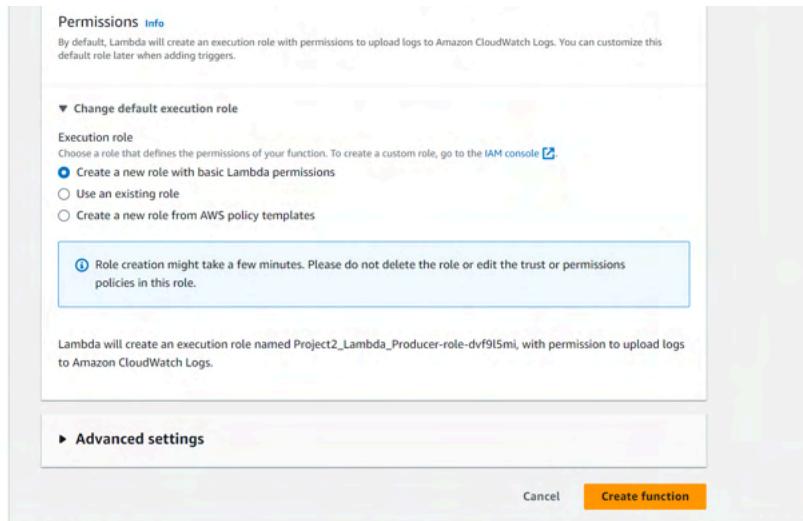
Example: “Project2_Lambda_Producer”.

You will be creating two lambda functions for this project; The first one is the producer for your Kinesis.



For Runtime, select Python 3.9 as your consumer set.

Keep Architecture at x86_64.



Under 'Change default execution role' in 'Execution Role'. You will receive a new role in IAM.

Click on "Create function."

Lambda Function created.

In your function, go to Configuration.

First, go to General Configuration

The screenshot shows the AWS Lambda Configuration page. The top navigation bar includes tabs for Code, Test, Monitor, Configuration (which is selected), Aliases, and Versions. On the left, a sidebar lists General configuration, Triggers, Permissions, Destinations, Function URL, and Environment variables. The main content area displays the General configuration settings:

- Description:** -
- Memory:** 128 MB
- Ephemeral storage:** 512 MB
- SnapStart:** None
- Timeout:** 0 min 3 sec

Below these settings, there are detailed sections for Memory, Ephemeral storage, SnapStart, and Timeout. The Timeout section is where the value is being modified from 3 seconds to 5 minutes. The Save button at the bottom right is highlighted.

Change the timeout setting from 3 seconds to 5 minutes. This way depending on the data you are streaming, the function won't time out prematurely.

The screenshot shows the AWS Lambda Configuration page with the Timeout setting changed. The main content area displays the General configuration settings:

- Timeout:** 5 min 0 sec

Below the Timeout field, the Execution role and Existing role sections remain the same, showing the selection of an existing role named "service-role/Project2_Lambda_Producer-role-dvf9l5mi". The Save button at the bottom right is highlighted.

Go to Permissions

The screenshot shows the AWS Lambda function configuration interface. The left sidebar has tabs for 'Code', 'Test', 'Monitor', and 'Configuration'. The 'Configuration' tab is selected. Under 'Execution role', the role name is listed as 'Project2_Lambda_Producer-role-dvf9l5mi'. A 'Resource summary' section shows permissions for 'Amazon CloudWatch Logs' with 3 actions and 2 resources. The 'By resource' tab is selected.

Click on role under Role Name. You are sent to your role in IAM:
Project2_Lambda_Producer-role-dvf9l5m

The screenshot shows the AWS IAM Roles page. The left sidebar lists 'Identity and Access Management (IAM)' with sections for 'Access management' (User groups, Users, Roles, Policies, Identity providers, Account settings), 'Access reports' (Access Analyzer, External access, Unused access, Analyzer settings), 'Credential report', 'Organization activity', and 'Service control policies'. The 'Roles' section is expanded. The main area shows the role 'Project2_Lambda_Producer-role-dvf9l5mi' with a 'Summary' tab. It includes details like creation date (August 14, 2024), ARN, last activity, and maximum session duration (1 hour). The 'Permissions' tab is selected, showing one attached policy: 'AWSLambdaBasicExecutionRole-c6cc2...'. The 'Add permissions' button is highlighted, with options to 'Attach policies' or 'Create inline policy'.

In permission policies, click on Add permissions > Attach policies.

Attach policy to Project2_Lambda_Producer-role-dvf9l5mi

▶ Current permissions policies (1)

Other permissions policies (2/948)

Filter by Type: All types | 9 matches

Policy name	Type	Description
<input type="checkbox"/> AmazonKinesisAnalyticsFullAccess	AWS managed	Provides full access to Amazon Kinesis ...
<input type="checkbox"/> AmazonKinesisAnalyticsReadOnly	AWS managed	Provides read-only access to Amazon ...
<input type="checkbox"/> AmazonKinesisFirehoseFullAccess	AWS managed	Provides full access to all Amazon Kine...
<input type="checkbox"/> AmazonKinesisFirehoseReadOnlyAccess	AWS managed	Provides read only access to all Amazo...
<input checked="" type="checkbox"/> AmazonKinesisFullAccess	AWS managed	Provides full access to all streams via t...
<input type="checkbox"/> AmazonKinesisReadOnlyAccess	AWS managed	Provides read only access to all stream...
<input type="checkbox"/> AmazonKinesisVideoStreamsFullAccess	AWS managed	Provides full access to Amazon Kinesis ...
<input type="checkbox"/> AmazonKinesisVideoStreamsReadOnlyAccess	AWS managed	Provides read only access to AWS Kine...
<input checked="" type="checkbox"/> AWSLambdaKinesisExecutionRole	AWS managed	Provides list and read access to Kinesis...

Cancel | Add permissions

In Add Permissions search for these two policies:

- [AmazonKinesisFullAccess](#)
- [AWSLambdaKinesisExecutionRole](#)

Click on Add permissions.

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (3) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
<input type="checkbox"/> AmazonKinesisFullAccess	AWS managed	1
<input type="checkbox"/> AWSLambdaBasicExecutionRole-c6cc2...	Customer managed	1
<input type="checkbox"/> AWSLambdaKinesisExecutionRole	AWS managed	1

Next, you need to add a layer of Python coding to your function. In order to perform Python correctly.

Follow these steps in your command line:

```
mkdir python #create your directory if you need to
```

```
pip install requests -t python/ #install Python Requests, this will provide the files you need
```

```
zip -r Layers.zip python #create zip file
```

In our case, we used Jupyter Notebook as we already have Python installed.

```

jupyter Python_AWS_test Last Checkpoint: 16 minutes ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [1]: M import requests
In [2]: M !mkdir python
In [3]: M !pip install requests -t python/
In [4]: M !ls
In [5]: M !zip -r Layers.zip python
adding: python/ (164 bytes security) (stored 0%)
adding: python/bin/ (164 bytes security) (stored 0%)
adding: python/bin/normalizer.exe (164 bytes security) (deflated 51%)
adding: python/certifi/ (164 bytes security) (stored 0%)
adding: python/certifi/cacert.pem (164 bytes security) (deflated 46%)
adding: python/certifi/core.py (164 bytes security) (deflated 71%)
adding: python/certifi/py_typed (164 bytes security) (stored 0%)
adding: python/certifi/_init_.py (164 bytes security) (deflated 14%)
adding: python/certifi/_main_.py (164 bytes security) (deflated 39%)
adding: python/certifi/_pycache_/_ (164 bytes security) (stored 0%)
adding: python/certifi/_pycache_/_core.cpython-38.pyc (164 bytes security) (deflated 46%)
adding: python/certifi/_pycache_/_cpython-38.pyc (164 bytes security) (deflated 16%)
adding: python/certifi/_pycache_/_main_.cpython-38.pyc (164 bytes security) (deflated 24%)
adding: python/certifi/_pycache_/_main_.py (164 bytes security) (stored 0%)
adding: python/certifi/_pycache_/_2024.7.4.dist-info/ (164 bytes security) (stored 0%)
adding: python/certifi/_2024.7.4.dist-info/INITIALIZER (164 bytes security) (stored 42%)
adding: python/certifi/_2024.7.4.dist-info/LICENSE (164 bytes security) (deflated 42%)
adding: python/certifi/_2024.7.4.dist-info/METADATA (164 bytes security) (deflated 59%)
adding: python/certifi/_2024.7.4.dist-info/RECORD (164 bytes security) (deflated 43%)
adding: python/certifi/_2024.7.4.dist-info/top_level.txt (164 bytes security) (stored 0%)
In [ ]: M

```

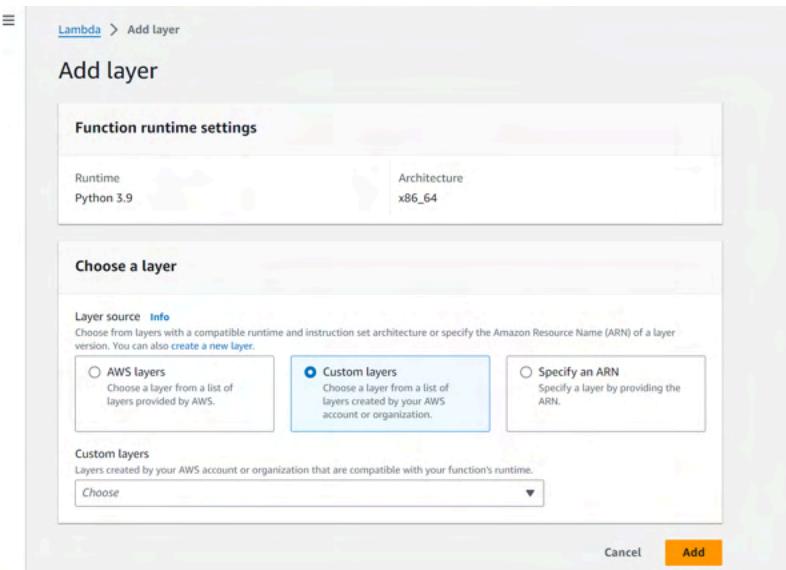
Don't worry about the error.

Use this zip file for Python layers: Layers.zip

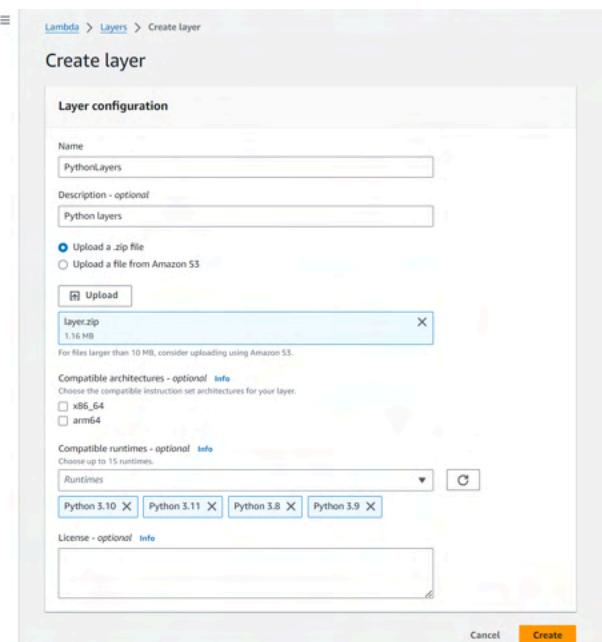
Go to Layers in your function.

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
There is no data to display.					

Click on Add a layer

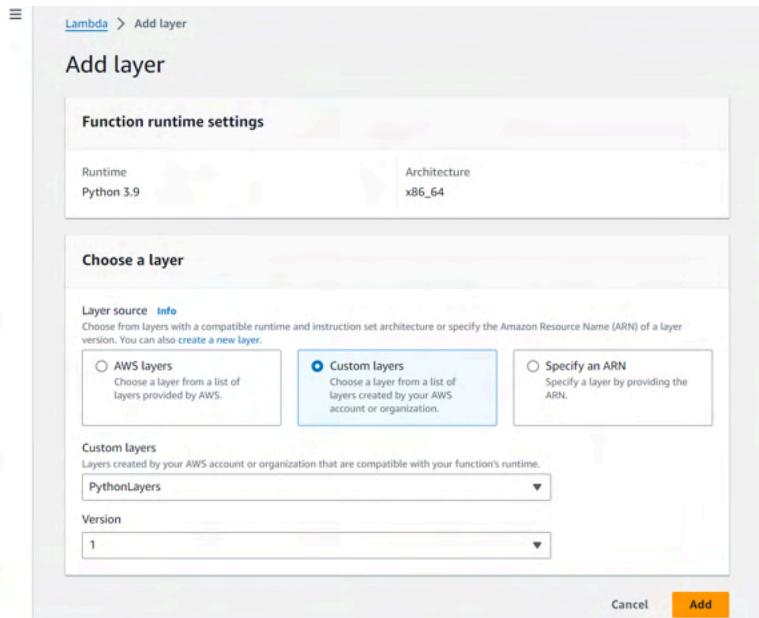


Click on 'Create a new layer'.



Provide a name for your layers and upload your zip file. Description and Compatible architectures are optional.

Select your compatible runtimes. In this case, select ALL versions of Python. Then click "Create".



Go back to *Add Layers*. Select custom layers and select the layer you created and the version.

Click “Add”.

Before making your Python code, select a database you will use as your sample test data. In this case, we will use data on the Iris flowers from UC Irvine Machine Learning databases:
<https://archive.ics.uci.edu/dataset/53/iris>

URL LINK: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

Enter the following code in Code Source under the lambda_function tab:

```
import json
```

```
import boto3 #python
import uuid #Universally Unique Identifier to identify object on internet
import requests #allows you to send HTTP requests in Python

def lambda_handler(event, context):
    kinesis_client = boto3.client('kinesis')
    data_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

    try:
        response = requests.get(data_url)
        response.raise_for_status()
        data_lines = response.text.strip().split('\n')

        #place in the names of each column, this will be better than putting it in the event
        JSON

        for line in data_lines:
            columns = line.split(',')
            if len(columns) == 5:
                data_to_send = {
                    'id': str(uuid.uuid4()),
                    'sepal length': columns[0],
                    'sepal width': columns[1],
                    'petal length': columns[2],
                    'petal width': columns[3],
                    'class': columns[4],
                    'source url': data_url #url link
                }

                response = kinesis_client.put_record(
```

```
    StreamName='simpliproject2_Kinesis', #name of your Kinesis data stream
    Data=json.dumps(data_to_send), # Data sent to Kinesis, ensure it's
JSON-encoded
    PartitionKey=data_to_send['id'] # key for partitioning records
)
print(f"Record sent to Kinesis: {response}")

return {
    'statusCode': 200,
    'body': json.dumps('All records sent to Kinesis data stream.')
}
```

```
#Notification on errors and causes
except requests.RequestException as e:
    print(f"Error fetching data from URL: {e}")
return {
    'statusCode': 500,
    'body': json.dumps(f'Error fetching data: {str(e)}')
}
```

The status code 200 is the HTTP 200 OK, meaning a request has succeeded and was successful.

The status code 500 is HTTP 500 Internal Server Error server error response code. This indicates that the server encountered an unexpected condition that prevented it from fulfilling the request.

The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test', 'Deploy', and a gear icon. The main area displays the 'lambda_function' code in Python:

```
import json
import boto3
import uuid #Universally Unique Identifier to identify object on internet
import requests #Allows you to send HTTP requests in Python

def lambda_handler(event, context):
    kinesis_client = boto3.client('kinesis')
    data_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

    try:
        response = requests.get(data_url)
        response.raise_for_status()
        data_lines = response.text.strip().split("\n")
    except requests.exceptions.RequestException as e:
        print(f"Error fetching data from URL: {e}")
        return {
            'statusCode': 500,
            'body': json.dumps(f'Error fetching data: {str(e)}')
        }

    #Place in the names of each column, this will be better than putting it in the event JSON
    for line in data_lines:
        columns = line.split(',')
        if len(columns) == 5:
            data_to_send = {
                'id': str(uuid.uuid4()),
                'sepal length': columns[0],
                'sepal width': columns[1],
                'petal length': columns[2],
                'petal width': columns[3],
                'class': columns[4],
                'source url': data_url #url link
            }

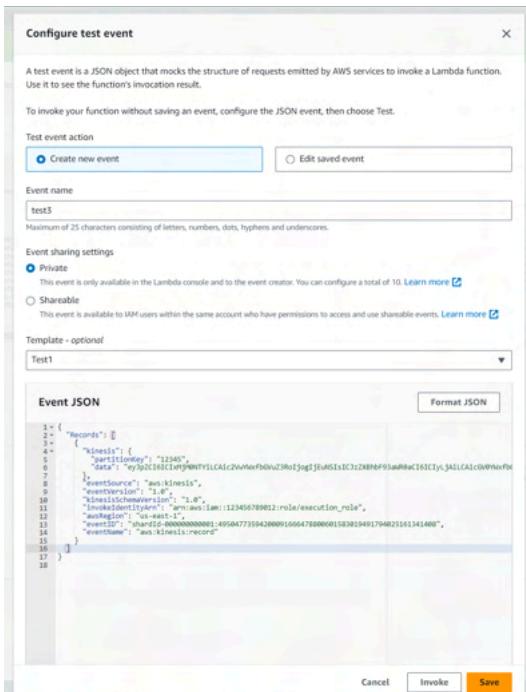
            response = kinesis_client.put_record(
                StreamName='simlproject2_Kinesis', #name of your Kinesis data stream
                Data=json.dumps(data_to_send), #Data sent to Kinesis, ensure it's JSON-encoded
                PartitionKey=data_to_send['id'] #key for partitioning records
            )
            print(f"Record sent to Kinesis: {response}")

    return {
        'statusCode': 200,
        'body': json.dumps('All records sent to Kinesis data stream.')
    }
```

The bottom status bar indicates "10:9 Python Spaces: 4".

Click Deploy.

In Test, click on configure test event.



Give a name to your test. In this case, "test3" (multiple tests were used as trial runs).

To ensure that the data will be sent to the `DynamoTable`:

```
{
  "Records": [
    {
      "kinesis": {
        "partitionKey": "12345",
        "data": "eyJpZCI6IClxMjM0NTYiLCAic2VwYWxfbGVuZ3R0jogljEuNSIsICJzZXBhbF93aWR0aCI6I
ClyLjAiLCAicGV0YWxfbGVuZ3R0jogljEuMCIsICJwZXRhBf93aWR0aCI6IClwljAiLCAlY2x
hc3MiOiAiU29tZSBDbGFzcylsICJzb3VyY2VfdXJsljoglmh0dHBzOi8vYXJjaGI2ZS5pY3MuZ
WN1LmVkdS9tbC9tYWNoaW5ILWxIYXJuaW5nLWRhdGFzYmFzZXMvcXJpcy9pcnMuZGF
0YSJ9",
        },
      "eventSource": "aws:kinesis",
      "eventVersion": "1.0",
      "kinesisSchemaVersion": "1.0",
      "invokedIdentityArn": "arn:aws:iam::123456789012:role/execution_role",
      "awsRegion": "us-east-1",
      "eventID": "shardId-000000000001:49504773594200091666478800601583019491794025161341408",
      "eventName": "aws:kinesis:record"
    }
  ]
}
```

This took multiple trials. Do not be worried to experiment.

Same successful event. Move on to the next Lambda Function.

2.2 Lambda Consumer

Repeat the same steps in creating the lambda function but have it be named "Project2_LambdaFn_Consumer".

This is your consumer lambda function that will be connected to your DynamnoDB table and Kinesis trigger.

Adjust the timeout to 5 minutes as well.

Then go to your Lambda function Role: Project2_LambdaFn_Consumer-role-xrqynfra.

Execution role

Role name: Project2_LambdaFn_Consumer-role-xrqynfra

Resource summary

To view the resources and actions that your function has permission to access, choose a service.

Resource	Actions
arn:aws:logs:us-east-1:339039570958:*	Allow: logs>CreateLogGroup
arn:aws:logs:us-east-1:339039570958:log-group:/aws/lambda/Project2_LambdaFn_Consumer:*	Allow: logs>CreateLogStream Allow: logs:PutLogEvents

Lambda obtained this information from the following policy statements:

- Managed policy AWSLambdaBasicExecutionRole-20f62a7c-7799-471d-a737-a1ac37cc27c3, statement 0

Add these policies to your consumer function:

- **AmazonKinesisFullAccess**
- **AWSLambdaKinesisExecutionRole**
- **AmazonDynamoDBFullAccess**

Attach policy to Project2_LambdaFn_Consumer-role-xrqynfra

▶ Current permissions policies (1)

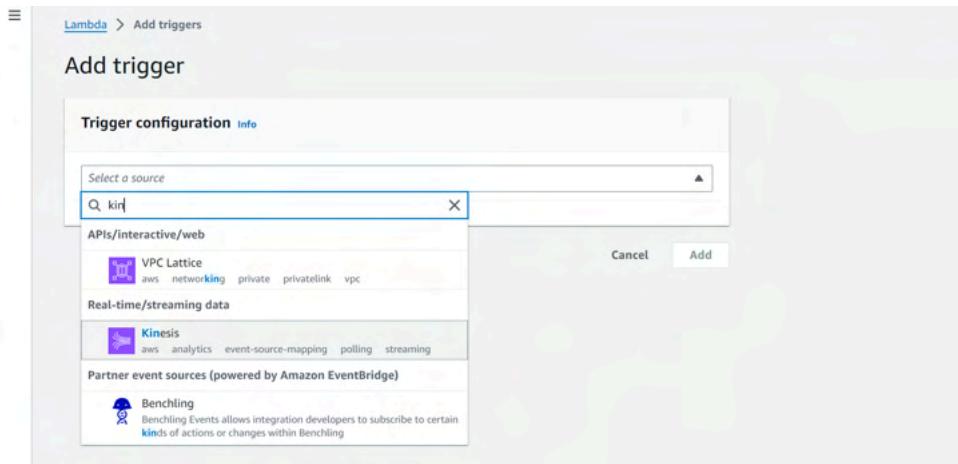
Other permissions policies (3/949)

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon Dynam...
<input type="checkbox"/> AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon D...
<input type="checkbox"/> AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to Dynam...
<input type="checkbox"/> AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Str...

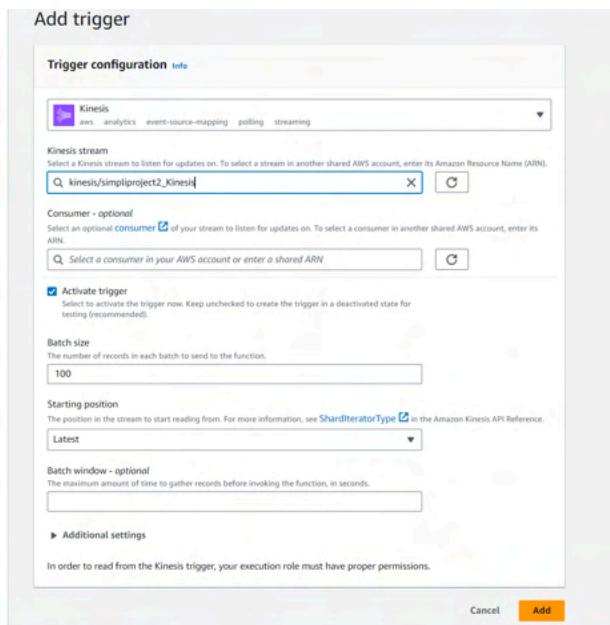
[Cancel](#) [Add permissions](#)

Add permissions. Then return to Lambda consumer function.

Go to Add Trigger.



Type in Kinesis.



Select your data stream “simpliproject2_Kinesis”. Then click on Add.

Before continuing, we recommend you make your DynamoDB table first then enter the code for consumer function to test your data.

Step 3: Create DynamoDB table

3.1 Create Table

Go to DynamoDB

Go to Tables. Click on Create Table.

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
 Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
 String ▾
1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 String ▾
1 to 255 characters and case sensitive.

Table settings

Default settings
The fastest way to create your table. You can modify these settings now or after your table has been created.

Customize settings
Use these advanced features to make DynamoDB work better for your needs.

Enter the desired Table name and Partition key. Ignore sort key.

For me: Table name is 'Book1_Iris' and Partition Key is 'id'.

NOTE: BE SURE your partition key matches with the first name column in your data set.

Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

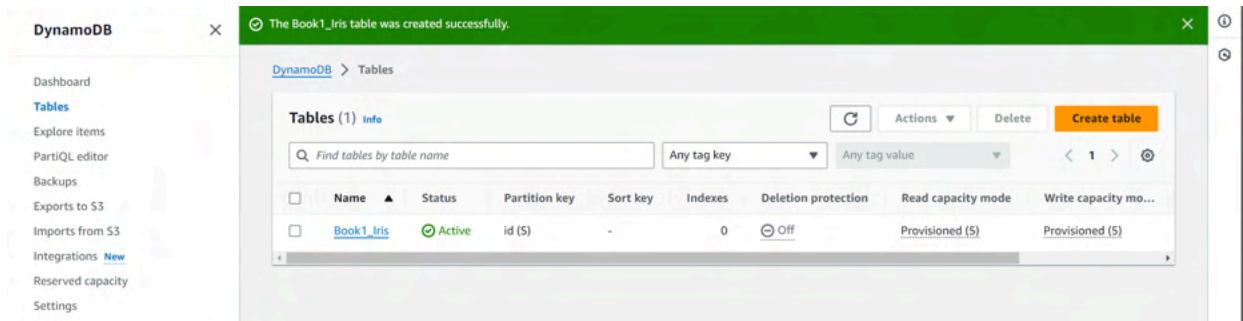
Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag
You can add 50 more tags.

Cancel

Click on Create table.



DynamoDB created.

3.2 Lambda Consumer Code

Return to your consumer lambda function.

```
import json
import boto3
import base64 #built-in module in Python
from botocore.exceptions import ClientError
#catching botocore exception

# Initialize DynamoDB client
dynamodb_client = boto3.resource('dynamodb')
table = dynamodb_client.Table('Book1_Iris') #DynamoDB table name

def lambda_handler(event, context):
    for record in event.get('Records', []):
        try:
            # Decode Kinesis data from Base64
            encoded_data = record['kinesis']['data']
            decoded_data = base64.b64decode(encoded_data)
            package = json.loads(decoded_data)

            # Extract data fields
            id = package.get('id')
```

```
sepal_length = package.get('sepal length')
sepal_width = package.get('sepal width')
petal_length = package.get('petal length')
petal_width = package.get('petal width')
iris_class = package.get('class')
source_url = package.get('source url') #Extract URL

# Write to DynamoDB
try:
    table.put_item(
        Item={
            'id': id,
            'sepal length': sepal_length,
            'sepal width': sepal_width,
            'petal length': petal_length,
            'petal width': petal_width,
            'class': iris_class,
            'source url': source_url # Store URL in DynamoDB
        }
    )
    print(f"Record wrote with id: {id}")
except ClientError as e:
    print(f"Error writing record to DynamoDB: {e}")
except Exception as e:
    print(f"Error processing record: {e}")

return {
```

```

'statusCode': 200,
'body': json.dumps('Data processed and stored.')
}

```

The status code 200 is the HTTP 200 OK, meaning a request has succeeded.

This screenshot shows the AWS Lambda Test interface. The top navigation bar includes File, Edit, Find, View, Go, Tools, Window, Test, Deploy, and a gear icon. Below the navigation is a search bar labeled "Go to Anything (Ctrl-P)". The main area displays the "lambda_function" code in a Python file named "lambda_function.py". The code reads Kinesis data from a stream, decodes it, and writes it to a DynamoDB table named "Book1_Iris". It handles exceptions and returns a JSON response with status code 200 and a success message. The bottom right corner shows the timestamp "17:47" and "Python Spaces: 4".

```

1 import json
2 import boto3
3 import base64 #built-in module in Python
4 from botocore.exceptions import ClientError
5 #catching botocore exception
6
7 # Initializing Dynamodb client
8 dynamodb_client = boto3.resource('dynamodb')
9 table = dynamodb_client.Table('Book1_Iris') #DynamodDB table name
10
11 def lambda_handler(event, context):
12     for record in event.get('Records', []):
13         try:
14             # Decode Kinesis data from Base64
15             encoded_data = record['kinesis']['data']
16             decoded_data = base64.b64decode(encoded_data)
17             package = json.loads(decoded_data)
18
19             # Extract data fields
20             id = package.get('id')
21             sepal_length = package.get('sepal length')
22             sepal_width = package.get('sepal width')
23             petal_length = package.get('petal length')
24             petal_width = package.get('petal width')
25             iris_class = package.get('class')
26             source_url = package.get('source url') #Extract URL
27
28             # Write to Dynamodb
29             try:
30                 table.put_item(
31                     Item={
32                         'id': id,
33                         'sepal length': sepal_length,
34                         'sepal width': sepal_width,
35                         'petal length': petal_length,
36                         'petal width': petal_width,
37                         'class': iris_class,
38                         'source url': source_url # Store URL in dynamoDB
39                     }
40                 )
41                 print(f"Record wrote with id: {id}")
42             except ClientError as e:
43                 print(f"Error writing record to dynamoDB: {e}")
44
45             except Exception as e:
46                 print(f"Error processing record: {e}")
47
48         return {
49             'statusCode': 200,
50             'body': json.dumps('Data processed and stored.')
51         }
52     }

```

This screenshot shows the AWS Lambda Test interface with the same basic structure as the previous one, but with more comprehensive error handling. The code now includes a try-except block to catch both ClientError and generic Exception types, printing detailed error messages to the log. The rest of the function logic remains the same, reading from Kinesis, decoding data, and writing to DynamoDB.

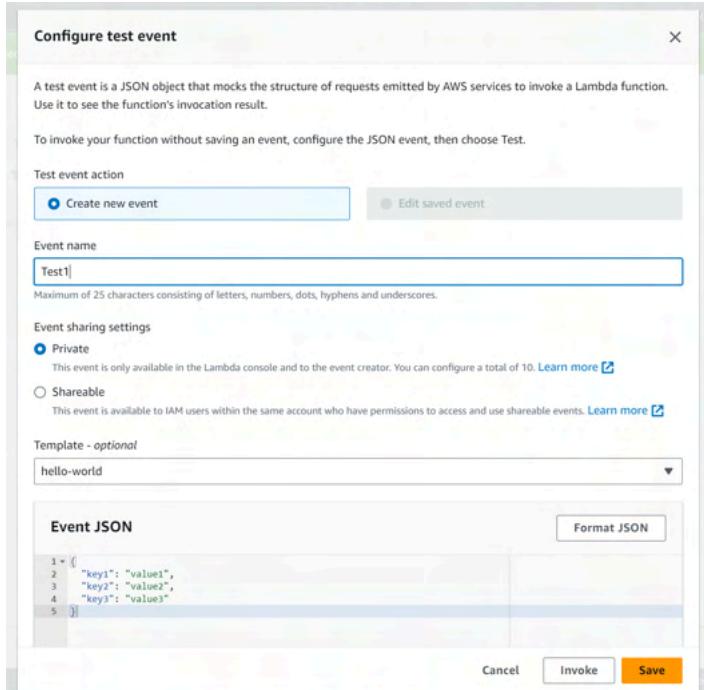
```

19             # Extract data fields
20             id = package.get('id')
21             sepal_length = package.get('sepal length')
22             sepal_width = package.get('sepal width')
23             petal_length = package.get('petal length')
24             petal_width = package.get('petal width')
25             iris_class = package.get('class')
26             source_url = package.get('source url') #Extract URL
27
28             # Write to Dynamodb
29             try:
30                 table.put_item(
31                     Item={
32                         'id': id,
33                         'sepal length': sepal_length,
34                         'sepal width': sepal_width,
35                         'petal length': petal_length,
36                         'petal width': petal_width,
37                         'class': iris_class,
38                         'source url': source_url # Store URL in dynamoDB
39                     }
40                 )
41                 print(f"Record wrote with id: {id}")
42             except ClientError as e:
43                 print(f"Error writing record to dynamoDB: {e}")
44
45             except Exception as e:
46                 print(f"Error processing record: {e}")
47
48         return {
49             'statusCode': 200,
50             'body': json.dumps('Data processed and stored.')
51         }
52     }

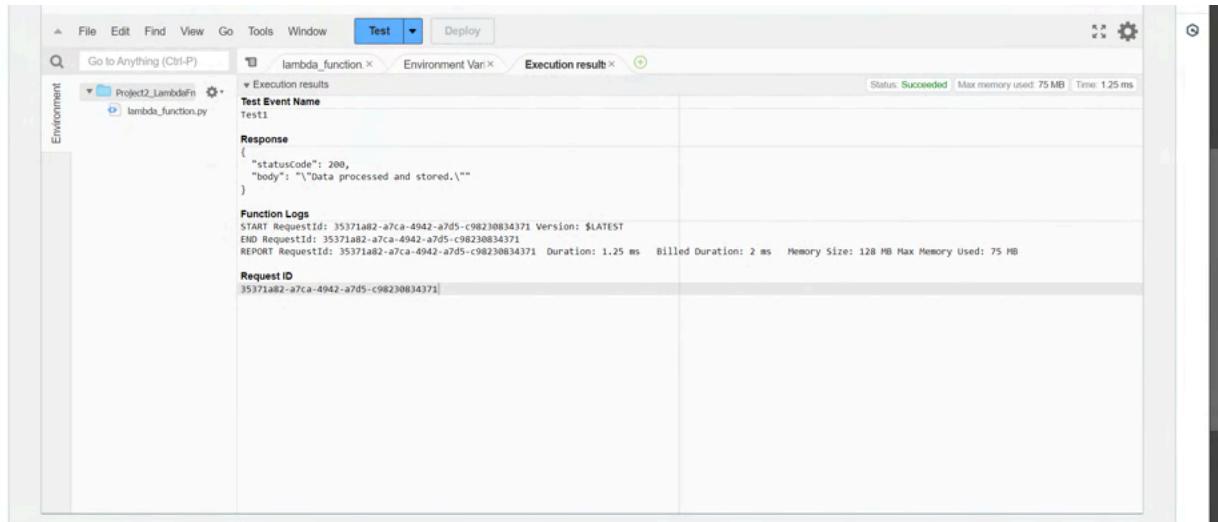
```

Press Deploy.

For Test, create a test name. Leave the JSON alone.



Run Test.



Test successful.

3.3 Upload Data to Table

Go to your DynamoDB table.

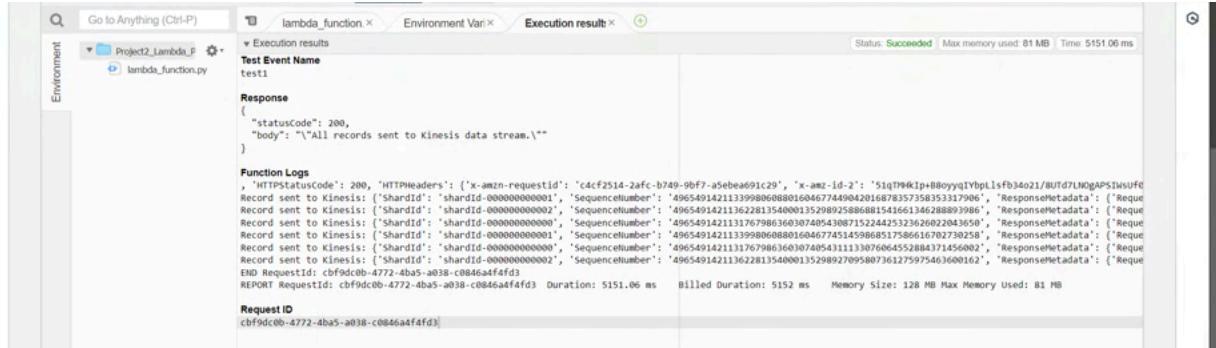
The screenshot shows the AWS DynamoDB console. On the left, the navigation bar includes 'Dashboard', 'Tables' (which is selected), 'Explore items', 'PartQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Under 'Tables', there's a 'DAX' section with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area displays the 'Tables (1) Info' page for 'Book1_Iris'. The table has one item: Name (Book1_Iris), Status (Active), Partition key (id (\$)), Sort key (-), Indexes (0), Deletion protection (Off), Read capacity mode (Provisioned (1)), and Write capacity (Provisioned (1)). Below this, the 'Book1_Iris' table details page is shown, featuring tabs for Overview, Indexes, Monitor, Global tables, Backups, and Exports and streams. It highlights Point-in-time recovery (PITR) and shows general information like Partition key (id String), Sort key (-), Capacity mode (Provisioned), and Table status (Active). A note says 'Protect your DynamoDB table from accidental writes and deletes'.

Click on 'Explore table items'.

The screenshot shows the 'Explore items' page for the 'Book1_Iris' table. The left sidebar is identical to the previous screenshot. The main area shows the 'Scan or query items' interface. It has two tabs: 'Scan' (selected) and 'Query'. It includes fields for 'Select a table or index' (set to 'Table - Book1_Iris') and 'Select attribute projection' (set to 'All attributes'). Below these are 'Filters' and 'Run' and 'Reset' buttons. A success message at the bottom says 'Completed. Read capacity units consumed: 0.5'. The 'Items returned (0)' section shows 'No items' and 'No items to display.' with a 'Create item' button.

You will see **Items returned (0)**.

Go to your Lambda producer function. Click on Test, be sure it is set your test event you want.



The screenshot shows the AWS Lambda console interface. In the top navigation bar, the project name "Project2_Lambda_P" is selected. The "lambda_function" tab is active, and the "Execution result" tab is selected. The "Test Event Name" dropdown is set to "test1". The "Response" section displays the following JSON output:

```
{ "statusCode": 200, "body": "\"All records sent to Kinesis data stream.\""}
```

The "Function Logs" section shows a detailed log of the Lambda function's execution, including records sent to Kinesis. The log entries are as follows:

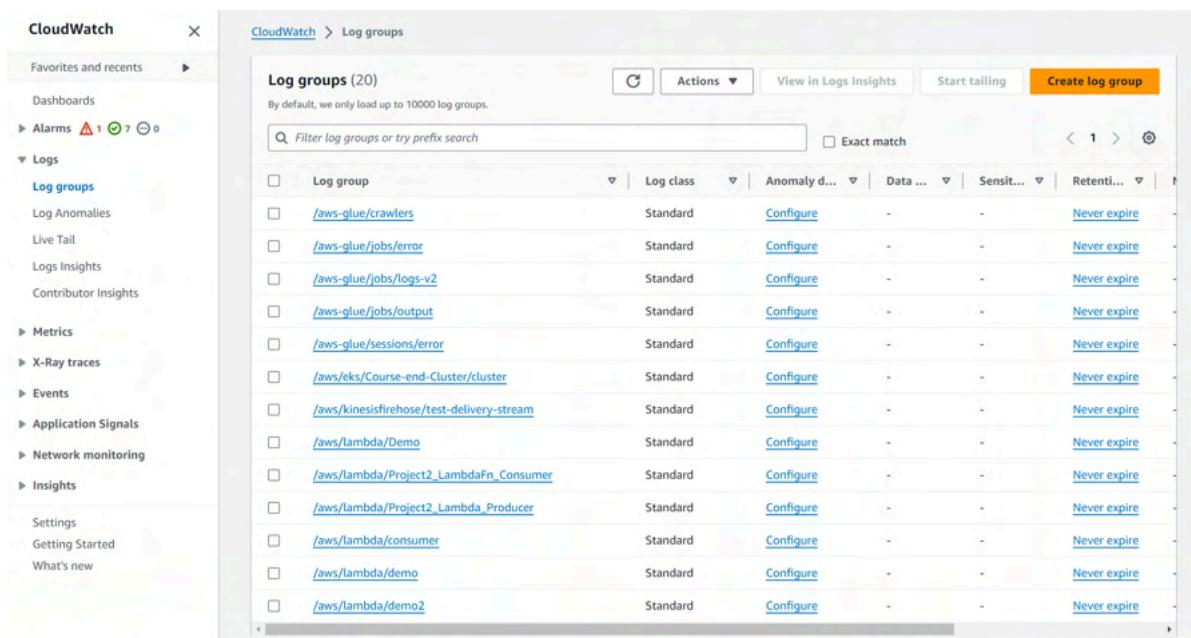
```
HTTPStatus: 200, 'HTTPHeaders': {'x-amz-requestid': 'c4cf2514-2afc-b749-9bf7-a5ebead91c29', 'x-amz-id-2': '51qTWRkIp+88oyyaIVbplLsfb3A021/8UfTdLN0gAPSJkwufc'}, Record sent to Kinesis: ('shardId': 'shardId-000000000001', 'SequenceNumber': '40654914211133000060000160467744994201687935753327996', 'ResponseMetadata': {'RequestId': 'cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3', 'ReportSignature': 'REPORT RequestId: cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3 Duration: 5151.06 ms Billed Duration: 5152 ms Memory Size: 128 MB Max Memory Used: 81 MB'}}, Record sent to Kinesis: ('shardId': 'shardId-000000000002', 'SequenceNumber': '4065491421113622813540001352980258868815416314628889598', 'ResponseMetadata': {'RequestId': 'cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3', 'ReportSignature': 'REPORT RequestId: cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3 Duration: 5151.06 ms Billed Duration: 5152 ms Memory Size: 128 MB Max Memory Used: 81 MB'}}, Record sent to Kinesis: ('shardId': 'shardId-000000000000', 'SequenceNumber': '40654914211137679863603074054308715224425323626022043650', 'ResponseMetadata': {'RequestId': 'cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3', 'ReportSignature': 'REPORT RequestId: cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3 Duration: 5151.06 ms Billed Duration: 5152 ms Memory Size: 128 MB Max Memory Used: 81 MB'}}, Record sent to Kinesis: ('shardId': 'shardId-000000000001', 'SequenceNumber': '406549142111339980688001604677451459868517586616702730258', 'ResponseMetadata': {'RequestId': 'cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3', 'ReportSignature': 'REPORT RequestId: cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3 Duration: 5151.06 ms Billed Duration: 5152 ms Memory Size: 128 MB Max Memory Used: 81 MB'}}, Record sent to Kinesis: ('shardId': 'shardId-000000000000', 'SequenceNumber': '40654914211137679863603074054311133076064552848471456000', 'ResponseMetadata': {'RequestId': 'cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3', 'ReportSignature': 'REPORT RequestId: cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3 Duration: 5151.06 ms Billed Duration: 5152 ms Memory Size: 128 MB Max Memory Used: 81 MB'}}, Record sent to Kinesis: ('shardId': 'shardId-000000000002', 'SequenceNumber': '406549142111362281354000135298027095807361275975463600162', 'ResponseMetadata': {'RequestId': 'cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3', 'ReportSignature': 'REPORT RequestId: cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3 Duration: 5151.06 ms Billed Duration: 5152 ms Memory Size: 128 MB Max Memory Used: 81 MB'}}, END RequestId: cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3 Duration: 5151.06 ms Billed Duration: 5152 ms Memory Size: 128 MB Max Memory Used: 81 MB
```

The "Request ID" field shows "cbf9dc0b-4772-4ba5-a038-c0846aa4f4fd3".

Results indicate that data has been sent to your DynamoDB Table.

To check, go to *CloudWatch*.

Go to Log Groups.



The screenshot shows the CloudWatch Metrics & Logs interface. The left sidebar is collapsed, showing "Logs" under "Logs". The main area is titled "Log groups (20)" and displays a table of log groups. The columns are: Log group, Log class, Anomaly d..., Data ..., Sensit..., Retentio..., and Last activity. The table lists 20 log groups, all of which are Standard log classes and have "Configure" actions. Most log groups have a retention period of "Never expire".

Log group	Log class	Anomaly d...	Data ...	Sensit...	Retentio...
/aws-glue/crawlers	Standard	Configure	-	-	Never expire
/aws-glue/jobs/error	Standard	Configure	-	-	Never expire
/aws-glue/jobs/logs-v2	Standard	Configure	-	-	Never expire
/aws-glue/jobs/output	Standard	Configure	-	-	Never expire
/aws-glue/sessions/error	Standard	Configure	-	-	Never expire
/aws/eks/Course-end-Cluster/cluster	Standard	Configure	-	-	Never expire
/aws/kinesisfirehose/test-delivery-stream	Standard	Configure	-	-	Never expire
/aws/lambda/Demo	Standard	Configure	-	-	Never expire
/aws/lambda/Project2_LambdaFn_Consumer	Standard	Configure	-	-	Never expire
/aws/lambda/Project2_Lambda_Producer	Standard	Configure	-	-	Never expire
/aws/lambda/consumer	Standard	Configure	-	-	Never expire
/aws/lambda/demo	Standard	Configure	-	-	Never expire
/aws/lambda/demo2	Standard	Configure	-	-	Never expire

For Lambda Producer

Log group details

Log class	Info	Metric filters	Anomaly detection
Standard	0	Subscription filters	Configure
ARN	arn:aws:logs:us-east-1:339039570958:log-group:/aws/lambda/Project2_Lambda_Producer;	Contributor Insights rules	Data protection
Creation time	3 hours ago	KMS key ID	Sensitive data count
Retention	Never expire		
Stored bytes	-		

Log streams (12)

Last event time
2024-08-15 00:07:37 (UTC)
2024-08-14 23:56:46 (UTC)
2024-08-14 23:29:17 (UTC)
2024-08-14 23:24:57 (UTC)

Click on the latest log stream.

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](#)

Timestamp	Message
2024-08-15T00:07:33.616Z	INIT_START Runtime Version: python:3.9.v55 Runtime Version ARN: arn:aws:lambda:us-east-1:runtime:be711d01364010e86158038000656c23df979e481793ee37b0e1b79fda12
2024-08-15T00:07:34.204Z	START RequestId: 2f60de09-7842-4838-96d6-e53d2ae947ef Version: \$LATEST
2024-08-15T00:07:36.582Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.615Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.655Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '496548826213108733151624004667704227732011395867360757762', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.663Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.715Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.755Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.797Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.932Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.942Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315162400466770424362488331565196535463910', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:36.975Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000001', 'SequenceNumber': '4965488262131087331517401438577300191255280231900133800861426', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:37.015Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000001', 'SequenceNumber': '4965488262131087331517401438577300191255280231900133800861426', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:37.052Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000001', 'SequenceNumber': '4965488262131087331517401438577300191255280231900133800861426', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:37.075Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000001', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:37.115Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:37.155Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000002', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:37.175Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000003', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:41.715Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:41.735Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000002', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:41.775Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000000', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:41.815Z	Record sent to Kinesis: ('ShardId': 'shardId-000000000001', 'SequenceNumber': '49654882621310873315174014385773001914852299482745122001379954', 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'HTTPHeaders': {}, 'Latency': 0}, 'Kinesis': {'PartitionKey': '...', 'Data': '...'})
2024-08-15T00:07:41.837Z	END RequestId: 2f60de09-7842-4838-96d6-e53d2ae047ef
2024-08-15T00:07:41.837Z	REPORT RequestId: 2f60de09-7842-4838-96d6-e53d2ae047ef Duration: 763.40 ms Billed Duration: 763 ms Memory Size: 128 MB Max Memory Used: 79 MB Init Duration: 587.22 ms

Now check your Lambda Consumer.

The screenshot shows the AWS CloudWatch Logs interface. On the left, the navigation pane includes sections for Favorites and recent dashboards, Alarms, Logs (selected), Log groups (highlighted in blue), Log Anomalies, Live Tail, Logs Insights, Contributor Insights, Metrics, X-Ray traces, Events, Application Signals, Network monitoring, and Insights. Under Logs, there are sub-sections for Settings, Getting Started, and What's new.

The main content area displays the details for the log group `/aws/lambda/Project2_LambdaFn_Consumer`. It shows the Log class as Info, Standard type, ARN as `arn:aws:logs:us-east-1:339039570958:log-group:/aws/lambda/Project2_LambdaFn_Consumer:*`, Creation time as 2 hours ago, Retention as Never expire, and Stored bytes as ~. Metric filters, Subscription filters, and Contributor Insights rules are listed under the Metrics tab. Data protection and Sensitive data count are listed under the Data protection tab.

Below the details, a table lists 21 log streams, each with a timestamp and duration:

Log stream	Last event time
2024/08/15/[SLATEST]77c35605c3174b439b362c9cf2a84459	2024-08-15 00:07:42 (UTC)
2024/08/15/[SLATEST]a1feac78621466fa43b1f42a87b94	2024-08-15 00:07:42 (UTC)
2024/08/15/[SLATEST]1a04740225d74b8090759069c0f6b57d	2024-08-15 00:07:39 (UTC)
2024/08/15/[SLATEST]255e605bad3c475ab6f217881bee56c4	2024-08-15 00:07:37 (UTC)

Click on the latest log stream.

The screenshot shows the AWS CloudWatch Log Events interface, specifically for the log stream `2024/08/15/[SLATEST]77c35605c3174b439b362c9cf2a84459`. The left sidebar is identical to the previous screenshot. The main area displays a table of log events with columns for Timestamp and Message.

The table shows numerous log entries, each with a timestamp and a detailed message. For example, the first entry is `2024-08-15T00:07:37.795Z INIT START Runtime Version: python3.9.v55 Runtime Version ARN: arn:aws:lambda:us-east-1:runtime:be9721d326401e86158b38d000656c23d4f970e0401793ee3709e2b79fda22`. The last few entries are:

Timestamp	Message
2024-08-15T00:07:41.756Z	END RequestId: 9047d175-cd24-476f-9e07-9d18137551ac REPORT RequestId: 9047d175-cd24-476f-9e07-9d18137551ac Duration: 100.33 ms Billed Duration: 101 ms Memory Size: 128 MB Max Memory Used: 74 MB
2024-08-15T00:07:41.658Z	START RequestId: 53618981-59c6-4100-9e02-33d7ae67c99e Version: \$LATEST
2024-08-15T00:07:41.677Z	Record wrote with Id: ab331d90-fc87-48d6-85fe-2e180d6cbab8
2024-08-15T00:07:41.697Z	END RequestId: 53618981-59c6-4100-9e02-33d7ae67c99e REPORT RequestId: 53618981-59c6-4100-9e02-33d7ae67c99e Duration: 38.87 ms Billed Duration: 39 ms Memory Size: 128 MB Max Memory Used: 74 MB

Cloud indicates data stream went through.

Now go back to DynamoDB Table.

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with navigation links like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, Settings, DAX Clusters, Subnet groups, Parameter groups, and Events. The main area is titled "Book1_Iris". It has a header with "Scan or query items" and tabs for "Scan" (selected) and "Query". Below that, it says "Select a table or index" and "Table: Book1_Iris". There's also a "Select attribute projection" dropdown set to "All attributes". A note says "This table has more items to retrieve. To retrieve the next page of items, choose Retrieve next page." At the bottom, there are two tables showing "Items returned (50)". Each table has columns: id (String), class, petal length, petal width, sepal length, sepal width, and source url. The first table has 5 rows of data, and the second table has 50 rows of data.

	id (String)	class	petal length	petal width	sepal length	sepal width	source url
1	391c295a-7f93-47ce...	iris-setosa	1.4	0.3	4.8	3.0	https://archive.ics.uci...
2	79a0cd64-a56e-4e93...	iris-setosa	1.0	0.2	4.6	3.6	https://archive.ics.uci...
3	7da6c298-9532-4507...	iris-setosa	1.4	0.2	5.0	3.6	https://archive.ics.uci...
4	a21a963e-5a29-4466...	iris-virginica	5.0	1.5	6.0	2.2	https://archive.ics.uci...
5	faff7e15-3365-4a6b...	iris-setosa	1.2	0.2	5.0	3.2	https://archive.ics.uci...
...
50	e2a7a118-1d49-4a73...	iris-versicolor	4.9	1.4	4.9	1.5	https://archive.ics.uci...

Data has been successfully. Uploaded.

Step 4: Perform Scan Operation

4.1 Scan Option

From the upload data, test Scan & Query options.

In “Scan or query items”, Select Scan.

Select Filter. With this option, you can filter out specific items in your data table.

First trial, have attribute name be “class”. Type as String, condition to “equal to”, and value put in “Iris-versicolor”.

Click on Run.

The screenshot shows the AWS DynamoDB console. On the left, the navigation pane includes 'Dashboard', 'Tables', 'Explore items' (which is selected), 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Under 'DAX', there are 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area is titled 'Book1_Iris' and contains a 'Scan or query items' section. It has a 'Scan' button (selected) and a 'Query' button. Below these are dropdowns for 'Table - Book1_Iris' and 'Select attribute projection' (set to 'All attributes'). A 'Filters' section shows a single filter: Attribute name 'class' (String type, set to 'Equal to' with value 'Iris-versicolor'). There are 'Add filter' and 'Run' buttons. A note at the bottom says 'This table has more items to retrieve. To retrieve the next page of items, choose Retrieve next page.' A 'Run' button is highlighted in orange.

The screenshot shows the results of the query. The title is 'Items returned (50)'. The table has columns: id (String), class, petal length, petal width, sepal length, sepal width, and source url. The first few rows are:

	id (String)	class	petal length	petal width	sepal length	sepal width	source url
<input type="checkbox"/>	59a7a338-1dd0-4e22...	Iris-versicolor	4.2	1.5	5.9	3.0	https://archive.ics.uci.edu...
<input type="checkbox"/>	a8dd644a-c0d3-480a...	Iris-versicolor	3.8	1.1	5.5	2.4	https://archive.ics.uci.edu...
<input type="checkbox"/>	c28da7b0-aa11-4561...	Iris-versicolor	3.3	1.0	4.9	2.4	https://archive.ics.uci.edu...
<input type="checkbox"/>	efdd4a7b2-a448-45c6...	Iris-versicolor	4.4	1.2	5.5	2.6	https://archive.ics.uci.edu...
<input type="checkbox"/>	8e5df4ab-b73a-4414...	Iris-versicolor	3.8	1.1	5.5	2.4	https://archive.ics.uci.edu...
<input type="checkbox"/>	0e49812b-e894-49b5...	Iris-versicolor	4.5	1.5	6.2	2.2	https://archive.ics.uci.edu...
<input type="checkbox"/>	e51fc716-e48f-45dc...	Iris-versicolor	4.7	1.6	6.3	3.3	https://archive.ics.uci.edu...
<input type="checkbox"/>	25d95d68-5cba-4df8...	Iris-versicolor	4.2	1.2	5.7	3.0	https://archive.ics.uci.edu...
<input type="checkbox"/>	551484a4-be3b-46e1...	Iris-versicolor	4.0	1.3	5.5	2.3	https://archive.ics.uci.edu...
<input type="checkbox"/>	97df51a-445b-4ec8...	Iris-versicolor	4.4	1.3	6.3	2.3	https://archive.ics.uci.edu...
<input type="checkbox"/>	b6df5845-b3e1-4c78...	Iris-versicolor	3.9	1.4	5.2	2.7	https://archive.ics.uci.edu...
<input type="checkbox"/>	a4a33bd8-af8f-467c...	Iris-versicolor	3.3	1.0	5.0	2.3	https://archive.ics.uci.edu...
<input type="checkbox"/>	2d5de4e3-a3b8-404c...	Iris-versicolor	4.4	1.4	6.7	3.1	https://archive.ics.uci.edu...
<input type="checkbox"/>	969e0fc1-8c5f-4739...	Iris-versicolor	3.9	1.2	5.8	2.7	https://archive.ics.uci.edu...
<input type="checkbox"/>	c1fc710d-2336-41f9...	Iris-versicolor	5.1	1.6	6.0	2.7	https://archive.ics.uci.edu...
<input type="checkbox"/>	251de0a1-0a7a-42bb...	Iris-versicolor	4.0	1.0	6.0	2.2	https://archive.ics.uci.edu...
<input type="checkbox"/>	2ad79448-470a-470a...	Iris-versicolor	4.6	1.5	5.0	2.0	https://archive.ics.uci.edu...

Your data is filtered to a specific class of irises based on your request.

Now try, attribute “petal width” and have condition set to “less than or equal to” at 1.0.

DynamoDB > Explore items > Book1_Iris

Book1_Iris

Scan or query items

Scan Query

Select a table or index: Table - Book1_Iris

Filters

Attribute name	Type	Begins with	Value
petal width	String	Less than or equal to	1.0

Add filter

Run Reset

This table has more items to retrieve. To retrieve the next page of items, choose Retrieve next page.

Autopreview View table details

Click on Run.

Items returned (50)

	id (String)	class	petal length	petal width	sepal length	sepal width	source url
<input type="checkbox"/>	faff7e15-3365-4a6b...	Iris-setosa	1.2	0.2	5.0	3.2	https://archive.ics.uci.edu...
<input type="checkbox"/>	f7beb361-345f-4aa7...	Iris-setosa	1.5	0.4	5.4	3.4	https://archive.ics.uci.edu...
<input type="checkbox"/>	e6feabf9-a3d0-479b...	Iris-setosa	1.4	0.2	5.5	4.2	https://archive.ics.uci.edu...
<input type="checkbox"/>	e369a0fb-67bf-4b64...	Iris-versicolor	3.5	1.0	5.0	2.0	https://archive.ics.uci.edu...
<input type="checkbox"/>	e0e37f92-9837-4979...	Iris-setosa	1.6	0.6	5.0	3.5	https://archive.ics.uci.edu...
<input type="checkbox"/>	e04d54be-a2b5-40e1...	Iris-setosa	1.4	0.2	5.1	3.5	https://archive.ics.uci.edu...
<input type="checkbox"/>	d0f0ed84-2723-4e86...	Iris-setosa	1.1	0.1	4.3	3.0	https://archive.ics.uci.edu...
<input type="checkbox"/>	c6620396-5185-4340...	Iris-setosa	1.7	0.2	5.4	3.4	https://archive.ics.uci.edu...
<input type="checkbox"/>	c2d4f6d4-5ca7-4ffb...	Iris-setosa	1.5	0.2	5.3	3.7	https://archive.ics.uci.edu...
<input type="checkbox"/>	c28da7b0-aa11-4561...	Iris-versicolor	3.3	1.0	4.9	2.4	https://archive.ics.uci.edu...
<input type="checkbox"/>	b1d43c29-1b59-4c6a...	Iris-setosa	1.3	0.2	4.4	3.2	https://archive.ics.uci.edu...
<input type="checkbox"/>	bc33d553-cb92-4ba4...	Iris-versicolor	3.3	1.0	5.0	2.3	https://archive.ics.uci.edu...
<input type="checkbox"/>	b5914302-907e-404...	Iris-setosa	1.7	0.3	5.7	3.8	https://archive.ics.uci.edu...
<input type="checkbox"/>	a4a33bdb-af8f-467c...	Iris-versicolor	3.3	1.0	5.0	2.3	https://archive.ics.uci.edu...
<input type="checkbox"/>	9e816148-e4f2-46af...	Iris-setosa	1.6	0.2	5.1	3.8	https://archive.ics.uci.edu...
<input type="checkbox"/>	9e149a24-eaa6-483f...	Iris-versicolor	4.1	1.0	5.8	2.7	https://archive.ics.uci.edu...
<input type="checkbox"/>	0c-fad1a...chv4...dsh...	Iris-virginica	1.6	0.2	4.8	3.1	https://archive.ics.uci.edu...

Items returned (50)								<input type="button" value="C"/>	Actions	<input type="button" value="Create item"/>
	id (String)	class	petal length	petal width	sepal length	sepal width	source url			
<input type="checkbox"/>	c28da7b0-aa11-4561...	Iris-versicolor	3.3	1.0	4.9	2.4	https://archive.ics.uci.edu...			
<input type="checkbox"/>	a4a33bdb-af8f-467c...	Iris-versicolor	3.3	1.0	5.0	2.3	https://archive.ics.uci.edu...			
<input type="checkbox"/>	251de0a1-0a7a-42bb...	Iris-versicolor	4.0	1.0	6.0	2.2	https://archive.ics.uci.edu...			
<input type="checkbox"/>	5ff5181c-3097-47cd...	Iris-versicolor	3.7	1.0	5.5	2.4	https://archive.ics.uci.edu...			
<input type="checkbox"/>	9e149a24-eaa6-483f...	Iris-versicolor	4.1	1.0	5.8	2.7	https://archive.ics.uci.edu...			
<input type="checkbox"/>	e369a0fb-67bf-4b64...	Iris-versicolor	3.5	1.0	5.0	2.0	https://archive.ics.uci.edu...			
<input type="checkbox"/>	006042c3-48b7-496c...	Iris-versicolor	4.1	1.0	5.8	2.7	https://archive.ics.uci.edu...			
<input type="checkbox"/>	bc33d553-cb92-4ba4...	Iris-versicolor	3.3	1.0	5.0	2.3	https://archive.ics.uci.edu...			
<input type="checkbox"/>	3fd1c18d0-0ff2-4f01-a...	Iris-versicolor	4.0	1.0	6.0	2.2	https://archive.ics.uci.edu...			
<input type="checkbox"/>	7e571dc5-932b-4572...	Iris-versicolor	3.3	1.0	4.9	2.4	https://archive.ics.uci.edu...			
<input type="checkbox"/>	e0e37f92-9837-4979...	Iris-setosa	1.6	0.6	5.0	3.5	https://archive.ics.uci.edu...			
<input type="checkbox"/>	50834453-8040-46e...	Iris-setosa	1.7	0.5	5.1	3.3	https://archive.ics.uci.edu...			
<input type="checkbox"/>	f7beb361-345f-4aa7-...	Iris-setosa	1.5	0.4	5.4	3.4	https://archive.ics.uci.edu...			
<input type="checkbox"/>	2580773e-66b5-4c0d...	Iris-setosa	1.9	0.4	5.1	3.8	https://archive.ics.uci.edu...			
<input type="checkbox"/>	391c295a-7f93-47ce...	Iris-setosa	1.4	0.3	4.8	3.0	https://archive.ics.uci.edu...			
<input type="checkbox"/>	949e5add-18b9-4152...	Iris-setosa	1.4	0.3	5.1	3.5	https://archive.ics.uci.edu...			

Your database is filtered to irises with petal width less than or equal to 1.0.

4.2 Query Option

Now try Query.

Select Query. Copy the id (string) of one of the items such as "551484a4-be3b-46e1-8d55-2ec0b704c67c".

DynamoDB > Explore items > Book1_Iris

Tables (1)

Any tag key

Any tag value

Book1_Iris

Autopreview

Book1_Iris

Scan or query items

Scan Query

Select a table or index

Select attribute projection

id (Partition key)

Filters

Add a filter to get started.

This table has more items to retrieve. To retrieve the next page of items, choose Retrieve next page.

Click on Run.

The screenshot shows the AWS Lambda function configuration interface. At the top, there are 'Run' and 'Reset' buttons. Below them is a green success message: 'Completed. Read capacity units consumed: 0.5'. The main area is titled 'Items returned (1)' and contains a table with one row of data. The columns are: id (String), class, petal length, petal width, sepal length, sepal width, and source url. The single item returned is: id 551484a4-be3b-46e1..., class Iris-versicolor, petal length 4.0, petal width 1.3, sepal length 5.5, sepal width 2.3, and source url https://archive.ics.uci.edu... .

	id (String)	class	petal length	petal width	sepal length	sepal width	source url
	551484a4-be3b-46e1...	Iris-versicolor	4.0	1.3	5.5	2.3	https://archive.ics.uci.edu...

Your data base was queried to a single item.

Your AWS data base is now managed!

Finish