Pokédex

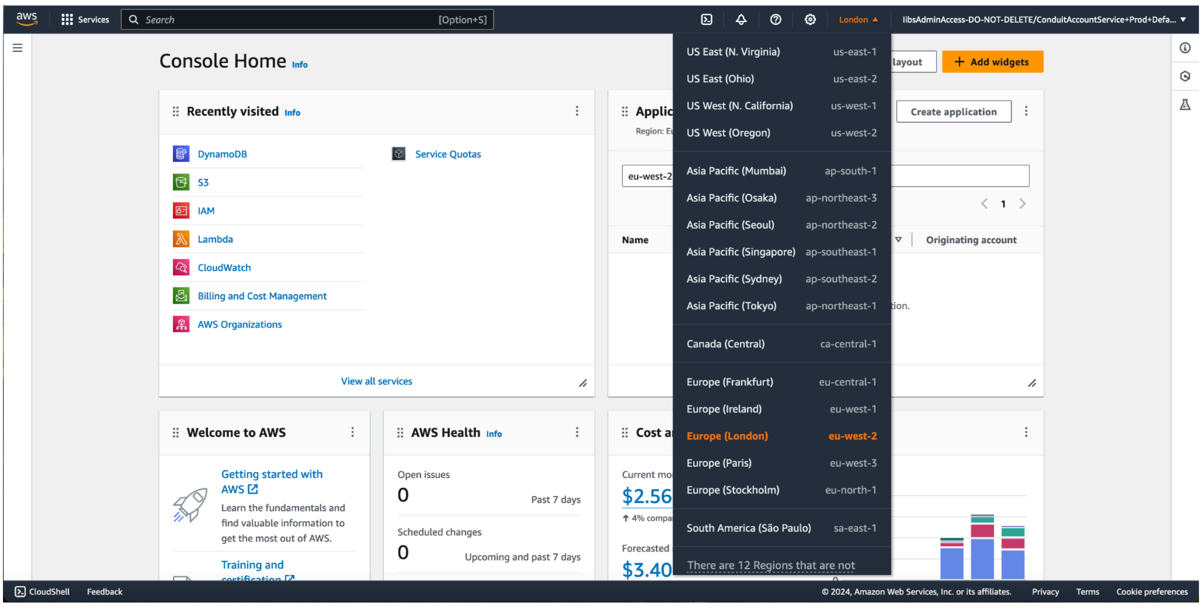
Pokemon Data Ingestion

This document explains the steps to ingest the Pokemon data that will be used by the Frontend and Backend workshops in the AWE-AUB context.

In this tutorial, you will learn how to:

Step 1 - Select London (eu-west-2) as your region

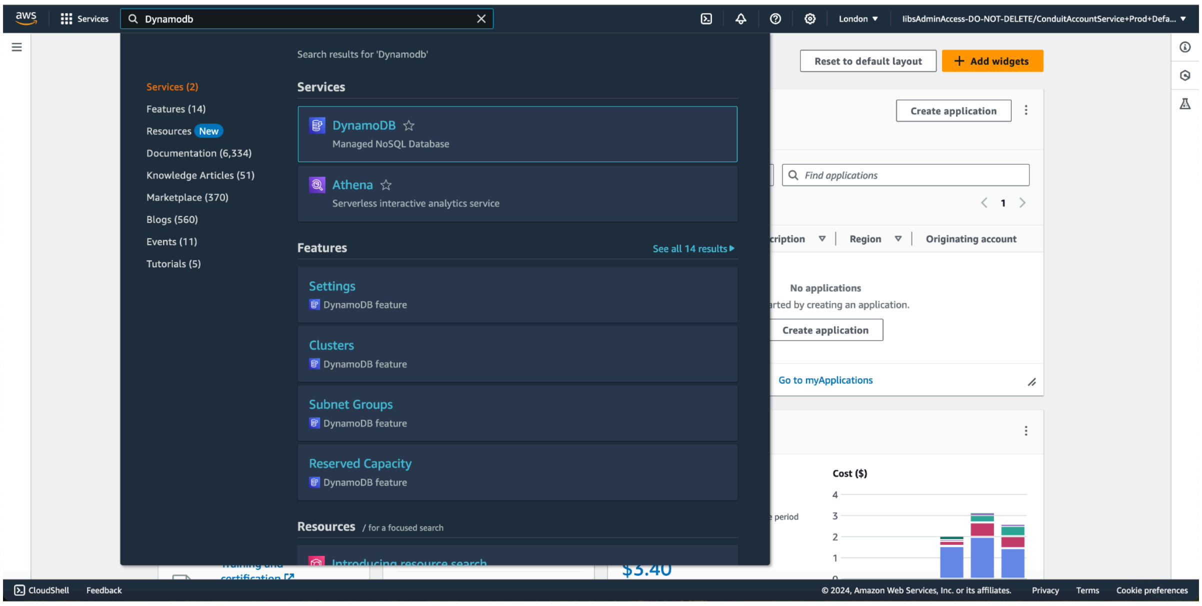
This tutorial expects that the resources created are in the London (eu-west-2) AWS region. Before starting any actual development, select this region on the AWS console.



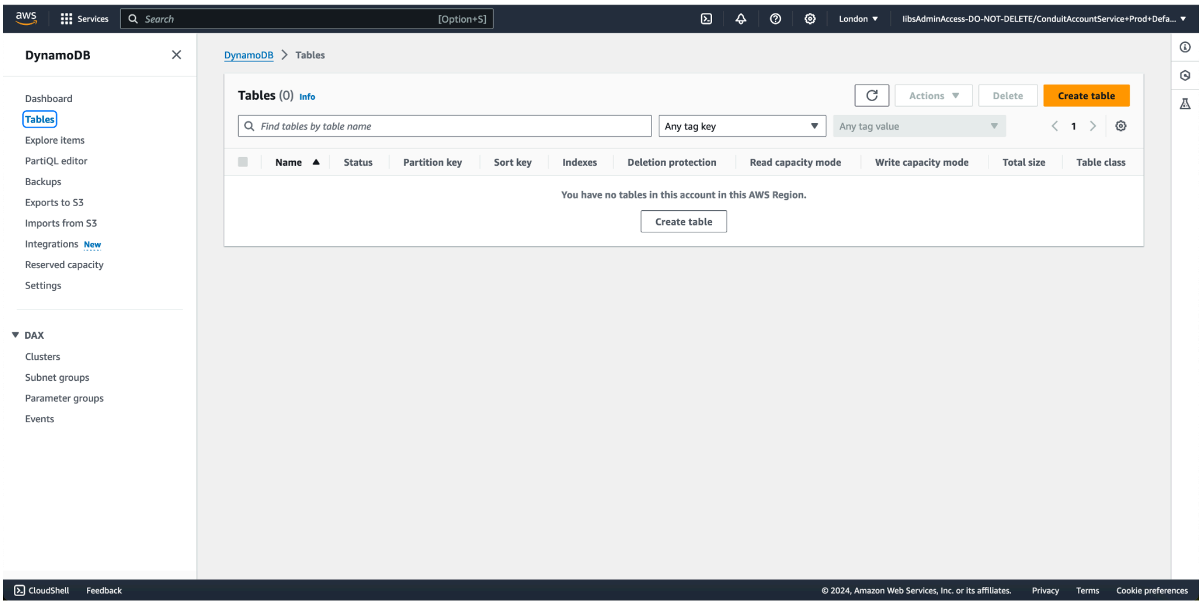
aws
Services
Search
[Option+S]
London ▲
IibsAdminAccess-DO-NOT-DELETE/ConduitAccountService+Prod+Defa... ▼
Console Home Info
layout
+ Add widgets
Recently visited Info
DynamoDB
Service Quotas
S3
IAM
Lambda
CloudWatch
Billing and Cost Management
AWS Organizations
View all services
Applic
Region: E
Create application
eu-west-2
Name
Originating account
1
tion.
Welcome to AWS
Getting started with AWS
Learn the fundamentals and find valuable information to get the most out of AWS.
Training and certification
AWS Health Info
Open issues
0
Past 7 days
Scheduled changes
0
Upcoming and past 7 days
Cost a
Current mo
$2.56
↑ 4% compa
Forecasted
$3.40
US East (N. Virginia) us-east-1
US East (Ohio) us-east-2
US West (N. California) us-west-1
US West (Oregon) us-west-2
Asia Pacific (Mumbai) ap-south-1
Asia Pacific (Osaka) ap-northeast-3
Asia Pacific (Seoul) ap-northeast-2
Asia Pacific (Singapore) ap-southeast-1
Asia Pacific (Sydney) ap-southeast-2
Asia Pacific (Tokyo) ap-northeast-1
Canada (Central) ca-central-1
Europe (Frankfurt) eu-central-1
Europe (Ireland) eu-west-1
Europe (London) eu-west-2
Europe (Paris) eu-west-3
Europe (Stockholm) eu-north-1
South America (São Paulo) sa-east-1
There are 12 Regions that are not
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Step 2 - Create a DynamoDB table

Amazon DynamoDB is **a serverless, NoSQL database service that enables you to develop modern applications at any scale**. As a serverless database, you only pay for what you use and DynamoDB has no extra latency for read requests when these requests are made very sporadically, no version upgrades, no maintenance windows, no need to perform software updates, and no downtime maintenance. The reason why we chose DynamoDB is because the records you add to this table can have different formats, which will be very helpful in the early phases of your project as the data model can change very quickly. Imagine that to launch a new feature to your project you need to start storing a new field in your records, with DynamoDB it’s very convenient to manipulate new records or to patch the old ones with new fields without having to perform migrations from one table to another. To start, on the AWS console, search for DynamoDB:



On the left side, select Tables. Then Create table on the right.



□	Name ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	Table class

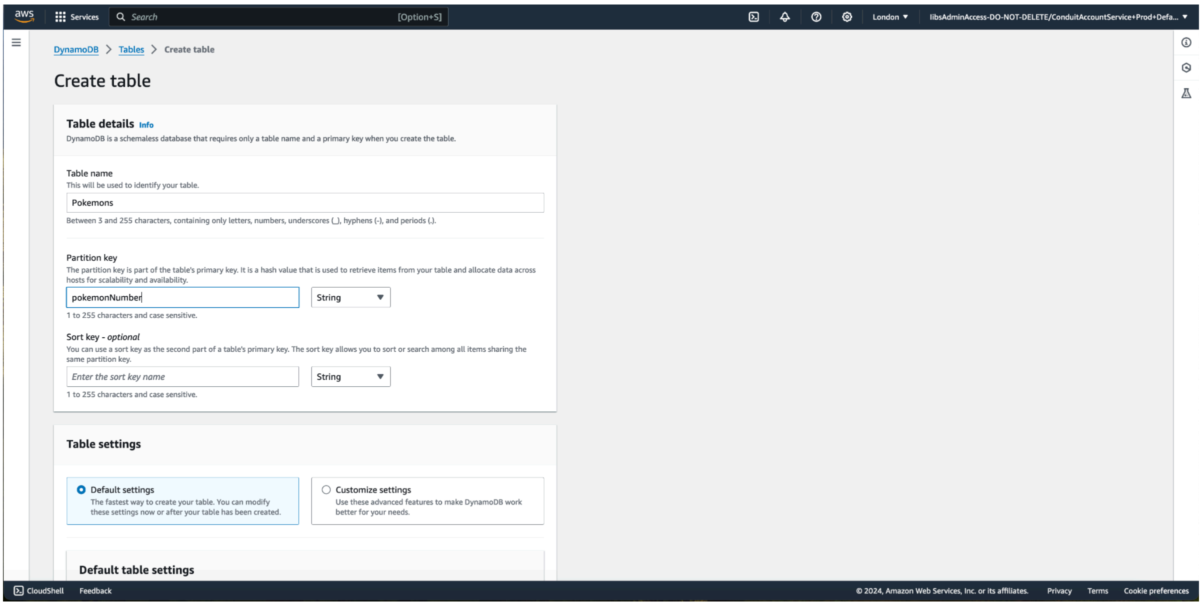
You have no tables in this account in this AWS Region.

Create table

Make sure to create a table with the following inputs, **future steps of this series of workshops have a hard dependency on these names**:

- Table name: Pokemons
- Partition key: pokemonNumber and the type should be: Number
- Everything else can be as is by default

Click create table at the bottom of the page and wait for a couple of seconds for your table to be created:


aws
Services
Search
[Option+S]
London ▾
iibsAdminAccess-DO-NOT-DELETE/ConduitAccountService+Prod+Defa... ▾
DynamoDB > Tables > Create table
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

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.

Pokemons

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.

pokemonNumber | 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.

Enter the sort key name | 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.

Default table settings

The table creation will take a while and you should see a spinning widget.

Creating the Pokemons table. It will be available for use shortly.

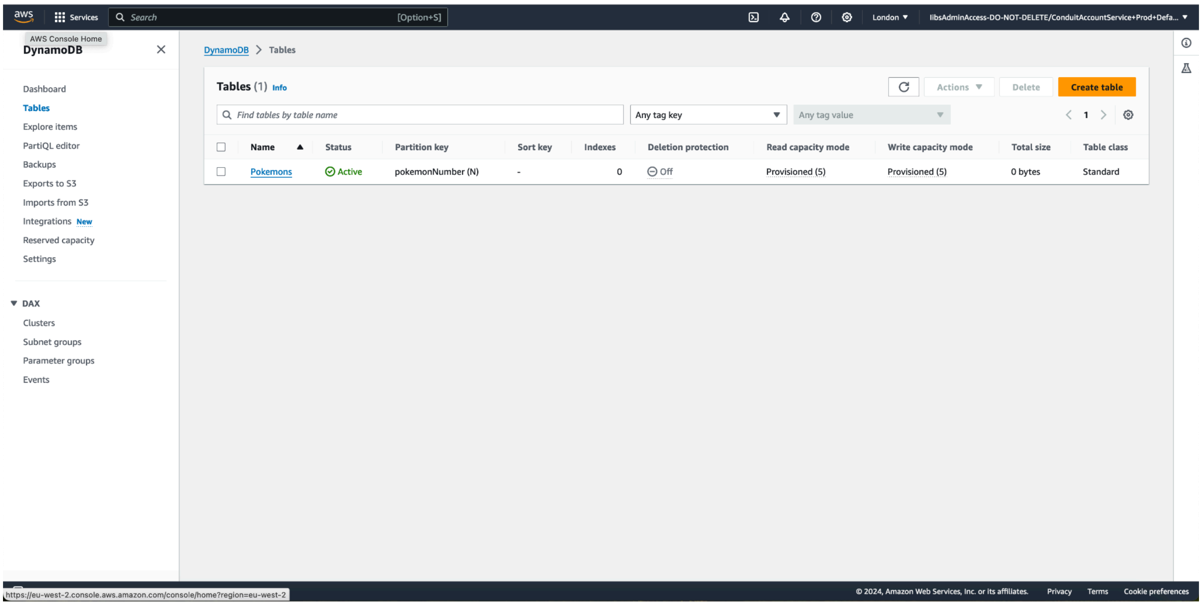
After a couple of seconds, you should see a successfully created table widget.

The Pokemons table was created successfully.

That table will contain all Pokemons that will later be used through the rest of this Pokemon workshop.

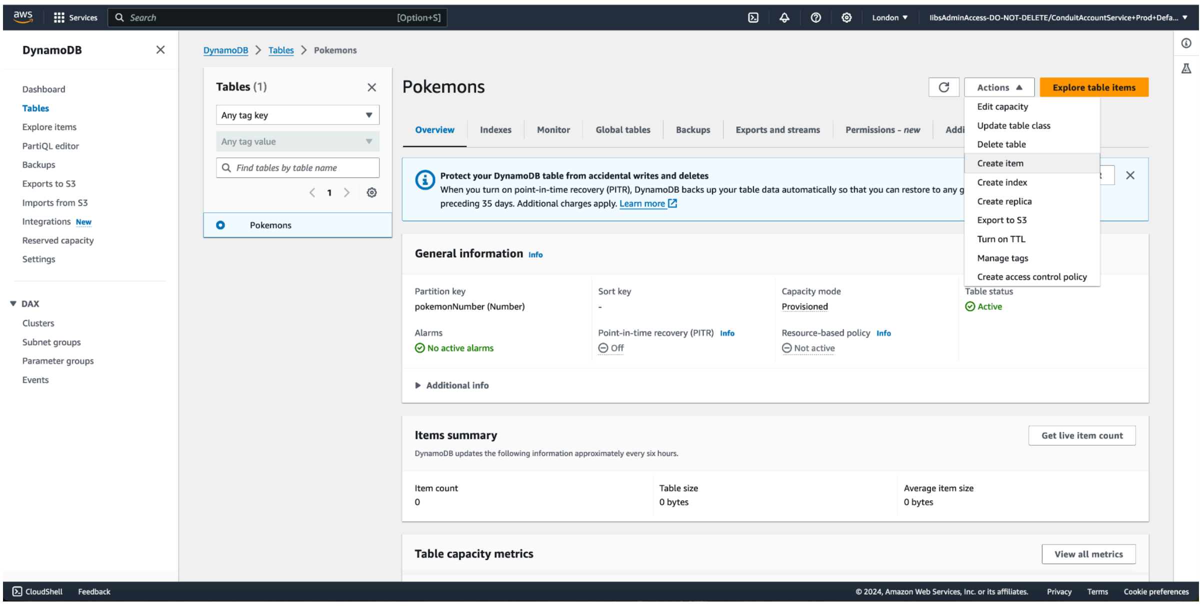
Step 3 - Ingest one Pokemon manually to a DynamoDB table

Open your table by clicking its name (you might need to refresh your screen to reload the dashboard with the created tables).



□	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	Table class
□	Pokemons	Active	pokemonNumber (N)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes	Standard

On the actions dropdown, select Create item:



Pokemons

Protect your DynamoDB table from accidental writes and deletes
When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any g... preceding 35 days. Additional charges apply. Learn more

General information Info

Partition key: pokemonNumber (Number)
Sort key: -
Capacity mode: Provisioned
Table status: Active
Alarms: No active alarms
Point-in-time recovery (PITR) Info: Off
Resource-based policy Info: Not active

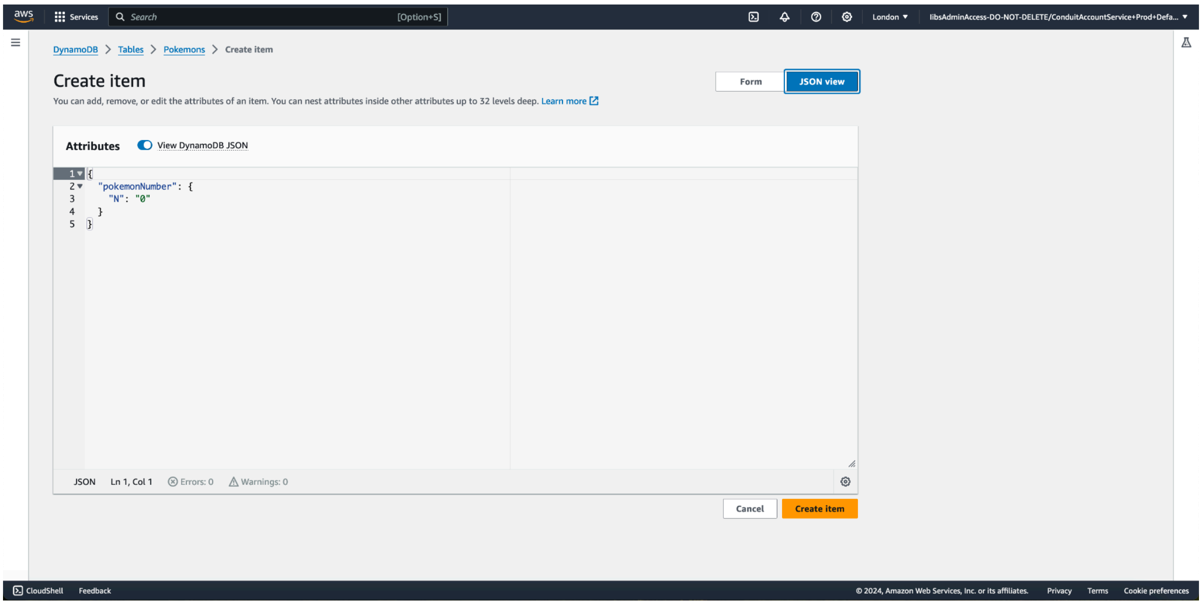
Additional info

Items summary
DynamoDB updates the following information approximately every six hours.

Item count: 0
Table size: 0 bytes
Average item size: 0 bytes

Table capacity metrics

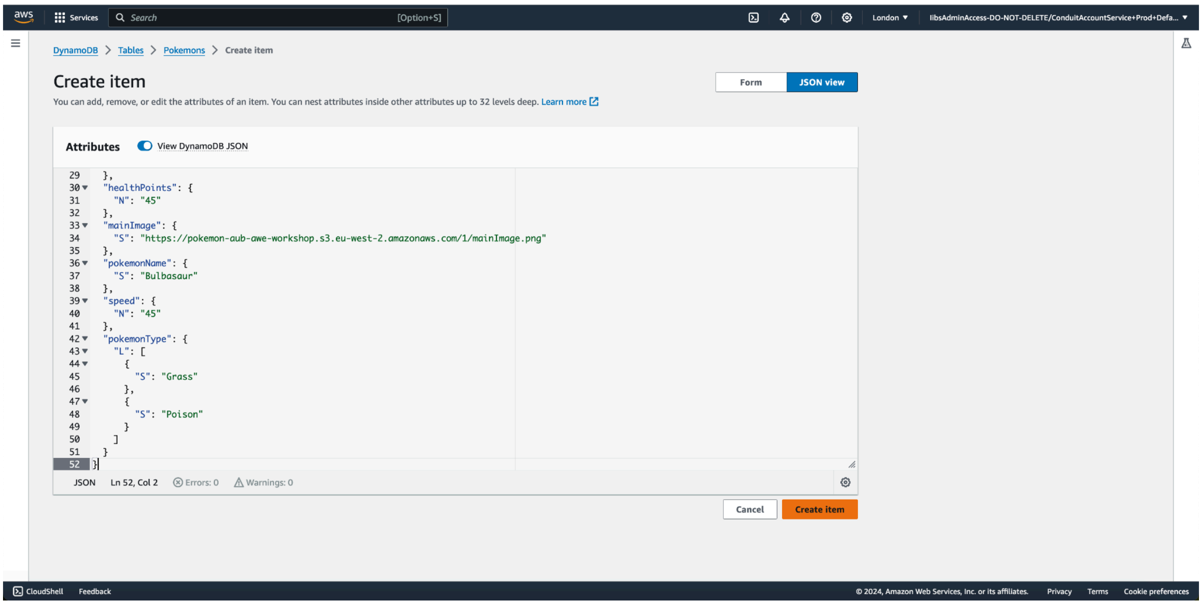
Select the JSON view on the top right and paste the following content to manually ingest a Bulbassaur to your database:



Create item
You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

```
{
  "pokemonNumber": {
    "N": "0"
  }
}
```

```json
{
  "pokemonNumber": {
    "N": "1"
  },
  "attack": {
    "N": "49"
  },
  "defense": {
    "N": "49"
  },
  "devolution": {
    "S": ""
  },
  "evolution": {
    "S": "Ivysaur"
  },
  "evolutionFamily": {
    "L": [
      {
        "S": "Bulbasaur"
      },
      {
        "S": "Ivysaur"
      },
      {
        "S": "Venusaur"
      }
    ]
  },
  "healthPoints": {
    "N": "45"
  },
  "mainImage": {
    "S":
"https://pokemon-aub-awe-workshop.s3.eu-west-2.amazonaws.com/1/mainImage
.png"
  },
  "pokemonName": {
    "S": "Bulbasaur"
  },
  "speed": {
    "N": "45"
  },
  "pokemonType": {
    "L": [
      {
        "S": "Grass"
      },
      {
        "S": "Poison"
      }
    ]
  }
}
```



Create item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

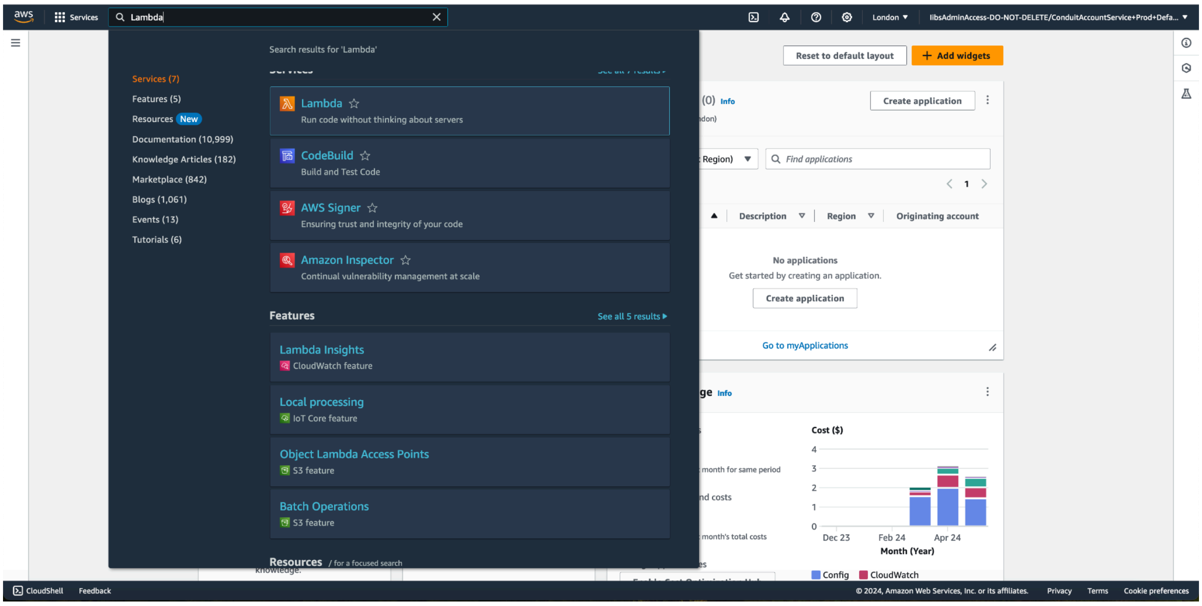
```
  },
  "healthPoints": {
    "N": "45"
  },
  "mainImage": {
    "S": "https://pokemon-aub-awe-workshop.s3.eu-west-2.amazonaws.com/1/mainImage.png"
  },
  "pokemonName": {
    "S": "Bulbasaur"
  },
  "speed": {
    "N": "45"
  },
  "pokemonType": {
    "L": [
      {
        "S": "Grass"
      },
      {
        "S": "Poison"
      }
    ]
  }
}
```

After pasting the content, click the Create item button. To verify that one item is added to the table click on Explore table items on the top right. Congratulations, you have added a Bulbassaur to your table.

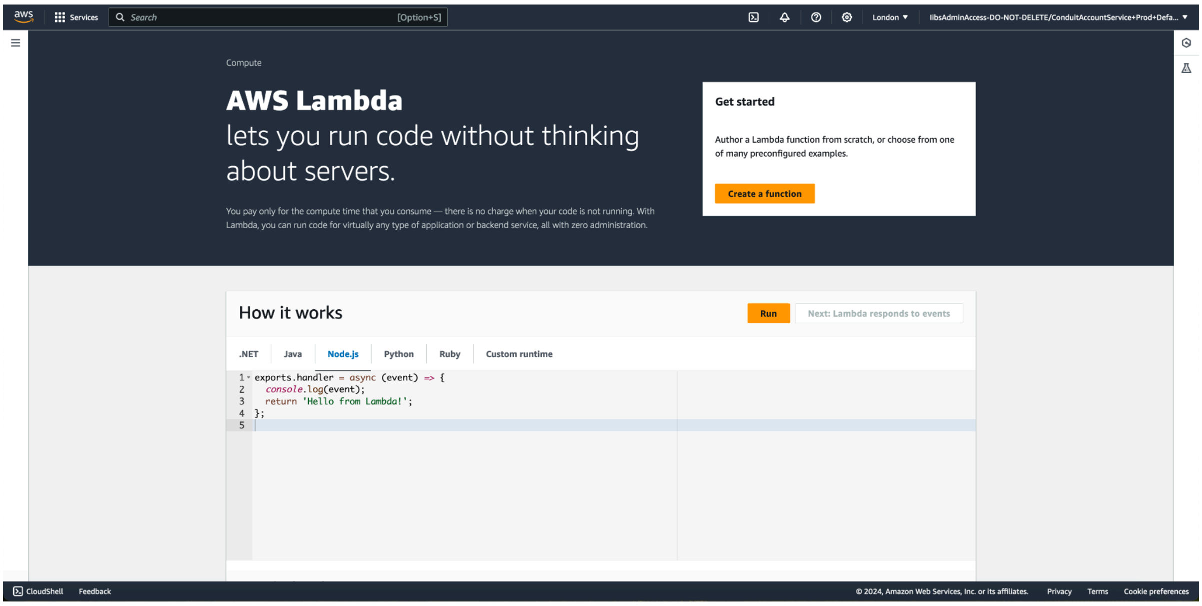
Step 4 - Create a DynamoDB data ingestion lambda

AWS Lambda is **a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers**. Now that we have created a table and tested the manual ingestion process to get familiar with the AWS ecosystem and the DynamoDB table we created, let's create an AWS Lambda that will ingest the remaining 150 Pokemons to the DynamoDB table for us. Notice that we could have inserted manually the remaining 150 Pokemons, but that would be a tedious task, but also notice that if we had to insert 5 records to our table, creating a lambda to perform this task would take more time than just inserting the data manually ourselves. Always balance the time to automate a task vs doing it manually, and account for the number of times you will need to do it manually again in the future.

To get started with your lambda creation, on the AWS console, search for Lambda:



Select Create function on the top right:



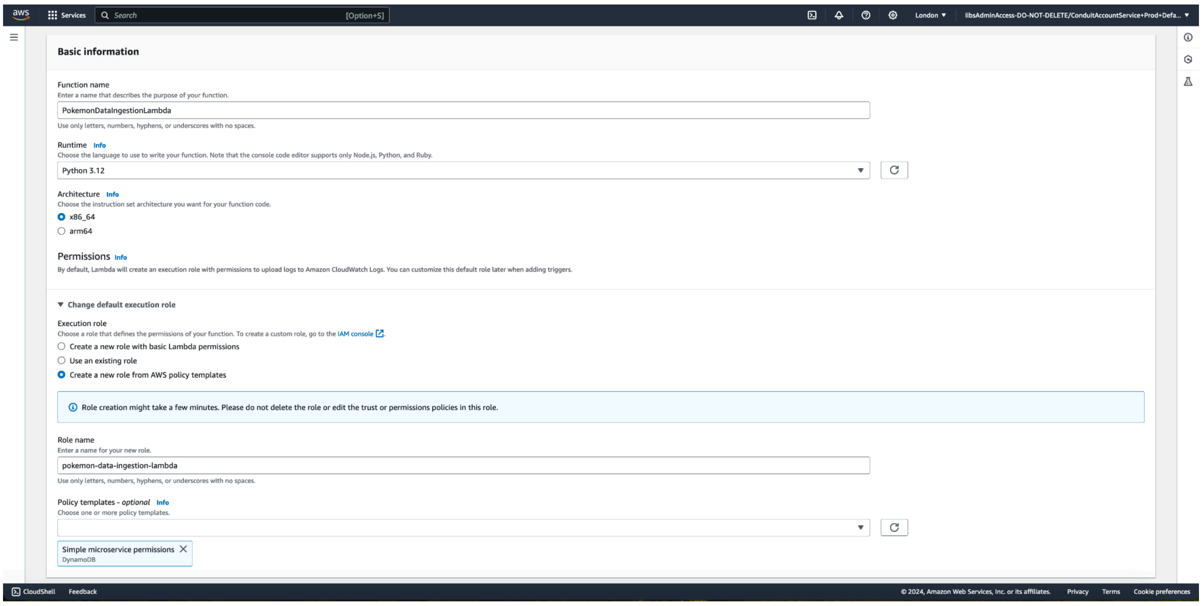
AWS Lambda lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

```
exports.handler = async (event) => {
  console.log(event);
  return 'Hello from Lambda!';
};
```

Select Author from scratch and make sure that the Runtime is Python 3.12 and the Architecture is x86_64. For the name, use something like PokemonDataIngestionLambda.



aws
Services
Search
[Option+S]
London
libsAdminAccess-DO-NOT-DELETE/ConduitAccountService+Prod+Defa...

Basic information

Function name
Enter a name that describes the purpose of your function.
PokemonDataIngestionLambda
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 3.12

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

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name
Enter a name for your new role.
pokemon-data-ingestion-lambda
Use only letters, numbers, hyphens, or underscores with no spaces.

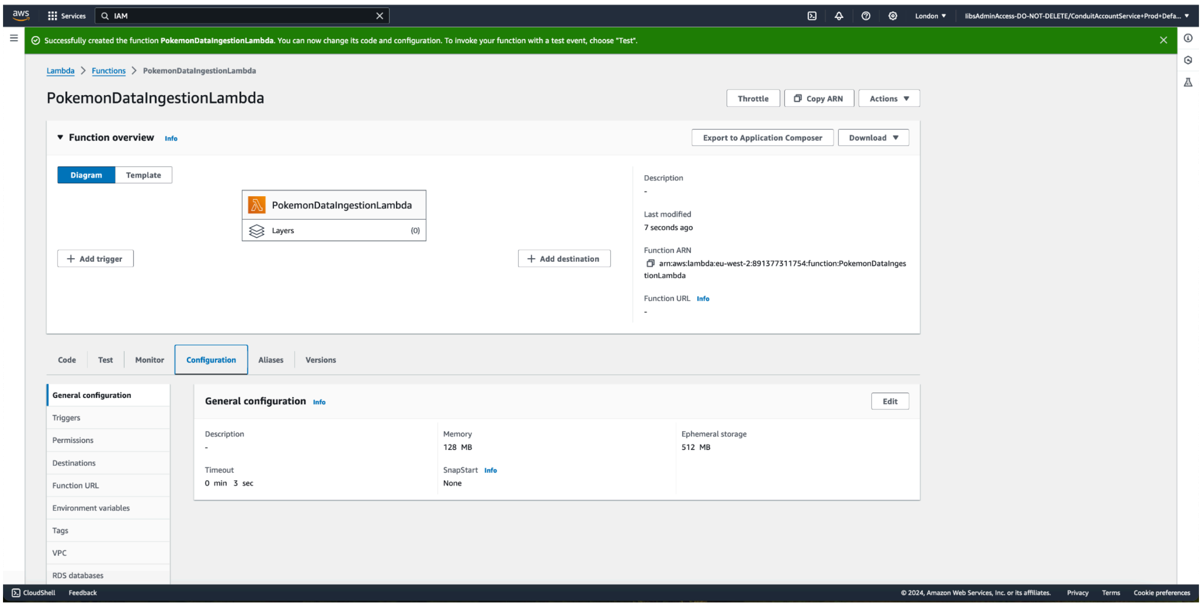
Policy templates - *optional* Info
Choose one or more policy templates.

Simple microservice permissions ✕
DynamoDB

Expand the Change default execution role dropdown and select:

- Create a new role from AWS policy templates
- For role name, use pokemon-data-ingestion-lambda
- And select Simple microservice permissions and Amazon s3 object read-only permissions as policy template, notice that these are DynamoDB permissions.

Then proceed with Create function. After your function is created, select the Configuration tab.



aws
Services
IAM
London
libsAdminAccess-DO-NOT-DELETE/ConduitAccountService+Prod+Defa...

Successfully created the function **PokemonDataIngestionLambda**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > PokemonDataIngestionLambda

PokemonDataIngestionLambda

Description
-

Last modified
7 seconds ago

Function ARN
arn:aws:lambda:eu-west-2:891377311754:function:PokemonDataIngestionLambda

Function URL Info
-

General configuration Info

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

By default, the timeout of lambdas is configured to 3 seconds. However, to ingest the pokemons to the database, it takes approximately 10 seconds. Click Edit and change the timeout to 1 minute and 0 seconds and then click Save.

Step 5 - Invoke the data ingestion lambda

From reading the python code below, you will notice that it's reading from an S3 bucket a csv file. **Amazon Simple Storage Service (Amazon S3) is an object storage service** that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

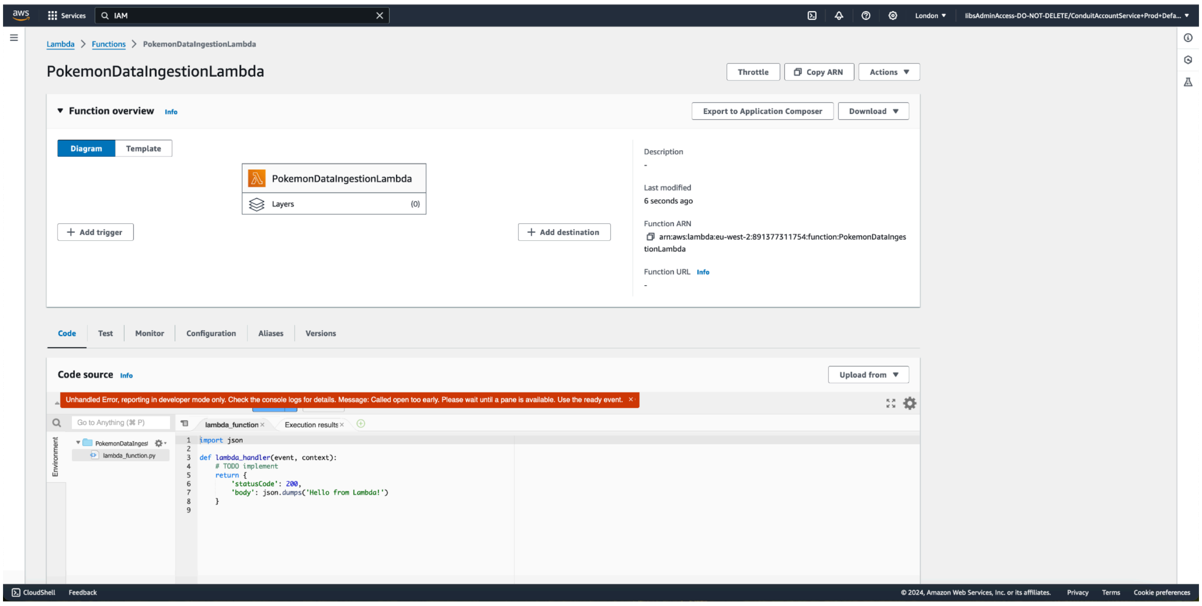
Below are the two first lines of the .csv file we are reading. It's a simple csv file containing the Pokemon information we need to build our pokédex. Notice that we are converting the rows from this csv file to records in the DynamoDB table, we could have designed our application in a way where we wouldn't need the DynamoDB table and we could make reads to this csv file. Going with the DynamoDB allows us to have more flexibility on the data model and allows us for faster reads. Imagine that you want to read information from a pokemonNumber of 3, using DynamoDB that's an O(1) operation, as the pokemonNumber is a key of that table. If we were using the csv file directly, that would be an O(n) operation as to get the pokemon with pokemonNumber of 3 we would potentially need to read the whole csv file. In that case, the csv file is sorted by pokemonNumber so you could try to guess where it is, but that assumption of the csv file being sorted could break at any time.

```
pokemonNumber,pokemonName,pokemonType,healthPoints,attack,defense,speed,evolutio
n,devolution,evolutionFamily,mainImage
1,Bulbasaur,"['Grass', 'Poison']",45,49,49,45,Ivysaur,,"['Bulbasaur', 'Ivysaur',
'Venusaur']",https://pokemon-aub-awe-workshop.s3.eu-west-2.amazonaws.com/1/mainI
mage.png
2,Ivysaur,"['Grass', 'Poison']",60,62,63,60,Venusaur,Bulbasaur,"['Bulbasaur',
'Ivysaur',
'Venusaur']",https://pokemon-aub-awe-workshop.s3.eu-west-2.amazonaws.com/2/mainI
mage.png
```

On the lambda you have created, move to the Code tab, and paste the following code in the lambda_function tab of the code editor:

```
import boto3
import ast
import io
import csv

region = 'eu-west-2'
# Initialize dynamodb resource
dynamodb = boto3.resource('dynamodb',  region_name=region)
# Initialize the S3 client
s3 = boto3.client('s3')
```

```
# Specify the S3 bucket and object key of the CSV file
bucket_name = 'pokemon-aub-awe-workshop'
file_key = 'final_pokemon.csv'

# DynamoDB table name
dynamodb_table = 'Pokemons'


def read_csv_from_s3_bucket(bucket_name: str, file_key: str):
    """
    Read a CSV file from S3 bucket

    Arguments:
    - bucket_name: name of the S3 bucket to read files from
    - file_key: file path where to read from
    """
    response = s3.get_object(Bucket=bucket_name, Key=file_key)
    csv_content = response['Body'].read().decode('utf-8')

    return csv_content


def insert_dynamoDB_items(tablename: str, items: list):
    """
    Insert items to a dynamodb table.

    Arguments:
    - tablename: name of the dynamodb table where items will be inserted
    - items: list of elements that will be added to the dynamodb table
    """
    ddb = dynamodb.Table(tablename)

    for record in items:
        ddb.put_item(Item=record)


def apply_literal_eval(items: list):
    """
    Iterate over a list of elements, where these elements have fields that are a
    string representation of a list, such as 'pokemonType' and 'evolutionFamily'.
And
    that are a string representation of a number such as 'pokemonNumber'.

    '["element_1", "element_2"]' -> ["element_1", "element_2"]
    '1' -> 1

    Arguments:
    - items: list of elements to apply literal_eval on
    """
    for record in items:
        for key in ["pokemonNumber", "attack", "defense", "evolutionFamily",
        "healthPoints", "speed", "pokemonType"]:
            record[key] = ast.literal_eval(record[key])


def lambda_handler(event, context):
    # Read pokemon dataset
    pokemons_csv_content = read_csv_from_s3_bucket(bucket_name, file_key)
    csv_reader = csv.DictReader(io.StringIO(pokemons_csv_content))
    pokemons = []
    for pokemon in csv_reader:
        pokemons.append(pokemon)

    apply_literal_eval(pokemons)

    # Insert pokemons to DynamoDB, pokemons[1:] because Bulbassaur was manually
inserted
    insert_dynamoDB_items(dynamodb_table, pokemons[1:])
```


aws
Services
IAM
London ▼
libsAdminAccess-DO-NOT-DELETE/ConduitAccountService+Prod+Defa... ▼
Lambda > Functions > PokemonDataIngestionLambda

PokemonDataIngestionLambda

Description
-

Last modified
6 seconds ago

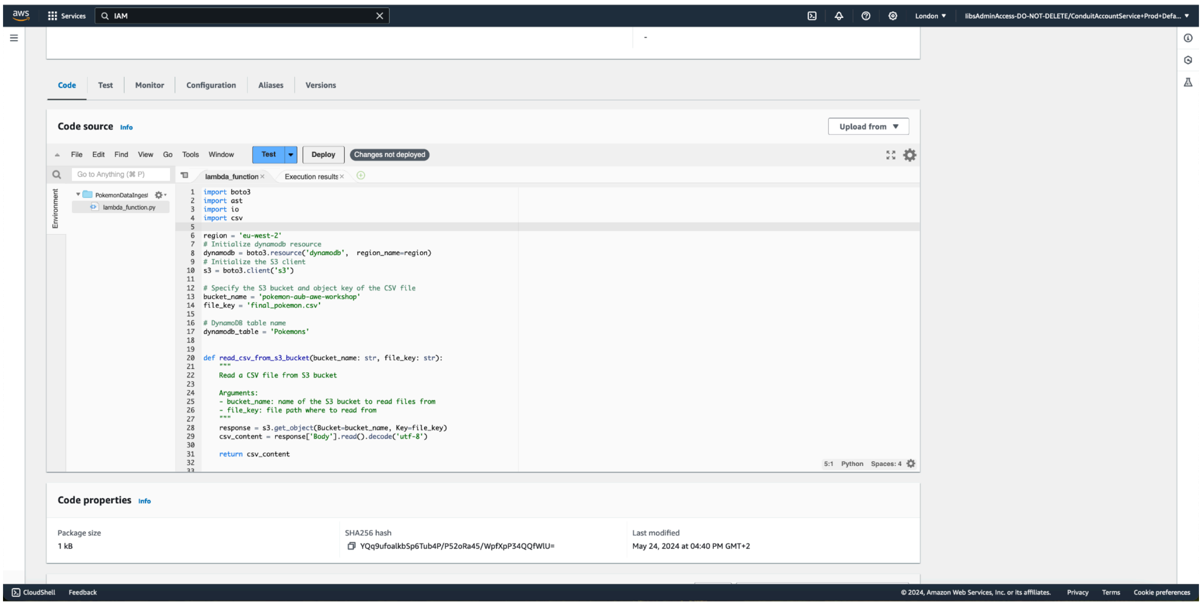
Function ARN
arn:aws:lambda:eu-west-2:891377311754:function:PokemonDataIngestionLambda

Function URL Info
-

Unhandled Error, reporting in developer mode only. Check the console logs for details. Message: Called open too early. Please wait until a pane is available. Use the ready event. ×

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

Make sure to Deploy your changes, and then Test them.



```python
import boto3
import ast
import io
import csv

region = 'eu-west-2'
# Initialize dynamodb resource
dynamodb = boto3.resource('dynamodb',  region_name=region)
# Initialize the S3 client
s3 = boto3.client('s3')

# Specify the S3 bucket and object key of the CSV file
bucket_name = 'pokemon-aub-awe-workshop'
file_key = 'final_pokemon.csv'

# DynamoDB table name
dynamodb_table = 'Pokemons'


def read_csv_from_s3_bucket(bucket_name: str, file_key: str):
    """
    Read a CSV file from S3 bucket

    Arguments:
    - bucket_name: name of the S3 bucket to read files from
    - file_key: file path where to read from
    """
    response = s3.get_object(Bucket=bucket_name, Key=file_key)
    csv_content = response['Body'].read().decode('utf-8')

    return csv_content

```

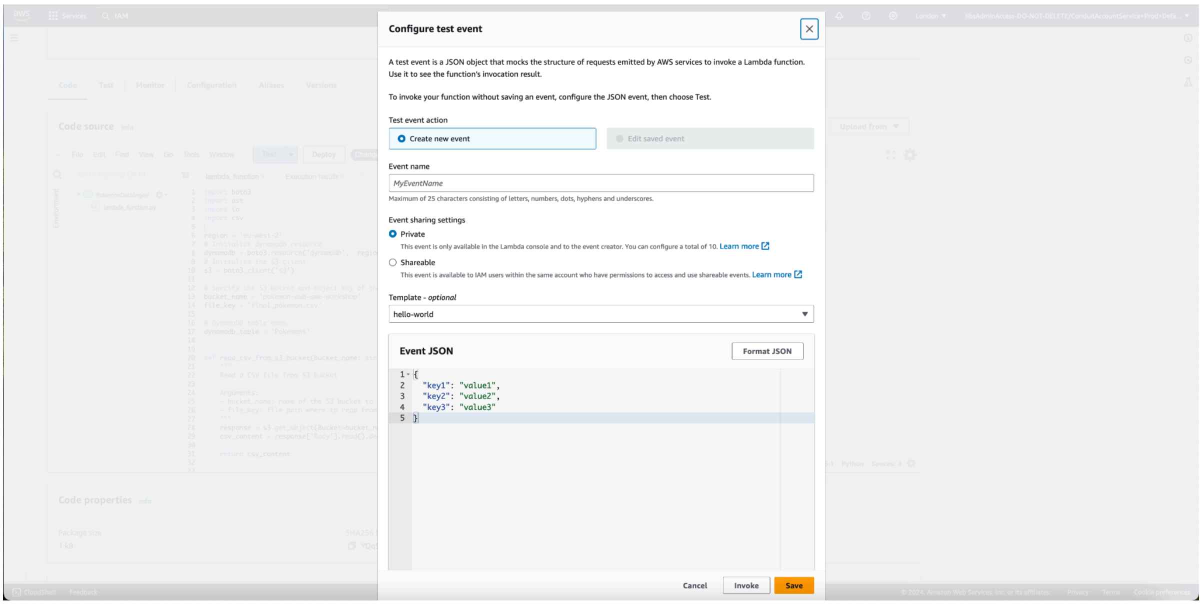
Code properties Info

Package size
1 kB

SHA256 hash
YQq9ufoalkbSp6Tub4P/P52oRa45/WpfXpP34QQfWlU=

Last modified
May 24, 2024 at 04:40 PM GMT+2

On the Configure test event page, click Invoke:



Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

● Create new event ○ Edit saved event

Event name

MyEventName

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

● Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

○ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

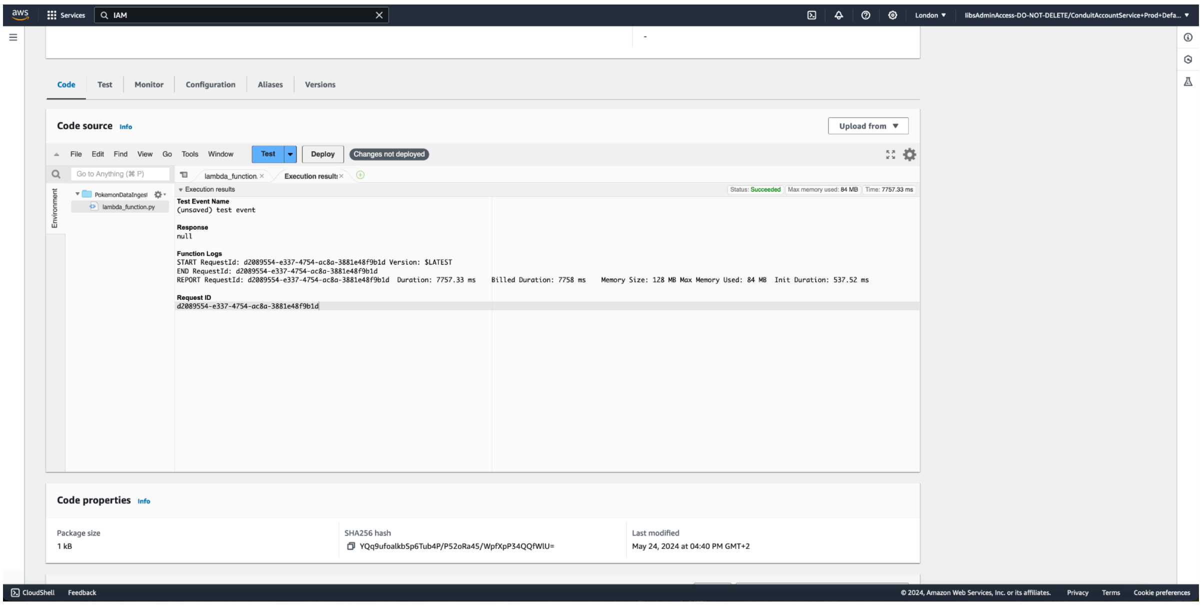
Template - *optional*

hello-world

Event JSON

```json
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

On the Execution results tab you should see something similar to the following, but notice that the requestId will change every time you execute your lambda.



Execution results — Status: Succeeded | Max memory used: 84 MB | Time: 7757.33 ms

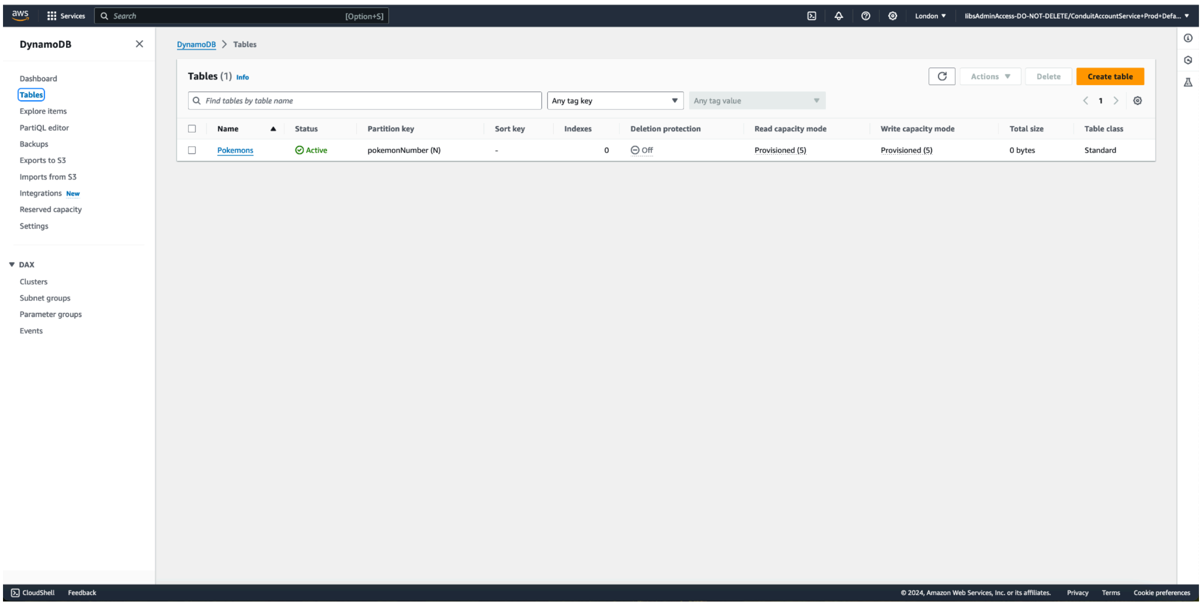
```
Test Event Name
(unsaved) test event

Response
null

Function Logs
START RequestId: d2089554-e337-4754-ac8a-3881e48f9b1d Version: $LATEST
END RequestId: d2089554-e337-4754-ac8a-3881e48f9b1d
REPORT RequestId: d2089554-e337-4754-ac8a-3881e48f9b1d  Duration: 7757.33 ms    Billed Duration: 7758 ms    Memory Size: 128 MB Max Memory Used: 84 MB  Init Duration: 537.52 ms

Request ID
d2089554-e337-4754-ac8a-3881e48f9b1d
```

Code properties

Package size	SHA256 hash	Last modified
1 kB	YQq9ufoalkbSp6Tub4P/P52oRa45/WpfXpP34QQfWlU=	May 24, 2024 at 04:40 PM GMT+2

Step 6 - Verify that your Pokemons DynamoDB was correctly populated

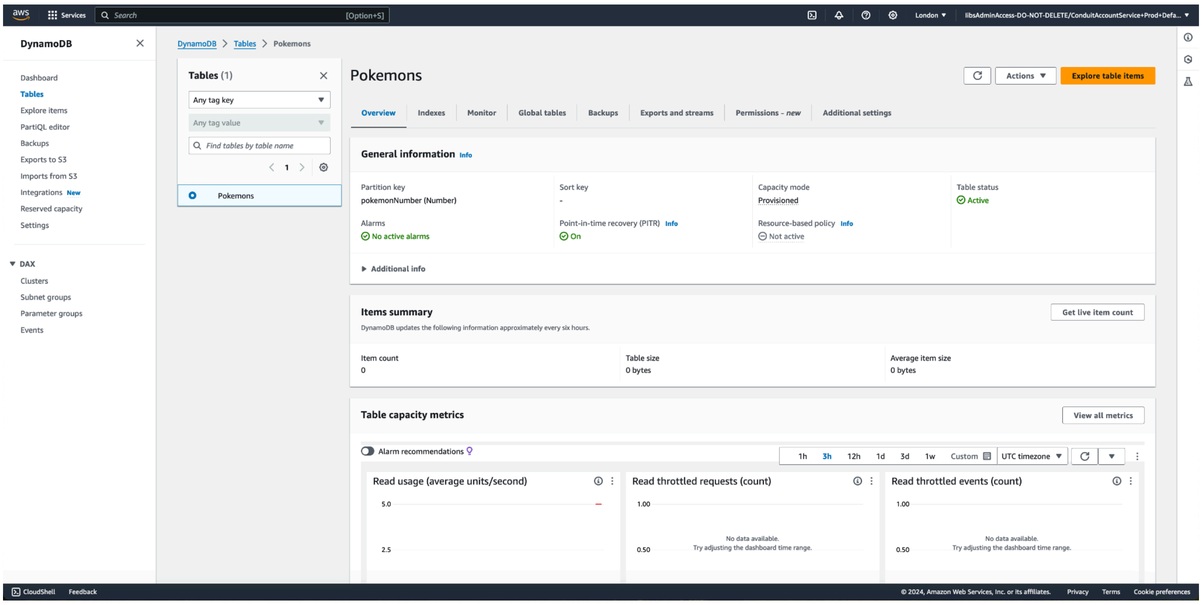
Now, if you go back to your DynamoDB table, you should see that there are 151 pokemons added. Search for DynamoDB again, and then open your Pokemons table:



Tables (1)

	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	Table class
□	Pokemons	Active	pokemonNumber (N)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes	Standard

And select Explore Table Items:



Pokemons

General information

Partition key: pokemonNumber (Number)
Sort key: -
Capacity mode: Provisioned
Table status: Active
Alarms: No active alarms
Point-in-time recovery (PITR): On
Resource-based policy: Not active

Additional info

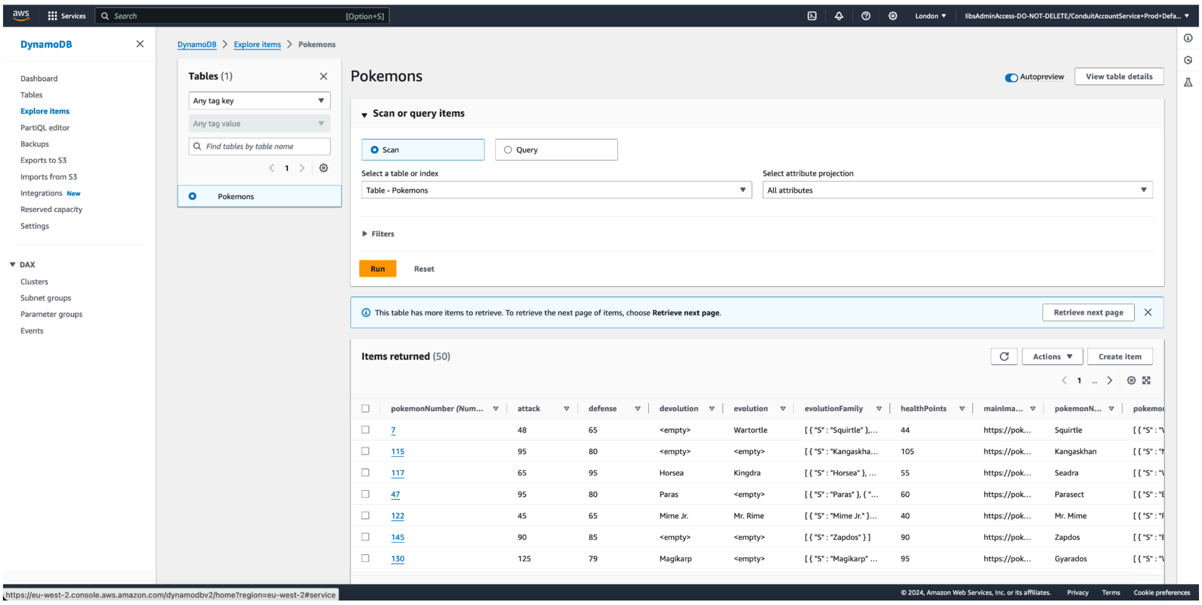
Items summary

DynamoDB updates the following information approximately every six hours.

Item count: 0
Table size: 0 bytes
Average item size: 0 bytes

Table capacity metrics

By default, DynamoDB loads the first 50 items from your table, but you can retrieve more using the Retrieve Next Page button:



Pokemons

Scan or query items

● Scan ○ Query

Select a table or index: Table - Pokemons

Select attribute projection: All attributes

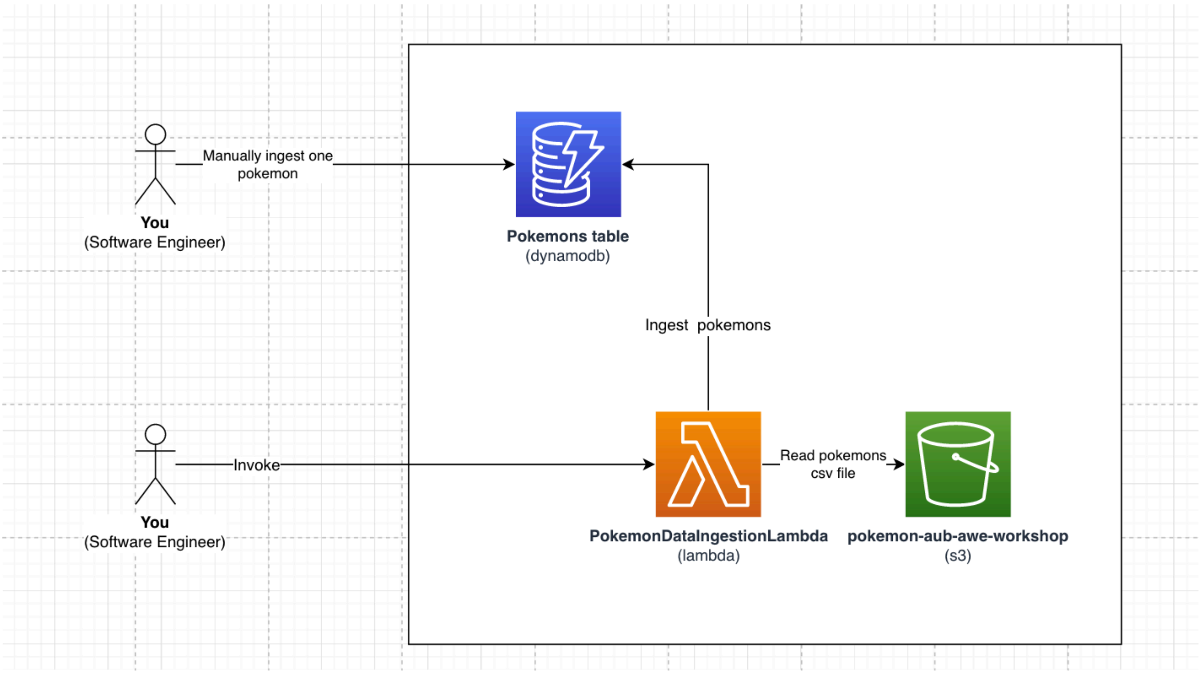
This table has more items to retrieve. To retrieve the next page of items, choose **Retrieve next page**.

Items returned (50)

	pokemonNumber (Num...	attack	defense	devolution	evolution	evolutionFamily	healthPoints	mainIma...	pokemonN...	pokemo
□	7	48	65	<empty>	Wartortle	[{ "S" : "Squirtle" },...	44	https://pok...	Squirtle	[{ "S" : "V
□	115	95	80	<empty>	<empty>	[{ "S" : "Kangaskha...	105	https://pok...	Kangaskhan	[{ "S" : "N
□	117	65	95	Horsea	Kingdra	[{ "S" : "Horsea" }, ...	55	https://pok...	Seadra	[{ "S" : "V
□	47	95	80	Paras	<empty>	[{ "S" : "Paras" }, { "...	60	https://pok...	Parasect	[{ "S" : "E
□	122	45	65	Mime Jr.	Mr. Rime	[{ "S" : "Mime Jr." }...	40	https://pok...	Mr. Mime	[{ "S" : "F
□	145	90	85	<empty>	<empty>	[{ "S" : "Zapdos" }]	90	https://pok...	Zapdos	[{ "S" : "E
□	130	125	79	Magikarp	<empty>	[{ "S" : "Magikarp" ...	95	https://pok...	Gyarados	[{ "S" : "V

Congratulations 🥳, you have successfully ingested data for all 151 pokemons from the 1st generation of Pokemons to a DynamoDB table. Let's do a quick recap of what we have achieved already:

- Created a DynamoDB table named Pokemons
- Ingested one pokemon manually using the AWS console to the Pokemons table
- Created a Lambda that ingests pokemons from data from a csv file stored in S3 to the DynamoDB table
- Invoked the Lambda



This DynamoDB table will be the data store used for the application we are building. In the next parts of this workshop, we will be creating an API that will make read requests to this data store; and also create a frontend that will call this API to display pokemons on a website.