Introducción a DynamoDB

¿Qué es DynamoDB?

Base de datos no relacional para aplicaciones que necesitan rendimiento a cualquier escala


NoSQL
Scalable
High Performance
Flexible

Base de Datos Relacional

ID	Nombre	Apellido	Correo	Teléfono
1	Carolina	Gómez	carolina9511@gmail.com	3147472706
2	Alejandro	Rendón	aerendon@gmail.com	3121231234
3	Angelica	Aguirre	anaguicas@gmail.com	3112345234

Base de Datos No Relacional

```
{
  "correo": "carolina9511@gmail.com",
  "nombre": "Carolina",
  "apellido": "Gómez",
  "telefono": "3147472706",
  "dirección": "Cr. 58 # 77-41"
},
{
  "correo": "aerendon@gmail.com",
  "nombre": "Alejandro",
  "apellido": "Rendón"
},
{
  "correo": "anaguicas@gmail.com",
  "nombre": "Angelica",
  "apellido": "Aguirre",
  "telefono": "3112345234"
}
```

Precios

Resumen de precios

ALMACENAMIENTO DE DATOS

DynamoDB cobra por GB de espacio de disco que consume su tabla. Los primeros 25 GB consumidos por mes son gratis; después, los precios comienzan a partir de 0,25 USD por GB-mes.

UNIDAD DE CAPACIDAD DE ESCRITURA

Una unidad de capacidad de escritura proporciona hasta una escritura por segundo, suficiente para lograr 2,6 millones de escrituras al mes. Las primeras 25 GB unidades de capacidad de escritura por mes son gratis; después, los precios comienzan a partir de 0,47 USD por unidad de capacidad de escritura-mes.

UNIDAD DE CAPACIDAD DE LECTURA

Una unidad de capacidad de lectura proporciona hasta dos lecturas por segundo, suficiente para lograr 5,2 millones de lecturas al mes. Las primeras 25 GB unidades de capacidad de lectura por mes son gratis; después, los precios comienzan a partir de 0,09 USD por unidad de capacidad de lectura-mes.

Capa gratuita

25 GB AL MES

de almacenamiento de datos (indizados)

200 MILLONES DE SOLICITUDES AL MES

a través de 25 unidades de capacidad de escritura y 25 unidades de capacidad de lectura

2,5 MILLONES DE SOLICITUDES DE STREAMING AL MES

de DynamoDB Streams

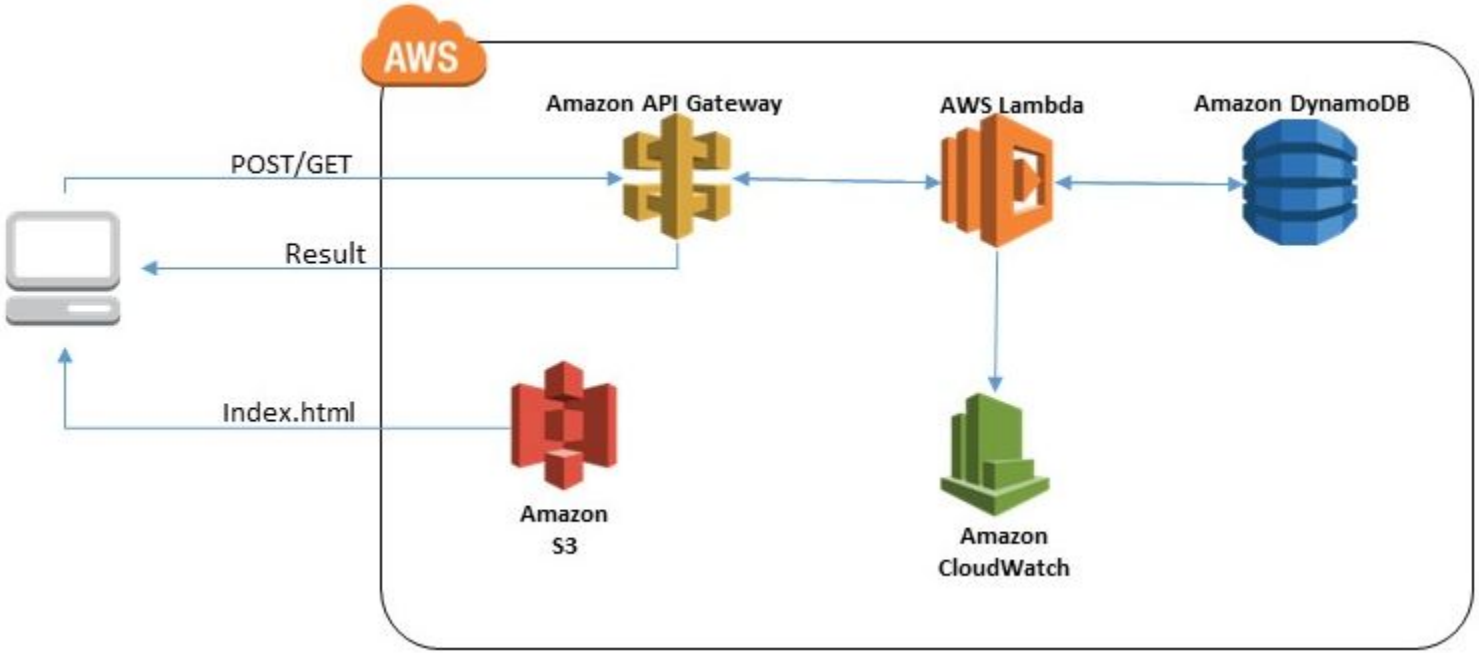
CAPACIDAD DE IMPLEMENTAR TABLAS GLOBALES DE DYNAMODB

en hasta dos regiones de AWS

Solo paga por los recursos que aprovisiona fuera de los límites de la capa gratuita. La capa gratuita de DynamoDB se aplica a todas las tablas de una región y no vence al final del periodo de 12 meses de la capa gratuita de AWS.

Casos Prácticos

Arquitectura Serverless



Infraestructura Como Código

HashiCorp

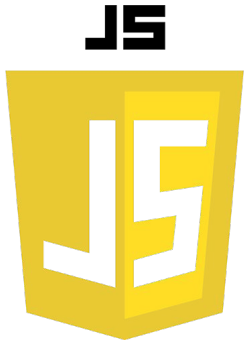
Terraform

Write, Plan, and Create Infrastructure as Code

Creando instancias con Terraform

```
13  resource "aws_dynamodb_table" "basic-dynamodb-table" {
14    name           = "GameScores"
15    read_capacity  = 5
16    write_capacity = 5
17    hash_key       = "UserId"
18
19    attribute {
20      name = "UserId"
21      type = "S"
22    }
23  }
```

Lenguajes




python™
Boto 3

Python Type	DynamoDB Type
string	String (S)
integer	Number (N)
`decimal.Decimal`	Number (N)
`boto3.dynamodb.types.Binary`	Binary (B)
boolean	Boolean (BOOL)
`None`	Null (NULL)
string set	String Set (SS)
integer set	Number Set (NS)
`decimal.Decimal` set	Number Set (NS)
`boto3.dynamodb.types.Binary` set	Binary Set (BS)
list	List (L)
dict	Map (M)

Obtener Registros

```python
def get_item_dynamo(table_name, primary_keys):
    """Get and item from dynamo table."""
    table = dynamodb_resource.Table(table_name)
    filter_key = primary_keys[0]
    if filter_key and primary_keys[1]:
        filtering_exp = Key(filter_key).eq(primary_keys[1])
        response = table.query(KeyConditionExpression=filtering_exp)
    else:
        response = table.query()
    items = response['Items']
    while True:
        if response.get('LastEvaluatedKey'):
            response = table.query(
                ExclusiveStartKey=response['LastEvaluatedKey'])
            items += response['Items']
        else:
            break
    return items
```

Agregar registros

```
34  def add_attribute_dynamo(table_name, primary_keys, attr_name, attr_value):
35      """Adding an attribute to a dynamo table."""
36      table = dynamodb_resource.Table(table_name)
37      item = get_item_dynamo(table_name, primary_keys)
38      if not item:
39          try:
40              response = table.put_item(
41                  Item={
42                      primary_keys[0]: primary_keys[1],
43                      attr_name: attr_value
44                  }
45              )
46              resp = response['ResponseMetadata']['HTTPStatusCode'] == 200
47              return resp
48          except ClientError as e:
49              print e
50              return False
51      else:
52          return update_attribute_dynamo(
53              table_name,
54              primary_keys,
55              attr_name,
56              attr_value)
```

Actualizar registros

```python
def update_attribute_dynamo(table_name, primary_keys, attr_name, attr_value):
    """Updating an attribute to a dynamo table."""
    table = dynamodb_resource.Table(table_name)
    try:
        response = table.update_item(
            Key={
                primary_keys[0]: primary_keys[1],
            },
            UpdateExpression='SET #attrName = :val1',
            ExpressionAttributeNames={
                '#attrName': attr_name
            },
            ExpressionAttributeValues={
                ':val1': attr_value
            }
        )
        resp = response['ResponseMetadata']['HTTPStatusCode'] == 200
        return resp
    except ClientError as e:
        print e
        return False
```

Eliminar registros

```
80  def delete_item_dynamo(table_name, primary_keys):
81      """ Delete an item in a dynamo table."""
82      table = dynamodb_resource.Table(table_name)
83      try:
84          response = table.delete_item(
85              Key={
86                  primary_keys[0]: primary_keys[1],
87              }
88          )
89          resp = response['ResponseMetadata']['HTTPStatusCode'] == 200
90          return resp
91      except ClientError as e:
92          print e
93          return False
```

Trabajando con Listas, Sets y Mapas

```python
def add_list_dynamo(table_name, primary_keys, attr_name, attr_value):
    """Adding list attribute in a table."""
    table = dynamodb_resource.Table(table_name)
    item = get_item_dynamo(table_name, primary_keys)
    if not item:
        try:
            response = table.put_item(
                Item={
                    primary_keys[0]: primary_keys[1],
                    attr_name: attr_value,
                }
            )
            resp = response['ResponseMetadata']['HTTPStatusCode'] == 200
            return resp
        except ClientError as e:
            print e
            return False
    else:
        return update_list_dynamo(
            table_name,
            primary_keys,
            attr_value,
            attr_name,
            item
        )
```

```python
def update_list_dynamo(table_name, primary_keys, attr_value, attr_name, item):
    """Update list attribute in a table."""
    table = dynamodb_resource.Table(table_name)
    try:
        if attr_name not in item[0]:
            table.update_item(
                Key={
                    primary_keys[0]: primary_keys[1],
                },
                UpdateExpression='SET #attrName = :val1',
                ExpressionAttributeNames={
                    '#attrName': attr_name
                },
                ExpressionAttributeValues={
                    ':val1': []
                }
            )
        update_response = table.update_item(
            Key={
                primary_keys[0]: primary_keys[1],
            },
            UpdateExpression='SET #attrName = list_append(#attrName, :val1)',
            ExpressionAttributeNames={
                '#attrName': attr_name
            },
            ExpressionAttributeValues={
                ':val1': attr_value
            }
        )
        resp = update_response['ResponseMetadata']['HTTPStatusCode'] == 200
        return resp
    except ClientError as e:
        print e
        return False
```

```python
def remove_list_dynamo(table_name, primary_keys, attr_name, index):
    """Remove list attribute in a table."""
    table = dynamodb_resource.Table(table_name)
    try:
        response = table.update_item(
            Key={
                primary_keys[0]: primary_keys[1],
            },
            UpdateExpression='REMOVE #attrName[' + str(index) + ']',
            ExpressionAttributeNames={
                '#attrName': attr_name
            }
        )
        resp = response['ResponseMetadata']['HTTPStatusCode'] == 200
        return resp
    except ClientError as e:
        print e
        return False
```

Preguntas ¿?

Gracias