# **Configuration of DynamoDB for Cloud Definitions**

## **Objective**

This stage aimed to set up Amazon DynamoDB as the backend datastore for storing cloud-related terms and their definitions in the Cloud Dictionary Application. DynamoDB was chosen due to its fully managed, serverless, and scalable nature, which ensures high availability and low latency for queries. By designing a dedicated table structure and batch-loading records, the application could achieve efficient storage and retrieval of dictionary entries, offering a seamless experience for users accessing cloud definitions.

## **Step 1: Creation of the DynamoDB Table**

I created a new DynamoDB table named CloudDefinitions to hold the dictionary records. The table was designed with a partition key to ensure unique identification of each term.

* Logged in to the AWS Management Console.
* Opened DynamoDB and navigated to the dashboard.
* Created the table with the following specifications:
  + Table Name: CloudDefinitions
  + Partition Key: term (String)
* Retained the default configurations, such as on-demand capacity mode.

This resulted in a fully configured table, ready to store the dictionary data.

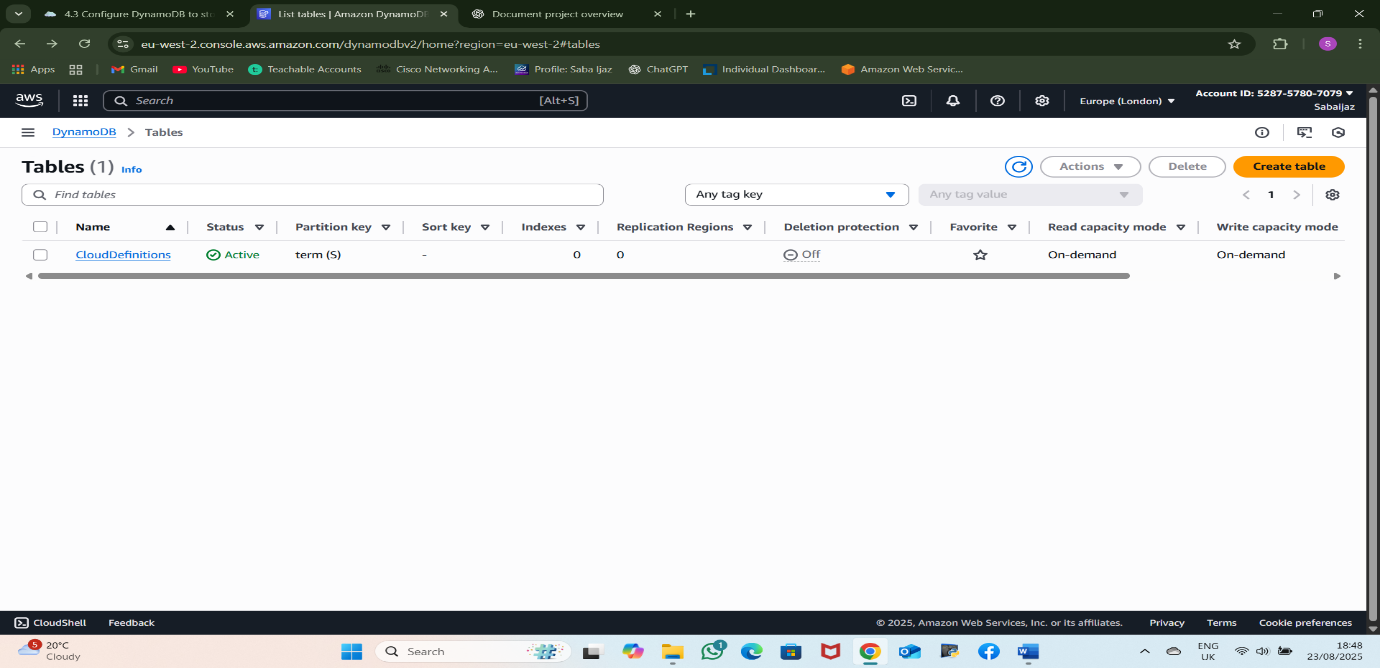


Figure 1: DynamoDB table "CloudDefinitions" created with partition key term

## **Step 2: Preparation of Records for Batch Upload**

The project included four JSON files (records-1.json to records-4.json) provided with the Tech With Lucy course. These files contained cloud-related terms and definitions formatted for DynamoDB batch write item requests.

The files were stored in a records folder within the project directory. Each JSON file contained no more than 25 items, adhering to DynamoDB’s batch-write-item limits.

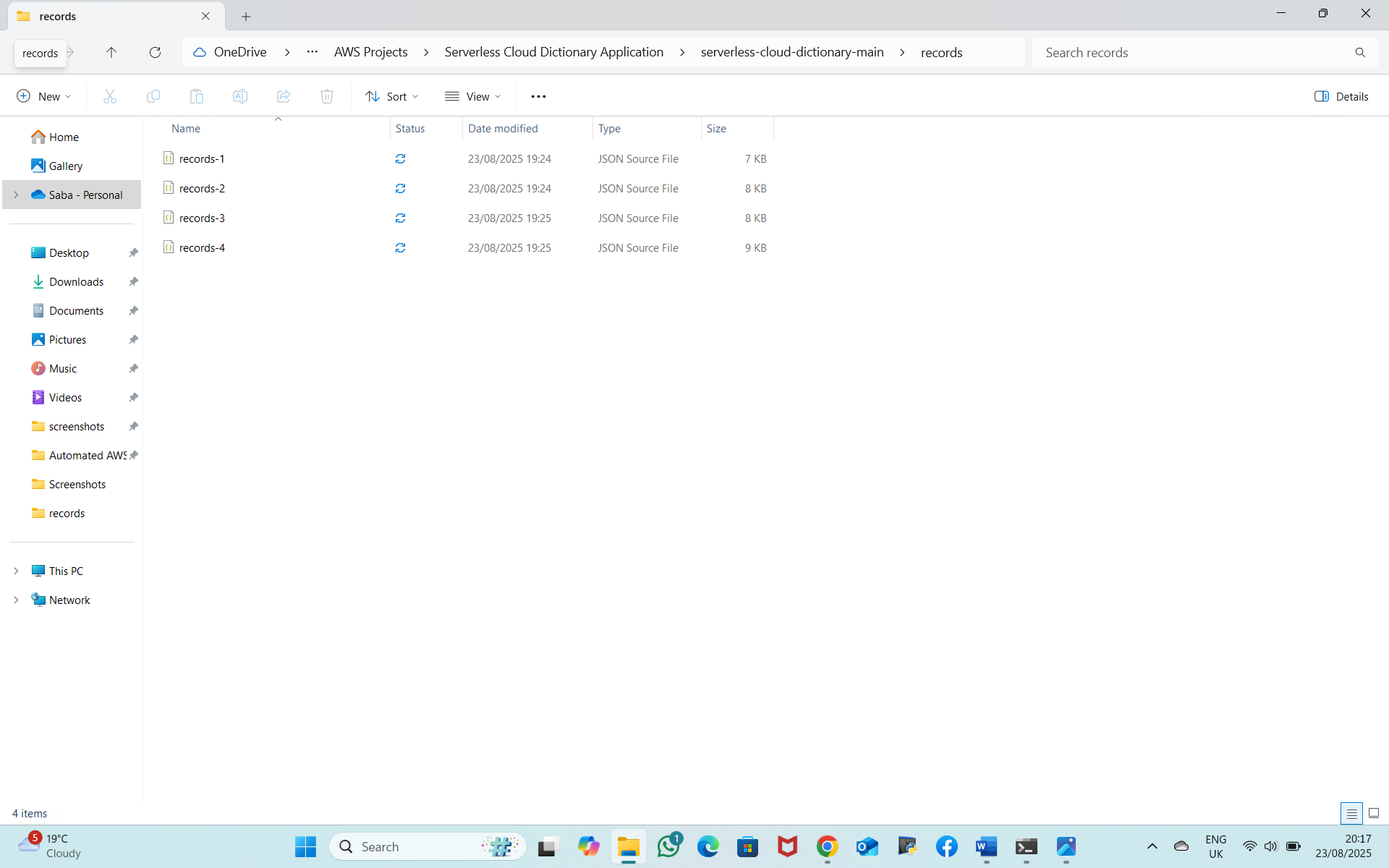


Figure 2: JSON record files structured for batch processing

## **Step 3: Uploading Records Using AWS CLI**

Using the AWS CLI, I uploaded the provided JSON files to the DynamoDB table. Before running the upload, I ensured the AWS CLI was installed and configured with IAM credentials that had access to DynamoDB.

I executed the following commands in the project’s root directory:

aws dynamodb batch-write-item --request-items file://records-1.json

aws dynamodb batch-write-item --request-items file://records-2.json

aws dynamodb batch-write-item --request-items file://records-3.json

aws dynamodb batch-write-item --request-items file://records-4.json

All commands executed successfully, and all records were inserted without errors ("UnprocessedItems": {}).

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 3: Batch upload of records using AWS CLI

## **Step 4: Verification of Records**

I verified the successful upload of the records by checking the Items tab in the DynamoDB console for the CloudDefinitions table. All terms and their corresponding definitions were correctly stored.

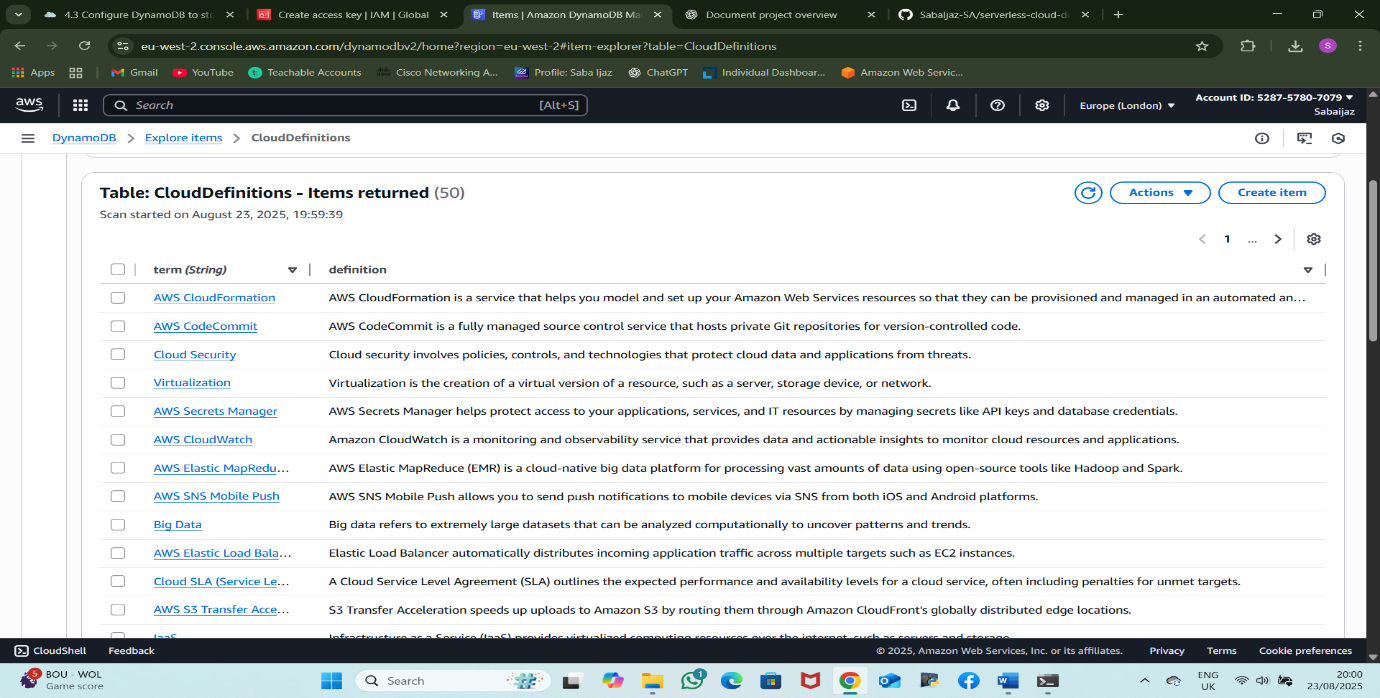


Figure 4: Verified records displayed in the DynamoDB table.

**Outcome**

By completing this step, the CloudDefinitions DynamoDB table was fully configured and populated with the cloud dictionary data. This table is now ready for use by the Lambda functions and API Gateway endpoints in the serverless Cloud Dictionary Application.