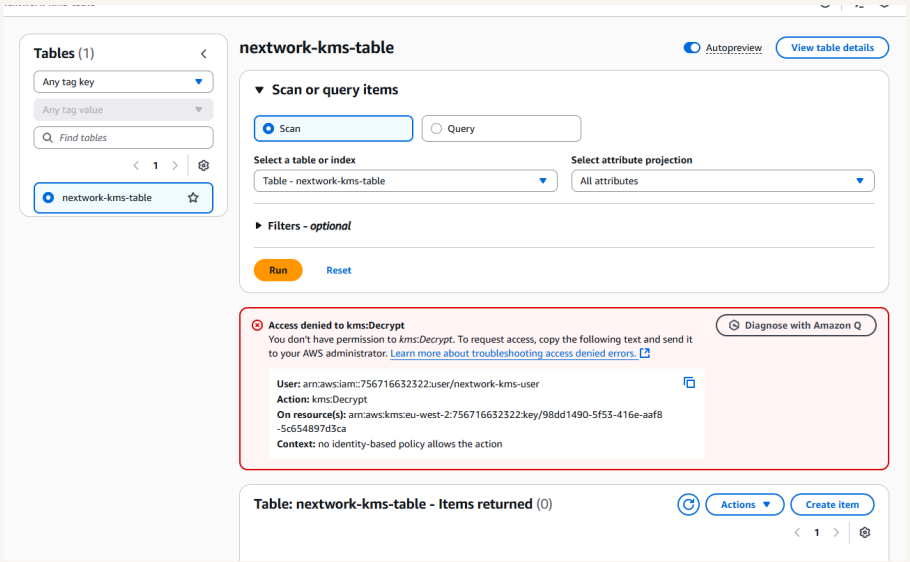
nextwork.org

Encrypt Data with AWS KMS

Louis Moyo



Louis Moyo
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to use AWS KMS to create and manage encryption keys, encrypt a DynamoDB database, and test secure access. The goal is to learn how encryption protects sensitive data and how permissions control who can use or decrypt that data.

Tools and concepts

Services I used include AWS KMS, DynamoDB, and IAM. Key concepts I learnt include encryption at rest, symmetric keys, KMS key policies, IAM permissions, and how encryption ensures that only authorised users can decrypt and access data.

Project reflection

This project took me approximately 1 hour. The most challenging part was understanding how KMS key policies interact with IAM permissions. It was most rewarding to see the test user blocked without KMS access and then successfully decrypt data once permissions were granted.

placeholder

I chose to do this project today because I wanted hands-on practice with encryption in AWS and to better understand how KMS works with DynamoDB. This project met my goals by showing me how encryption protects data even when database access is granted.

Something that would make learning with NextWork even better is adding more advanced scenarios that combine encryption with logging and monitoring for real-world security.

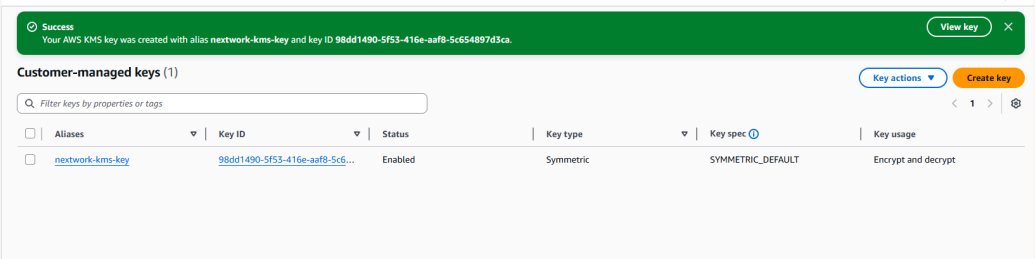
L
Louis Moyo
NextWork Student

Encryption and KMS

Encryption is the process of converting readable data into a scrambled format called ciphertext using algorithms and encryption keys. Companies and developers do this to protect sensitive information from unauthorised access, ensuring that only those with the right keys can decrypt and read it. Encryption keys are digital codes that lock and unlock the data.

AWS KMS is a managed service that lets you easily create, manage, and control encryption keys used to secure data across AWS services. Key management systems are important because they centralise and automate how encryption keys are generated, rotated, and controlled, helping organisations keep data secure and compliant.

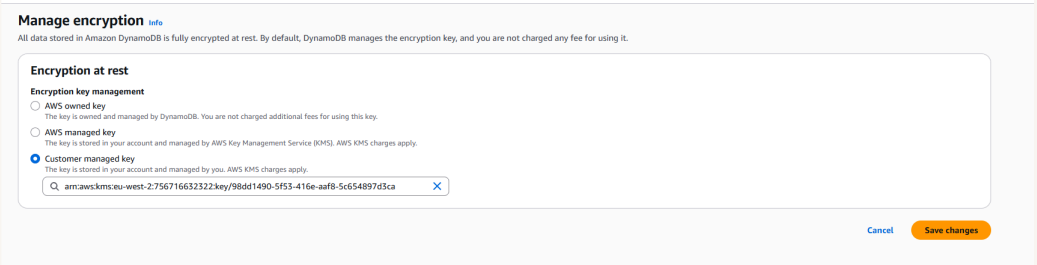
Encryption keys are broadly categorized as symmetric or asymmetric. I set up a symmetric key because it uses a single key for both encryption and decryption, which makes it faster and more efficient for protecting large amounts of data, such as a DynamoDB table.


Success
Your AWS KMS key was created with alias nextwork-kms-key and key ID 98dd1490-5f53-416e-aaf8-5c654897d3ca.
View key
Customer-managed keys (1)
Key actions
Create key
Filter keys by properties or tags
1
Aliases
Key ID
Status
Key type
Key spec
Key usage
nextwork-kms-key
98dd1490-5f53-416e-aaf8-5c6...
Enabled
Symmetric
SYMMETRIC_DEFAULT
Encrypt and decrypt

Encrypting Data

My encryption key will safeguard data in DynamoDB, which is a fully managed NoSQL database service from AWS that stores and retrieves data quickly at any scale.

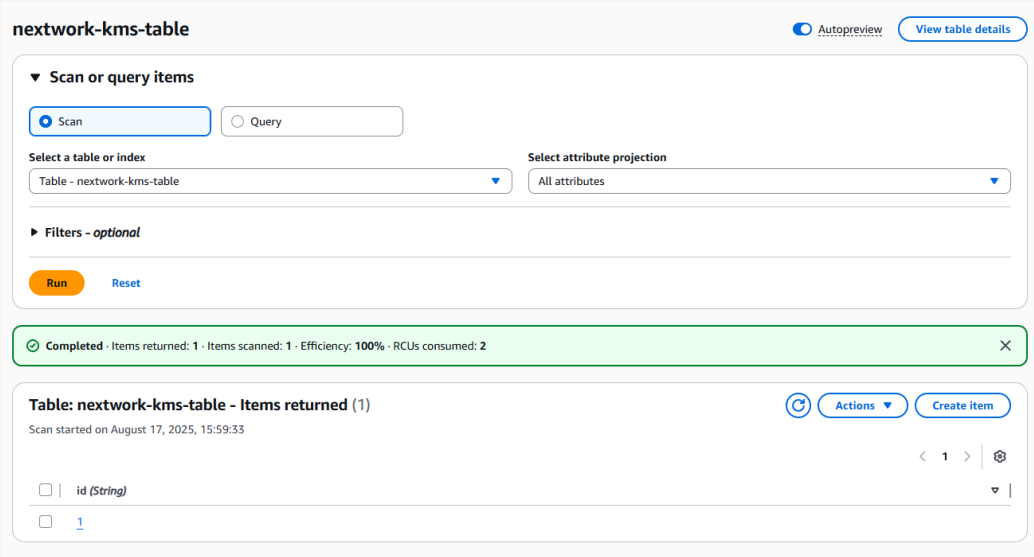
The different encryption options in DynamoDB include AWS owned keys, AWS managed keys, and customer managed keys. Their differences are based on how much control you have - AWS owned keys are automatic with no control, AWS managed keys give limited visibility, and customer managed keys let you fully control permissions and key policies. I selected a customer managed key to have full control over access and encryption settings.



Data Visibility

Rather than controlling who has access to the key, KMS manages user permissions by defining which users or roles can perform actions like encrypt, decrypt, or re-encrypt with that key. This means even if someone has access to the database, they still need explicit KMS permissions to use the key and actually read or protect the data.

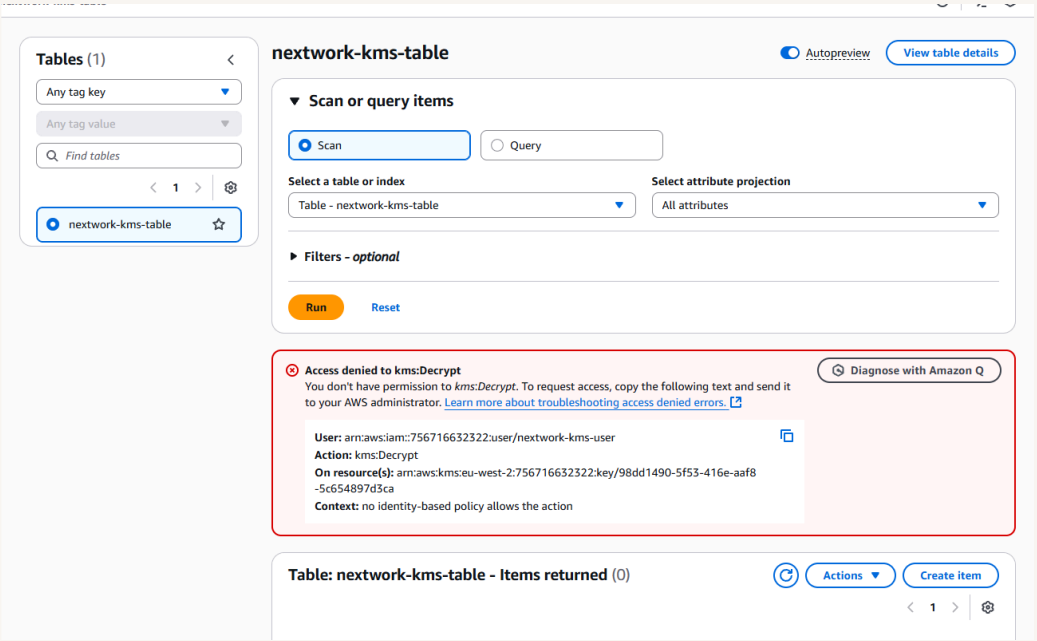
Despite encrypting my DynamoDB table, I could still see the table's items because I'm an authorised user with permission to decrypt the data. DynamoDB uses transparent data encryption, which means the service automatically encrypts data at rest and decrypts it for authorised users without changing how the data looks in the console.


nextwork-kms-table
Autopreview
View table details
Scan or query items
Scan
Query
Select a table or index
Table - nextwork-kms-table
Select attribute projection
All attributes
Filters - optional
Run
Reset
Completed · Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCUs consumed: 2
Table: nextwork-kms-table - Items returned (1)
Scan started on August 17, 2025, 15:59:33
Actions
Create item
1
id (String)
1

Denying Access

I configured a new IAM user to act as a test account. The permission policies I granted this user are full access to DynamoDB, but not any permissions to use my KMS key.

After accessing the DynamoDB table as the test user, I encountered an Access denied error when trying to view items, because the user did not have permission to use the KMS key to decrypt the data. This confirmed that the encryption was working as expected and only authorised users with KMS permissions can read the data.



EXTRA: Granting Access

To let my test user use the encryption key, I updated the KMS key policy to include the IAM user nextwork-kms-user as a principal. My key's policy was updated to allow this user actions such as kms:Encrypt, kms:Decrypt, kms:ReEncrypt*, kms:GenerateDataKey*, and kms:DescribeKey, giving them the ability to decrypt and access data in the DynamoDB table.

Using the test user, I retried accessing the DynamoDB table. I observed that the user could now view and decrypt the table's items, which confirmed that the updated KMS key policy successfully granted them permission to use the encryption key.

Encryption secures data instead of just restricting who can log in or query it. I could combine encryption with IAM access controls and resource policies to make sure that even if someone gets database access, they still need the correct KMS key permissions to actually read the data.


Key policy
36],
37 "Resource": "*"
38 },
39 {
40 "Sid": "Allow use of the key",
41 "Effect": "Allow",
42 "Principal": {
43 "AWS": [
44 "arn:aws:iam::756716632322:user/louismoyo",
45 "arn:aws:iam::756716632322:user/nextwork-kms-user"
46]
47 },
48 "Action": [
49 "kms:Encrypt",
50 "kms:Decrypt",
51 "kms:ReEncrypt*",
52 "kms:GenerateDataKey*",
53 "kms:DescribeKey"
54],

The place to learn & showcase your skills

Check out nextwork.org for more projects