



Key Management Service

AWS Key Management Service (KMS) is a managed service that makes it easy to create and control cryptographic keys used to secure your data. Here's a brief overview:

Key Features:

1. **Create and Manage Keys:**
 - Allows you to create, rotate, disable, and delete encryption keys.
 - Supports symmetric and asymmetric keys for different encryption needs.
2. **Integrated Encryption:**
 - Seamlessly integrates with other AWS services like S3, RDS, EBS, and Lambda to provide built-in encryption.
3. **Access Control:**
 - Uses AWS Identity and Access Management (IAM) policies and grants to control who can use and manage keys.
 - Supports fine-grained permissions for specific actions on keys.
4. **Key Usage Auditing:**
 - Integrates with AWS CloudTrail to log all key usage and management activities, helping with compliance and security audits.
5. **Automatic Key Rotation:**
 - Supports automatic rotation of keys, which can reduce the risk of compromised keys by periodically replacing them.
6. **Customer-Managed Keys (CMKs):**
 - Allows you to create and manage your own encryption keys with full control over their lifecycle.

Common Use Cases:

- **Encrypting Data at Rest:** Secure data stored in AWS services like S3, RDS, and EBS.
- **Secure Communication:** Encrypt sensitive data before sending it over the network.
- **Digital Signatures:** Use asymmetric keys to create and verify digital signatures for data integrity and authenticity.

Benefits:

- **Enhanced Security:** Provides strong encryption and key management practices to protect your data.
- **Ease of Use:** Simplifies the complex task of managing encryption keys while integrating with AWS services.

- **Compliance:** Helps meet regulatory and compliance requirements by providing detailed logs and audit trails.

AWS KMS is crucial for managing encryption keys in a secure and scalable manner, ensuring that your data is protected both at rest and in transit.

😊 To begin with the Lab:

1. Login to AWS Console and search for KMS. Choose this service accordingly.

2. Now from KMS dashboard if you will go to AWS Managed keys, there are several keys that are managed by AWS.
3. You can also create your own key, for that click on Customer managed keys.

Aliases	Key ID	Status
aws/sns	01ee9d91-f480-416f-a109-4c3797ab481e	Enabled
aws/redshift	1596e278-69f7-4877-aecf-8c905bf0aa98	Enabled
aws/lambda	793bbff-3ab2-44e7-8b2a-fccb8f7b59f9	Enabled
aws/rds	89e531f6-465c-4738-8e41-76fbad3a0861	Enabled
aws/elasticfilesystem	8edbc375-3f8a-43b3-b515-77c667c4552b	Enabled
aws/backup	a895f96f-8be5-49c9-991f-ea14c64627ca	Enabled

4. Here click on create key.

5. Now choose Symmetric key type for encrypt and decrypt usage. Then click on next.

Configure key

Key type [Help me choose](#)

Symmetric
A single key used for encrypting and decrypting data or generating and verifying HMAC codes

Asymmetric
A public and private key pair used for encrypting and decrypting data or signing and verifying messages

Key usage [Help me choose](#)

Encrypt and decrypt
Use the key only to encrypt and decrypt data.

Generate and verify MAC
Use the key only to generate and verify hash-based message authentication codes (HMAC).

► **Advanced options**

[Cancel](#) [Next](#)

6. On the next page you just need to give it a name. then go to next page.

Add labels

Alias
You can change the alias at any time. [Learn more](#)

Alias

7. For step 3 skip it then on step 4 you need to choose an IAM user. If you don't have any then create one.
8. After that just create your key.

KMS > Customer managed keys > Create key

Step 1
[Configure key](#)

Step 2
[Add labels](#)

Step 3
[Define key administrative permissions](#)

Step 4
Define key usage permissions

Step 5
Review

Define key usage permissions

Key users (1/44)
Select the IAM users and roles that can use the KMS key in cryptographic operations. [Learn more](#)

<input type="checkbox"/>	Name	Path	Type
<input checked="" type="checkbox"/>	demouser	/	User
<input type="checkbox"/>	AmazonEC2RoleForSSM	/	Role
<input type="checkbox"/>	AmazonSSMRoleForInstances...	/	Role
<input type="checkbox"/>	aws-elasticbeanstalk-service-r...	/service-role/	Role
<input type="checkbox"/>	AWS-QuickSetup-StackSet-Lo...	/	Role
<input type="checkbox"/>	AWS-QuickSetup-StackSet-Lo...	/	Role
<input type="checkbox"/>	AWSReservedSSO_AdminLeve...	/aws-reserved/sso.amazonaws...	Role
<input type="checkbox"/>	AWSServiceRoleForAmazonEKS	/aws-service-role/eks.amazon...	Role
<input type="checkbox"/>	AWSServiceRoleForAmazonEK...	/aws-service-role/eks-nodegro...	Role
<input type="checkbox"/>	AWSServiceRoleForAmazonEla...	/aws-service-role/elasticfilesys...	Role

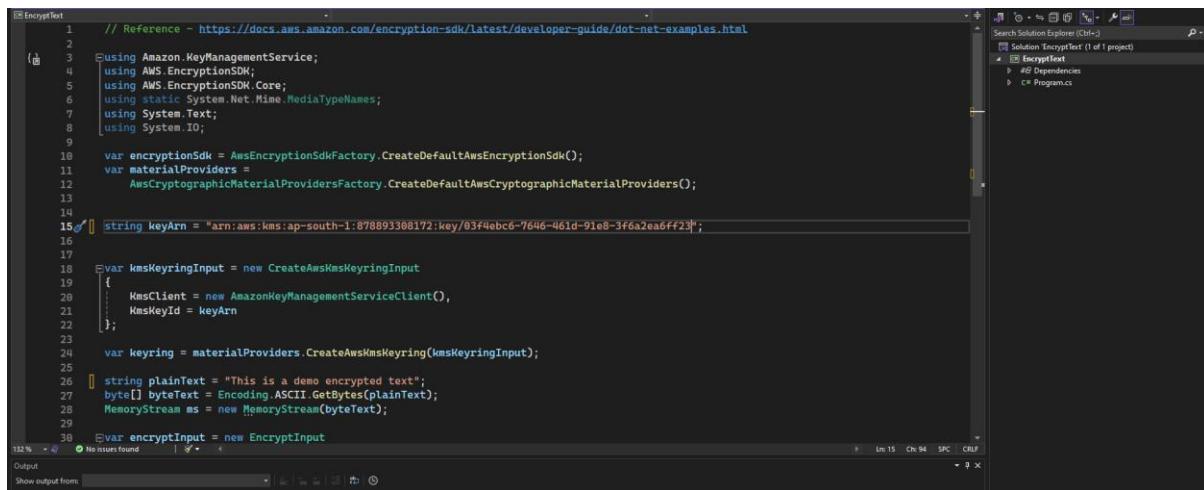
KMS > Customer managed keys

Customer managed keys (1)

<input type="checkbox"/>	Aliases	Key ID	Status	Key type	Key spec	Key usage
<input type="checkbox"/>	demokey	03f4ebc6-7646-461d-91e8-3f6a2ea6ff23	Enabled	Symmetric	SYMMETRIC_DEFAULT	Encrypt and decrypt

[Key actions](#) [Create key](#)

9. After that you need to download a program from GitHub with name encrypttext.zip. it will be in zip format, so download it and unzip it.
10. After that you need to open this program in visual studio.
11. Now go back to AWS Console and open your key, in there you will see your ARN for the key copy it and paste it in the program.



```

1 // Reference - https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/dot-net-examples.html
2
3 using Amazon.KeyManagementService;
4 using AWS.EncryptionSDK;
5 using AWS.EncryptionSDK.Core;
6 using static System.Net.Mime.MediaTypeNames;
7 using System.Text;
8 using System.IO;
9
10 var encryptionSdk = AwsEncryptionSdkFactory.CreateDefaultAwsEncryptionSdk();
11 var materialProviders =
12     AwsCryptographicMaterialProvidersFactory.CreateDefaultAwsCryptographicMaterialProviders();
13
14 string keyArn = "arn:aws:kms:ap-south-1:878893308172:key/03f4ebc6-7646-461d-91e8-3f6a2ea6ff23";
15
16 var kmsKeyringInput = new CreateAwsKmsKeyringInput
17 {
18     KmsClient = new AmazonKeyManagementServiceClient(),
19     KmsKeyId = keyArn
20 };
21
22 var keyring = materialProviders.CreateAwsKmsKeyring(kmsKeyringInput);
23
24 string plainText = "This is a demo encrypted text";
25 byte[] byteText = Encoding.ASCII.GetBytes(plainText);
26 MemoryStream ms = new MemoryStream(byteText);
27
28 var encryptInput = new EncryptInput
29 {
30     Keyring = keyring,
31     PlainText = plainText
32 };
33
34 var encryptOutput = encryptionSdk.EncryptAsync(ms, encryptInput).Result;
35
36 byte[] encryptedText = encryptOutput.Ciphertext;
37
38 var decryptedText = materialProviders.Decrypt(keyring, encryptOutput);
39
40 string decryptedTextContent = Encoding.ASCII.GetString(decryptedText);
41
42 Output.WriteLine("Decrypted text: " + decryptedTextContent);

```

12. After that also remember to configure your IAM user in your local system.
13. Now just run this program locally.
14. You will see the encrypted text.

15. For all the key management service is doing, is it helping you to store that encryption key. At any point in time, you can make a call onto the key management service from your program. Use that encryption key to encrypt your data.