

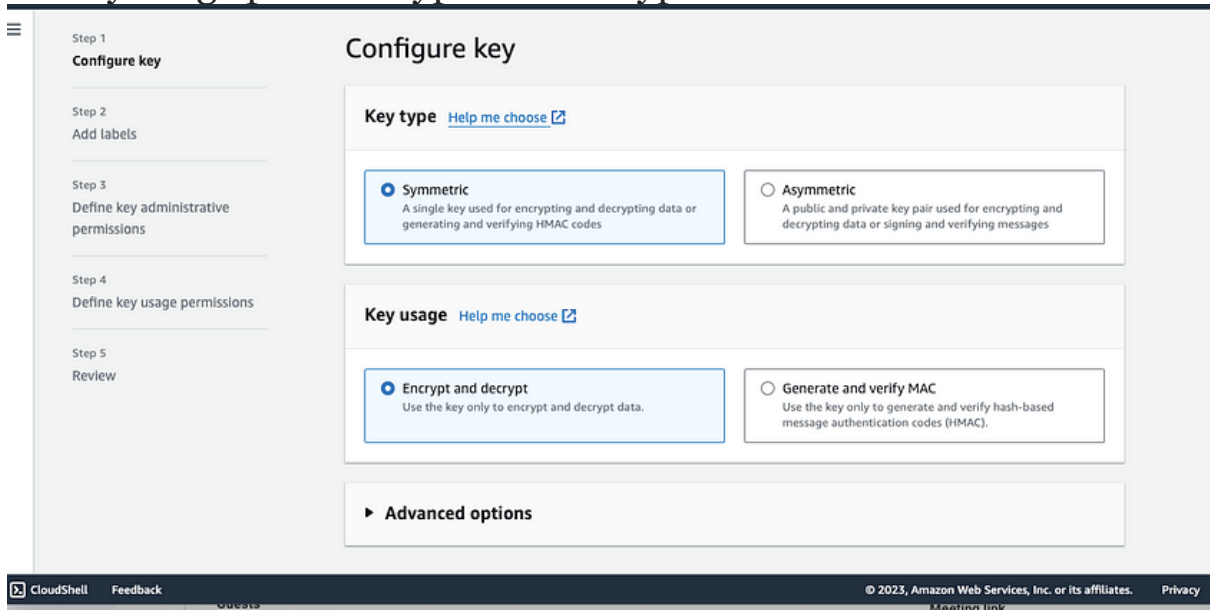
1. Log into your aws account.

2. Search for KMS service and click on it.

3. Click on the “create a key” orange button on the console.

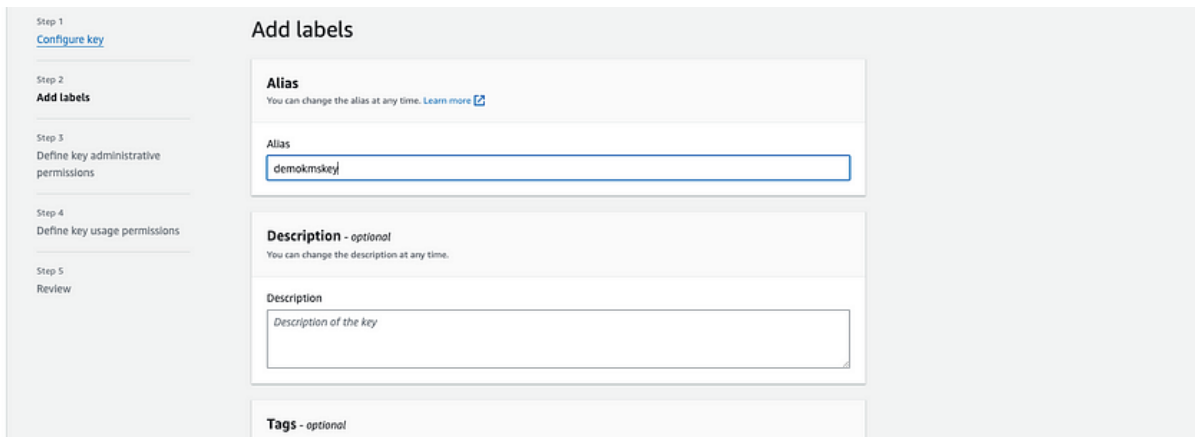


4. Create Customer Managed Key. Pick Symmetric for key type and for key usage pick Encrypt and Decrypt. Then Click Next.



5. You will need to give this Customer Managed Key an alias name.

I named mine demokmskey. Feel free to give your CMK key a basic name as well. We will use this alias name later. After typing your alias name click next



Step 1  
[Configure key](#)

Step 2  
**Add labels**

Step 3  
Define key administrative permissions

Step 4  
Define key usage permissions

Step 5  
Review

### Add labels

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

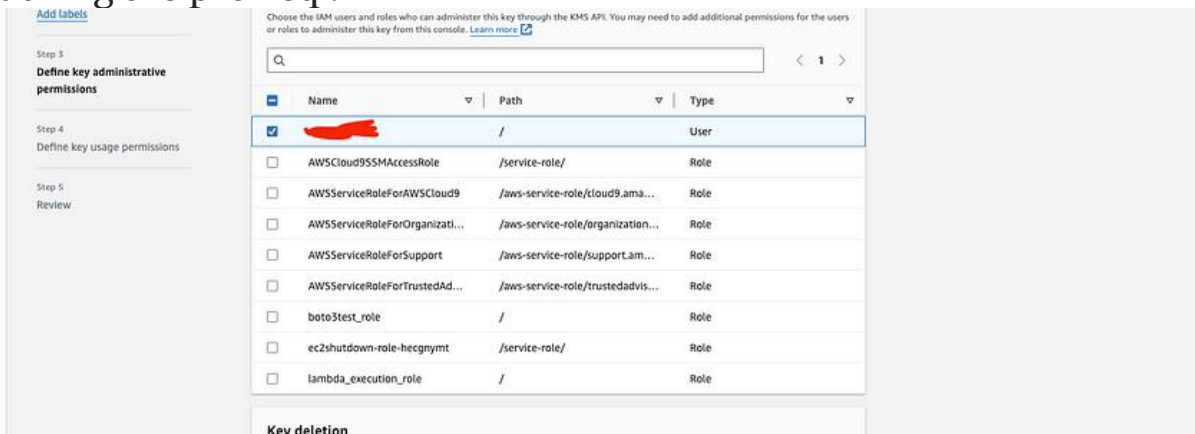
Alias  
demokmskey

**Description - optional**  
You can change the description at any time.

Description  
Description of the key

**Tags - optional**

6. Pick a key administrator for this key. To make things easy I will choose the user that I am currently logged in with. We created a user during the pre-req .



[Add labels](#)

Step 3  
**Define key administrative permissions**

Step 4  
Define key usage permissions

Step 5  
Review

Choose the IAM users and roles who can administer this key through the KMS API. You may need to add additional permissions for the users or roles to administer this key from this console. [Learn more](#)

Search

<input checked="" type="checkbox"/>	Name	Path	Type
<input checked="" type="checkbox"/>	[REDACTED]	/	User
<input type="checkbox"/>	AWSCloud9SSMAccessRole	/service-role/	Role
<input type="checkbox"/>	AWSServiceRoleForAWSCloud9	/aws-service-role/cloud9.ama...	Role
<input type="checkbox"/>	AWSServiceRoleForOrganizati...	/aws-service-role/organization...	Role
<input type="checkbox"/>	AWSServiceRoleForSupport	/aws-service-role/support.am...	Role
<input type="checkbox"/>	AWSServiceRoleForTrustedAd...	/aws-service-role/trustedadv...	Role
<input type="checkbox"/>	boto3test_role	/	Role
<input type="checkbox"/>	ec2shutdown-role-hecgnymt	/service-role/	Role
<input type="checkbox"/>	lambda_execution_role	/	Role

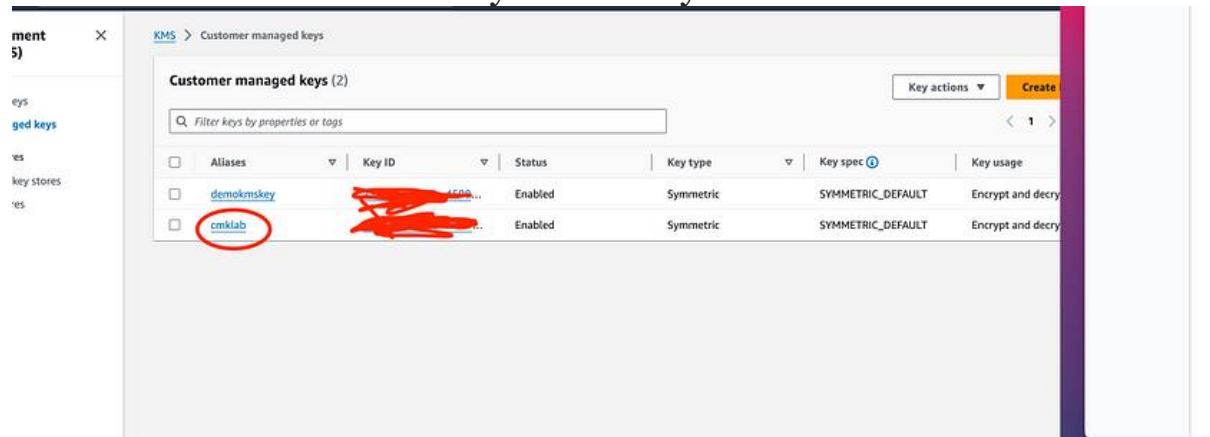
**Key deletion**

7. Ensure the key deletion box is checked, then click next.

8. For key usage permissions pick the same user picked from the previous . This is permission on who can use the kms key for cryptographic operations.

9. Click next. Now you will be at the review stage. Click finish at the bottom right.

10. You now have a kms key. Check if your cmk has been created.



## Using the CMK for encryption and Decryption

1. Open up your terminal

2. Create a folder called kmslab

3. Navigate to this folder in the command line.

4. Make sure you have aws cli configured by running “aws configure” on the command line. Configure your credentials regions and ensure the output format is json.

```
amohamed@Abdurahmans-MacBook-Pro kmslab % aws configure
AWS Access Key ID [*****V604]:
AWS Secret Access Key [*****nwxN]:
Default region name [us-east-1]:
Default output format [text]: json
```

5. Run the command nano passwords.txt

6. We are going to create a mock text file that contains a few passwords. Remember this is a mock text file.

```
GNU nano 2.9.6 File: passwords.txt
Database Password: Password123
Server Password: Password321
```

7. Once you create the text file, save and exit the file.

8. We are now going to use the KMS cmk we created earlier via the cli create data encryption keys for the CMK. Make sure you remember the alias of that cmk we created in the previous .

9. Run this aws cli command and ensure that you change the alias and the region to the alias you created originally as well as the region you created that cmk. For my case, my alias is called cmklab and the region is us-east-1.

Run “aws kms generate-data-key — key-id alias/cmklab — key-spec AES\_256 — region us-east-1”

10. After running that command you should get a ciphertextblob, plaintext, and a key id listed.

11. The plaintext portion that we received is our unencrypted datakey the ciphertext portion is our encrypted data key.

12. Since both of these keys are base64 encoded we will have to decode them and then save them into their own separate files.

13. Copy the command below. Make sure you get the plaintext output from the previous and past it inside the quotations.

```
echo "Insert plaintext data key from previous " | base64 -- decode > datakey.txt
```

14. You should now have a datakey.txt file. Run ls in your kmslab folder you should see passwords.txt and datakey.txt

15. Now, we are going to do the same thing but this time we are going to create an encrypteddatakey file with our ciphertext instead of the plaintext.

16. Run the command:

```
Echo "Insert ciphertext data key from 10" | base64 -- decode > encrypteddatakey.txt
```

16. You should now have 3 files in your kmslab folder: datakey.txt, encrypteddatakey.txt and passwords.txt

18. Now that we have decoded both files we are going to use the datakey.txt (our DEK) to encrypt our passwords.txt file. This will create a file called passwords-encrypted. Run this command:

```
openssl enc -in ./passwords.txt -out ./passwords-encrypted.txt -e -  
aes256 -k fileb:///./datakey
```

19. After running this command you should now have a file called passwords-encrypted.txt. Run the ls command to ensure its there.

20. After running ls, run “cat passwords-encrypted.txt” on the command line. You should get a response that says “Salted....”

21. At this time we have 2 data encryption keys, ones public data key and ones an encrypted data key. Having both is not essential, so we will remove the the public datakey.txt

Run “rm datakey.txt”

22. Now lets say if we wanted to decrypt data with an encrypted data encryption key, how do we do that? We will use are encrypted data key but we will have to decrypt it first.

Run this command:

```
aws kms decrypt — ciphertext-blob fileb:///./encrypted-datakey.txt  
— region us-east-1
```

23. AFter running this command you should get an output that has a plaintext a cipher text and a keyid.

24. We are now going to make a file called datakey.txt again (it got deleted earlier). Like last time we have to make sure that it base64 decoded (its encoded at this time).

Run this command:

```
echo "Insert plaintext from 22" | base64 -- decode > datakey.txt
```

25. Now that we have our decrypted datakey.txt back we will try using this key to decrypt our passwords-encrypted.txt file

26. Run this command

```
openssl enc -in ./passwords-encrypted.txt -out ./passwords-decrypted.txt -d -aes256 -k file:///./datakey.txt
```

27. Now that you have your passwords decrypted run “cat passwords-decrypted.txt” on your command prompt. You should see your old passwords again