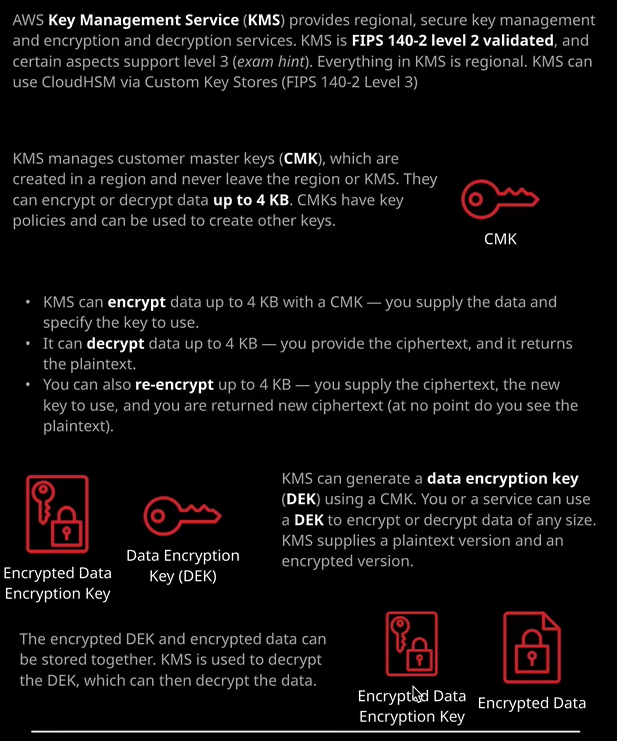
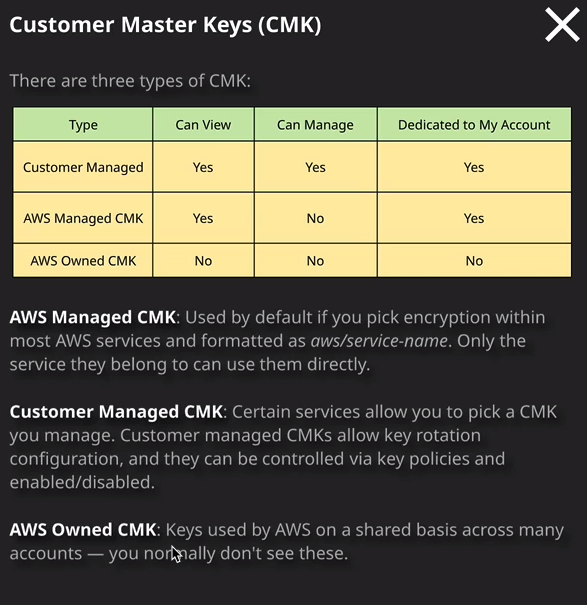
Welcome back, and in this lesson, I want to talk about AWS Key Management Service, which is, in my opinion, one of the most important services in AWS if you perform any form of encryption.



**The Key Management Service, or KMS, provides regional, secure key management and encryption and decryption services. Now KMS is FIPS 140-2 level 2 validated.** That might not mean much if you not a security professional but just remember those words. So if you see any questions that mention FIPS 140-2 level 2 then it's likely to be KMS. If you see any questions that talk about FIPS 140-2 level 3 then it's likely to be a product called Cloud HSM.

Now, historically, KMS has been a product that hasn't really featured all that heavily at an associate level, but it's becoming more important**. KMS used to be part of the IAM product, but it's now been split off into its own product. It's got its own features, and its own area of the console. Now KMS allows you to create, modify, and delete what are known as customer master keys or CMKs. Now, as well as being able to interact with CMKs KMS also allows you to encrypt, decrypt, and reencrypt data of up to four kilobytes in size using those CMKs.** So that's a really important limit to understand. By **default, KMS is capable of performing cryptographic operations on data up to four kilobytes in size, and it can do that using CMKs**.



Now there are a number of different types of CMKs. We start with the AWS owned CMKs which are kind of hidden you don't see them in any area of the console, and they're not dedicated to your account. So these are things that are used across all AWS accounts, and they're generally used for AWS level encryption and decryption. I mentioned them only so that you're aware of them. You probably won't have anything to do with them if you use this in production, and certainly not for any exam questions. **The two main types of CMKs that you will need to understand are AWS managed CMKs and customer managed CMKs.** So whenever you configure any form of encryption within AWS products, in all likelihood, it's going to use KMS in some way. So earlier in the course, I talked about S3, and I mentioned how S3 is capable of encrypting objects inside a bucket. So every object is capable of being given an encryption state either it's encrypted or it's not encrypted.

**Now the default method of encryption inside S3 is to use S3 and KMS but it uses AWS managed customer master keys, which essentially means the product manages everything on your behalf. So when you're using AWS managed CMKs you can view them, but you can't manage them. They are dedicated to your account, but essentially the only thing that can use AWS manage CMKs is the service that they're intended for.**

So if I go to the console and I look at AWS managed keys any service that I've used in this account, which has had encryption configured, will have generated its own AWS managed customer master key. So in this case, it's RDS, EBS, Lambda, Connect, and Secrets Manager. Now I want to contrast that with customer managed customer master keys, these are keys that you can create so you can view them, you can manage them, and they are dedicated to your account. So if, for example, you configure KMS encryption on S3, then it uses a customer managed customer master key. So this is a key that you'll create and you'll be able to manage the permissions on. You'll be able to enable and disable it, configure rotation, and many other aspects. So essentially, just remember this for the exam: **If you are configuring encryption in a product and you get to select which key to use, it's likely to be a customer managed key. If you're using the default encryption with any AWS services, it's likely to be an AWS managed key and for any of the AWS managed keys you don't get to configure any details. It essentially exists. It can be used by the product that created it. You don't get to manage it. You can't configure it and can't make any changes to it. But if you've got a customer managed key, then you have the ability to update the key policy and control rotation.**

Now I want to demonstrate exactly how KMS works for basic encryption of up to four kilobytes in size and to do that, I'm going to move across to my command line. Now, on this command line, I've got it configured using the AWS CLI tools to a user in my AWS account and that user has got full administrative permissions so I can do anything I want in this account. So I'm going to go ahead and create a customer managed CMK. So to do that, go to AWS and then KMS, and then the command I need is create-key. I need to specify double hyphen and then description, and I need to give this key a description. So I'm going to call it LA for Linux Academy KMS DMO CMK and then I'll press enter. Now my command line tools are set up to use U.S. East one as my default region. Remember, **KMS is a regional service, so everything that you create by default is done in a region, and it never leaves that region. Any CMKs that you create inside KMS you can't get access to them directly. It has to go through the KMS service. So I've just created a CMK and I will never have access to this CMK unless I'm using the AWS APIs to interact with it. So that's important to realize it's a completely secure service but you can't move the keys out of this region and you can't get access to them. So if you want to, for example, encrypt an S3 object with the CMK and then move that S3 object to a different region, you need to reencrypt that object with a CMK in the destination region.**

Now, **by default, every single CMK that you create inside KMS has a unique key ID and that's the key ID that you can use to reference that key if ever you need to interact with it. But luckily, you can also create a key alias, and you can think about this as a short cut. So I can create a key alias** and have it pointed at one key and then I could change that alias and point it at another key and as long as my applications were set to use the alias, I wouldn't need to make any other configuration changes. So if I wanted to make an alias to this key I'd go aws kms then create-alias. I'd need to specify the target key ID. So that's the CMK I've just created so target-key-id and then it's this key ID that I need to specify. So I copy that into my clipboard and paste that in. I'll need to provide an alias name. So --alias-name and then speech marks and I'm going to give it the name "alias/lakmsdemo" and that's created the alias. If I go back to the KMS console and I go to customer managed keys and I refresh the page, you can see there we've got the LA KMS demo. So that's been set up as an alias and this is the key ID for that key and I can open that up and because it's a key that I manage I'm able to update the key policy by default any keys that you create in an account do trust everyone in that account. So the key, by definition, trusts the account, and then you need to give identity policies that allow individual identities in the account to use that key. Do be especially careful. If you do delete this default key policy, you won't have any access, and you'll need to log a support ticket.

**You can update the key rotation on a customer managed key so you can specify that it rotates the underlying key material to the actual data that's used to encrypt and decrypt. You can specify that it rotates that, and for a customer managed key, you can set that to be enabled or disabled and the possible value is to rotate it every year. For an AWS managed key that's enabled. You can't disable it, and it's set to rotate every three years, which is 1095 days**.

Okay, so now I want to demonstrate encrypting something with the CMK. So I'm going to go back to my terminal, and I'm going to generate some plain text. So I'm going to go echo. This is a secret message. I could just output that to the screen but what I want to do is generate a file from it. So I'm going to go topsecret.txt. So that's redirecting that message to that file. If I cat topsecret.txt we can see that the message is inside and that is the only file in this folder. So next I want to run an AWS KMS encrypt. So that's an encrypt operation. I'll need to specify the key ID and I copied this down in my clipboard earlier. So that's the unique key ID for the CMK that I created earlier in this lesson then I'll need to specify the plain text that I want to encrypt. That's --plaintext text and I can give it the name of a file by specifying file: and then double forward slash topsecret.txt --output text because I wanted to return some text and then --query and then what I want is the cipher text output so cipher text blob. So this is going to ask KMS to encrypt this plain text. I'm specifying the file topsecret.txt. I'm specifying that I want the output to be text and I'm using the double hyphen query to specifically ask for the cipher text so the encrypted output. So I'll press "Enter" and what I get back is the base 64 encoded version of our cipher text. Now, **KMS generally uses base 64 encoded text for any operations where you need to send data to it or receive data from it**. So just keep that in mind. I don't want to make this too complicated, but you do need to be aware that this is base 64. Now what I need to do to actually generate a file with this encrypted data is to take the previous command that I used and the pipe it to base 64 which is the utility that I've got on Mac OS double hyphen decode and then redirect that out to a file called topsecret.encrypted, and this will become my actual encrypted file and if I cat this file, you'll see that it's now binary data. It's actually the encrypted cipher text version of the file with my message in. Now if you are using a Windows operating system, I'll make sure that I include links in the lesson description, which show you the exact commands that you'll need to run on Windows because they are slightly different. But what I have essentially done there is I've used a CMK to encrypt some plain text, and I've got back some cipher text. Now, this lesson is getting a little bit on the long side. So I wanted to split it into a number of different parts and give you the opportunity to take a small break. So this is the end of part one. Go ahead and mark this lesson as complete and when you're ready, you can join me in part two.