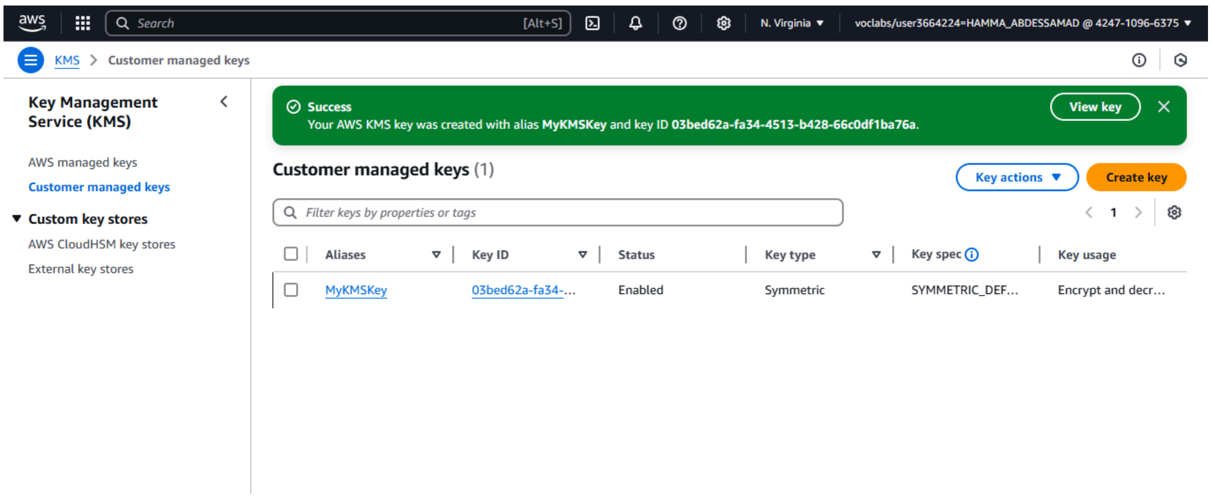
Lab: Protecting Data in Your Application

Hamma Abdessmad - GL

Creating an AWS KMS key:

In this task, we created our first AWS KMS key, setting up a symmetric key with voclabs permissions for both administration and usage. This key will be fundamental as we progress through the lab to encrypt data in S3 and EC2.



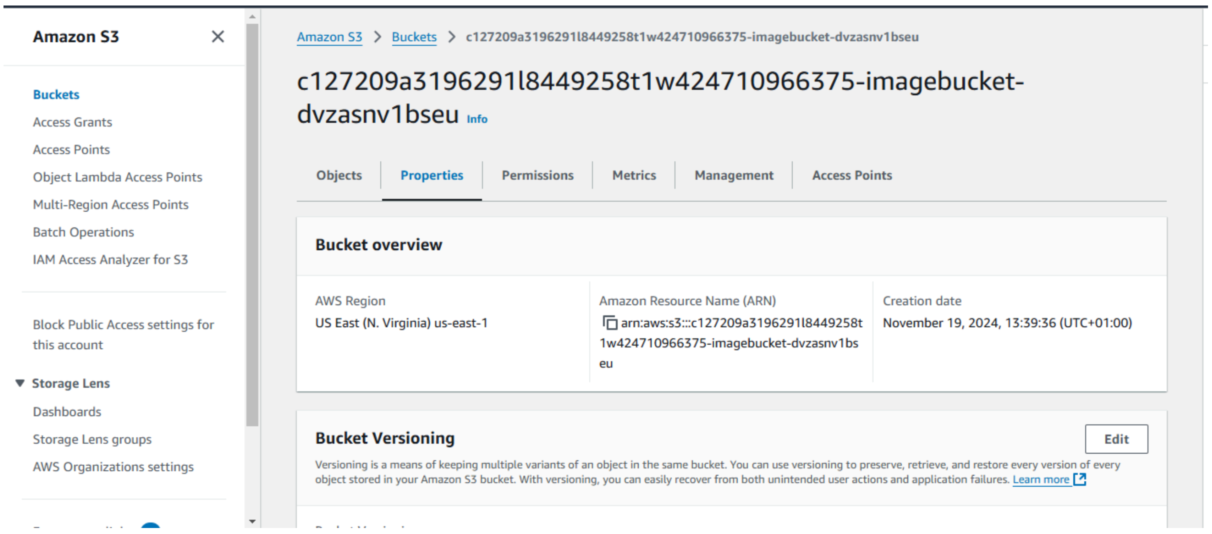
Success
Your AWS KMS key was created with alias MyKMSKey and key ID 03bed62a-fa34-4513-b428-66c0df1ba76a.

Customer managed keys (1)

Aliases	Key ID	Status	Key type	Key spec	Key usage
MyKMSKey	03bed62a-fa34-...	Enabled	Symmetric	SYMMETRIC_DEF...	Encrypt and decr...

Storing an encrypted object in an S3 bucket:

First we proceeded to examine the S3 bucket's encryption configuration in the Properties tab. We confirmed that default encryption is already enabled on our imagebucket, ensuring automatic encryption of any new objects we store.



Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Amazon S3 > Buckets > c127209a3196291l8449258t1w424710966375-imagebucket-dvzasnv1bseu
Objects
Properties
Permissions
Metrics
Management
Access Points
Edit

c127209a3196291l8449258t1w424710966375-imagebucket-dvzasnv1bseu Info

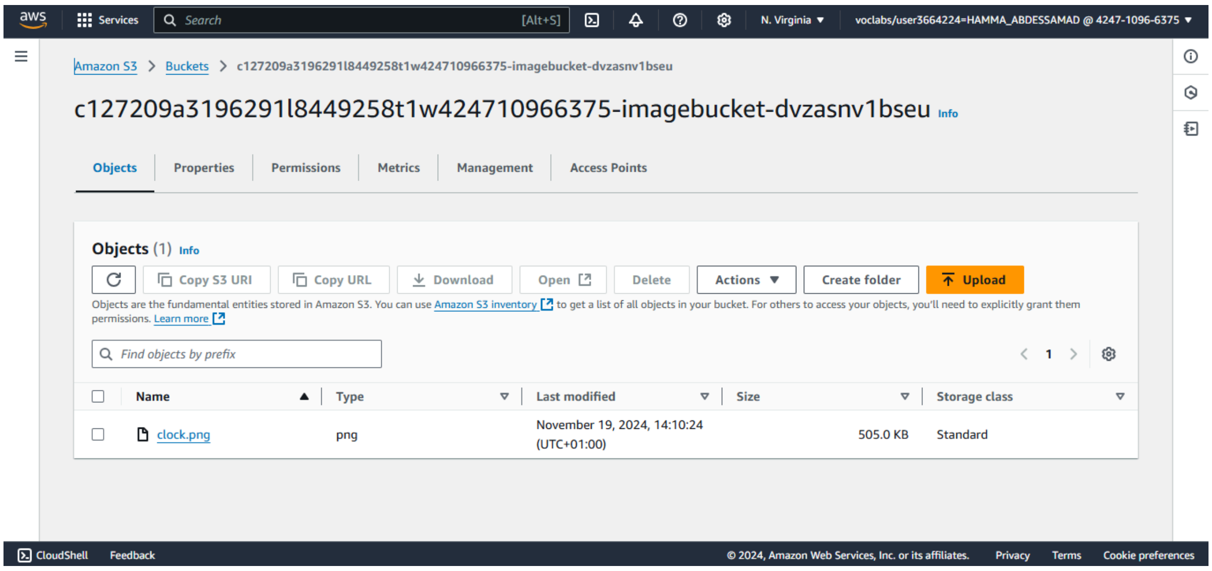
Bucket overview

AWS Region	Amazon Resource Name (ARN)	Creation date
US East (N. Virginia) us-east-1	arn:aws:s3:::c127209a3196291l8449258t1w424710966375-imagebucket-dvzasnv1bseu	November 19, 2024, 13:39:36 (UTC+01:00)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

We uploaded a test image (clock.png) to our S3 bucket using advanced encryption settings. Instead of relying on default bucket encryption, we specifically chose our custom KMS key 'MyKMSKey' to implement SSE-KMS encryption, giving us more control over our security implementation. Through the console, we could verify that the object was successfully encrypted by checking its server-side encryption settings.



c127209a3196291l8449258t1w424710966375-imagebucket-dvzasnv1bseu Info

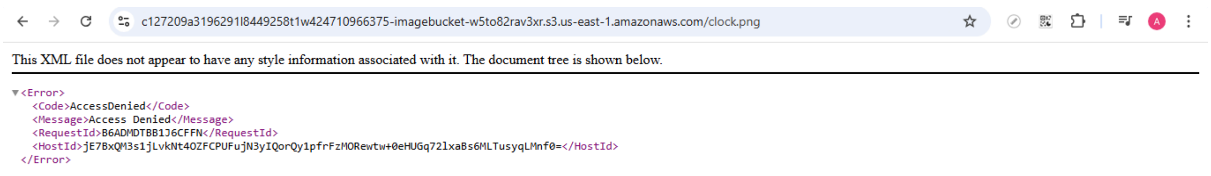
Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
clock.png	png	November 19, 2024, 14:10:24 (UTC+01:00)	505.0 KB	Standard

Attempting public access to the encrypted object

We tested the security of our encrypted object by attempting to access it through its S3 object URL. As expected, we received an 'Access Denied' error, demonstrating how AWS S3's default security posture blocks public access to our encrypted objects even when having their direct URL. This validates both the encryption and bucket-level security controls we have in place.

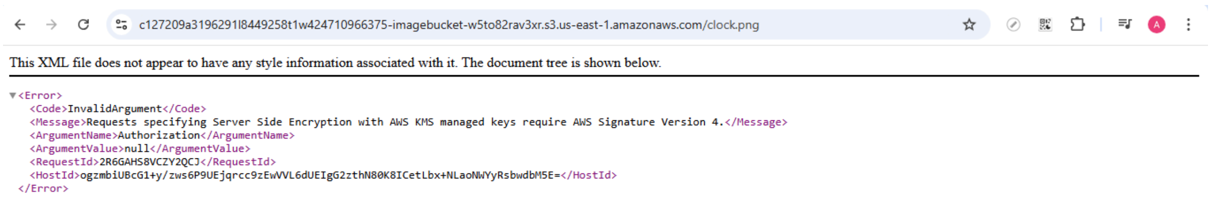


c127209a3196291l8449258t1w424710966375-imagebucket-w5to82rav3xr.s3.us-east-1.amazonaws.com/clock.png

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>B6ADMDTBB1J6CFFN</RequestId>
  <HostId>jE7BxQM3s1jLvkNt4OZFCPUFujN3yIQorQy1pfrFzMORewtw+0eHUGq72lxaBs6MLTusyqLMnf0=</HostId>
</Error>
```

Then we performed a detailed security test by manipulating access controls in multiple layers. First, we disabled the bucket's public access blocks and enabled ACLs, then specifically made our clock.png object public. Despite these permissions changes, when attempting to access the encrypted object via its URL, we received an 'Invalid Argument' error instead of the previous 'Access Denied'. This demonstrates a crucial security principle: even if an object becomes publicly accessible, the encryption we implemented with our KMS key provides an additional security layer, requiring proper AWS authentication to decrypt the content.



c127209a3196291l8449258t1w424710966375-imagebucket-w5to82rav3xr.s3.us-east-1.amazonaws.com/clock.png

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Error>
   <Code>InvalidArgument</Code>
   <Message>Requests specifying Server Side Encryption with AWS KMS managed keys require AWS Signature Version 4.</Message>
   <ArgumentName>Authorization</ArgumentName>
   <ArgumentValue>null</ArgumentValue>
   <RequestId>2R6GAHS8VCZY2QCJ</RequestId>
   <HostId>ogzmbiUBcG1+y/zws6P9UEjqrcc9zEwVVL6dUEIgG2zthN80K8ICetLbx+NLaoNWYyRsbwdbM5E=</HostId>
 </Error>
```

Attempting signed access to the encrypted object:

We successfully accessed our encrypted object through the S3 console as an authenticated user. The URL revealed AWS's Signature Version 4 authentication elements, demonstrating how proper AWS credentials allow authorized decryption through the KMS key, while maintaining security through a complex request signing process.