Prashant Lakhera
Mar 6, 2019 · 4 min read · Listen

100 Days of DevOps — Day 24- How to encrypt S3 Bucket using KMS

Welcome to Day 24 of 100 Days of DevOps, Let continue our journey with terraform and on Day22 I give you a brief introduction about KMS, day 23 we learned how to encrypt EBS volume using KMS. Today I am going to demonstrate how to encrypt S3 bucket using KMS.

100 Days of DevOps — Day 23- How to encrypt EBS Volume using KMS
Welcome to Day 23 of 100 Days of DevOps, Let continue our journey with terraform and on Day22 I give you a brief…
medium.com

Before I am going to define how to encrypt S3 bucket using KMS, let get this thing out of the way S3 doesn't encrypt buckets, objects are encrypted and the setting are defined at an object level.

Earlier it was not possible to define encryption at a bucket level and there are many use cases to prevent uploads of unencrypted objects to an Amazon S3 bucket

To upload an object to S3, you use a Put request, regardless if called via the console, CLI, or SDK. The Put request looks similar to the following.

```
PUT /example-object HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Wed, 8 Jun 2016 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 11434
x-amz-meta-author: Janet
Expect: 100-continue
[11434 bytes of object data]
```

To encrypt an object at the time of upload, you need to add a header called x-amz-server-side-encryption to the request to tell S3 to encrypt the object using SSE-C, SSE-S3, or SSE-KMS. The following code example shows a Putrequest using SSE-S3.

```
PUT /example-object HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Wed, 8 Jun 2016 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 11434
x-amz-meta-author: Janet
Expect: 100-continue
x-amz-server-side-encryption: AES256
[11434 bytes of object data]
```

In order to enforce object encryption, create an S3 bucket policy that denies any S3 Put request that does not include the x-amz-server-side-encryption header. There are two possible values for the x-amz-server-side-encryption header: AES256, which tells S3 to use S3-managed keys, and aws:kms, which tells S3 to use AWS KMS–managed keys.

Bucket Policy will look like this

```
{
    "Version": "2012-10-17",
    "Id": "PutObjPolicy",
    "Statement": [
          {
                "Sid": "DenyIncorrectEncryptionHeader",
                "Effect": "Deny",
                "Principal": "*",
                "Action": "s3:PutObject",
                "Resource": "arn:aws:s3:::<bucket_name>/*",
                "Condition": {
                        "StringNotEquals": {
                               "s3:x-amz-server-side-encryption": "AES256"
                         }
                }
          },
          {
                "Sid": "DenyUnEncryptedObjectUploads",
                "Effect": "Deny",
                "Principal": "*",
                "Action": "s3:PutObject",
                "Resource": "arn:aws:s3:::<bucket_name>/*",
                "Condition": {
                        "Null": {
                               "s3:x-amz-server-side-encryption": true
                        }
               }
          }
    ]
}
```

s3_bucket_policy_with_encryption hosted with ❤ by GitHub — view raw

- *Now if we try to upload the object to S3 bucket without encryption it should fail*

```
$ aws s3 cp testingbucketencryption s3://mytestbuclet-198232055

upload failed: ./testingbucketencryption to s3://mytestbuclet-198232055/testingbucketencryption An error occurred (AccessDenied) when calling the PutObject operation: Access Denied
```

- *Let try it with encryption enabled*

```
$ aws s3 cp testingbucketencryption s3://mytestbuclet-198232055 --sse AES256

upload: ./testingbucketencryption to s3://mytestbuclet-198232055/testingbucketencryption
```

How to Prevent Uploads of Unencrypted Objects to Amazon S3 | Amazon Web Services
There are many use cases to prevent uploads of unencrypted objects to an Amazon S3 bucket, but the underlying objective…
aws.amazon.com

- *If you want to deny via UI i.e if you want everyone to access your bucket via https*

```
{
    "Version": "2012-10-17",
    "Id": "Policy1504640911349",
    "Statement": [
        {
            "Sid": "Stmt1504640908907",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

bucket_policy_to_restrict_https hosted with ❤ by GitHub — view raw

How to Use Bucket Policies and Apply Defense-in-Depth to Help Secure Your Amazon S3 Data | Amazon…
Amazon S3 provides comprehensive security and compliance capabilities that meet even the most stringent regulatory…
aws.amazon.com

There are two ways to use AWS KMS with AWS S3

- *Server Side Encryption(I am only going to discuss this)*
- *Client Side Encryption*

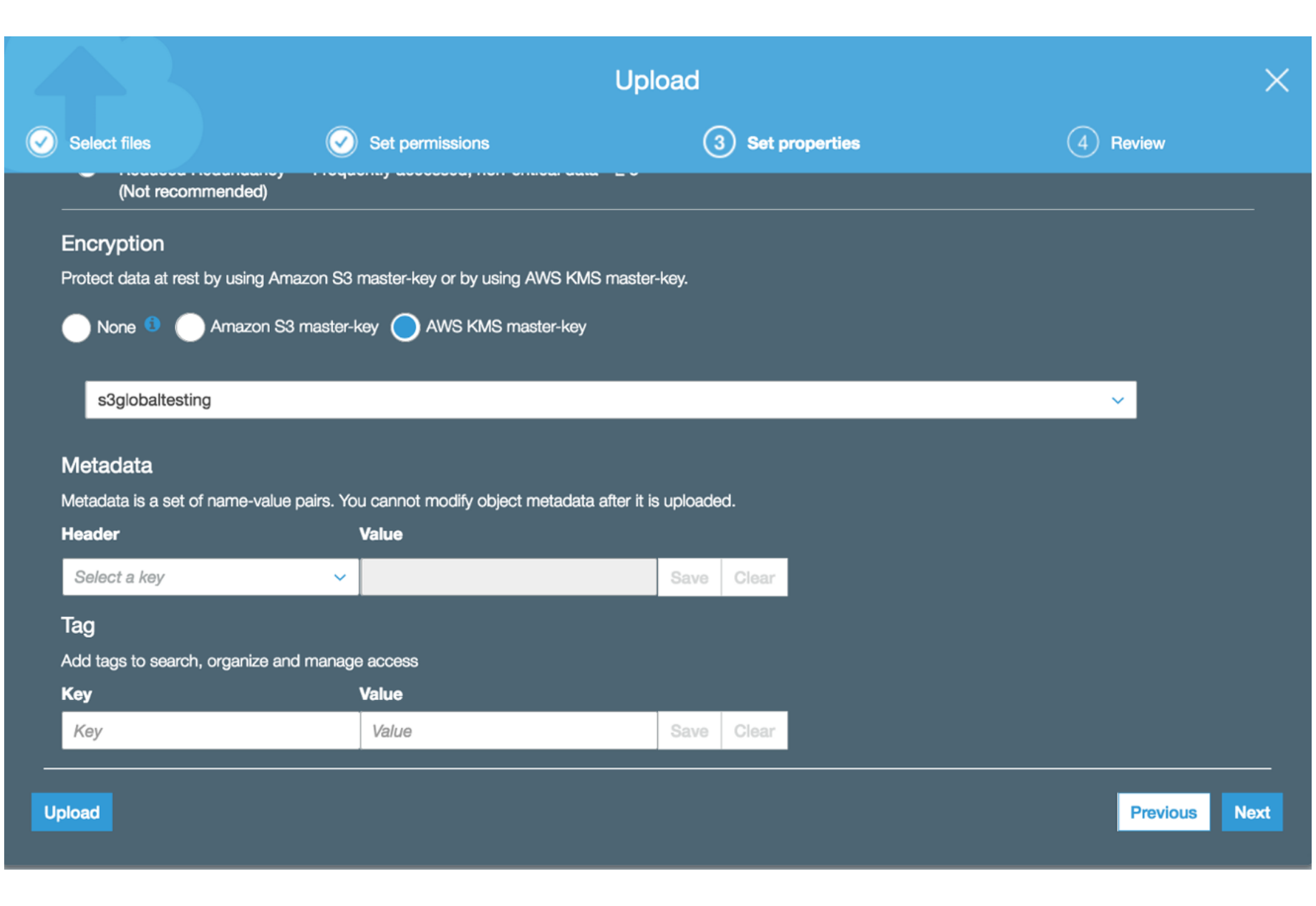
Server Side Encryption

We can protect data at rest in Amazon S3 using three different modes of server-side encryption: SSE-S3, SSE-C, or SSE-KMS

- *SSE-S3 requires that Amazon S3 manage the data and master encryption keys.*
- *SSE-C requires that you manage the encryption key.*
- *SSE-KMS requires that AWS manage the data key but you manage the master key in AWS KMS.*

> Go to S3 console --> https://s3.console.aws.amazon.com/s3 --> Particular bucket

Now when uploading any file to your bucket choose the KMS key



This is what happens behind the scene

- *Amazon S3 requests a plaintext data key and a copy of the key encrypted under the specified CMK.*
- *AWS KMS creates a data key, encrypts it by using the master key, and sends both the plaintext data key and the encrypted data key to Amazon S3.*
- *Amazon S3 encrypts the data using the data key and removes the plaintext key from memory as soon as possible after use.*
- *Amazon S3 stores the encrypted data key as metadata with the encrypted data.*

Now during decrypt operation

- *Amazon S3 sends the encrypted data key to AWS KMS.*
- *AWS KMS decrypts the key by using the appropriate master key and sends the plaintext key back to Amazon S3.*
- *Amazon S3 decrypts the ciphertext and removes the plaintext data key from memory as soon as possible.*

Terraform Code

```
provider "aws" {
  region = "us-west-2"
}
resource "aws_s3_bucket" "mybucket" {
  bucket = "mys3bucket-withkms-serverside-encryption"

  server_side_encryption_configuration {
    rule {
      apply_server_side_encryption_by_default {
        kms_master_key_id = "${var.kms_key}"
        sse_algorithm     = "aws:kms"
      }
    }
  }
}
```

s3_kms.tf hosted with ❤ by GitHub — view raw

GitHub Link

100daysofdevops/100daysofdevops
Contribute to 100daysofdevops/100daysofdevops development by creating an account on GitHub.
github.com

Looking forward from you guys to join this journey and spend a minimum an hour every day for the next 100 days on DevOps work and post your progress using any of the below medium.

- *Twitter: @100daysofdevops OR @lakhera2015*
- *Facebook: https://www.facebook.com/groups/795382630808645/*
- *Medium: https://medium.com/@devopslearning*
- *Slack: https://devops-myworld.slack.com/messages/CF41EFG49/*
- *GitHub Link:https://github.com/100daysofdevops*

Reference

100 Days of DevOps — Day 0
D-day is just one day away and finally, this is a continuation of the post(I posted a month earlier)
medium.com