Apply policy to restrict permissions on bucket

ronjon@ronjon-VirtualBox: ~/lab4

```
import json
import boto3
s3_client=boto3.client('s3')
BUCKET_NAME='23215183-cloudstorage'
def create_bucket_policy():
    bucket_policy = {
        "Version": "2012-10-17",
        "Statement": {
            "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
            "Effect": "DENY",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::23215183-cloudstorage/rootdir*",
            "Condition": {
            "StringNotLike": {
                "aws:username":"23215183@student.uwa.edu.au"
            }
            }
        }
    }

    policy_string = json.dumps(bucket_policy)

    response=s3_client.put_bucket_policy(
        Bucket=BUCKET_NAME,
        Policy=policy_string
    )
    response1 = s3_client.get_bucket_policy(Bucket = BUCKET_NAME)
    return response1['Policy']
print(create_bucket_policy())
```

```
ronjon@ronjon-VirtualBox:~/lab4$ python3 policy.py
{"Version":"2012-10-17","Statement":[{"Sid":"AllowAllS3ActionsInUserFolderFor
UserOnly","Effect":"Deny","Principal":"*","Action":"s3:*","Resource":"arn:aws
:s3:::23215183-cloudstorage/rootdir*","Condition":{"StringNotLike":{"aws:user
name":"23215183@student.uwa.edu.au"}}}]}
```

Code for create KMS:

```
import boto3
import json

client=boto3.client('kms')
policy={
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
```

```
  "Principal": {
    "AWS": "arn:aws:iam::523265914192:root"
  },
  "Action": "kms:*",
  "Resource": "*"
},
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::523265914192:user/23215183@student.uwa.edu.au"
  },
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:TagResource",
    "kms:UntagResource",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
},
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::523265914192:user/23215183@student.uwa.edu.au"
  },
  "Action": [
```

```
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::523265914192:user/23215183@student.uwa.edu.au"
      },
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    }
  ]
}
response=client.create_key(Policy=json.dumps(policy))
#print(response)
#print(client.list_keys())
response1= client.create_alias(
        AliasName='alias/23215183',
        TargetKeyId=response['KeyMetadata']['KeyId']
        )
print("key_Id_is", response['KeyMetadata']['KeyId'])
print("key_region_is:", response['KeyMetadata']['Arn'])
```

output:

```
ronjon@ronjon-VirtualBox:~/lab4$ python3 create_KMS.py
key_Id_is ab447b7c-fc9a-40b0-a1eb-8956223ec0cd
key_region_is: arn:aws:kms:ap-southeast-2:523265914192:key/ab447b7c-fc9a-40b0
-a1eb-8956223ec0cd
```

Code for new KMS:

```
ronjon@ronjon-VirtualBox: ~/lab4
import boto3
import json

client = boto3.client('kms')

response = client.get_key_policy(
    KeyId = 'ab447b7c-fc9a-40b0-a1eb-8956223ec0cd',
    PolicyName = 'default'
)

print(response)
```

Output:

```
ronjon@ronjon-VirtualBox:~/lab4$ python3 create_New_KMS.py
{'Policy': '{\n  "Version" : "2012-10-17",\n  "Id" : "key-consolepolicy-3",\n  "Statement" : [ {\n    "Sid" : "Enable IAM User Permissions",\n    "Effect" : "Allow",\n    "Principal" : {\n      "AWS" : "
arn:aws:iam::523265914192:root"\n    },\n    "Action" : "kms:*",\n    "Resource" : "*"\n  }, {\n    "Sid" : "Allow access for Key Administrators",\n    "Effect" : "Allow",\n    "Principal" : {\n      "AW
S" : "arn:aws:iam::523265914192:user/23215183@student.uwa.edu.au"\n    },\n    "Action" : [ "kms:Create*", "kms:Describe*", "kms:Enable*", "kms:List*", "kms:Put*", "kms:Update*", "kms:Revoke*", "kms:Disa
ble*", "kms:Get*", "kms:Delete*", "kms:TagResource", "kms:UntagResource", "kms:ScheduleKeyDeletion", "kms:CancelKeyDeletion" ],\n    "Resource" : "*"\n  }, {\n    "Sid" : "Allow use of the key",\n    "Ef
fect" : "Allow",\n    "Principal" : {\n      "AWS" : "arn:aws:iam::523265914192:user/23215183@student.uwa.edu.au"\n    },\n    "Action" : [ "kms:Encrypt", "kms:Decrypt", "kms:ReEncrypt*", "kms:GenerateDa
taKey*", "kms:DescribeKey" ],\n    "Resource" : "*"\n  }, {\n    "Sid" : "Allow attachment of persistent resources",\n    "Effect" : "Allow",\n    "Principal" : {\n      "AWS" : "arn:aws:iam::523265914192:user/23215183@student.uwa.edu.au"\n    },\n    "Action" : [ "kms:CreateGrant", "kms:ListGrants", "kms:RevokeGrant" ],\n    "Resource" : "*",\n    "Condition" : {\n      "Bool" : {\n        "kms:GrantIsForAWSResource" : "true"\n      }\n    }\n  } ]\n}', 'ResponseMetadata': {'RequestId': 'a85ec189-cce5-4324-ade2-031b0310ce09', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'a85ec189-cce5-4324-ade2-031b0310ce09', 'cache-control': 'no-cache, no-store, must-revalidate, private', 'expires': '0', 'pragma': 'no-cache', 'date': 'Mon, 05 Sep 2022 03:15:57 GMT', 'content-type': 'application/x-amz-json-1.1', 'content-length': '1616'}, 'RetryAttempts': 0}}
ronjon@ronjon-VirtualBox:~/lab4$
```

AES Encryption using KMS

Code for encryption and download the file:

 # REFERENCE: https://hands-on.cloud/working-with-kms-in-python-using-boto3/#How-to-encrypt-files-using-KMS-and-Boto3-in-Python

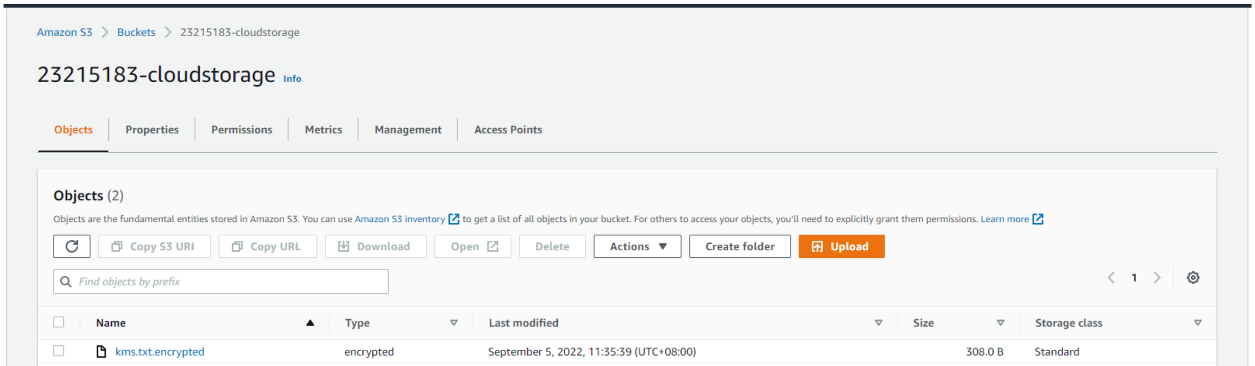
import boto3
import base64
from cryptography.fernet import Fernet

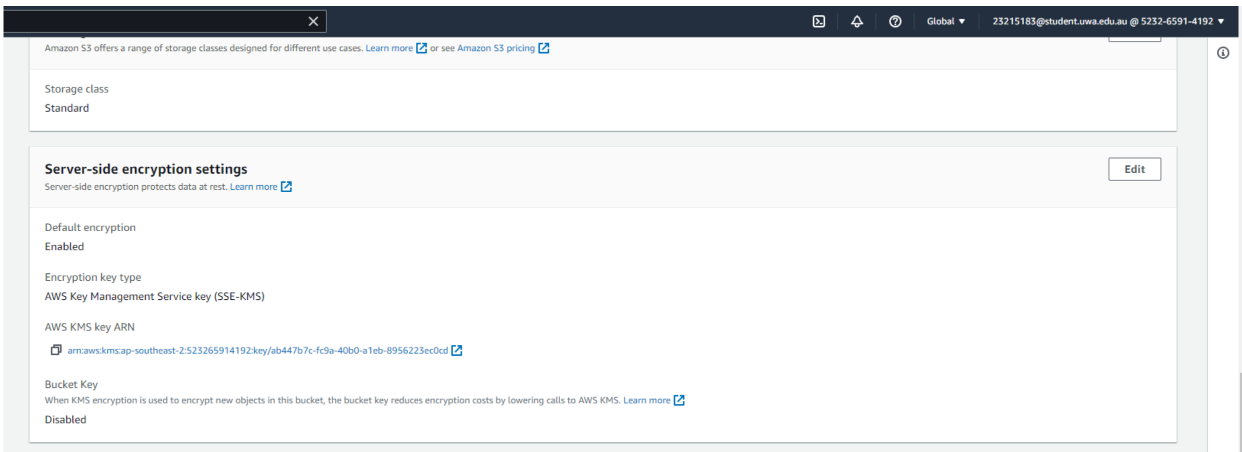
client = boto3.client('kms')

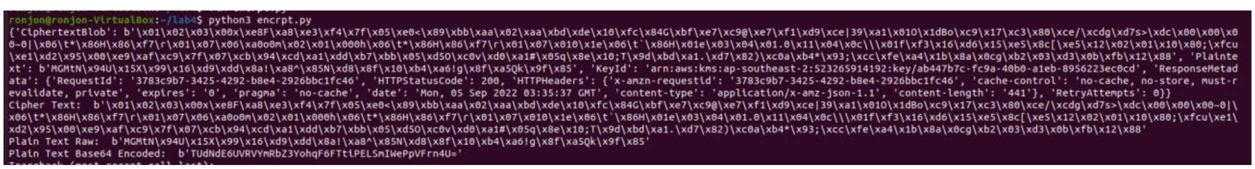
```
response = client.generate_data_key(
   KeyId = 'ab447b7c-fc9a-40b0-a1eb-8956223ec0cd',
   KeySpec = 'AES_256'
)
print(response)
print("Cipher Text: ", response['CiphertextBlob'])
print("Plain Text Raw: ", response['Plaintext'])
print("Plain Text Base64 Encoded: ", base64.b64encode(response['Plaintext']))

data_key_encrypted = response['CiphertextBlob']
data_key_plaintext = base64.b64encode(response['Plaintext'])


filename = 'kms.txt'
# Read the entire file into memory
with open(filename, 'rb') as file:
   file_contents = file.read()
# Encrypt the file
f = Fernet(data_key_plaintext)
file_contents_encrypted = f.encrypt(file_contents)
# Write the encrypted data key and encrypted file contents together
with open(filename + '.encrypted', 'wb') as file_encrypted:
   file_encrypted.write(
      len(data_key_encrypted).to_bytes(4,
                                       byteorder='big'))
   file_encrypted.write(data_key_encrypted)
   file_encrypted.write(file_contents_encrypted)

ROOT_DIR = '.'
ROOT_S3_DIR = '23215183-cloudstorage' #Bucket Name
s3 = boto3.client("s3")
bucket_config = {'LocationConstraint': 'ap-southeast-2'}
s3.upload_file('./kms.txt.encrypted', ROOT_S3_DIR, 'kms.txt.encrypted',
            ExtraArgs = {"ServerSideEncryption": "aws:kms",
"SSEKMSKeyId":"arn:aws:kms:ap-southeast-2:523265914192:key/ab447b7c-
fc9a-40b0-a1eb-8956223ec0cd"})
```

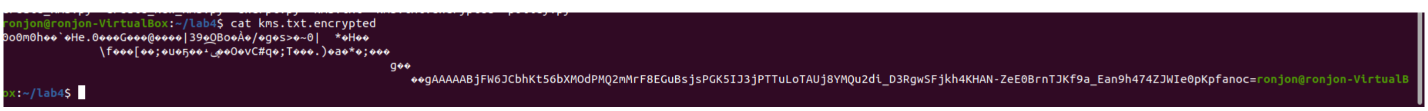
```
s3_resource = boto3.resource('s3')
bucket = s3_resource.Bucket('23215183-cloudstorage')
bucket.download_file('kms.txt.encrypted', '/home/lab4/kms.txt.encrypted')
encrypted_downloaded_file = 'kms.txt.encrypted'

# Decrypt the encrypted data key
def decrypt_data_key(data_key_encrypted):
    response = client.decrypt(CiphertextBlob=data_key_encrypted)
    # Return plaintext base64-encoded binary data key
    return base64.b64encode((response['Plaintext']))
# Decrypt the encrypted file
def decrypt_file(filename):
    # Read the encrypted file into memory
    with open(filename, 'rb') as file_encrypted:
        file_contents_encrypted = file_encrypted.read()
    data_key_encrypted_len = int.from_bytes(file_contents_encrypted[:4],
                                            byteorder='big') \
                             + 4
    data_key_encrypted = file_contents_encrypted[
        4:data_key_encrypted_len]
    # Decrypt the data key before using it
    data_key_plaintext = decrypt_data_key(data_key_encrypted)
    # Decrypt the rest of the file
    f = Fernet(data_key_plaintext)
    file_contents_decrypted =
f.decrypt(file_contents_encrypted[data_key_encrypted_len:])
    # Write the decrypted file contents
    with open(filename + '.decrypted' , 'wb') as file_decrypted:
        file_decrypted.write(file_contents_decrypted)
# Call the function
decrypt_file(encrypted_downloaded_file)
```

output:









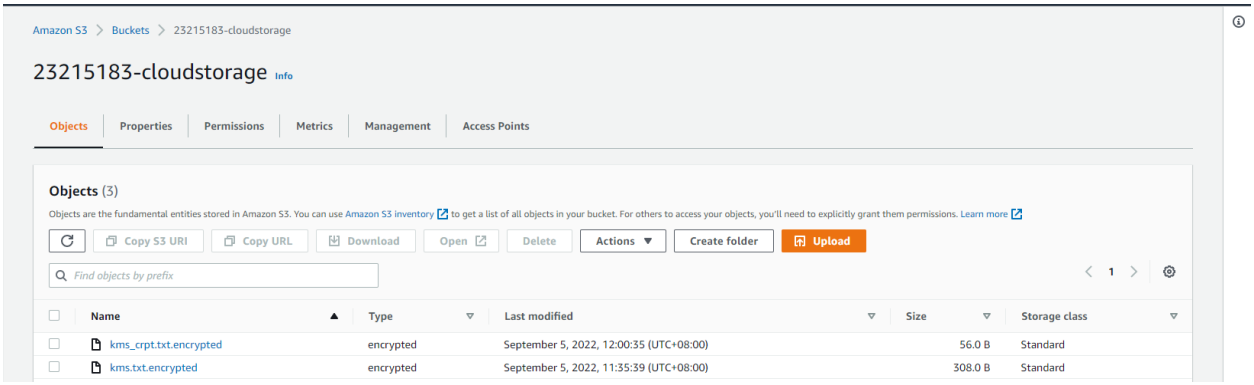
AES Encryption and Decryption using local python library pycryptodome

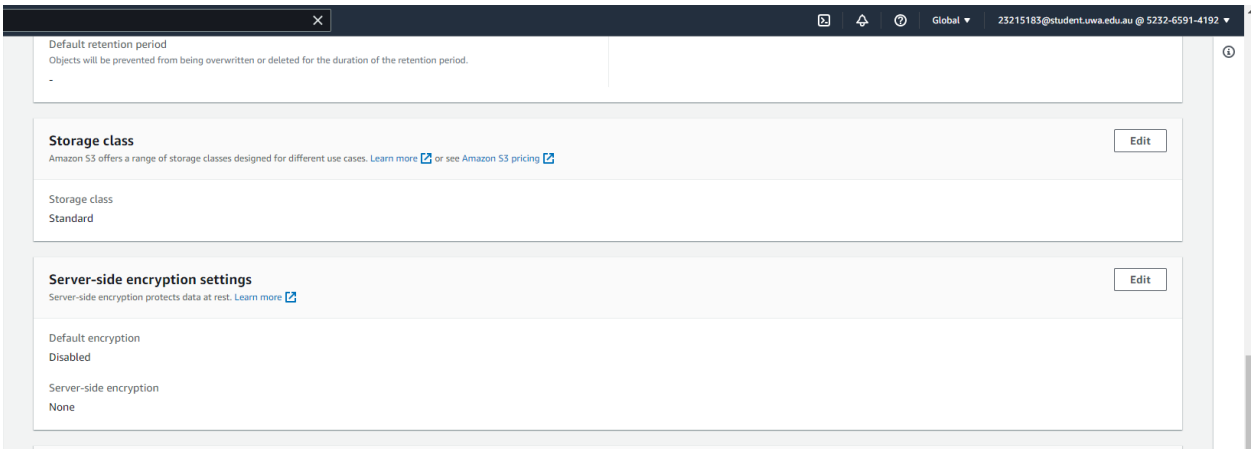
Code:

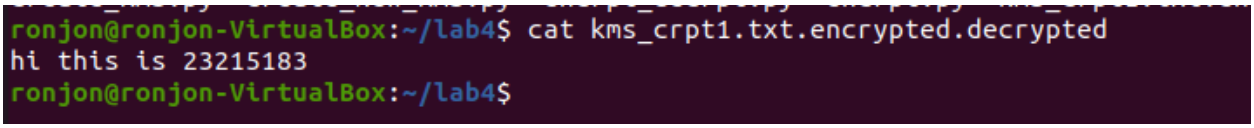
import os, random, struct
from Crypto.Cipher import AES

```
from Crypto import Random
import boto3
import base64
import hashlib

BLOCK_SIZE = 16
CHUNK_SIZE = 64 * 1024

def encrypt_file(password, in_filename, out_filename):

    key = hashlib.sha256(password.encode("utf-8")).digest()

    iv = Random.new().read(AES.block_size)
    encryptor = AES.new(key, AES.MODE_CBC, iv)
    filesize = os.path.getsize(in_filename)

    with open(in_filename, 'rb') as infile:
        with open(out_filename, 'wb') as outfile:
            outfile.write(struct.pack('<Q', filesize))
            outfile.write(iv)

            while True:
                chunk = infile.read(CHUNK_SIZE)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    chunk += ' '.encode("utf-8") * (16 - len(chunk) % 16)

                outfile.write(encryptor.encrypt(chunk))


def decrypt_file(password, in_filename, out_filename):

    key = hashlib.sha256(password.encode("utf-8")).digest()

    with open(in_filename, 'rb') as infile:
        origsize = struct.unpack('<Q', infile.read(struct.calcsize('Q')))[0]
        iv = infile.read(16)
        decryptor = AES.new(key, AES.MODE_CBC, iv)
```

```
        with open(out_filename, 'wb') as outfile:
            while True:
                chunk = infile.read(CHUNK_SIZE)
                if len(chunk) == 0:
                    break
                outfile.write(decryptor.decrypt(chunk))

            outfile.truncate(origsize)


password = 'kitty and the kat'

encrypt_file(password,"kms_crpt.txt", out_filename="kms_crpt.txt.encrypted")

ROOT_DIR = '.'
ROOT_S3_DIR = '23215183-cloudstorage' #Bucket Name

s3 = boto3.client("s3")
bucket_config = {'LocationConstraint': 'ap-southeast-2'}

s3.upload_file('./kms_crpt.txt.encrypted', ROOT_S3_DIR, 'kms_crpt.txt.encrypted')

# Download the uploaded encrypted file
s3_resource = boto3.resource('s3')
bucket = s3_resource.Bucket('23215183-cloudstorage')

bucket.download_file('kms_crpt.txt.encrypted',
'/home/lab4/kms_crpt.txt.encrypted')

encrypted_downloaded_file = 'kms_crpt.txt.encrypted'

# Decrypt the downloaded file
decrypt_file(password, "kms_crpt.txt.encrypted",
out_filename="kms_crpt.txt.encrypted.decrypted")
```


Amazon S3 > Buckets > 23215183-cloudstorage
23215183-cloudstorage Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (3)
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
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class
kms_crpt.txt.encrypted
encrypted
September 5, 2022, 12:00:35 (UTC+08:00)
56.0 B
Standard
kms.txt.encrypted
encrypted
September 5, 2022, 11:35:39 (UTC+08:00)
308.0 B
Standard


Global
23215183@student.uwa.edu.au @ 5232-6591-4192
Default retention period
Objects will be prevented from being overwritten or deleted for the duration of the retention period.
-
Storage class
Amazon S3 offers a range of storage classes designed for different use cases. Learn more or see Amazon S3 pricing
Edit
Storage class
Standard
Server-side encryption settings
Server-side encryption protects data at rest. Learn more
Edit
Default encryption
Disabled
Server-side encryption
None


ronjon@ronjon-VirtualBox:~/lab4$ cat kms_crpt1.txt.encrypted.decrypted
hi this is 23215183
ronjon@ronjon-VirtualBox:~/lab4$