Practical: S3 blob storage service (Python)

Table of Contents

S3 is AWS's blob storage service. In this practical you will learn how to read and write data with S3 using Python and the `boto3` SDK.

Prerequisites and references

- **AWS S3 documentation (https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html)**
- **Boto3 documentation for S3 (https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html)**
- Complete code for this prac: s3demo_python.py

Creating an S3 bucket

S3 uses the concept of a *bucket* to organise objects. Objects are stored within a single bucket, and it is typical for an application to have multiple buckets to organise objects according to their usage.

It is straightforward to create buckets and perform other S3 operations using the AWS console. In this practical we will concentrate on using the Python API.

1. Get authenticated

You can do this practical on an EC2 instance or on your local development environment, provided you have authentication set up.

- If you are using an EC2 instance, ensure that Python 3 is installed, that you are the `ubuntu` user, etc. as in previous practicals.
- Refer to the previous practical for details on setting up authentication for the AWS SDK for Python (`boto3`).

2. Create the Python app and install packages

Each AWS service is accessed via the `boto3` client. To get set up, run these commands:

```
mkdir s3demo_python
cd s3demo_python
python3 -m venv venv
source venv/bin/activate
pip install boto3
```

3. Write the code

In the `s3demo_python` directory, create `s3_demo.py` with the following contents, changing `n1234567` to your own username:

```
import boto3
from botocore.exceptions import ClientError

bucket_name = 'n1234567-test'  # Change to your own username
region = 'ap-southeast-2'

s3_client = boto3.client('s3', region_name=region)

# Create bucket
try:
    response = s3_client.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={'LocationC
onstraint': region})
    print('Bucket created at:', response.get('Location'))
except ClientError as e:
    print(e)
```

Be sure to change the username in the bucket's name to your own, as buckets need to have unique names.

4. Run the app

- Run `python3 s3_demo.py`

If the bucket already exists, you will see an error, but you can proceed to the next steps.

Tagging an S3 bucket

Like EC2 instances, we require that you tag all S3 buckets that you create. This can be done with the SDK.

1. Add code for tagging

Add the following to your script after bucket creation:

```
qut_username = 'n1234567@qut.edu.au'  # Change to your own
purpose = 'prac'

try:
    response = s3_client.put_bucket_tagging(
        Bucket=bucket_name,
        Tagging={
            'TagSet': [
                {'Key': 'qut-username', 'Value': qut_username},
                {'Key': 'purpose', 'Value': purpose}
            ]
        }
    )
    print('Tagging response:', response)
except ClientError as e:
    print(e)
```

2. Run the app

- `python3 s3_demo.py`

Writing to an S3 bucket

We can write to a bucket by creating (or updating) an object. For this we need a key, which is the object's name, and the data for the object.

Add the following to your script:

```python
object_key = 'myAwesomeObjectKey'
object_value = 'This could be just about anything.'

try:
    response = s3_client.put_object(Bucket=bucket_name, Key=object_key, Body=object_value)
    print('PutObject response:', response)
except ClientError as e:
    print(e)
```

Reading from an S3 bucket

Add the following to your script:

```python
try:
    response = s3_client.get_object(Bucket=bucket_name, Key=object_key)
    body = response['Body'].read().decode()
    print('Object value:', body)
except ClientError as e:
    print(e)
```

Accessing S3 with Pre-signed URLs

Pre-signed URLs allow clients to access S3 objects directly for a limited time.

Add the following to your script:

```python
try:
    presigned_url = s3_client.generate_presigned_url(
        'get_object',
        Params={'Bucket': bucket_name, 'Key': object_key},
        ExpiresIn=3600
    )
    print('Pre-signed URL:', presigned_url)
except ClientError as e:
    print(e)
```

You can use this URL in your browser or with `requests.get(presigned_url)` to retrieve the object.

Delete an object

If you no longer need an object you can delete it.

Add the following to your script:

```python
try:
    response = s3_client.delete_object(Bucket=bucket_name, Key=object_key)
    body = response
    print('Delete_object response:', response)
except ClientError as e:
    print(e)
```

Delete your bucket

After you are done, please delete your bucket. You can do this through the AWS console or with the SDK:

```python
try:
    s3_client.delete_bucket(Bucket=bucket_name)
    print('Bucket deleted.')
except ClientError as e:
    print(e)
```