Module 8
Object Storage with S3

In prior modules you used S3 to demonstrate some other concept (e.g., in Module 6 you created an S3 bucket for the purpose of testing that you can connect to AWS and use some API call). In this module we look at object storage in the cloud in more detail. The AWS service that provides object storage is S3. It is one of many other AWS storage-related services such as EFS, S3 Glacier, and AWS Backup.

Objectives of this module

a) Continue practice using the AWS SDKs.
b) Learn about the object storage technology of AWS: S3.

Overview

There are 3 types of storage architectures: file system storage, block storage, and object storage. The one that users are most familiar with is file system storage, where files are stored in a hierarchy.

Block storage is another storage architecture that is more low-level than file storage. In this case data is stored as blocks within sectors and tracks of a hard-disk or in flash-based cells. In block storage, each block has an address. A file, depending on how big it is, can be saved in one block or multiple blocks. One advantage the granularity of block storage brings is that one can do incremental update. That is, a minor change require only that the affected block be updated (not the entire file which might be using several blocks).

Because block storage is low level, one can build any file system on top of it, where high-level file system operations are translated to low-level block operations. This is what an operating system file system does: It abstracts low level block storage (hard disks or flash cells) into high-level files with names that are easier for end users to remember and use. Some mission critical systems such as databases where speed is of utmost importance bypass the file system and store data directly in blocks. Block storage does not include metadata.

Object storage stores data as objects. Objects have a unique identifier that identifies the object and a variable amount of metadata can be associated with each object. One can think of hundreds of scenarios where metadata can be of great use for applications to make sense of the data stored. Unlike file system storage, there is no concept of hierarchy in object storage. Object storage is more scalable than other types of storage. Expanding the storage capacity translates to simply adding more nodes to the storage cluster. One disadvantage of block storage is that updates are made at the object level. For example, if you have a 100 MB file and you need to make change to a single word, the entire object has to be replaced. That's because the application using object storage does not have access to any low-level storage. Instead it saves and retrieves data at the level of the object.

The AWS object storage service is S3, which allows you to store and retrieve any amount of data on the Internet. Like any other service, you can do so via the AWS Console, the CLI, or the API from any Internet-connected application. S3 stores objects in buckets, which can be thought of as a container of objects. An object consists of a file and additional metadata that you can attach to the file. Security permissions can be set on the bucket itself or on individual objects in the bucket.

Because S3 objects can be accessed as URIs, bucket names must be unique across the globe.
Do this small exercise:

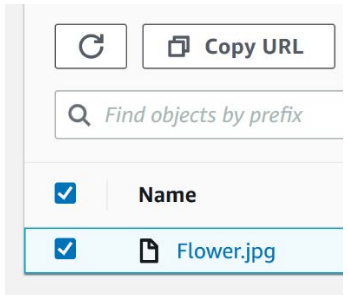
1. Start the lab, go to the AWS console, then go to the S3 service.
2. Create a bucket in S3 and make it public. If you already have one to use, make sure it is public:

    a. Go to S3 and click the bucket.
    b. Go to the **Permissions** tab. Click the **Edit** button in the **Block public access** section.
    c. Uncheck the **Block all public access** checkbox.
    d. Click the **Save changes** button.
    e. Confirm and click the **Confirm** button.

3. Upload the enclosed image (flower.jpg) file to the bucket.

4. In the **Permissions** section, choose option **Grant public-read access**.
   Confirm the warning and click the **Upload** button.

5. Once you upload the flower, go back to the bucket and Flower.jpg should be listed. Check the checkbox next to it and click the **Copy URL** button.



6. Open another tab in your browser and paste the link. You should see the image.

Because the bucket name makes part of the URL string, there cannot be two buckets with identical names. Although not necessarily required, AWS has guidelines and best practices on how to name buckets. You can find that in the S3 online documentations. Also note that you cannot create a bucket inside another bucket.

Although S3 allows you to create a folders structure hierarchy (similar to how you do it in a file system on a PC), that folder structure is a logical construct that gives the illusion of a hierarchy. As described on page 1, object storage does not have a hierarchy. We will do an example to demonstrate this.

When you create a bucket you specify the region where AWS stores your bucket. The choice of region may depend on say minimizing latency (e.g., if 90% of your app's users are based in the US, it doesn't make sense to create your bucket in Ireland or Tokyo. You reduce latency if you store it in Virginia or Ohio). Other reasons of region choice may depend on government regulations.

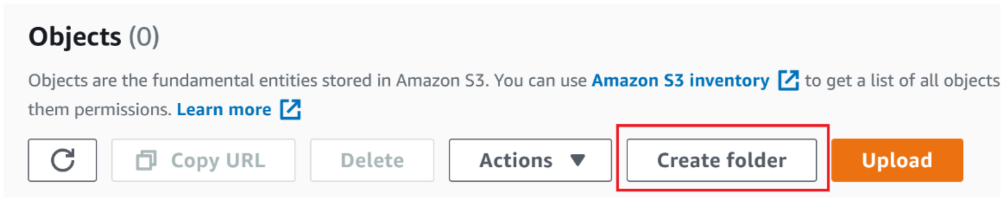
A very important feature of S3 is notifications. S3 can publish notifications when some event happens (e.g., an example event is when say a file is added to a bucket). The notification that is published can call another AWS service (e.g., a Lambda function). With notifications we can start doing interesting things without having code do constant polling to check if some file has arrived. Instead S3 notifies code that some event in S3 has happened. It is like event-driven programming in the cloud. **We will do code examples when we cover AWS Lambda, a compute service, in a future module**.

Finally, you might be curious about the cost to store objects in S3. Browse this page to get an idea of how much it costs per GB.

S3 also has storage classes (or types) that affect cost (e.g., **S3 standard**, **S3 intelligent tiering**, **S3 standard – infrequent access**, **S3 Glacier**, etc.). Read about these storage classes in the Storage tab of the link above. For example, files that are not frequently accessed can be stored in a lower cost type (such as S3 Glacier). Retrieval time increases with lower-priced storage types. For example, notice with S3 glacier that it takes anywhere from 1 minute to 12 hours to retrieve a file. On the other hand, it takes only milliseconds to retrieve files if the S3 Standard - Infrequent Access type is used. What type you choose depend on the file being stored and its frequency of access.

Create Folders Hierarchy in S3

1. Login to the AWS Console and go to the S3 service.
2. Under your S3 bucket, create some folders hierarchy and add some dummy files to them. You create a folder by clicking the Create folder button in the Overview tab.


Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects
them permissions. Learn more

For instance you can create something like the folders hierarchy of Figure 1 (folders appear in orange and files in blue).

Your bucket

- Images
  - Flower.jpg
  - House.jpg
- Data
  - sensors.txt
  - Archived
    - sensors.txt
- config.xml
- todo.txt

Figure 1: folders hierarchy under some bucket (orange text are folders while blue text are files).

3. Although the above appears as some sort of folders hierarchy (similar to what you can do in an OS file system), internally it is not. In S3, the above is really like this:

   Your bucket

   - Images/
   - Images/Flower.jpg
   - Images/House.jpg
   - Data/
   - Data/Archived
   - Data/sensors.txt
   - Data/Archived/sensorts.txt
   - config.xml
   - todo.txt

   There is only one hierarchy of items under a given bucket. And each item is an S3 object with the following 3 characteristics:

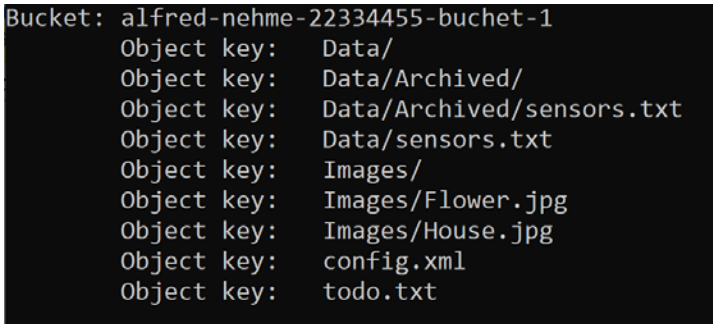
   a) Data: the file itself.
   b) Key: uniquely identifies an object in a bucket. The green text above are the keys of the objects. A prefix and the "/" delimiter are used to give a logical view that looks as if there is a hierarchy of folders.
   c) Metadata: key/value pairs that you can associate with a given object. There are system metadata and user-defined metadata. User-defined metadata keys must begin with "x-amz-meta-". For example, one can create a custom metadata to tag if a file is a configuration file: x-amz-meta-is-config, and set its value to say 1.
   d) Tags: key/value custom data.
   e) Objects can also be encrypted at rest – that is while saved in S3.

Print the keys of all objects in your bucket

1. For this exercise, you don't need to create a new project. You can use the same project of Module 6. You just need to add a few lines of codes.

   Create a folders hierarchy similar to Figure 1 (dummy files for you to use are provided in the enclosed Module8TestFiles folder).

2. Modify the program of Module 6 to print the keys of all objects in your bucket. An example output should look like the one in Figure 2.

   ```
   Bucket: alfred-nehme-22334455-buchet-1
           Object key:    Data/
           Object key:    Data/Archived/
           Object key:    Data/Archived/sensors.txt
           Object key:    Data/sensors.txt
           Object key:    Images/
           Object key:    Images/Flower.jpg
           Object key:    Images/House.jpg
           Object key:    config.xml
           Object key:    todo.txt
   ```

   Figure 2: keys of objects in an S3 bucket.

   What you should notice from your code is that you can accomplish the above without writing nested loops or a recursive function (what you would need to do if you were to enumerate files in an OS folder hierarchy). The reason is because there is no real hierarchy. The hierarchy is only a logical view that S3 is maintaining for you in order to simulate a folder hierarchy.

3. Read about S3 Notifications here.

What to submit:

Complete **Module8-Quiz** after you finish this module.