Simple Storage Service (S3)

Dr Peadar Grant

October 8, 2021

1 Object storage concepts

Object storage is comprised of **buckets** situated within **regions** that hold **objects** identified by **keys**. In more detail:

Bucket is a high-level grouping of related objects. Similar concept to a database on a DBMS or a file share on a NAS. Buckets are identified by a name:

- Note that bucket names must be globally unique across AWS.

Region: each bucket is associated with a specific region. Data never leaves that region unless manually transferred to another region.

Objects are pieces of data, aka files, stored in S3. These can be broken down into:

Data: the actual content, e.g. the image, music file, text document. These are said to be *opaque* to S3, meaning that S3 doesn't interpret or process the content. It only stores and retrieves it for you.

Metadata describes the object. It consists of a series of key-value pairs. Some it is default and system-defined, such as the date last modified. You can define any other metadata tags you want.

Key: a key uniquely identifies each object within a bucket. Each object has exactly one key mapping to it. An object is uniquely identified within S3 by its bucket name, its key and optionally its version number.

1.1 S3 URIs

Buckets are identified by a so-called S3 URI in the form `s3://bucket-name-here`.

Objects / files within buckets are identified in the form:

- `s3://bucket-name/file1.pdf`
- `s3://bucket-name/file2.pdf`
- `s3://bucket-name/images/img1.jpg`
- `s3://bucket-name/images/img2.png`

1.2 Lack of hierarchy

Unlike filesystems, S3's namespace is flat. It does *not* have any concept of hierarchy. There are no folders as such.

However, we can infer hierarchy using common prefixes. Additionally, the S3 console and command-line tools map the slash character as a pseudo directory separator.

1.3 Access

S3 storage can be accessed in a number of ways:

AWS console using point-and-click file operations.

AWS CLI allowing S3 operations from the command-line. Can be combined, scripted, scheduled etc easily.

AWS SDK where an application uses the AWS SDK library to directly read/write objects to/from S3 as required.

The AWS CLI and SDK make use of the web-service endpoint that provides a URL for any object in S3. This can be used directly by an application in a similar way to using an API, but most applications make use either of the CLI or the SDK. We will focus on the web console.

2 S3API Operations

The necessary commands are under the `s3` and `s3api` sub-commands. The `s3` commands are a simplified coherent set of commands that utilise the more general `s3api` command set. See AWS blog post about S3 and S3API. We will work mainly with `s3api`.

The `s3api` commands are similar to the AWS commands we have met so far, in that they return JSON which can be used to create PowerShell objects using the `ConvertFrom-Json` cmdlet. Remember to use the inbuilt help as you try out these commands.

Assuming `my-bkt1` in region `us-east-1` the basic commands are:

2.1 Bucket creation

```
aws s3api create-bucket
--bucket my-bkt1
--region us-east-1
--create-bucket-configuration LocationConstraint=us-east-1
```

Remember bucket names must be globally unique across all AWS users and accounts.

2.2 Listing buckets

```
aws s3api list-buckets
```

2.3 PUTting file into S3

An object (file) is placed into a specified S3 bucket identified by a key (file name).

```
aws s3api put-object
--bucket my-bkt1
--key agenda.txt
--body 2020_12_04_agenda.txt
# key = the name S3 will use for object
# body = the file name on your local system
```

2.4 Listing bucket contents

```
aws s3api list-objects --bucket my-bkt1
```

2.5 GETting a file from S3

An object is requested from a specified S3 bucket identified by a key.

```
aws s3api get-object
--bucket my-bkt1
--key agenda.txt agendadl.txt
# key = name S3 uses for object
# output file name has no parameter switch
# info prints to terminal, file gets saved
```

Command will return error if file doesn't exist.

2.6 DELETE a file

The object with specified key in specified bucket is deleted.

```
aws s3api delete-object
--bucket my-bkt1
--key agenda.txt
```

The `delete-object` operation is idempotent and returns nothing.

2.7 Delete bucket

```
aws s3api delete-bucket --bucket my-bucket
```

3 Resource naming

On a cloud service, we need ways to unambiguously identify the resources we provision. On AWS, we use Amazon Resource Names, or ARNs. ARNs follow a common pattern:

```
arn:partition:service:region:account-id:resource-id
arn:partition:service:region:account-id:resource-type/resource-id
arn:partition:service:region:account-id:resource-type:resource-id
```

Partition is normally just `aws` but can vary in certain regions.

Service identifies the service. Today we will be using `s3`.

Region specifies the region that the resource is located in. Some resources don't need a region to be specified.

Account-id is the **ID number** of the account that the resource is owned by. This is different to the login name for the account.

Resource specifies the specific resource. Sometimes the resource requires a type or hierarchy. This will vary according to the specific services.

ARNs appear in many places where one resource needs to make use of another, or where we want to control access to a specific resource.

3.1 Getting account ID number

Each AWS account has a numeric ID number. It can be found under Account Information in the Console but easiest via the CLI:

```
# basic command
aws sts get-caller-identity

# get ID as PS variable
$AccountId = (aws sts get-caller-identity
              | ConvertFrom-Json).Account
```

3.2 S3 bucket ARN format

ARNs for S3 buckets take the format `arn:aws:s3:::bucket-name` for the bucket `bucket-name`

4 Usage scenarios

Using S3 solely as a cloud-based file store offers a number of possibilities. With appropriate scripting (and some permissions management) you could easily implement:

- Managed file-transfer as a replacement for FTP, SFTP and similar services:
  - Don's pizza has large plasma menu boards in each location. Menu images are stored in an S3 bucket `dons-menu-images`. Graphic designers upload new menu image files using a script that calls the `s3api put-object`. Each menu board is driven by a raspberry pi computer. It downloads the daily menu images using a script that calls the `s3api get-object`.
- A creative professional sends large files to clients that are too big for e-mail. They have a script that uploads a file to S3, creating a link for the client to use to download the image.
- A retail store has a central host computer at its HQ that manages pricing information for the chain. The stores each have a server that controls the tills in that shop.
  - The central host places the master price file. Later the store servers download this file from S3.
  - The store servers upload sales transaction data for each day into S3. The central host collects these files each day from S3 later on.

5 Exercise

5.1 Basic S3 usage

Create an S3 bucket, populate it with some files, ensure that you can list and delete them. Confirm that you can list the S3 buckets you have.

6 Static web hosting

S3 includes a simple inbuilt static web hosting system. Once it has been enabled, the website will appear at the URL shown in the console. This will look something like:

`http://example-bucket.s3-website-eu-west-1.amazonaws.com`

The index document support allows a particular document to be returned when no key is specified. This *only* works at the root/top level (remember S3 has no concept of a folder!) Other options you can explore include traffic logging, custom error document support and redirect configuration.

6.1 Permissions

The website endpoint technically is allowing users other than the logged in account holder to get content from the bucket. The bucket permissions need to be updated by inserting a *policy* containing a *statement* allowing them to do so. If this isn't done, a forbidden error will occur. Here is a bucket policy for a bucket named `example-buc` to allow static web hosting:

```
{
  "Version":"2012-10-17",
  "Statement":[{
        "Sid":"PublicReadGetObject",
        "Effect":"Allow",
          "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::example-bucket/*"
      ]
    }
  ]
}
```

Looking at the above policy, it contains one statement. Breaking it down:

Sid names/labels the statement.

Effect is either to `Allow` or `Deny`

Principal is the AWS user that the statement is to apply to. The asterisk `*` means any user, including an unknown one.

Action is a *list* of actions to allow. Here we allow the `s3:GetObject` action.

Resource is a *list* of the AWS resources that this policy is to apply to.

6.2 Limitations

S3 web hosting is very rudimentary (can't support SSL, custom domains). CloudFront significantly expands web hosting capabilities (see later).

In this lab we will use the S3 object store to host a static website without the need for using a web server (or even a VM).

7 Website content

You will need a basic static website:

- If you can't get one, go to `https://github.com/peadargrant/test_static_website` and get the site by cloning the Git repository.

- Open the `index.html` and confirm that all site elements are present.

8 Static website on S3

These instructions are quite sparse, and you will have to use AWS S3 (and other) resources to help you.

Follow the instructions at

`https://docs.aws.amazon.com/AmazonS3/latest/dev/HostingWebsiteOnS3Setup.html`

so that your site is world-visible at the S3 endpoint. Exceptions:

- In Step 3, upload all files from your static website rather than just one document.

8.1 Bucket policy

Bucket policies are a way to specify who (the principal) is allowed to do what (the action) on a specific resource. For example:

```
  {
   "Version":"2012-10-17",
   "Statement":[{
         "Sid":"PublicReadForGetBucketObjects",
         "Effect":"Allow",
           "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::example-bucket/*"
      ]
     }
   ]
}
```

- A policy consists of a number of statements.
- Each statement has:
  - An Sid to identify the statement
  - An *effect*, saying whether the statement allows or prohibits.
  - A list of *Principals* or AWS usernames that this statement applies to.
  - A list of *Action*s that the statement allows or denys.
  - The resource (possibly including wildcards) that this statement applies to.

9 Cleanup

Remember to delete resources you have finished using them. This lab should cost nothing to do, but you should get into the habit of cleaning things up to avoid unexpected surprises.