Amazon S3 Cookbook

Amazon S3 is one of the most famous and trailblazing cloud object storage services, which is highly scalable, low-latency, and economical. Users only pay for what they use and can store and retrieve any amount of data at any time over the Internet, which attracts Hadoop users who run clusters on EC2.

The book starts by showing you how to install several AWS SDKs such as Ruby, Java, Node.js, PHP, and Python, and shows you how to manage objects. Then, you’ll be taught how to use the installed AWS SDKs to develop applications with Amazon S3 and you will explore the Amazon S3 pricing model. In addition to this, the book covers several practical recipes about how to distribute your content with CloudFront, secure your content with IAM, and notify S3 events with Lambda. By the end of this book, you will be successfully implementing pro-level practices and solutions in Amazon S3.

What you will learn from this book

- Host a static website on Amazon S3
- Calculate costs with the AWS simple monthly calculator
- Deploy a static website via CloudFormation
- Distribute your content via CloudFront
- Secure resources with bucket policies and IAM
- Protect objects using server-side and client-side encryption
- Enable cross-origin resource sharing
- Manage objects’ lifecycles to incur lower costs
- Optimize performance for uploading as well as downloading objects

Who this book is written for

This book is for cloud developers who have experience of using Amazon S3 and are also familiar with Amazon S3.

Over 30 hands-on recipes that will get you up and running with Amazon Simple Storage Service (S3) efficiently
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 7 'Sending Authenticated Requests with AWS SDKs'
- A synopsis of the book’s content
- More information on Amazon S3 Cookbook
Naoya Hashimoto has worked on system designing, implementing, and system maintenance as an infrastructure engineer in a data center, a management service provider, and housing/hosting service provider for years. After he was introduced to public cloud services a few years ago, his career, interest, and motive shifted to the public cloud, including private- and hybrid-cloud-computing-related services (such as network, storage, orchestration, job automation, and monitoring), as well as to open source software.

He has been a technical reviewer of many books, such as Mastering AWS Development, Icinga Network Monitoring, PostgreSQL Cookbook, and Building Networks and Servers Using Beaglebone, all by Packt Publishing.
Amazon Simple Storage Service (Amazon S3) is one of the most popular online object storage services with high scalability, durability, and automatic self-healing. It also enables programmatic access with AWS SDKs that simplify your programming tasks.

Amazon S3 Cookbook is a recipe-based practical guide that will get you up and running with using Amazon S3 efficiently. This book will not only tell you how to use several functions of Amazon S3, but it will also give you valuable information and a deeper understanding of, for example, managing buckets and objects with AWS SDKs, cost calculation, how to secure your contents, lifecycle management, and performance optimization to leverage Amazon S3 to build amazing cloud-based apps.

What this book covers

Chapter 1, Managing Common Operations with AWS SDKs, introduces what AWS SDKs can do with Amazon S3 by using the official AWS SDK sample application code to create S3 buckets and upload, list, get, and download objects into and from a bucket.

Chapter 2, Hosting a Static Website on Amazon S3 Bucket, covers hosting a static website's contents by using a custom domain on Amazon S3 instead of using web servers such as Apache or Nginx on EC2 through a management console (GUI) and AWS CLI (command line). You will also learn the merits of using Amazon S3 as a website.

Chapter 3, Calculating Cost with the AWS Simple Monthly Calculator, talks about calculating the total cost of storing data and delivering objects through S3 with the Amazon Web Services Simple Monthly Calculator (the AWS calculator), based on a couple of scenarios.

Chapter 4, Deploying a Static Website with CloudFormation, covers deploying a template of a static website with CloudFormation via the S3 console and using AWS CLI.

Chapter 5, Distributing Your Contents via CloudFront, talks about delivering a static website on S3 buckets through the CloudFront edge location (CDN), configuring S3 buckets as an origin store to minimize network latency.
Chapter 6, Securing Resources with Bucket Policies and IAM, covers managing access to resources such as buckets and objects, configuring bucket policies, and IAM users, groups, and policies.

Chapter 7, Sending Authenticated Requests with AWS SDKs, talks about making requests using IAM and federated users’ temporary credentials with AWS SDKs to grant permissions to temporarily access Amazon S3 resources.

Chapter 8, Protecting Data Using Server-side and Client-side Encryption, deals with encrypting and decrypting your data using server-side and client-side encryption to securely upload and download your contents.

Chapter 9, Enabling Cross-origin Resource Sharing, shows you how to enable cross-origin resource sharing (CORS) and allow cross-origin access to S3 resources to interact with resources in a different domain for client web applications.

Chapter 10, Managing Object Lifecycle to Lower the Cost, talks about configuring lifetime cycle policies on S3 buckets to automatically delete after a certain time, using Reduced Redundancy Storage (RRS) or by archiving objects into Amazon Glacier.

Chapter 11, S3 Performance Optimization, deals with improving the performance of uploading, downloading, and getting and listing objects.

Chapter 12, Creating Triggers and Notifying S3 Events to Lambda, covers sending notifications to let AWS Lambda execute Lambda functions that enable S3 event notifications.
In this chapter, you will learn:

- How to make requests using IAM user temporary credentials with AWS SDK
- How to make requests using federated user temporary credentials with AWS SDK

Introduction

In Chapter 6, Securing Resources with Bucket Policies and IAM, you not only learned how to secure your buckets or objects using bucket policies and user policies but also how to manage common operations for an S3 bucket with AWS SDKs, which we came across in Chapter 1, Managing Common Operations with AWS SDKs. In the real world, for example, when you make your applications, you will need to use methods in the library or SDK to simplify using AWS services in your application. We will follow how to use AWS SDKs to make requests using the IAM user’s temporary credentials with AWS SDKs.
How to make requests using IAM user temporary credentials with AWS SDK

There are situations where you need to grant permissions to temporarily access Amazon S3 resources. For example, your applications create temporary users to get objects in an S3 bucket for a certain period of time and the permissions granted to the temporary user need to be disabled or removed after the duration expires. IAM users support to request temporary security credentials using the AWS Security Token Service (AWS STS). You will learn how to make requests using IAM user temporary credentials with AWS SDK for PHP.

Getting ready

As we use AWS SDK for PHP in this chapter, you need to have the AWS SDK for PHP properly installed in your server or client PC. Install AWS SDK for PHP following the instruction Learning AWS SDK for PHP and basic S3 operations with sample code from Chapter 1, Managing Common Operations with AWS SDKs.

How to do it...

First, we create an IAM policy to allow temporary security credentials for IAM users and list the bucket action, and then attach the policy to an IAM group. Next, we create an IAM user to make requests using temporary security credentials. Finally, we use the IAM user's temporary security credentials and list objects in a bucket using a sample PHP script:

1. Sign in to the AWS management console and move to the S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket following the instructions in How to configure a static website on Amazon S3 bucket from Chapter 2, Hosting a Static Website on Amazon S3 Bucket.
3. Create an IAM policy and IAM group, and attach the policy to the IAM group, then create an IAM user and note down the credentials following the instructions at Walkthrough 1-4Bucket and User policy examples: Allowing a user to access to a folder in a bucket in a specific region in a user policy. The policy to be attached to your IAM group is as follows:
S3BucketName is to be replaced with the Bucket name.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "sts:GetFederationToken*",
            "s3:ListBucket"
         ],
         "Resource": [
            "*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::S3BucketName"
         ]
      }
   ]
}
```

4. Update the AWS config file by entering the IAM credentials:

   ```bash
   $ aws configure
   AWS Access Key ID [None]: access-key
   AWS Secret Access Key [None]: secret-access-key
   Default region name [None]: us-east-1
   Default output format [None]:
   ```

**How it works...**

As we have configured an IAM policy to allow IAM users to use temporary security credentials and an IAM group, attached the IAM policy to the IAM group, create an IAM user, and attach the IAM user to the IAM group. Here, we will be making S3 requests using temporary security credentials and a sample PHP script.

1. Download the sample SDK application:

   ```bash
   $ git clone https://github.com/awslabs/aws-php-sample.git
   $ cd aws-php-sample/
   ```
2. Set up the following sample PHP script under the `aws-php-sample` directory:

```php
<?php

// Include the AWS SDK using the Composer autoloader.
require 'vendor/autoload.php';

// Retrieve variable from command-line
$bucket = $argv[1];
$duration = $argv[2];

use Aws\Sts\StsClient;
use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$sts = StsClient::factory();
$credentials = $sts->getSessionToken()->get('Credentials');
$s3 = S3Client::factory(array(
    'key' => $credentials['AccessKeyId'],
    'secret' => $credentials['SecretAccessKey'],
    'token' => $credentials['SessionToken'],
    'DurationSeconds' => $duration
));

// Debugging credentials
echo "## -- AWS IAM Credentials for debugging -- ##" . "\n";
echo "AccessKeyId: " . $credentials['AccessKeyId'] . "\n";
echo "SecretAccessKey: " . $credentials['SecretAccessKey'] . "\n";
echo "SessionToken: " . $credentials['SessionToken'] . "\n";
echo "\n";

try {
    $objects = $s3->getIterator('ListObjects', array(
        'Bucket' => $bucket
    ));

    echo "## -- Objects in bucket: $bucket -- ##" . "\n";
echo "LastModified" . "\t\t" . "Size" . "\t\t" . "Object" . "\n";
```
foreach ($objects as $object) {
    echo $object['LastModified'] . "\t" . $object['Size'] . "\t" . $object['Key'] . "\n";
    echo "\n";
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}

3. Set the S3 bucket name referred to as bucket, and the session duration referred to as duration, and then execute the sample PHP script using the following command:

```
$ bucket=S3BucketName
$ duration=session-duration
$ phpsample_code.php $bucket $duration
```

## -- AWS IAM Credentials for debugging -- ##

AccessKeyId: access-key
SecretAccessKey: secret-access-key
SessionToken: session-token

## -- Objects in bucket: hashnao-policy-test -- ##

<table>
<thead>
<tr>
<th>LastModified</th>
<th>Size</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-03-02T00:33:41.000Z</td>
<td>1048576</td>
<td>file_001.txt</td>
</tr>
<tr>
<td>2015-03-02T00:33:43.000Z</td>
<td>2097152</td>
<td>file_002.txt</td>
</tr>
<tr>
<td>2015-03-02T00:33:47.000Z</td>
<td>3145728</td>
<td>file_003.txt</td>
</tr>
</tbody>
</table>

You can specify the duration in seconds between 1 and 36 hours. The session duration is 1 hour (which is 3,600 seconds), by default.

Finally, let's examine the sample PHP script so that we can understand which method requests temporary security credentials, and requests to access the objects in the bucket using the temporary security credentials.

The following class methods import functions as follows:

- The Aws\Sts\StsClient class creates a new Amazon STS client object
- The Aws\S3\S3Client class creates a new Amazon S3 client
- The Aws\S3\Exception/S3Exception class imports a default service exception class:
  ```php
  use Aws\Sts\StsClient;
  use Aws\S3\S3Client;
  use Aws\S3\Exception\S3Exception;
  ```
Sending Authenticated Requests with AWS SDKs

The `getSessionToken` call returns a set of temporary credentials for an IAM user and the credentials consist of an access key ID, a secret access key, and a security token in the array:

```php
$credentials = $sts->getSessionToken()->get('Credentials');
$s3 = S3Client::factory(array(
    'key'    => $credentials['AccessKeyId'],
    'secret' => $credentials['SecretAccessKey'],
    'token'  => $credentials['SessionToken'],
    'DurationSeconds'  => $duration
));
```

In the following part, the `getIterator` method calls operations and enumerates through the resources from a result set with `foreach`, iterates objects in the bucket, and then displays the `LastModified` property, the `size` property, and the `key` property. Finally, it catches the exception in the `S3Exception` class and displays its messages:

```php
try {
    $objects = $s3->getIterator('ListObjects', array(
        'Bucket' => $bucket
    ));
    echo "## -- Objects in bucket: $bucket -- ##" . "\n"
    echo "LastModified" . "\t" . "Size" . "\t" . "Object" . "\n";
    foreach ($objects as $object) {
        echo $object['LastModified'] . "\t" . $object['Size'] . "\t" . $object['Key'] . "\n";
    }
    echo "\n"
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

See also

How to make requests using federated user temporary credentials with AWS SDK

We configured IAM users to control permissions of each IAM user to access AWS resources. AWS IAM supports identity federation as external identities, to securely access to your AWS resources without the necessity of creating IAM users. We can provide permissions to the federated user using temporary credentials without having to create IAM users.

For further information about Federation Management, see http://aws.amazon.com/iam/details/manage-federation/.

Getting ready

As we use AWS SDK for PHP in this chapter, you need to have the AWS SDK for PHP properly installed in your server or client PC. Install AWS SDK for PHP following the instructions in the Learning AWS SDK for PHP and basic S3 operations with sample code section of Chapter 1, Managing Common Operations with AWS SDKs.

How to do it...

First, we create an IAM policy to allow temporary security credentials for federated users and list bucket actions, and then attach the policy to an IAM group. Next, we create an IAM user to make requests using temporary security credentials. Finally, we grant permissions for a federated user and list objects in a bucket using a sample PHP script:

1. Sign in to the AWS management console and move to the S3 console at https://console.aws.amazon.com/s3/.

2. Create a bucket following the instructions in the How to configure a static website on Amazon S3 bucket section of Chapter 2, Hosting a Static Website on Amazon S3 Bucket.

3. Create an IAM policy, IAM group, and attach the policy to the IAM group, then create an IAM user and note down the credentials following the instructions at Walkthrough 1-4Bucket and User policy examples: Allowing a user to access to a folder in a bucket in a specific region in a user policy. The policy to be attached to your IAM group is as follows:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFolderAccess",
            "Effect": "Allow",
            "Action": "s3:ListObjects",
            "Resource": ["arn:aws:s3:::S3BucketName/*"]
        }
    ]
}
```
"Statement": [ 
    { 
      "Effect": "Allow",
      "Action": [ "sts:GetSessionToken*", "s3:ListBucket" ],
      "Resource": [ "*" ]
    },
    { 
      "Effect": "Allow",
      "Action": [ "s3:ListBucket" ],
      "Resource": [ "arn:aws:s3:::S3BucketName" ]
    }
  ]

4. Update the AWS config file entering the IAM credentials:

   $ aws configure
   AWS Access Key ID [None]: access-key
   AWS Secret Access Key [None]: secret-access-key
   Default region name [None]: us-east-1
   Default output format [None]:

   How it works...

   We have configured an IAM policy and an IAM group, attached the IAM policy to the IAM group, created an IAM user, and attached the IAM user to the IAM group. This is the final part to verify the permissions granted to the federated user:

   1. Download the sample SDK application:

      $ git clone https://github.com/awslabs/aws-php-sample.git
      $ cd aws-php-sample/

   2. Set the following sample PHP script under the aws-php-sample directory:

      <?php

      // Include the AWS SDK using the Composer autoloader.

      160
require 'vendor/autoload.php';

// Retrieve variable from command-line
$bucket = $argv[1];
$duration = $argv[2];

use Aws\Sts\StsClient;
use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$sts = StsClient::factory();

$credentials = $sts->getSessionToken()->get('Credentials');
$s3 = S3Client::factory(array(
    'key'    => $credentials['AccessKeyId'],
    'secret' => $credentials['SecretAccessKey'],
    'token'  => $credentials['SessionToken'],
    'DurationSeconds'  => $duration
));

// Debugging credentials
echo "## -- AWS IAM Credentials for debugging -- ##" . "\n";
echo "AccessKeyId: " . $credentials['AccessKeyId']  . "\n";
echo "SecretAccessKey: " . $credentials['SecretAccessKey'] . "\n";
echo "SessionToken: " . $credentials['SessionToken'] . "\n";
echo "\n";
try {
    $objects = $s3->getIterator('ListObjects', array(
        'Bucket' => $bucket
    ));

    echo "## -- Objects in bucket: $bucket -- ##" . "\n";
    echo "LastModified" . "\t" . "Size" . "\t" . "Object" . "\n";
    foreach ($objects as $object) {
        echo $object['LastModified'] . "\t" . $object['Size'] . "\t" . $object['Key'] . "\n";
    }
}
Sending Authenticated Requests with AWS SDKs

```php
    echo "\n";
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

3. Set the S3 bucket name referred to as `bucket`, and the federated username referred to as `user`, and the session duration referred to as `duration`, and then execute the sample PHP script using the following command:

```bash
$ bucket=S3BucketName
$ user=FederatedUserName
$ duration=session-duration
$ phpsample_code.php $bucket $duration
```

```markdown
## -- AWS IAM Credentials for debugging -- ##
AccessKeyId: access-key
SecretAccessKey: secret-access-key
SessionToken: session-token

## -- Objects in bucket: hashnao-policy-test -- ##
<table>
<thead>
<tr>
<th>LastModified</th>
<th>Size</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-03-02T00:33:41.000Z</td>
<td>1048576</td>
<td>file_001.txt</td>
</tr>
<tr>
<td>2015-03-02T00:33:43.000Z</td>
<td>2097152</td>
<td>file_002.txt</td>
</tr>
<tr>
<td>2015-03-02T00:33:47.000Z</td>
<td>3145728</td>
<td>file_003.txt</td>
</tr>
</tbody>
</table>
```

You can specify the duration in seconds between 1 and 36 hours.
The session duration is 1 hour (which is 3,600 seconds), by default.

Finally, let’s examine the sample PHP script so that we can understand which method makes requests to federated users and your applications and how federated users and your applications can send authenticated requests to access the objects in the bucket.

In the `StsClient::factory` method, several parameters, with key and value, are provided in its array and fetched the federated credentials as follows:

- **Name** is referred as the name of the federated user and used as an identifier for the temporary security credentials.
- **DurationSeconds** is the duration in seconds in which the session should last. The acceptable duration range is from 900 seconds (15 minutes) to 129,600 seconds (36 hours) and 43,200 seconds (12 hours), as the default.
Policy defines an IAM policy in the JSON format and is passed with the GetFederationToken call, and then evaluated along with the policy attached to the IAM user. The policy is used to define the permissions that are available to the IAM user:

```php
$sts = StsClient::factory();
$credentials = $sts->getSessionToken()->get('Credentials');
$s3 = S3Client::factory(array(
    'key'    => $credentials['AccessKeyId'],
    'secret' => $credentials['SecretAccessKey'],
    'token'  => $credentials['SessionToken'],
    'DurationSeconds' => $duration
));
```

The following part provides temporary security credentials in order to send authenticated requests to Amazon S3:

```php
$credentials = $result->get('Credentials');
$s3 = S3Client::factory(array(
    'key'    => $credentials['AccessKeyId'],
    'secret' => $credentials['SecretAccessKey'],
    'token'  => $credentials['SessionToken']
));
```

See also

Where to buy this book

You can buy Amazon S3 Cookbook from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.