Signed Download URLs

In this lesson, you will learn how to sign download URLs and configure access to the files after they are uploaded.

WE'LL COVER THE FOLLOWING

Another URL is required

In the previous chapter, you used the same URL for both the form display function and the form processing function. This was possible because the browser web form workflow used two different HTTP methods. To display the form, it sent a GET request. To upload the submission details, it sent a POST request. Because S3 redirects work as GET method calls, you cannot use the same URL with different methods anymore. The browser will send a POST request directly to S3, which will redirect to another URL. You'll need a different URL to handle that redirect, so you can create a new API endpoint.

When S3 sends a redirect after the upload, it will include information about the uploaded file in the query string. This allows you to do some post-processing on the uploaded file or to let the users access it. To keep things simple for now, just allow users to download a file they uploaded. In theory, you could read the file out using a Lambda function and send it back through API Gateway. By now, though, you know that there is a better way to handle that.

Configuring access

S3 can serve public files directly to browsers. For example, public files in a bucket called `testbucket` in the `us-east-1` region will be accessible directly from `https://testbucket.s3.amazonaws.com`. For other regions, the second URL

from `https://testbucket.s3.amazonaws.com`. For other regions, the second URL component is `s3-` followed by the region (for example `https://testbucket.s3-eu-west-1.amazonaws.com`). Letting users download files is a great option if you don't care about protecting read access, for example if you are making a public gallery. To upload publicly readable files, just change the access control list in the upload policy from `private` to `public-read` (line 19 in `show-form.js`).

To tightly control who can read the uploaded files, instead of a public link you can pre-sign a download request. A signed download request effectively provides a temporary grant to access a particular file on S3 directly. Similar to file uploads, directly downloading files from S3 is so common that the AWS SDK has a shortcut operation for it. You can use `s3.getSignedUrl` to sign generic S3 requests. You need to give it the related object key and the expiry time for the grant and specify the type of the S3 REST API operation. For retrieving objects, the REST API operation is called `getObject`. The signature method returns a URL with query string parameters prepared according to the authorised grant.

A new file is created in the Lambda code directory, called `confirm-upload.js`, using the following listing:

```
const htmlResponse = require('./html-response');
const aws = require('aws-sdk');
const s3 = new aws.S3();

exports.lambdaHandler = async (event, context) => {
  const params = {
    Bucket: process.env.UPLOAD_S3_BUCKET,
    Key: event.queryStringParameters.key,
    Expires: 600
  };
  const url = s3.getSignedUrl('getObject', params);
  const responseText = `
    <html><body>
    <h1>Thanks</h1>
    <a href="${url}">
      check your upload
    </a>
        (the link expires in 10 minutes)
    </body></html>
  `;
  return htmlResponse(responseText);
};
```

code/ch8/user-form/confirm-upload.js

Expired credentials

Although the signing methods allow you to specify arbitrary expiry periods, if the credentials used to sign a request expire before the policy, the signature will no longer be valid. Lambda functions work under temporary credentials that are only valid for a short period. AWS does not publish any data about this officially, but tests suggest that this period is 10 to 20 minutes. This is fine for immediate operations, but not much more. If you need to generate signed URLs that last longer, you'll need to create a separate IAM user, allow it access to the bucket, generate a set of AWS access keys for the user, and then instantiate the S3 service object in Lambda using those credentials.

In the next lesson, you will learn how to protect S3 files.