Passing Resource References to Functions

In this lesson, you will learn how to let SAM configure the environment variables for your application.

WE'LL COVER THE FOLLOWING

- Environment variables

In order to complete the function, you'll need to tell it which bucket to use (line 8 from the code in the previous lesson). Lambda functions don't really know about SAM and CloudFormation resources, so the function can't just ask for the `UploadS3Bucket` resource. SAM will create the bucket using a randomised name, and you need to tell the Lambda function about the actual value.

You definitely don't want to hard code a bucket name in the function source, because then SAM can't automatically manage buckets. You could change the API Gateway resource to stick that information onto the request while it's passing through the API, for example as an additional header, but that requires messy request transformations. For situations such as this one, it's best to set a Lambda environment variable.

Environment variables

Environment variables are textual key-value pairs assigned to a running process. With a Node.js Lambda runtime, you can read them using the standard Node `process.env` object. SAM lets you configure those values in the template, so it's easy to pass the actual bucket name to a function based on the logical bucket reference.

Environment variables are great for configuring Lambda functions with references to other resources in the same SAM template. For more complex configuration scenarios, such as rotating secrets and reconfiguring services without redeployment, check out the AWS Systems Manager Parameter Store and AWS Secrets Manager.

It helps to use all uppercase letters for environment variable names to clearly differentiate them from other values, but you can use any naming convention you like. The `process-form.js` file is updated to look like in the following listing:

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

exports.lambdaHandler = async (event, context) => {
  const bucketName = process.env.UPLOAD_S3_BUCKET;
  await s3.putObject({
    Bucket: bucketName,
    Key: context.awsRequestId,
    Body: JSON.stringify(event)
  }).promise();
  const thanksHtml = `
    <html>
    <head>
      <meta charset="utf-8"/>
    </head>
    <body>
      <h1>Thanks</h1>
      <p>We received your submission</p>
      <p>Reference: ${context.awsRequestId}</p>
      </p>
    </body>
    </html>
  `;

  return htmlResponse(thanksHtml);
};
```

code/ch7/user-form/process-form.js

SAM can configure environment variables for a function using the `Environment` property. It must have a `Variables` sub-property, which can then contain a map of keys and values. You can use the `!Ref` function to convert the logical name of a CloudFormation resource into its physical ID. You can add the following three lines to the `ProcessFormFunction` template, at the same indentation level as other function properties (so `Environment` should be aligned with `Events`):

```
Environment:
  Variables:
    UPLOAD_S3_BUCKET: !Ref UploadS3Bucket
```

Line 47 to Line 49 of code/ch7/template.yaml

In the next lesson, you will configure IAM policies for the application. See you there!