Using `mogrify`

In this lesson, you'll learn about Mogrify for image processing and about SAR which is a public library for AWS SAM applications!

WE'LL COVER THE FOLLOWING

This chapter explains how to compose applications from third-party components by using Lambda layers. You will also learn about the AWS Serverless Application Repository, a public library of reusable application templates.

`mogrify`

In the previous chapter, you built a skeleton for asynchronous processing. In this chapter, you'll use that structure to create thumbnails from uploaded images. JavaScript isn't really designed for working with binary files, and it would be better if you used a low-level system utility for image processing.

For example, the `mogrify` command-line utility from the ImageMagick package can transform a wide range of image formats and is very easy to use. To modify a file into a thumbnail version, you can just use the `-thumbnail` option. Mogrify understands resolutions in the format `Width x Height`, but it can preserve aspect ratios if you provide only the width followed by the letter `x`. For example, to convert an image file into a 300-pixel wide thumbnail, you just need to run the following command:

```
mogrify -thumbnail 300x <FILE>
```

Different people might want to set up different thumbnail sizes, so you'll want to add the parameter for that into your template. You can add the block from the following listing to the `Parameters` section of your template, at the same indentation level as other parameter definitions (for example, `UploadLimitInMb`).

```
ThumbnailWidth:
  Type: Number
  Default: 300
  Description: Thumbnail width in pixels
  MinValue: 10
  MaxValue: 1000
```

Line 19 to Line 24 of code/ch10/template.yaml

Lambda runs a reduced version of Amazon Linux, a clone of CentOS, which is itself a clone of Red Hat Enterprise Linux. ImageMagick tools are usually present on Linux systems; however, they are not included in the more recent Lambda environments. Up until the Node.js 8 Lambda runtime, Lambda containers also included ImageMagick tools. In order to improve start-up performance, the standard Lambda environment comes with only the essential system libraries. In order to use `mogrify`, you'll have to add that binary tool to the Lambda virtual machines running your conversion function.

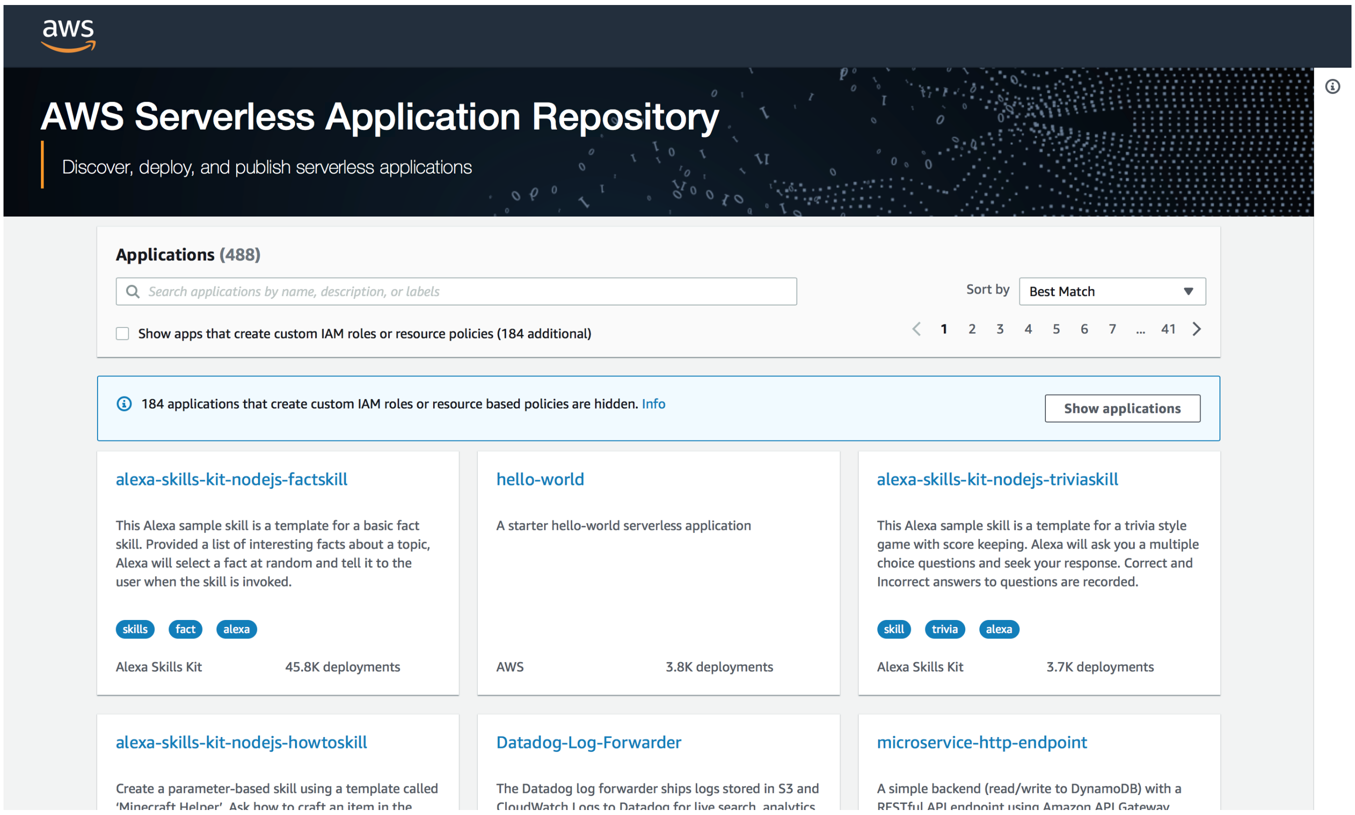
The AWS Serverless Application Repository

The *AWS Serverless Application Repository* (SAR) is a library of AWS SAM applications (or, more correctly, it is a library of SAM application

applications (or, more correctly, it is a library of SAM application
components). SAR has two primary purposes. The first is to enable

organisations to share reusable components internally, so teams can easily deploy infrastructural templates built by other teams. The second is to work as a public library, making a few hundred open-source components available to everyone.

The public part of SAR makes it easy to discover blueprints for applications, to find integration components for third-party services and to set up utility resources (see figure below). Some of these applications were published by AWS, and some by third-party developers. In order to handle image conversions with the new Lambda runtimes in MindMup, I had to package ImageMagick as a SAM component. I then published it to SAR so anyone could get started with image conversions in Lambda easily. You will find that component by searching for `image-magick-lambda-layer` in the repository.


aws
AWS Serverless Application Repository
Discover, deploy, and publish serverless applications
Applications (488)
Search applications by name, description, or labels
Sort by
Best Match
Show apps that create custom IAM roles or resource policies (184 additional)
1 2 3 4 5 6 7 ... 41
Show applications

184 applications that create custom IAM roles or resource based policies are hidden. Info

alexa-skills-kit-nodejs-factskill

This Alexa sample skill is a template for a basic fact skill. Provided a list of interesting facts about a topic, Alexa will select a fact at random and tell it to the user when the skill is invoked.

skills fact alexa

Alexa Skills Kit 45.8K deployments

hello-world

A starter hello-world serverless application

AWS 3.8K deployments

alexa-skills-kit-nodejs-triviaskill

This Alexa sample skill is a template for a trivia style game with score keeping. Alexa will ask you a multiple choice questions and seek your response. Correct and Incorrect answers to questions are recorded.

skill trivia alexa

Alexa Skills Kit 3.7K deployments

alexa-skills-kit-nodejs-howtoskill

Create a parameter-based skill using a template called 'Minecraft Helper'. Ask how to craft an item in the

Datadog-Log-Forwarder

The Datadog log forwarder ships logs stored in S3 and CloudWatch Logs to Datadog for live search, analytics

microservice-http-endpoint

A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway

SAR makes it easy to discover third-party components for SAM applications and to deploy pre-packaged components into your AWS account. Find something interesting using the repository web page then click the *Deploy* button. Perhaps more interestingly, it's also possible to directly include anything published to SAR in a SAM application. Technically, a SAR component is just a published CloudFormation template and they can be embedded in other templates. The marketing term AWS uses for such reused components is *nested applications*. To import a SAR component into a SAM template, you just need to declare a resource with the type

`AWS::Serverless::Application` and point to the application ID and required application version.

For example, to deploy version `1.0.0` of the `image-magick-lambda-layer` SAR component, you just add the following lines to your application template, in the `Resources` section (at the same indentation level as other resources, for example `WebApi`).

```
ImageMagick:
  Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: arn:aws:serverlessrepo:us-east-1:145266761615:applications/image-magick-
      SemanticVersion: 1.0.0
```

Line 83 to Line 88 of code/ch10/template.yaml

Compiling binary tools

One of the best aspects of Linux software is that most of it is open source. If you want to use a binary tool that is not available from the SAR, you can compile it for the correct operating system version and deploy as part of your Lambda function. A great starting point for that is to use the Docker container images from the LambCI project, available from https://github.com/lambci/docker-lambda. They are intended to simulate Lambda runtimes in continuous integration environments, but can also be used to ensure that you are building a binary tool using the correct system libraries for a Lambda runtime environment.

Michael Hart, the author of the LambCI project, even provided a set of templates for compiling many standard Linux tools using the RPM package format. For more information, see the Yumda project https://github.com/lambci/yumda on GitHub.

You'll learn about Lambda layers in the next lesson. Stay tuned!