Invoking System Utilities

Here, you'll learn how to invoke system utilities!

WE'LL COVER THE FOLLOWING

- `child_process`

`child_process`

Now that `mogrify` is available to your Lambda function, you can change the source code for the conversion function to execute it.

To start a command-line utility from JavaScript, you'll need to use the Node.js child process features.

The `spawn` function from the Node.js `child_process` module uses callbacks. You need to wrap that function into a `Promise`, so it can be used directly in `async` functions. (This is a limitation of Node.js, so if you are using a different language, the whole issue with multiple types of asynchronous processes does not apply.) Another file is added to your `image-conversion` function directory and is called `child-process-promise.js`. The file should contain the following listing, which is a relatively generic function that will create a sub-process to invoke an external command and print out any console output into the logs so you can troubleshoot more easily.

```
const childProcess = require('child_process');
exports.spawn = function (command, argsarray, envoptions) {
  return new Promise((resolve, reject) => {
    console.log('executing', command, argsarray.join(' '));
    const childProc = childProcess.spawn(command, argsarray, envoptions);
    childProc.stdout.on('data', buffer => console.log(buffer.toString()));
    childProc.stderr.on('data', buffer => console.error(buffer.toString()));
    childProc.on('exit', (code, signal) => {
      console.log(`${command} completed with ${code}:${signal}`);
      if (code || signal) {
        reject(`${command} failed with ${code || signal}`);
      } else {
        resolve();
      }
    });

    childProc.on('error', reject);
  });
};
```

code/ch10/image-conversion/child-process-promise.js

Next, you can change the `index.js` file in the `image-conversion` function directory to include the new child process function, read out the thumbnail width from the environment variables, and run `mogrify` to process the temporary file. The function should look similar to the code in the following listing (the important changes compared to the code in the previous chapter are lines 8, 9 and 25-28).

```
const path = require('path'),
  os = require('os'),
  s3Util = require('./s3-util'),
  extractS3Info = require('./extract-s3-info'),
  silentRemove = require('./silent-remove'),
  OUTPUT_BUCKET = process.env.OUTPUT_BUCKET,
  supportedFormats = ['jpg', 'jpeg', 'png', 'gif'],
  THUMB_WIDTH = process.env.THUMB_WIDTH,
  childProcessPromise = require('./child-process-promise');

exports.handler = async (event, context) => {
  const s3Info = extractS3Info(event),
    id = context.awsRequestId,
    extension = path.extname(s3Info.key).toLowerCase(),
    tempFile = path.join(os.tmpdir(), id + extension),
    extensionWithoutDot = extension.slice(1),
    contentType = `image/${extensionWithoutDot}`;

  console.log('converting', s3Info.bucket, ':', s3Info.key, 'using', tempFile);

  if (!supportedFormats.includes(extensionWithoutDot)) {
    throw new Error(`unsupported file type ${extension}`);
  }
  await s3Util.downloadFileFromS3(s3Info.bucket, s3Info.key, tempFile);
  await childProcessPromise.spawn(
    '/opt/bin/mogrify',
    ['-thumbnail', `${THUMB_WIDTH}x`, tempFile],
  );
  await s3Util.uploadFileToS3(OUTPUT_BUCKET, s3Info.key, tempFile, contentType);
  await silentRemove(tempFile);
};
```

code/ch10/image-conversion/index.js

To send the new version of the function to the cloud, you can run `sam build` and `sam package` as usual. For deployment, you'll need to activate another

CloudFormation feature so it can process nested applications. You add `CAPABILITY_AUTO_EXPAND` into the list of capabilities:

```
sam deploy --template-file output.yaml --stack-name sam-test-1 --capabilit
ies CAPABILITY_IAM CAPABILITY_AUTO_EXPAND
```

After SAM has deployed the application, open the main application web page, upload a new image, *wait a few seconds* and then click on the resulting link. You should see the image resized into a 300-pixel wide thumbnail. Don't click on the link immediately, as the result won't be ready instantly. You will address this problem in the next chapter.

Environment Variables

Key:	Value:
AWS_ACCESS_KEY_ID	Not Specified...
AWS_SECRET_ACCE...	Not Specified...
BUCKET_NAME	Not Specified...
AWS_REGION	Not Specified...

```
{
  "body": "{\"message\": \"hello world\"}",
  "resource": "/{proxy+}",
  "path": "/path/to/resource",
  "httpMethod": "POST",
  "isBase64Encoded": false,
  "queryStringParameters": {
    "foo": "bar"
  },
  "pathParameters": {
    "proxy": "/path/to/resource"
  },
  "stageVariables": {
    "baz": "qux"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate, sdch",
    "Accept-Language": "en-US,en;q=0.8",
    "Cache-Control": "max-age=0",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Desktop-Viewer": "true",
    "CloudFront-Is-Mobile-Viewer": "false",
    "CloudFront-Is-SmartTV-Viewer": "false",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Viewer-Country": "US",
    "Host": "1234567890.execute-api.us-east-1.amazonaws.com",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Custom User Agent String",
    "Via": "1.1 08f323deadbeefa7af34d5feb414ce27.cloudfront.net (CloudFront)",
    "X-Amz-Cf-Id": "cDehVQoZnx43VYQb9j2-nvCh-9z396Uhbp027Y2JvkCPNLmGJHqlaA==",
    "X-Forwarded-For": "127.0.0.1, 127.0.0.2",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"
  },
  "requestContext": {
    "accountId": "123456789012",
    "resourceId": "123456",
    "stage": "prod",
    "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
    "requestTime": "09/Apr/2015:12:34:56 +0000",
    "requestTimeEpoch": 1428582896000,
    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "accessKey": null,
      "sourceIp": "127.0.0.1",
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,
      "userAgent": "Custom User Agent String",
      "user": null
    },
    "path": "/prod/path/to/resource",
    "resourcePath": "/{proxy+}",
    "httpMethod": "POST",
    "apiId": "1234567890",
    "protocol": "HTTP/1.1"
  }
}
```

In the next lesson, you will learn how to publish the application to SAR.