swift-lambda Documentation

Release 0.1

swift-lambda team

Feb 17, 2017

Overview

swift-lambda brings the power and ease-of-use of the Swift programming language to the scalability and buzzword-compliance of AWS Lambda. Use the tools you already know and love to power the backend of your mobile application.

Check out the Quickstart to get up and running in a few minutes.

CHAPTER 1

Quickstart

swift-lambda brings the power and ease-of-use of the Swift programming language to the scalability and buzzword-compliance of AWS Lambda. Use the tools you already know and love to power the backend of your mobile application.

Installation

swift-lambda can be installed using Homebrew:

```
$ brew install --HEAD awswift/swift-lambda/swift-lambda
```

Pre-requisites

- A Swift programming environment. On macOS, this means installing Xcode. On Linux, it means downloading Swift from the official website.
- Docker. We use Docker to emulate a local version of the Linux-y AWS Lambda runtime environment. Because Swift compiles to native code, we need to compile the code on a computer that is as similar to Lambda as possible.

  Note that this is still needed even if you are running on Ubuntu, because Lambda is different enough from Ubuntu that compiling on your own machine won't work.
- **Interim**: The AWS CLI. Porting the AWS APIs is still a work in progress in the `awswift/Awswift` repo. Until that is complete we have relied on shelling out to the CLI tool for the alpha release.
- **Interim**: The `stackup` Ruby gem. This is a great utility for dealing with CloudFormation and we are shelling out to it until such functionality has been replicated in a Swift library.

Usage

swift lambda setup

`setup` is a once-off command that will get an AWS account ready for Swift-powered AWS Lambda functions. Right now it creates an S3 bucket for storing your code and an IAM execution role to grant your Lambda functions permission to write logs.

swift lambda init <name>

`init` will initialise a brand-new *Hello, World!* Swift-powered AWS Lambda function. Everything you need to get up and running will be created for you. `<name>` is how you tell the `init` command what you want your function to be called!

swift lambda build

`build` will create the Zip archive that you upload to AWS Lambda in order to run your code. swift-lambda will compile your code in a Lambda-like Docker container, zip it up (alongside the Swift runtime) and stick in a few extra files that Lambda requires you to have.

swift lambda deploy [--new-version] [--skip-libs]

`deploy` is how you get the Zip file from the previous command onto Lambda itself. You could upload the zip using the AWS web console, but we provide this convenience helper so you can iterate on your code as quickly as you can type.

`--skip-libs` skips uploading the native libraries that the Swift function depends on. This makes for a much faster deployment process. Note that this flag is temporary and will be automatically calculated and hence removed soon.

~~`--new-version` lets you take advantage of AWS Lambda function versioning. It will increment your function's version number with this latest upload.~~

swift lambda logs [--tail]

`logs` allows you to see everything you `print()` in your Lambda function from the comfort of your terminal. This can be invaluable while debugging to see what is going on.

`--tail` will output log lines to your terminal as they happen in real-time.

swift lambda destroy

`destroy` deletes all artifacts of your Lambda function on AWS. Maybe you're done with a dev version of your code and ready to start a new project - stop paying for storage and cluttering up your Lambda web console.

swift lambda debug

`debug` allows you to step through your Lambda function from within Xcode, as if you were running it locally.

swift lambda invoke [--async] [--local]

`invoke` will execute your Lambda function and return its output to your terminal.

`--async` can be used for long-running Lambda functions where you don't want to wait for it to finish. You can still monitor its progress using `logs`.

`--local` will run your Lambda function inside a Lambda-like Docker container on your own computer. This might be preferable if you have no Internet connectivity and can't upload your function to AWS.

Function structure

Beyond the Swift code itself, swift-lambda also needs a `swift-lambda.json` file to know a bit about your intentions for this code. It is a JSON file that has some of the values required for publishing your code to Lambda. The structure is as follows:

```
{
    "Name": "Swifty McLambdaface",
    "Description": "My awesome new backend function written in Swift",
    "Memory": 128,
    "Timeout": 30
}
```

You will also have a `Package.swift` as per Swift Package Manager conventions. It can be as simple as:

```
import PackageDescription

let package = Package(
    name: "lambdaface",
    dependencies: [
        .Package(url: "https://github.com/awswift/swift-lambda-runtime",␣
↪majorVersion: 0, minor: 1)
    ]
)
```

Finally, the last required piece is `Sources/main.swift`. This is the entrypoint to the Swift executable that swift-lambda will call. An example file would be:

```
import Foundation
import SwiftLambdaRuntime

SwiftLambdaRuntime().run { event, context, callback in
    let name = event["name"] ?? "World"
    callback(["output": "Hello, \(name)"])
}
```

CHAPTER 2

swift-lambda.json manifest

Projects utilising the swift-lambda project include a `swift-lambda.json` manifest file in their root directory alongside SwiftPM's `Package.swift`. This file is mandatory and includes information that swift-lambda needs in order to successfully deploy your function to AWS Lambda.

Basic

In its basic form, this is a JSON file that looks like the following:

```
{
    "Name": "Swifty McLambdaface",
    "Description": "My awesome new backend function written in Swift",
    "Memory": 128,
    "Timeout": 30,
    "AptDependencies": [
        "openssl-dev"
    ]
}
```

- The `Name` field is how we refer to our function. AWS Lambda lets you define as many functions as you need, so being able to refer to them by name is helpful.
- The `Description` field is similarly useful for distinguishing between Lambda functions when you have many defined.
- The `Memory` field is mandatory and should be an integer between 128 and 1536. This is how much memory Lambda will make available to your Swift program while it runs. CPU performance is also allocated to your function proportionate to the amount of memory allocated.
- The `Timeout` field is a mandatory integer between 1 and 300. This is how many seconds that Lambda will wait before considering your function to have timed out and be forcibly terminated. You are billed per 100ms of actual runtime, not how much time you allocate.
- The `AptDependencies` field is optional. It is an array of native dependencies that your Swift program (or its packages) needs in order to compile.

Advanced

Eventually your needs may exceed that provided by the basic manifest file format. In this case, you can upgrade to using an AWS CloudFormation template to describe your function and infrastracture-as-code.

CloudFormation allows you to describe database tables, push notification topics, S3 file upload buckets and any other AWS resources you might need in addition to your Lambda function to support your mobile application.

Here is an example of a CloudFormation swift-lambda manifest file:

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Metadata": {
        "Name": ""
    },
    "Resources": {

    },
}
```

CloudFormation rationale

Our rationale for exposing raw CloudFormation is that a) you shouldn't have to learn a new standard when an existing one will do and b) we don't want to reinvent the wheel.

CHAPTER 3

Required IAM permissions

In order to operate correctly, swift-lambda needs permission to access certain AWS APIs on your behalf. If you're a solo dev and you've set up the AWS CLI using `aws configure`, you will have unlimited access and running swift-lambda from the terminal should work just fine.

However, if swift-lambda is running on a CI machine or your AWS access has been granted by your company's AWS gurus, your access may be restricted. In order to operate fully, swift-lambda requires the following permissions. They are described in an AWS IAM Policy document in JSON format below. This policy covers everything you need to run swift-lambda's `setup`, `deploy`, `invoke` and `destroy` commands.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1482554544203",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:ListExports",
                "cloudformation:UpdateStack",
                "iam:AttachRolePolicy",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:DetachRolePolicy",
                "iam:GetRole",
                "iam:PassRole",
                "iam:PutRolePolicy",
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:InvokeFunction",
                "lambda:UpdateFunctionCode",
```

```
                "lambda:UpdateFunctionConfiguration",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "s3:CreateBucket",
                "s3:DeleteBucket",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutBucketVersioning",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

CHAPTER 4

Continuous Integration

swift-lambda uses a self-hosted Jenkins instance its continuous integration system. It is hosted at jenkins.awswift.ge.cx and is publicly accessible.

All pull requests to swift-lambda are built and have unit tests and a series of checks run against them. The results of these tests are posted to GitHub on their respective PR threads, but are also available directly on Jenkins.

The CI process that is run against each commit is defined in the swift-lambda repo itself in the Jenkinsfile in the root of the repo. Here you can also see the battery of checks that each PR is subjected to. Essentially if swiftlint reports any style issues, `xcodebuild` has any build warnings (or errors!) or `swift test` is unhappy the Awswift bot will let you know via the PR thread on GitHub.

CHAPTER 5

swift-lambda Licence

```
                              Apache License
                        Version 2.0, January 2004
                     http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

  "License" shall mean the terms and conditions for use, reproduction,
  and distribution as defined by Sections 1 through 9 of this document.

  "Licensor" shall mean the copyright owner or entity authorized by
  the copyright owner that is granting the License.

  "Legal Entity" shall mean the union of the acting entity and all
  other entities that control, are controlled by, or are under common
  control with that entity. For the purposes of this definition,
  "control" means (i) the power, direct or indirect, to cause the
  direction or management of such entity, whether by contract or
  otherwise, or (ii) ownership of fifty percent (50%) or more of the
  outstanding shares, or (iii) beneficial ownership of such entity.

  "You" (or "Your") shall mean an individual or Legal Entity
  exercising permissions granted by this License.

  "Source" form shall mean the preferred form for making modifications,
  including but not limited to software source code, documentation
  source, and configuration files.

  "Object" form shall mean any form resulting from mechanical
  transformation or translation of a Source form, including but
  not limited to compiled object code, generated documentation,
  and conversions to other media types.

  "Work" shall mean the work of authorship, whether in Source or
```

```
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:
```

```
8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```