報道各位

2019 年 9 月 18 日
株式会社インプレスR&D
https://nextpublishing.jp/

AWS Lambda のカスタムランタイムを使いこなす！

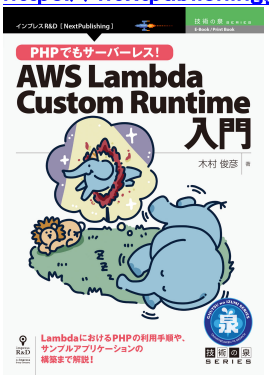
『PHP でもサーバーレス！AWS Lambda Custom Runtime 入門』発行

技術の泉シリーズ、9 月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレス R&D は、『PHP でもサーバーレス！AWS Lambda Custom Runtime 入門』(著者:木村 俊彦)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『PHPでもサーバーレス！AWS Lambda Custom Runtime入門』
https://nextpublishing.jp/isbn/9784844378068



著者:木村 俊彦
小売希望価格:電子書籍版 1600 円(税別)／印刷書籍版 1800 円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5 判／カラー／本文 104 ページ
ISBN:978-4-8443-7806-8
発行:インプレス R&D

＜＜発行主旨・内容紹介＞＞

AWS Lambda の新しい機能として、好きな言語でサーバーレス関数を実行できるカスタムランタイム(Lambda Custom Runtime)が追加されました。本書は、このカスタムランタイムを PHP で使うための解説書です。

基本的な実行方法から独自のランタイムを作成し利用する方法まで紹介し、さらにいくつかの PHP フレームワークを実際に Lambda 上で動かすまでの流れも解説しています。

PHP 以外の言語でカスタムランタイムを実行する際にも役に立つ一冊です。

(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

カスタムランタイムのカスマイズやビルド、デプロイについて丁寧に解説

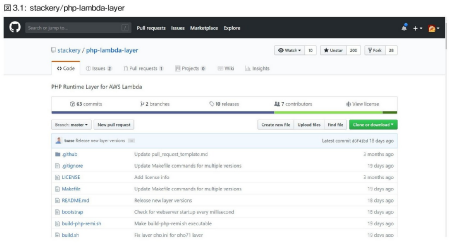
第3章　応用して使ってみよう

前章であらかじめ用意した実行環境で動作させましたが、そのままではモジュールの追加や設定変更が行えず不便です。本章ではベースとなるツールを使用して、カスタムランタイムのカスタマイズやビルド、デプロイを行ってみます。

3.1　必要なものを用意する

カスタムランタイムをPHPで動作させる手段は多々あります[1]が、今回はstackery/php-lambda-layer[2]を使用します。

図3.1: stackery/php-lambda-layer



これはサーバーレスアプリケーションのモニタリングツールを開発しているStackery社が作成したツールです。ビルドに必要なスクリプトやカスタムランタイム自体を動作させるのに必要なものがまとめて入っています（2019年7月現在では、まだ試験中のためプロダクション向きではないとの記述があります）。

このリポジトリーをgit cloneするか、zipで一式をダウンロードして展開します。

1: 今回使用するものの他に、Bref（https://bref.sh/）もあります
2:https://github.com/stackery/php-lambda-layer

3.2　PHP7.1をビルドする

準備ができたら早速ビルドします。ビルドにはスクリプトを使用しますが、スクリプトはリポジトリー内にbuild.shとbuild-php-remi.shのふたつがあります。今回はbuild.shを使用してPHP7.1の実行環境を作成します[3]。

スクリプトはAmazon Linuxベースの Dockerコンテナ内で動かす必要があります。しかし、Makefileがあるためmakeが利用できる環境であれば、次のコマンドでDockerコンテナの起動からzip圧縮までの一連の動作を行うことができます。

```
$ make php71.zip
```

Windowsなど、makeが利用できない環境の場合は、次のDockerコマンドでビルドが行えます。[4]

```
PS> docker run --rm -v ${PWD}:/opt/layer lambci/lambda:build-nodejs8.10
/opt/layer/build.sh
```

実行後、php71.zipが生成されていれば問題なく実行できています。

build.shの内容を見てみる

build.shの内容を見て、どのようにビルドしているか見てみたいと思います。まず初めに、全体像は次のようになっています。

リスト3.1: build.sh

```
#!/bin/bash
yum install -y php71-mbstring.x86_64 zip php71-pgsql php71-mysqli

mkdir /tmp/layer
cd /tmp/layer
cp /opt/layer/bootstrap .
sed "s/PHP_MINOR_VERSION/1/g" /opt/layer/php.ini >php.ini

mkdir bin
cp /usr/bin/php bin/

mkdir lib
for lib in libncurses.so.5 libtinfo.so.5 libpcre.so.0; do
  cp "/lib64/${lib}" lib/
```

3:build-php-remi.shはPHP7.3の環境を作成します
4. コマンドはPowerShellで実行する場合です。$(PWD)の部分を実行する環境に応じて置き換えます

AWS のマネジメントコンソールではなく、コマンドでデプロイする方法を紹介

定のみ行っています。

```
PHPRuntimeAppAPI:
  Type: AWS::Serverless::Api
  Properties:
    StageName: v1
```

次のリソースで関数の名前や保存先などを定義しています。Runtimeをprovidedにすることでカスタムランタイムで実行されます。レイヤーの設定はLayersの下に配列でレイヤーARNを入れています。

```
PHPRuntimeAppFunction:
  Type: AWS::Serverless::Function
  Properties:
    FunctionName: PHPRuntimeFunction
    CodeUri: src
    Runtime: provided
    Handler: index.php
    Layers:
      - （レイヤーARN）
```

Eventsの下に関数に紐付く各種イベントを定義します。ここでは初めに設定したAPI Gatewayを紐付けていて、手動で実行した時と同じくすべてのメソッドでのリクエストを許可しています。

```
    Events:
      api:
        Type: Api
        Properties:
          RestApiId: !Ref PHPRuntimeAppAPI
          Path: /
          Method: ANY
```

続けて、実行するスクリプトを用意します。今回はphpinfo()ではなく、少しPHPらしいスクリプトを使います。作業ディレクトリー内にsrcディレクトリーを作成し、その中に次の内容でindex.phpを作成します。

リスト4.2: index.php

```
Hello World! Runnning <?php echo $_SERVER['SERVER_SOFTWARE'] ?> on Lambda <?php
echo $_ENV['AWS_LAMBDA_FUNCTION_NAME'] ?> Function!
```

今度は文字を表示するだけですが、Lambda上で実行していなければ取れない環境変数が含まれているものです。これでファイルの準備は完了しました。

S3バケットの作成

SAMでデプロイする場合、S3にバケットを作成する必要があります。SAMのテンプレートからCloudFormationのテンプレートへ変換する過程で、ソースコードをS3へアップロードする必要があるためです。

バケットもAWS CLIで作成できるため、次のコマンドで作成します。3章で作成した時と同様に、アカウント内ではなくリージョン内の全アカウントでユニークである必要があるため注意が必要です。

```
$ aws s3api create-bucket --bucket （バケット名） --create-bucket-configuration
LocationConstraint=ap-northeast-1
```

実行して、エラーなく処理が終われば完了です。バケット名は控えておいてください。

デプロイする

準備が整いましたので、デプロイを行います。最初にテンプレートの形式変換とスクリプトのS3へのデプロイを行う必要があるため、次のコマンドを実行します。

```
$ sam package --template-file template.yaml --output-template-file output.yaml
--s3-bucket （バケット名）
```

「Successfully packaged artifacts and wrote output template to file output.yaml」と表示されれば成功です。そのまま次のコマンドで、テンプレート本体のデプロイを行います。実行することでAWSの各種リソースの作成が行われるため、少し待つ必要があります。

```
$ sam deploy --template-file output.yaml --stack-name CustomRuntimeApp
--capabilities CAPABILITY_IAM
```

「Successfully created/updated stack」と表示されれば成功です。

さっそくAPIを確認します。API Gatewayのコンソールに「CustomRuntimeApp」というAPIが

応用編としてフレームワークを Lambda 上で動作させる方法を解説

図 5.4: Laravel のデフォルトトップページ

Laravel

DOCS LARACASTS NEWS BLOG NOVA FORGE GITHUB

5.7 Phalcon

最後は Phalcon です。Phalcon は数あるフレームワークの中で最速と呼ばれていますが、その理由はフレームワーク本体が PHP の拡張として C 言語で書かれており、起動時にすべて読み込んでしまうためです。そんな風変わりのフレームワークなので、動かすのも一苦労ですが頑張って動かします。

最初に専用のレイヤーのビルドを行います。先ほど書いたように PHP の拡張として書かれているため、レイヤーの中に埋め込む必要があるためです。

php.ini は次の内容にします。必須モジュールに加えて、Phalcon 自体も読み込んでいることが分かります。

リスト 5.11: php.ini

```
extension_dir=/opt/lib/php/7.PHP_MINOR_VERSION/modules
display_errors=On

extension=curl.so
extension=json.so
extension=mbstring.so
extension=mysqlnd.so
extension=pgsql.so
extension=pdo.so
extension=pdo_mysql.so
extension=gettext.so
extension=fileinfo.so
extension=gd.so
extension=phalcon.so
```

build.sh は次の内容です。Phalcon 自体をビルドしたり、php71-gd に必要なライブラリーをコピーするよう追記しています。

リスト 5.12: build.sh

```
#!/bin/bash

yum reinstall -y libedit
yum install -y php71-mbstring.x86_64 zip php71-pgsql php71-mysqli php71-devel php
71-gd

git clone https://github.com/phalcon/cphalcon
cd cphalcon
git checkout tags/v3.4.3
cd build
./install

mkdir /tmp/layer
cd /tmp/layer
cp /opt/layer/bootstrap .
sed "s/PHP_MINOR_VERSION/1/g" /opt/layer/php.ini >php.ini

mkdir bin
cp /usr/bin/php bin/

mkdir lib
for lib in libncurses.so.5 libtinfo.so.5 libpcre.so.0; do
  cp "/lib64/${lib}" lib/
done

cp /usr/lib64/libedit.so.0 lib/
cp /usr/lib64/libpq.so.5 lib/
cp /usr/lib64/libXpm.so.4 lib/

cp -a /usr/lib64/php lib/

zip -r /opt/layer/php71.zip .
```

加えて、bootstrap も調整する必要があります。124 行目から 144 行目にかけてリクエストパラ

82 | 第 5 章 フレームワークを使ってみよう

第 5 章 フレームワークを使ってみよう | 83

＜＜目次＞＞

＜＜著者紹介＞＞

木村 俊彦
宮城県出身・在住。学生時代からプログラミングに勤しみ、そのままシステム開発会社へ就職。その後 Web 制作会社を経て、現在は PHP や Node.js のアプリケーション開発や AWS や Azure の設計構築を担当。最近は Docker や Kubernetes などのコンテナ周りや IoT 関係に興味あり。
PHP カンファレンス仙台コアスタッフ、技術同人誌サークル「杜の都の開発室」主宰。

＜＜販売ストア＞＞

電子書籍:
Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER
印刷書籍:
Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス
※ 各ストアでの販売は準備が整いしだい開始されます。
※ 全国の一般書店からもご注文いただけます。

フォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp