

The AWS logo, consisting of a white cube icon followed by the lowercase letters "aws" in a white, sans-serif font.

DEV DAY

GLOBAL SERIES



サーバレスで 王道 Web フレームワークを 使う方法

Amazon Web Services Japan K.K.
Solutions Architect, **Akihiro Tsukada**

2017.06.02

THANKS TO OUR FRIENDS AT:



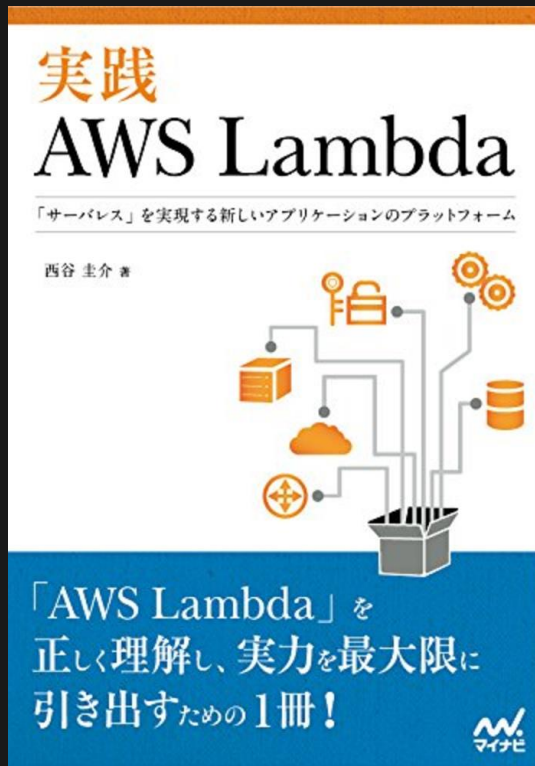
本セッションのFeedbackをお願いします

受付でお配りしたアンケートに本セッションの満足度やご感想などをご記入ください
アンケートをご提出いただきました方には、もれなく**素敵なAWSオリジナルグッズ**を
プレゼントさせていただきます



アンケートは各会場出口、パミール3FのEXPO展示会場内にて回収させていただきます

AWS Lambdaの本が出ます



- 👤 AWS Lambdaを網羅した本を出します
- 👤 2017年6月9日マイナビ出版より出版予定
- 👤 3,240円（税込）
- 👤 Amazonで予約受け付け中
<http://amzn.asia/ew2WWPm>
- 👤 AWS Summit 本会場のExpoエリア奥、マイナビ出版様ブースで先行販売中



👉 つかだ あきひろ

👉 スタートアップ

モバイル

サーバレス

ブロックチェーン

👉 健康診断全部A 🙄



 @akitsukada

 karaage.jackey

アジェンダ

- 🦊 このセッションについて
- 🦊 [github/awslabs](#)
- 🦊 [Express.js](#)
- 🦊 [Spring Framework](#)
- 🦊 [AWS CodeStar](#)
- 🦊 まとめ

このセッションについて

- 🔥 解決したい課題
- 🔥 セッションの対象者
- 🔥 このセッションで話さないこと

解決したい課題

- 👤 「AWS Lambda と Amazon API Gateway でバックエンドを全部作ってしまえないものか」
- 👤 「でも Lambda で大きい Web アプリを開発するのって大変なんじゃない？」
- 👤 多くの Lambda ファンクションをどう管理すれば？
- 👤 デプロイや運用のベストプラクティスは？
- 👤 「いつものフレームワークが使えるれば楽なのに 😞」

このセッションの対象者

- 👤 一般的な Web フレームワークを使ったアプリケーション開発の経験がある方
- 👤 Lambda と API Gateway は知っているが Web アプリはいつも EC2 上に構築する方
- 👤 便利なのはいいけど複雑なのは勘弁という方


このセッションで話さないこと


- 🦊 各サービス・機能・フレームワーク等の基礎的な説明
 - 🦊 AWS Lambda, Amazon API Gateway, AWS CloudFormation, Amazon S3 etc
 - 🦊 Swagger, Express.js, Spring Framework
- 🦊 継続的な運用、テスト、CI/CDパイプラインの詳細
 - 🦊 本日の各セッションを参照のこと



github / awslabs

https://github.com/awslabs

 [Features](#) [Business](#) [Explore](#) [Marketplace](#) [Pricing](#) This organization [Sign in](#) or [Sign up](#)



Amazon Web Services - Labs

Seattle, WA <http://aws.amazon.com/>

[Repositories](#) People 35

Pinned repositories

aws-shell
An integrated shell for working with the AWS CLI.
Python ★ 2.9k 🍴 155

s2n
s2n : an implementation of the TLS/SSL protocols
C ★ 2.8k 🍴 243

chalice
Python Serverless Microframework for AWS
Python ★ 2.7k 🍴 199

Type: All ▾ Language: All ▾

aws-codedeploy-samples
Samples and template scenarios for AWS CodeDeploy
Shell ★ 354 🍴 243 Updated 38 minutes ago

Top languages

- Python
- Java
- JavaScript
- Ruby
- HTML

<https://github.com/awslabs>

- 👉 AWS が実験的・先進的なお役立ちツール、サンプル、デベロッパープレビューなライブラリなどを OSS として開発・公開している GitHub リポジトリ群
 - 👉 2017年6月現在300強のリポジトリが存在
 - 👉 対して github/aws はより Fixed なものの置き場所

Express.js と Java Container も awslabs に

Features Business Explore Marketplace Pricing This repository Search Sign in or Sign up

awslabs / aws-serverless-express Watch 33 Star 723 Fork 83

Code Issues 15 Pull requests 1 Projects 0 Insights

Run serverless applications and REST APIs using your existing Node.js application framework, on top of AWS Lambda and Amazon API Gateway

aws-serverless

109 commits 1 branch 6 releases 13 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

| File | Commit | Time |
|---------------|--|--------------|
| __tests__ | Eaddrinuse (#71) | 13 days ago |
| example | update example dependencies including aws-serverless-express to 3.0.0 | 12 days ago |
| .gitignore | update to use SAM, and add deconfigure script | 4 months ago |
| .shintrc | Binary support for API Gateway. | 4 months ago |
| .travis.yml | Not running test on io.js | 4 months ago |
| LICENSE | Initial commit | 8 months ago |
| NOTICE.txt | Add library | 8 months ago |
| README.md | fix: decodes base64 requests (fixes #64); added form to index view to... | 18 days ago |
| index.js | renaming err as error to log createServer errors correctly (#79) | 5 days ago |
| middleware.js | Eaddrinuse (#71) | 13 days ago |
| package.json | 3.0.1 | 5 days ago |

Features Business Explore Marketplace Pricing This repository Search Sign in or Sign up

awslabs / aws-serverless-java-container Watch 26 Star 140 Fork 24

Code Issues 7 Pull requests 1 Projects 0 Wiki Insights

A Java wrapper to run Spring, Jersey, Spark, and other apps inside AWS Lambda.
<https://aws.amazon.com/serverless/>

jersey api-gateway aws-lambda spring serverless aws api api-server rest-api sparkjava sparkjava-framework

89 commits 3 branches 3 releases 8 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

SAPessi committed on GitHub Merge pull request #32 from awslabs/servlet-improvements Latest commit 9d16cff 18 days ago

| | | |
|--------------------------------------|---|--------------|
| aws-serverless-java-container-core | Changed encodeURL logic in servlet response | 20 days ago |
| aws-serverless-java-container-jer... | Added Jersey ServletContext factory to address #30 | 20 days ago |
| aws-serverless-java-container-sp... | [maven-release-plugin] prepare for next development iteration | a month ago |
| aws-serverless-java-container-sp... | [maven-release-plugin] prepare for next development iteration | a month ago |
| samples | Merge pull request #29 from awslabs/cognito-user-pool | 20 days ago |
| .gitignore | First import | 5 months ago |
| .travis.yml | Adding travis build file | a month ago |
| LICENSE | Initial commit | 6 months ago |
| README.md | Added Jersey ServletContext factory to address #30 | 20 days ago |
| pom.xml | [maven-release-plugin] prepare for next development iteration | a month ago |

aws-serverless-express

aws-serverless-java-container

awslabs にあるリポジトリ例

- 👉 <https://github.com/awslabs/git-secrets>
 - 👉 AWS Access Key や ~/.aws/credentials ファイルの誤コミットを防止
- 👉 <https://github.com/awslabs/s2n>
 - 👉 シンプルで高速な TLS/SSL 実装
- 👉 <https://github.com/awslabs/aws-{\platform}-sample>
 - 👉 各言語やフレームワークなどの実装サンプル集
 - 👉 swift, android, machinelearning, serverless...
- 👉 <https://github.com/awslabs/chalice>
 - 👉 非常にコンパクトな Python 製の Serverless フレームワーク

Express.js on Serverless

 **How to Get Started**
始め方を知る

 **How It Works**
動作原理を知る

Express.js on Serverless

 **How to Get Started**
始め方を知る

 **How It Works**
動作原理を知る

<https://expressjs.com/>

Express 4.15.3

Fast, unopinionated,
minimalist web
framework for Node.js

Express.js – How to Get Started

- 🐱 awslabs の example を走らせてみる – on Local/AWS
- 🐱 既存の Express プロジェクトを
aws-serverless-express にマイグレーションする

Express.js – 1. example を走らせてみる

🐱 ソースコードの取得

```
$ git clone https://github.com/awslabs/aws-serverless-express
```

```
$ cd aws-serverless-express/example
```

```
$ ls -1
```

```
README.md
```

```
api-gateway-event.json
```

```
app.js
```

```
app.local.js
```

```
cloudformation.yaml
```

```
lambda.jspackage.json
```

```
sam-logo.png
```

```
scripts
```

```
simple-proxy-api.yaml
```

```
vanilla-server.js
```

```
views
```

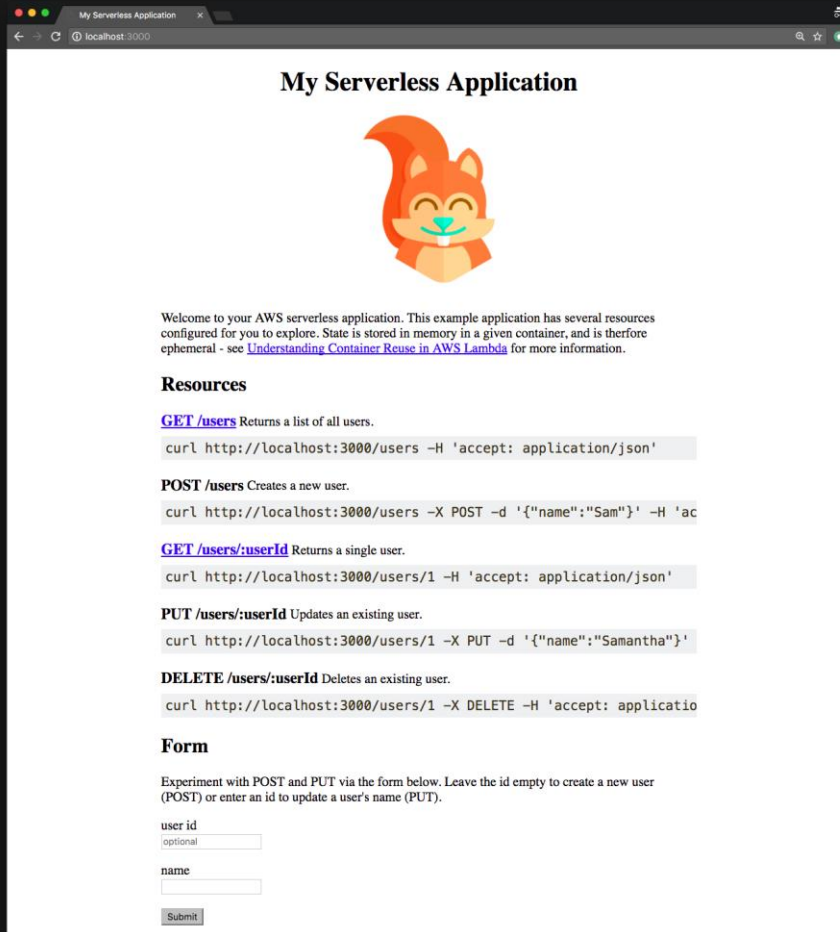
Express.js – 1. example を走らせてみる

🦊 ローカルで実行

```
$ npm install && npm run start  
(snip)  
listening on http://localhost:3000
```

<http://localhost:3000> =>

※ ローカルで開発したい場合は
Node.js ≥ 4 が必要



The screenshot shows a web browser window titled "My Serverless Application" with the URL "localhost:3000". The page content includes:

- My Serverless Application** (title)
- A fox logo (the same as the one in the text above).
- Introductory text: "Welcome to your AWS serverless application. This example application has several resources configured for you to explore. State is stored in memory in a given container, and is therefore ephemeral - see [Understanding Container Reuse in AWS Lambda](#) for more information."
- Resources** section:
 - GET /users**: Returns a list of all users. Example curl: `curl http://localhost:3000/users -H 'accept: application/json'`
 - POST /users**: Creates a new user. Example curl: `curl http://localhost:3000/users -X POST -d '{"name":"Sam"}' -H 'accept: application/json'`
 - GET /users/:userId**: Returns a single user. Example curl: `curl http://localhost:3000/users/1 -H 'accept: application/json'`
 - PUT /users/:userId**: Updates an existing user. Example curl: `curl http://localhost:3000/users/1 -X PUT -d '{"name":"Samantha"}'`
 - DELETE /users/:userId**: Deletes an existing user. Example curl: `curl http://localhost:3000/users/1 -X DELETE -H 'accept: application/json'`
- Form** section:

Experiment with POST and PUT via the form below. Leave the id empty to create a new user (POST) or enter an id to update a user's name (PUT).

user id
optional

name

Express.js – 1. example を走らせてみる

🔥 AWS上で実行

```
$ npm run config -- --account-id="<accountId>" ¥  
--bucket-name="<bucketName>" ¥  
[--region="<region>" --function-name="<functionName>"]
```

region, function-name を省略すると

us-east-1, AwsServerlessExpressFunction になる

```
$ npm run setup # Windowsの場合 npm run win-setup  
(snip) Successfully created/updated stack – AwsServerlessExpressStack
```

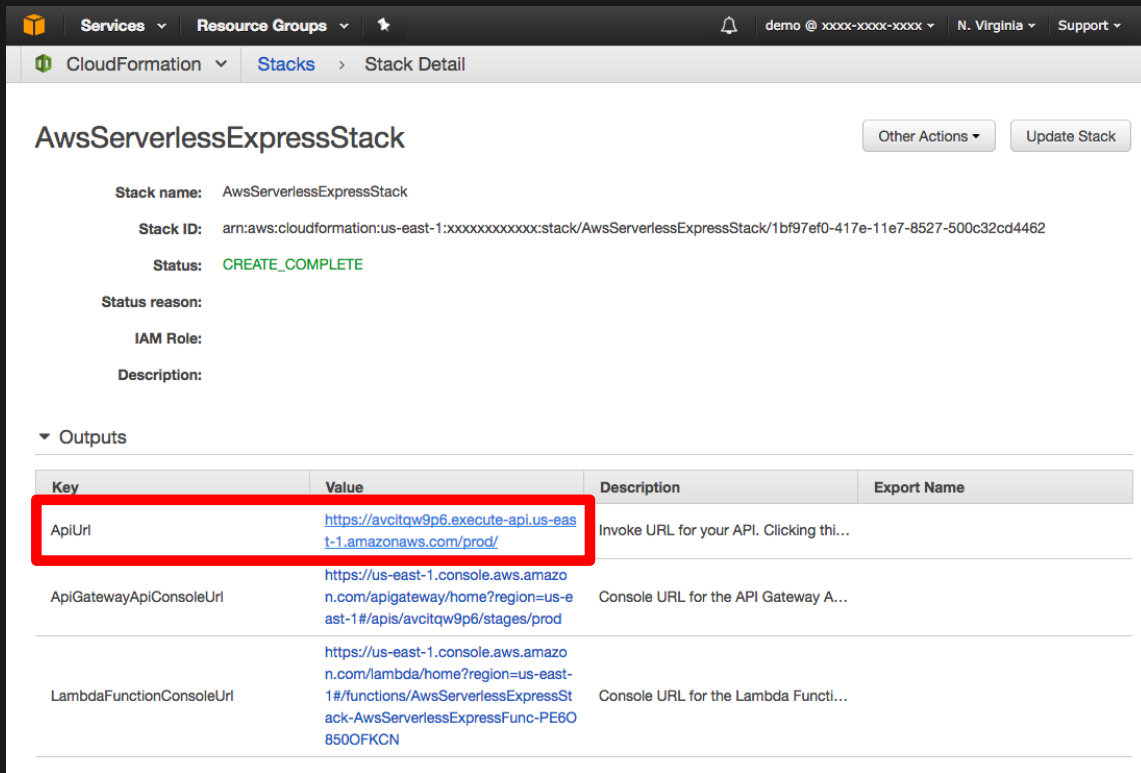
Tips:

"ZIP does not support timestamps before 1980" なエラーが出たときは

find . -mtime +1200 -exec touch {} ¥; のように回避できる

Express.js – 1. example を走らせてみる

👉 デプロイされた URL を確認して開く



Services Resource Groups CloudFormation Stacks Stack Detail

AwsServerlessExpressStack [Other Actions] [Update Stack]

Stack name: AwsServerlessExpressStack
Stack ID: arn:aws:cloudformation:us-east-1:xxxxxxxxxxxx:stack/AwsServerlessExpressStack/1bf97ef0-417e-11e7-8527-500c32cd4462
Status: **CREATE_COMPLETE**
Status reason:
IAM Role:
Description:

▼ Outputs

| Key | Value | Description | Export Name |
|--------------------------|---|--|-------------|
| ApiUrl | https://avc1qw9p6.execute-api.us-east-1.amazonaws.com/prod/ | Invoke URL for your API. Clicking thi... | |
| ApiGatewayApiConsoleUrl | https://us-east-1.console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/avc1qw9p6/stages/prod | Console URL for the API Gateway A... | |
| LambdaFunctionConsoleUrl | https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/AwsServerlessExpressStack-AwsServerlessExpressFunc-PE6O8500FKCN | Console URL for the Lambda Functi... | |

CloudFormation Home



AwsServerlessExpressStack

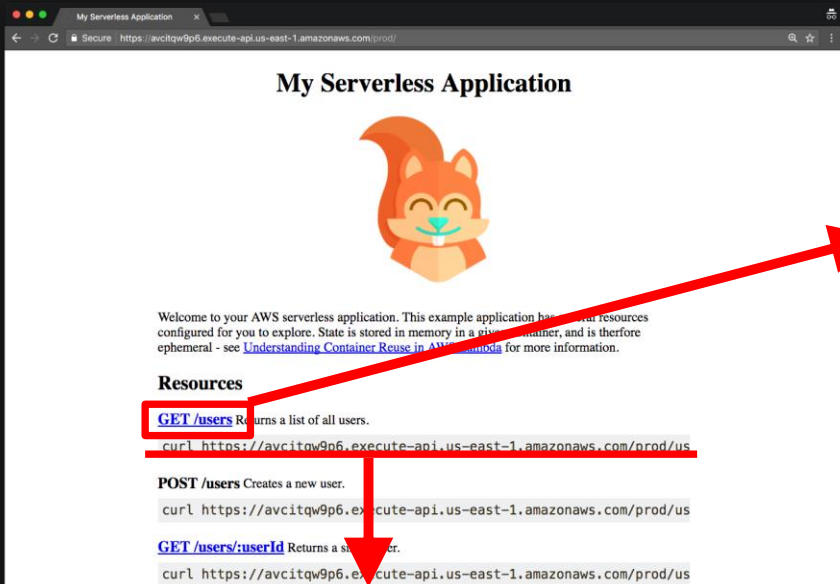


Outputs




ApiUrl を選択

Express.js – 1. example を走らせてみる



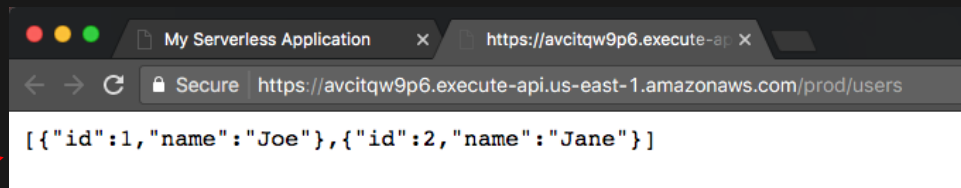
My Serverless Application



Welcome to your AWS serverless application. This example application has several resources configured for you to explore. State is stored in memory in a given container, and is therefore ephemeral - see [Understanding Container Reuse in AWS Lambda](#) for more information.

Resources

- GET /users** Returns a list of all users.
`curl https://avcitzw9p6.execute-api.us-east-1.amazonaws.com/prod/users`
- POST /users** Creates a new user.
`curl https://avcitzw9p6.execute-api.us-east-1.amazonaws.com/prod/users`
- GET /users/:userId** Returns a single user.
`curl https://avcitzw9p6.execute-api.us-east-1.amazonaws.com/prod/users/:userId`



```
[{"id":1,"name":"Joe"}, {"id":2,"name":"Jane"}]
```

```
$ curl https://avcitzw9p6.execute-api.us-east-1.amazonaws.com/prod/users -H ¥  
'accept: application/json'  
[{"id":1,"name":"Joe"}, {"id":2,"name":"Jane"}]
```

Form


Experiment with POST and PUT via the form below. Leave the id empty to create a new user (POST) or enter an id to update a user's name (PUT).

user id

name

Express.js – 1. example を走らせてみる

My Serverless Application



GET https://(snip)/sam
バイナリファイルに対応

Welcome to your AWS serverless application. This example application has several resources configured for you to explore. State is stored in memory in a given container, and is therefore ephemeral - see [Understanding Container Reuse in AWS Lambda](#) for more information.

Resources

[GET /users](#) Returns a list of all users.
`curl https://avc1qw9p6.execute-api.us-east-1.amazonaws.com/prod/us`

POST /users Creates a new user.
`curl https://avc1qw9p6.execute-api.us-east-1.amazonaws.com/prod/us`


[GET /users/:userId](#) Returns a single user.
`curl https://avc1qw9p6.execute-api.us-east-1.amazonaws.com/prod/us`

PUT /users/:userId Updates an existing user.
`curl https://avc1qw9p6.execute-api.us-east-1.amazonaws.com/prod/us`

DELETE /users/:userId Deletes an existing user.

optional

name



```
$ curl -Iso /dev/null https://avc1qw9p6.execute-api.us-east-1.amazonaws.com/prod ¥  
-w '{content_type}'  
text/html; charset=utf-8
```

HTMLに対応

Express.js – How to Get Started

~~👾 awslabs の example を走らせてみる – on Local/AWS~~

👾 既存の Express プロジェクトを
aws-serverless-express にマイグレーションする

Express.js – 2. 既存プロジェクトのマイグレート

1. example から必要なファイルをコピー

```
$ cp -R ¥  
api-gateway-event.json ¥ # ローカルテスト用のモックデータ  
cloudformation.yaml ¥ # CFn テンプレート  
lambda.js ¥ # Lambda フังก์ションの handler  
simple-proxy-api.yaml ¥ # Swagger の API GW 定義ファイル  
scripts ¥ # 便利なスクリプト群  
$existing_express_dir # 既存プロジェクトのディレクトリ
```

2. example の package.json から既存プロジェクトの package.json へ、scripts と config の記述をコピー

参考 <https://github.com/aws-labs/aws-serverless-express/tree/master/example>

Express.js – 2. 既存プロジェクトのマイグレート

3. (未実行の場合) 設定スクリプトを実行

```
$ npm run config -- --account-id="<accountId>" ¥  
--bucket-name="<bucketName>" ¥  
[--region="<region>" --function-name="<functionName>"]
```

4. npm install の実行

```
$ npm install --save aws-serverless-express
```

5. デプロイの実行

```
$ npm run package-deploy  
# Windows の場合 npm run win-package-deploy
```

参考 <https://github.com/awslabs/aws-serverless-express/tree/master/example>

Express.js – 2. 既存プロジェクトのマイグレート

Tips (1/2)

👉 既存プロジェクトのメインファイル名が `app.js` でない場合、`lambda.js` の `require` を既存プロジェクトのファイル名に合わせる

👉 例: `require('./app')` → `require('./server')`

👉 ソースコードをトップ以外のディレクトリに配置している場合、`cloudformation.yml` 内の `CodeUri` で指定する

👉 例: `CodeUri: ./` → `CodeUri: ./src`

👉 Webpack 等の build ツールを使っている場合はそのアウトプットディレクトリを指定

Express.js – 2. 既存プロジェクトのマイグレート

Tips (2/2)

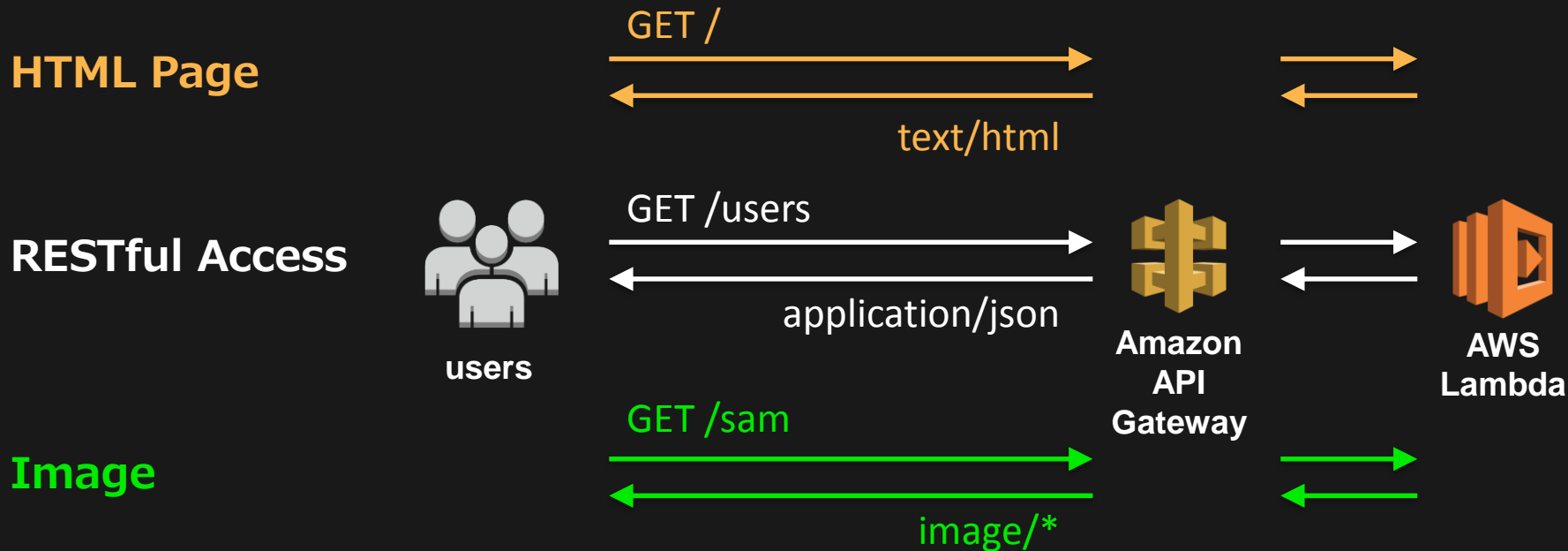
- 👤 ローカルの Node サーバで Lambda のテストをするには `npm run local` を実行
 - 👤 `api-gateway-event.json` のデータが `event` として渡される
 - 👤 Lambda とローカルの Node.js バージョンを合わせることを推奨
- 👤 API Gateway の構成を変更したい場合は `simple-proxy-api.yaml` を、AWS リソースの構成を変更したい場合は `cloudformation.yaml` を編集して `npm run package-deploy` を実行

Express.js on Serverless

✓ How to Get Started
始め方を知る

 How It Works
動作原理を知る

Express.js – aws-serverless-express の振る舞い



Express.js – aws-serverless-express の構造

```
/  
ANY  
OPTIONS  
/{proxy+}  
ANY  
OPTIONS
```

Lambda
 Integration Proxy

Binary media:
/



lambda.js
(handler)

app.js

```
app.get('/', (req, res) => {  
  res.render('index', {  
    (snip)  
  })  
})  
  
app.get('/sam', (req, res) => {  
  res.sendFile(  
    `_${__dirname}/sam-logo.png`  
  )  
})  
  
app.get('/users', (req, res) => {  
  res.json(users)  
})
```



Express.js – aws-serverless-express の構造

/ に対する
すべての
メソッド、
/* に対する
すべての
メソッド

/
ANY
OPTIONS
/{proxy+}
ANY
OPTIONS

Lambda
 Integration
Proxy

Binary media:
/



lambda.js
(handler)

app.js

```
app.get('/', (req, res) => {  
  res.render('index', {  
    (snip)  
  })  
})  
  
app.get('/sam', (req, res) => {  
  res.sendFile(  
    `_${__dirname}/sam-logo.png`  
  )  
})  
  
app.get('/users', (req, res) =>  
{  
  res.json(users)  
})
```



Express.js – aws-serverless-express の構造

レスポンス
ヘッダは
Lambdaで
生成する

```
/  
ANY  
OPTIONS  
/{proxy+}  
ANY  
OPTIONS
```

**Lambda
Integration
Proxy**

Binary media:
/



lambda.js
(handler)

app.js

```
app.get('/', (req, res) => {  
  res.render('index', {  
    (snip)  
  })  
})  
  
app.get('/sam', (req, res) => {  
  res.sendFile(  
    `_${__dirname}/sam-logo.png`  
  )  
})  
  
app.get('/users', (req, res) =>  
{  
  res.json(users)  
})
```



Express.js – aws-serverless-express の構造

```
/  
ANY  
OPTIONS  
/{proxy+}  
ANY  
OPTIONS
```

Lambda
 Integration Proxy

Binary media:
/



lambda.js
(handler)

app.js

```
app.get('/', (req, res) => {  
  res.render('index', {  
    (snip)  
  })  
})  
  
app.get('/sam', (req, res) => {  
  res.sendFile(  
    `_${__dirname}/sam-logo.png`  
  )  
})  
  
app.get('/users', (req, res) => {  
  {  
    res.json(users)  
  }  
})
```



画像ファイル
をレスポンス
できるように

Express.js – aws-serverless-express の構造

複数の .js ファイルを
含む一つの Lambda
ファンクション

lambda.js
(handler)

app.js

```
app.get('/', (req, res) => {
  res.render('index', {
    (snip)
  })
})

app.get('/sam', (req, res) => {
  res.sendFile(
    `_${__dirname}/sam-logo.png`)
})

app.get('/users', (req, res) => {
  res.json(users)
})
```



Express.js – GET / のシーケンス(HTML)

GET /

text/html

/

ANY

OPTIONS

/{proxy+}

ANY

OPTIONS

Lambda

Integration
Proxy

Binary media:

/



lambda.js
(handler)

app.js

```
app.get('/', (req, res) => {  
  res.render('index', {  
    (snip)  
  })  
})
```

res.render メソッドは
Content-Type:text/html
を返す Express.js のメソッド

@see <https://expressjs.com/en/api.html>



Express.js – GET /sam のシーケンス(Image)

GET /sam

image/png

/

ANY

OPTIONS

/{proxy+}

ANY

OPTIONS

Lambda



Integration
Proxy

Binary media:

/



lambda.js
(handler)

app.js

```
app.get('/sam', (req, res) => {  
  res.sendFile(  
    `${__dirname}/sam-logo.png`  
  )  
})
```

res.sendFile メソッドは
拡張子に応じて
image/(png|jpeg|gif…)
を返す

@see <https://expressjs.com/en/api.html>



Express.js – GET /users のシーケンス (RESTful API)

GET /users

application/json

/
ANY
OPTIONS
/{proxy+}
ANY
OPTIONS

Lambda
 Integration
Proxy

Binary media:
/



lambda.js
(handler)

app.js

res.json メソッドは
拡張子に応じて

application/json
を返す

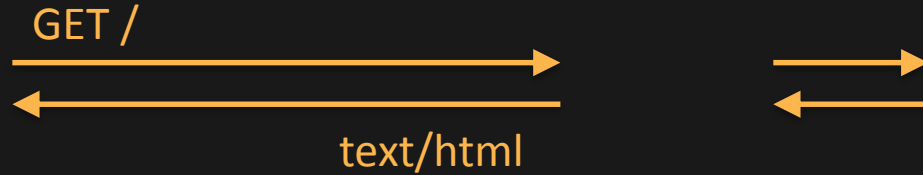
@see <https://expressjs.com/en/api.html>

```
app.get('/users', (req, res) => {  
  res.json(users)  
})
```

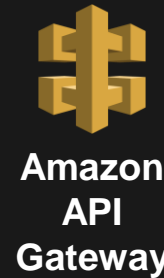
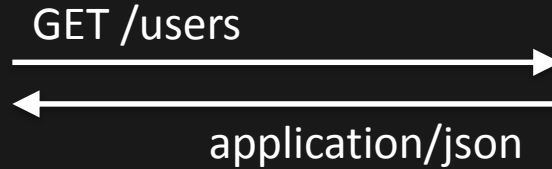


Express.js – aws-serverless-express の振る舞い

✓ HTML Page



✓ RESTful Access



✓ Image



Express.js on Serverless

- ✓ **How to Get Started**
始め方を知る
- ✓ **How It Works**
動作原理を知る



Spring Framework

- 🧙 How to Get Started
始め方を知る
- 🧙 How It Works
動作原理を知る



Spring Framework

- 🧙 **How to Get Started**
始め方を知る
- 🧙 **How It Works**
動作原理を知る

awslabs / aws-serverless-java-container

GitHub - awslabs/aws-serverless-java-container

Features Business Explore Marketplace Pricing

This repository Search Sign in or Sign up

awslabs / aws-serverless-java-container

Watch 26 Star 142 Fork 24

Code Issues 7 Pull requests 1 Projects 0 Wiki Insights

A Java wrapper to run Spring, Jersey, Spark, and other apps inside AWS Lambda.
<https://aws.amazon.com/serverless/>

jersey api-gateway aws-lambda spring serverless aws api api-server rest-api sparkjava sparkjava-framework

89 commits 3 branches 3 releases 8 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

SAPessi committed on GitHub Merge pull request #32 from awslabs/servlet-improvements Latest commit 9d16cff 23 days ago

- aws-serverless-java-container-core Changed encodeURL logic in servlet response 25 days ago
- aws-serverless-java-container-jer... Added Jersey ServletContext factory to address #30 25 days ago
- aws-serverless-java-container-sp... [maven-release-plugin] prepare for next development iteration 2 months ago
- aws-serverless-java-container-sp... [maven-release-plugin] prepare for next development iteration 2 months ago
- samples Merge pull request #29 from awslabs/cognito-user-pool 25 days ago
- .gitleignore First import 6 months ago
- .travis.yml Adding travis build file a month ago
- LICENSE Initial commit 6 months ago
- README.md Added Jersey ServletContext factory to address #30 25 days ago
- pom.xml [maven-release-plugin] prepare for next development iteration 2 months ago



API Gateway と
Lambda で



Spring



Jersey



Spark

などを動作させるための
ラッパーライブラリ

参考 <https://github.com/awslabs/aws-serverless-java-container/wiki>

Spring Framework – How to Get Started

- 🐱 awslabs の sample(pet-store) を走らせてみる
- 🐱 既存の Spring プロジェクトに
aws-serverless-java-container を適用する

Spring Framework – pet-store on Serverless

🐱 ソースコードの取得

```
$ git clone https://github.com/aws-labs/aws-serverless-java-container
```

```
$ cd aws-serverless-java-container/samples/spring/pet-store
```

```
$ ls -l
```

```
README.md
```

```
output-sam.yaml
```

```
pom.xml
```

```
sam.yaml
```

```
src
```

```
target
```


Spring Framework – pet-store on Serverless

The screenshot shows an IDE window titled "workspace-sts-3.8.4.RELEASE - Spring - pet-store/src/main/java/com/amazonaws/serverless/sample/spring/PetsC". The Package Explorer on the left shows the project structure:

- pet-store [aws-serverless-java-container master]
 - src/main/java
 - com.amazonaws.serverless.sample.spring
 - LambdaHandler.java
 - PetsController.java
 - PetStoreSpringAppConfig.java
 - StreamLambdaHandler.java
 - com.amazonaws.serverless.sample.spring.model
 - Error.java
 - Pet.java
 - PetData.java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - output-sam.yaml
 - pom.xml
 - README.md
 - sam.yaml

The main editor displays the code for `PetsController.java`:

```
20 * Copyright 2016 Amazon.com, Inc. or its affiliates. All Rights Reserved.
13 package com.amazonaws.serverless.sample.spring;
14
15 import com.amazonaws.serverless.sample.spring.model.Pet;
22
23 @RestController
24 @EnableWebMvc
25 public class PetsController {
26     @RequestMapping(path = "/pets", method = RequestMethod.POST)
27     public Pet createPet(@RequestBody Pet newPet) {
28         if (newPet.getName() == null || newPet.getBreed() == null) {
29             return null;
30         }
31
32         Pet dbPet = newPet;
33         dbPet.setId(UUID.randomUUID().toString());
34         return dbPet;
35     }
36
37     @RequestMapping(path = "/pets", method = RequestMethod.GET)
38     public Pet[] listPets(@RequestParam("limit") Optional<Integer> limit) {
39         int queryLimit = 10;
40         if (limit.isPresent()) {
41             queryLimit = limit.get();
42         }
43     }
44 }
```

Spring Framework – pet-store on Serverless

🐿 Maven で package を作成

```
$ mvn package
```

```
[INFO] Scanning for projects...
```

```
[INFO]
```

```
[INFO] -----
```

```
[INFO] Building Spring example for the aws-serverless-java-container library 1.0-SNAPSHOT
```

```
...(snip)...
```

```
[INFO] -----
```

```
[INFO] BUILD SUCCESS
```

```
[INFO] -----
```

```
[INFO] Total time: 39.695 s
```

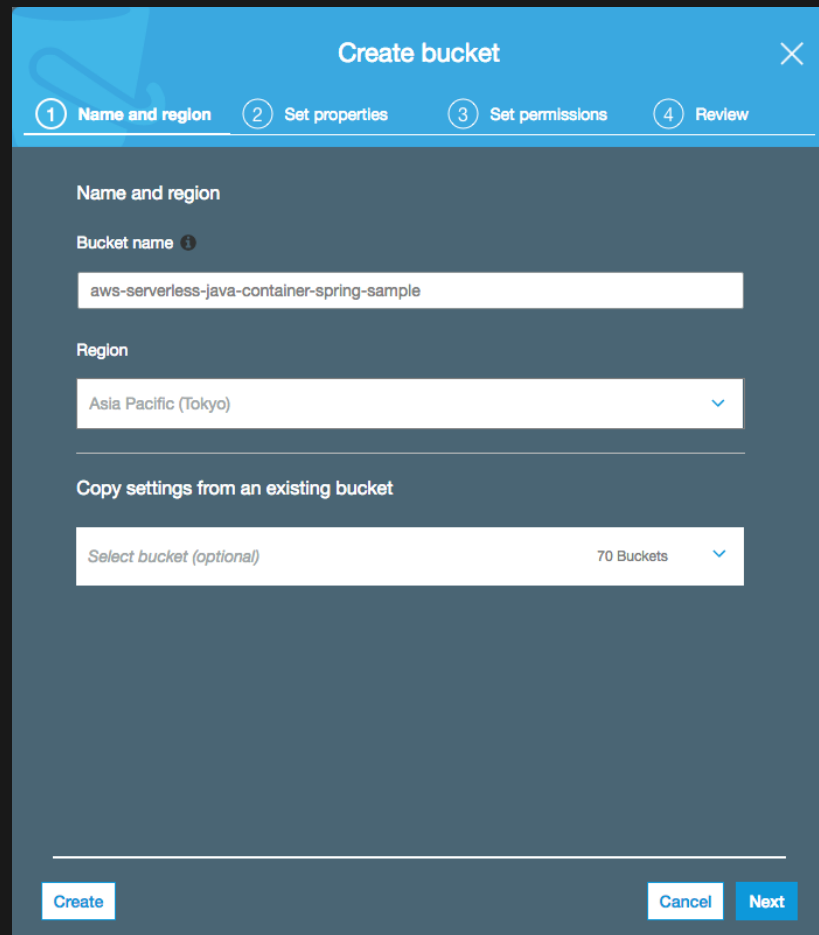
```
[INFO] Finished at: 2017-05-25T15:26:32+09:00
```

```
[INFO] Final Memory: 33M/243M
```

```
[INFO] -----
```

Spring Framework – pet-store on Serverless

🦊 CloudFormation の
artifact を置く
S3 バケットを作成



The screenshot shows the 'Create bucket' wizard in the AWS Management Console. The title bar is blue with a close button (X) on the right. Below the title bar is a progress indicator with four steps: 1. Name and region (highlighted), 2. Set properties, 3. Set permissions, and 4. Review. The main content area is dark blue and contains the following fields:

- Name and region**
 - Bucket name**: A text input field containing 'aws-serverless-java-container-spring-sample'.
 - Region**: A dropdown menu showing 'Asia Pacific (Tokyo)' with a downward arrow.
- Copy settings from an existing bucket**
 - A dropdown menu showing 'Select bucket (optional)' and '70 Buckets' with a downward arrow.

At the bottom of the form, there are three buttons: 'Create' (highlighted in blue), 'Cancel', and 'Next'.

Spring Framework – pet-store on Serverless

CloudFormation package の作成

```
$ aws cloudformation package --template-file sam.yaml ¥
```

```
--output-template-file output-sam.yaml ¥
```

```
--s3-bucket <YOUR S3 BUCKET NAME>
```

```
Uploading to 26175c16dff8870d7dc34225dc7aa47a 7553653 / 7553653.0 (100.00%)
```

```
Successfully packaged artifacts and wrote output template to file output-sam.yaml.
```

```
Execute the following command to deploy the packaged template
```

```
aws cloudformation deploy --template-file /Path/to/output-sam.yaml --stack-name <YOUR STACK NAME>
```

デプロイ実行用のコマンドが示される

Spring Framework – pet-store on Serverless

👉 デプロイ実行

```
$ aws --region ap-northeast-1 cloudformation deploy ¥  
  --template-file /Path/to/output-sam.yaml ¥  
  --stack-name <YOUR STACK NAME> [--capabilities CAPABILITY_IAM]
```

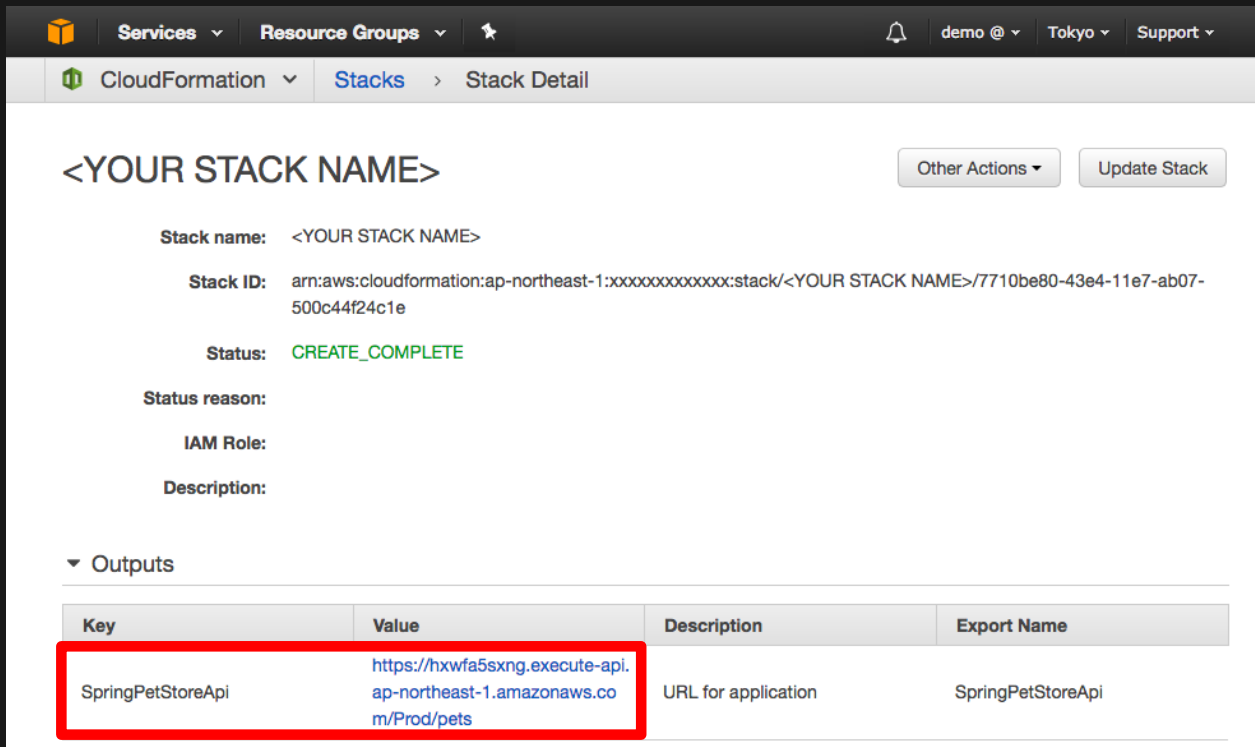
Waiting for changeset to be created..

Waiting for stack create/update to complete

Successfully created/updated stack - <YOUR STACK NAME>

Spring Framework – pet-store on Serverless

🐱 デプロイされた URL を確認して開く



Services ▾ Resource Groups ▾ 🔍 demo @ ▾ Tokyo ▾ Support ▾

CloudFormation ▾ Stacks > Stack Detail

<YOUR STACK NAME> Other Actions ▾ Update Stack

Stack name: <YOUR STACK NAME>

Stack ID: arn:aws:cloudformation:ap-northeast-1:xxxxxxxxxxxx:stack/<YOUR STACK NAME>/7710be80-43e4-11e7-ab07-500c44f24c1e

Status: CREATE_COMPLETE

Status reason:

IAM Role:

Description:

▾ Outputs

| Key | Value | Description | Export Name |
|-------------------|---|---------------------|-------------------|
| SpringPetStoreApi | https://hxwfa5sxng.execute-api.ap-northeast-1.amazonaws.com/Prod/pets | URL for application | SpringPetStoreApi |

CloudFormation Home



<YOUR STACK NAME>



Outputs

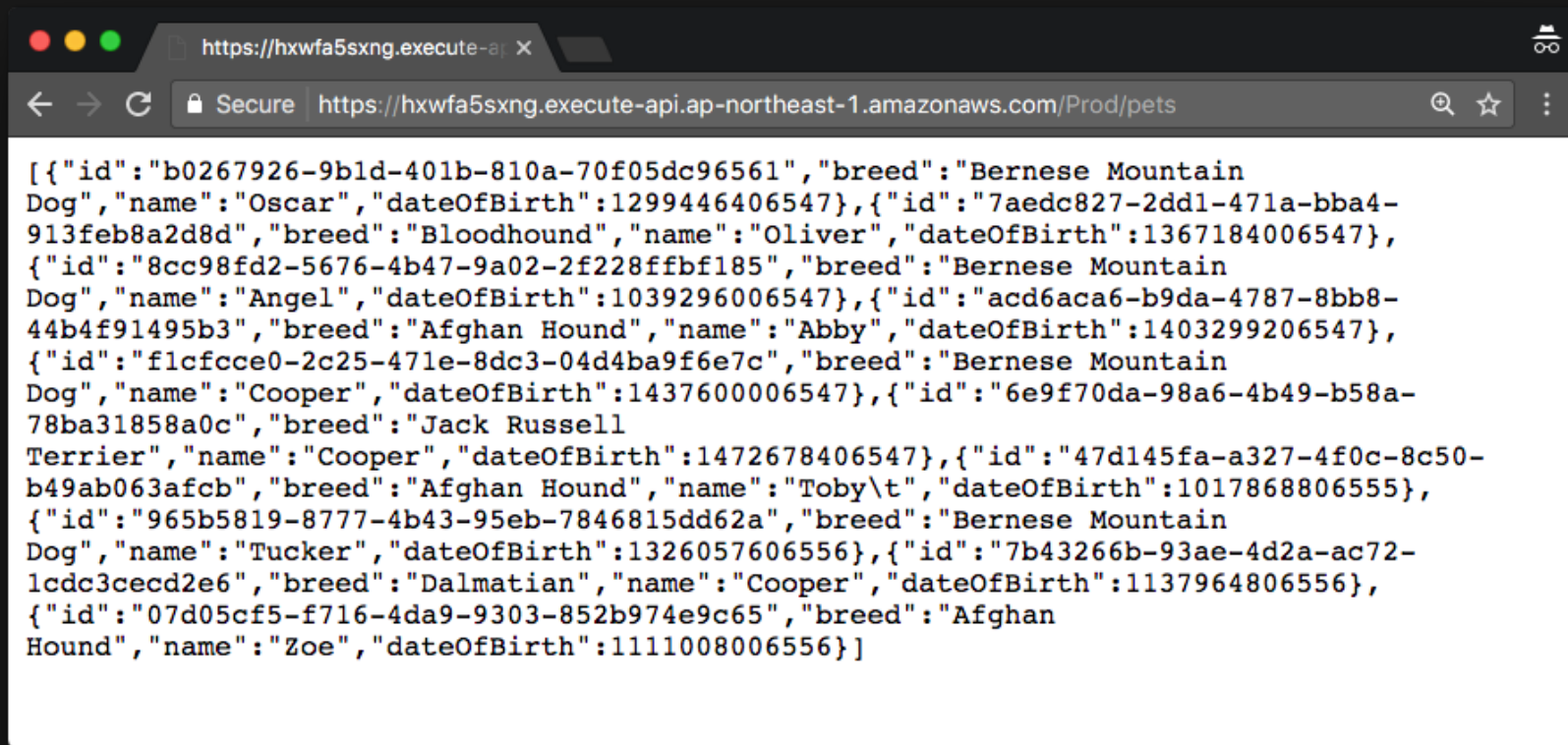


SpringPetStoreApi

を選択

Spring Framework – pet-store on Serverless

🐱 デプロイされた URL を確認して開く



The screenshot shows a web browser window with the address bar displaying the URL `https://hwxfa5sxng.execute-api.ap-northeast-1.amazonaws.com/Prod/pets`. The page content is a JSON array of pet objects. The JSON data is as follows:

```
[{"id": "b0267926-9b1d-401b-810a-70f05dc96561", "breed": "Bernese Mountain Dog", "name": "Oscar", "dateOfBirth": 1299446406547}, {"id": "7aedc827-2ddl-471a-bba4-913feb8a2d8d", "breed": "Bloodhound", "name": "Oliver", "dateOfBirth": 1367184006547}, {"id": "8cc98fd2-5676-4b47-9a02-2f228ffbf185", "breed": "Bernese Mountain Dog", "name": "Angel", "dateOfBirth": 1039296006547}, {"id": "acd6aca6-b9da-4787-8bb8-44b4f91495b3", "breed": "Afghan Hound", "name": "Abby", "dateOfBirth": 1403299206547}, {"id": "f1fcfce0-2c25-471e-8dc3-04d4ba9f6e7c", "breed": "Bernese Mountain Dog", "name": "Cooper", "dateOfBirth": 1437600006547}, {"id": "6e9f70da-98a6-4b49-b58a-78ba31858a0c", "breed": "Jack Russell Terrier", "name": "Cooper", "dateOfBirth": 1472678406547}, {"id": "47d145fa-a327-4f0c-8c50-b49ab063afcb", "breed": "Afghan Hound", "name": "Toby", "dateOfBirth": 1017868806555}, {"id": "965b5819-8777-4b43-95eb-7846815dd62a", "breed": "Bernese Mountain Dog", "name": "Tucker", "dateOfBirth": 1326057606556}, {"id": "7b43266b-93ae-4d2a-ac72-1cdc3cecd2e6", "breed": "Dalmatian", "name": "Cooper", "dateOfBirth": 1137964806556}, {"id": "07d05cf5-f716-4da9-9303-852b974e9c65", "breed": "Afghan Hound", "name": "Zoe", "dateOfBirth": 1111008006556}]
```

Spring Framework – How to Get Started

- ~~🐱 awslabs の sample(pet-store) を走らせてみる~~
- 🐱 既存の Spring プロジェクトに
aws-serverless-java-container を適用する

Spring Framework – 既存プロジェクトへの適用

依存関係の設定 (pom.xml)

```
<dependency>  
  <groupId>com.amazonaws.serverless</groupId>  
  <artifactId>aws-serverless-java-container-spring</artifactId>  
  <version>LATEST</version>  
</dependency>
```

@see <https://github.com/aws-labs/aws-serverless-java-container/wiki/Quick-start---Spring>

Spring Framework – 既存プロジェクトへの適用

LambdaHandler の作成

```
public class LambdaHandler implements
    RequestHandler<AwsProxyRequest, AwsProxyResponse> {
    private SpringLambdaContainerHandler<AwsProxyRequest, AwsProxyResponse> handler;

    public AwsProxyResponse handleRequest(AwsProxyRequest awsProxyRequest,
        Context context) {
        if (handler == null) {
            try {
                handler = SpringLambdaContainerHandler.
                    getAwsProxyHandler(PetStoreSpringAppConfig.class);
            } catch (ContainerInitializationException e) {
                e.printStackTrace();
                return null;
            }
        }
        return handler.proxy(awsProxyRequest, context);
    }
}
```

Spring Framework – 既存プロジェクトへの適用

🦊 Lambda フังก์ションをパッケージングしてデプロイ

🦊 AWS Documentation "デプロイパッケージの作成 (Java)" を参照のこと

http://docs.aws.amazon.com/ja_jp/lambda/latest/dg/lambda-java-how-to-create-deployment-package.html

🦊 API Gateway、CloudFormation (AWS SAM) の作成や 設定は pet-store を参考にすることをオススメ

@see <https://github.com/aws-labs/aws-serverless-java-container/wiki/Quick-start---Spring>

Spring Framework – How to Get Started

- ~~🐾 awslabs の sample(pet-store) を走らせてみる~~
- ~~🐾 既存の Spring プロジェクトに
aws-serverless-java-container を適用する~~



Spring Framework

✓ How to Get Started
始め方を知る

🧙 How It Works
動作原理を知る

Spring Framework – How It Works

👉 バックエンドの構成、原理

→ Express.js の example とあまり変わらないので割愛

👉 API Gateway は /{proxy+} ANY で、Lambda Proxy Integration で…

👉 awslabs/aws-serverless-java-container

👉 spring, sample/pet-store の

(ソフトウェアとしての) アーキテクチャ俯瞰

👉 クラスレベル

👉 パッケージレベル

Spring on Serverless Classes (1/2)

com.amazonaws

serverless.sample.spring

```
LambdaHandler
+ handleRequest(...)
```

↓ Create

```
@Configuration,
@ComponentScan("...sample.spring")
PetStoreSpringAppConfig
```

↓ Scan

```
@RestController, @EnableWebMvc
PetsController

@RequestMapping(POST "/pets")
+ createPet(...):Pet

@RequestMapping(GET "/pets")
+ listPets(...):Pet[]

@RequestMapping(GET "/pets/{userId}")
+ listPets():Pet
```

Create

services.lambda.runtime

```
<<Interface>>RequestHandler
+ handleRequest(...)
```

serverless.proxy.spring

```
SpringLambdaContainerHandler
# handleRequest(...)
# getContainerResponse(...)
```

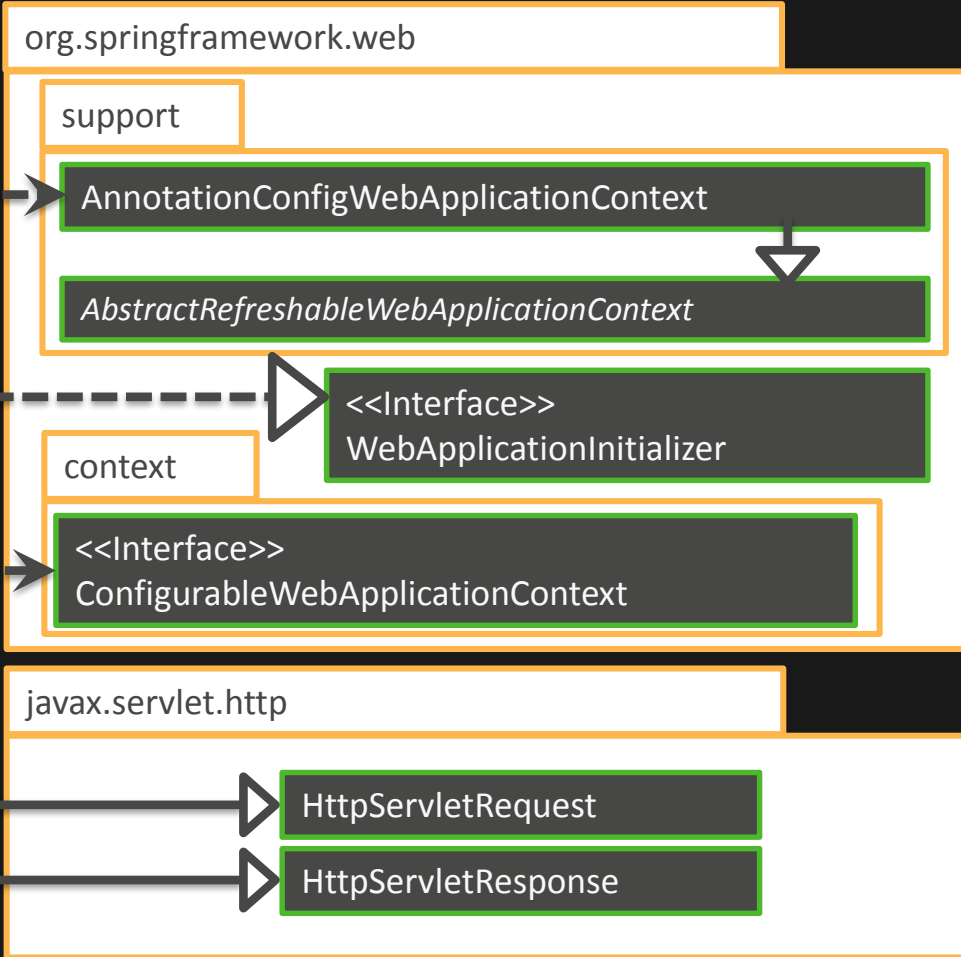
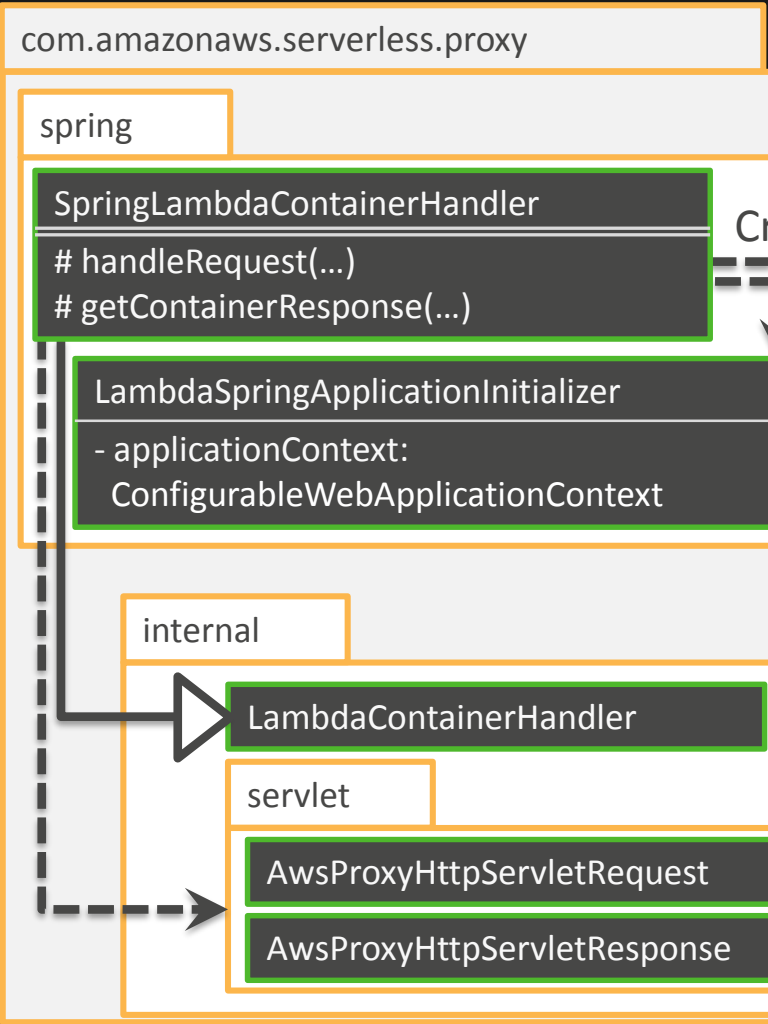
```
LambdaSpringApplicationInitializer
- applicationContext
```

serverless.proxy.internal

```
LambdaContainerHandler
+ proxy(...)
# handleRequest(...)
# getContainerResponse(...)
```



Spring on Serverless Classes (2/2)



Spring on Serverless アプリケーションアーキテクチャ

com.amazonaws.serverless.sample.spring

Pet-storeの実装

com.amazonaws.services.lambda.runtime

Lambda Java コアライブラリ

com.amazonaws.serverless.proxy.spring

Springのラッパーライブラリ

org.springframework.web

Spring Framework

com.amazonaws.serverless.proxy.internal

**Serverless Java Container
コアライブラリ**

javax.servlet.http

Servlet

Spring on Serverless アプリケーションアーキテクチャ

com.amazonaws.serverless.sample.spring

Pet-storeの実装

com.amazonaws.services.lambda.runtime

Lambda Java コアライブラリ

com.amazonaws.serverless.proxy.spring

Springのラッパーライブラリ

org.springframework.web

Spring Framework

com.amazonaws.serverless.proxy.internal

**Serverless Java Container
コアライブラリ**

javax.servlet.http

Servlet

従来どおりこれを

← 設計・実装すればよい

com.amazonaws.serverless.sample.spring

Pet-storeの実装

com.amazonaws.services.lambda.runtime

Lambda Java コアライブラリ

com.amazonaws.serverless.proxy.spring

Springのラッパーライブラリ

org.springframework.web

Spring Framework

com.amazonaws.serverless.proxy.internal

Serverless Java Container
コアライブラリ

javax.servlet.http

Servlet

アジェンダ

 ~~このセッションについて~~

 ~~github/awslabs~~

 ~~Express.js~~

 ~~Spring Framework~~

 AWS CodeStar

 まとめ



AWS CodeStar

Quickly develop, build, and deploy applications on AWS

<https://aws.amazon.com/jp/codestar/>

AWS CodeStar で Web フレームワーク on Serverless

Services Resource Groups demo @ N. Virginia Support

AWS CodeStar Create project

Select template Set up tools Start coding

express

Choose a project template

Start a new software project on AWS in minutes using a project template. [Help me choose](#)

Express.js

- Web application
- AWS Elastic Beanstalk (runs in a managed application environment)

Express.js

- Web application
- Amazon EC2 (runs on virtual servers that you manage)

Express.js

- Web service
- AWS Lambda (running serverless)

Express.js

- Web service
- Amazon EC2 (runs on virtual servers that you manage)

Application category

- Web application
- Web service
- Alexa Skill
- Static Website

Programming languages

- Ruby
- Node.js
- Java
- Python
- PHP
- HTML 5

AWS services

- AWS Elastic Beanstalk
- Amazon EC2
- AWS Lambda

express と入力



Express.js, Web service,
AWS Lambda
(running serverless)
を選択

AWS CodeStar で Web フレームワーク on Serverless

The screenshot shows the AWS CodeStar 'Create project' interface. At the top, there are navigation links for 'Services', 'Resource Groups', and a search icon. Below this, the breadcrumb 'AWS CodeStar > Create project' is visible. A progress bar indicates the current step is 'Select template', with 'Set up tools' and 'Start coding' as subsequent steps.

The main content area is titled 'Choose a project template' and includes the instruction: 'Start a new software project on AWS in minutes using a project template. [Help me choose](#)'. A search bar on the left contains the text 'Spring'. Below the search bar, there are filters for 'Application category' (Web application, Web service, Static Website) and 'Programming languages' (Ruby, Node.js, Java, Python, PHP, HTML 5). Under 'AWS services', there are checkboxes for 'AWS Elastic Beanstalk', 'Amazon EC2', and 'AWS Lambda'.

The template selection area displays four 'Java Spring' templates. The bottom-most template is highlighted with a red box. This template is categorized as a 'Web service' and is powered by 'AWS Lambda (running serverless)'. The other three templates are categorized as 'Web application' and are powered by either 'AWS Elastic Beanstalk (runs in a managed application environment)' or 'Amazon EC2 (runs on virtual servers that you manage)'.

Spring も対応

AWS CodeStar で Web フレームワーク on Serverless

Services Resource Groups

demo @ 1882-0532-4091 N. Virginia Support

AWS CodeStar Create project

Select template Set up tools Start coding

Project name Project ID [Edit](#)

AWS CodeStar includes all of the tools and services you need for a development project.
This project includes an AWS CodePipeline connected with the following tools:

Source Build Test Deploy Monitoring

AWS CodeCommit AWS CodeBuild AWS CloudFormation Amazon CloudWatch

AWS CodeStar would like permission to administer AWS resources on your behalf. [Learn more](#)

[Previous](#) [Create Project](#)

プロジェクト名を入力



Create Project ボタン
をポチっと

AWS CodeStar で Web フレームワーク on Serverless

AWS CodeStar > Create project

Select template Set up tools Start coding Provisioning 0% complete

Choose how you want to edit your project code
You can always change this choice after the project has been created

Visual Studio
Configure the AWS Toolkit for Visual Studio to edit your AWS CodeStar project code in Microsoft Visual Studio 2015 (or higher).

Eclipse
Configure the AWS Toolkit for Eclipse to edit your AWS CodeStar project code in Eclipse.

Command line tools
Edit AWS CodeStar project code by connecting directly to your project's Git source repository.

Clone repository URL
HTTPS We're creating your repository. It should be ready soon. Copy Credential details

Previous Skip

CloudFormation > Stacks

Create Stack Actions Design template

Filter: Active By Stack Name Showing 15 stacks

| Stack Name | Created Time | Status | Description |
|---|------------------------------|--------------------|------------------------------|
| <input checked="" type="checkbox"/> awscodestar-aa2express-on-l | 2017-05-26 16:12:21 UTC+0900 | CREATE_IN_PROGRESS | A Node.js web service deplc |
| <input type="checkbox"/> AwsServerlessExpressStack | 2017-05-26 14:12:35 UTC+0900 | CREATE_COMPLETE | Serverless Express Applicati |

Overview

Stack name: awscodestar-aa2express-on-l

Stack ID: arn:aws:cloudformation:us-east-1:xxxxxxxxxxxx:stack/awscodestar-aa2express-on-l/d7b4eed0-418e-11e7-b8e1-500c286374d1

Status: CREATE_IN_PROGRESS

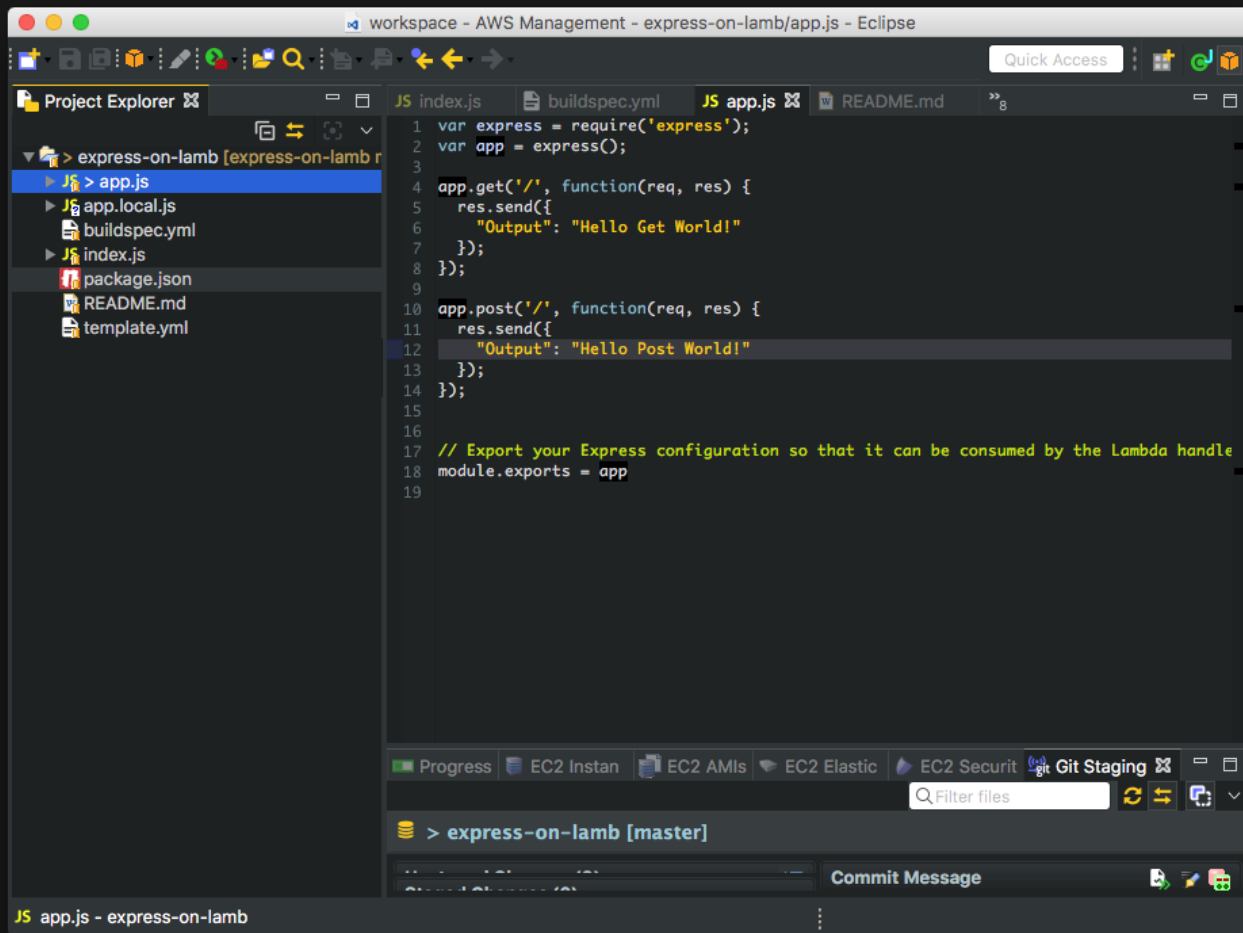
Status reason:

IAM Role:

Description: A Node.js web service deployed to AWS Lambda.

※ 生成されるのは awslabs のライブラリに含まれるサンプルとは別の実装

AWS CodeStar で Web フレームワーク on Serverless



The screenshot shows the Eclipse IDE interface with a workspace titled "workspace - AWS Management - express-on-lamb/app.js - Eclipse". The Project Explorer on the left shows a project named "express-on-lamb" with files like "app.js", "app.local.js", "buildspec.yml", "index.js", "package.json", "README.md", and "template.yml". The main editor displays the content of "app.js":

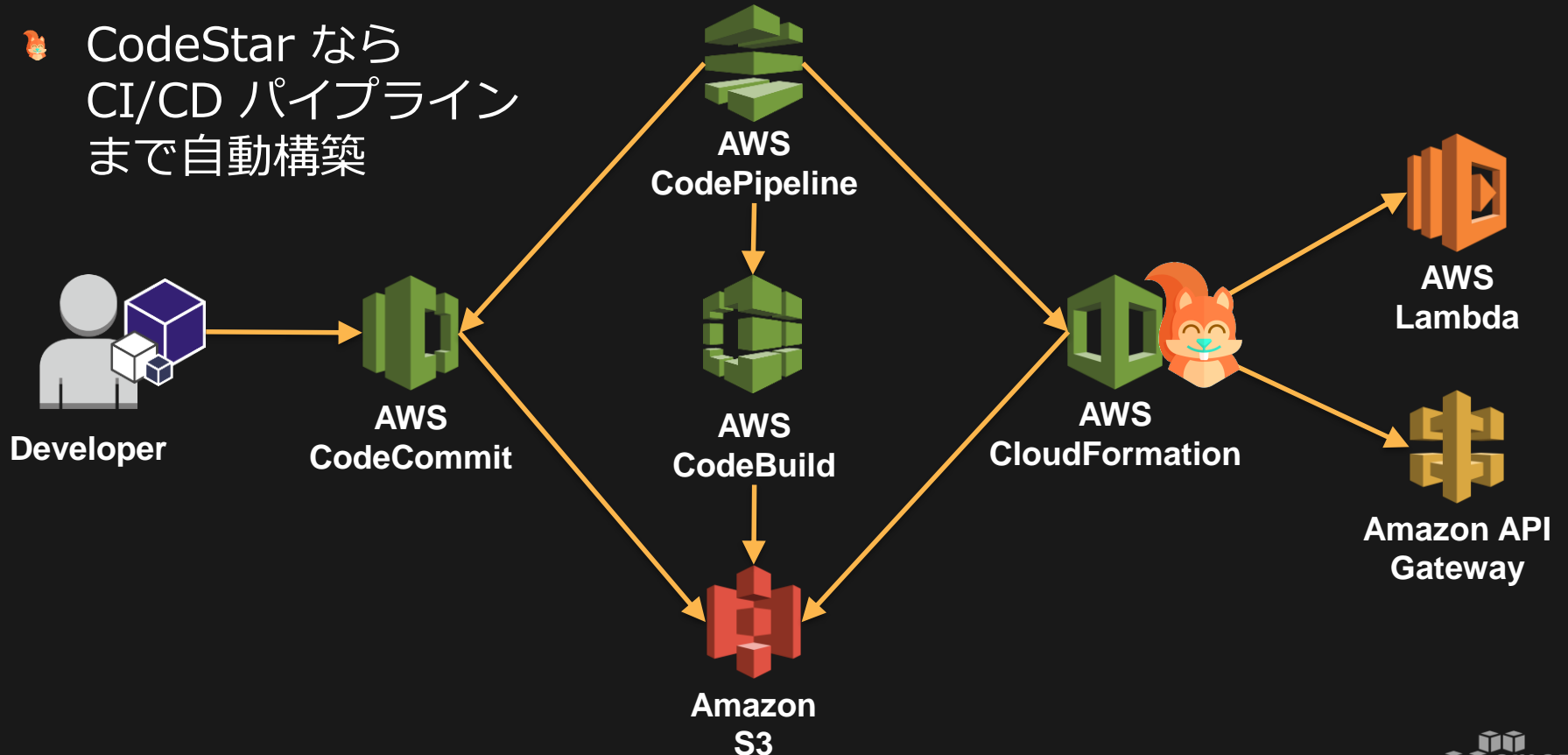
```
1 var express = require('express');
2 var app = express();
3
4 app.get('/', function(req, res) {
5   res.send({
6     "Output": "Hello Get World!"
7   });
8 });
9
10 app.post('/', function(req, res) {
11   res.send({
12     "Output": "Hello Post World!"
13   });
14 });
15
16
17 // Export your Express configuration so that it can be consumed by the Lambda handle
18 module.exports = app
19
```

At the bottom, the Git Staging view shows the current branch as "express-on-lamb [master]" and a "Commit Message" field.

**AWS Toolkit を
IDE に追加し
CodeStar プロジェクト
をインポート
↓
編集し Commit & Push
すればデプロイされる**

AWS CodeStar で Web フレームワーク on Serverless

🦊 CodeStar なら
CI/CD パイプライン
まで自動構築



まとめ



まとめ

- 👉 **github/awslabs** を見ると楽しい 🙌 ('ω' 🙌)三 🙌 ('ω') 🙌 三(🙌 'ω') 🙌
- 👉 **Express.js** と **Spring Framework** は
AWS 提供のライブラリや **AWS CodeStar** を使って
API Gateway と **Lambda** で従来どおりの開発が可能
- 👉 **awslabs** の **sample** を走らせるのも既存プロジェクトをマ
イグレーションするのも非常に簡単
- 👉 **AWS CodeStar** は **CI/CD** パイプラインまで自動構築

Thank You!

Don't Forget Evaluations!

The AWS logo, consisting of a white cube icon followed by the lowercase letters "aws" in a white, sans-serif font.

DEV DAY

GLOBAL SERIES