

[はじめに]

本ワークショップシナリオのオリジナル版は以下の URL になります。

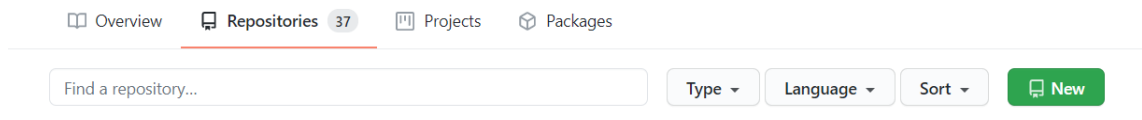
<https://www.apprunnerworkshop.com/>

本シナリオはオリジナル版を短縮し、App Runner の基本機能にフォーカスしています。余裕がある方は是非オリジナル版もお試してください。

1. GitHub にアクセスしてログインします。
2. [Repositories]をクリックします



3. [New]ボタンをクリックします



4. [simple-express-app]と入力し、[Create repository]ボタンをおします

github.com/new

Search or jump to...

Pulls Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

forkfork / simple-express-app ✓

Great repository names are short and memorable. Need inspiration? How about [improved-garbanzo?](#)

Description (optional)

☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☒ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

© 2021 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#)
[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

5. [creating a new file]のリンクをクリックします

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

6. [index.js]と入力し、以下のコマンドを貼り付けます

simple-express-app / index.js in main

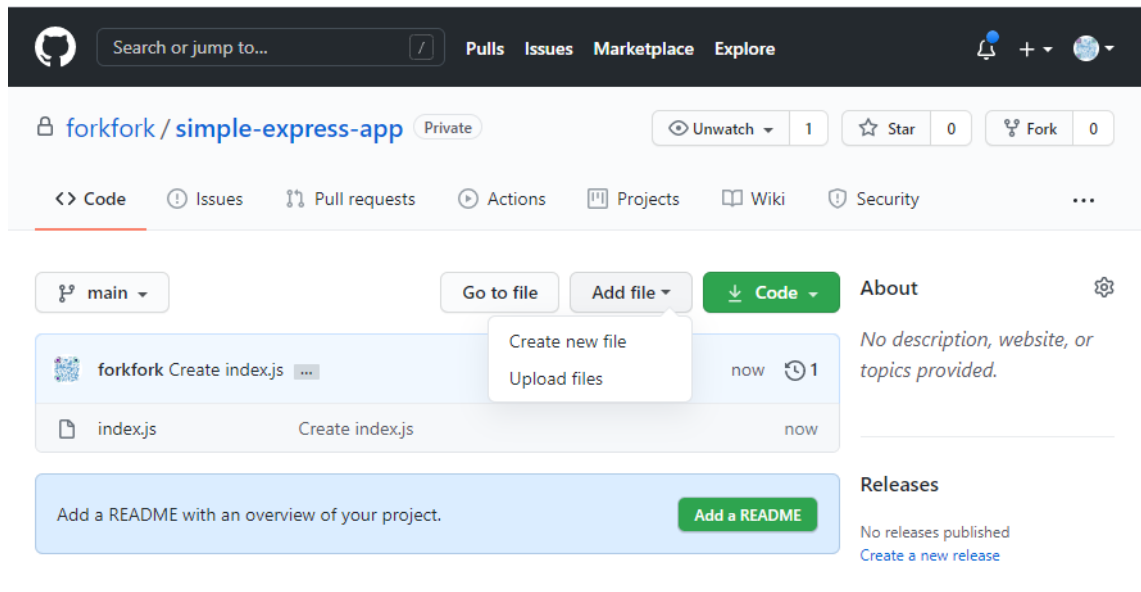
Cancel changes

<> Edit new file Preview Spaces 2 No wrap

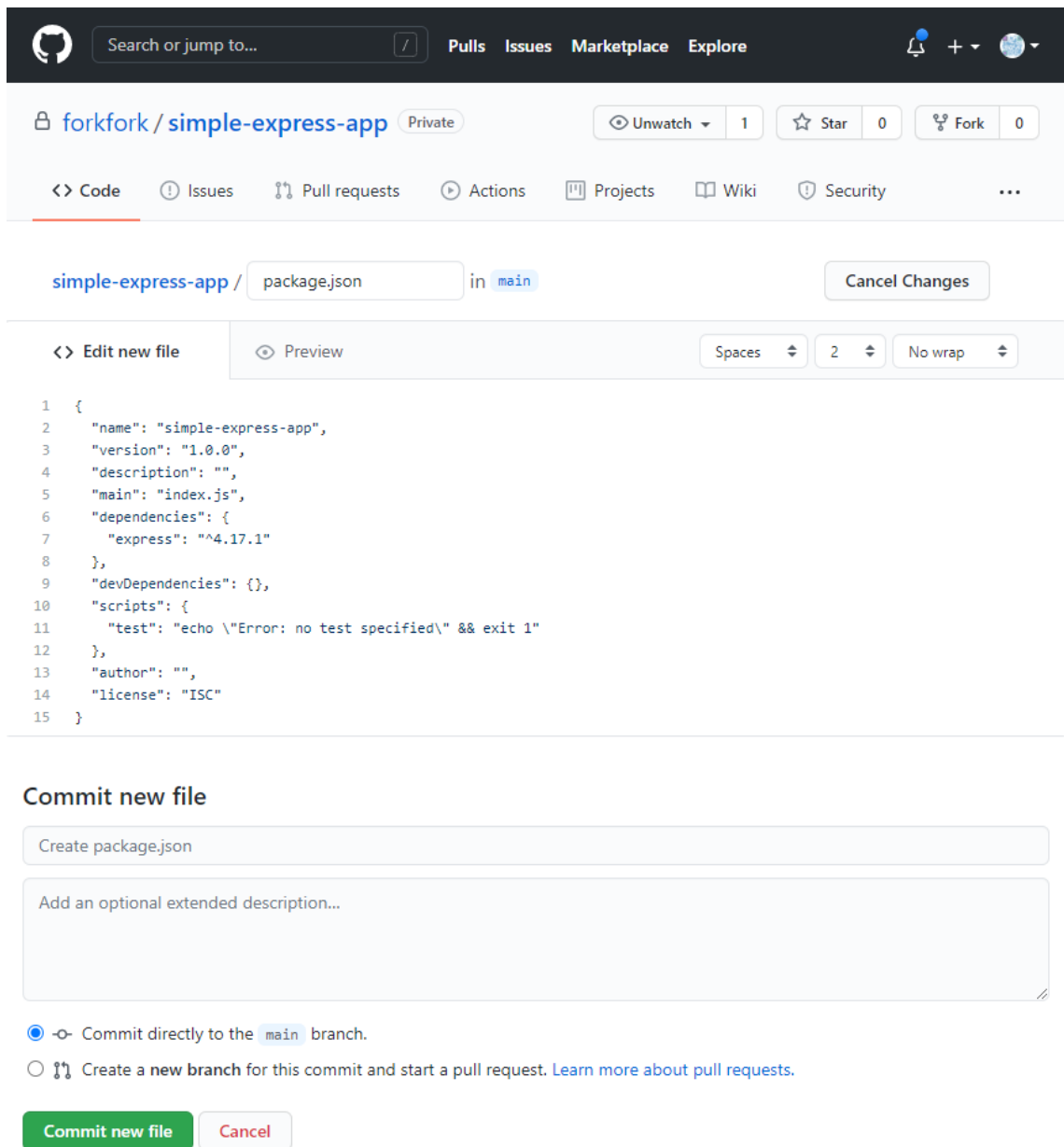
```
1 const express = require('express');
2 const app = express();
3 const port = 3000;
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!');
7 });
8
9 app.listen(port, () => {
10   console.log(`Example app listening at http://localhost:${port}`);
11 });
12
```

7. [Commit new file]をおします

8. 次に、[Add file]から[Create new file]を選びます。



9. [package.json]と名前を付けて、コマンド 2 番をコピーします。



GitHub interface showing the repository `forkfork / simple-express-app` (Private). The file `package.json` is selected in the `main` branch. The code editor displays the following content:

```
1 {
2   "name": "simple-express-app",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "dependencies": {
7     "express": "^4.17.1"
8   },
9   "devDependencies": {},
10  "scripts": {
11    "test": "echo \"Error: no test specified\" && exit 1"
12  },
13  "author": "",
14  "license": "ISC"
15 }
```

Below the editor, the **Commit new file** section is visible. It includes a text input for the commit message (containing "Create package.json") and an optional extended description field. The commit options are:

- ☒ Commit directly to the `main` branch.
- ☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

The **Commit new file** button is highlighted in green, and the **Cancel** button is in red.

10. [Commit new file]をおします
11. AWS マネージメントコンソールの App Runner にアクセスし、[サービスの作成]をおします
12. [ソースコードレポジトリ]を選びます



ソース

リポジトリタイプ

- ☐ コンテナレジストリ
コンテナレジストリに保存されているコンテナイメージからサービスをデプロイします。
- ☒ ソースコードレポジトリ
ソースコードレポジトリにホストされているコードからサービスをデプロイします。

13. (すでに過去 AWS と git の連携をしている方は画面遷移が異なります) [GitHub に接続]から[新規追加]をおします
14. [別のアプリケーションをインストールする]ボタンをおします
15. [Save]ボタンをおします

Repository access

☒ **All repositories**
This applies to all current *and* future repositories.

☐ **Only select repositories**

SaveCancel

16. GitHub アプリケーションにご自身の Git ユーザー名が表示されていることを確認したら、適当な名前を接続名につけて[次へ]をおします

GitHub connection

接続により、App Runner は GitHub と AWS をリンクさせることでソースコードからのデプロイが可能になります。接続は、アカウントに GitHub アプリケーションをインストールすることによって確立されます。この接続は、将来の App Runner サービスに使用できます。

接続名
接続名には、再度使用するときのために、わかりやすい名前を付けます。

この名前は一意である必要があり、使用できるのは文字、数字、ハイフンです。名前を後で編集することはできません。

GitHub アプリケーション
既に GitHub アプリケーションがインストールされている場所から選択するか、アカウント内の別の場所にアプリケーションをインストールします。

▼ or

[キャンセル](#) [次へ](#)

17. App Runner の画面に戻ると、先程作成した Git への接続が選択できるようになっているので選択します

ソースおよびデプロイ 情報

App Runner サービスのソースとそのデプロイ方法を選択します。

ソース

リポジトリタイプ

☐ コンテナレジストリ
コンテナレジストリに保存されているコンテナイメージからサービスをデプロイします。

☒ ソースコードリポジトリ
ソースコードリポジトリにホストされているコードからサービスをデプロイします。

GitHub に接続 情報

App Runner は、アカウントに「AWS Connector for GitHub」というアプリをインストールすることで、ソースコードをデプロイします。このアプリは、メインの GitHub アカウントまたは GitHub 組織にインストールできます。

apprunner0630 ▼

新規追加

18. レポジトリから[simple-express-app]を選びます。(ブランチはそのままで大丈夫です)

リポジトリ

simple-express-app ▼

🔄

ブランチ

main ▼

🔄

19. [次へ]をおします
20. ランタイムで[Nodejs12]を選びます

構築を設定 情報

構築設定

設定ファイル

☒ ここですべての設定を構成する
App Runner コンソールで、サービスのすべての設定を指定します。

☐ 設定ファイルを使用
App Runner がソースリポジトリの apprunner.yaml ファイルから設定を読み取るようにします。

ランタイム

サービスの App Runner ランタイムを選択します。

Nodejs 12 ▼

21. 構築コマンドに[npm install]、開始コマンドに[node index.js]、ポートに[3000]と入力して[次へ]をおします

構築コマンド

このコマンドは、新しいコードバージョンがデプロイされると、リポジトリのルートディレクトリで実行されます。これは、依存関係のインストールやコードのコンパイルに使用します。

`npm install`

開始コマンド

このコマンドは、サービスのルートディレクトリで実行され、サービスプロセスを開始します。これを使用して、サービスのウェブサーバーを起動します。このコマンドは、App Runner および定義した環境変数にアクセスできます。

`node index.js`

ポート

サービスではこの IP ポートが使用されます。

`3000`

キャンセル

戻る

次へ

- サービス名に適切なものを入力し[次へ]をおします
- 最後に[作成とデプロイ]をおします
- サービスが起動中です。Docker イメージの作成を含めて行われています

App Runner > サービス > simpleservice0630

simpleservice0630

情報

アクション

デプロイ

サービスの概要

ステータス

Operation in progress

デフォルトドメイン

<https://xpmv8m4iuc.ap-northeast-1.amazonaws.com>

サービス ARN

arn:aws:apprunner:ap-northeast-1:294963776963:service/simpleservice0630/1294737bbcd44119b34868d46831f982

ソース

<https://github.com/harunobukameda/simple-express-app/main>

- ログのタブをクリックすると作業ログが確認できます

イベントログ

ダウンロード

CloudWatchで表示

Search

```
1 06-30-2021 04:12 PM [AppRunner] Health check is successful. Routing traffic to application.
2 06-30-2021 04:10 PM [AppRunner] Performing health check on path '/' and port '3000'.
3 06-30-2021 04:10 PM [AppRunner] Provisioning instance and deploying image.
4 06-30-2021 04:10 PM [AppRunner] Successfully built source code.
5 06-30-2021 04:08 PM [AppRunner] Starting source code build.
6 06-30-2021 04:08 PM [AppRunner] Successfully pulled source code.
7 06-30-2021 04:07 PM [AppRunner] Service status is set to OPERATION_IN_PROGRESS.
8 06-30-2021 04:07 PM [AppRunner] Service creation started.
```

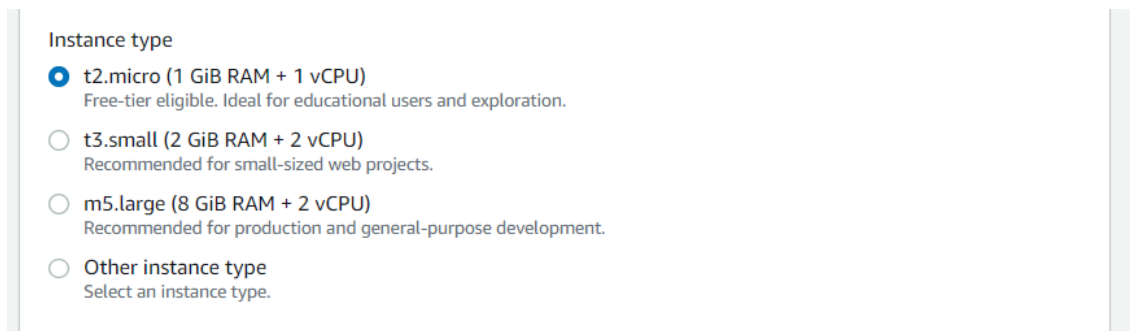
- デフォルトドメインをクリックするとアプリにアクセスができます
Hello World!
と表示されたら成功です。

[ECR との連携]

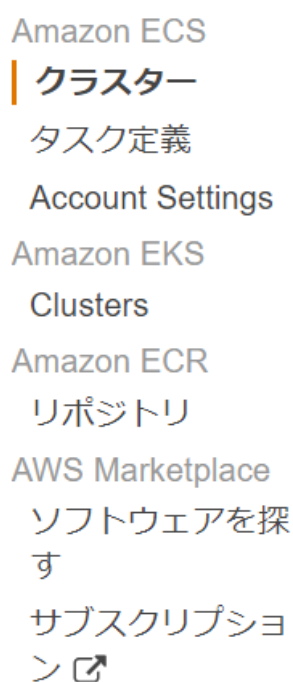
ここから、ECR に先行して Docker イメージを作成したのち Push しておき、App Runner でサービスを起動します。

- Cloud9 のマネージメントコンソールにいき、[Create environment]をおします
- 適当な名前をつけ、[Next Step]をおします

29. [Instance Type]で[t3.small]を選択し、あとはデフォルトのまま[Next Step]をおし、次の画面で[Create environment]をおします。Cloud9 が起動中の画面に遷移します。（作業しているリージョンにデフォルト VPC が無い場合、適当な VPC と Public Subnet を Advanced Network setting から指定してください）



30. 起動の間にブラウザの別のタブで ECR のマネージメントコンソールを開きます。ECS の以下の画面から[レポジトリ]をクリックすることで遷移できます



31. [レポジトリを作成]をおします



32. レポジトリに適切な名前をつけ、[レポジトリを作成]をおします
33. 作成されたレポジトリを選択し、[プッシュコマンドの表示]をおし、出力されるコマン

ドを 4 つともコピーしておきます

apprunner のプッシュコマンド

macOS / LinuxWindows

AWS CLI および Docker の最新バージョンがインストールされていることを確認します。詳細については、[Amazon ECR の開始方法](#) を参照してください。

次の手順を使用して、リポジトリに対してイメージを認証し、プッシュします。Amazon ECR 認証情報ヘルパーなどの追加のレジストリ認証方法については、[レジストリの認証](#) を参照してください。

1. 認証トークンを取得し、レジストリに対して Docker クライアントを認証します。

AWS CLI を使用します。

```
aws ecr get-login-password --region ap-northeast-1 | docker login --username AWS --password-
```

注意: AWS CLI の使用中にエラーが発生した場合は、最新バージョンの AWS CLI と Docker がインストールされていることを確認してください。

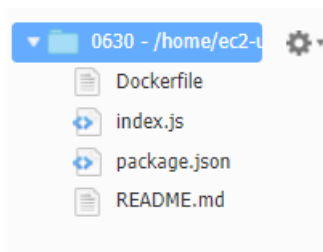
2. 以下のコマンドを使用して、Docker イメージを構築します。一から Docker ファイルを構築する方法については、「[こちらをクリック](#)」の手順を参照してください。既にイメージが構築されている場合は、このステップをスキップします。

```
docker build -t apprunner .
```

3. 構築が完了したら、このリポジトリにイメージをプッシュできるように、イメージにタグを付けます。

閉じる

34. Cloud9 のターミナルが利用できるようなっていますので、[File]→[New File]と選び、Commands.txt の 3 番をペーストして[index.js]で保存します
35. 同じく新しいファイルを作成し、Commands.txt の 4 番をペーストして[package.json]で保存します
36. 同じく新しいファイルを作成し、Commands.txt の 5 番をペーストして[Dockerfile]で保存します
37. 以下のようになれば OK です



38. 先程 ECR の画面からコピペした 4 つのコマンドを 1 個ずつ順番に実行していきます。特に 2 個目のコマンドは最後が、スペースとドットで終わっておりミスが発生しやすいので注意してください。

39. これで、Docker イメージの作成と ECR レポジトリへの Push が完了しましたので ECR の画面で確認をします。以下のように latest とタグが付いたイメージができていれば OK です。



40. [latest]をクリックし次の画面で[イメージの URI]をコピーしておいてください。この際、必ず以下のコピーボタンを使ってください。単純に OS の機能でのコピーだとこの後の手順に必要な情報(:latest)が含まれずコピーされます



41. App Runner の画面に戻り、[サービスの作成]をおします
42. 先程と異なり今度は[コンテナレジストリ]を選択し、プロバイダーで Amazon ECR を選び、上記でコピーした URI をペーストします。この際必ず:latest が URI の最後に付与されていることを確認してください

ソースおよびデプロイ 情報

App Runner サービスのソースとそのデプロイ方法を選択します。

ソース

リポジトリタイプ

☒ **コンテナレジストリ**
コンテナレジストリに保存されているコンテナイメージからサービスをデプロイします。

☐ ソースコードリポジトリ
ソースコードリポジトリにホストされているコードからサービスをデプロイします。

プロバイダー

☒ **Amazon ECR**

☐ Amazon ECR パブリック

コンテナイメージの URI
アクセスできるイメージの URI を入力するか、Amazon ECR アカウントでイメージを参照します。

43. App Runner から ECR へアクセスするための新しい IAM ロールを[新しいサービスロールの作成]を選択して作成します。適当な名前を付けたら[次へ]をおします。

44. サービス名に適当な名前をつけ、ポートは 3000 番を指定し、[次へ]をおします

サービスを設定 情報

サービス設定

サービス名

ECRprod

一意の名前を入力します。文字、数字、ダッシュを使用します。サービスの作成後に変更することはできません。

仮想 CPU とメモリ

1 vCPU ▼

2 GB ▼

環境変数 - オプション

カスタム設定値を保存するために使用できるキーと値のペア。
環境変数が定義されていません。

環境変数を追加

ポート

サービスではこの IP ポートが使用されます。

3000

▶ Additional configuration

45. 確認画面で[作成とデプロイ]をおしたらサービスが起動されますので、数分間待ちます

Create service が進行中です。

App Runner > サービス > ECRprod

ECRprod 情報

アクション ▼  デプロイ

サービスの概要

ステータス

⊖ Operation in progress

デフォルトドメイン

<https://frev3r3k3u.ap-northeast-1.amazonaws.com>

サービス ARN

 am:aws:apprunner:ap-northeast-1:294963776963:service/ECRprod/5b63203928c0498aafb2493020da2117

ソース

 294963776963.dkr.ecr.ap-northeast-1.amazonaws.com/apprunner:latest

46. [Create service が成功しました。]と表示されたら、デフォルトドメインにアクセスしてください。[Hello World!]と表示されたら成功です。

オリジナルシナリオには英語ですがさらに追加のシナリオがあるので余裕がある方はチャレンジしてみてください。

おつかれさまでした！

削除は以下を行ってください。

- App Runner のサービス
- ECR レジストリ

- Cloud9
- GitHub 接続（CodeDeploy の画面ではなく、App Runner の画面です）
- GitHub レポジトリ