
目次

はじめに	1.1
セクション1	1.2

面倒臭いことはもうしない！ AWS App RunnerでWebアプリを爆速でデプロイ！

1. 事前準備

- Dockerのインストール
- AWSアカウント

2. 今回の流れ

App Runnerを活用して、アプリケーションを実際にデプロイしてみます。

2.1. 技術要素

- [AWS AppRunner](#)
- [Amazon Elastic Container Registry: ECR](#)
- [Docker](#)
- [Node.js](#)
- [p5.js](#)
- [jest](#)

3. 開発手順

- p5.jsのアプリを「git fork」し、ローカルにアプリを配置する
- ローカル環境でP5.jsのアプリを動かしてみる
- AWSにログインしECRのリポジトリを作成する
- Dockerイメージの作成
- ECRにDockerイメージをpushする
- AWSのApp Runnerの画面に移動し、App Runnerの設定、デプロイを行う
- デプロイが完了後、App Runnerの画面にURLがあるのでアクセスし、アプリが動いているかを確認する

[面倒臭いことはもうしない！AWS App RunnerでWebアプリを爆速でデプロイ！](#)

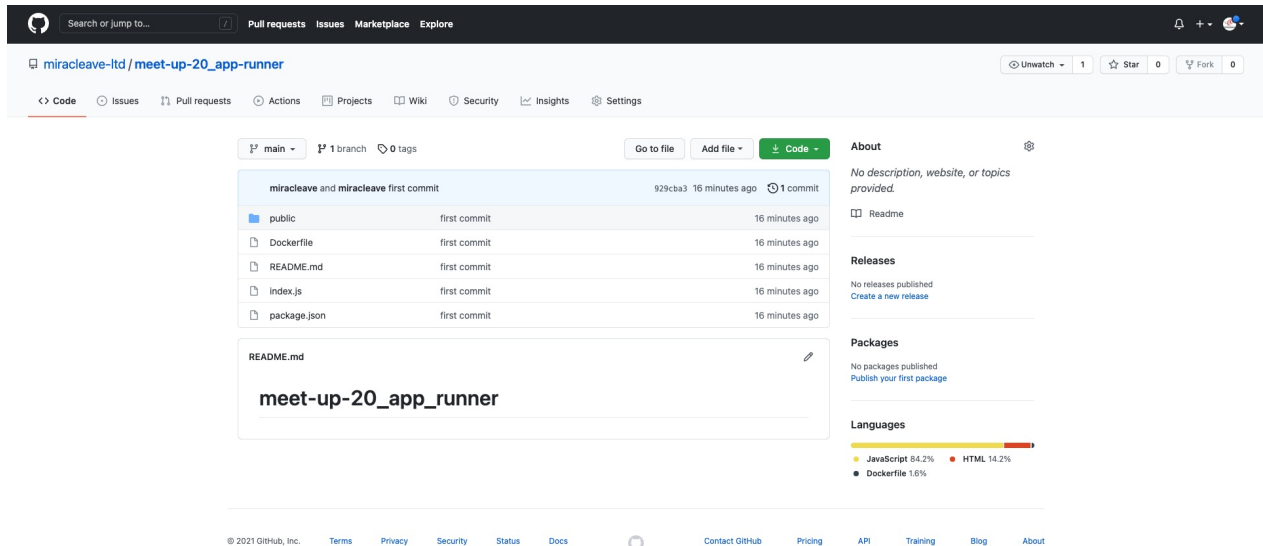
- [1. 事前準備](#)
- [2. 今回の流れ](#)
 - [2.1. 技術要素](#)
- [3. 開発手順](#)

Section1

1. やってみよう！

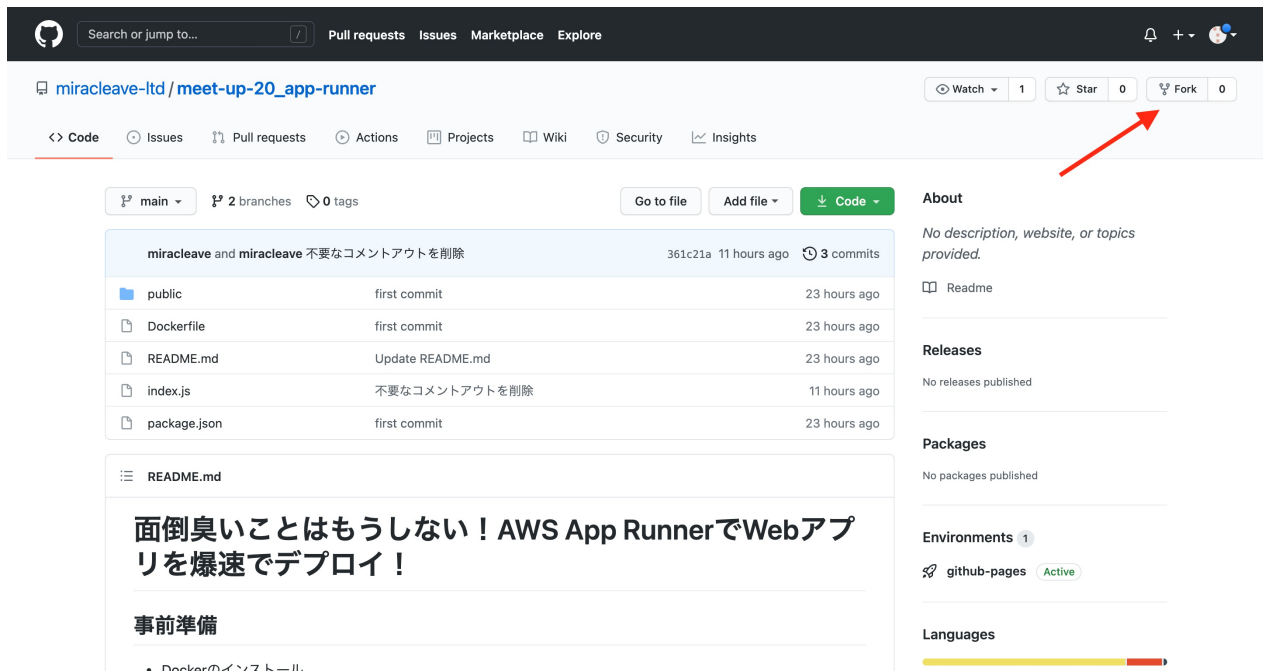
1.1. . P5.jsのアプリを「Git clone」し、ローカルにアプリを配置する

1.1.1. -1 Githubのリポジトリにアクセスする



URL:https://github.com/miracleleave-ltd/meet-up-20_app-runner

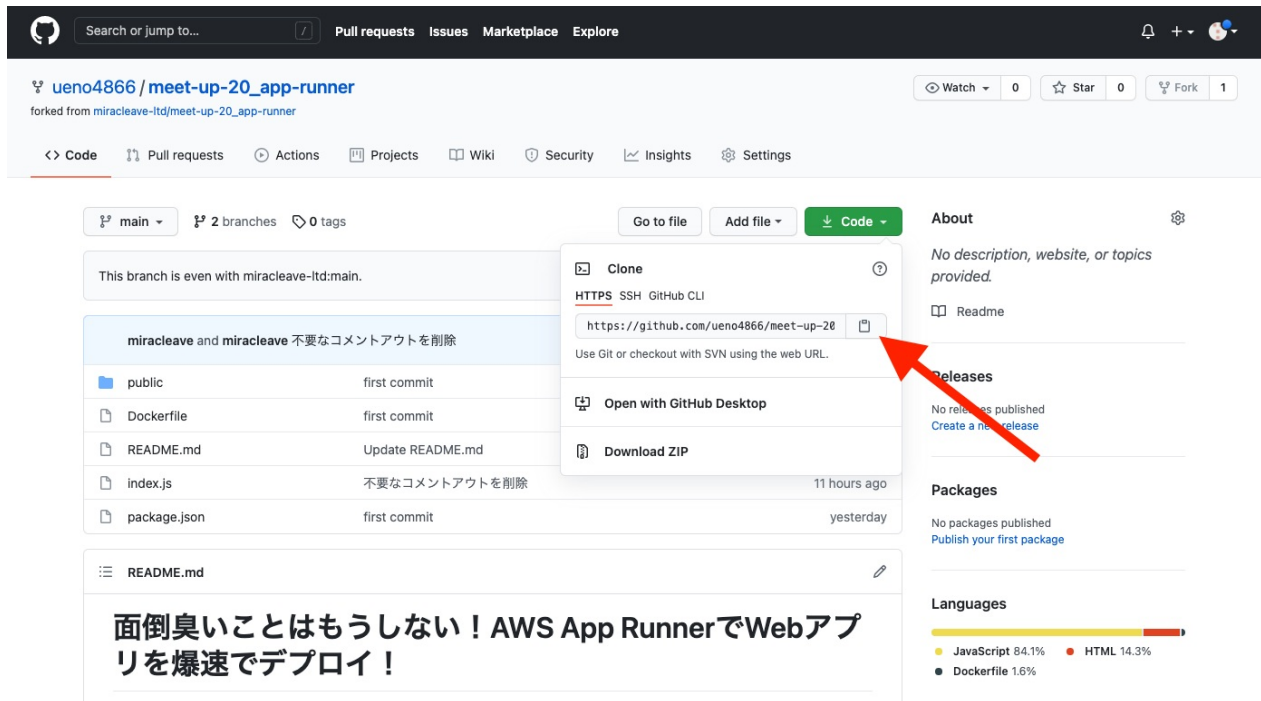
1.1.2. -2 リポジトリをコピーする（Forkボタンを押下する）



※Forkボタンをクリック後、Githubにログインしている方は自動で自分のリポジトリに遷移します。

1.1.3. -3 リポジトリをローカルに配置する

自分のリポジトリにコピーされたアプリのURLをコピーする



ターミナルに移動し、git cloneを行う

```
例：Desktopにクローンする場合
~ $ cd ~/Desktop
~/Desktop $ git clone [コピーしたURL]
```

1.2. . ローカル環境でP5.jsのアプリを動かしてみる

1.2.1. -1 イメージの作成

```
# meet-up-20_app-runnerフォルダに移動
~/Desktop $ cd meet-up-20_app-runner
# Dockerイメージの作成
~/Desktop/meet-up-20_app-runner $ docker build . -t app-runner-example
```

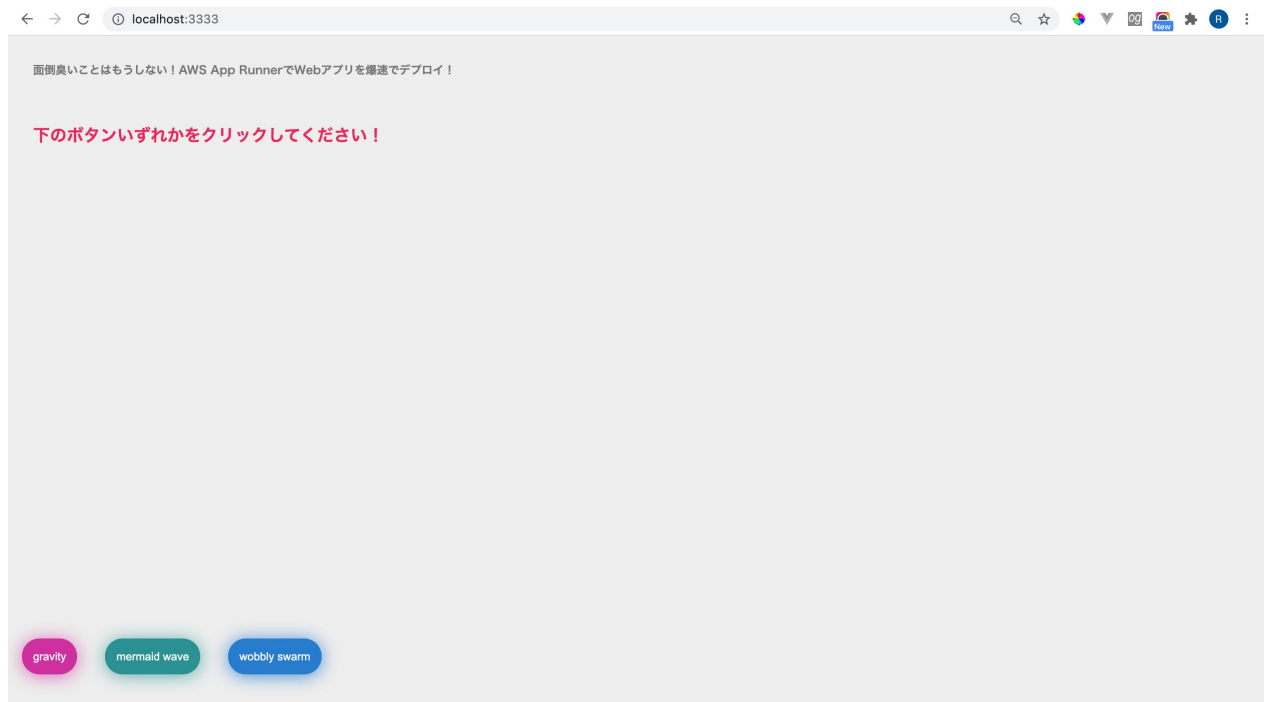
1.2.2. -2 コンテナの作成、起動

```
~/Desktop/meet-up-20_app-runner $ docker run -p 3333:3333 -d app-runner-example
```

1.2.3. -3 アプリが動いているかを確認

<http://localhost:3333>

※実際に動いていれば下記の画面が表示されます。



1.3. . AWSにログインしECRのリポジトリを作成する

1.3.1. -1 ECRの画面に移動し、リポジトリを作成ボタンをクリックする

検索欄から「ECR」と検索

The screenshot displays the AWS Management Console interface for Amazon ECR. The top panel shows the 'Services' section with 'Elastic Container Registry' highlighted. The bottom panel shows the 'Repositories' section with a red arrow pointing to the 'Create Repository' button.

Top Panel: Services

- サービス (2)
 - ドキュメンテーション (\$1,897)
 - ナレッジ記事 (\$0)
 - Marketplace (13)
- 機能 (3)
 - Secrets Manager
 - ライフサイクルを通じてシークレットを簡単に更新、管理、取得する
 - Elastic Container Service
 - 安全性と信頼性に優れ、スケーラブルなコンテナ実行方法
 - レジストリ
 - Elastic Container Registry の機能
 - リポジトリ
 - Elastic Container Registry の機能
 - シークレット
 - Secrets Manager の機能
- ドキュメンテーション
 - ドキュメンテーションの 31,897 件の結果をすべて表示
 - Pushing a Docker image - Amazon ECR
 - Amazon ECR private repositories - Amazon ECR
 - Pulling an image - Amazon ECR

Bottom Panel: Repositories

Private | Public

プライベートリポジトリ

リポジトリを作成

リポジトリ名	URI	作成時刻	タグのイミュータビリティ	プッシュ時にスキャン	暗号化タイプ
リポジトリが見つかりません リポジトリが見つかりませんでした					

1.3.2. -2 ECRの設定を行い、リポジトリを作成する

Amazon Container Services

Amazon ECS

Clusters

Task definitions

Amazon EKS

Clusters

Amazon ECR

Repositories

Registries

Public gallery

ECR > リポジトリ > リポジトリを作成

リポジトリを作成

一般設定

可視性設定 [Info](#)
リポジトリの可視性設定を選択します。
☒ プライベート
アクセスは IAM およびリポジトリポリシーのアクセス許可によって管理されます。
☐ パブリック
イメージについて、パブリックに表示およびアクセス可能。
リポジトリ名
簡潔な名前を指定します。デベロッパーが名前でもリポジトリの内容を識別できる必要があります。

dkr.ecr.ap-northeast-1.amazonaws.com/

app-runner-example

最大文字数 256 のうち 18 (最小文字数 2)。The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.
タグのイミュータビリティ [Info](#)
タグのイミュータビリティを有効にすると、同じタグを使用した後続イメージのプッシュによるイメージタグの上書きを防ぎます。タグのイミュータビリティを無効にするとイメージタグを上書きできるようになります。
☐ 無効

① リポジトリが作成されると、リポジトリの可視性設定を変更することはできません。

イメージスキャンの設定

プッシュ時にスキャン
プッシュ時にスキャンを有効にすると、各イメージがリポジトリにプッシュされた後に自動的にスキャンされます。無効にした場合、スキャン結果を取得するには、各イメージのスキャンを手動で開始する必要があります。
☐ 無効

暗号化設定

KMS 暗号化
デフォルトの暗号化設定を使用する代わりに、AWS Key Management Service (KMS) を使用して、このリポジトリに保存されているイメージを暗号化できます。
☐ 無効

① KMS 暗号化設定は、リポジトリの作成後に変更または無効にすることはできません。

キャンセル

リポジトリを作成

1.4. . ローカルから ECR に push するための IAM User を作成

注意：すでに「AdministratorAccess」権限を持ち、プログラムのアクセスの権限のあるユーザーを作成されている場合は、このステップを飛ばしてください。

1.4.1. -1 IAM の画面に移動し、左メニューからポリシーを選択し、ポリシーを作成ボタンをクリックする

7

Identity and Access Management (IAM)

新しいポリシーリスト エクスペリエンスの紹介
を簡単に使用できるように ポリシーリスト のエクスペリエンスを再設計しました。 ご意見をお聞かせください。

ダッシュボード

▼ アクセス管理
値のユーザーグループ
ユーザー
ロール
ポリシー
ID プロバイダ
アカウント設定

▼ アクセスレポート
アクセスアナライザー
アーカイブルール
アナライザー
設定
認証情報レポート
組織アクティビティ
サービスコントロールポリシー (SCP)

IAM > ポリシー

ポリシー (840) [情報](#)
ポリシーはアクセス許可を定義する AWS のオブジェクトです。

Q プロパティまたは値でポリシーをフィルタリングし、Enter キーを押します

アクション ▼ [ポリシーを作成](#)

ポリシー名	タイプ	として使用	説明
AccessEcrForAppRunner	カスタマー管理	なし	AccessEcrForAppRu
Application	カスタマー管理	アクセス許可ポリシー (I)	Application
AwsAndinfraDeveloperPolicy	カスタマー管理	なし	AwsAndinfraDevelop
AwsAndinfraDirectorPolicy	カスタマー管理	なし	AwsAndinfraDirector
AWSLambdaBasicExecutionRole-72a993fb-35e0-4e67-a7ff-db220b0d8713	カスタマー管理	アクセス許可ポリシー (I)	
Batch-S3	カスタマー管理	アクセス許可ポリシー (I)	Batch-S3
Operation	カスタマー管理	アクセス許可ポリシー (I)	Operation
s3corr_for_udemy-s3-test-20200802_to_udemy-aws-replication-20200803	カスタマー管理	アクセス許可ポリシー (I)	
AWSDirectConnectReadOnlyAccess	AWS 管理	なし	Provides read only a
AmazonGlacierReadOnlyAccess	AWS 管理	なし	Provides read only a
AWSMarketplaceFullAccess	AWS 管理	なし	Provides the ability t
ClientVPNServiceRolePolicy	AWS 管理	なし	Policy to enable AWI
AWSSSODirectoryAdministrator	AWS 管理	なし	Administrator access
AWSIoTClickReadOnlyAccess	AWS 管理	なし	Provides read only a
AutoScalingConsoleReadOnlyAccess	AWS 管理	なし	Provides read-only a

1.4.2. -2 JSONを選択、下記をコピーし貼り付ける

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:*"
      ],
      "Resource": "*"
    }
  ]
}
```

ポリシーの作成

1 2 3

ポリシーにより、ユーザー、グループ、またはロールに割り当てることができる AWS アクセス権限が定義されます。ビジュアルエディタで JSON を使用してポリシーを作成または編集できます。 [詳細はこちら](#)

ビジュアルエディタ **JSON** [管理ポリシーのインポート](#)

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "ecr:GetDownloadUrlForLayer",
8         "ecr:BatchGetImage",
9         "ecr:DescribeImages",
10        "ecr:GetAuthorizationToken",
11        "ecr:BatchCheckLayerAvailability"
12      ],
13       "Resource": "*"
14     }
15   ]
16 }
```

① セキュリティ: 0 ② エラー: 0 ③ 警告: 0 ④ 検索: 0

文字数 215 / 6,144

[キャンセル](#) [次のステップ: タグ](#)

1.4.3. -3 タグは設定せず、次のステップボタンをクリックし、ポリシー名を設定して作成する

名前: AccessEcrForAppRunner

ポリシーの作成

1 2 3

ポリシーの確認

名前* AccessEcrForAppRunner
英数字と「+','=','_」を使用します。最大 128 文字。

説明 AccessEcrForAppRunner
最大 1000 文字。英数字と「+','=','_」を使用します。

概要

Q フィルター

サービス ▼	アクセスレベル	リソース	リクエスト条件
許可 (285 サービス中 1) 残りの 284 を表示			
Elastic Container Registry	制限: 読み込み	すべてのリソース	なし

タグ

キー ▲ 値 ▼

リソースに関連付けられたタグはありません。

* 必須

キャンセル

戻る

ポリシーの作成

1.4.4. -4 左メニューからユーザーを選択し、ユーザーを追加ボタンをクリックする

Identity and Access Management (IAM)

新しいユーザーリスト エクスペリエンスの紹介
IAM ユーザーは、アカウントで AWS を操作するために長期的な認証情報を持つアイデンティティです。この機能を簡単に使用できるようにユーザーリストのエクスペリエンスを再設計しました。ご意見をお聞かせください。

IAM > ユーザー

ユーザー (4) 情報

IAM ユーザーは、アカウントで AWS を操作するために長期的な認証情報を持つアイデンティティです。

Q ユーザー名またはアクセスキーでユーザーを検索

	ユーザー名 ▼	グループ ▼	最後のアクティビティ ▼	MFA ▼	パスワードが作成されてから経過した期間 ▼	アクティブなキーが作成されてから経過した期間 ▼
<input type="checkbox"/>	[REDACTED]	なし	6 日前	なし	なし	21 日前
<input type="checkbox"/>	[REDACTED]	なし	8 日前	なし	8 日前	8 日前
<input type="checkbox"/>	[REDACTED]	administrator	276 日前	なし	368 日前	368 日前
<input type="checkbox"/>	[REDACTED]	なし	21 分前	なし	588 日前	-

ユーザーを追加

1.4.5. -5 IAMユーザーを作成する

ユーザー名: meet-up-app-runner-user プログラムによるアクセスにチェック

ユーザーを追加

1 2 3 4 5

ユーザー詳細の設定

同じアクセスの種類とアクセス権限を使用して複数のユーザーを一度に追加できます。 [詳細はこちら](#)

ユーザー名*

[+ 別のユーザーの追加](#)

AWS アクセスの種類を選択

これらのユーザーから AWS にアクセスする方法を選択します。アクセスキーと自動生成パスワードは前のステップで提供されています。 [詳細はこちら](#)

- アクセスの種類* ☒ **プログラムによるアクセス**
AWS API、CLI、SDK などの開発ツールの **アクセスキー ID** と **シークレットアクセスキー** を有効にします。
- ☐ **AWS マネジメントコンソールへのアクセス**
ユーザーに AWS マネジメントコンソールへのサインインを許可するための **パスワード** を有効にします。

* 必須

[キャンセル](#)

[次のステップ: アクセス権限](#)

既存のポリシーを直接アタッチを選択し、「AccessEcrForAppRunner」にチェックを入れ、確認画面までスキップする

ユーザーを追加

1 2 3 4 5

▼ アクセス許可の設定

 ユーザーをグループに追加

 アクセス権限を既存のユーザーからコピー

 既存のポリシーを直接アタッチ

ポリシーの作成 ポリシーのフィルタ  4 件の結果を表示中

▶ アクセス権限の境界の設定

キャンセル

戻る

次のステップ: タグ


確認画面でCSVをダウンロードする

ユーザーを追加

1 2 3 4 5

✔ 成功

以下に示すユーザーを正常に作成しました。ユーザーのセキュリティ認証情報を確認してダウンロードできます。AWS マネジメントコンソールへのサインイン手順を E メールでユーザーに送信することもできます。今回が、これらの認証情報をダウンロードできる最後の機会です。ただし、新しい認証情報はいつでも作成できます。

AWS マネジメントコンソールへのアクセス権を持つユーザーは「 amazon.com/console」でサインインできます

 .csv のダウンロード

	ユーザー	アクセスキー ID	シークレットアクセスキー
▶	✔ meet-up-app-runner-user	AKIA2QBOUKIYGAWSTXC 	***** 表示

1.5. . ECRにDockerイメージをpushする

1.5.1. -1 認証情報の設定

credentialsの設定

```
# .awsに移動
~/Desktop/meet-up-20_app-runner $ cd ~/.aws
# credentialsを作成、修正
.aws $ vi credentials
```

先ほど、CSVでダウンロードしたアクセスIDとアクセスキーを下記のイコールの後に値を設定する

```
[default]
aws_access_key_id =
aws_secret_access_key =
```

※設定後は esc -> :wp -> Enter の順番でキーボードを打ち、保存する

configの設定

```
.aws $ vi config
```

下記の形式で設定する

```
[default]
region = ap-northeast-1
output = json
```

※設定後は esc -> :wp -> Enter の順番でキーボードを打ち、保存する

1.5.2. -2 ECR画面に移動し、プッシュコマンドを確認し、ECRにpushする

今回はAWS CLIをローカルにダウンロードせず、Dockerを通してAWS CLIコマンドを実行します。 Dockerを通してAWS CLIコマンドを実行する際は下記コマンドをベースに実行します

```
docker run --rm -ti -v ~/.aws:/root/.aws -v $(pwd):/aws amazon/aws-cli [AWSコマンド]
```

プッシュコマンドを確認

app-runner-example のプッシュコマンド

macOS / Linux | Windows

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 app-runner-example .
```

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。

```
docker tag app-runner-example:latest [REDACTED].dkr.ecr.ap-northeast-1.amazonaws.com/[REDACTED]
```

4. 以下のコマンドを実行して、新しく作成した AWS リポジトリにこのイメージをプッシュします。

```
docker push [REDACTED].dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example:latest
```

閉じる

ECRにプッシュする

```
# クローンしてきたフォルダに移動
```

```
cd ~/Desktop/meet-up-20_app-runner
# ログインする (ログインが成功すると「Login Succeeded」が表示される)
docker run --rm -ti -v ~/.aws:/root/.aws -v $(pwd):/aws amazon/aws-cli [app-runner-exampleのプッシュコマンドの1をコピー (先頭のawsは省く)]
例: docker run --rm -ti -v ~/.aws:/root/.aws -v $(pwd):/aws amazon/aws-cli ecr get-login-password --region ap-northeast-1 | docker login --username AWS --password-stdin 0000000000.dkr.ecr.ap-northeast-1.amazonaws.com
# app-runner-exampleのプッシュコマンドの3を実行
例: docker tag app-runner-example:latest 0000000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example:latest
# app-runner-exampleのプッシュコマンドの4を実行
例: docker push 0000000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example:latest
```

1.6. . AWSのApp Runnerの画面に移動し、App Runnerの設定、デプロイを行う

1.6.1. -1 App Runnerの画面に移動し、App Runnerサービスを作成するボタンをクリック

The screenshot shows the AWS App Runner console. The top navigation bar includes the AWS logo, a search bar, and a 'App Runner' filter. The main content area is divided into two columns. The left column lists services, with 'AWS App Runner' highlighted. The right column shows the 'AWS App Runner' service details, including a description, pricing, and a 'Get started' button. A red arrow points to the 'AWS App Runner' service in the list.

サービス (21)

- ドキュメント (191,24)
- ナレッジ記事 (10)
- Marketplace (1,434)

サービス

- AWS App Runner**
Build and run production-ready web applications at scale.
- AWS App Mesh
マイクロサービスの簡単なモニタリングと管理。
- AWS AppConfig
AWS AppConfig では、実行時にアプリケーション設定を更新します。
- AWS AppSync
モバイルアプリおよびウェブアプリ用の GraphQL を使用したオンラインとオフラインの...

機能

- AppConfig
Systems Manager の機能
- Serverless Application Lens
AWS Well-Architected Tool の機能
- API の作成
AWS AppSync の機能
- アプリケーション
Elastic Beanstalk の機能

ドキュメント

- Release: AWS App Runner general availability on May 18, 2021 - AWS App Runner
- Release Notes
- About AWS App Runner release notes - AWS App Runner

App Runner の使用を開始

App Runner サービスを作成

コンピューティングコスト (米国)

リソース	コスト
vCPU	\$0.064 / 時間
GB	\$0.007 / 時間

オプションのアドオンコスト (米国)

オプション	コスト
自動デプロイ	\$1 / アプリケーション / 月
構築	\$0.005 / 分

開始方法

App Runner の開始方法

ドキュメント

FAQ
ドキュメント

AWS App Runner
本稼働ウェブアプリケーションを大規模に構築して実行する

AWS App Runner は、デベロッパーがソースコードやコンテナイメージからスケールで安全なウェブアプリケーションに直接簡単にデプロイできるようにする、フルマネージド型サービスです。

仕組み

メリット

- 使いやすい**
AWS を使用した経験がなくても、数回クリックするだけで、ウェブスケールのアプリケーションを構築して実行できます。サーバー設定、ネットワークワキング、スケーリング、負荷分散、アプロバライメントに関する知識は必要ありません。
- 時間を削減**
App Runner のリソースとインフラストラクチャを管理する必要は AWS によって完全に削減されます。
- トラフィックに応じてスケール**
App Runner は、トラフィックに応じてリソースをシームレスにスケールアップし、トラフィックがないときはプロビジョニングされたコンテナの数に合わせて自動的にスケールダウンするため、高い可用性でアプリケーションを実行できます。

1.6.2. -2 App Runnerサービスの設定、デプロイを行う

コンテナイメージのURIは、先ほど作成したECRリポジトリを選択する

App Runner > サービスの作成

ステップ 1
ソースおよびデプロイ

ステップ 2
サービスを設定

ステップ 3
確認および作成

ソースおよびデプロイ Info

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。

参照

デプロイ設定

デプロイトリガー

☐ 手動
App Runner コンソールまたは AWS CLI を使用して、各デプロイを自分で開始します。

☒ 自動
App Runner はレジストリを監視し、イメージプッシュごとにサービスの新しいバージョンをデプロイします。

ECR アクセスロール Info
このロールは、ECR にアクセスするための App Runner アクセス許可を付与します。カスタムロールを作成するには、[IAM コンソール](#) に移動します。

☒ 新しいサービスロールの作成

☐ 既存のサービスロールを使用

サービスロール名
App Runner が ECR アクセス用の管理ポリシーがアタッチされたアカウントで作成する IAM ロールの名前。

キャンセル

次へ

サービスを設定する

App Runner > サービスの作成

ステップ 1
ソースおよびデプロイ

ステップ 2
サービスを設定

ステップ 3
確認および作成

サービスを設定 Info

サービス設定

サービス名

app-runner-example

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 TCP ポートが使用されます。

3333

▶ Additional configuration

▶ Auto Scaling Info

Auto Scaling の動作を設定します。

▶ ヘルスチェック Info

TCP ヘルスチェックを設定します。

▶ セキュリティ Info

インスタンスロールと AWS KMS 暗号化キーを指定

▶ タグ Info

タグを使用して、リソースの検索とフィルタリング、AWS コストの追跡、およびアクセス許可の管理を行います。

キャンセル 戻る 次へ

作成とデプロイボタンをクリック

キー	値
環境変数が定義されていません。	

▶ その他の設定

▼ Auto Scaling

同時実行	最大サイズ
100	25
最小サイズ	
1	

▼ ヘルスチェック

タイムアウト	非正常のしきい値
5 秒	5 リクエスト
間隔	正常性のしきい値
10 秒	1 リクエスト

▼ セキュリティ

インスタンスロール	AWS KMS 暗号化キー
—	AWS マネージド型

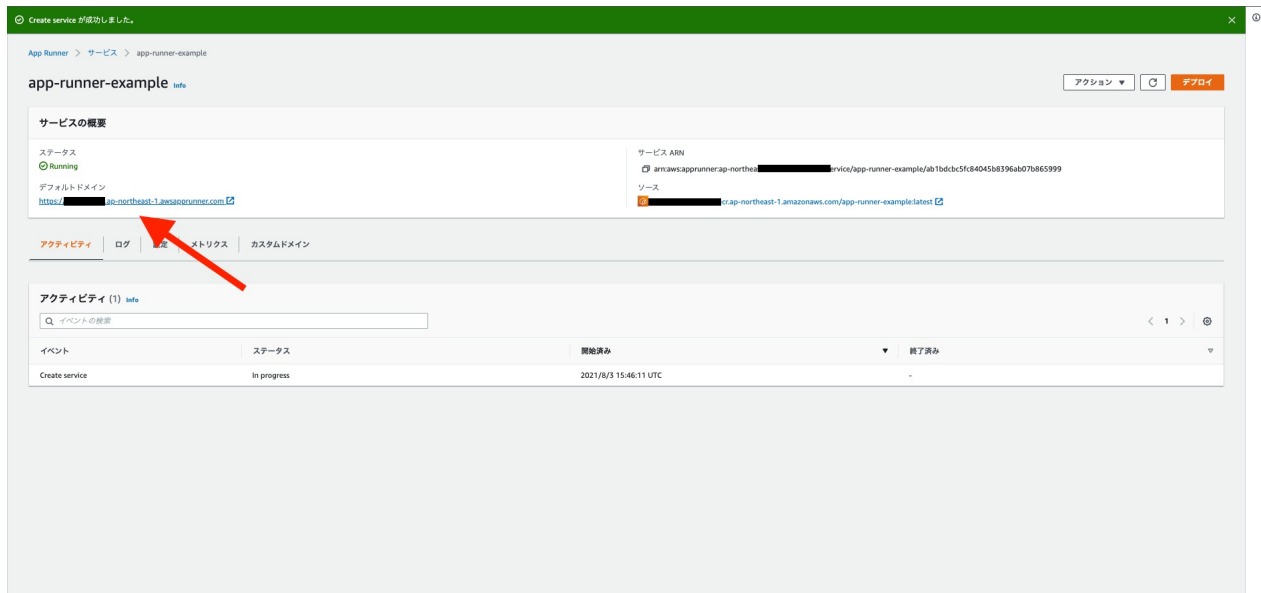
▼ タグ

キー	値
タグが設定されていません。	

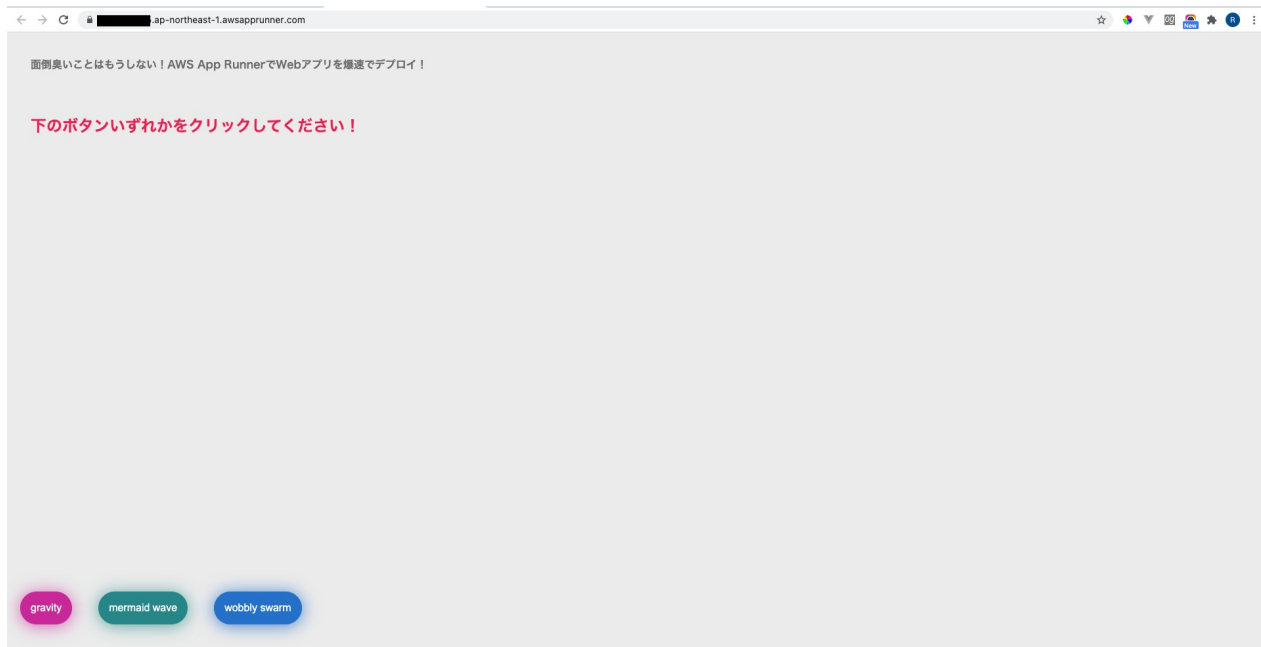
キャンセル 戻る 作成とデプロイ

1.7. . デプロイが完了後、App Runner画面のURLにアクセスし、アプリが動いているかを確認する

1.7.1. -1 公開されたURLにアクセスし、実際にアプリが動いているか確認する



1.7.2. -2 アプリが問題なく動作している場合、下記の画面が表示されます



Section1

- 1. やってみよう！
 - 1.1. . P5.jsのアプリを「Git clone」し、ローカルにアプリを配置する
 - 1.1.1. -1 Githubのリポジトリにアクセスする
 - 1.1.2. -2 リポジトリをコピーする（Forkボタンを押下する）
 - 1.1.3. -3 リポジトリをローカルに配置する
 - 1.2. . ローカル環境でP5.jsのアプリを動かしてみる
 - 1.2.1. -1 イメージの作成
 - 1.2.2. -2 コンテナの作成、起動
 - 1.2.3. -3 アプリが動いているかを確認
 - 1.3. . AWSにログインしECRのリポジトリを作成する
 - 1.3.1. -1 ECRの画面に移動し、リポジトリを作成ボタンをクリックする
 - 1.3.2. -2 ECRの設定を行い、リポジトリを作成する
 - 1.4. . ローカルからECRにpushするためのIAM Userを作成
 - 1.4.1. -1 IAMの画面に移動し、左メニューからポリシーを選択し、ポリシーを作成ボタンをクリックする

- 1.4.2. -2 JSONを選択、下記をコピーし貼り付ける
- 1.4.3. -3 タグは設定せず、次のステップボタンをクリックし、ポリシー名を設定して作成する
- 1.4.4. -4 左メニューからユーザーを選択し、ユーザーを追加ボタンをクリックする
- 1.4.5. -5 IAMユーザーを作成する
- 1.5. . ECRにDockerイメージをpushする
 - 1.5.1. -1 認証情報の設定
 - 1.5.2. -2 ECR画面に移動し、プッシュコマンドを確認し、ECRにpushする
- 1.6. . AWSのApp Runnerの画面に移動し、App Runnerの設定、デプロイを行う
 - 1.6.1. -1 App Runnerの画面に移動し、App Runnerサービスを作成するボタンをクリック
 - 1.6.2. -2 App Runnerサービスの設定、デプロイを行う
- 1.7. . デプロイが完了後、App Runner画面のURLにアクセスし、アプリが動いているかを確認する
 - 1.7.1. -1 公開されたURLにアクセスし、実際にアプリが動いているか確認する
 - 1.7.2. -2 アプリが問題なく動作している場合、下記の画面が表示されます