

目次

はじめに	1.1
1. デモアプリクローン	1.2
2. アプリ動作確認	1.3
3. コンテナレジストリパターン	1.4
4. ソースコードリポジトリパターン	1.5
5. ゴミ掃除	1.6

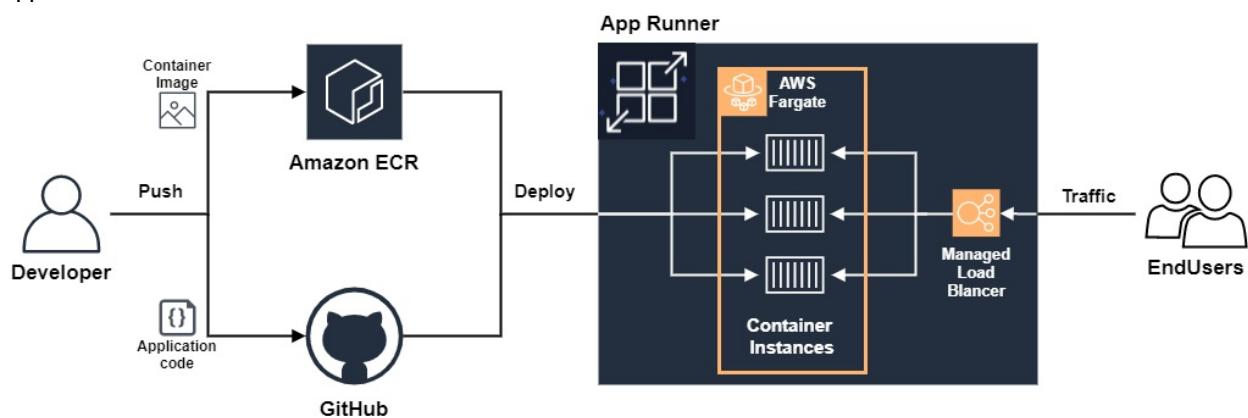
面倒臭いことはもうしない！ AWS App RunnerでWebアプリを爆速でデプロイ！

1. 事前準備

- Dockerインストール
- GitHubアカウント
- 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ます。



大きな流れとしては、次の2パターンを実施します。

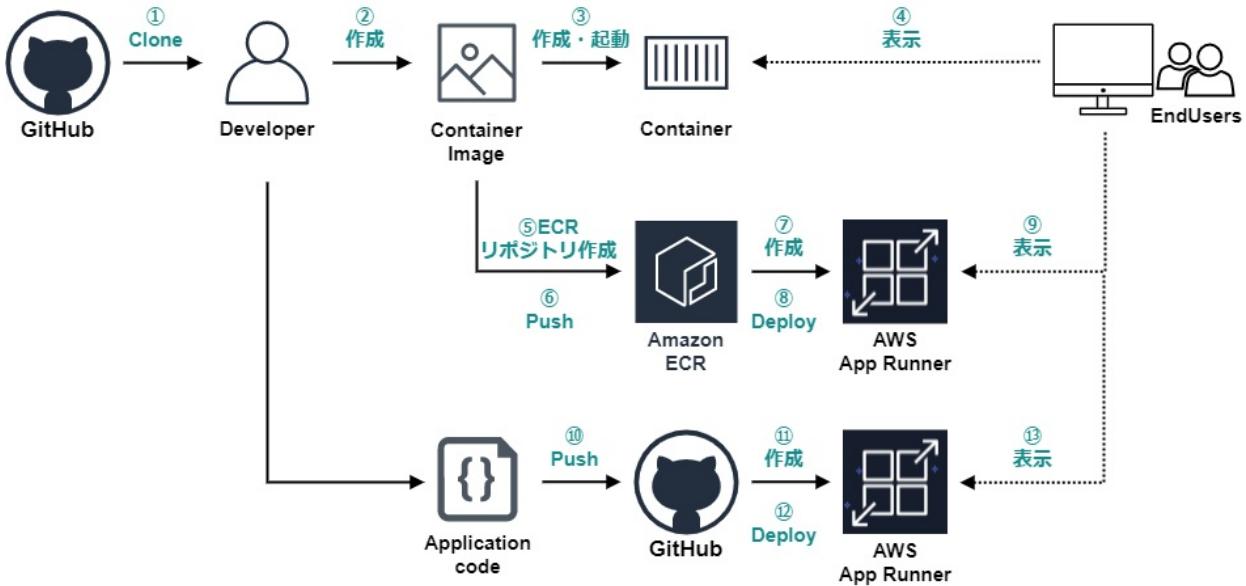
- コンテナレジストリパターン
- ソースコードリポジトリパターン

2.1. 技術要素

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

3. 手順

全体手順としては次の流れで進めます。



- ①デモアプリクローン
- ②③④デモアプリ動作確認
- ⑤⑥⑦⑧⑨コンテナレジストリパターン(Amazon ECRパターン)
- ⑩⑪⑫⑬ソースコードリポジトリパターン(GitHubパターン)
- ゴミ掃除

4. 注意事項

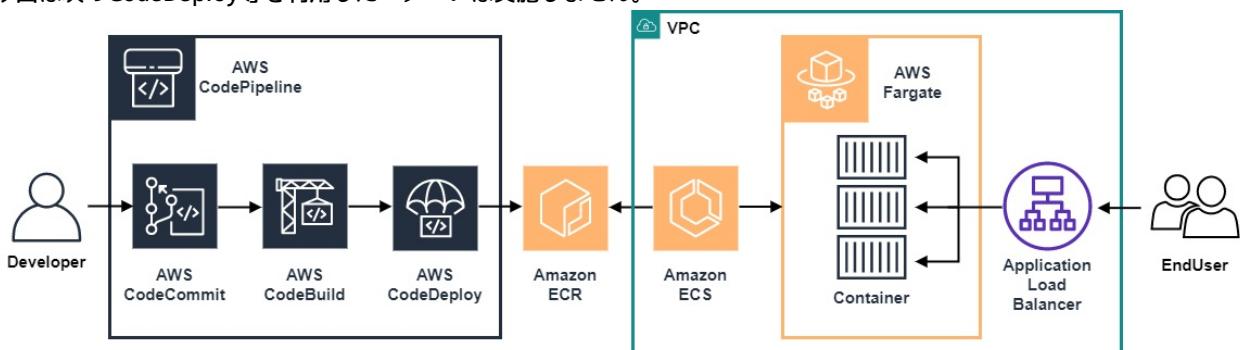
App Runnerでは、まだ制限が多くあります。

利用する際は、事前に制限内容を確認の上、検討をお願いします。

ご参考までにロードマップ等ご確認ください。

[App Runner - ロードマップ](#)

今回は次のCodeDeploy等を利用したパターンは実施しません。



Windows/Macの方向けに作成しております。

コマンドラインツールは、個々の利用しているもので良いのですが、今回の手順は次のものを利用します。

- Windows : コマンドプロンプト
- Mac : ターミナル

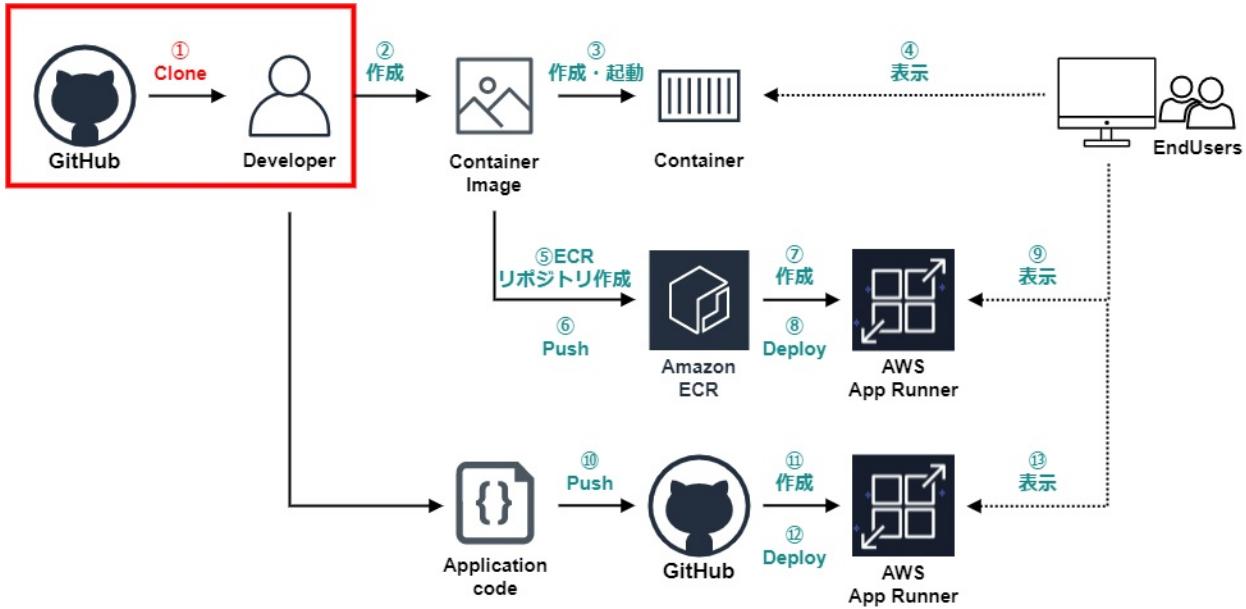
面倒臭いことはもうしない！AWS App RunnerでWebアプリを爆速でデプロイ！

- 1. 事前準備

- 2. 今回の流れ
 - 2.1. 技術要素
- 3. 手順
- 4. 注意事項

デモアプリクローン

この手順では次の手順を進めていきます。



1. GitHubリポジトリ表示

次のリンクよりデモアプリがあるGitHubリポジトリにアクセスします。

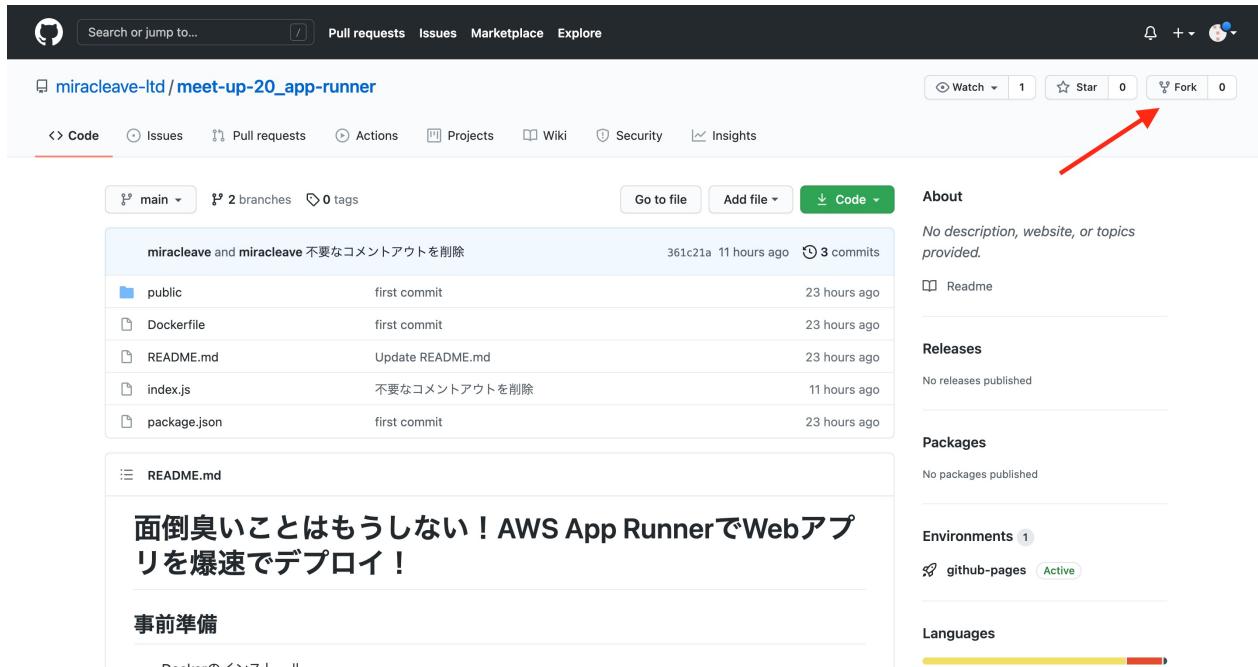
URL:https://github.com/miracleave-ltd/meet-up-20_app-runner

The screenshot shows the GitHub repository page for 'miracleave-ltd/meet-up-20_app-runner'. The repository has 1 branch and 0 tags. The main branch contains 1 commit by 'miracleave and miracleave' made 16 minutes ago. The commit includes files: public, Dockerfile, README.md, index.js, and package.json. The README.md file contains the text 'meet-up-20_app_runner'. The repository has 1 star, 0 forks, and 0 issues. The 'About' section notes 'No description, website, or topics provided.' and links to 'Readme'. The 'Releases' section says 'No releases published. Create a new release.' The 'Languages' section shows a chart with JavaScript at 84.2%, HTML at 14.2%, and Dockerfile at 1.6%.

2. リポジトリコピー

1. デモアプリクローン

Forkボタンをクリックします。

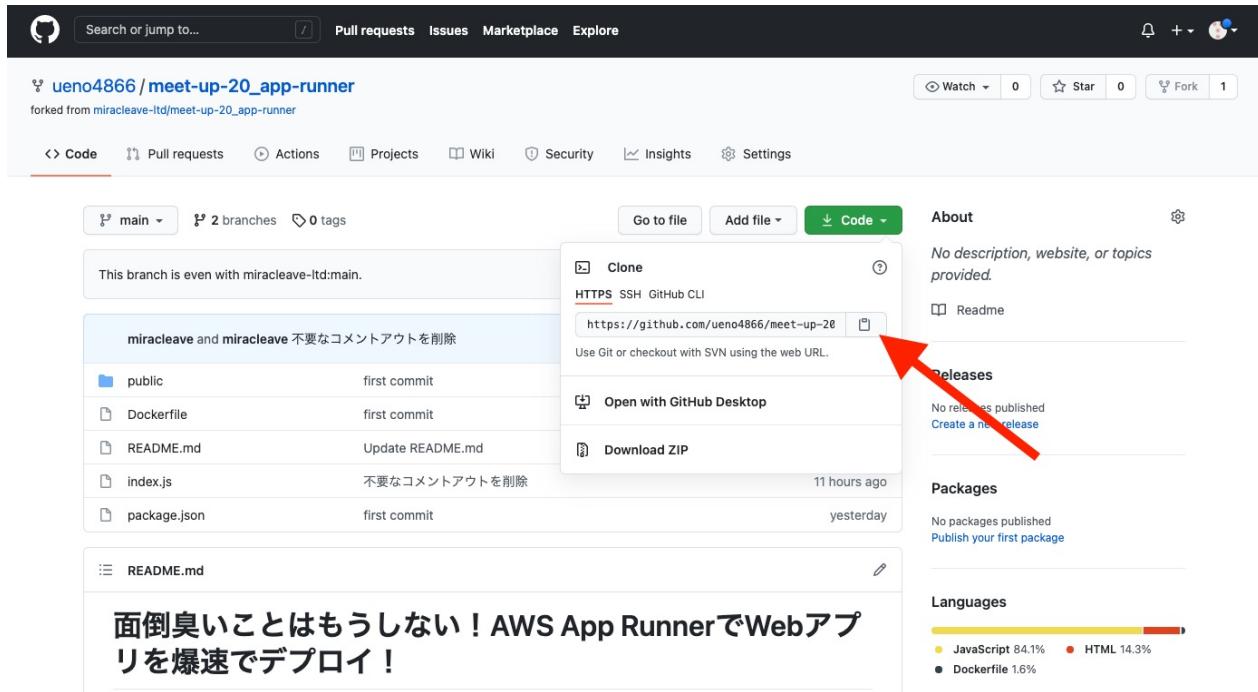


The screenshot shows a GitHub repository page. At the top, there's a navigation bar with links for 'Search or jump to...', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the repository name 'miracleave-ltd / meet-up-20_app-runner' is displayed. On the far right of the header, there are buttons for 'Watch' (with 1), 'Star' (with 0), 'Fork' (with 0), and a profile icon. A red arrow points to the 'Fork' button. The main content area shows the repository's codebase, which includes files like 'public', 'Dockerfile', 'README.md', 'index.js', and 'package.json'. The 'README.md' file contains the text: '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！' (No more annoying things! Deploy your Web app quickly with AWS App Runner!). Below the README, there's a section titled '事前準備' (Preparation) with a bullet point: 'Dockerのインストール'. On the right side of the page, there are sections for 'About', 'Releases', 'Packages', 'Environments', and 'Languages'.

※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。

3. クローン用URL取得

自分のリポジトリにコピーされたアプリのURLをコピーします。



The screenshot shows a GitHub repository page for a forked repository. The URL in the address bar is 'https://github.com/ueno4866/meet-up-20_app-runner'. The repository structure and README content are identical to the original. In the top right, there are buttons for 'Watch' (0), 'Star' (0), 'Fork' (1), and a profile icon. A red arrow points to the 'Clone' button in the 'Code' dropdown menu, which displays options for 'HTTPS', 'SSH', 'GitHub CLI', and a copy link. The right sidebar contains sections for 'About', 'Releases', 'Packages', 'Environments', and 'Languages'.

4. アプリクローン

以下の操作を行い、GitHubよりアプリを取得します。

例：Desktopにクローンする場合(Windowsの型はスキップしてください。)

```
cd ~/Desktop
```

アプリをクローンします。
GitHubよりコピーしたURLを使用してください。

```
git clone [コピーしたURL]
```

最終行に `done.` が表示されれば、完了です。

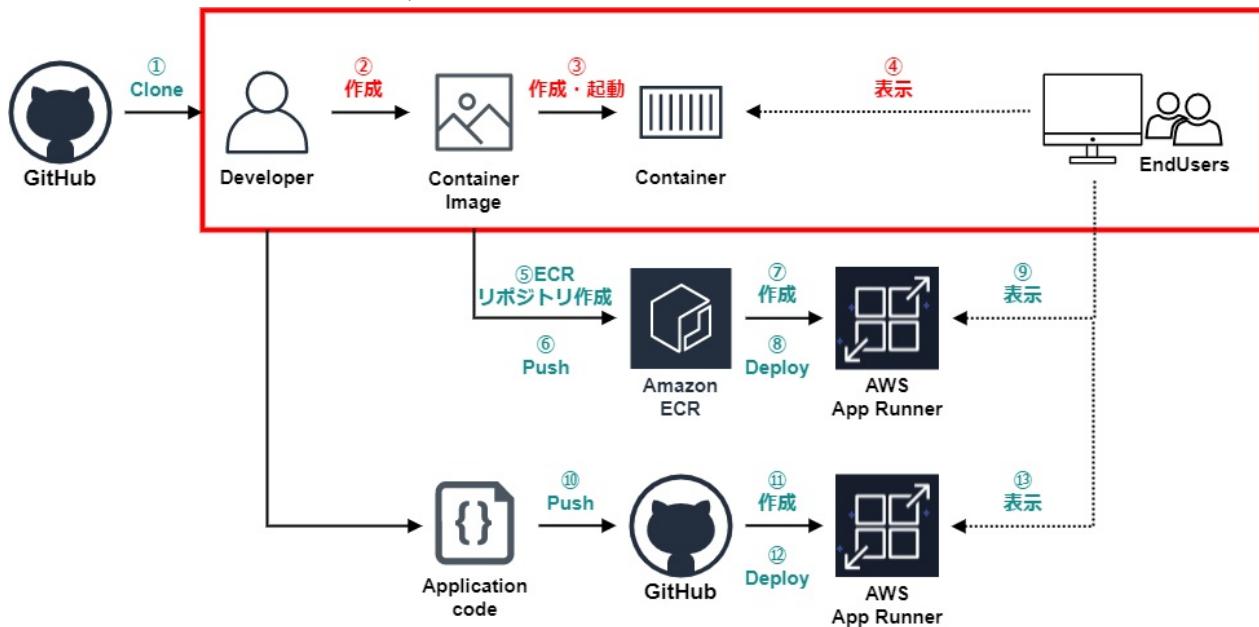
```
Cloning into 'meet-up-20_app-runner'...
remote: Enumerating objects: 363, done.
remote: Counting objects: 100% (363/363), done.
remote: Compressing objects: 100% (289/289), done.
remote: Total 363 (delta 78), reused 329 (delta 54), pack-reused 0
Receiving objects: 100% (363/363), 8.38 MiB | 24.38 MiB/s, done.
Resolving deltas: 100% (78/78), done.
```

デモアプリクローン

- 1. Githubリポジトリ表示
- 2. リポジトリコピー
- 3. クローン用URL取得
- 4. アプリクローン

デモアプリ動作確認

この手順では次の手順を進めていきます。



1. イメージ作成

meet-up-20_app-runner フォルダに移動します。

```
cd meet-up-20_app-runner
```

Dockerイメージのビルドします。

```
docker build . -t app-runner-example
```

2. コンテナの作成、起動

次のコマンドを実行し、ローカル環境でビルドしたイメージを実行します。

```
docker run -p 3333:3333 -d app-runner-example
```

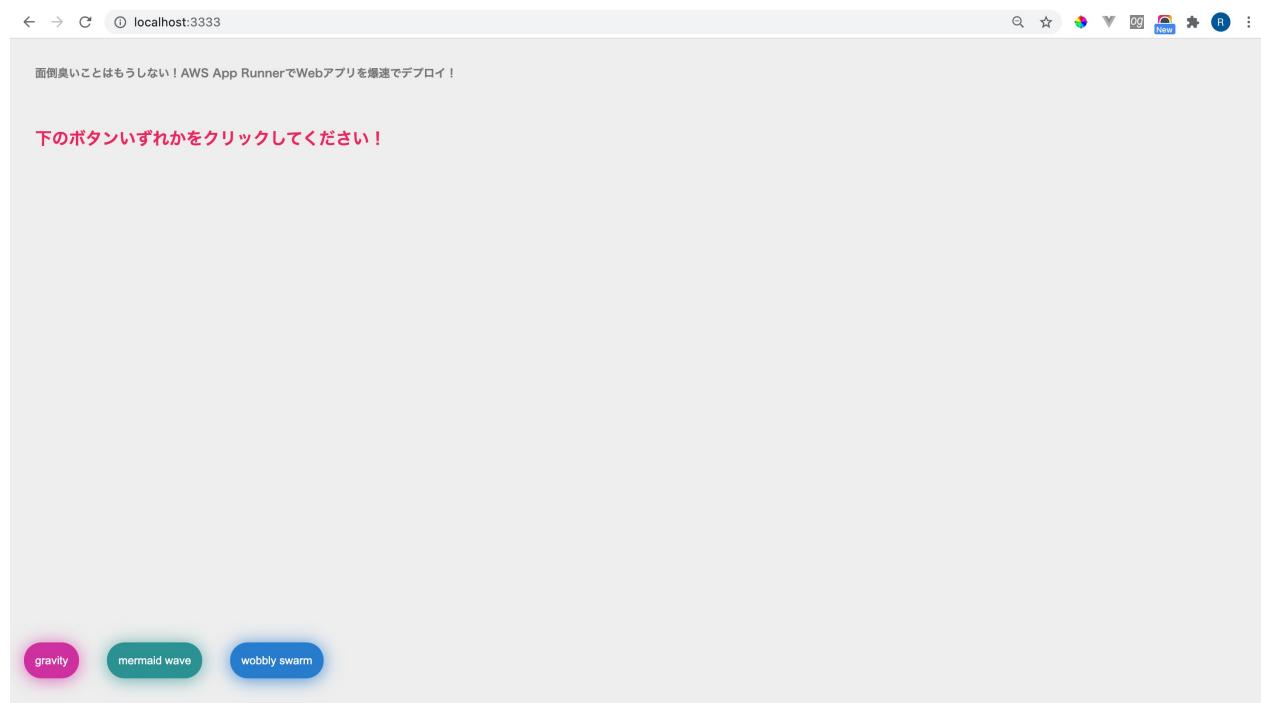
3. アプリ確認

次のリンクよりデモアプリが動作しているか確認します。

URL: <http://localhost:3333>

※実際に動いていれば下記の画面が表示されます。

2. アプリ動作確認

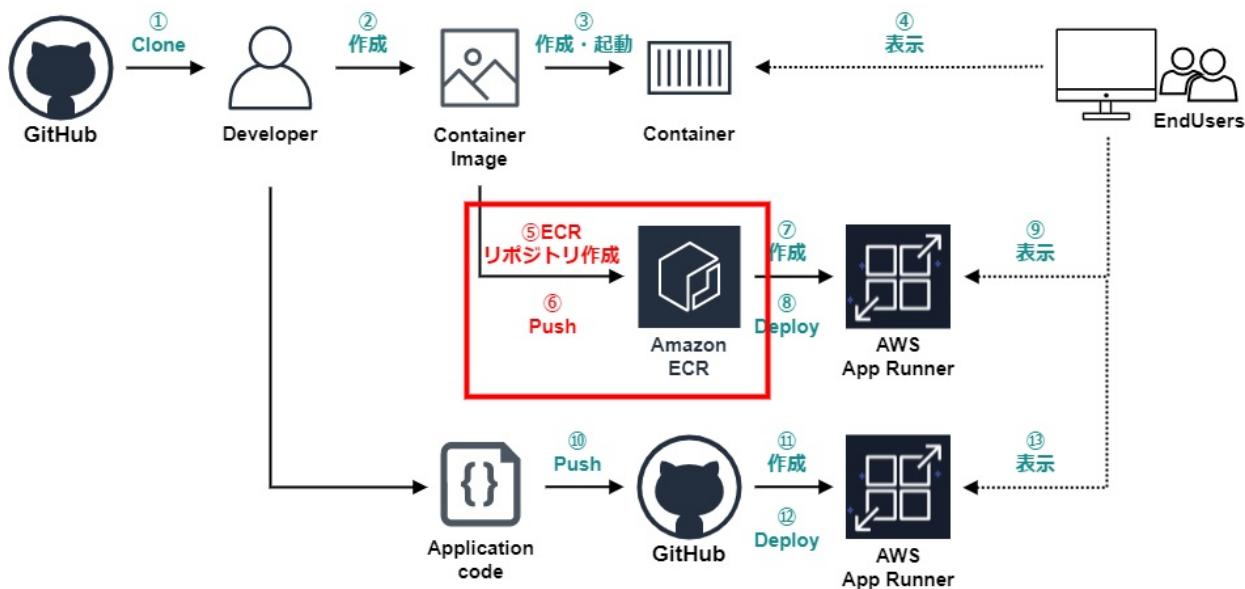


デモアプリ動作確認

- 1. イメージ作成
- 2. コンテナの作成、起動
- 3. アプリ確認

コンテナレジストリパターン

この手順では序盤について、次の手順を進めていきます。



今回はコンテナレジストリにECRを利用します。

1. ECRリポジトリ作成

AWSにログインし、検索欄から「ECR」と検索します。

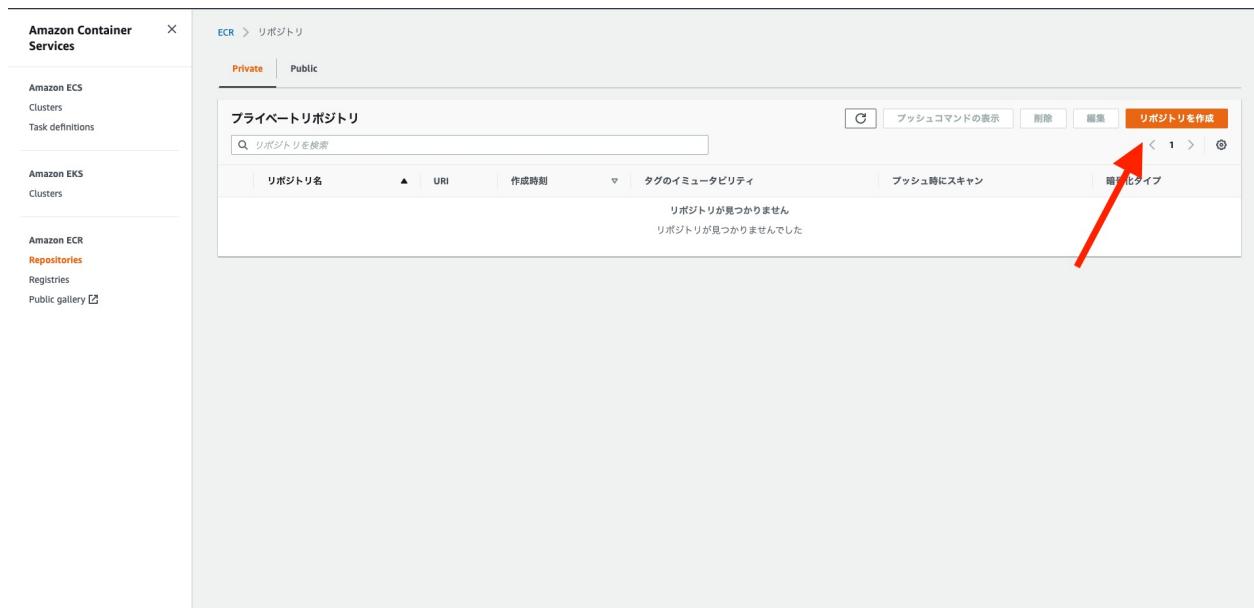
※次のリンクでも表示します。

Amazon ECR Repositories

The screenshot shows the AWS Management Console with the search bar set to 'ECR'. The left sidebar lists services like Amazon ECS, Amazon EKS, and Amazon ECR. The main search results page displays various AWS services and features. A red arrow points to the 'Repository' section under the 'Elastic Container Registry' heading.

リポジトリを作成ボタンをクリックします。

3. コンテナレジストリパターン



リポジトリ名に app-runner-example と入力します。

Amazon Container Services

ECR > リポジトリ > 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
イメージフルについて、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/

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が上書きを防ぎます。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**

2. ECR用IAM User作成

ローカルからECRにpushするためのIAM Userを作成します。

注意：すでに「Administrator Access」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。

2.1. ECR用ポリシー作成

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します。

The screenshot shows the AWS IAM console with the 'Policy' section selected. The main area displays a list of existing policies, each with a name, type, and description. At the top right of this list, there is a blue button labeled 'Create Policy'. A red arrow points directly at this button.

ポリシー名	タイプ	として使用	説明
AccessForAppRunner	カスタマー管理	なし	AccessForAppRu...
Application	カスタマー管理	アクセス許可ポリシー (1)	Application
AwsAndInfraDeveloperPolicy	カスタマー管理	なし	AwsAndInfraDevelop...
AwsAndInfraDirectorPolicy	カスタマー管理	なし	AwsAndInfraDirect...
AWSLambdaBasicExecutionRole-72a993fb-35e0-4e67-a7f7-db220b0d8713	カスタマー管理	アクセス許可ポリシー (1)	
Batch-S3	カスタマー管理	アクセス許可ポリシー (1)	Batch-S3
Operation	カスタマー管理	アクセス許可ポリシー (1)	Operation
s3crr_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 (1)	
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 AW...
AWSSSOAdministrator	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...

JSONを選択、下記をコピーし貼り付けてください。

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

貼り付け後、次のステップ：タグ ボタンをクリックします。

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. }

```

セキュリティ: 0 エラー: 0 警告: 0 提案: 0 文字数: 215 / 6,144。 キャンセル 次のステップ: タグ

タグの設定は不要です。

次のステップ：確認 ボタンをクリックします。

次の設定値を入力し、ポリシーの作成 ボタンをクリックし、ポリシーを作成します。

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
Elastic Container Registry	制限: 読み込み	すべてのリソース	なし

許可 (285 サービス中 1) 残りの 284 を表示

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

2.2. IAM User作成

3. コンテナレジストリパターン

左メニューからユーザーを選択し、**ユーザーを追加**ボタンをクリックします。

The screenshot shows the AWS IAM User Management console. On the left, there's a sidebar with navigation links like Dashboard, Access Management, and User Management. The main area shows a table of users with columns for User Name, Group, Last Activity, MFA, Password Last Used, and Active. There are four users listed. At the top right of the table, there's a blue button labeled 'User to Add'. A red arrow points to this button.

次の設定値を入力し、**次のステップ：アクセス制限**ボタンをクリックします。

ユーザー名：
アクセスの種類： プログラムによるアクセスにチェックします。

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

3. コンテナレジストリパターン

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します。

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 ▼ Q ECR 4件の結果を表示中

ポリシー名	タイプ	次として使用
<input checked="" type="checkbox"/> AccessEcrForAppRunner	ユーザーによる管理	なし
<input type="checkbox"/> AWSAppRunnerServicePolicyForECRAccess	AWS による管理	Permissions policy (1)
<input type="checkbox"/> EC2InstanceProfileForImageBuilderECRContainerBuilds	AWS による管理	なし
<input type="checkbox"/> SecretsManagerReadWrite	AWS による管理	なし

▼ 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 のダウンロード ボタンをクリックし、IAM User の認証情報が記載されている CSV をダウンロードします。

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「[REDACTED] 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
<input checked="" type="checkbox"/> meet-up-app-runner-user	AKIA2QBOUKIYGAWSCTXC	***** 表示

3. Dockerイメージプッシュ

3.1. AWS認証情報の設定

aws-cliを利用して、ECRにイメージをプッシュします。
それに伴い、credentialsの設定を行います。

```
# awsフォルダを作成  
mkdir aws
```

先ほど、CSVでダウンロードしたIAM Userの認証情報(アクセスID,アクセスキー)を次のイコールの後に値を設定します。

Macの場合

```
cat <<EOF > aws/credentials  
[default]  
aws_access_key_id = [Access key ID]  
aws_secret_access_key = [Secret access key]  
EOF
```

Windowsの場合

```
(  
echo [default]  
echo aws_access_key_id = [Access key ID]  
echo aws_secret_access_key = [Secret access key]  
) >> aws/credentials
```

configの設定をします。

Macの場合

```
cat <<EOF > aws/config  
[default]  
region = ap-northeast-1  
output = json  
EOF
```

Windowsの場合

```
(  
echo [default]  
echo region = ap-northeast-1  
echo output = json  
) >> aws/config
```

ECR画面に移動し、プッシュコマンドを確認し、ECRにpushします。

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。

プッシュコマンドを確認します。

後続のECRイメージプッシュ手順で利用します。

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 [\[リンク\]](#) 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 [\[リンク\]](#) 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/app-runner-example:latest
```

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

3.2. ECRイメージプッシュ

1. AWS CLIでAWSにログインします。

Macの場合

```
例: docker run --rm -ti $(pwd)/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
```

```
docker run --rm -ti -v $(pwd)/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）]
```

Windowsの場合

```
例: docker run --rm -ti -v %cd%/aws:/root/.aws -v %cd%:/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
```

```
docker run --rm -ti -v %cd%/aws:/root/.aws -v %cd%:/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）]
```

ログインが成功すると次のメッセージが表示されます。

Login Succeeded

2. app-runner-exampleのプッシュコマンドの3を実行します。

```
例: docker tag app-runner-example:latest 0000000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example:latest
```

3. app-runner-exampleのプッシュコマンドの4を実行します。

```
例: docker push 0000000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example:latest
```

プッシュコマンドが成功すると、次のメッセージが表示されます。

```
The push refers to repository [0000000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example]
```

```
00000d76f0b8: Pushed
```

```
00000cbf4ab0: Layer already exists
```

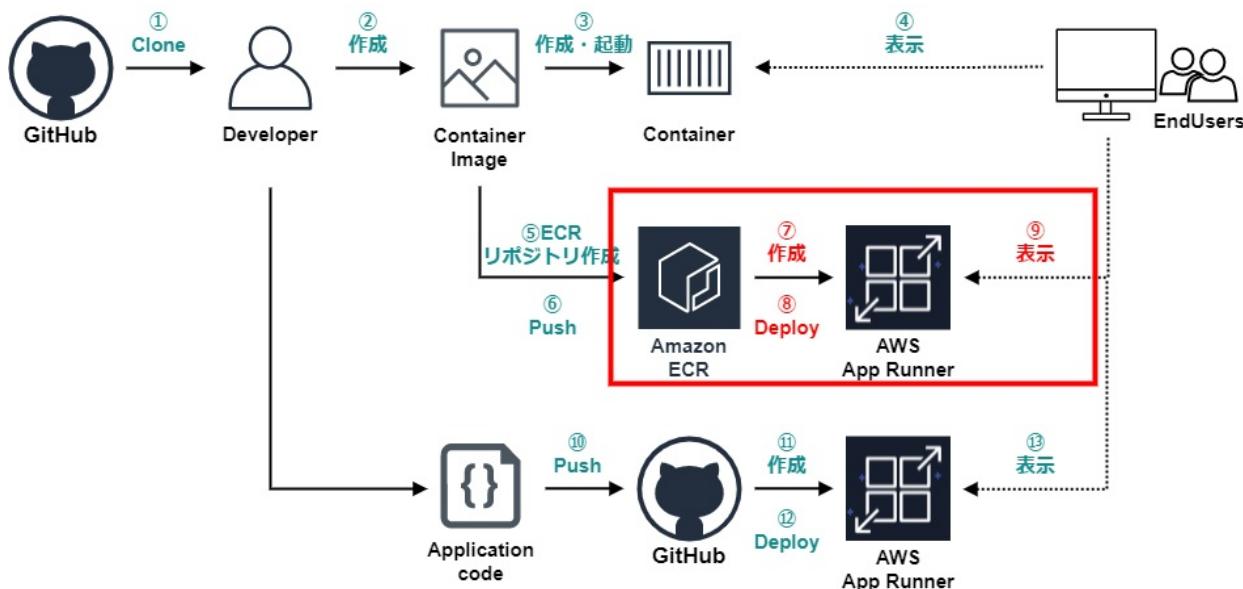
```
...
```

```
latest: digest: sha256:97c000000000... size: 3051
```

4. App Runnerデプロイ

AWSのApp Runnerの画面に移動し、App Runnerの設定、デプロイを行います。

以降の手順では次の手順を進めていきます。



4.1. AppRunner作成

App Runnerの画面に移動します。※次のリンクでも表示します。

AWS App Runner

The screenshot shows the AWS App Runner service page. In the search bar at the top right, 'App Runner' is typed. Below the search bar, a list of services is displayed, with 'AWS App Runner' highlighted in orange. A red arrow points from the text above to this highlighted item. To the right of the search results, there's a sidebar with sections for 'App Runner の使用を開始' (Get started with App Runner), 'コンピューティングコスト (米国)' (Computing costs (US)), 'オプションのアドオンコスト (米国)' (Optional add-on costs (US)), and 'FAQ'.

App Runnerサービスを作成するボタンをクリックします。

The screenshot shows the 'Create New Service' wizard for AWS App Runner. The first step, 'AWS App Runner', is selected. The wizard consists of five steps: 1. AWS App Runner (selected), 2. Add a source code or container image, 3. Configure build and service settings, 4. Review and verify all your settings, Create and deploy your service, and 5. Receive a secure URL. The sidebar on the right contains sections for 'App Runner の使用を開始', 'コンピューティングコスト (米国)', 'オプションのアドオンコスト (米国)', and 'FAQ'.

4.1.1. App Runnerサービスの設定、デプロイを行います。

コンテナイメージのURIは、先ほど作成したECRリポジトリを選択します。

App Runner > サービスの作成

ステップ1
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](#)

サービスを設定します。

3. コンテナレジストリパターン

App Runner > サービスの作成

サービスを設定 Info

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 譲号化キーを指定

▶ 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します。

キーコード

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
100
最小サイズ
1 最大サイズ
25

▼ ヘルスチェック

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

▼ セキュリティ

インスタンスロール
— AWS KMS 譲号化キー
AWS マネージド型

▼ タグ

キーコード

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**



4.2. デプロイ内容確認

デプロイが完了後、App Runner画面の公開されたURLにアクセスし、アプリが動いているかを確認します。

The screenshot shows the AWS App Runner service details page for 'app-runner-example'. It displays the service ARN and the deployed URL: [https://\[REDACTED\].ap-northeast-1.awssapprunner.com](https://[REDACTED].ap-northeast-1.awssapprunner.com). A red arrow points to the URL field.

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。

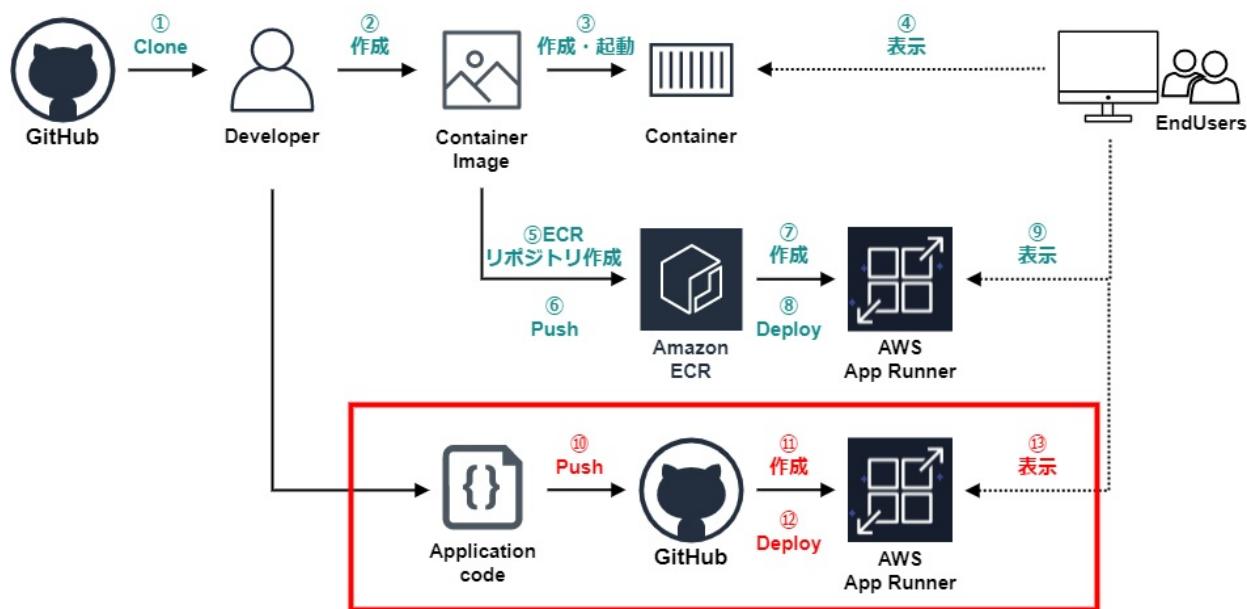
The screenshot shows a browser window displaying a success message from AWS App Runner: '面倒臭いことはもうしない！AWS App RunnerでWebアプリを爆速でデプロイ！' (No more trouble! Deploy your Web application quickly with AWS App Runner!). Below the message, it says '下のボタンいずれかをクリックしてください！' (Click any of the buttons below!). At the bottom, there are three colored buttons: 'gravity' (pink), 'mermaid wave' (teal), and 'wobbly swarm' (light blue).

コンテナレジストリパターン

- 1. ECRリポジトリ作成
- 2. ECR用IAM User作成
 - 2.1. ECR用ポリシー作成
 - 2.2. IAM User作成
- 3. Dockerイメージプッシュ
 - 3.1. AWS認証情報の設定
 - 3.2. ECRイメージプッシュ
- 4. App Runnerデプロイ
 - 4.1. AppRunner作成
 - 4.1.1. App Runnerサービスの設定、デプロイを行います。
 - 4.2. デプロイ内容確認

ソースコードリポジトリパターン

この手順では次の手順を進めていきます。



注意事項

- ソースコードリポジトリは、GitHubのみ対応しています。
- サービスランタイムは、Python3 / Nodejs 12のみとなります。

次のリンクよりApp Runnerページへ遷移します。

[App Runner](#)

App Runnerサービスを作成 ボタンをクリックします。

The screenshot shows the AWS App Runner service creation interface. At the top right, there is a red box around the 'App Runner の使用を開始' button, which contains another red box around the 'App Runner サービスを作成' button. A red arrow points from the text above to this button. On the left, there is a diagram titled '仕組み' showing a flow from 'AWS App Runner' to 'Add a source', then 'Configure build and service settings', 'Review and create', and finally 'Receive a secure URL'. To the right of the diagram, there are two tables: 'コンピューティングコスト (米国)' and 'オプションのアドオンコスト (米国)'. The first table shows costs for vCPU (\$0.064 /時間) and GB (\$0.007 /時間). The second table shows costs for 自動デプロイ (\$1 /アプリケーション/月) and 構築 (\$0.005 /分). At the bottom, there are links for 'フィードバック' and '日本語'.

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)

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)

自動デプロイ	\$1 /アプリケーション/月
構築	\$0.005 /分

ソースコードリポジトリを選択します。

4. ソースコードリポジトリパターン

The screenshot shows the AWS App Runner 'Create Service' wizard, Step 1: Source and Deployment. The 'Source' tab is selected. Under 'Deployment Type', the 'Source Code Repository' option is highlighted with a red box and a red arrow pointing to it. Other options like 'Container Registry' and 'Amazon ECR' are shown but not selected. The 'Deployment Provider' section shows 'Amazon ECR' selected. The 'Container Image URI' field contains '111111111111.dkr.ecr.us-east-1.amazonaws.com/myfirstrepo:latest'. The 'Deployment Settings' section has 'Manual' selected for deployment triggers. The 'ECR Access Role' section shows 'Custom Role' selected. At the bottom, there are 'Cancel' and 'Next Step' buttons.

GitHubとAWSを連携させるため、新規追加ボタンをクリックします。

4. ソースコードリポジトリパターン

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。

GitHub に接続 [Info](#)

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。

新規追加

リポジトリ

プランチ

デプロイ設定

デプロイトリガー

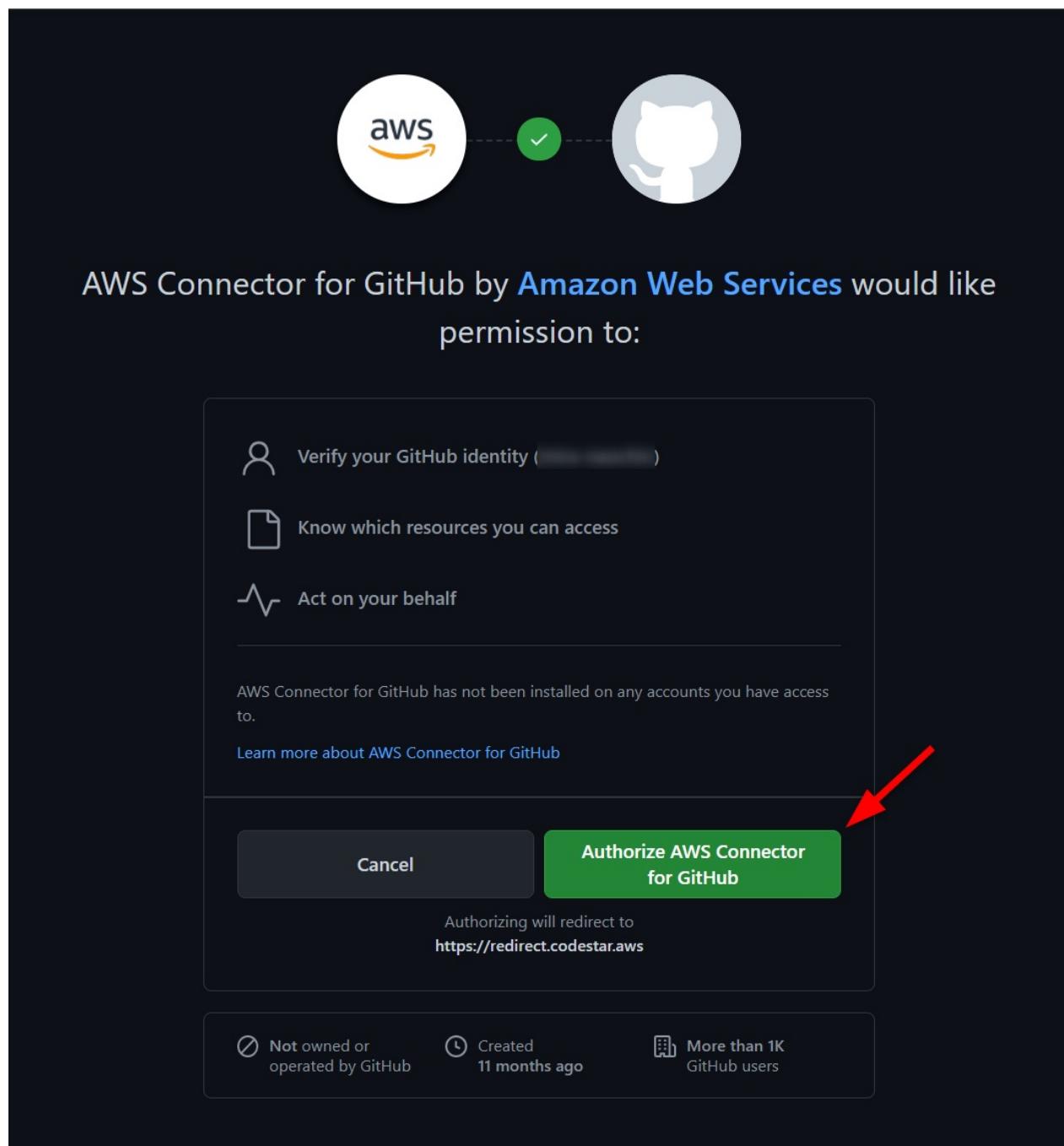
手動
App Runner コンソールまたは AWS CLI を使用して、各デプロイを自分で開始します。

自動
このプランチへのすべてのプッシュは、サービスの新しいバージョンをデプロイします。

キャンセル 次へ

「AWS Connector for GitHub」というアプリをGitHubアカウントにインストールするための同意画面が表示されます。

Authorize AWS Connector for GitHub ボタンをクリックします。



別のアプリケーションをインストールする をクリックします。

Create a new connection

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。

meetup-example

この名前は一意である必要があります。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合から選択するか、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 **別のアプリケーションをインストールする**

キャンセル **次へ**

フィードバック 日本語 ▾ プライバシーポリシー 利用規約 Cookie の設定

© 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。

「AWS Connector for GitHub」をインストールする GitHub アカウントを選択してください。

Search or jump to... / Pulls Issues Marketplace Explore

aws

Install AWS Connector for GitHub

Where do you want to install AWS Connector for GitHub?

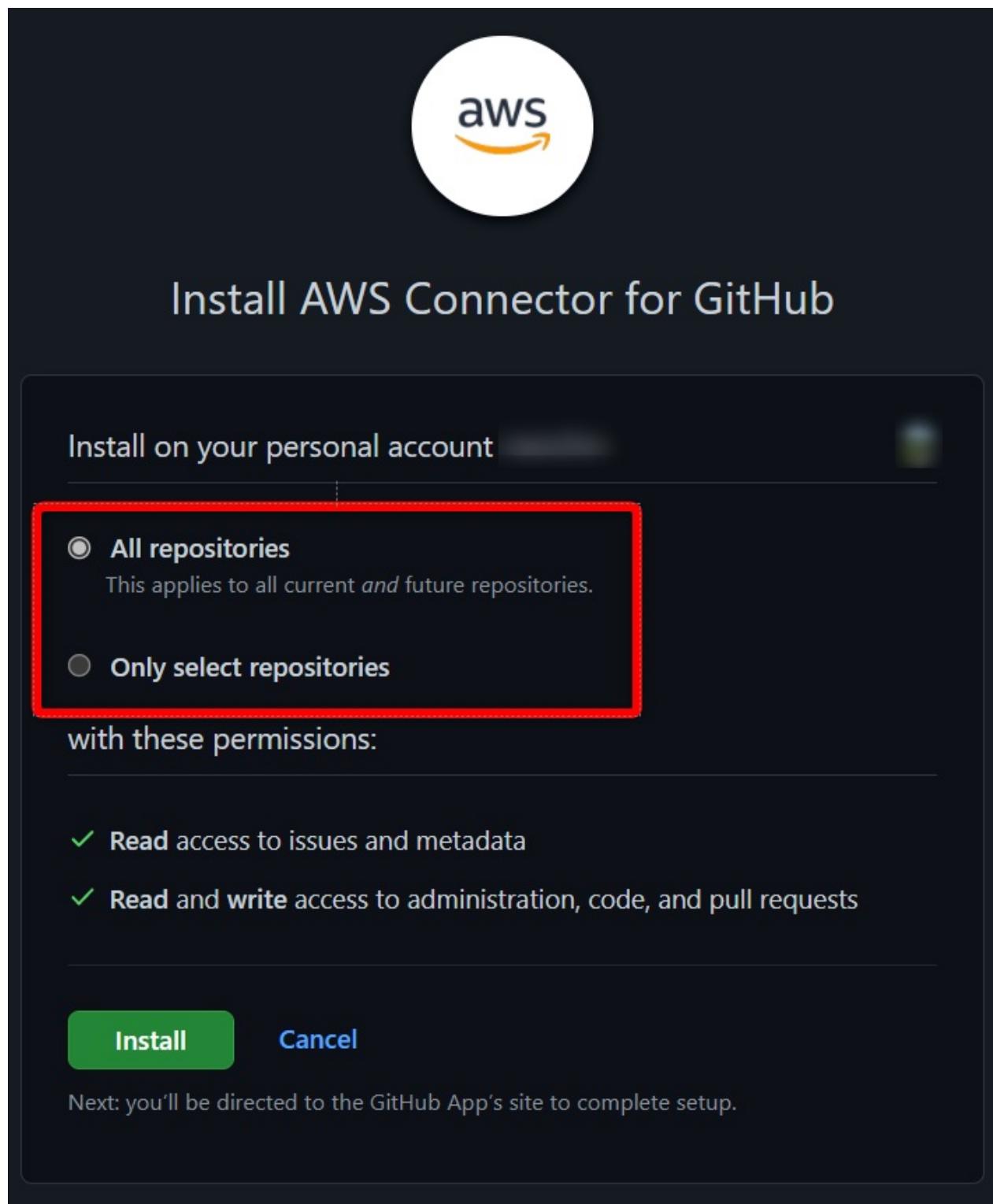
... >

... >

... >

© 2021 GitHub, Inc. Terms Privacy Security Status Docs
Contact GitHub Pricing API Training Blog About

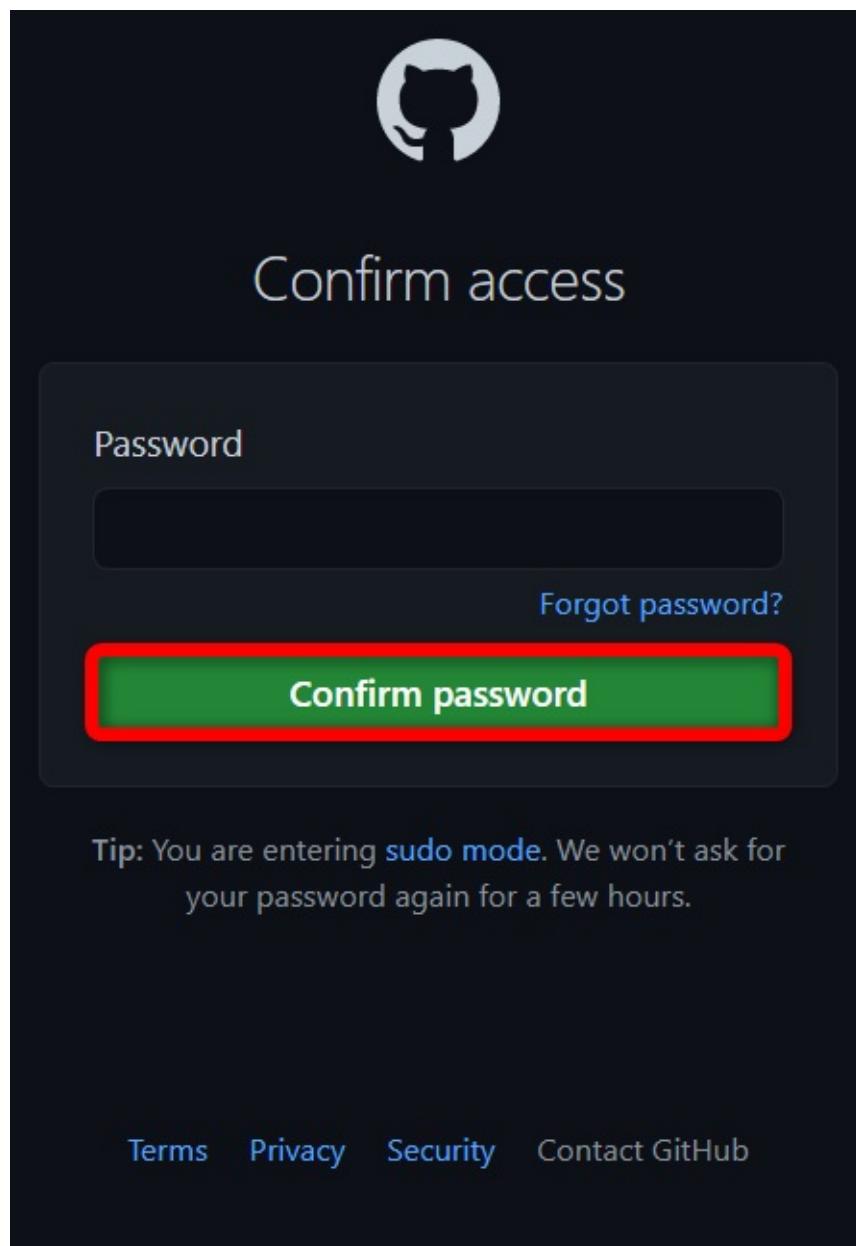
「AWS Connector for GitHub」をインストールするリポジトリを選択してください。



選択したリポジトリが次のように「XXXXXX/meet-up-20_app-runner」となっていることを確認し、install ボタンをクリックしてください。



パスワード入力画面が表示した場合は、パスワードを入力し、Confirm password ボタンをクリックしてください。



GitHubアプリケーションのセレクトボックスに対象GitHubアカウント名が表示されたら、 次へ ボタンをクリックしてください。

Create a new connection

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。

meetup-example

この名前は一意である必要があります。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合から選択するか、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。

別のあるアプリケーションをインストールする

キャンセル 次へ

以下の設定値を選択し、次へ ボタンをクリックしてください。

GitHubに接続 : meetup-example リポジトリ : meet-up-20_app-runner ブランチ : main デプロイトリガー : 自動

Sources and Deployment

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。

GitHub に接続 Info

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。

meetup-example

リポジトリ
meet-up-20_app-runner

ブランチ
main

Deployment Settings

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
このブランチへのすべてのプッシュは、サービスの新しいバージョンをデプロイします。

キャンセル 次へ

次の設定値を入力し、 次へ ボタンをクリックしてください。

設定ファイル：ここですべての設定を構成する ランタイム：Nodejs 12 構築コマンド：npm install 開始コマンド：node index.js ポート：3333

構築を設定 [Info](#)

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

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 TCP ポートが使用されます。

3333

キャンセル 戻る **次へ**

次の設定値を入力し、 次へ ボタンをクリックしてください。

サービス名：meetup-app-runner 仮想CPU：1 vCPU メモリ：2 GB

4. ソースコードリポジトリパターン

サービス名
meetup-app-runner

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

▶ 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 次へ

内容の確認を行い、特に問題がなければ、**作成とデプロイ** ボタンをクリックします。

4. ソースコードリポジトリパターン

The screenshot shows the AWS App Runner 'Create Service' wizard. The steps are as follows:

- Step 1: Source and Deployment**
 - Sources**: GitHub, Connection: arn:aws:apprunner:ap-northeast-1:123456789012:connection/meetup-example/, Branch: main.
 - Deployment Settings**: Deployment method: Automatic Deployment.
- Step 2: Build Settings**
 - Build Settings**: Runtime: Node.js 12, Port: 3000, Build Command: npm install, Start Command: node index.js.
- Step 3: Service Settings**
 - Service Settings**: Service Name: meetup-app-runner, Resource Type: Virtual CPU & Memory: 1 vCPU & 2 GB.
 - Auto Scaling**: Maximum Size: 25, Minimum Size: 1, Concurrent Executions: 100.
 - Health Check**: Health Check Interval: 10 seconds, Unhealthy Threshold: 5 requests, Healthy Threshold: 1 request.
 - Security**: Instance Role: - (empty), AWS KMS Encryption Key: AWS Managed.
 - Tags**: No tags are defined.
- Buttons at the bottom**: Cancel, Back, Create and Deploy (highlighted in orange).

ステータスが Running になると環境作成完了です。

ドメインが発行されますので、デフォルトドメインのURLをクリックします。

The screenshot shows the AWS App Runner service details page for 'meetup-app-runner'. The status is 'Running' (highlighted with a red box). The default domain is listed as 'https://.ap-northeast-1.awsapprunner.com' (also highlighted with a red box). A red arrow points to this domain URL. Below the status, there is a table with columns for 'イベント', 'ステータス', '開始済み', and '終了済み'. One entry shows 'Create service' with 'Pending' status and a timestamp of '2021/8/4 8:11:06 UTC'. At the bottom of the page, there is a footer with links for 'フィードバック', '日本語', 'Cookie の設定', and copyright information.

次のページが表示されれば、デプロイ完了です。

The screenshot shows a deployment completion page. The main message is '面白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！' (No more fun things! Deploy your Web app quickly with AWS App Runner!). Below this, there is a pink button labeled 'gravity', a teal button labeled 'mermaid wave', and a blue button labeled 'wobbly swarm'. The background is light gray.

ゴミ掃除

1. App Runnerサービス削除

次のリンクよりApp Runnerサービス一覧を表示します。

[App Runnerサービス一覧](#)

The screenshot shows the AWS App Runner service list. A specific service named "meetup-app-runner" is selected. In the top right corner of the service card, there is a "Actions" dropdown menu with options: "One-time Stop", "Restart", and "Delete". The "Delete" button is highlighted with a red box.

サービスの概要

- ステータス: Running
- デフォルトドメイン: <https://.ap-northeast-1.awsapprunner.com>
- サービス ARN: arn:aws:apprunner:ap-northeast-1:...:service/meetup-app-runner/
- ソース: <https://github.com/>

アクティビティ (1)

イベント	ステータス	開始済み	終了済み
Create service	Succeeded	2021/8/4 8:11:06 UTC	2021/8/4 8:14:57 UTC

画面に表示されている指示通り、入力欄に `delete` を入力し、削除ボタンをクリックします。

meetup-app-runner を削除

この App Runner サービスを削除すると、リソース(カスタムドメインなど)とのすべての関連付けも削除されます。

削除を確認するには、このフィールドに `delete` と入力します。

ここでは、大文字と小文字が区別されます。

削除

本日作成したサービスが削除されていることを確認します。

The screenshot shows the AWS App Runner service list. At the top, there's a search bar labeled "サービスをフィルタリング" and a button "サービスの作成". Below the header, there are filters for "サービス名", "ステータス", "デフォルトドメイン", "作成済み", "最後のアクティビティ", and "ARN". A large red box highlights the main content area which displays the message "サービスなし" (No services) and "表示するサービスがありません。" (No services to display). At the bottom, there are two buttons: "C" and "サービスの作成".

2. IAM削除

2.1. IAMユーザー削除

次のリンクよりIAMユーザー一覧を表示します。

[IAMユーザー](#)

今回作成したユーザーを選択し、削除ボタンをクリックします。

The screenshot shows the IAM User list. At the top, it says "ユーザー (選択済み 1/1) 情報" and "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。". Below is a search bar "ユーザー名またはアクセスキーでユーザーを検索". The user list table has columns: チェックボックス (with one checked), ユーザー名 (meet-up-app-runner-user), グループ (なし), 最後のアクティビティ (1時間前), MFA (なし), and パスワードが作成されてから経過した時間 (なし). A red arrow points to the "削除" (Delete) button at the top right of the table.

画面に表示されている指示通り、入力欄に `meet-up-app-runner-user` を入力し、削除ボタンをクリックします。

The screenshot shows a confirmation dialog box. The title is "meet-up-app-runner-user を削除しますか?". The message inside says "meet-up-app-runner-user を完全に削除しますか? これにより、そのすべてのユーザーデータ、セキュリティ認証情報、およびインラインポリシーも削除されます。". Below the message is the text "このアクションは元に戻すことができません。". At the bottom, there is a text input field containing "meet-up-app-runner-user" and two buttons: "キャンセル" and a large blue "削除" button.

対象のユーザーが削除されていることを確認します。

The screenshot shows the AWS IAM 'Users' list. At the top right are buttons for 'Delete' and 'Add User'. Below the search bar, there are filters for 'User Name', 'Group', 'Last Activity', 'MFA', 'Password Last Used', and 'Active Session'. A red box highlights the message '表示するリソースがありません' (No resources to display) at the bottom.

2.2. IAMポリシー削除

次のリンクよりIAMポリシー一覧を表示します。

IAMポリシー

作成したIAMポリシーを選択し、アクションボタンのドロップダウンより削除ボタンをクリックします。

The screenshot shows the AWS IAM 'Policies' list. The URL is 'IAM > Policies'. At the top right are buttons for 'Actions ▲' and 'Create Policy'. A red arrow points from the 'Actions ▲' button to the 'Delete' option in the dropdown menu. Another red box highlights the 'Delete' button in the list table. A red circle highlights the policy name 'AccessEcrForAppRunner'.

画面に表示されている指示通り、入力欄に AccessEcrForAppRunner を入力し、削除ボタンをクリックします。

A confirmation dialog box with the text 'AccessEcrForAppRunner を削除しますか?' (Do you want to delete the AccessEcrForAppRunner policy?) and a red 'X' icon in the top right corner.

AccessEcrForAppRunner ポリシーとそのすべてのバージョンを完全に削除しますか?

このアクションは元に戻すことができません。

削除を確認するには、テキスト入力フィールドにポリシー名を入力します。

The confirmation dialog has a red border around the input field containing 'AccessEcrForAppRunner'. To the right are 'Cancel' and 'Delete' buttons, with 'Delete' highlighted by a red box.

検索欄に AccessEcrForAppRunner を入力し、対象のポリシーが削除されていることを確認します。

The screenshot shows the AWS IAM 'Policies' list. The search bar contains 'AccessEcrForAppRunner'. The table lists policies including 'WorkLinkServiceRolePolicy', 'WellArchitectedConsoleReadOnlyAccess', and 'WellArchitectedConsoleFullAccess'. A red box highlights the search bar and the first policy row in the table.

3. ローカルに構築したDockerイメージ削除

次のコマンドを実行し、コンテナ状態を確認します。

```
docker ps
---
CONTAINER ID   IMAGE          COMMAND           CREATED          STATUS          PORTS
NAMES
e572bd192f8d  app-runner-example "docker-entrypoint.s..."  00 seconds ago Up 00 seconds  0.0.0.0:3333->3333/tcp, :::3333->3333/tcp  nice_kare
```

上記で実行した CONTAINER ID を次のコマンドで利用します。

起動中のコンテナを停止します。

```
docker stop [CONTAINER ID]
```

ビルドして作成したコンテナイメージを削除します。

```
docker rmi -f app-runner-example
```

次のメッセージが表示されれば、成功です。

```
| Untagged: 000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example...
```

ECRにプッシュしたイメージを削除します。

コンテナ名は次の手順で利用した 000000000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example を入力してください。

```
docker rmi -f [コンテナ名]
```

次のメッセージが表示されれば、成功です。

```
| Untagged: 000000.dkr.ecr.ap-northeast-1.amazonaws.com/app-runner-example...
```

```
| Deleted: sha256:f382b74e...
```

4. フォルダの削除

次のコマンドでフォルダを完全に削除します。

Macの場合

```
# 一つ上の階層に移動
cd ..
# meet-up-20_app-runner フォルダが存在しているかを確認
ls
# meet-up-20_app-runner フォルダが存在している場合、下記コマンドを実行しフォルダを削除
rm -rf meet-up-20_app-runner
```

Windowsの場合

```
# 一つ上の階層に移動
cd ..
# meet-up-20_app-runner フォルダが存在しているかを確認
dir
# meet-up-20_app-runner フォルダが存在している場合、下記コマンドを実行しフォルダを削除
rd /s /q meet-up-20_app-runner
```

以上。

ゴミ掃除

- 1. App Runnerサービス削除
- 2. IAM削除

- 2.1. IAMユーザー削除
- 2.2. IAMポリシー削除
- 3. ローカルに構築したDockerイメージ削除
- 4. フォルダの削除