AWS App Runner ワークショップ

2021/06/30

シニアエバンジェリスト 亀田治伸

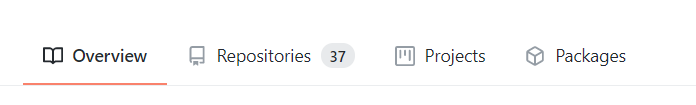
**[はじめに]**

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。

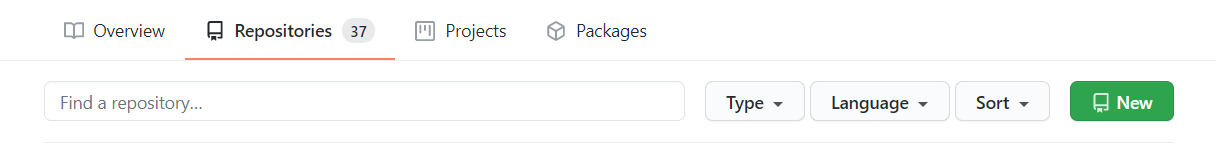
https://www.apprunnerworkshop.com/

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。

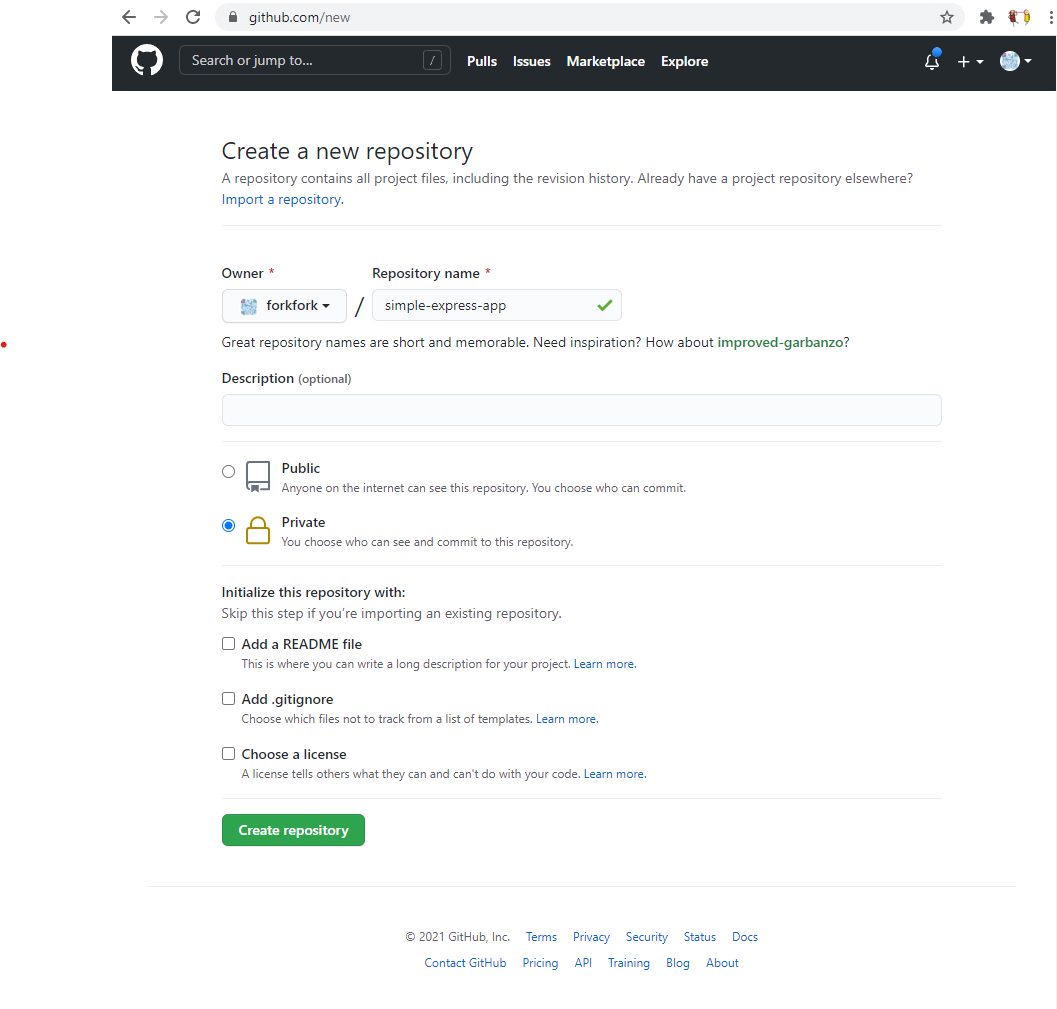
1. GitHubにアクセスしてログインします。
2. [Repositories]をクリックします



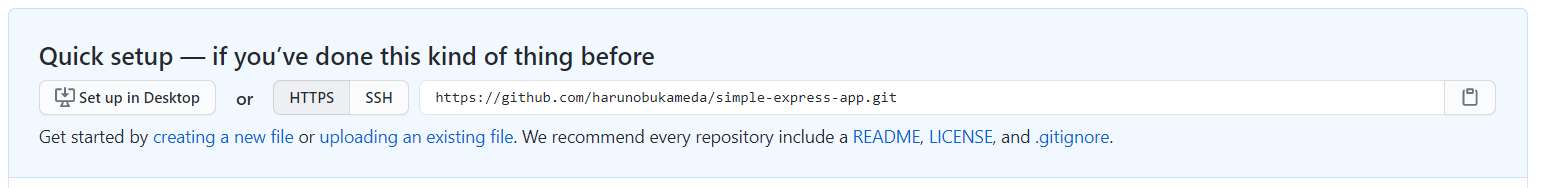
1. [New]ボタンをクリックします



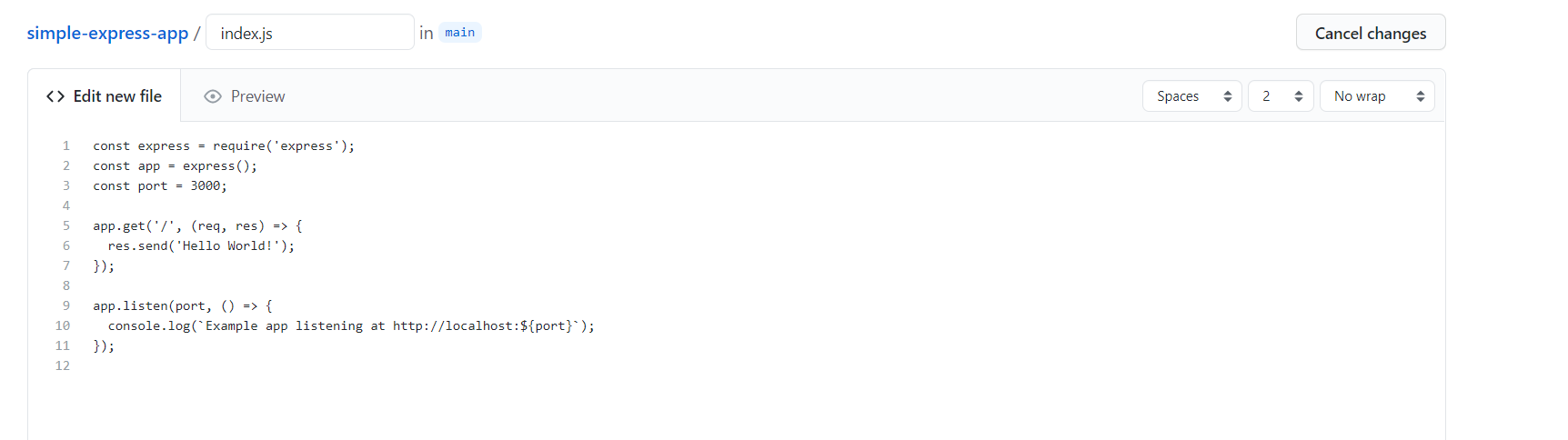
1. [simple-express-app]と入力し、[Create repository]ボタンをおします



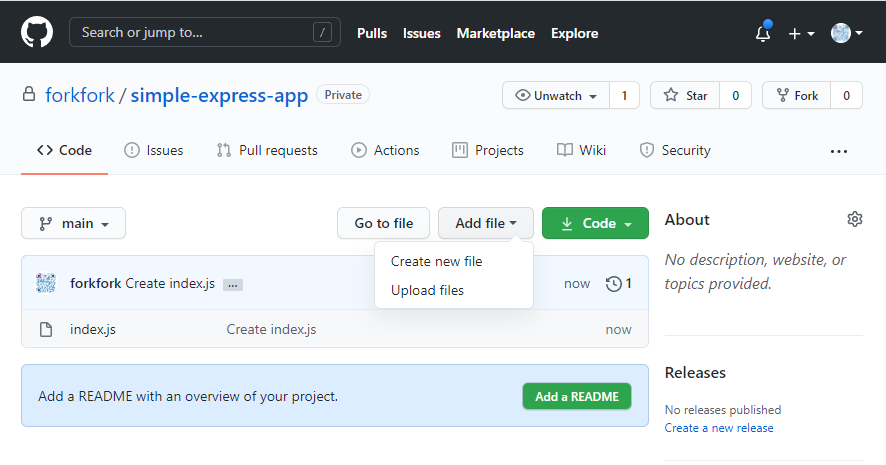
1. [creating a new file]のリンクをクリックします



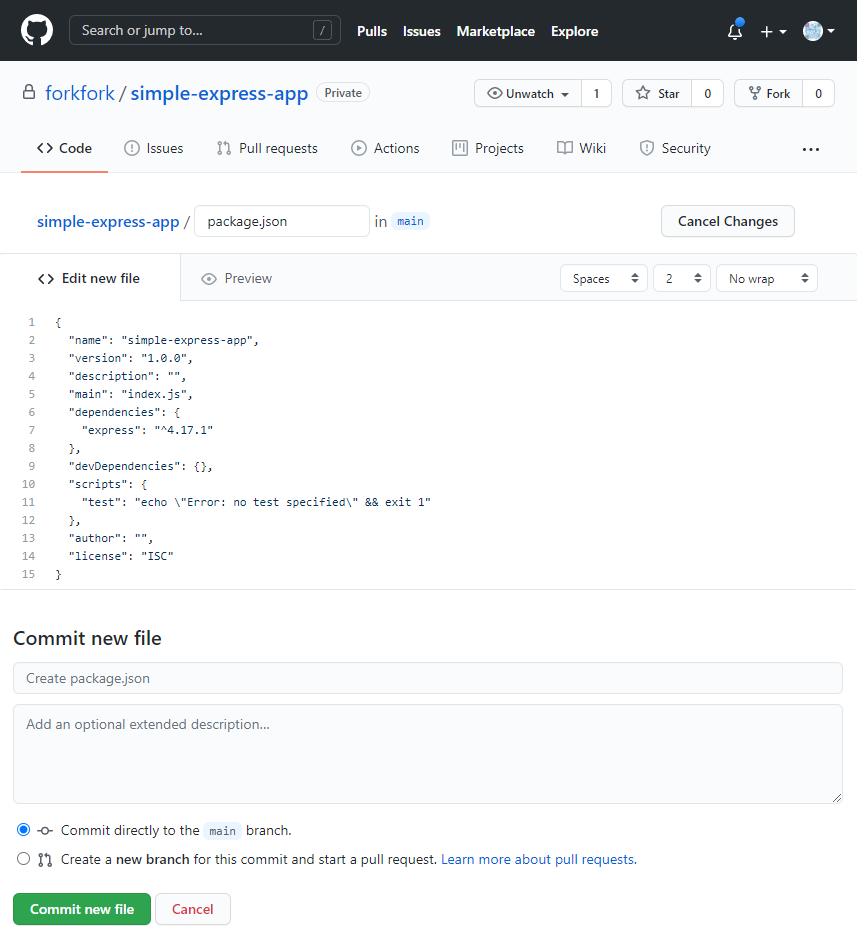
1. [index.js]と入力し、以下のコマンドを貼り付けます（commands.txtの1番）



1. [Commit new file]をおします
2. 次に、[Add file]から[Create new file]を選びます。



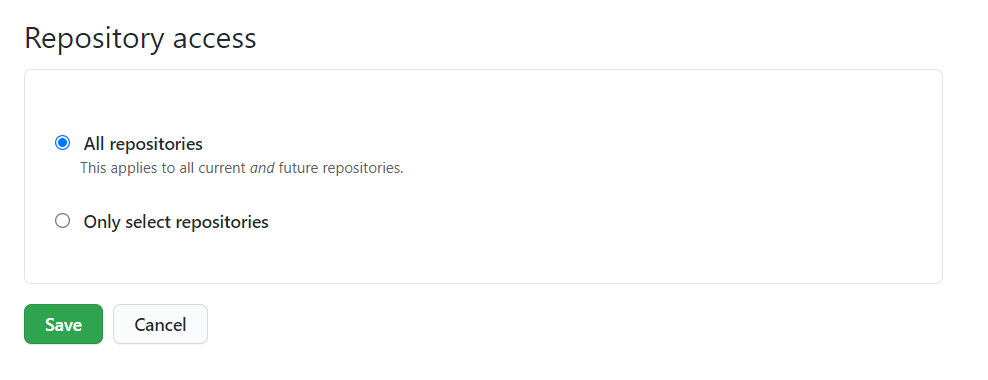
1. [package.json]と名前を付けて、コマンド2番をコピーします。



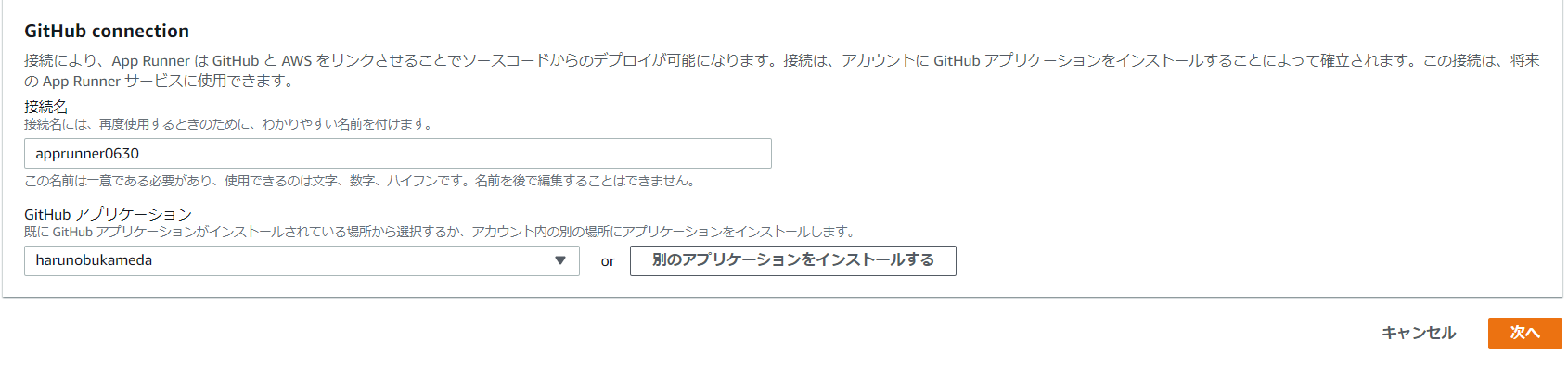
1. [Commit new file]をおします
2. AWSマネージメントコンソールのApp Runnerにアクセスし、[サービスの作成]をおします
3. [ソースコードレポジトリ]を選びます



1. （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
2. [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
3. [Save]ボタンをおします



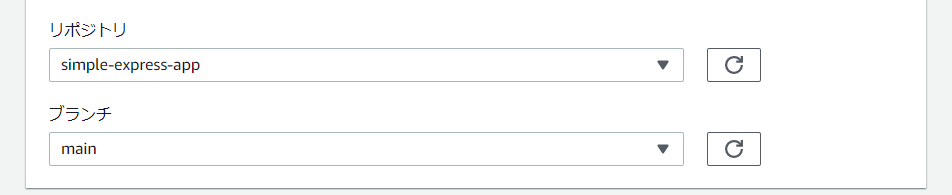
1. GitHubアプリケーションにご自身のGitユーザー名が表示されていることを確認したら、適当な名前を接続名につけて[次へ]をおします



1. 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



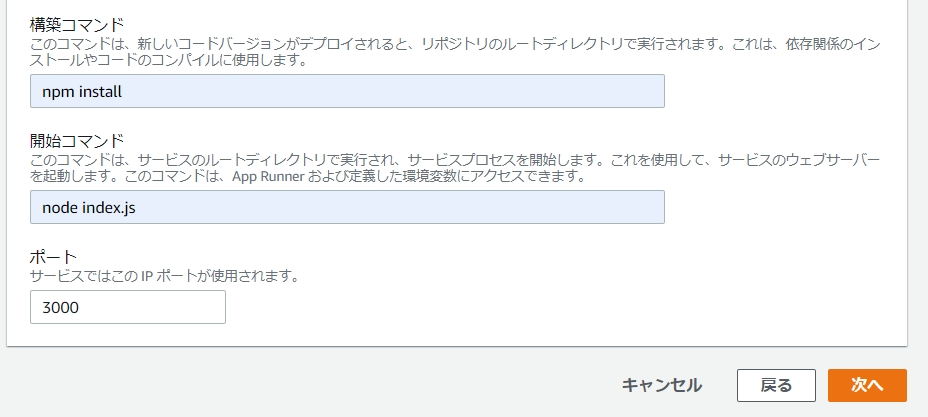
1. レポジトリから[simple-express-app]を選びます。（ブランチはそのままで大丈夫です）



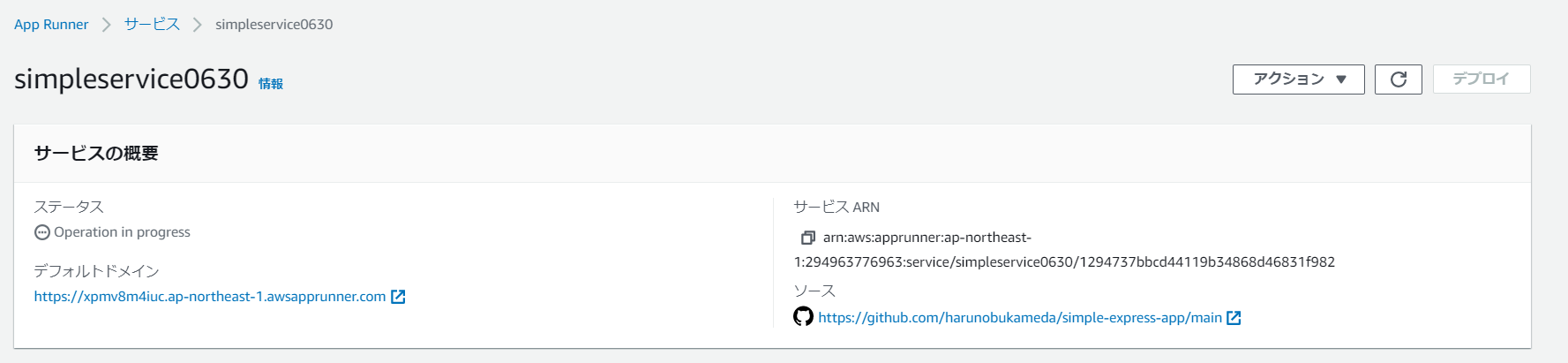
1. [次へ]をおします
2. ランタイムで[Nodejs12]を選びます



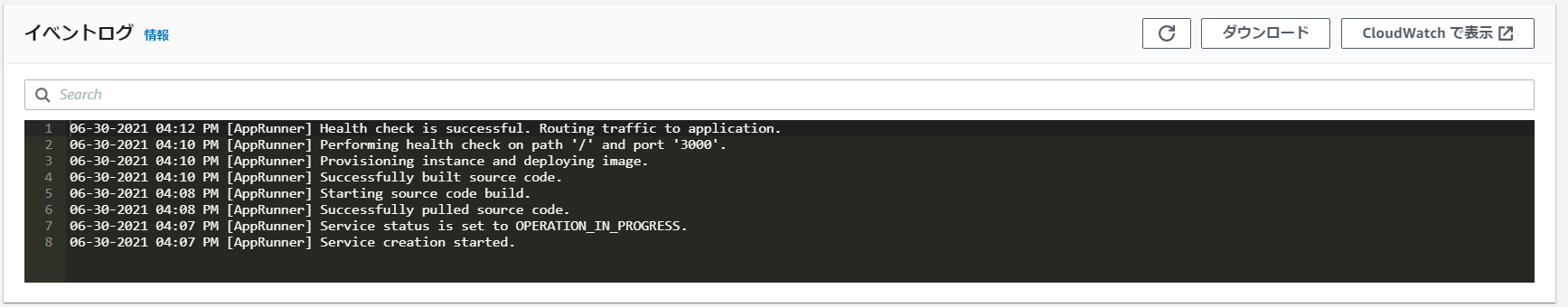
1. 構築コマンドに[npm install]、開始コマンドに[node index.js]、ポートに[3000]と入力して[次へ]をおします



1. サービス名に適当なものを入力し[次へ]をおします
2. 最後に[作成とデプロイ]をおします
3. 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



1. ログのタブをクリックすると作業ログが確認できます



1. 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

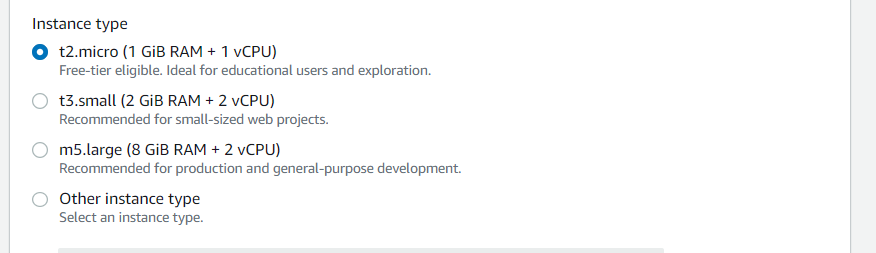
Hello World!

と表示されたら成功です。

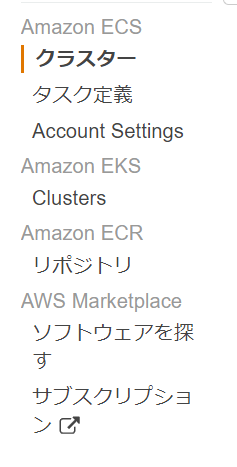
**[ECRとの連携]**

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。

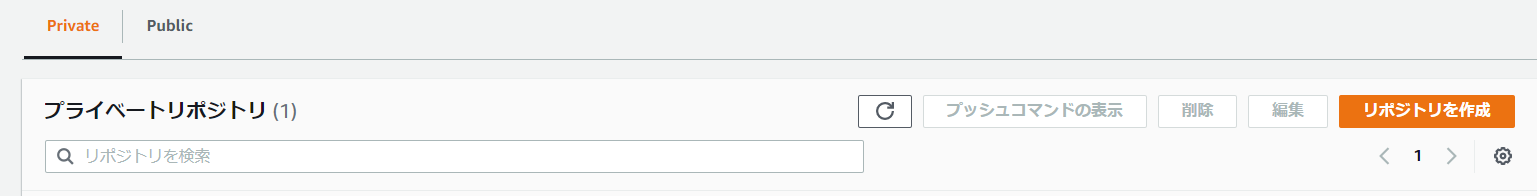
1. Cloud9のマネージメントコンソールにいき、[Create environment]をおします
2. 適当な名前をつけ、[Next Step]をおします
3. [Instance Type]で[t3.small]を選択し、あとはデフォルトのまま[Next Step]をおし、次の画面で[Create environment]をおします。Cloud9が起動中の画面に遷移します。（作業しているリージョンにデフォルトVPCが無い場合、適当なVPCとPublic SubnetをAdvanced Network settingから指定してください



1. 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



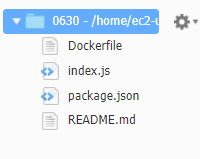
1. [レポジトリを作成]をおします



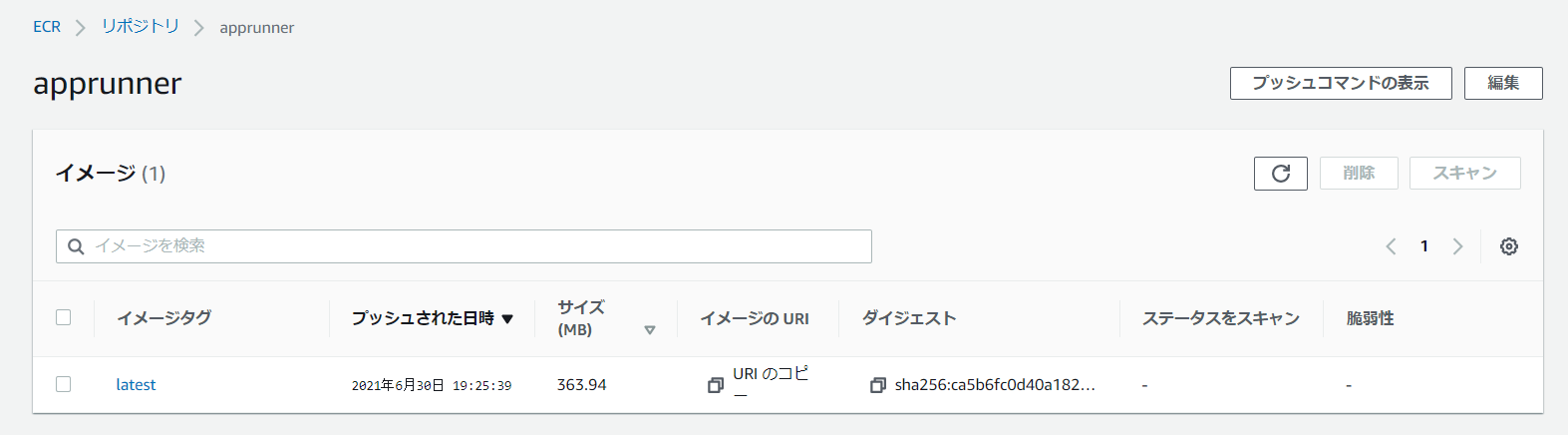
1. レポジトリに適当な名前をつけ、[レポジトリを作成]をおします
2. 作成されたレポジトリを選択し、[プッシュコマンドの表示]をおし、出力されるコマンドを4つともコピーしておきます



1. Cloud9のターミナルが利用できるようになっていますので、[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
2. 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
3. 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
4. 以下のようになればOKです



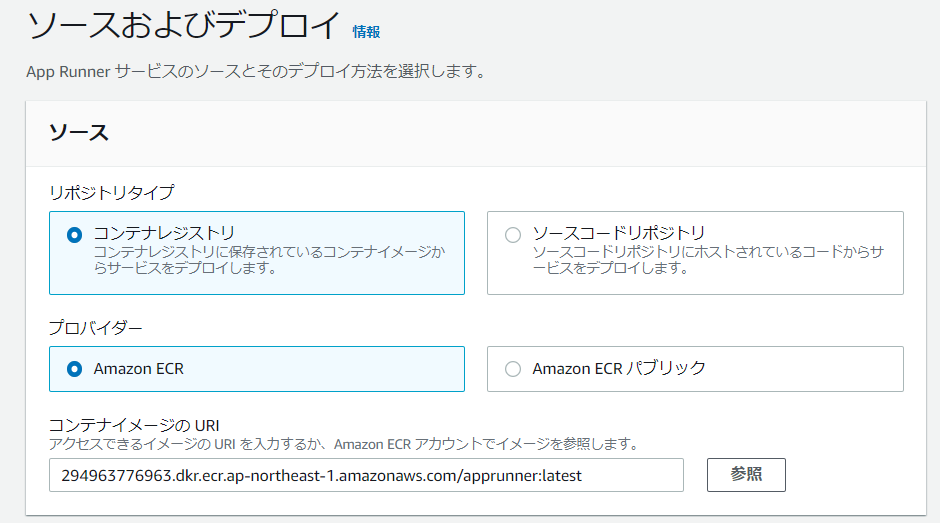
1. 先程ECRの画面からコピペした4つのコマンドを1個づつ順番に実行していきます。特に2個目のコマンドは最後が、スペースとドットで終わっておりミスが発生しやすいので注意してください。
2. これで、Dockerイメージの作成とECRレポジトリへのPushが完了しましたのでECRの画面で確認をします。以下のようにlatestとタグが付いたイメージができていればOKです。



1. [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



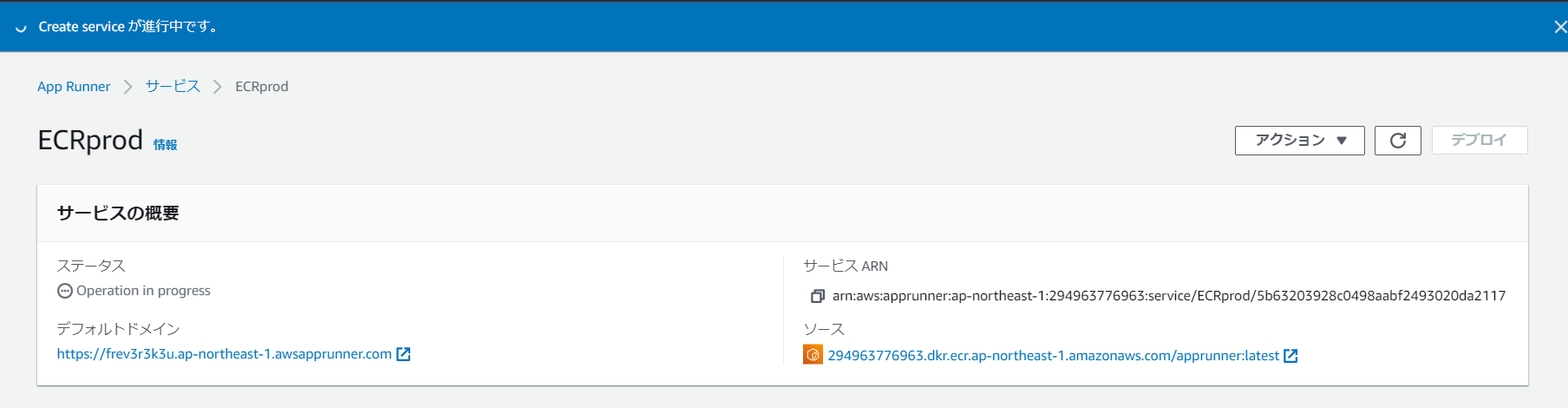
1. App Runnerの画面に戻り、[サービスの作成]をおします
2. 先程と異なり今度は[コンテナレジストリ]を選択し、プロバイダーでAmazon ECRを選び、上記でコピーしたURIをペーストします。この際必ず:latestがURIの最後に付与されていることを確認してください



1. 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。
2. 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



1. 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



1. [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

・IAMロール　（[AppRunnerECRAccessRole](https://console.aws.amazon.com/iam/home#/roles/AppRunnerECRAccessRole)）