AWS Migration Hub Refactor Spaces ワークショップ

2022/01/17

シニアエバンジェリスト 亀田

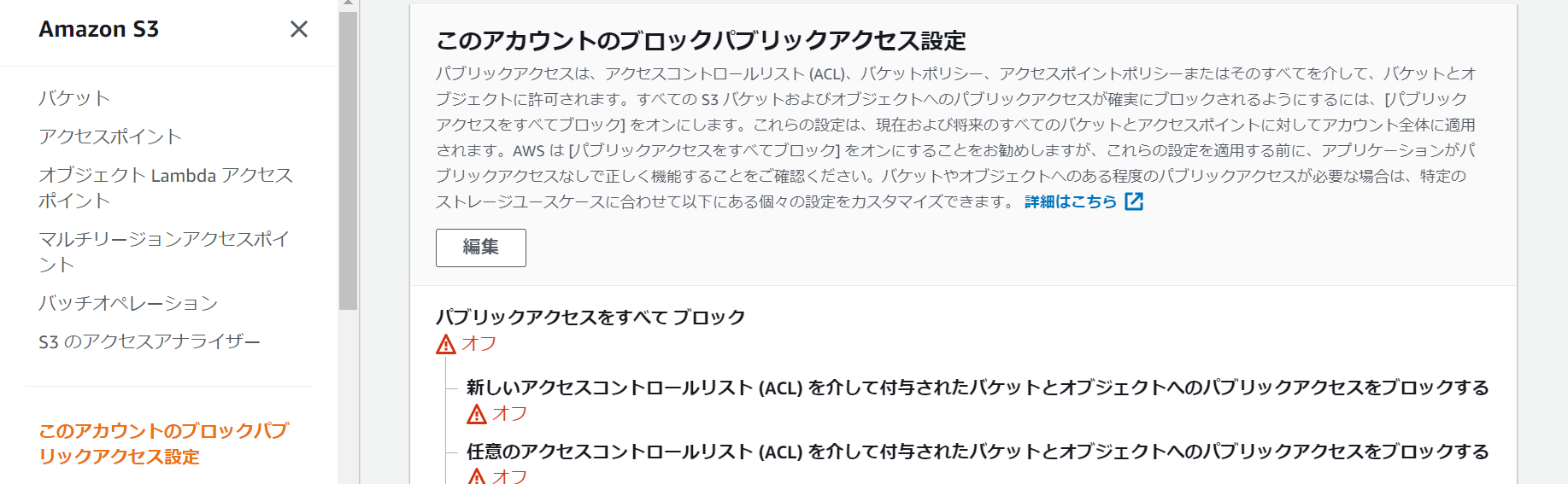
**はじめに：**

Migration Hub Refactor Spaces はAWS上で動作しているRosatiを持つモノリシックなウェブアプリケーションをマイクロサービスへ移行させる際に、ストラングラーフィグパターンによる移行作業を簡素化するために、ネットワーク構成を構築するサービスです。現在Preview中で、HTTPS RestAPIのみをサポートし、また移行先はRestAPIの処理単位に対するLambdaのみをサポートしています。

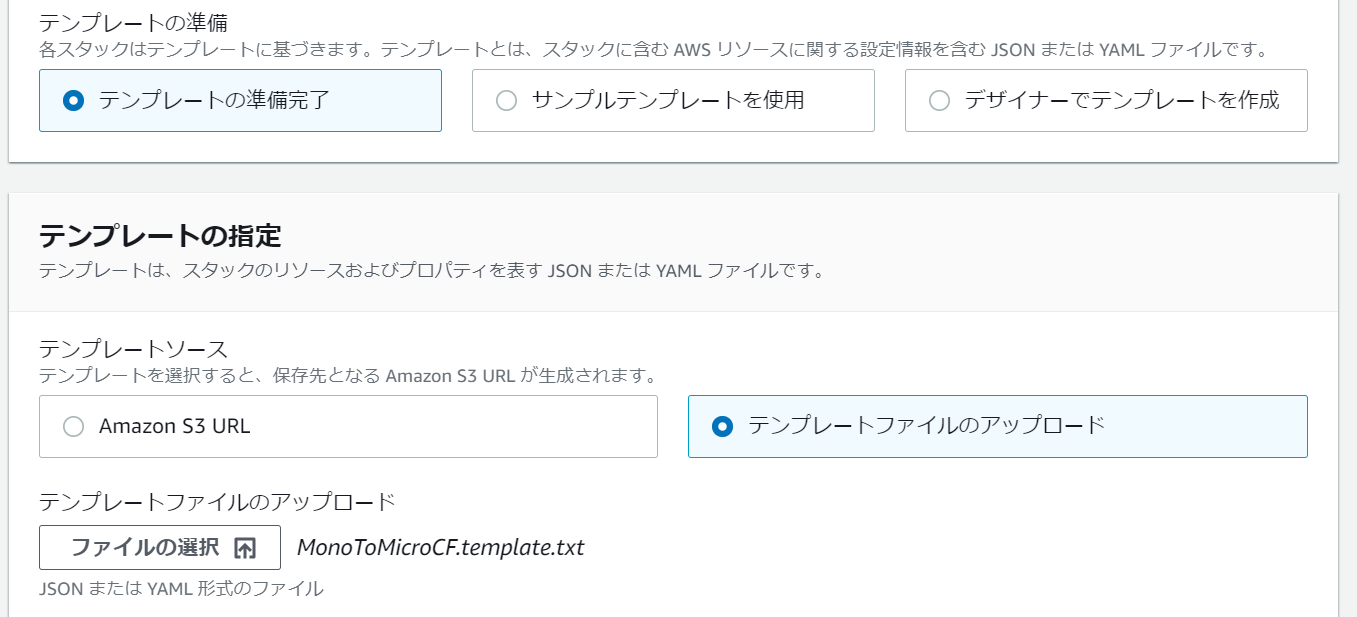
本シナリオは[AWS公式シナリオ](https://catalog.us-east-1.prod.workshops.aws/v2/workshops/f2c0706c-7192-495f-853c-fd3341db265a/en-US/intro)の単純な日本語化バージョンです。サービスのバージョンアップなどで動作しなくなっている場合、英語版を参照ください。その際作成者に一方いただけると幸いです。

モノリシックアプリケーションの構築

以下の手順ではS3を用いたウェブサイトホスティングを行うため、バケットへのパブリックアクセスが必須となります。ブロック設定がオフとなっていることを確認してください。



1. [MonoToMicroCF.template.txt]をGitからダウンロードします
2. CloudFormationの画面から、[スタックの作成]をおします
3. 先程ダウンロードしたテンプレートファイルをアップロードし、[次へ]をおします



1. [MonoToMicro]と名前をつけ、[次へ]をおします
2. 次の画面ではそのまま[次へ]をおし、最後の確認画面で以下にチェックをつけ[スタックの作成]をおします



1. 20分ほど待つと、S3によるHTMLのホスティング、EC2上のJavaアプリケーション、RDSMySQLデータベースンなどが作成されます。
2. 出力タブのPublicDNSの値をコピーしておきます。リソースタブから作成されたVPCのIDもコピーします。
3. S3バケットで[[monotomicro-uibucket-](https://s3.console.aws.amazon.com/s3/buckets/monotomicro-uibucket-1wygu7gcrr7m6?region=ap-northeast-1)xxxx]を特定しクリックし、[プロパティ]タブをクリックします
4. 画面の一番下のURLをブラウザで開きます



1. 適当な名前でSignupをし、そのパスワードでLoginを行います。その後適当な商品をカートに入れたりカートから削除したり操作してみてください。

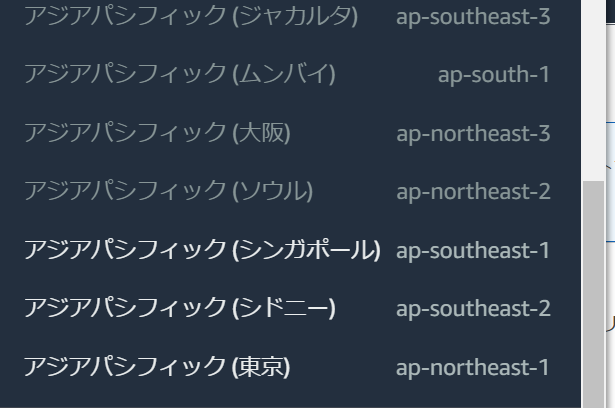
これでモノリスアプリケーションの作成は完了です。Chromeをお使いの方は、デベロッパーツールを開くと、EC2インスタンスへ通信していることがわかります。



**Refactor Spacesの起動：**

このハンズオンでは、先に構築したモノリシックアプリケーションから、カートへの商品追加、商品の削除、カートに一時的に保存されている商品一覧の表示、3つの機能**([baseUrl]/unicorns/basket)**のみをマイクロサービスアーキテクチャ（Lambdaへ移行します）

1. マネージメントコンソールでMigration Hubにアクセスします。MigrationHubにはデフォルトリージョンという設定が存在し、AWSアカウント単位でどこか単一のリージョンがメインのダッシュボードを提供するため、別のリージョンでアクセスしても、デフォルトリージョンのダッシュボードが表示されます。**ただしRefactor Spacesは別のリージョンでも作業が可能です。**
2. 左ペインから[Refactor Spaces]をクリックした後、作業を行いたいリージョンに変更します



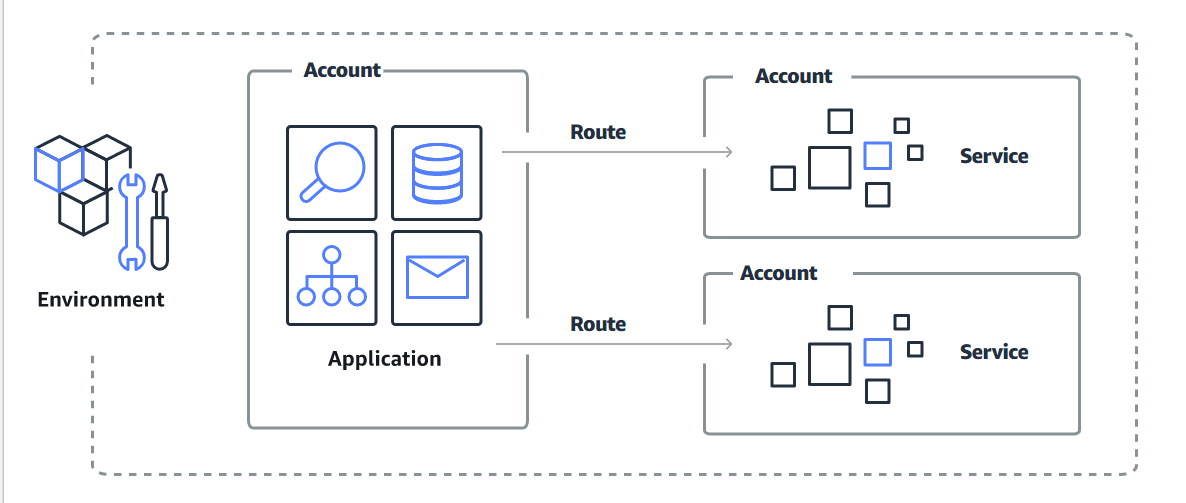
1. [環境]をクリックし、[環境の作成]をクリックします



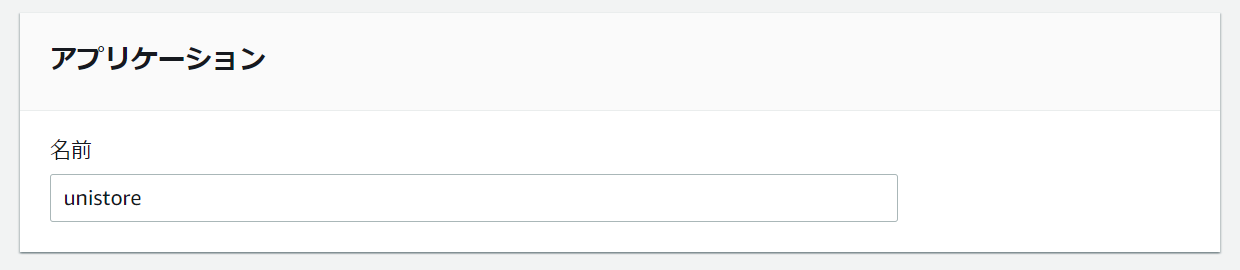
1. 名前を” unistore-dev”と入力して、[次へ]をおします



画面に概要図が表示されますが、右下がモノリシックなアプリケーション環境（VPC）、右上がマイクロサービス化されたLambda環境です。そして左側が、Refactor Spaces が作成するネットワーク環境とAPI Gatewayになります。

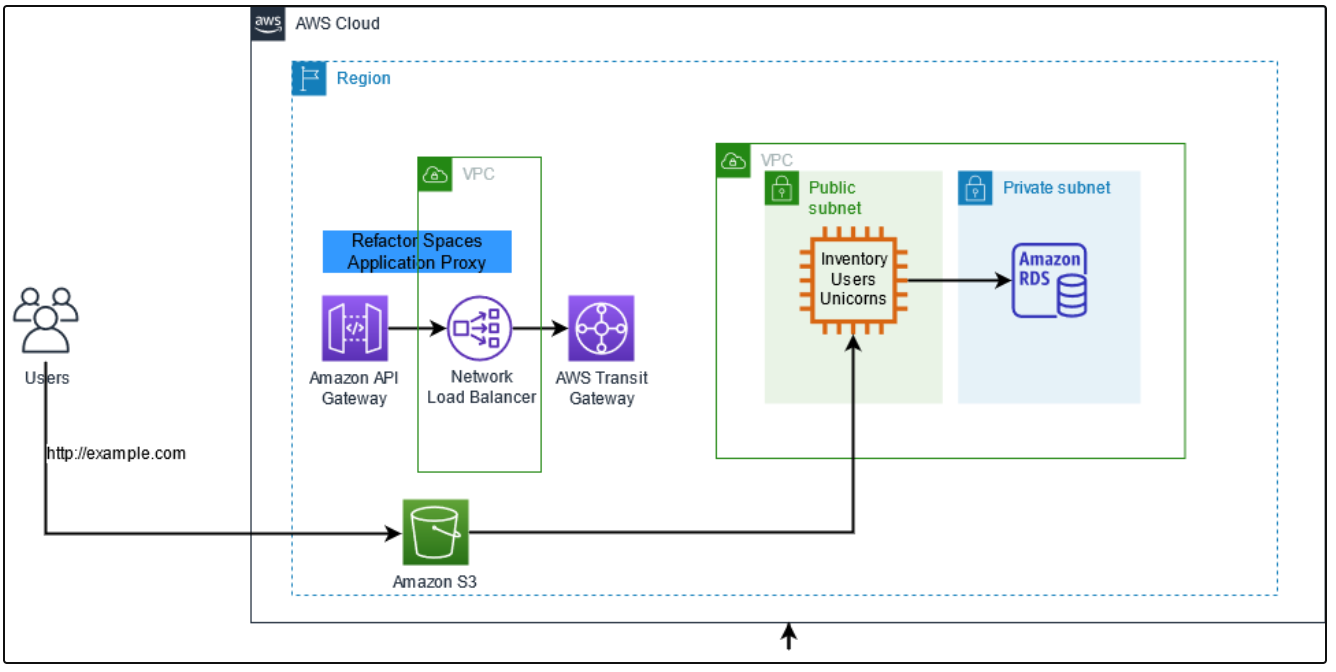


1. アプリケーションに”unistore”と入力します

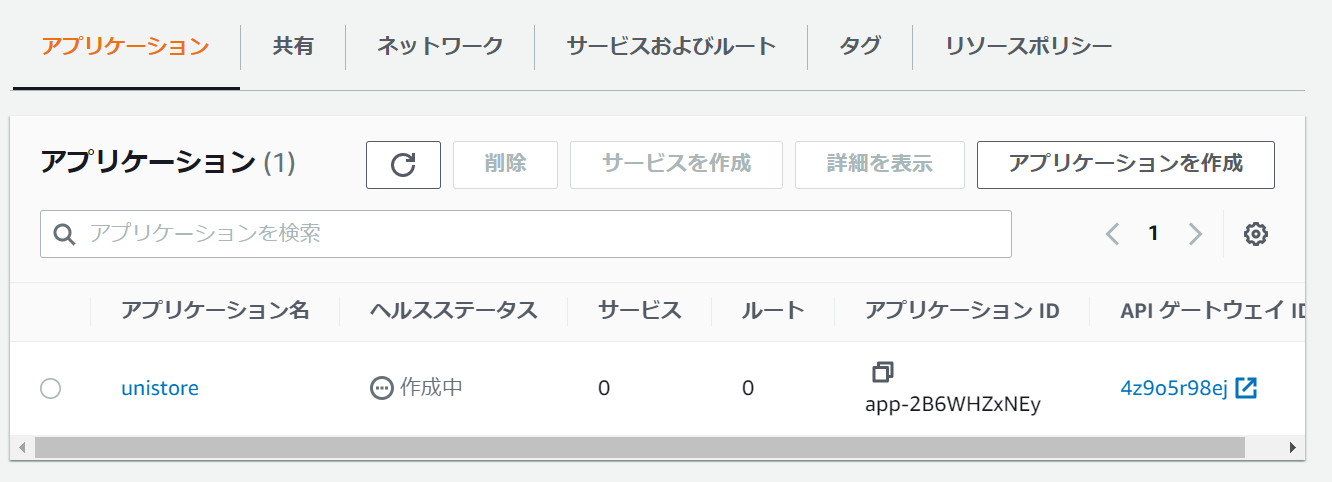


1. 先程コピーしたVPCのIDを以下のダイアログで選択し、[次へ]をおします。このStepで指定したVPCに対して、Refactor Spacesがプロキシ用ネットワークとAPI Gatewayを構築します。





1. 次の画面はデフォルトのまま[次へ]をおします
2. 次の画面で[環境の作成]をおします。5分程度で画面が遷移します。
3. 遷移先の画面でしばらくまちます。



ヘルスステータスが[正常]になれば設定が完了です。

**サービスとルートの設定：**

いままでの手順でプロキシネットワークが出来上がりましたが、モノリシックアプリケーションをサービスとして登録し、そのルート設定を行うことで、ユーザーからのアクセス（リクエスト）を作成されたネットワークが処理できるようになります。

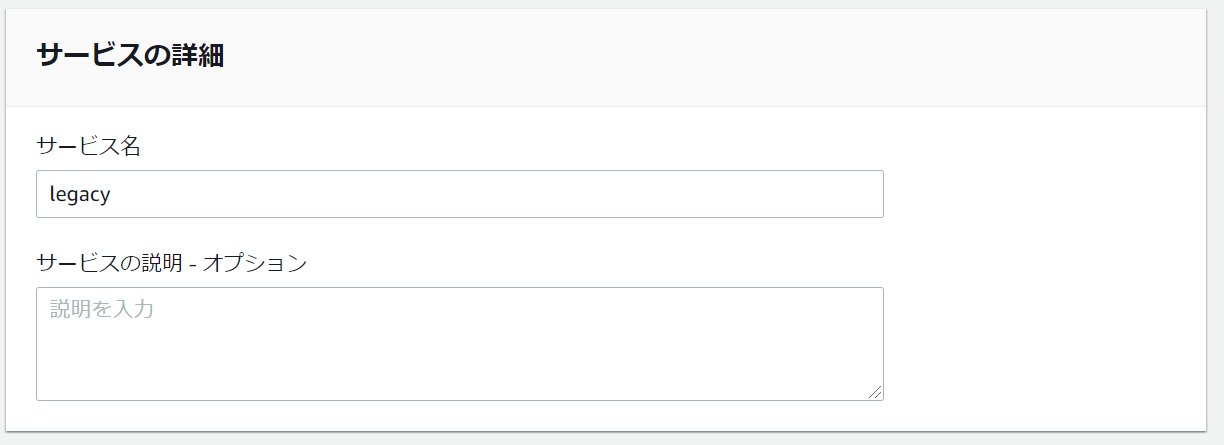
1. 左ペインから[サービスの作成]をクリックします



1. 今までの手順で作成された、環境とアプリケーションを選択します



1. サービス名に[legacy]と入力します



1. VPCにJavaアプリケーションがホスティングされているEC2のVPCIDを指定し、エンドポイントにEC2のエンドポイントを入力します。ヘルスチェックエンドポイントには、以下を入力します。

[http://ec2-XXX-XXX-XXX-XXX.compute-1.amazonaws.com/](http://ec2-xxx-xxx-xxx-xxx.compute-1.amazonaws.com/)actuator/health

注意点：ユーザーがS3上のHTMLを読み込んだのちアクセスする先が現在EC２エンドぽいと担っていますが。そのアクセス先をRefactor Spacesのネットワークに変更するための設定です。したがって、ここでの設定はS3のURLではなく、EC2のエンドポイントURLを設定してください。

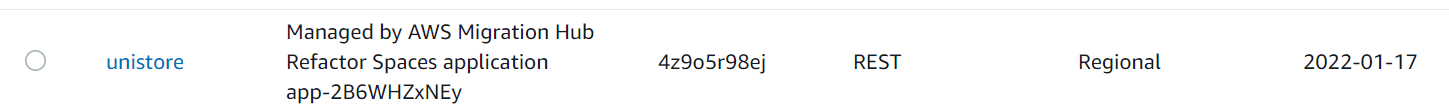
1. “このサービスをアプリケーション (unistore) のデフォルトルートとして設定します。”にチェックを付けます。これにより、一旦すべてのRefactor SpacesへのリクエストがEC2エンドポイントにルーティングされることとなります。



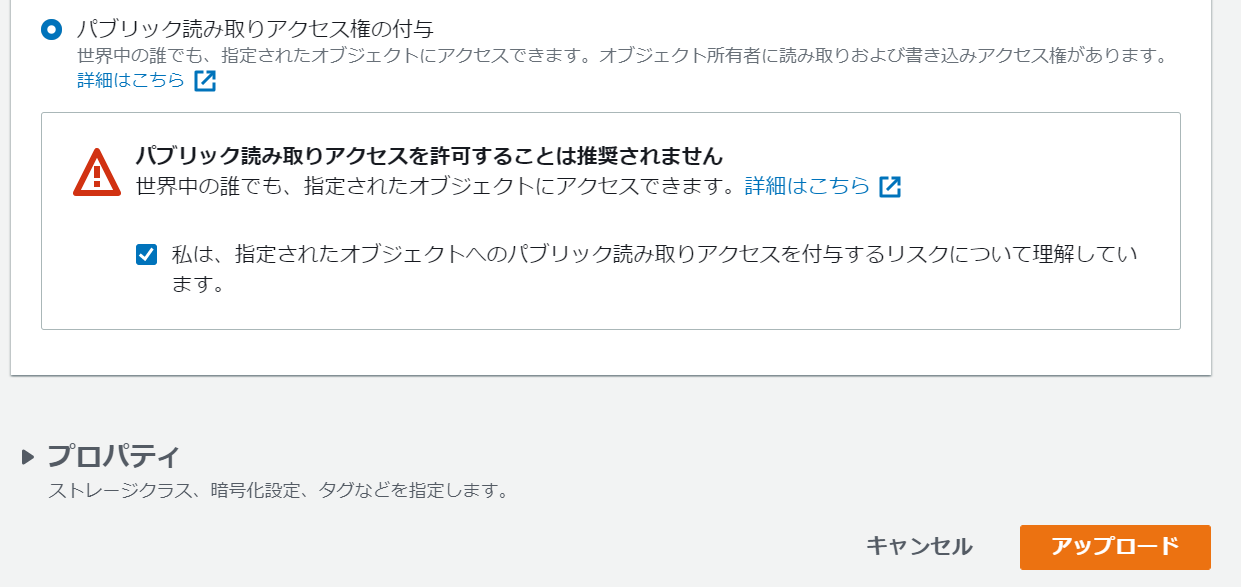
1. [サービスの作成]をおします
2. しばらく待つと以下のようにサービス、ルートともに以下のように表示されれば成功です



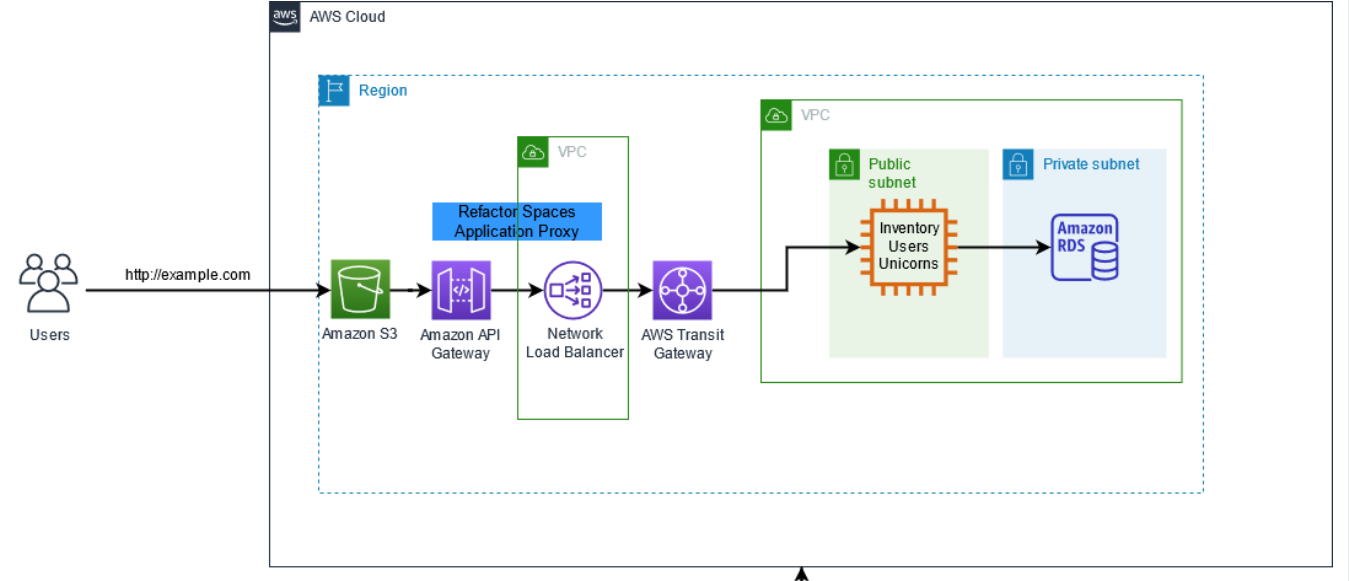
1. API Gatewayのマネージメントコンソールに移動すると、APIが1つ作成されていることがわかります。これがすべてのリクエストを一旦モノリシックなアプリケーションへルーティングするプロキシのエンドポイント（EC2エンドポイントに代わるもの）になります。



1. [ステージ]→[Prod]とクリックしてＵＲＬをコピーしておきます
2. Ｓ3バケットの” **MonoToMicro-uibucket-xxxxx”から”** **config.json”をダウンロードしエディタで開きます**
3. URLをEC2エンドポイントからAPI Gatewayのものに差し替え保存し、Ｓ3へ上書きアップロードします。その際、必ず以下の設定を行いパブリックアクセスを可能としてください。

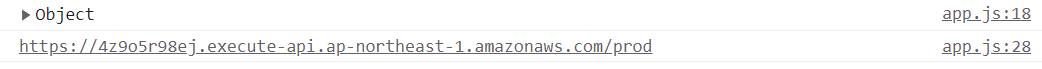


1. 以下のような構成に変更されました。



API GatewayがEC2のアプリケーション（HTTPS REST API）に対するプロキシとなっていることに注目してください。

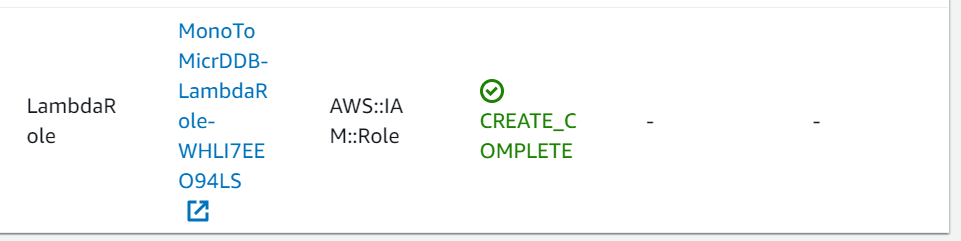
1. 先程テストしたS3上のHTMLにアクセスし、再度正しく動作するか確認してください。Chromeをお使いの方はデベロッパーツールで以下のようにアクセス先がEC2ではなくなっていることを確認してください。



**マイクロサービス環境の構築：**

ではここから、カート処理を担うマイクロサービス環境を作成し、順次リクエストをRefactor Spacesが管理するAPI Gateway経由でこちらにルーティングする設定を行います。

1. “MonoToMicroDDBCF.yaml”をGitからダウンロードします
2. 先程と同様の手順でCFnにてスタックを作成します。スタックの名前は” MonoToMicrDDB”としてください。最大5分程度待ちます。ステータスが”CREATE\_COMPLETE“になったら、念のためマネージメントコンソールでDynamoDBの”unishop”テーブルができていることを確認します。
3. CFnのリソースタブで、作成されたIAM Roleの名前をコピーしておきます



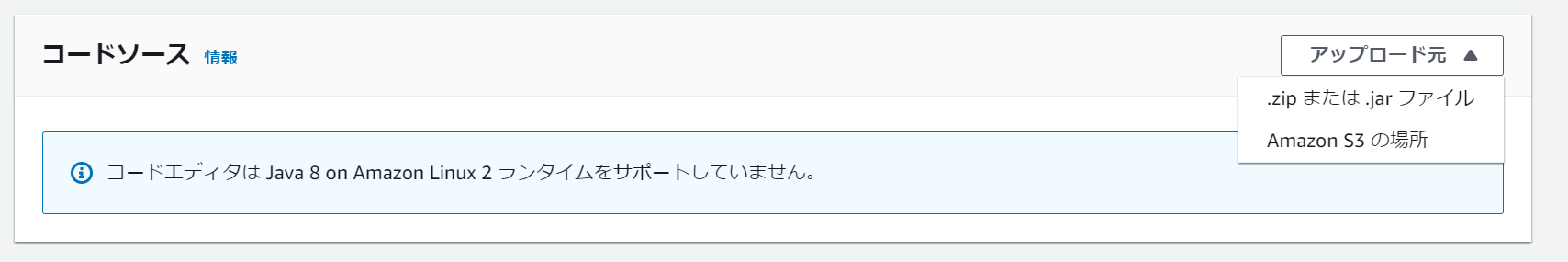
1. Lambdaマネージメントコンソールに移動し、[関数の作成]ボタンをおします
2. 関数名とランタイムを以下のように設定します



1. 先程作成されたLambda用ロールを設定します。（Lambda関数がDynamoDBへのアクセス権限を持つためのものです）

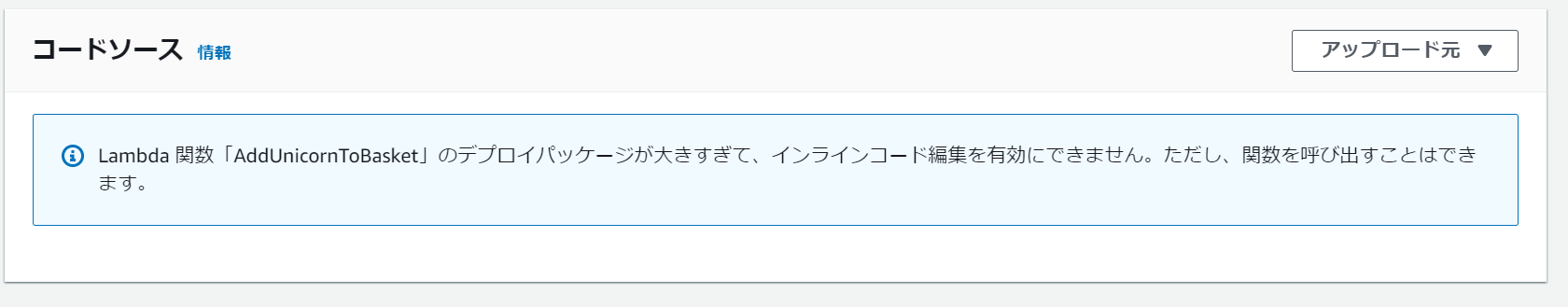


1. [関数の作成]ボタンをおします
2. S3バケット” MonoToMicro-assetbucket-xxxxx”からMonoToMicroLambda-0.0.1.jarをダウンロードし適当な個所に保存します
3. Lambda関数のコードソースから先程ダウンロードしたjarをアップロードします





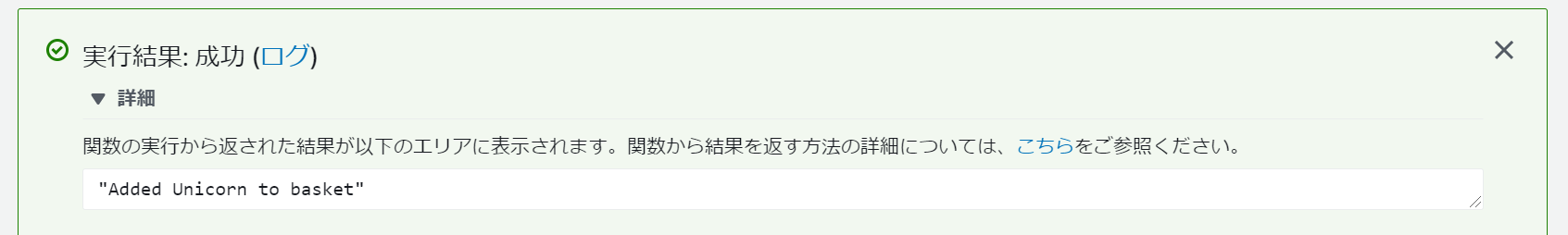
1. 以下の注意表記は正常ですので問題ありません。



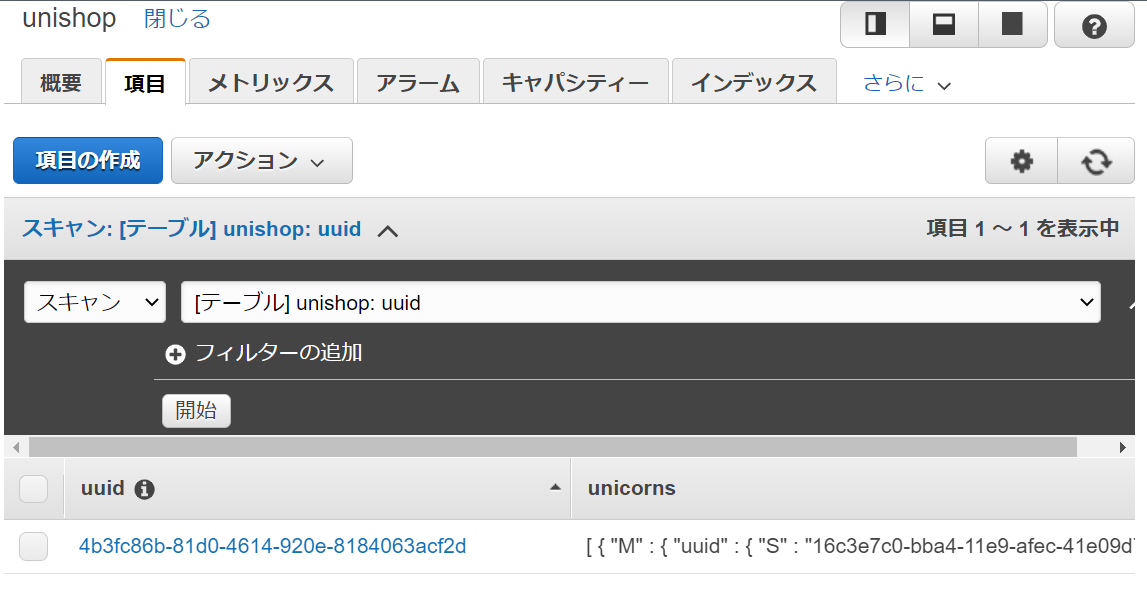
1. ランタイムの[編集]ボタンをおして、以下の値をハンドラとして設定し、[保存]をおします



1. [テスト]タブをクリックして、名前に”test”と入力します。
2. “testcommand.txt”を開いて、[AddUnicornToBasket]の中身を以下にコピペし、[テスト]ボタンをおします
3. 以下の通り成功と表示されれば設定は完了です



1. DynamoDBのテーブルの[項目]タブを見ると値が1個追加されていることがわかります。



1. 同様の手順でもう2つLambda関数を作成してください。

名前：RemoveUnicornFromBasket

ハンドラ：com.monoToMicro.Lambda.UnicornBasketImpl::removeUnicornFromBasket

名前：GetUnicornsBasket

ハンドラ：com.monoToMicro.Lambda.UnicornBasketImpl::getUnicornsBasket

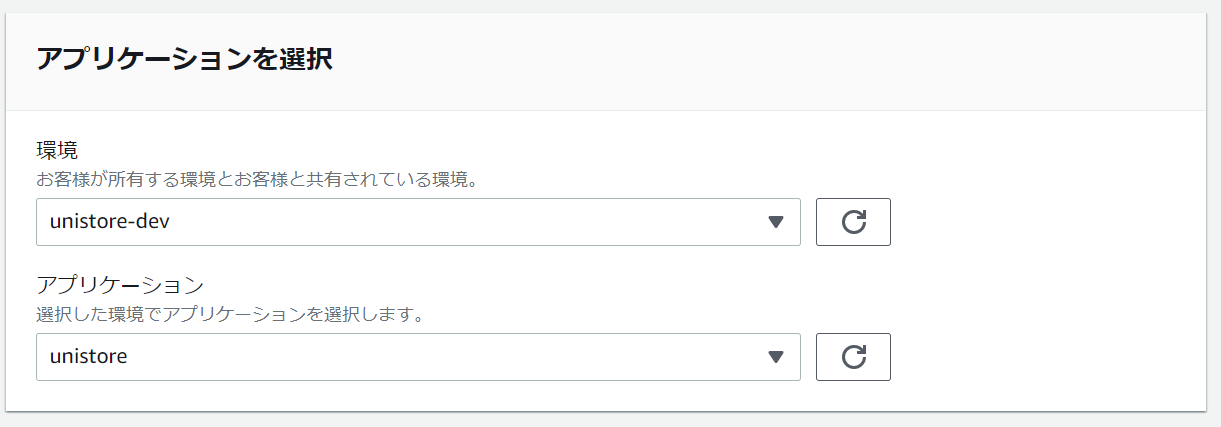
テストパラメータ（共通）

“testcommand.txt”のRemoveUnicornFromBasket / GetUnicornsBasket

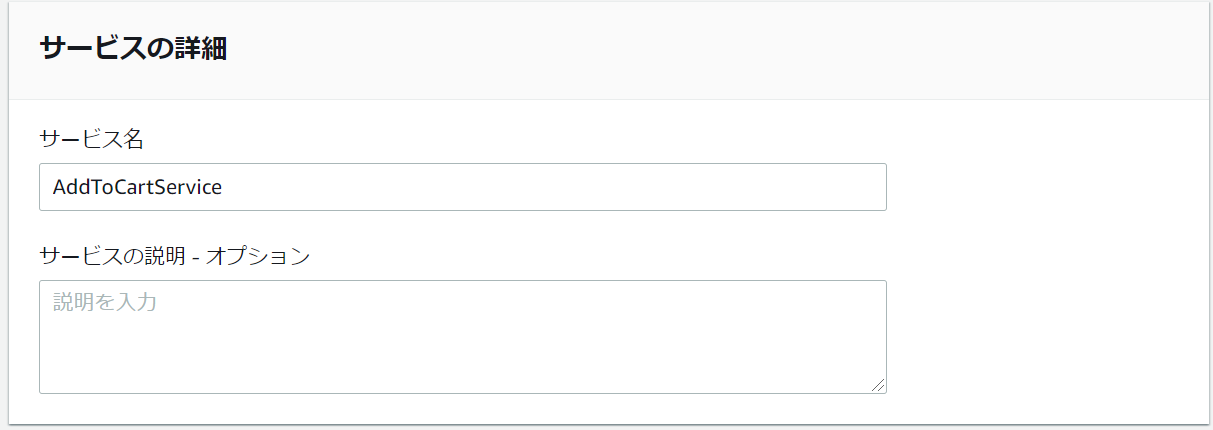
**ストラングラーフィグパターンの実装：**

Lambda関数ができましたので、APIGatewayの向き先を変更していきます。現在Refactor Spacesではデフォルトとして全てが旧モノリシック環境へのルーティングがなされていますが、関数単位（HTTP REST API）でLambdaへルーティングを変更していきます

1. Refactor Spacesの左ペインで”サービスを作成”をクリックします
2. 先程と同じ値を[環境][アプリケーション]に指定します



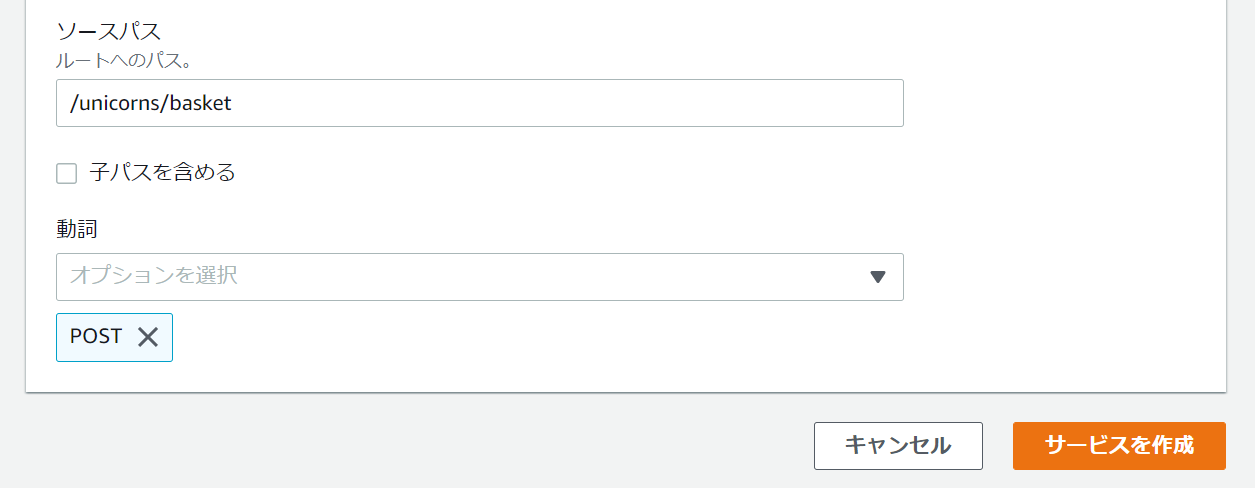
1. [AddToCartService]をサービス名に登録します



1. 先程と異なり、サービスエンドポイントの設定は[VPC]ではなく[Lambda]を指定し、[AddUnicornToBasket]を指定します



1. ソースパスに[/unicorns/basket]を入力し、**”子パスを含める”のチェックを外してください**
2. 動詞のドロップダウンから[POST]を選択し、[サービスを作成]をおします



1. 同様に以下の内容でサービスをもう1つ作成します

名前: RemoveCartService

エンドポイント: Lambda

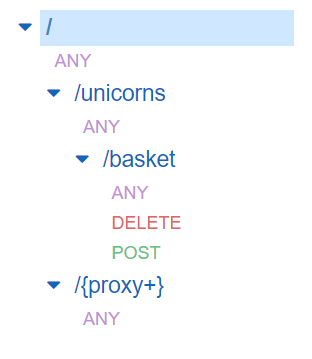
Lambda : RemoveUnicornFromBasket

ソースパス: /unicorns/basket

**子パスを含める: オフ**

動詞：DELETE

1. ブラウザの別タブでAPI Gatewayのマネージメントコンソールを開きます
2. APIの“unistore”をクリックします
3. 先程作成したDELETEとPOSTが作成されていることがわかります



1. 左ペインの[モデル]をクリックし[作成]をおします



1. モデル名に[UnicornBasket]と入力し、コンテンツタイプに[application/json]と入力します
2. Testcommand.txtのAPI Gateway Get Schemaの値をコピーして、[モデルのスキーマ]にコピーして[モデルの作成]をおします
3. Refactor Spacesに戻り、先程と同様に3回目の[サービスの作成]を行います。以下の値を入力してください。

名前: GetCartService

エンドポイント: Lambda

Lambda : GetUnicornBasket

ソースパス: /unicorns/basket

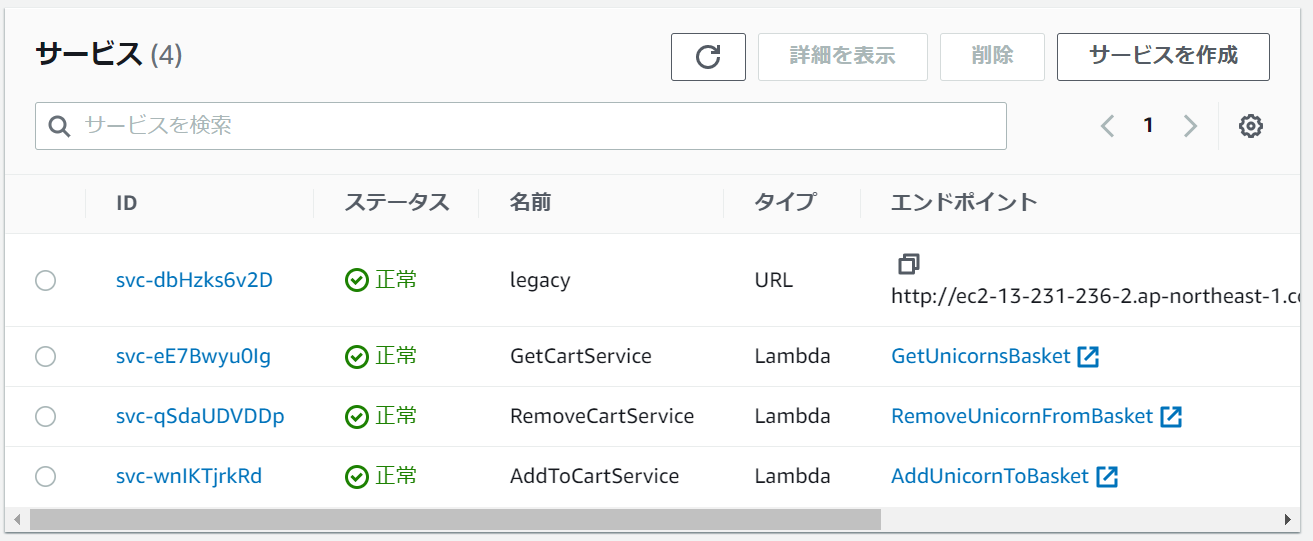
**子パスを含める: オン（今までと異なるので要注意）**

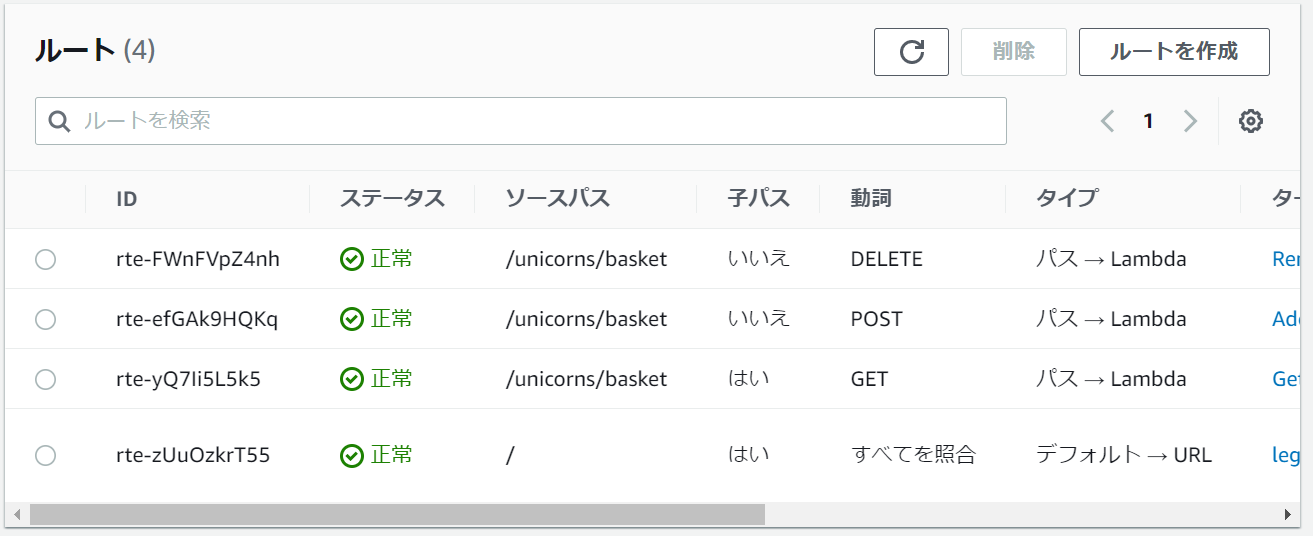
動詞: GET

1. マネージメントコンソールAPI Gatewayのリソースにアクセスし、以下の状態になっているか確認をしてください。



ずれがある場合、サービスの作成にミスがありますので、サービスを一度デフォルト以外消してやり直してください。

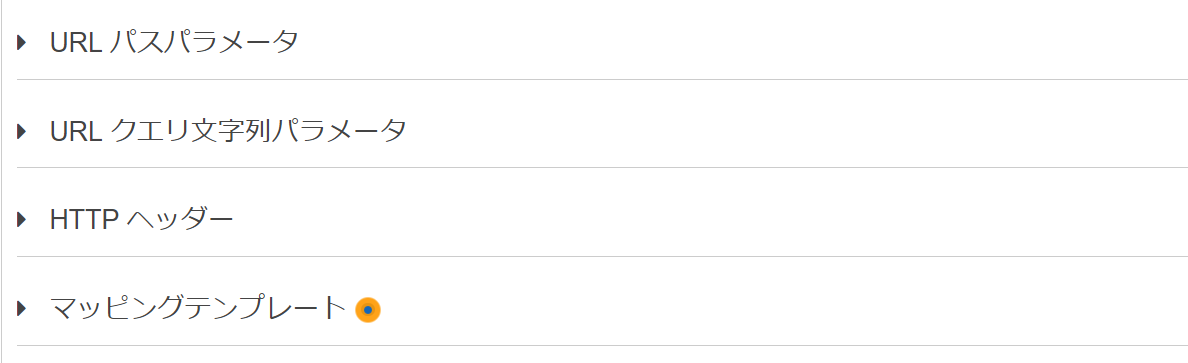




1. /unicorns/basket/{proxy+}配下のGETをクリックしてください
2. [統合リクエスト]をクリックします
3. [Lambda プロキシ統合の使用]のチェックをはずします



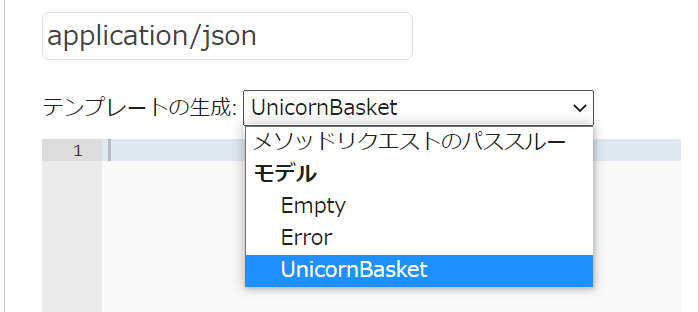
1. [/unicorns/basket]のPOSTとDELETEで同様の作業を行います
2. 再度[/unicorns/basket/{proxy+}/GET]の[統合リクエスト]画面に戻り、画面一番下の[マッピングテンプレート]を開きます



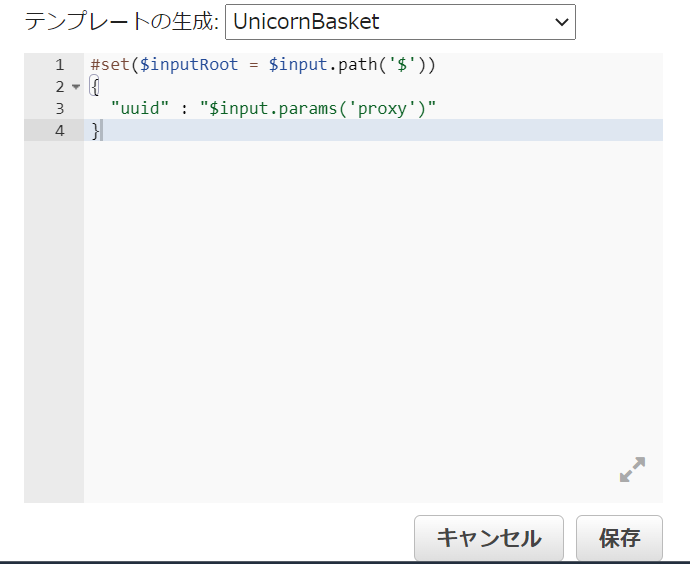
1. [テンプレートが定義されていない場合 (推奨)]を選択し、[マッピングテンプレートの追加]のプラスボタンをおします
2. [application/json]を入力し、チェックをおします



1. [テンプレートの生成]ドロップダウンから”UnicornBasket”を選びます

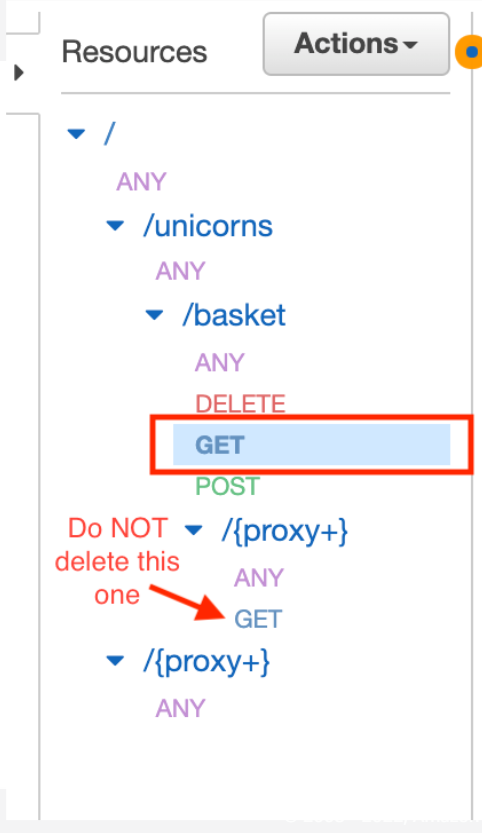


1. testcommand.txtの[UnicornBasket model]の値をコピーし、[保存]をおします

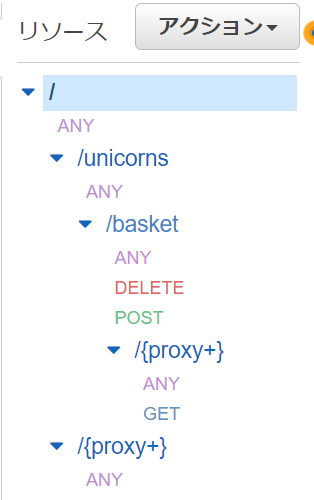


[保存]をおすと、ドロップダウンから[UnicornBasket]の値が空欄に代わりますが正常です。

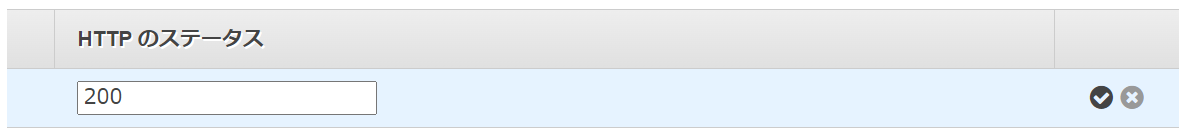
1. /unicorns/basket配下のGETを選んで、アクションからメソッドの削除を選びます

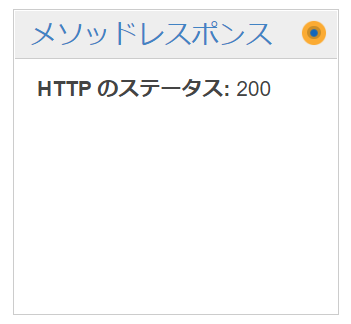


1. 以下の状態になります

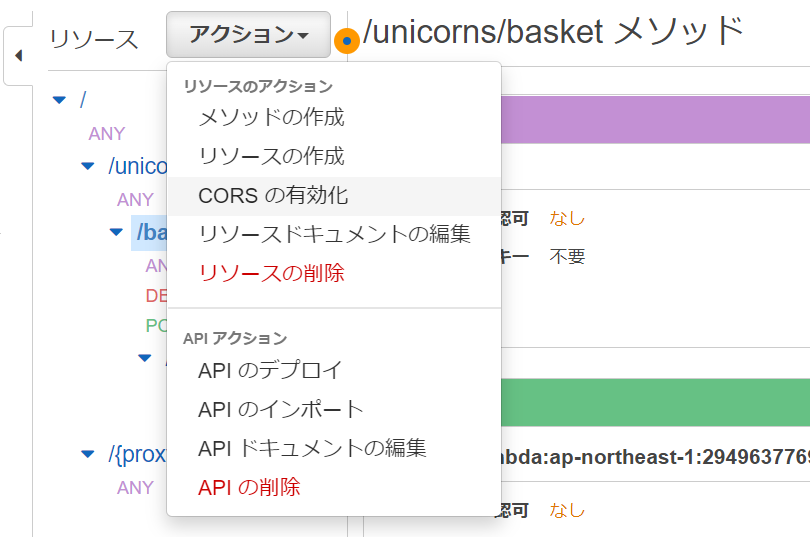


1. 再度GETをクリックし、[メソッドレスポンス]をクリックします
2. [レスポンスの追加]をおして“200”と入力しチェックをおします



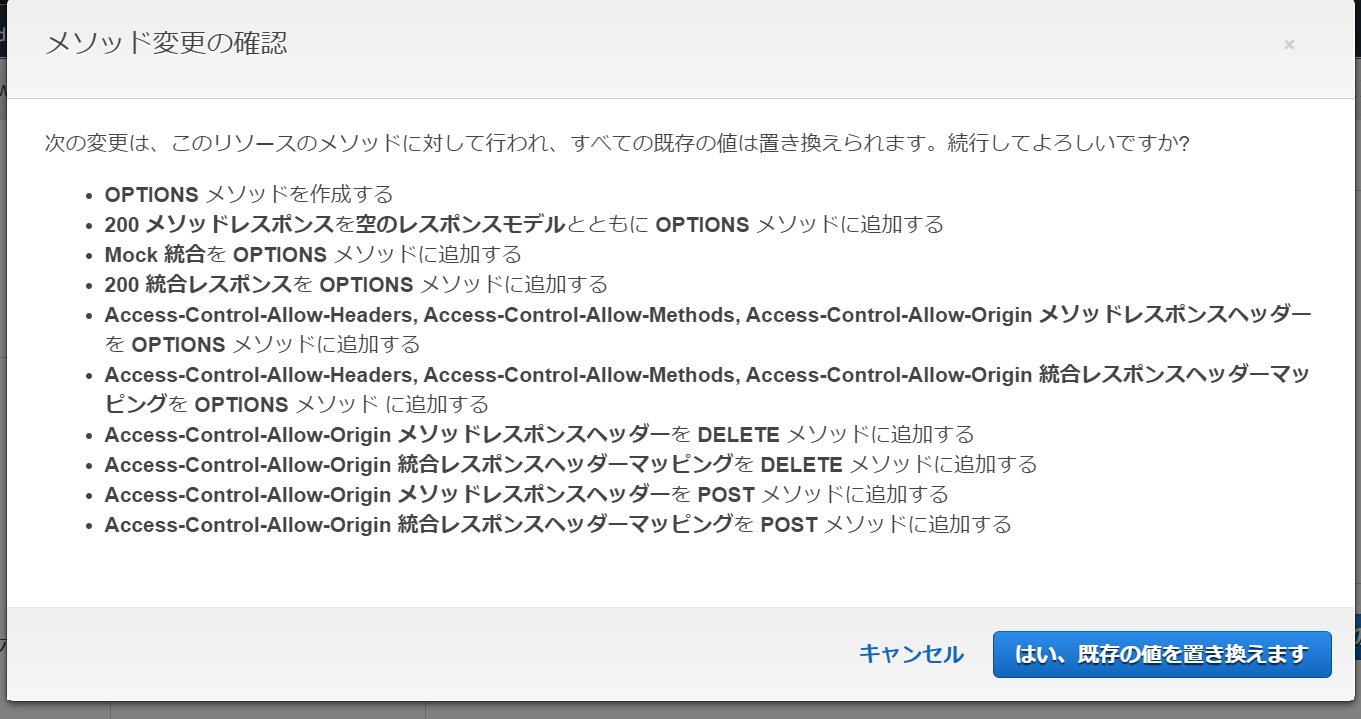


1. POSTとDELETEで同じようにメソッドレスポンスを登録します。
2. [basket]を設定して、アクションからCORSの有効化を選択します





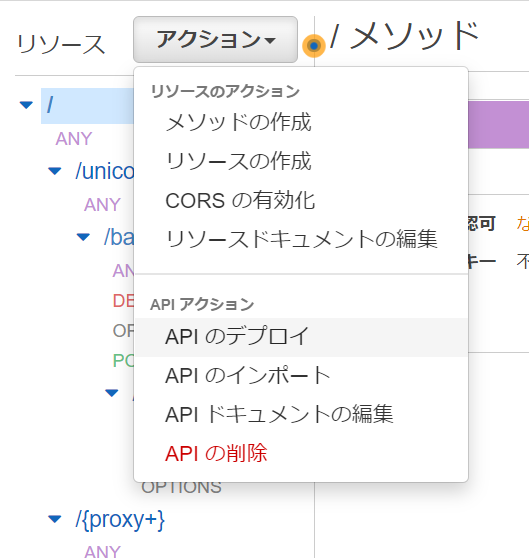
1. [はい、既存の値を置き換えます]をおします。



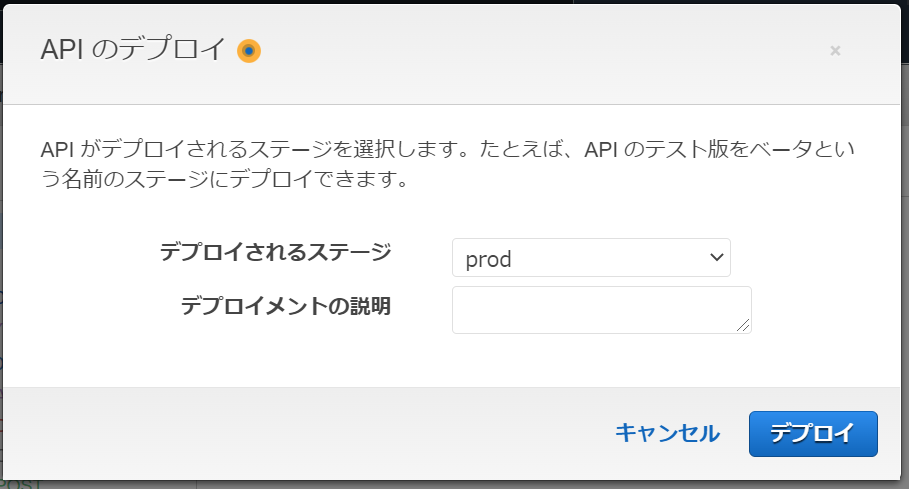
ここまでの設定が正しければ全てにチェックが付きます



1. [/unicorns/basket/{proxy+}]を選択し同様にCORSを選択します
2. ルート( / )を選択し[APIのデプロイ]を選びます



1. [prod]を選択し[デプロイ]をおします



1. 以上で設定が完了です。カートへの追加、削除、一覧表示のみがLambdaへ移行されています。（このシナリオでは、Lambda関数は大きいJavaで作成されており、初回起動に少し時間がかかることに注意してください。

おつかれさました！

削除は以下を行って下さい

Lambda関数３つ

Refactor Spacesのルート

Refactor Spacesのサービス

Refactor Spacesの環境

CFn２つ