メシウス株式会社

AWS Lambda と DioDocs で Excel や PDF ファイルを出力する

2024 年 4 月 8 日

※本資料は、弊社ブログに投稿された記事「**AWS Lambda と DioDocs で Excel や PDF ファイルを出力する（1）～（3）**」の連載記事をベースに資料化した内容となります。下記が記事の原文となります。

AWS Lambda と DioDocs で Excel や PDF ファイルを出力する（1）
AWS Lambda と DioDocs で Excel や PDF ファイルを出力する（2）
AWS Lambda と DioDocs で Excel や PDF ファイルを出力する（3）

目次

AWS Lambda と DioDocs で Excel や PDF ファイルを出力する（1）

本記事では、AWS Lambda で「DioDocs（ディオドック）」を使用した C#（.NET 8）の Lambda 関数アプリケーションを作成し、Excel や PDF ファイルを出力する方法について紹介します。

AWS Lambda とは

AWS Lambda は Amazon Web Services で提供されている、各種イベントをトリガーに処理を実行するサーバーレスなアプリケーションを作成できるクラウドサービスです。

AWS Lambda は.NET 8 をサポートしており、C#で .NET 8 ベースの Lambda 関数を作成できます。今回は AWS Toolkit for Visual Studio を使用して Visual Studio 2022 で Lambda 関数を作成し、AWS へデプロイして確認してみます。

実装する内容

今回実装する内容は非常にシンプルです。AWS Lambda アプリケーションで Amazon API Gateway から HTTP リクエストを受け取る Lambda 関数を作成します。この関数の実行時に DioDocs を使用して Excel と PDF ファイルを作成し、HTTP リクエストのクエリパラメータで受け取った文字列を追加します。その後、作成した Excel と PDF ファイルを関数から Amazon API Gateway に渡して HTTP レスポンスで直接ローカルへ出力する、といった内容です。

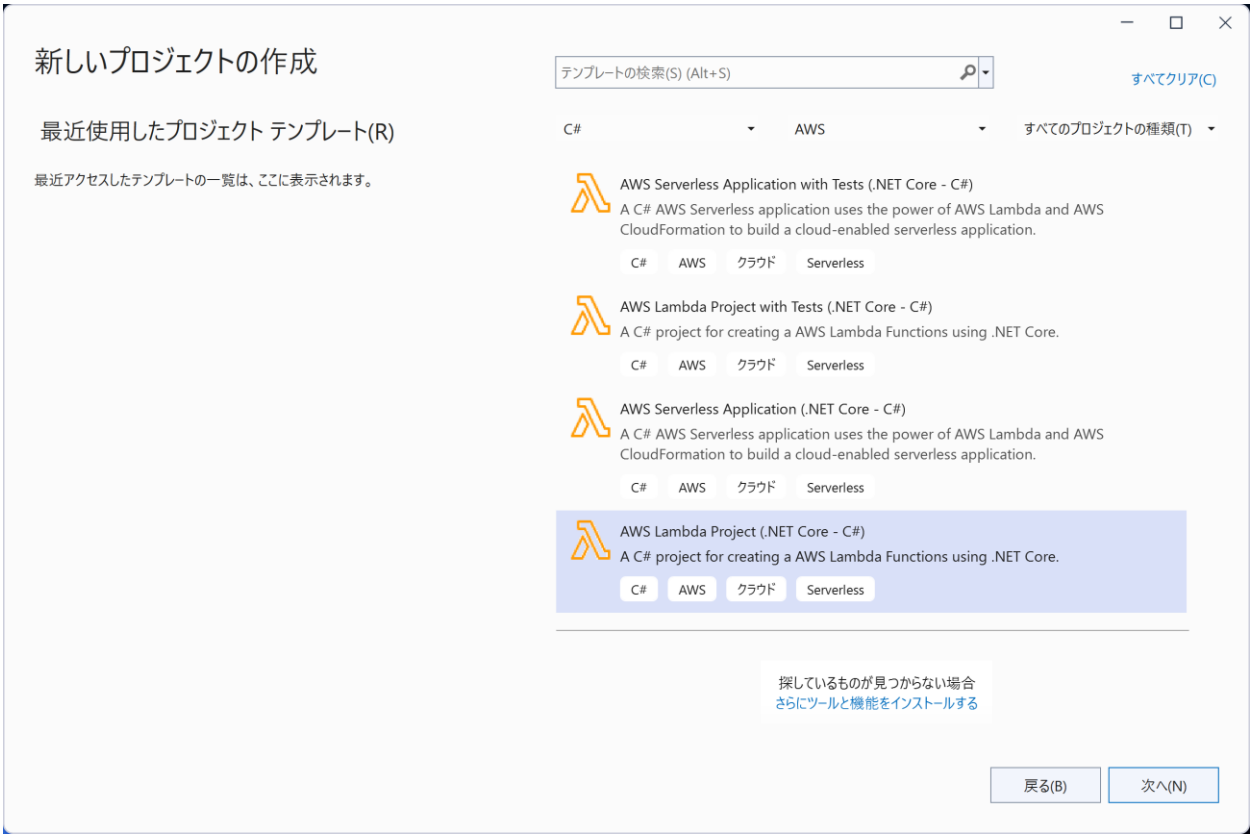
AWS Toolkit for Visual Studio のセットアップ

Visual Studio 2022 への AWS Toolkit for Visual Studio のインストールと AWS の認証情報の設定は以下を参考にして準備しておきます。
AWS Toolkit for Visual Studio のインストールとセットアップ
AWS IAM 認証情報

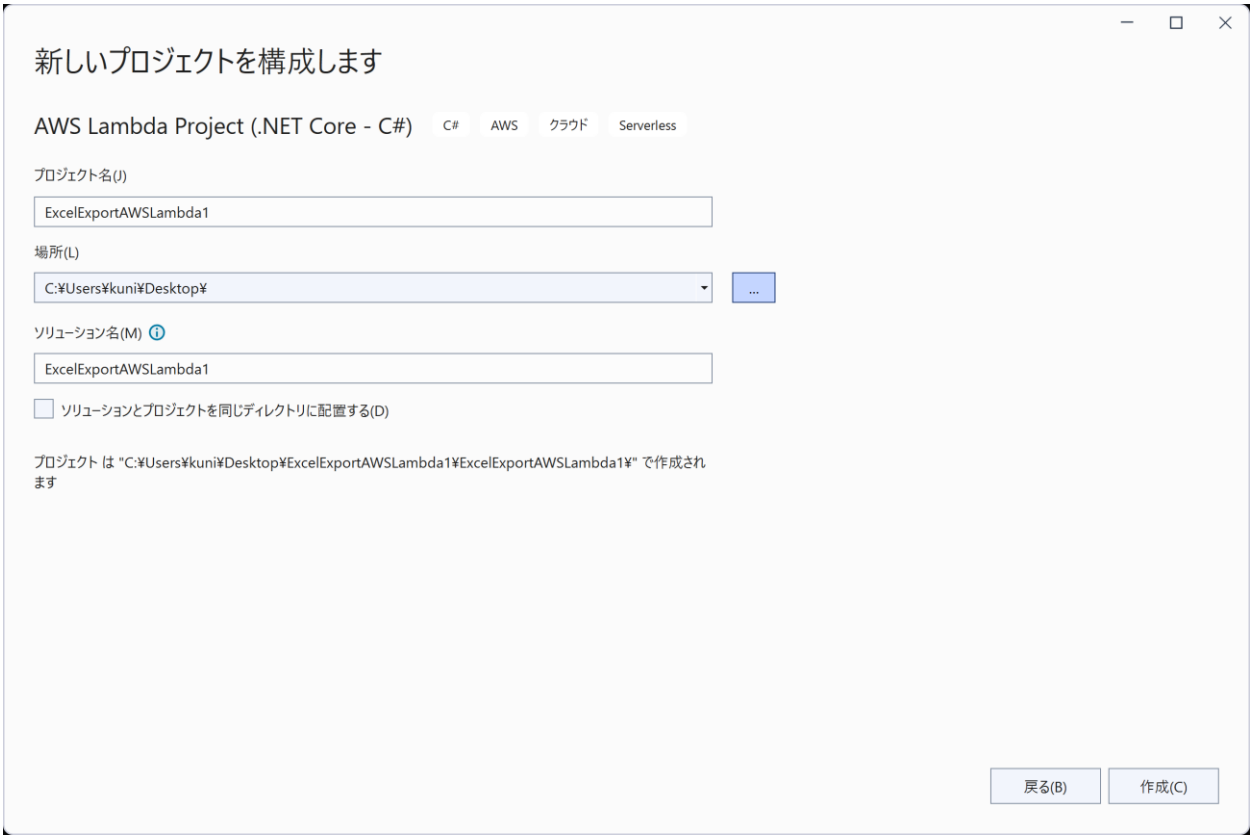
AWS Lambda アプリケーションを作成

以下のドキュメントを参考に AWS Lambda アプリケーションを作成していきます。
基本 AWS Lambda プロジェクト

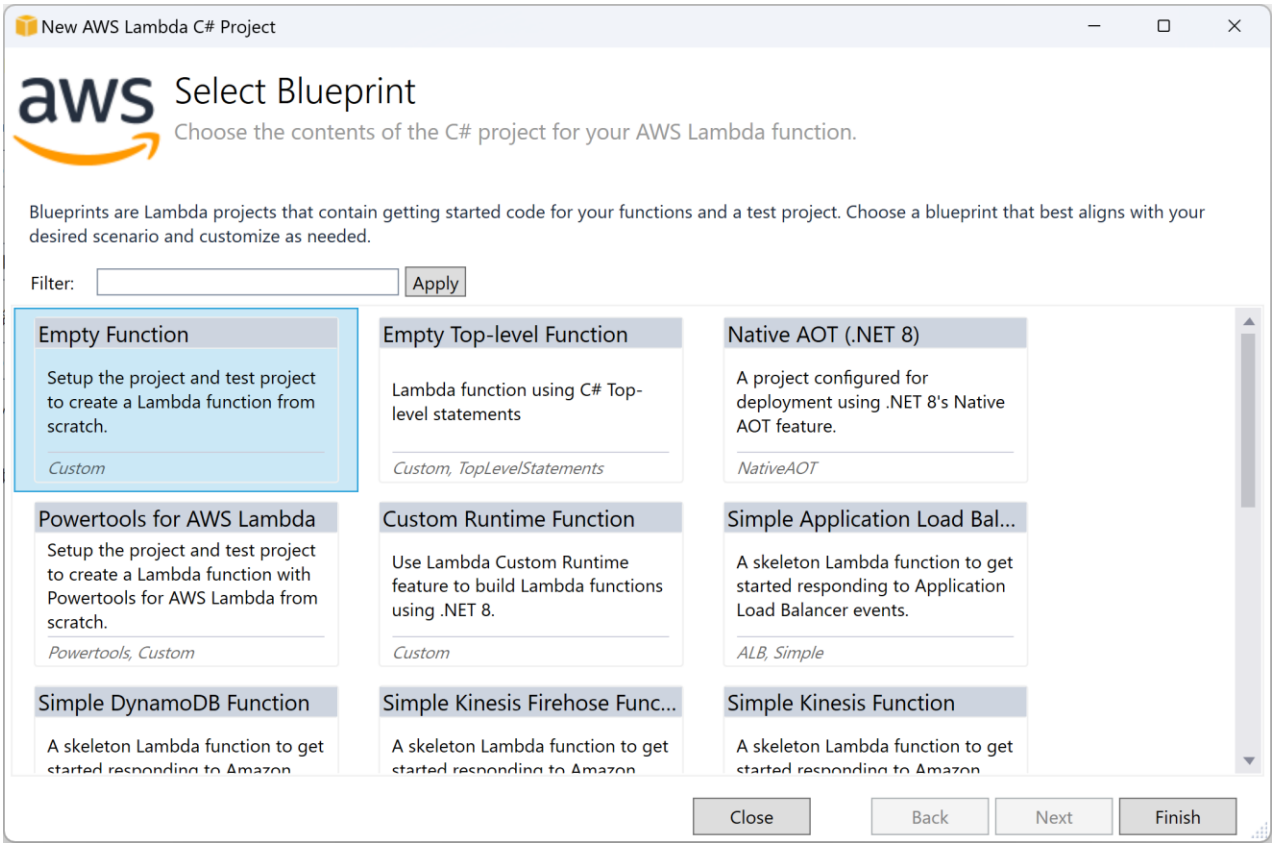
Visual Studio 2022 でプロジェクトテンプレート「AWS Lambda Project (.NET Core – C#)」を選択して［次へ］をクリックします。



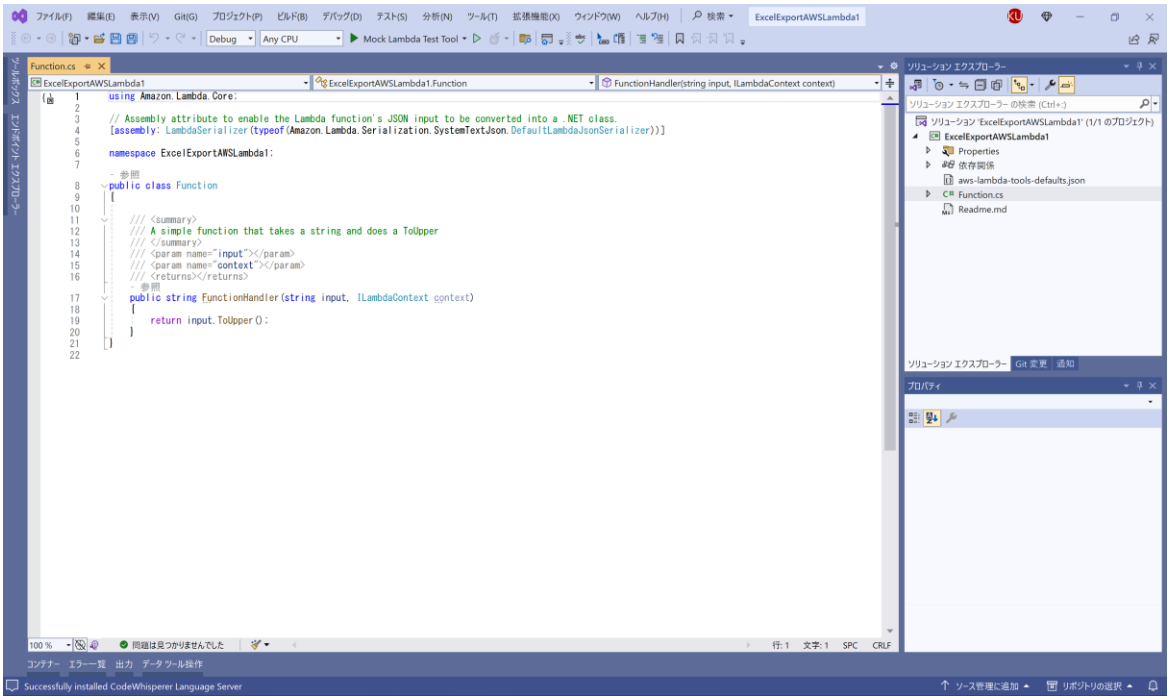
プロジェクト名に「ExcelExportAWSLambda1」を入力して［作成］をクリックします。



AWS Lambda Project のテンプレートを選択します。「Empty Function」を選択して［Finish］をクリックします。

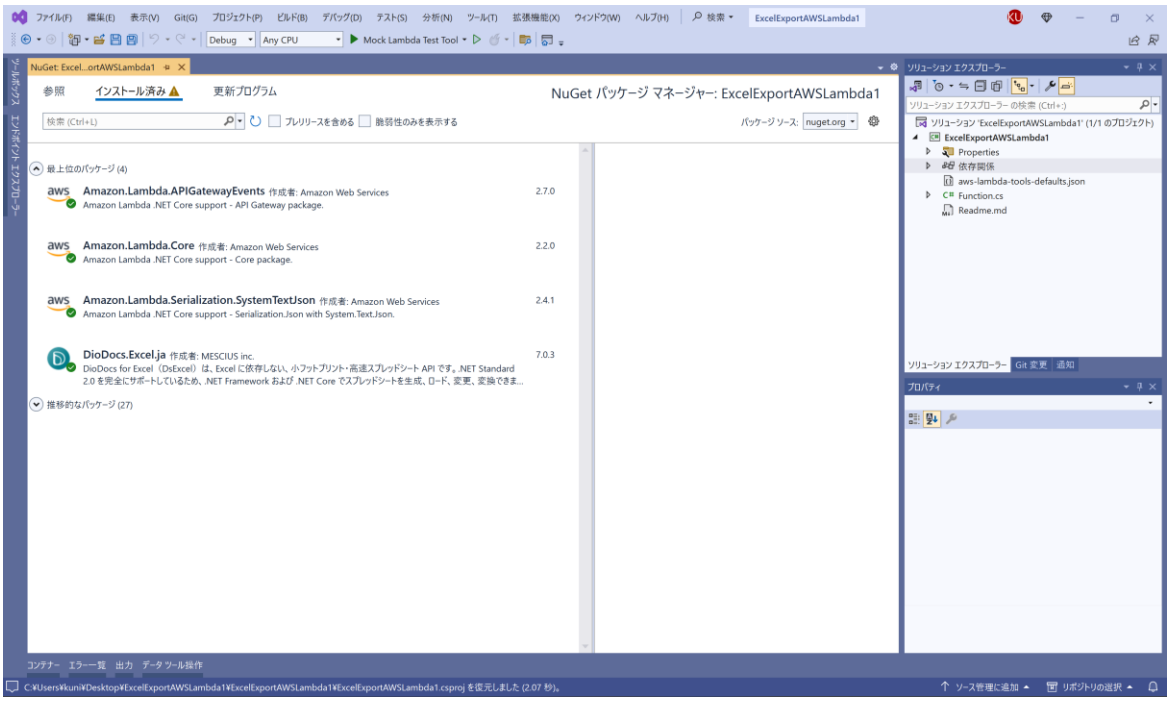


「ExcelExportAWSLambda1」プロジェクトが作成されます。



NuGet パッケージの追加

Visual Studio の「NuGet パッケージ マネージャー」から Amazon API Gateway のイベントを処理するためのパッケージ「`Amazon.Lambda.APIGatewayEvents`」と DioDocs for Excel のパッケージ「`DioDocs.Excel.ja`」をインストールします。



Amazon API Gateway を使うコードを追加

Lambda 関数が Amazon API Gateway から HTTP リクエストを受け取り、Lambda 関数から API Gateway へ HTTP レスポンスを返すために、以下のように `FunctionHandler` の引数と戻り値に `APIGatewayProxyRequest` と `APIGatewayProxyResponse` を設定します。

```
public APIGatewayProxyResponse FunctionHandler(APIGatewayProxyRequest input, ILambdaContext context)
```

DioDocs for Excel を使うコードを追加

DioDocs for Excel で Excel ファイルを作成するコードを追加して `FunctionHandler` を以下のように更新します。

```
public APIGatewayProxyResponse FunctionHandler(APIGatewayProxyRequest input, ILambdaContext context)
{
    APIGatewayProxyResponse response;

    string ? queryString;
    input.QueryStringParameters.TryGetValue("name", out queryString);

    string Message = string.IsNullOrEmpty(queryString)
        ? "Hello, World!!"
        : $"Hello, {queryString}!!";

    //Workbook.SetLicenseKey("製品版またはトライアル版のライセンスキーを設定");

    Workbook workbook = new Workbook();
```

```
    workbook.Worksheets[0].Range["A1"].Value = Message;

    var base64String = "";

    using (var ms = new MemoryStream())
    {
        workbook.Save(ms, SaveFileFormat.Xlsx);
        base64String = Convert.ToBase64String(ms.ToArray());
    }

    response = new APIGatewayProxyResponse
    {
        StatusCode = (int)HttpStatusCode.OK,
        Body = base64String,
        IsBase64Encoded = true,
        Headers = new Dictionary<string, string> {
            {"Content-Type", "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"},
            {"Content-Disposition", "attachment; filename=Result.xlsx"},
        }
    };

    return response;
}
```

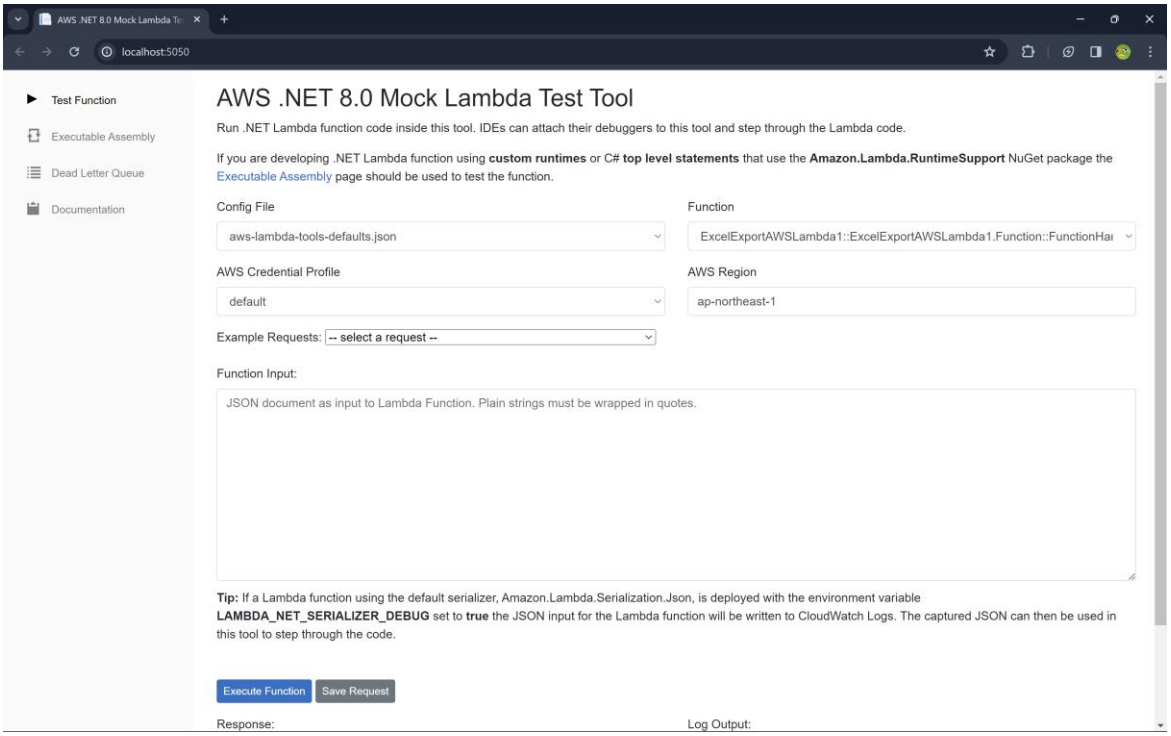
DioDocs for Excel で作成した Excel ファイルを `MemoryStream` に保存し、これを一旦 base64 エンコードしています。これを文字列 `base64String` として `APIGatewayProxyResponse` の `Body` に設定して Amazon API Gateway に渡すようにしています。

作成した Excel ファイルを base64 にエンコードする理由ですが、Lambda 関数と Amazon API Gateway を連携させる「AWS Lambda プロキシ統合」を利用する際の決まり事になっています。

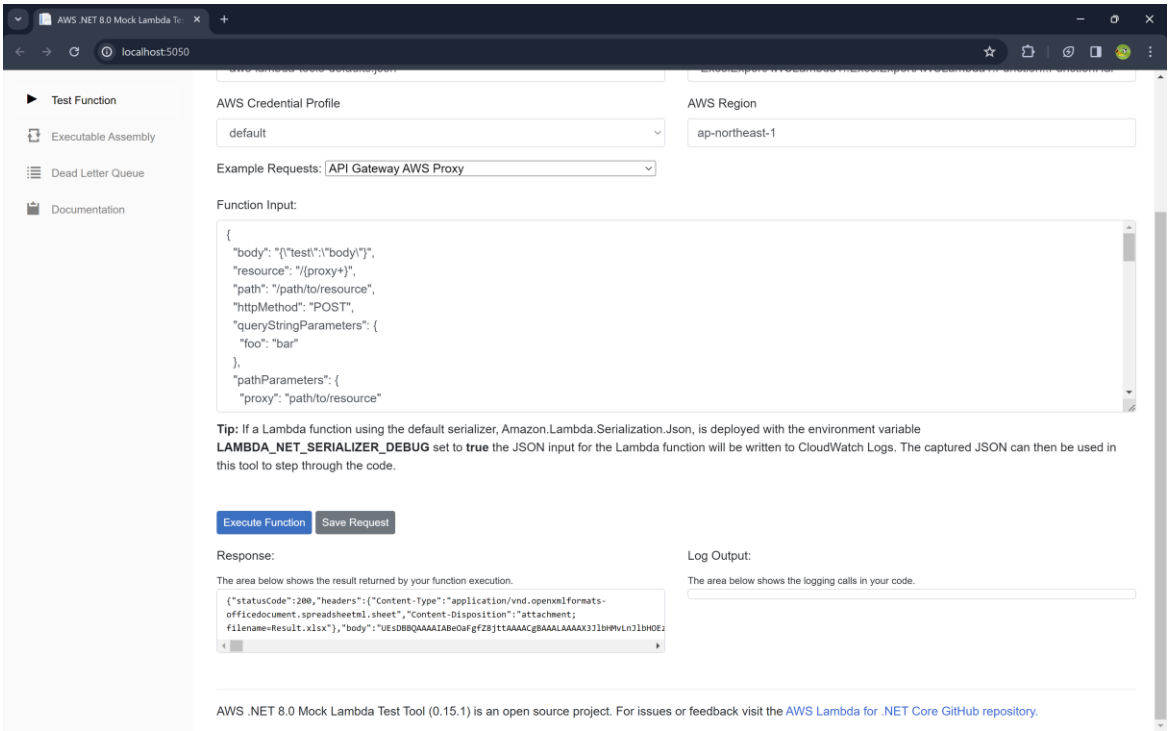
> *AWS Lambda プロキシ統合から**バイナリメディアを返すには、Lambda 関数からのレスポンスを base64 でエンコードします。**また、API のバイナリメディアタイプを設定する必要があります。*
>
> *https://docs.aws.amazon.com/ja_jp/apigateway/latest/developerguide/lambda-proxy-binary-media.html*

デバッグ実行で確認

作成した Lambda 関数アプリケーションをローカルでデバッグ実行して確認します。[F5] キーをクリックすると Mock Lambda Test Tool が起動します。

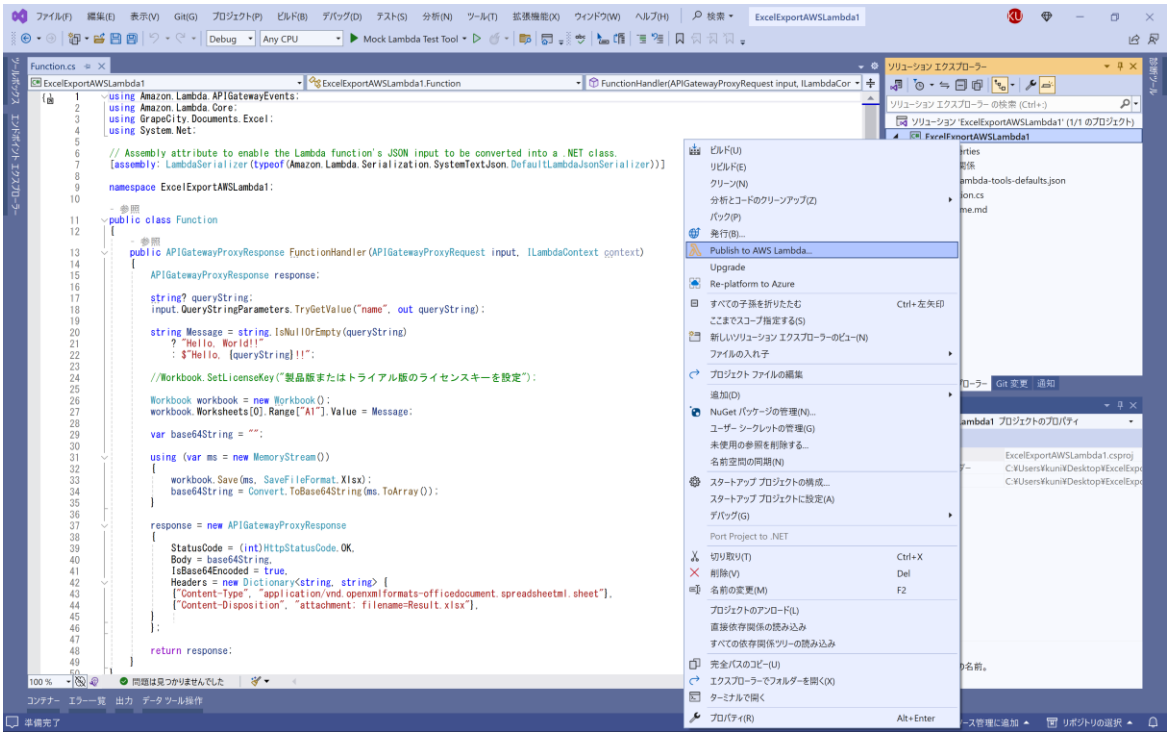


Example Requests に「API Gateway AWS Proxy」を設定して［Exceute Function］をクリックします。Response の body に base64 にエンコードされた文字列が格納されていれば OK です。

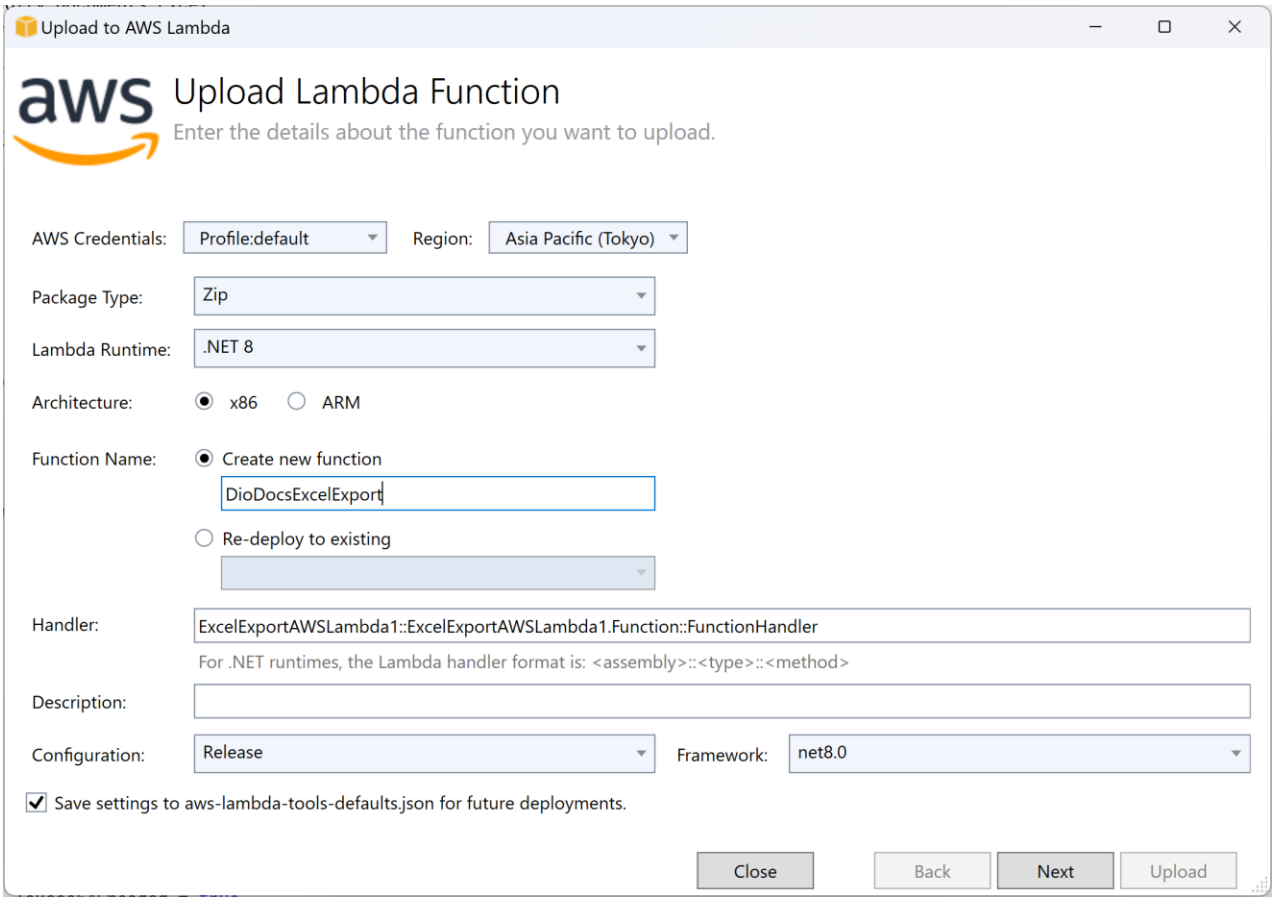


AWS へデプロイ

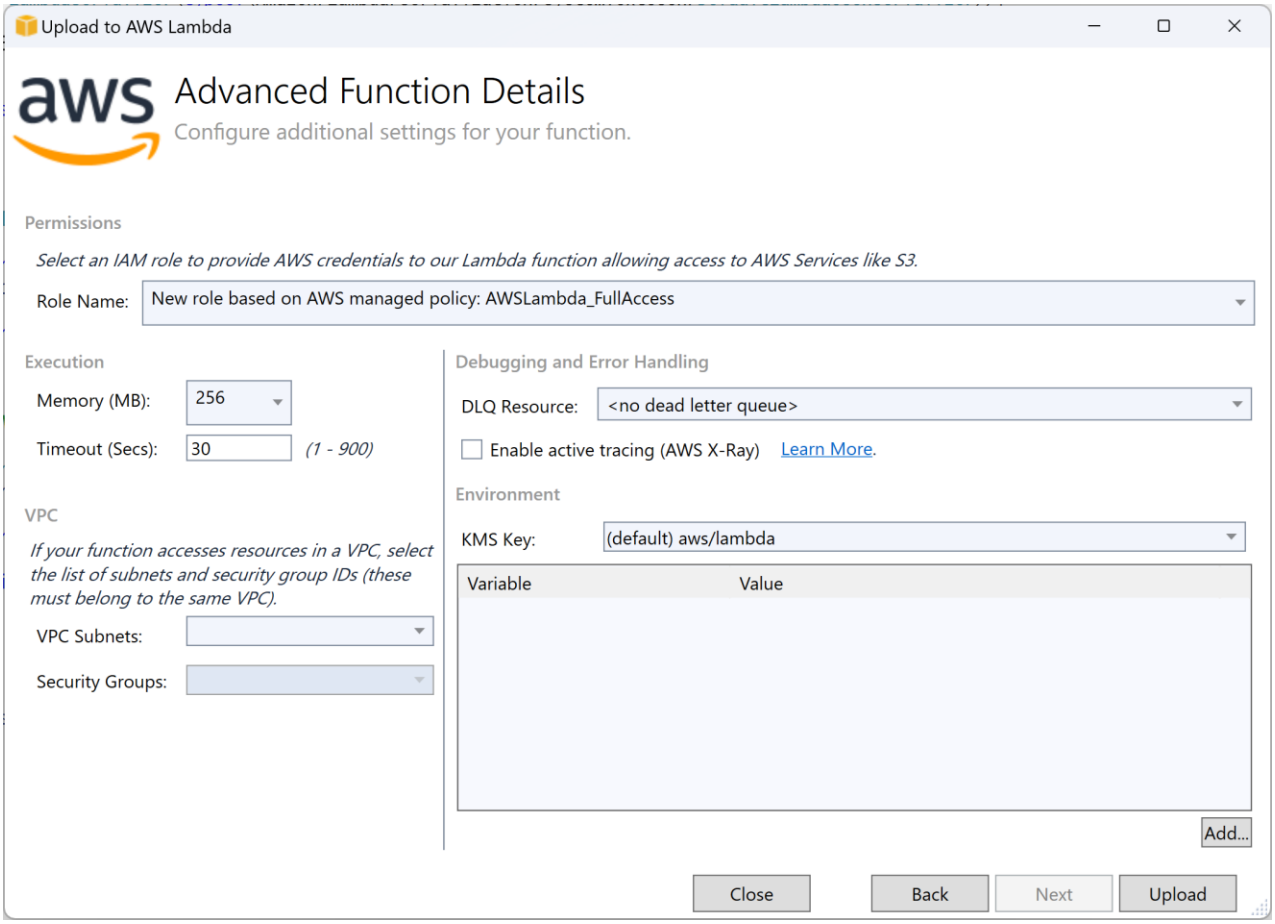
作成した Lambda 関数アプリケーションを AWS へデプロイして確認します。ソリューションエクスプローラーから「ExcelExportAWSLambda1」プロジェクトを右クリックして「Publish to AWS Lambda」を選択します。



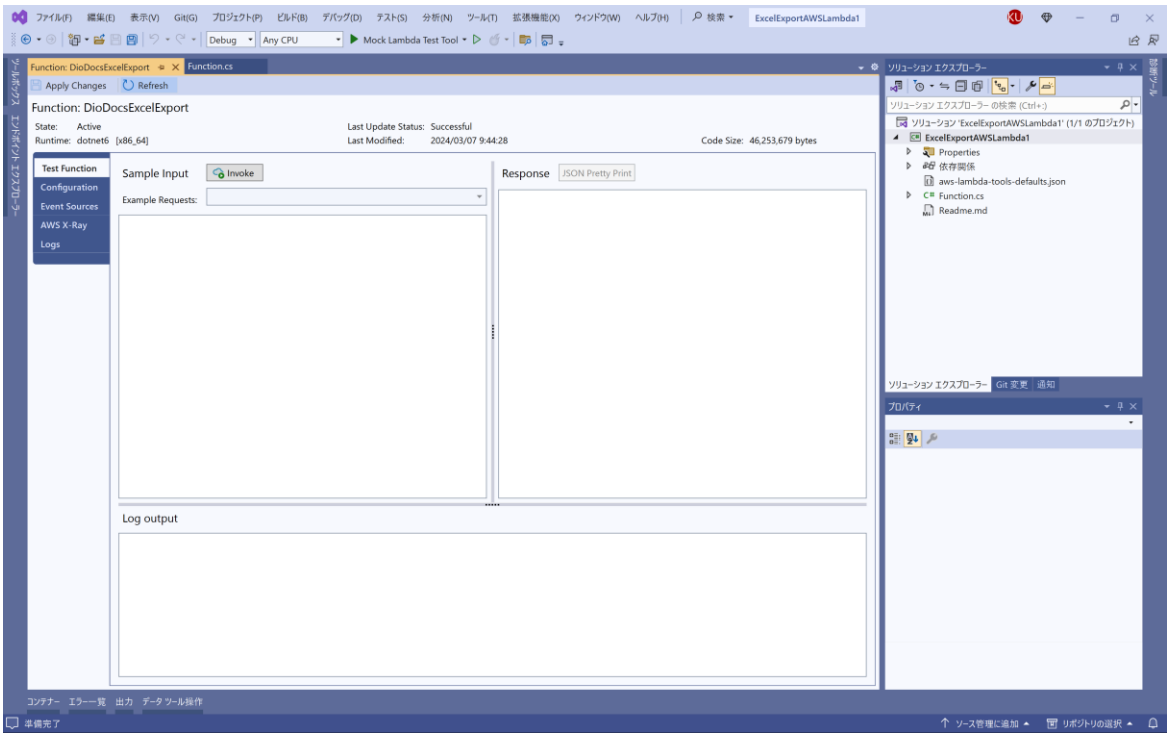
「Function Name」に`DioDocsExcelExport`を入力して［Next］をクリックします。



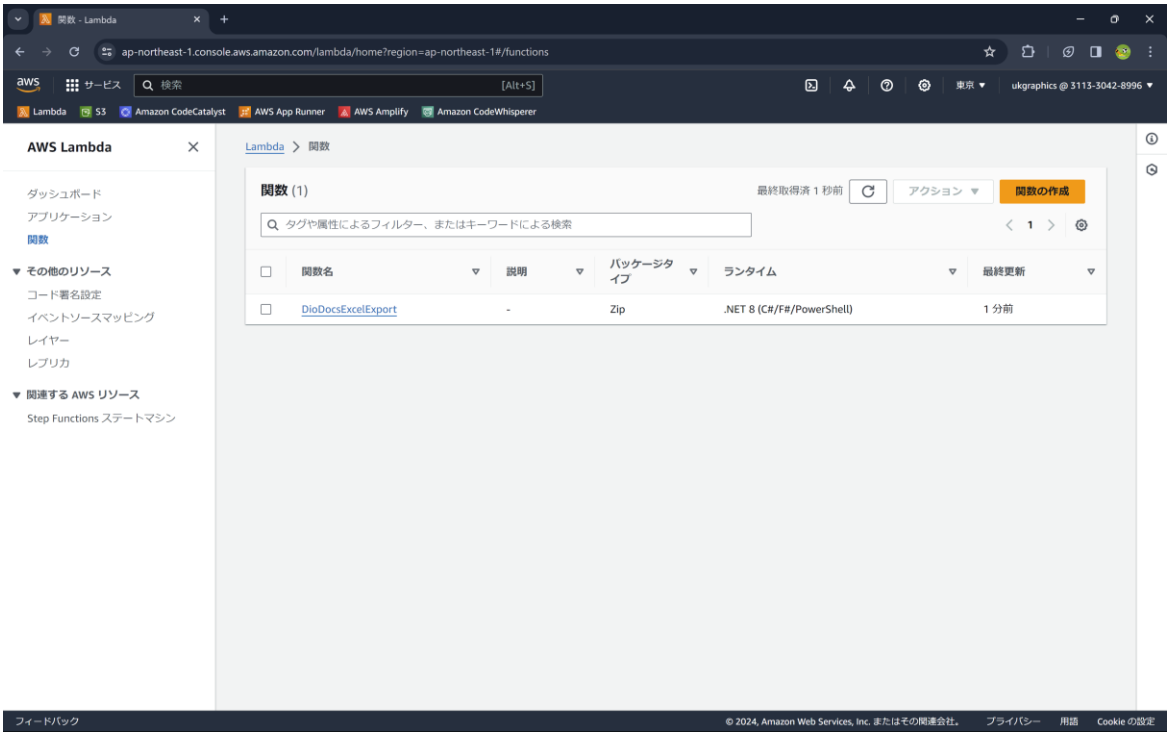
「Role Name」に`New role based on AWS managed policy: AWSLambda_FullAccess`を設定して［Upload］をクリックします。



成功すると以下の画面が表示されます。

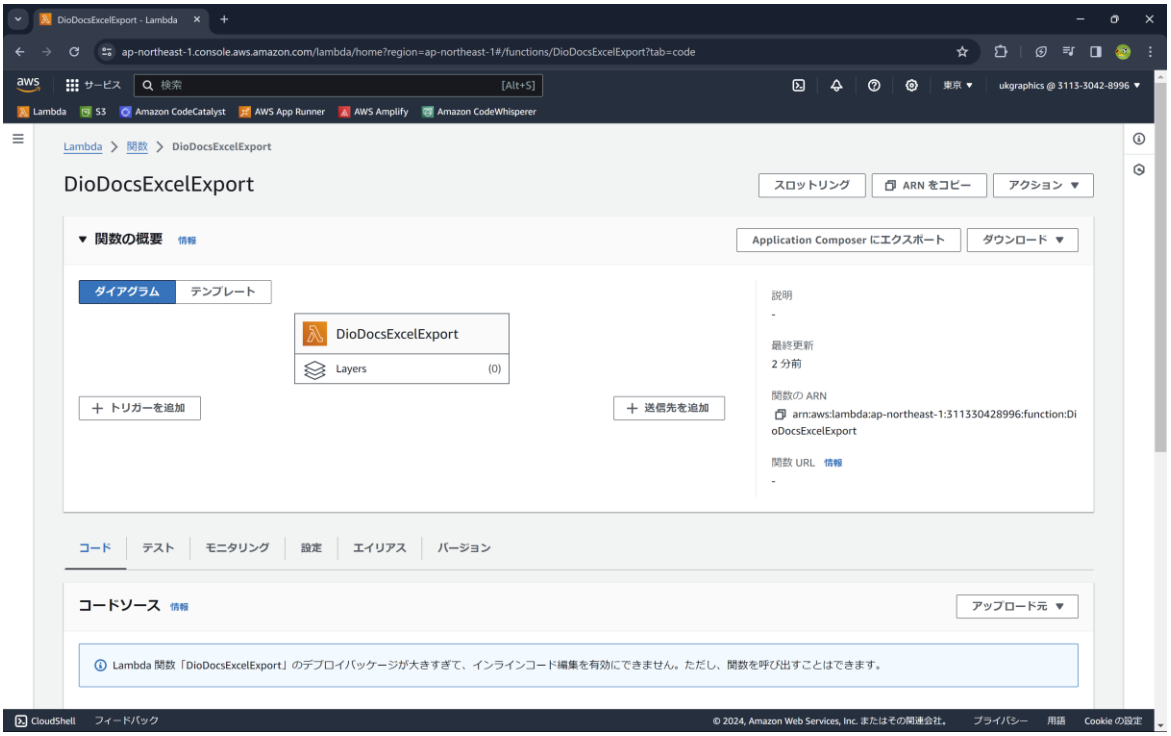


AWS のコンソールで AWS Lambda の「関数」を選択するとデプロイした Lambda 関数「DioDocsExcelExport」が表示されます。

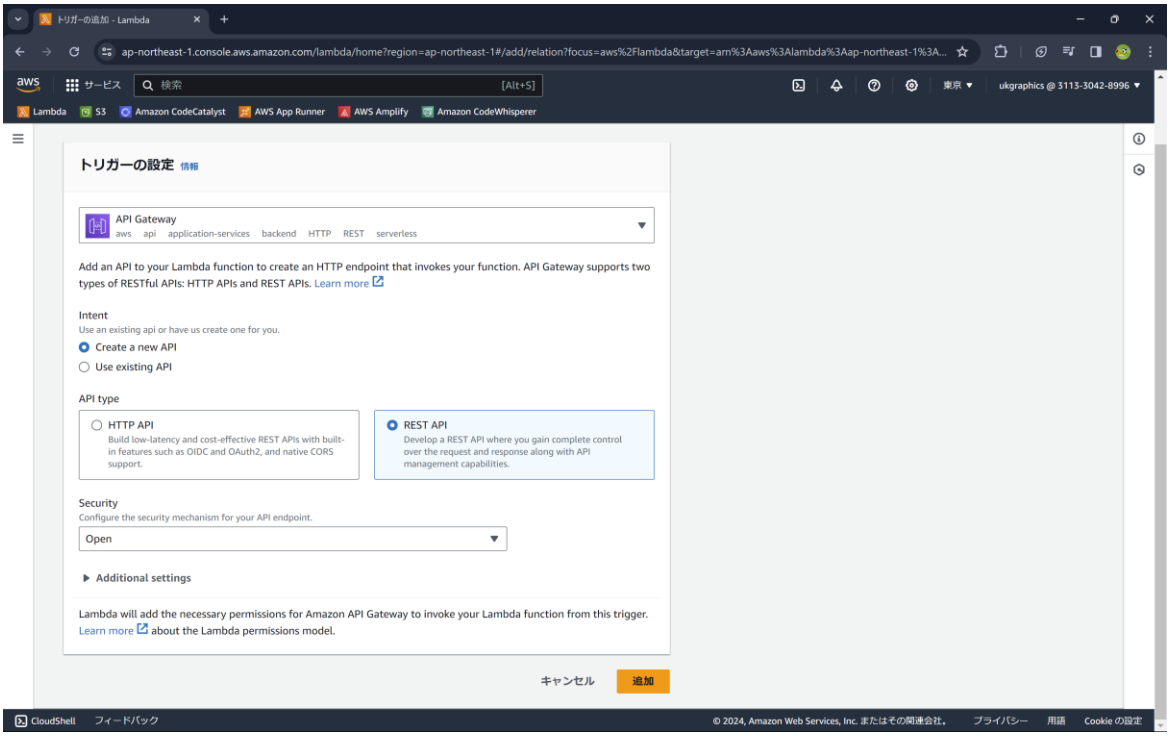


トリガーの追加

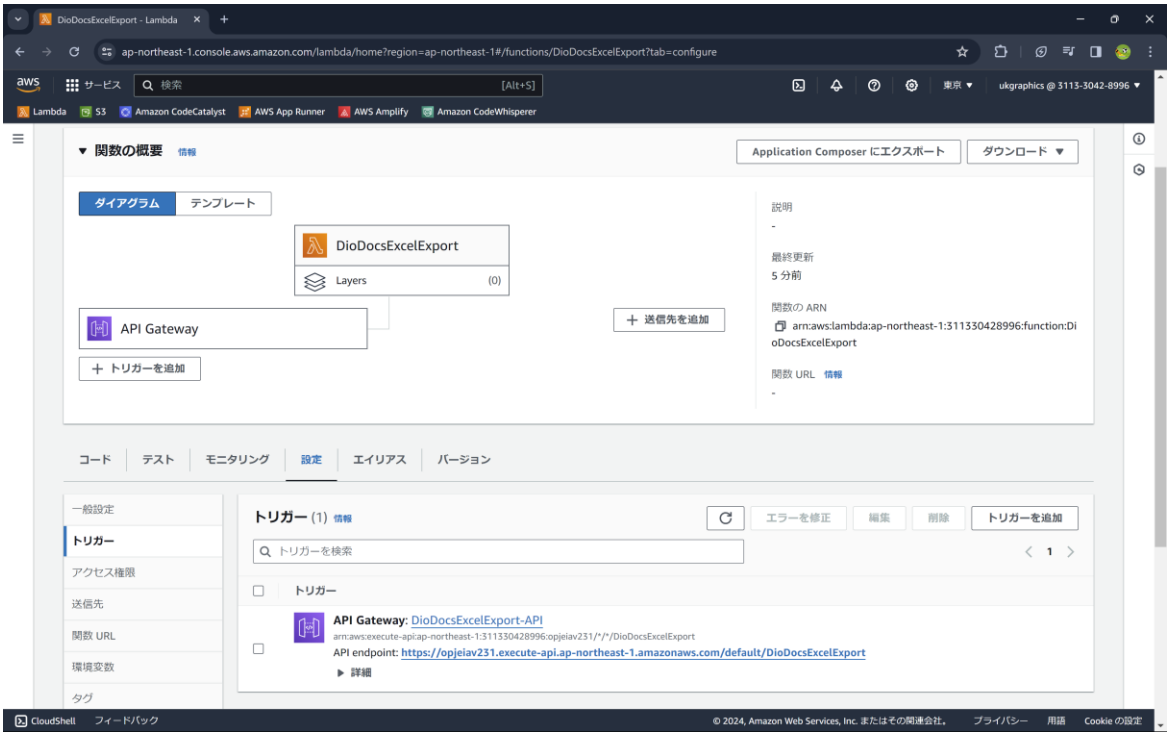
デプロイした Lambda 関数「DioDocsExcelExport」をクリックして以下の画面から［トリガーを追加］をクリックします。



「API Gateway」を選択し、さらに「Create a new API」を選択します。作成する API タイプは「REST API」を選択して、セキュリティは「Open」を選択します。この状態で［追加］をクリックします。

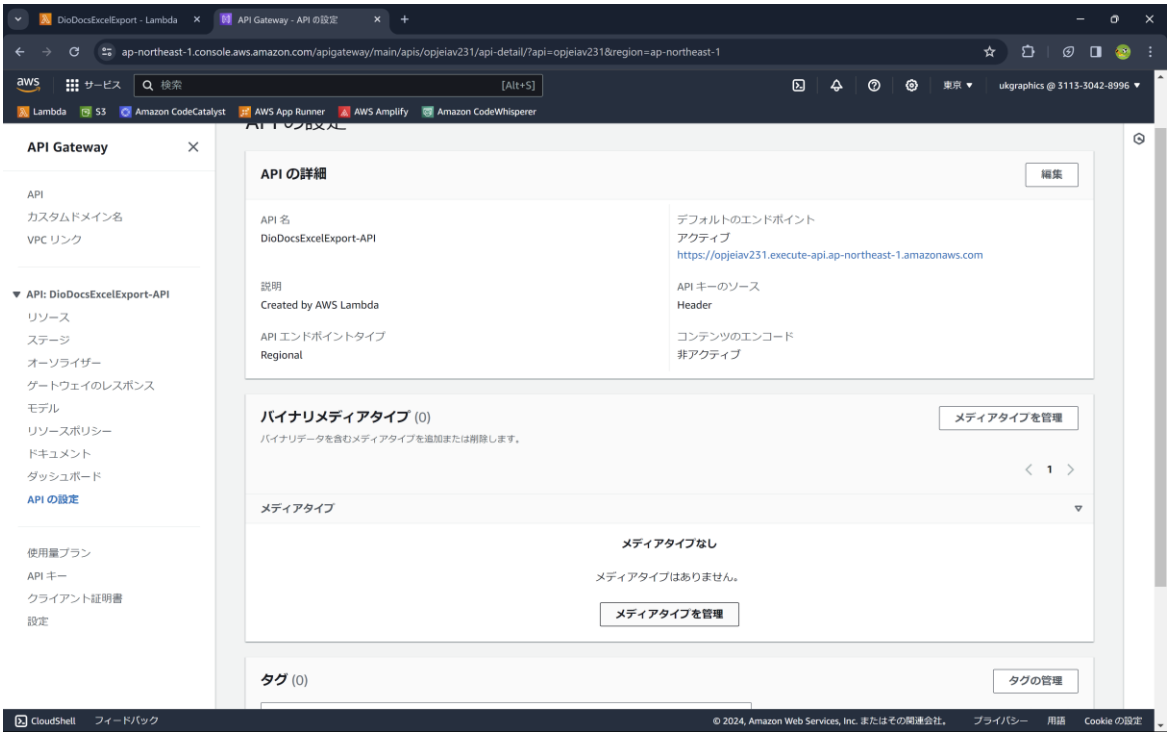


以下のようにトリガーに API Gateway が追加されます。

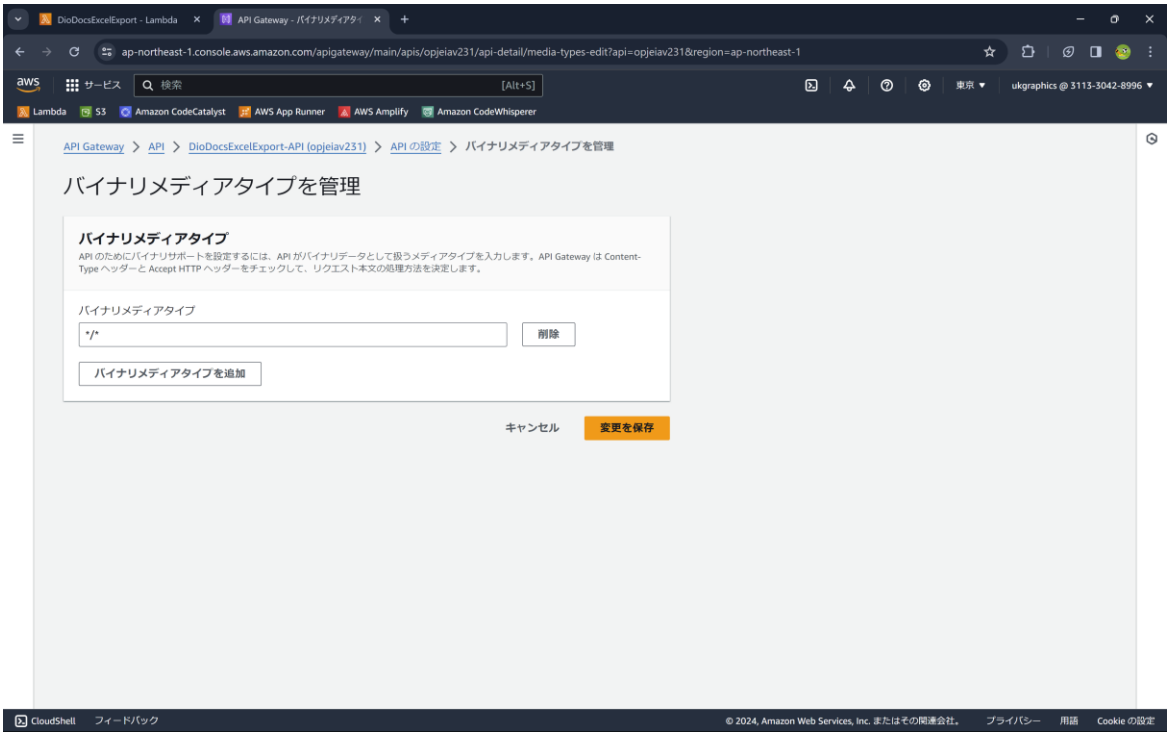


API Gateway のバイナリメディアタイプを設定

AWS のコンソールで作成した API「DioDocsExcelExport-API」の「API の設定」から「メディアタイプの管理」をクリックします。



「バイナリメディアタイプを追加」をクリックして「*/*」を追加します。追加後に［変更を保存］をクリックします。



「*/*」を設定する理由ですが、 Lambda 関数と Amazon API Gateway を連携させる「AWS Lambda プロキシ統合」を利用する際の決まり事になっています。

この統合例でウェブブラウザを使用して API を呼び出すには、API のバイナリメディアタイプを `*/*` に設定します 。API Gateway は、クライアントからの最初の `Accept` ヘッダーを使用して、レスポンスがバイナリメディアを返すかどうかを判断します。ブラウザからのリクエストなど、`Accept` ヘッダー値の順序を制御できない場合に、バ

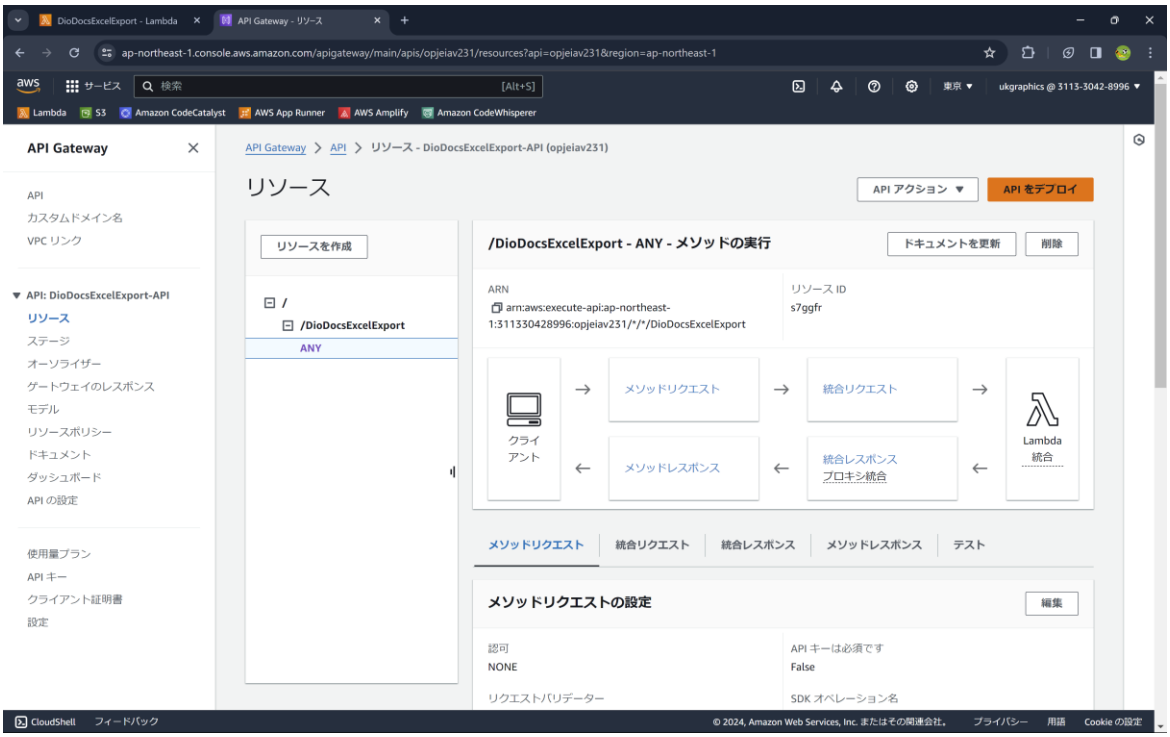
イナリメディアを返すには、API のバイナリメディアタイプを `*/*` (すべてのコンテンツタイプ) に設定します。

https://docs.aws.amazon.com/ja_jp/apigateway/latest/developerguide/lambda-proxy-binary-media.html

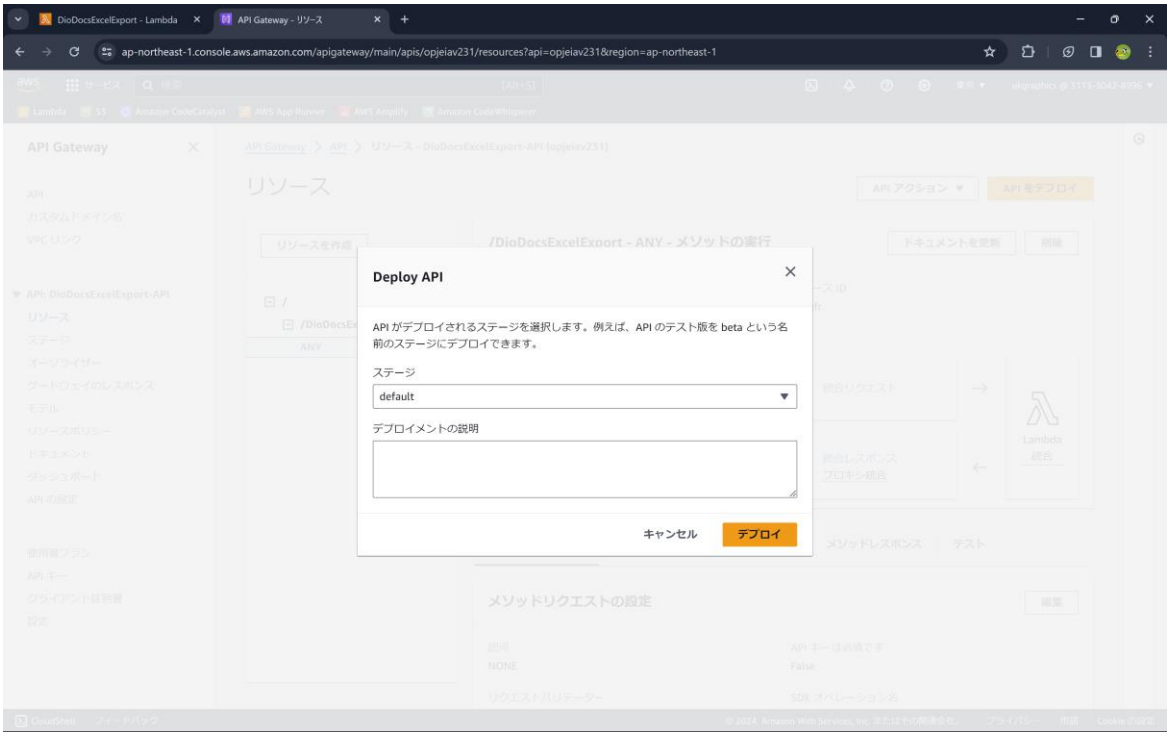
この設定ですが、気を利かせたつもりで `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet` など、固有のバイナリメディアタイプを設定してしまうと base64 エンコードされたただの文字列が出力されてしまうので注意が必要です。

デプロイしたアプリケーションを確認

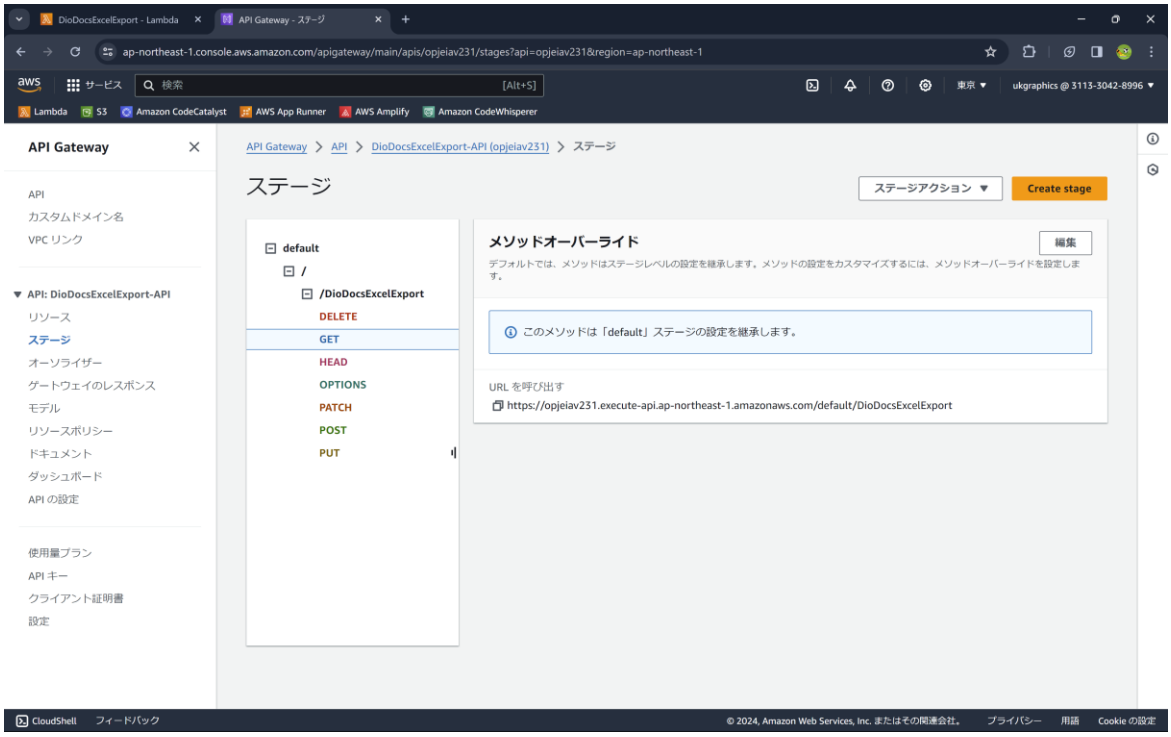
AWS のコンソールで作成した API「DioDocsExcelExport-API」の「リソース」から「API のデプロイ」を選択します。



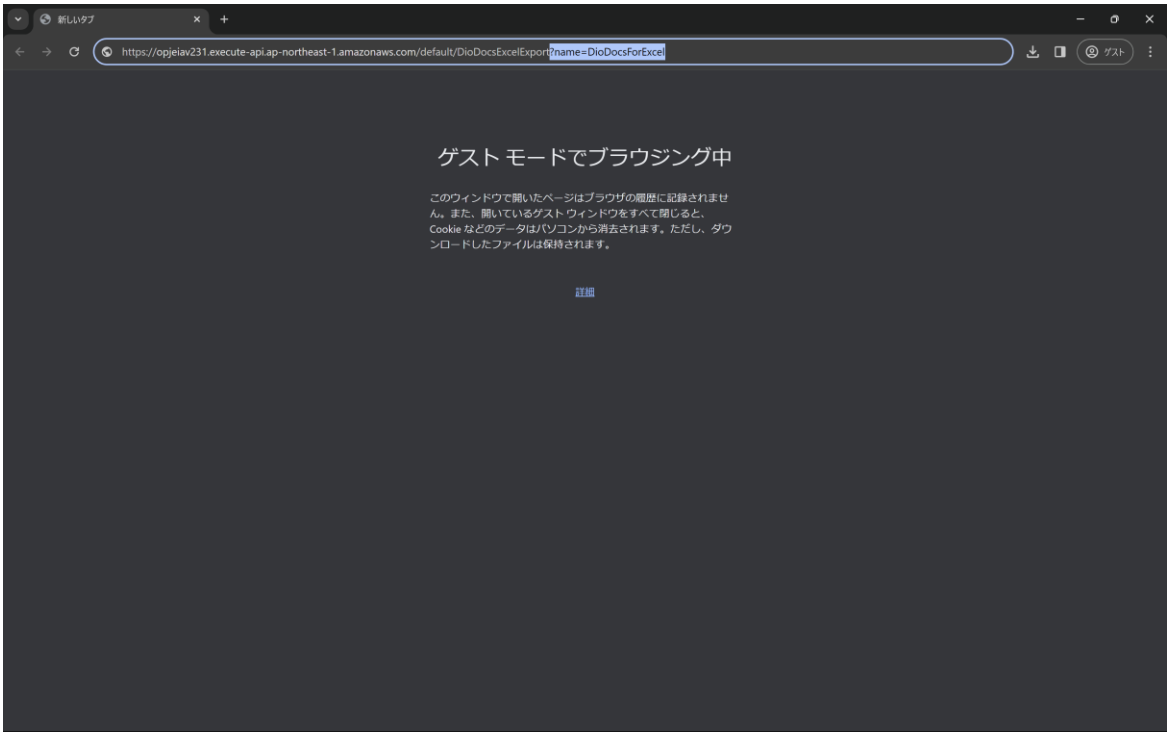
デプロイされるステージは「default」を選択して［デプロイ］をクリックします。



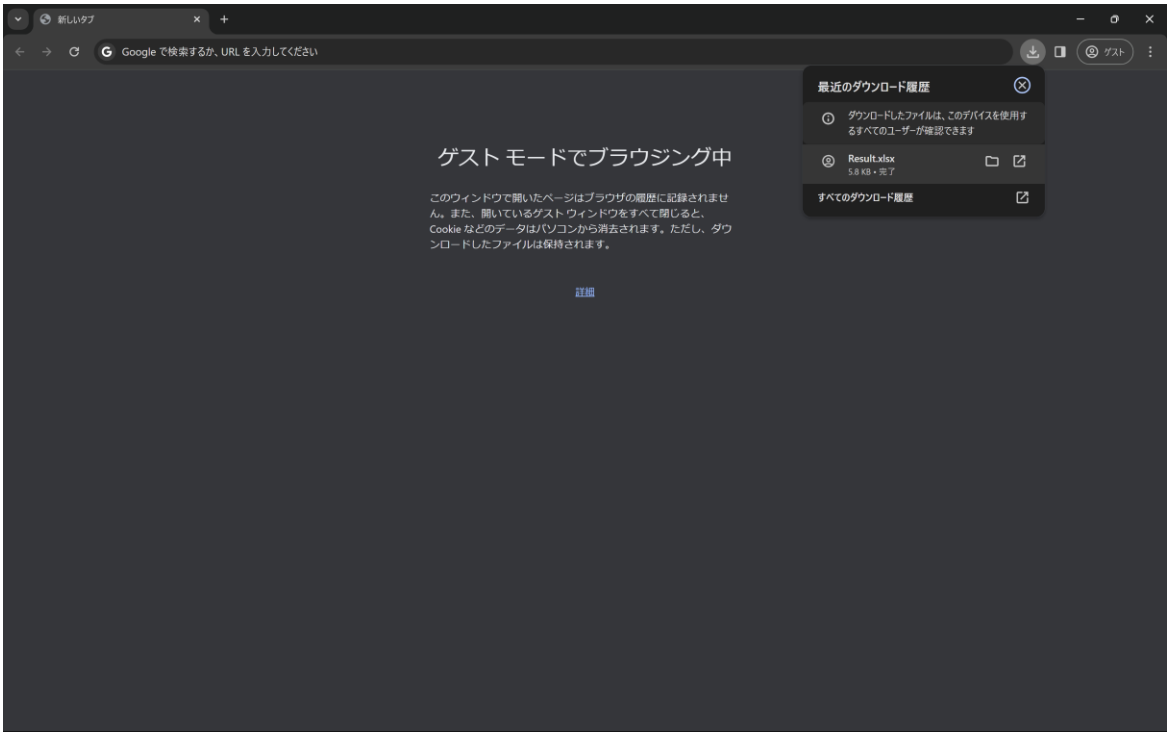
API「DioDocsExcelExport-API」の「ステージ」から「default - / - /DioDocsExcelExport - GET」を選択します。

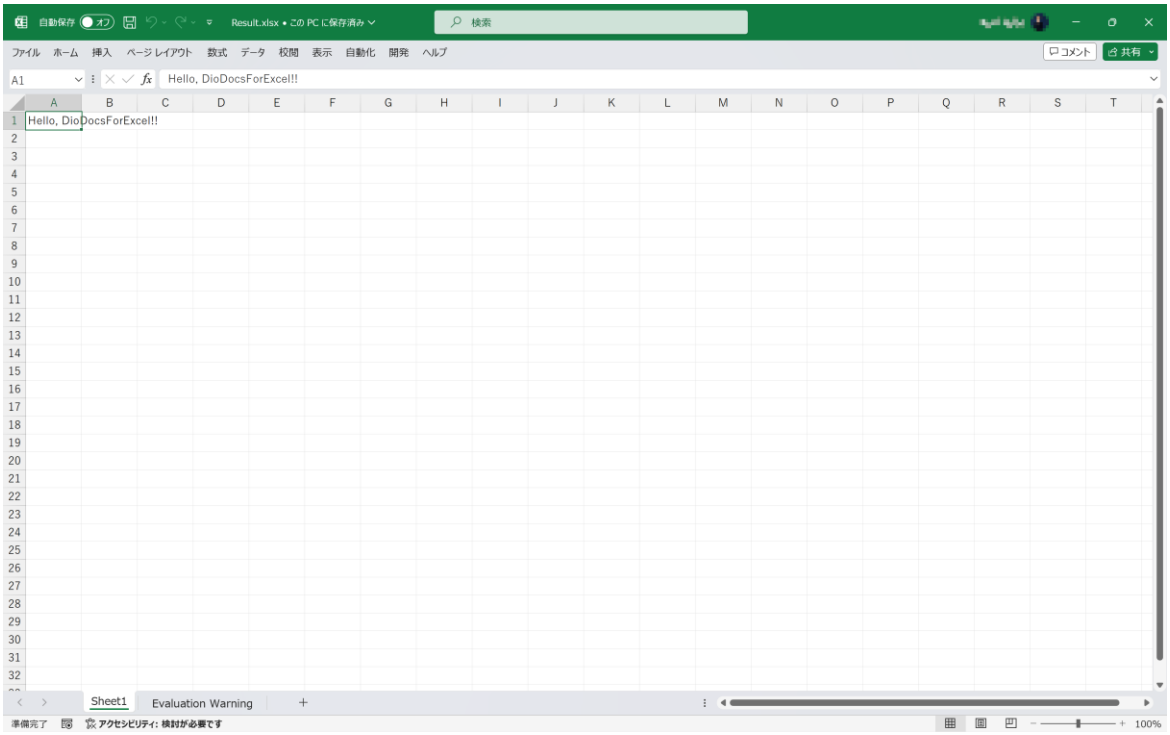


「URL の呼び出し」に表示されている API の URL をコピーしてブラウザに張り付けて、さらにクエリパラメータと文字列「`?name=DioDocsForExcel`」を追加します。



この API を実行するとクエリパラメータで渡した文字列「DioDocsForExcel」が追加された Excel ファイル「Result.xlsx」がローカルに出力されます。





PDF を出力するには？

Visual Studio の「NuGet パッケージ マネージャー」から DioDocs for PDF のパッケージ「`DioDocs.Pdf.ja`」をインストールします。 DioDocs for PDF で PDF ファイルを作成するコードを追加して `FunctionHandler` を以下のように更新します。PDF ファイルを出力するのでそれに合わせて `APIGatewayProxyResponse` の `Header` の内容も変更しています。

```
public APIGatewayProxyResponse FunctionHandler(APIGatewayProxyRequest input, ILambdaContext context)
{
    APIGatewayProxyResponse response;

    string? queryString;
    input.QueryStringParameters.TryGetValue("name", out queryString);

    string Message = string.IsNullOrEmpty(queryString)
        ? "Hello, World!!"
        : $"Hello, {queryString}!!";

    //GcPdfDocument.SetLicenseKey("製品版またはトライアル版のライセンスキーを設定");

    GcPdfDocument doc = new GcPdfDocument();
    GcPdfGraphics g = doc.NewPage().Graphics;

    g.DrawString(Message,
        new TextFormat() { Font = StandardFonts.Helvetica, FontSize = 12 },
```

```
        new PointF(72, 72));

    var base64String = "";

    using (var ms = new MemoryStream())
    {
        doc.Save(ms, false);
        base64String = Convert.ToBase64String(ms.ToArray());
    }

    response = new APIGatewayProxyResponse
    {
        StatusCode = (int)HttpStatusCode.OK,
        Body = base64String,
        IsBase64Encoded = true,
        Headers = new Dictionary<string, string> {
            {"Content-Type", "application/pdf"},
            {"Content-Disposition", "attachment; filename=Result.pdf"},
        }
    };

    return response;
}
```

さいごに

動作を確認できる AWS Lambda アプリケーションのサンプルはこちらです。
https://github.com/MESCIUSJP/ExcelExportAWSLambda1
https://github.com/MESCIUSJP/PDFExportAWSLambda1

AWS Lambda と DioDocs で Excel や PDF ファイルを出力する（2）

前回に引き続き、本記事でも AWS Lambda で「DioDocs（ディオドック）」を使用した C#（.NET 8）の Lambda 関数アプリケーションを作成し、Excel や PDF ファイルを出力する方法について紹介します。

実装する内容

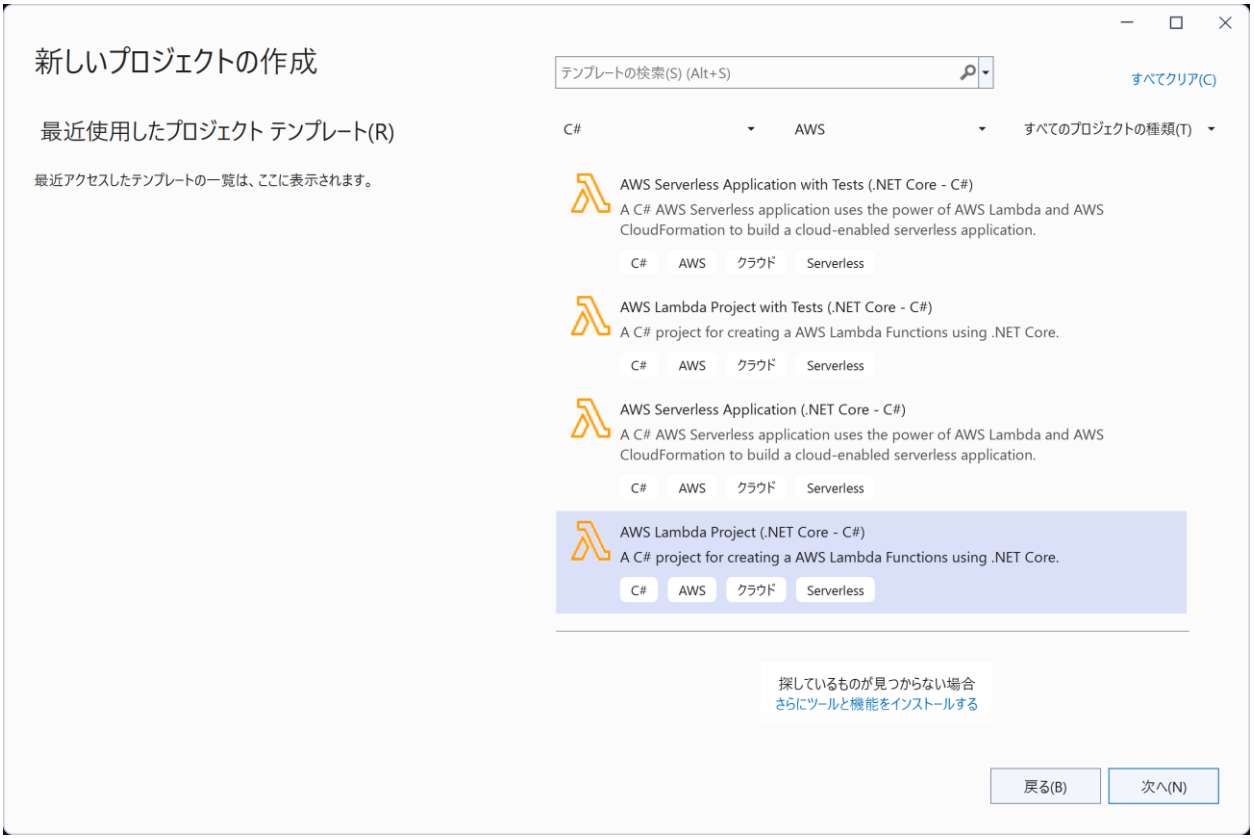
今回も AWS Lambda アプリケーションで Amazon API Gateway から HTTP リクエストを受け取る Lambda 関数を作成します。この関数の実行時に DioDocs を使用して Excel と PDF ファイルを作成し、 HTTP リクエストのクエリパラメータで受け取った文字列を追加します。

前回は作成した Excel と PDF ファイルを関数から Amazon API Gateway に渡して HTTP レスポンスで直接ローカルへ出力していましたが、**今回は作成した Excel と PDF ファイルを Amazon S3 へ出力します。**Amazon S3 への保存には AWS SDK for .NET を使用します。

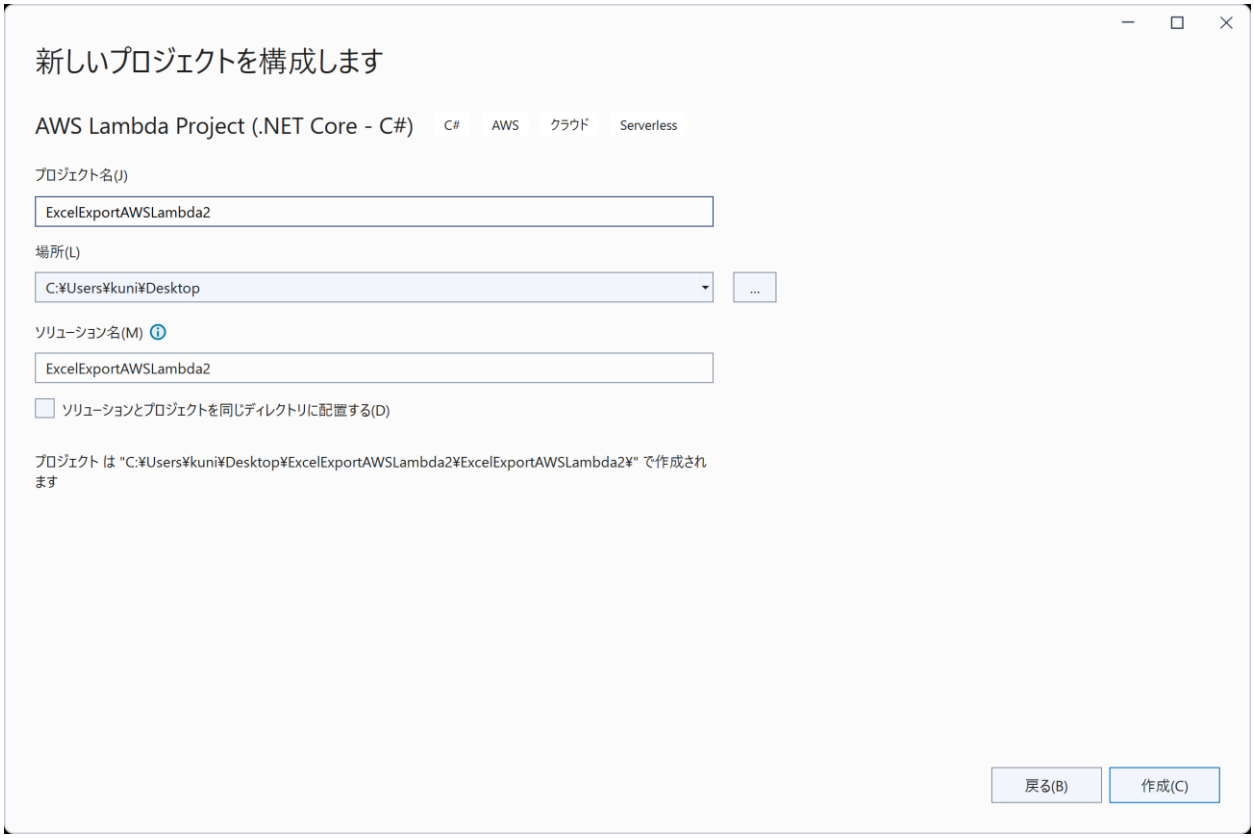
AWS Lambda アプリケーションを作成

以下のドキュメントを参考に AWS Lambda アプリケーションを作成していきます。
基本 AWS Lambda プロジェクト - AWS Toolkit for Visual Studio

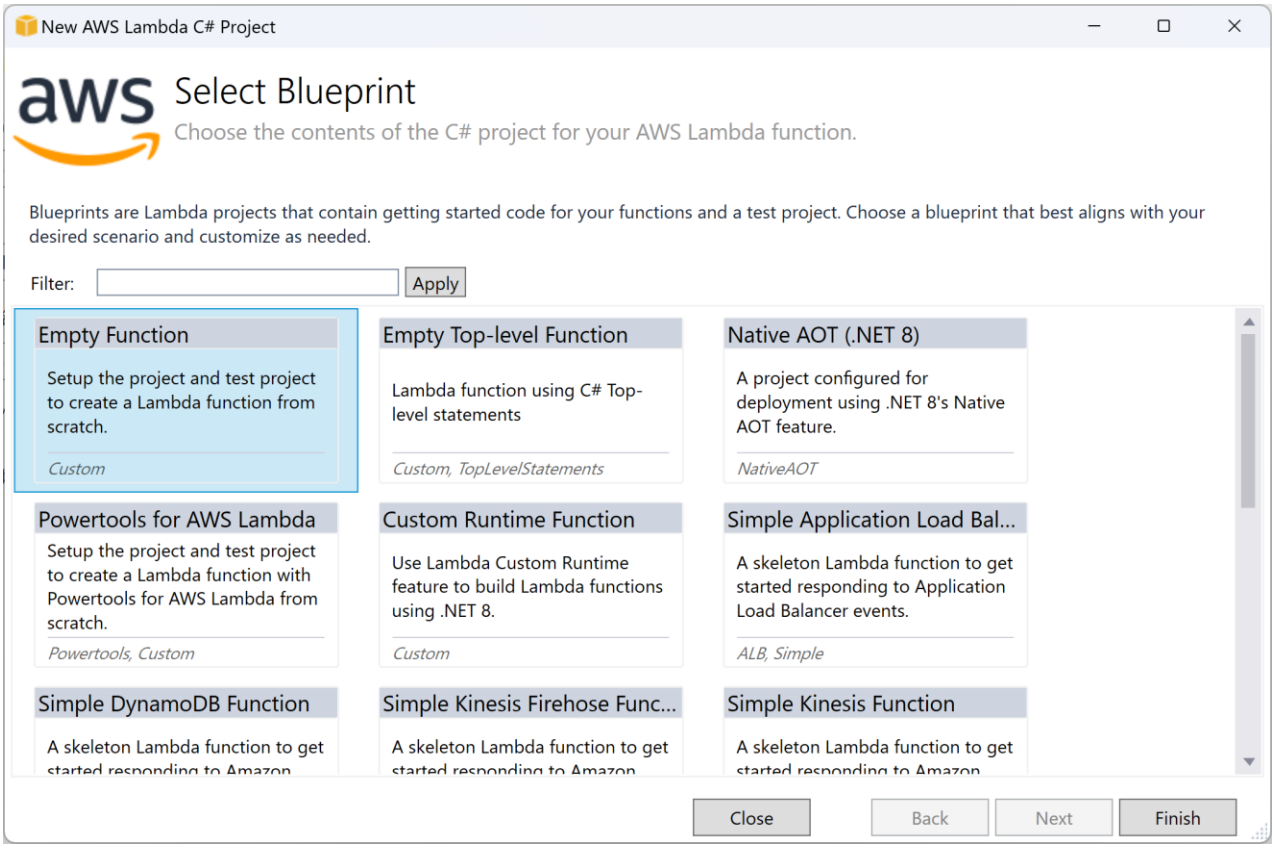
Visual Studio 2022 でプロジェクトテンプレート「AWS Lambda Project (.NET Core – C#)」を選択して［次へ］をクリックします。



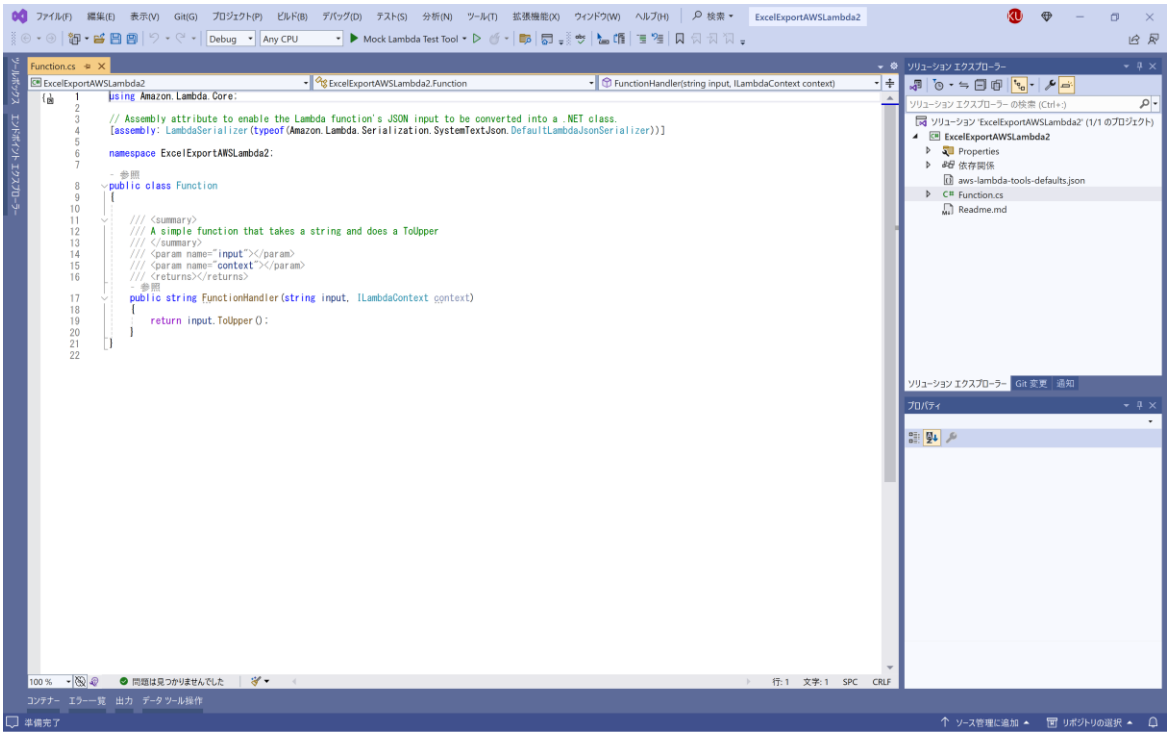
プロジェクト名に「ExcelExportAWSLambda2」を入力して［作成］をクリックします。



AWS Lambda Project のテンプレートを選択します。「Empty Function」を選択して［Finish］をクリックします。

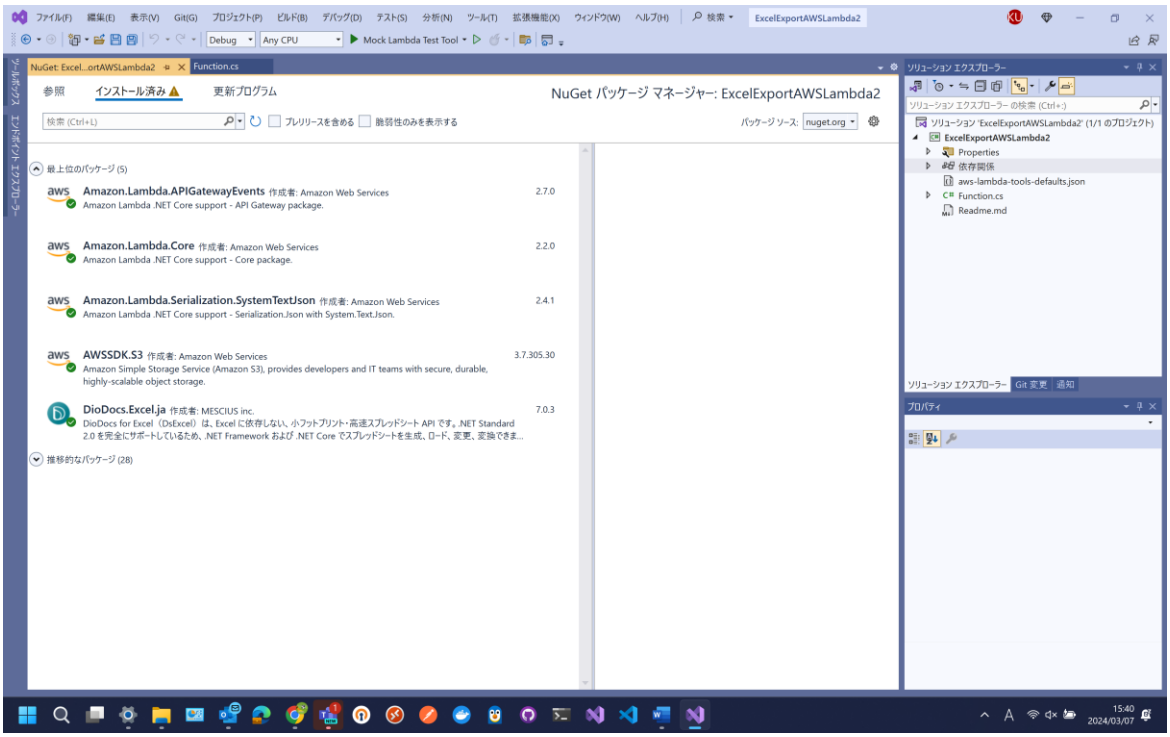


「ExcelExportAWSLambda2」プロジェクトが作成されます。



NuGet パッケージの追加

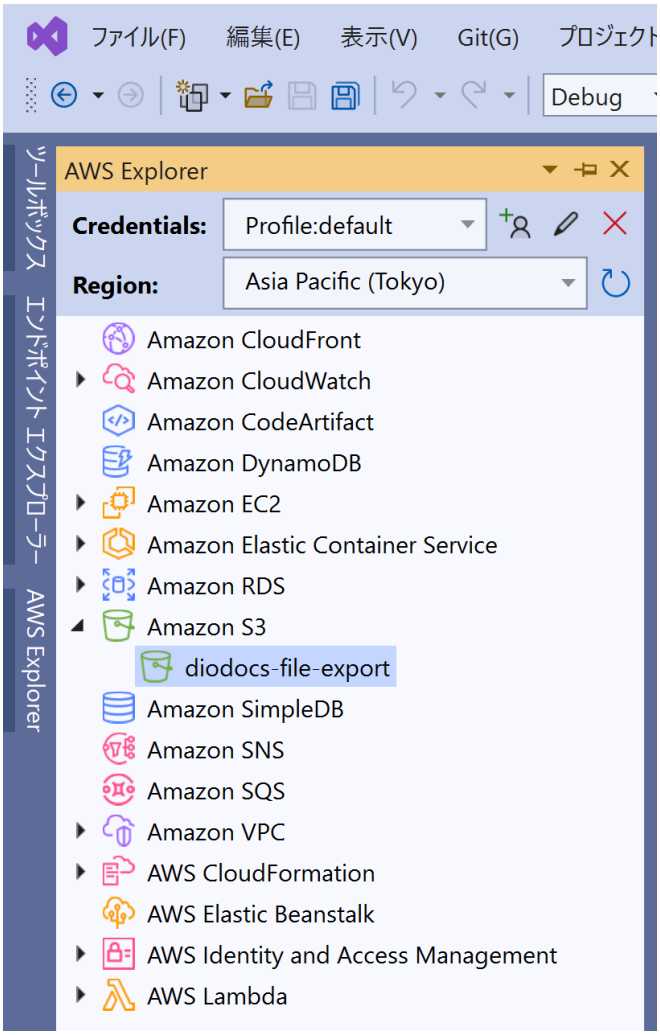
Visual Studio の「NuGet パッケージ マネージャー」から Amazon API Gateway のイベントを処理するためのパッケージ「`Amazon.Lambda.APIGatewayEvents`」と Amazon S3 を使うためのパッケージ「`AWSSDK.S3`」、そして DioDocs for Excel のパッケージ「`GrapeCity.DioDocs.Excel.ja`」をインストールします。



Amazon S3 にバケットを作成

Amazon S3 に DioDocs for Excel で作成した Excel ファイルの保存先になるバケット「diodocs-file-export」を、

Visual Studio の「AWS Explorer」から作成します。



Amazon API Gateway を使うコードを追加

Lambda 関数が Amazon API Gateway から HTTP リクエストを受け取り、Lambda 関数から Amazon API Gateway へ HTTP レスポンスを返すために、以下のように `FunctionHandler` の引数と戻り値に `APIGatewayProxyRequest` と `APIGatewayProxyResponse` を設定します。

```
public async Task<APIGatewayProxyResponse> FunctionHandler(APIGatewayProxyRequest input, ILambdaContext
 context)
```

DioDocs for Excel を使うコードを追加

DioDocs for Excel で Excel ファイルを作成するコードを追加して `FunctionHandler` を以下のように更新します。

```
public async Task<APIGatewayProxyResponse> FunctionHandler(APIGatewayProxyRequest input, ILambdaContext
 context)
{
    APIGatewayProxyResponse response;
```

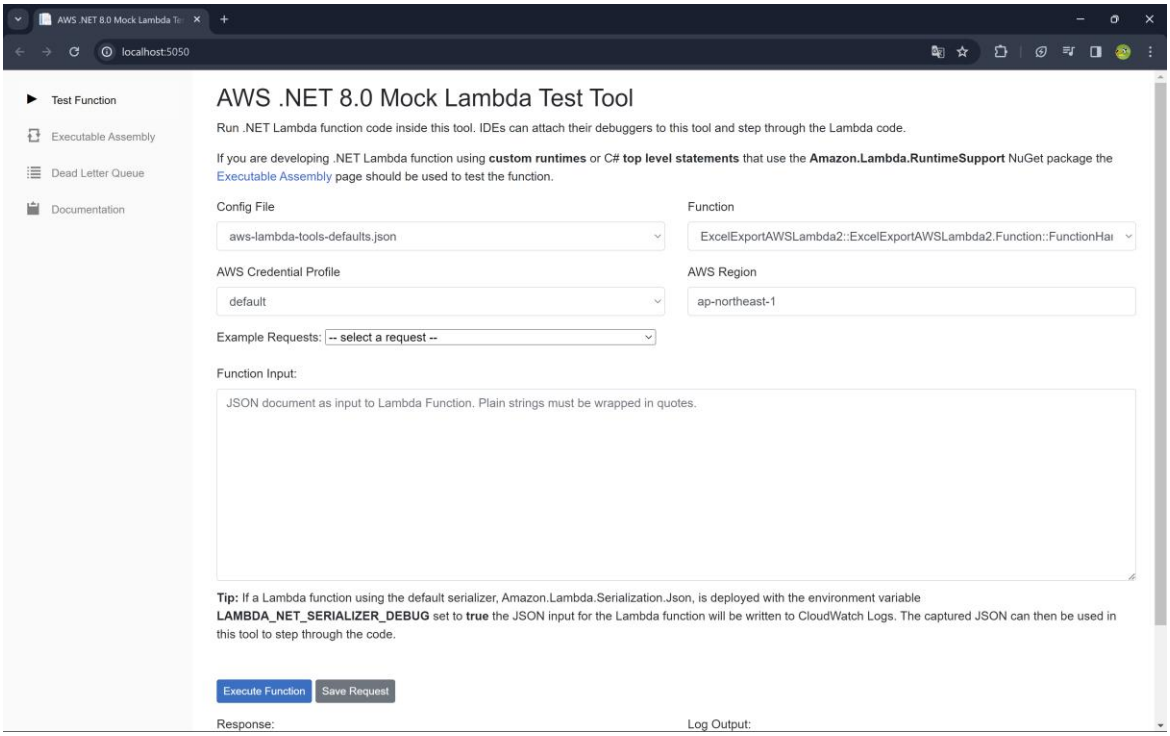
```
try
{
    // クエリ文字列を取得
    string? queryString;
    input.QueryStringParameters.TryGetValue("name", out queryString);

    // ワークシートに追加するテキスト
    string Message = string.IsNullOrEmpty(queryString)
        ? "Hello, World!!"
        : $"Hello, {queryString}!!";

    //Workbook.SetLicenseKey("製品版またはトライアル版のライセンスキーを設定");

    Workbook workbook = new Workbook();
    workbook.Worksheets[0].Range["A1"].Value = Message;

    using (var ms = new MemoryStream())
    {
        workbook.Save(ms, SaveFileFormat.Xlsx);

        // S3 にアップロード
        AmazonS3Client client = new AmazonS3Client(RegionEndpoint.APNortheast1);
        var request = new PutObjectRequest
        {
            BucketName = "diodocs-file-export",
            Key = "Result.xlsx",
            InputStream = ms
        };

        await client.PutObjectAsync(request);
    }

    response = new APIGatewayProxyResponse
    {
        StatusCode = (int)HttpStatusCode.OK,
        Body = "ファイルが保存されました。",
        Headers = new Dictionary<string, string> {
            { "Content-Type", "text/plain; charset=utf-8" }
        }
    };
}
catch (Exception e)
```

```
    {
        response = new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.InternalServerError,
            Body = e.Message,
            Headers = new Dictionary<string, string> {
                { "Content-Type", "text/plain" }
            }
        };
    }

    return response;
}
```

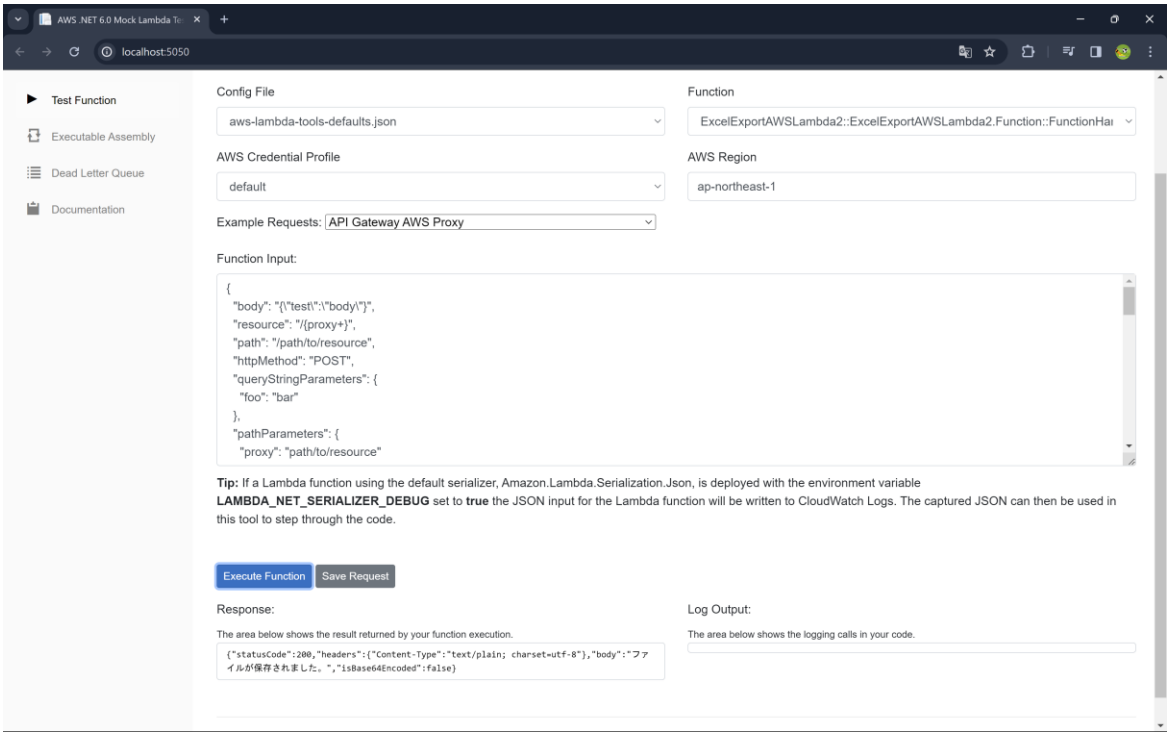
DioDocs for Excel で作成した Excel ファイルを `MemoryStream` に保存し、これを `AmazonS3Client` クラスの `PutObjectAsync` メソッドを使用して Amazon S3 へアップロードしています。

デバッグ実行で確認

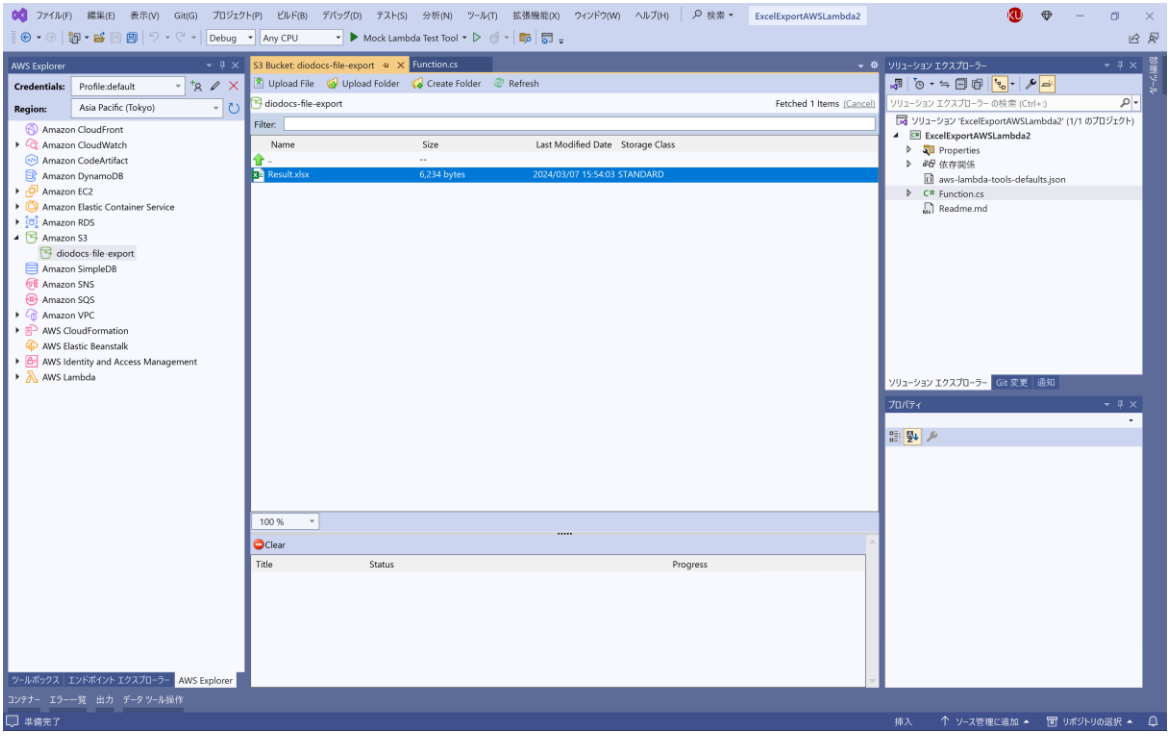
作成した Lambda 関数アプリケーションをローカルでデバッグ実行して確認します。[F5] キーをクリックすると Mock Lambda Test Tool が起動します。



Example Requests に「API Gateway AWS Proxy」を設定して [Exceute Function] をクリックします。Response で以下のように body に「ファイルが保存されました。」が表示されていれば OK です。

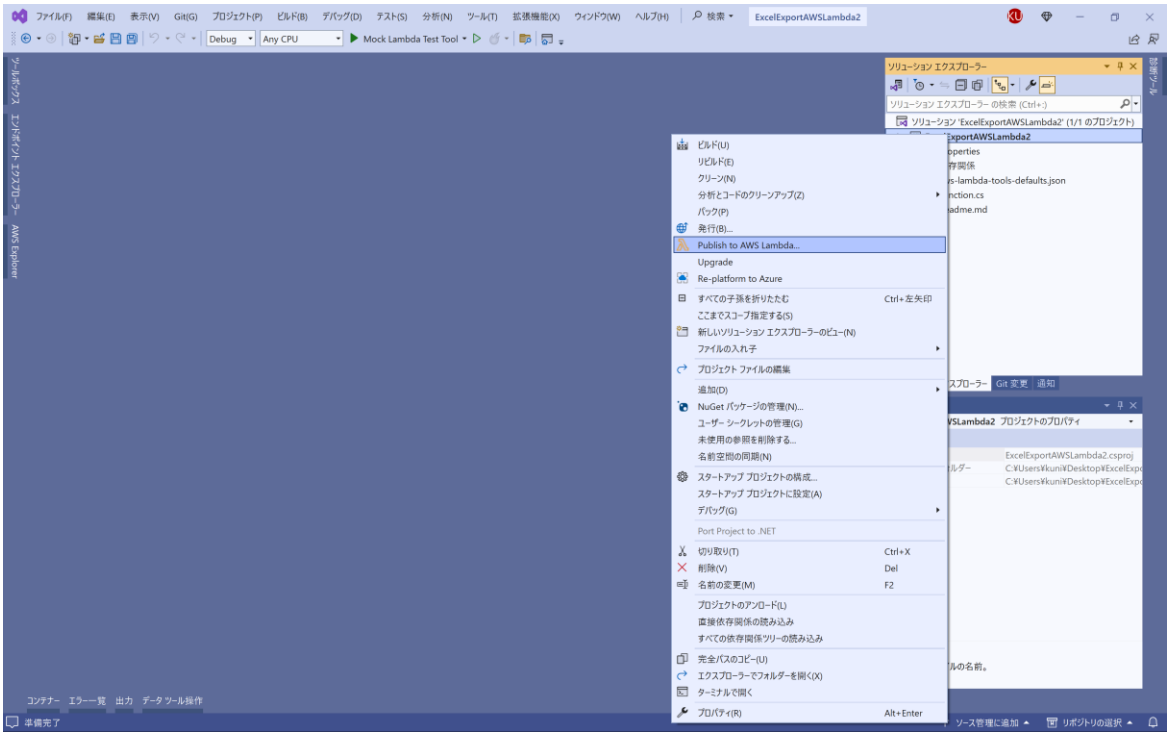


AWS Explorer から「diodocs-file-export」をクリックして、DioDocs for Excel で作成した Excel ファイル「Result.xlsx」がアップロードされているか確認できます。なお、AWS へデプロイした後にも確認するのでアップロードした Excel ファイルはここで一旦削除しておきます。

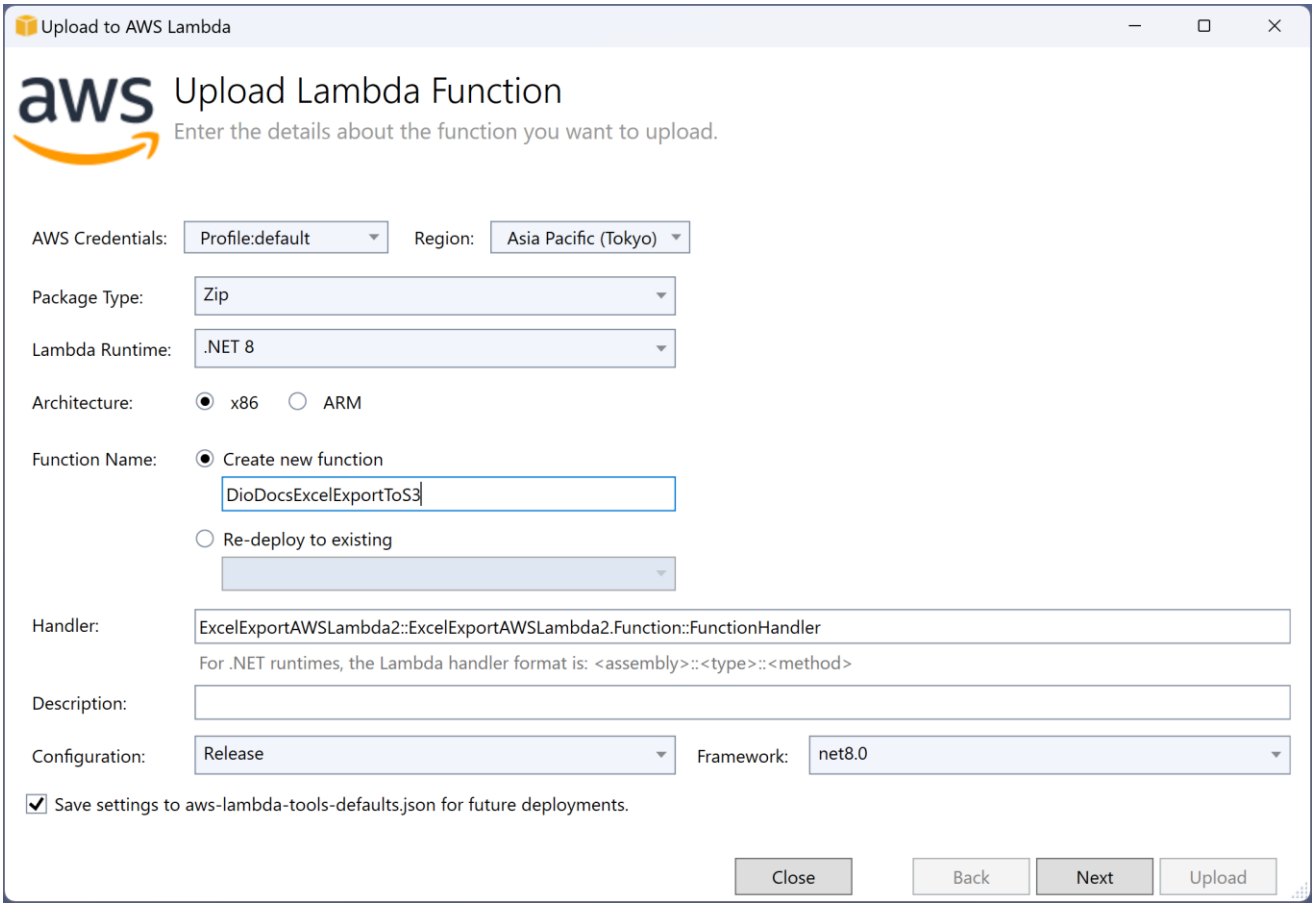


AWS へデプロイ

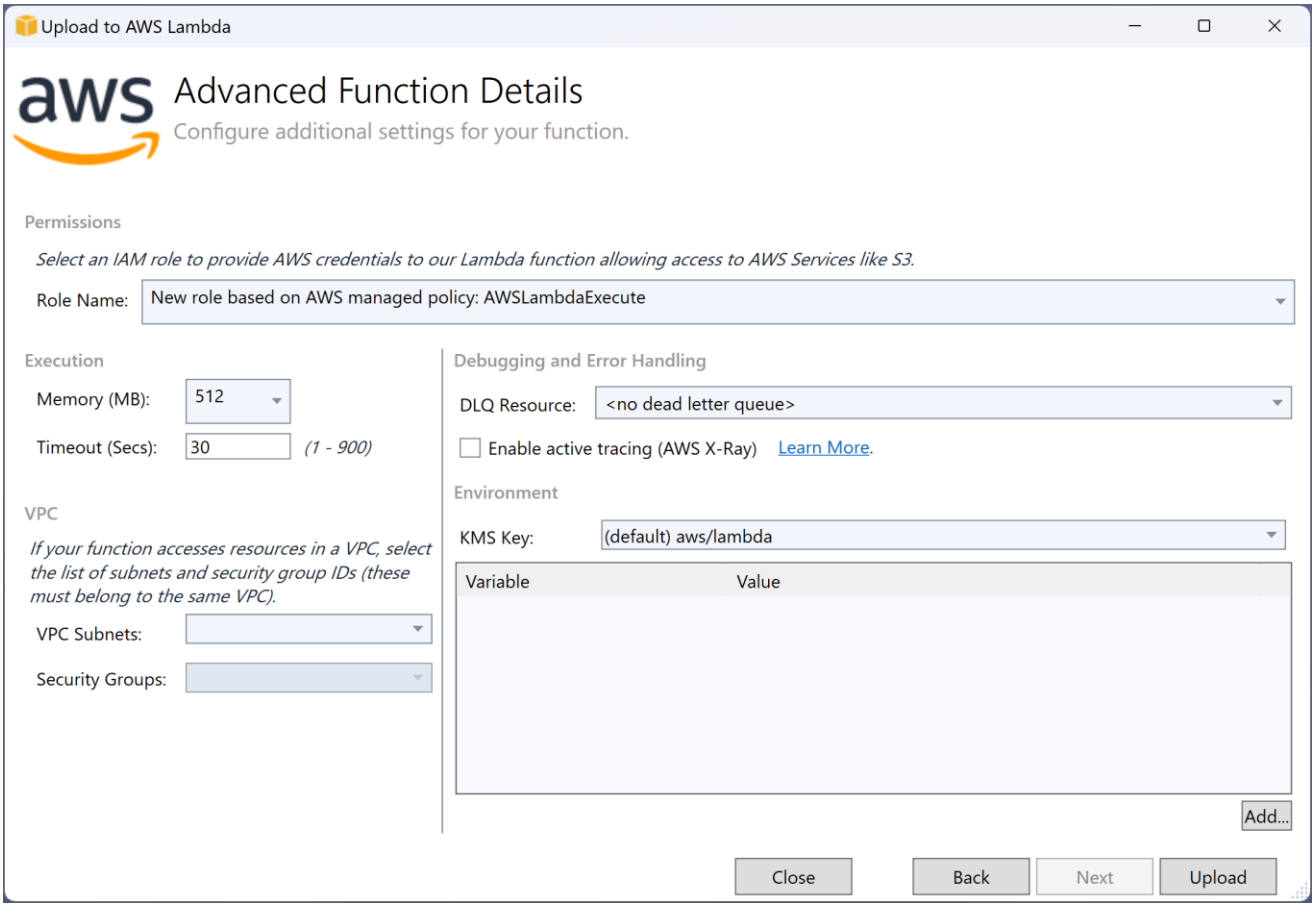
作成した Lambda 関数 アプリケーションを AWS へデプロイして確認します。ソリューションエクスプローラーから「ExcelExportAWSLambda2」プロジェクトを右クリックして「Publish to AWS Lambda」を選択します。



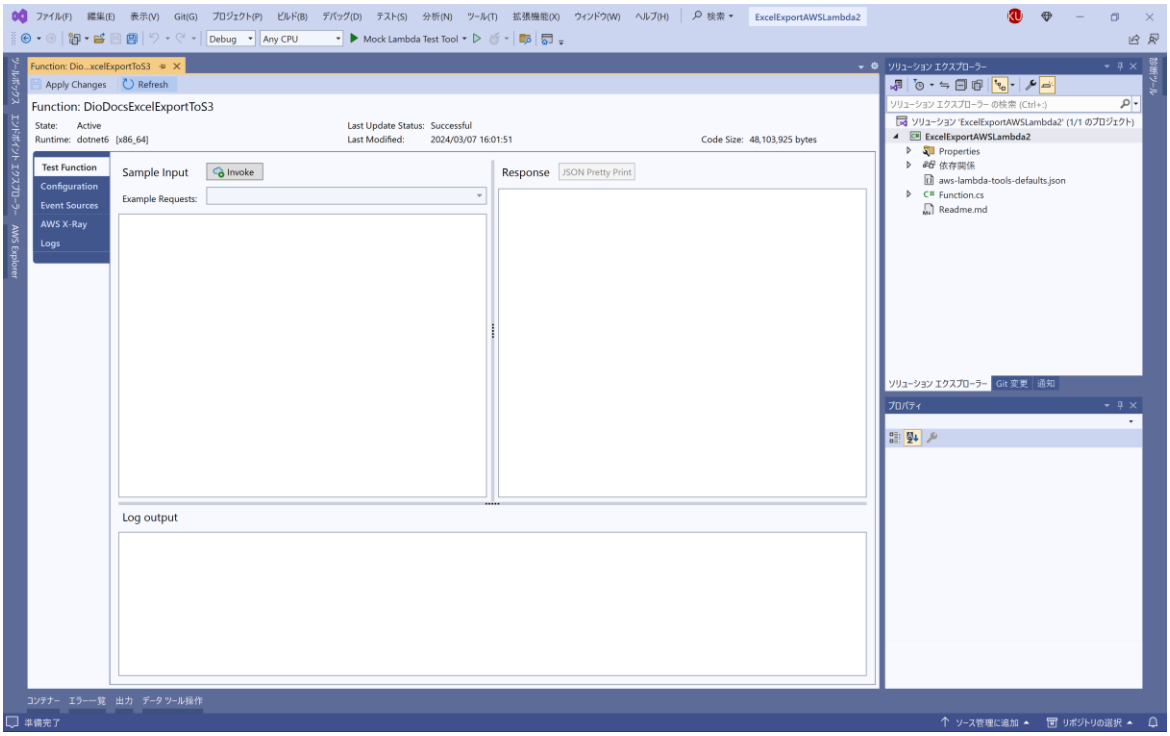
「Function Name」に`DioDocsExcelExportToS3`を入力して［Next］をクリックします。



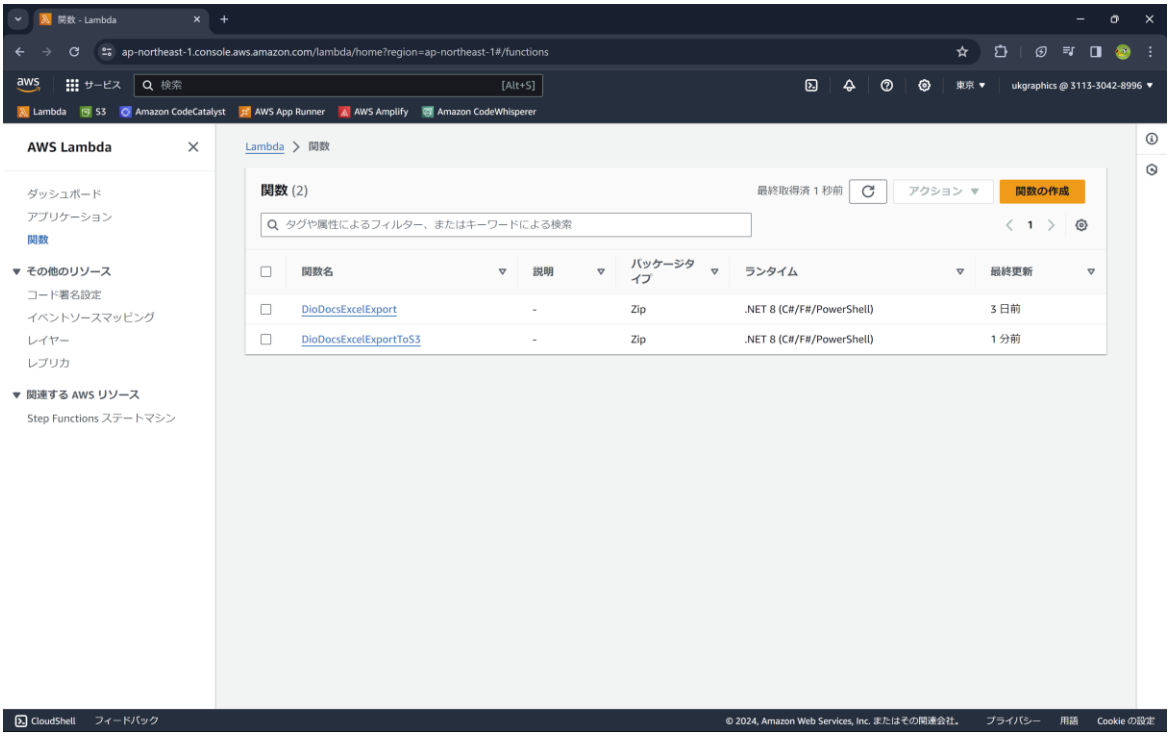
「Role Name」に`New role based on AWS managed policy: AWSLambdaExecute`を設定して［Upload］をクリックします。



成功すると以下の画面が表示されます。

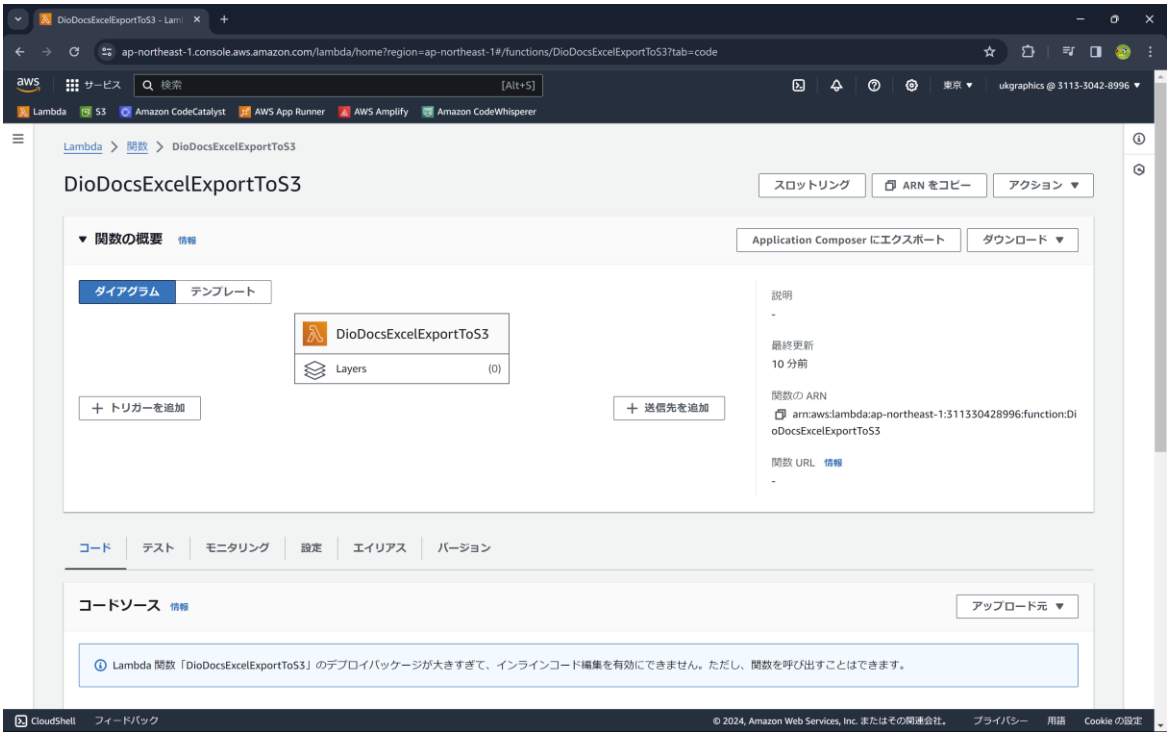


AWS のコンソールで AWS Lambda の「関数」を選択するとデプロイした Lambda 関数「DioDocsExcelExportToS3」が表示されます。

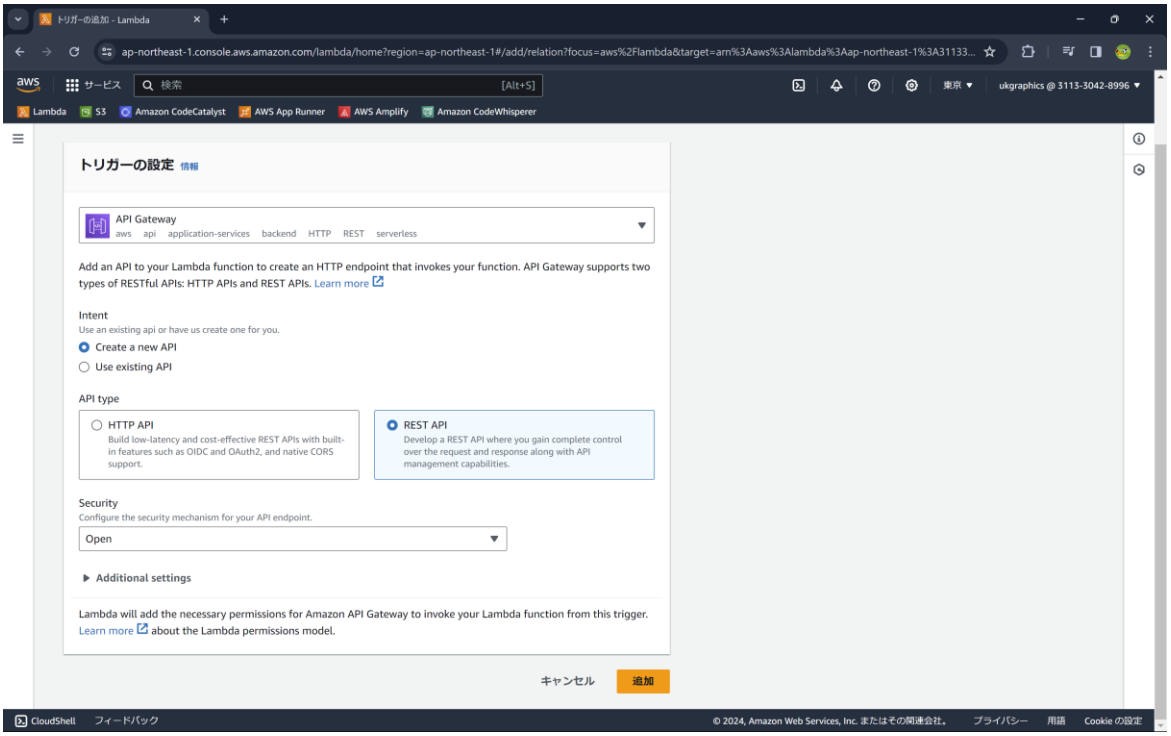


トリガーの追加

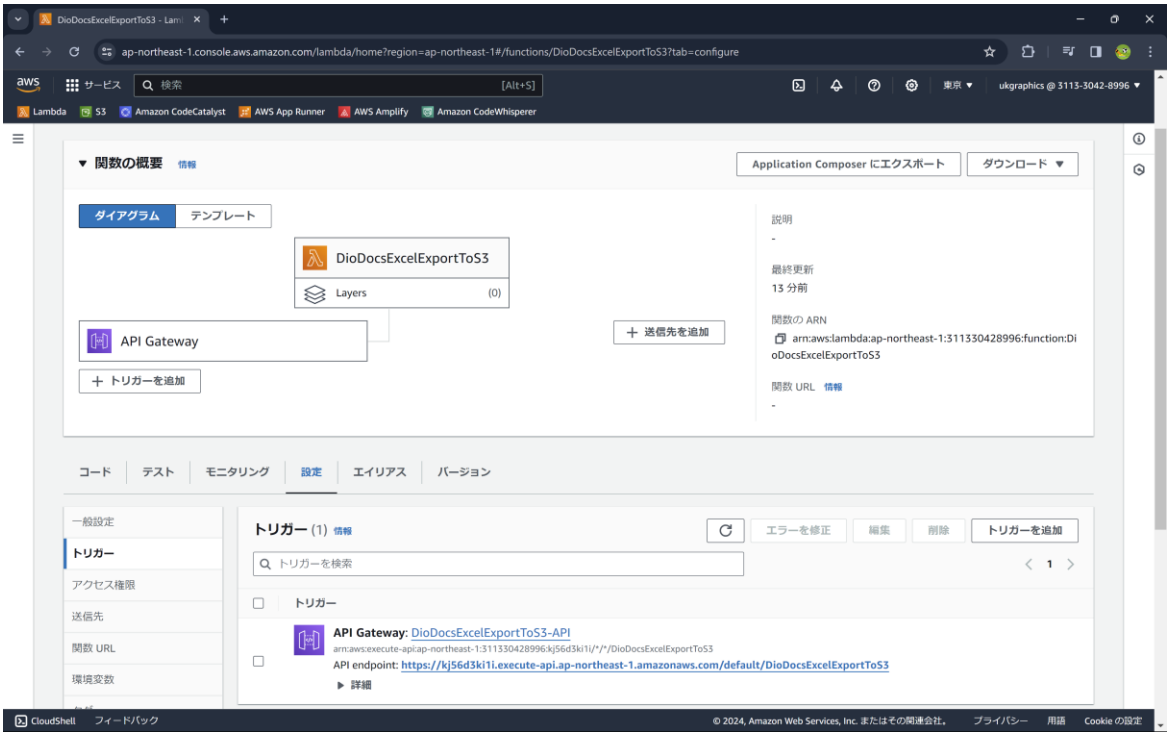
デプロイした Lambda 関数「DioDocsExcelExportToS3」をクリックして以下の画面から［トリガーを追加］をクリックします。



「API Gateway」を選択し、さらに「Create a new API」を選択します。作成する API タイプは「REST API」を選択して、セキュリティは「Open」を選択します。この状態で［追加］をクリックします。

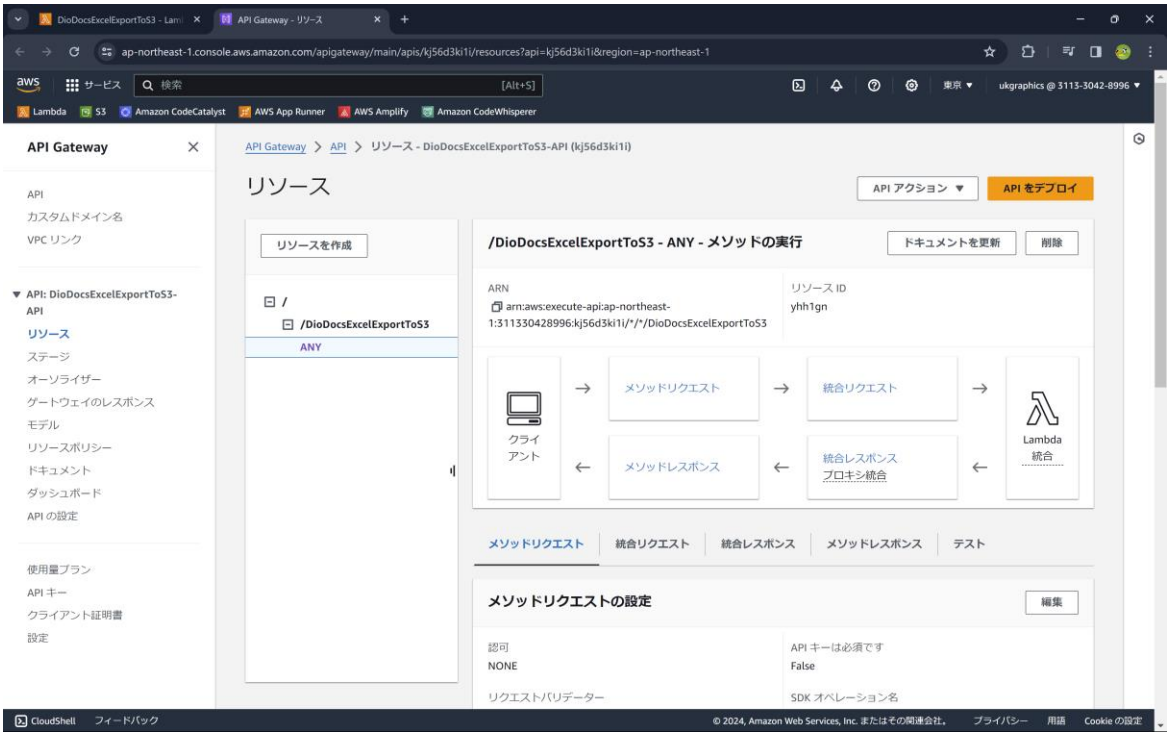


以下のようにトリガーに API Gateway が追加されます。

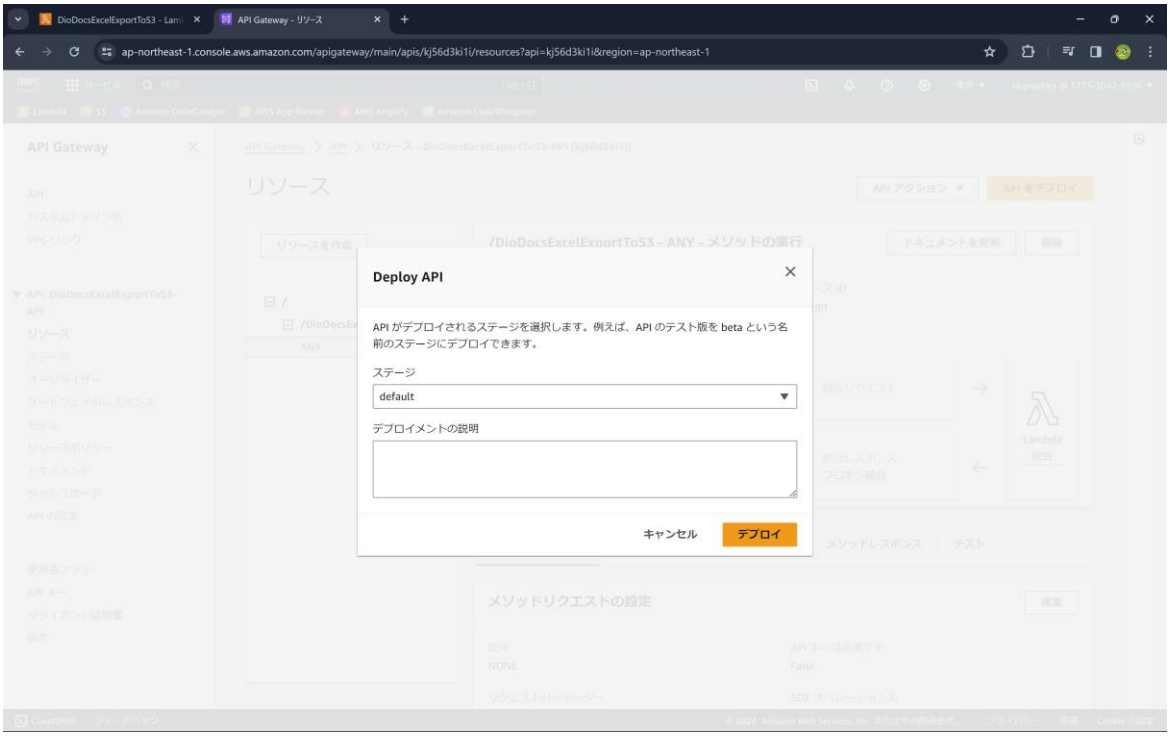


デプロイしたアプリケーションを確認

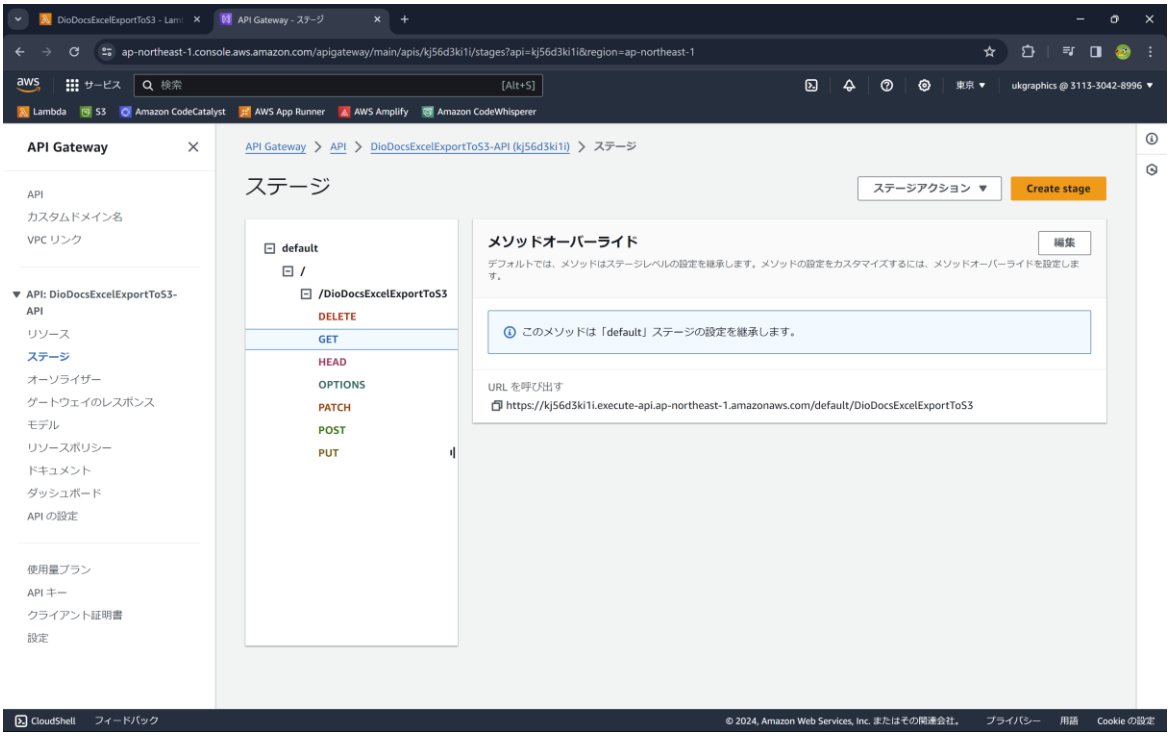
AWS のコンソールで作成した API「DioDocsExcelExportToS3-API」の「リソース」から「API のデプロイ」を選択します。



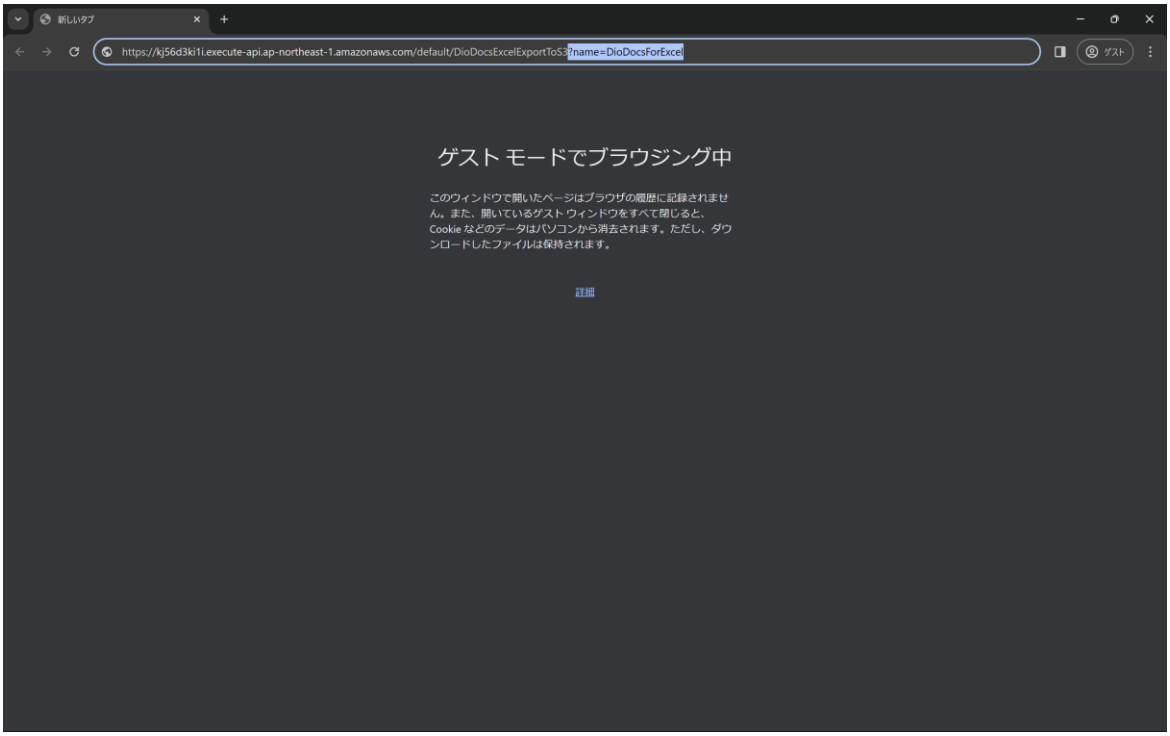
デプロイされるステージは「default」を選択して［デプロイ］をクリックします。



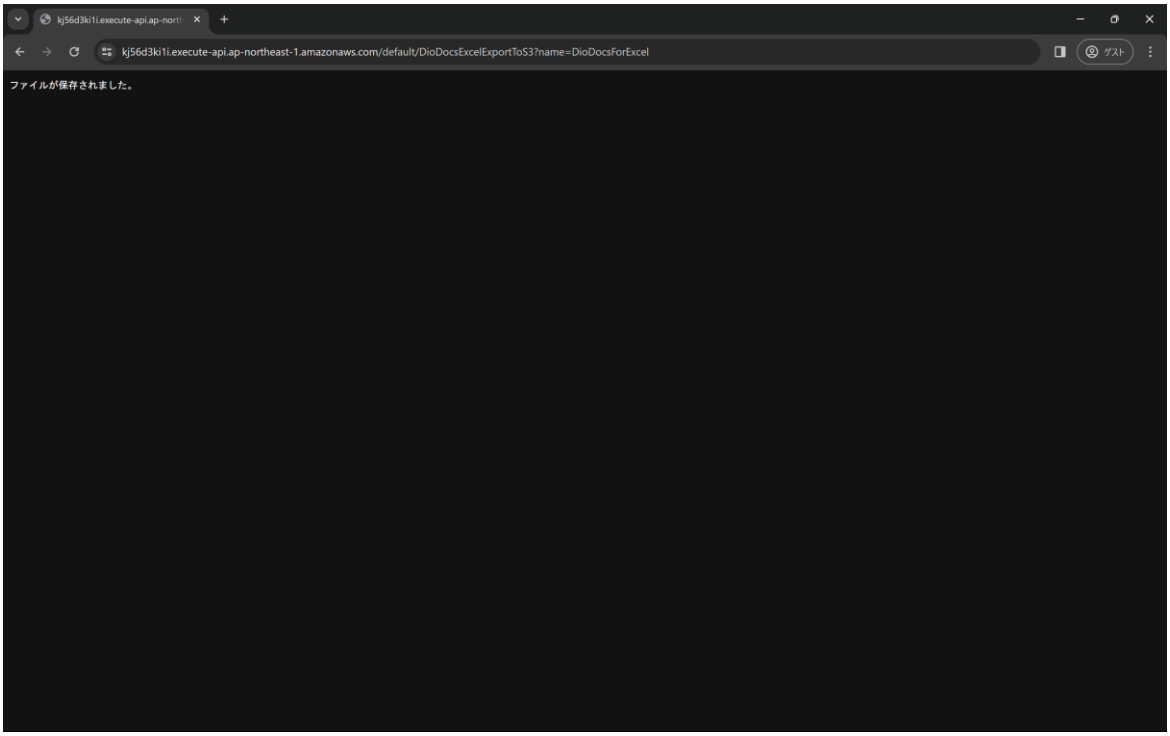
API「DioDocsExcelExportToS3-API」の「ステージ」から「GET」 を選択します。

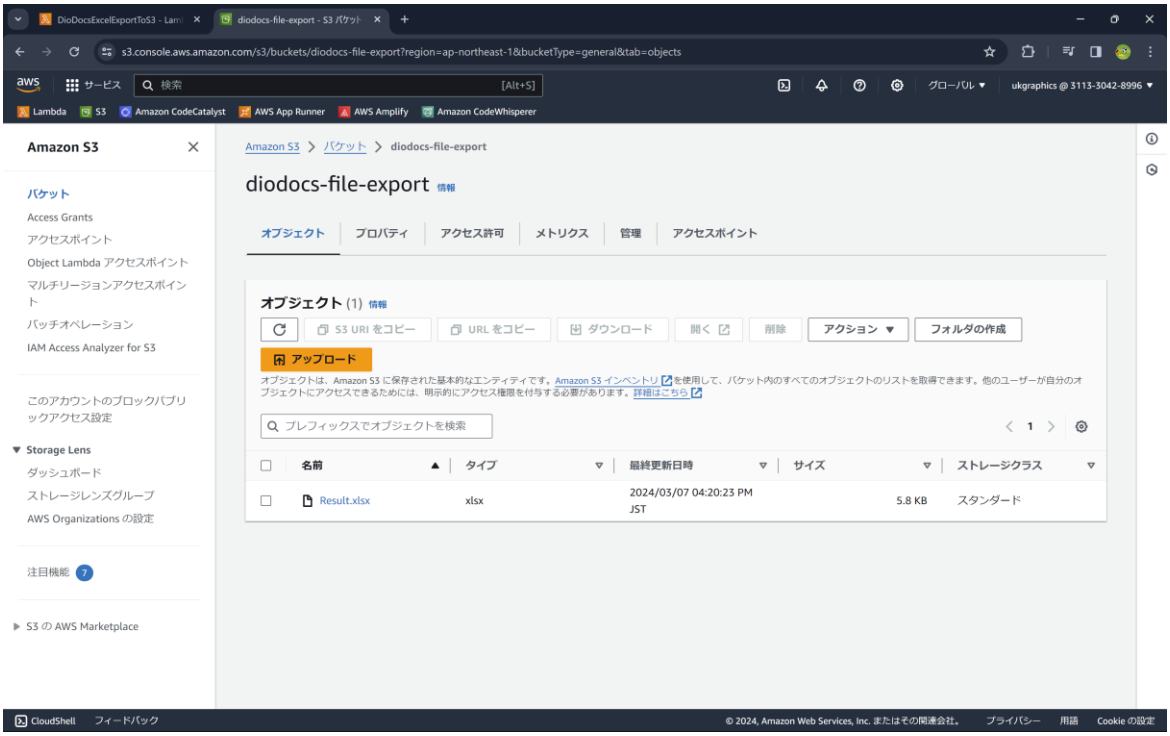


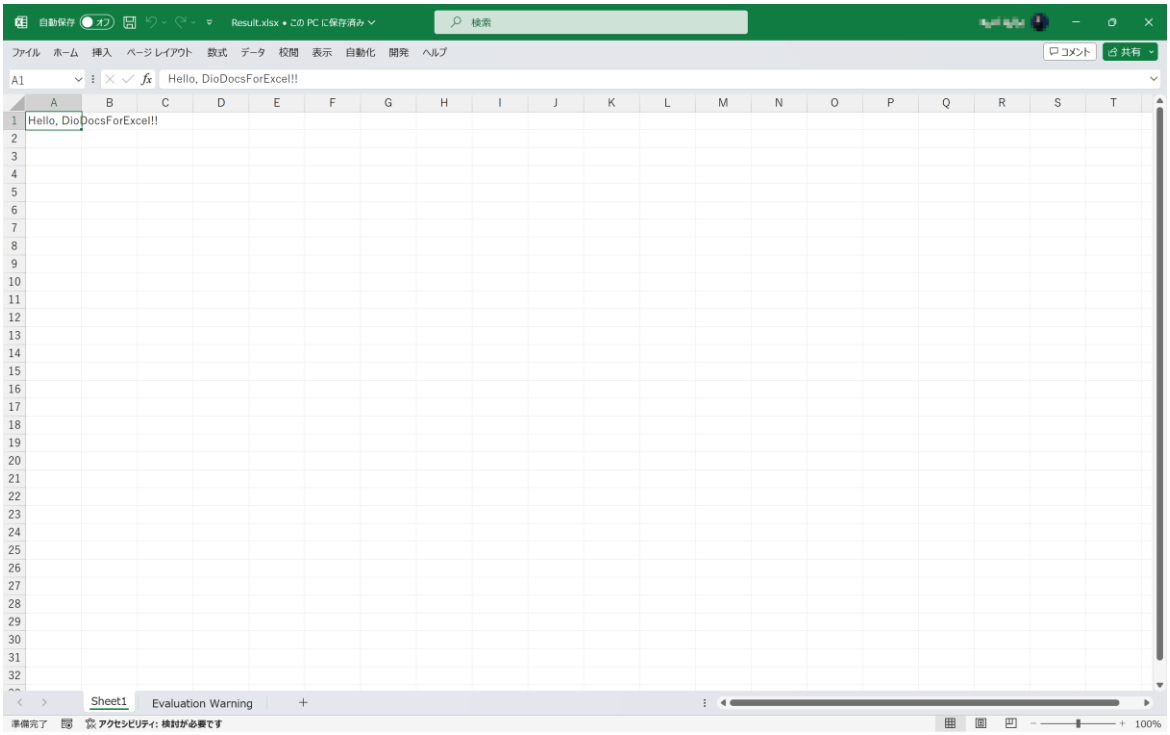
「URL の呼び出し」に表示されている API の URL をコピーしてブラウザに張り付けて、さらにクエリパラメータと文字列「`?name=DioDocsForExcel`」を追加します。



この API を実行するとクエリパラメータで渡した文字列「DioDocsForExcel」が追加された Excel ファイル「Result.xlsx」が Amazon S3 のバケット「diodocs-file-export」に出力されます。


kj56d3ki1i.execute-api.ap-northeast-1.amazonaws.com/default/DioDocsExcelExportToS3?name=DioDocsForExcel
ファイルが保存されました。


s3.console.aws.amazon.com/s3/buckets/diodocs-file-export?region=ap-northeast-1&bucketType=general&tab=objects
Amazon S3
バケット
Access Grants
アクセスポイント
Object Lambda アクセスポイント
マルチリージョンアクセスポイント
バッチオペレーション
IAM Access Analyzer for S3
このアカウントのブロックパブリックアクセス設定
Storage Lens
ダッシュボード
ストレージレンズグループ
AWS Organizations の設定
注目機能
S3 の AWS Marketplace
Amazon S3 > バケット > diodocs-file-export
diodocs-file-export 情報
オブジェクト
プロパティ
アクセス許可
メトリクス
管理
アクセスポイント
オブジェクト (1) 情報
S3 URI をコピー
URL をコピー
ダウンロード
開く
削除
アクション
フォルダの作成
アップロード
プレフィックスでオブジェクトを検索
名前
タイプ
最終更新日時
サイズ
ストレージクラス
Result.xlsx
xlsx
2024/03/07 04:20:23 PM JST
5.8 KB
スタンダード
CloudShell
フィードバック
© 2024, Amazon Web Services, Inc. またはその関連会社。
プライバシー
用語
Cookie の設定



PDF を出力するには？

Visual Studio の「NuGet パッケージ マネージャー」から DioDocs for PDF のパッケージ「`DioDocs.Pdf.ja`」をインストールします。次に DioDocs for PDF で PDF ファイルを作成するコードを追加して `FunctionHandler` を以下のように更新します。

```
public async Task<APIGatewayProxyResponse> FunctionHandler(APIGatewayProxyRequest input, ILambdaContext context)
{
    APIGatewayProxyResponse response;

    try
    {
        // クエリ文字列を取得
        string? queryString;
        input.QueryStringParameters.TryGetValue("name", out queryString);

        // ドキュメントに追加するテキスト
        string Message = string.IsNullOrEmpty(queryString)
            ? "Hello, World!!"
            : $"Hello, {queryString}!!";

        //GcPdfDocument.SetLicenseKey("製品版またはトライアル版のライセンスキーを設定");

        GcPdfDocument doc = new GcPdfDocument();
```

```
        GcPdfGraphics g = doc.NewPage().Graphics;

        g.DrawString(Message,
            new TextFormat() { Font = StandardFonts.Helvetica, FontSize = 12 },
            new PointF(72, 72));

        using (var ms = new MemoryStream())
        {
            doc.Save(ms, false);

            // S3 にアップロード
            AmazonS3Client client = new AmazonS3Client(RegionEndpoint.APNortheast1);
            var request = new PutObjectRequest
            {
                BucketName = "diodocs-export",
                Key = "Result.pdf",
                InputStream = ms
            };

            await client.PutObjectAsync(request);
        }

        response = new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.OK,
            Body = "ファイルが保存されました。",
            Headers = new Dictionary<string, string> {
                { "Content-Type", "text/plain; charset=utf-8" }
            }
        };
    }
    catch (Exception e)
    {
        response = new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.InternalServerError,
            Body = e.Message,
            Headers = new Dictionary<string, string> {
                { "Content-Type", "text/plain" }
            }
        };
    }
```

```
    return response;
}
```

さいごに

動作を確認できる AWS Lambda アプリケーションのサンプルはこちらです。
https://github.com/MESCIUSJP/ExcelExportAWSLambda2
https://github.com/MESCIUSJP/PDFExportAWSLambda2

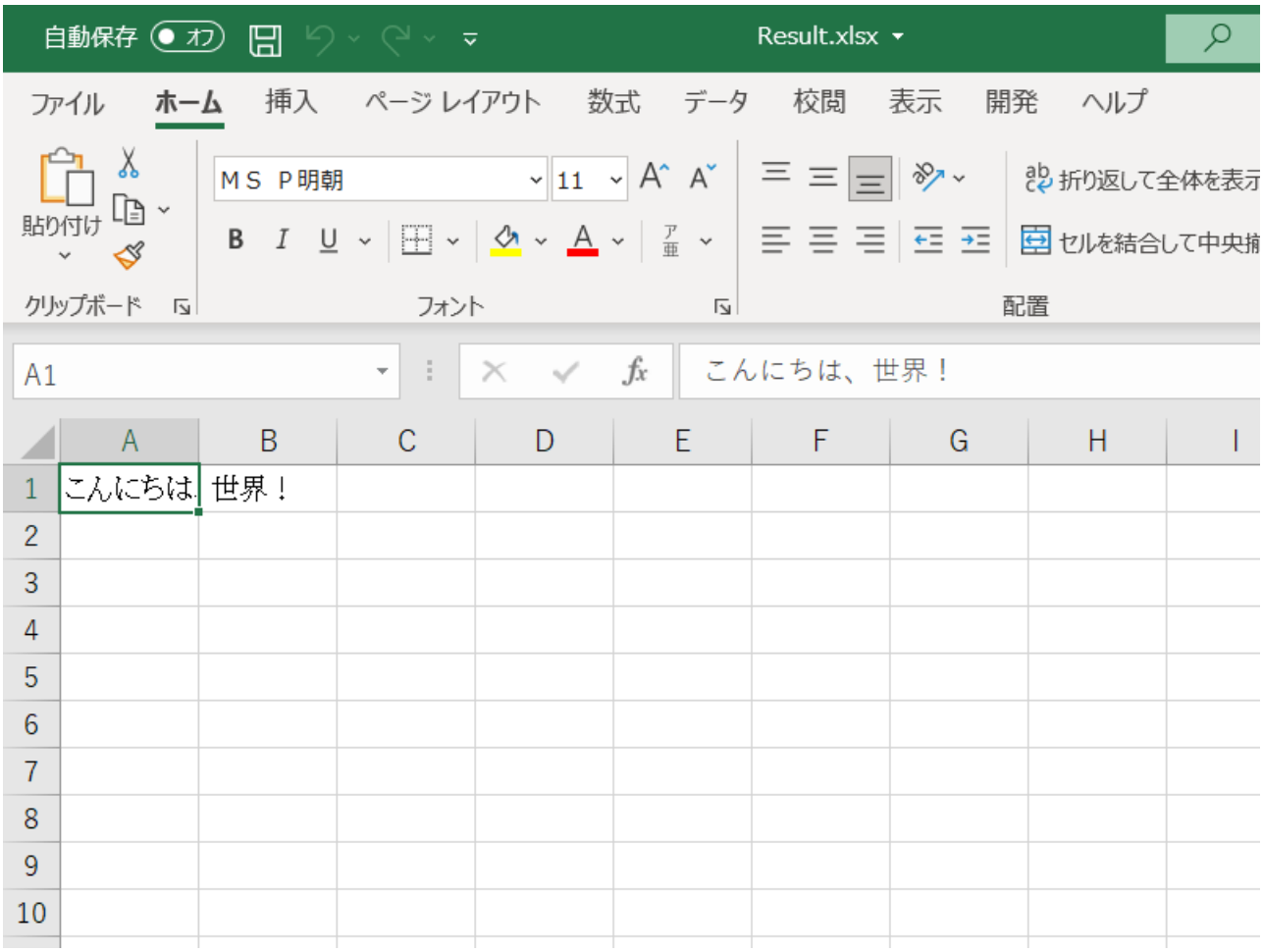
AWS Lambda と DioDocs で Excel や PDF ファイルを出力する（3）

前回と前々回の記事では AWS Lambda で「DioDocs（ディオドック）」を使用した C#（.NET 8）の Lambda 関数アプリケーションを作成し、Excel や PDF ファイルを出力する方法について紹介しました。今回は AWS Lambda で DioDocs を利用する際に、日本語フォントを使用する Tips を紹介します。

セルに追加するテキストの日本語フォント（DioDocs for Excel）

セルに追加するテキストの日本語フォントを設定したい場合は、`Font` プロパティを使用します。

```
Workbook workbook = new Workbook();
workbook.Worksheets[0].Range["A1"].Font.Name = "ＭＳ Ｐ明朝";
```



セルではなくシート全体のフォントを設定したい場合はこちらのナレッジベースを参考にしてください。
「シート全体のフォントを設定する方法」を見る

ワークシートを PDF 出力する際の日本語フォント（DioDocs for Excel）

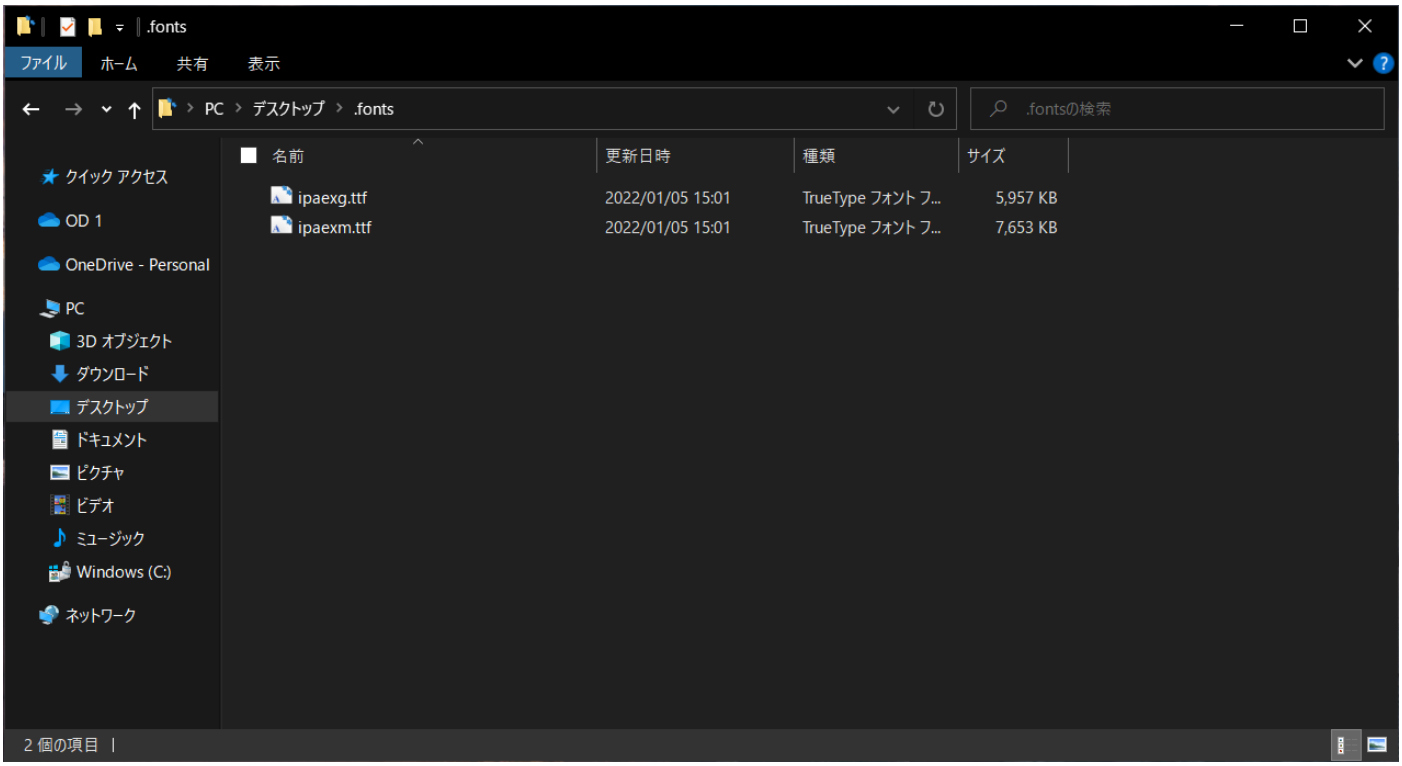
.NET 8 のランタイムが含まれる Lambda 関数の実行環境 OS は、「Amazon Linux 2023」になっています。Windows OS とは違って Amazon Linux 2023 には日本語フォントは含まれていませんので、デフォルトの状態で PDF 出力を実行すると文字化けが発生する、または文字列そのものが表示されない現象が発生します。

参考：「Linux 環境で PDF エクスポートすると文字化けが発生する」を見る

そこで AWS Lambda で DioDocs を使用する場合は、Lambda レイヤーを使用して日本語フォントを追加する必要があります。

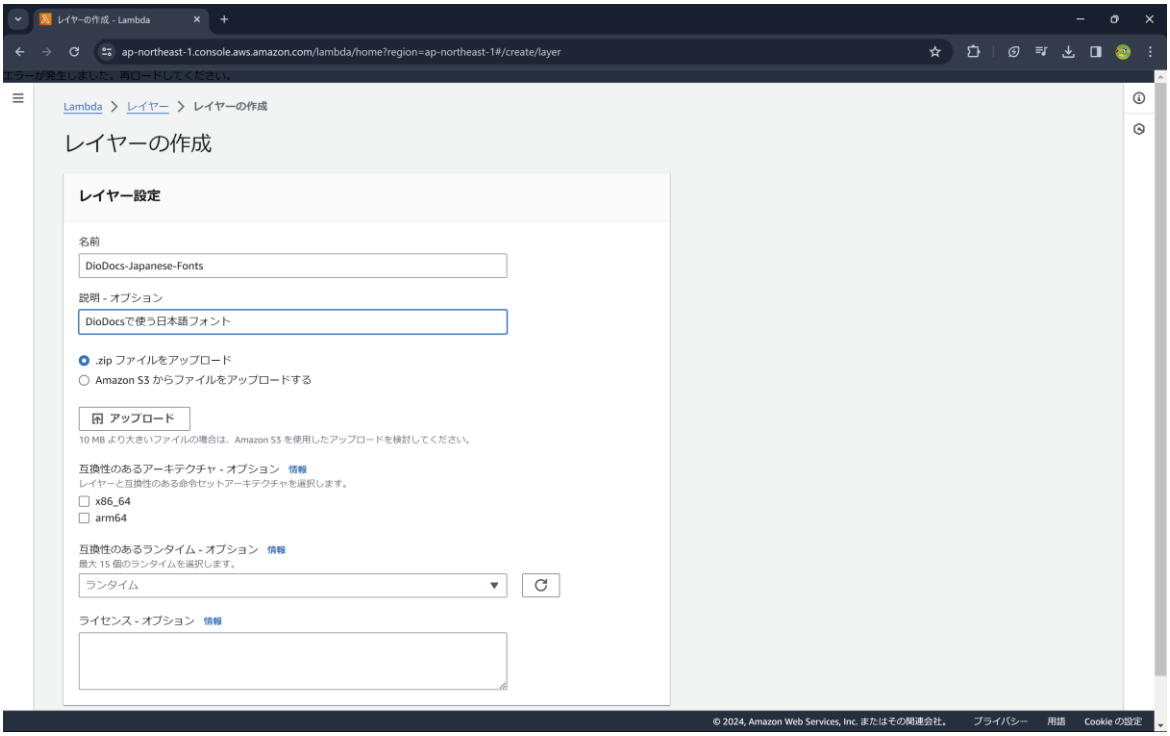
参考：「レイヤーの使用方法」を見る

まずローカル環境の適当な場所に「.fonts」フォルダを作成し、そこにフォントファイルを格納後、「.fonts」フォルダを ZIP ファイルへ圧縮します。本記事では「IPAex フォント」を「.fonts」フォルダにコピーしています。

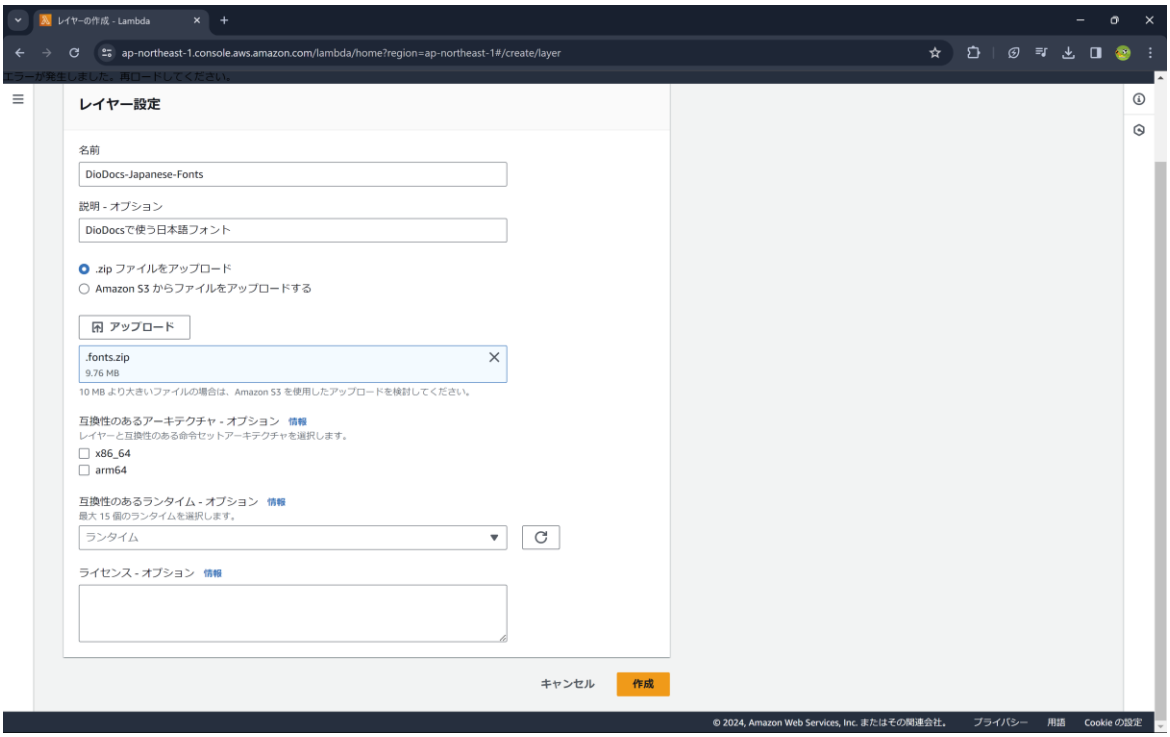


AWS のコンソールで AWS Lambda の「レイヤー」を選択し、［レイヤーの作成］をクリックします。

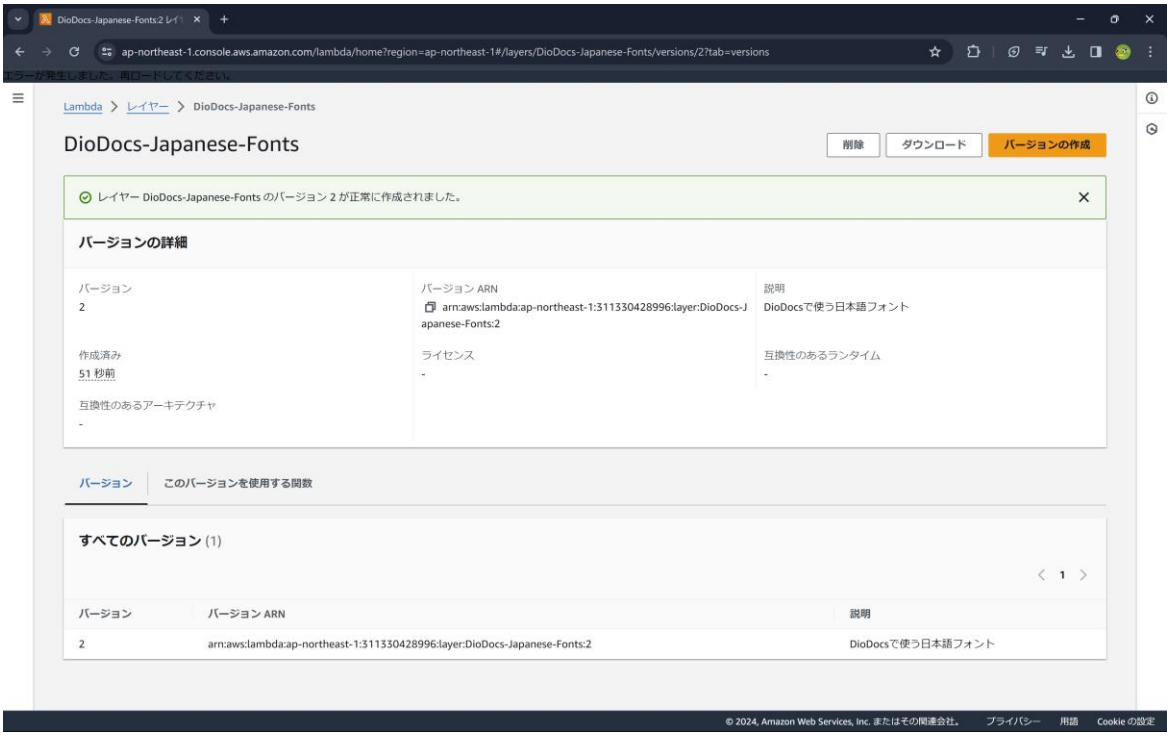
レイヤーの名前に「DioDocs-Japanese-Fonts」を、説明に「DioDocs で使う日本語フォント」を設定して［アップロード］をクリックします。



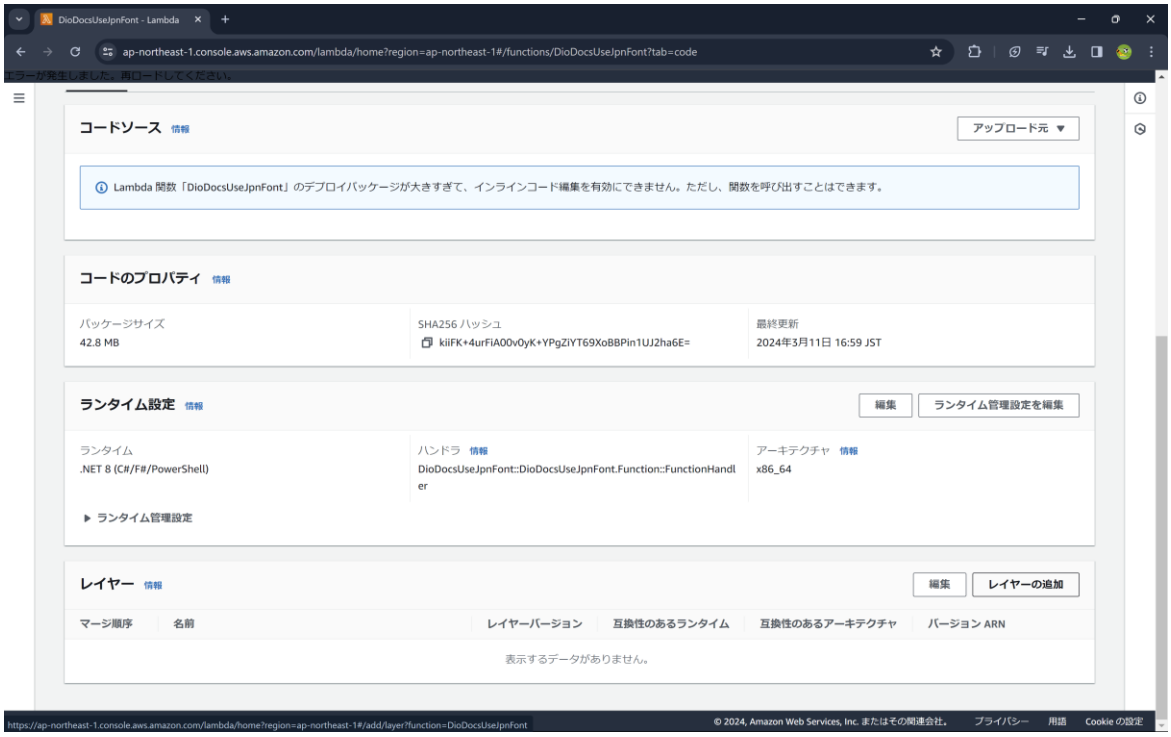
先ほどローカル環境で作成した ZIP ファイル「.fonts.zip」を選択して［作成］ボタンをクリックします。



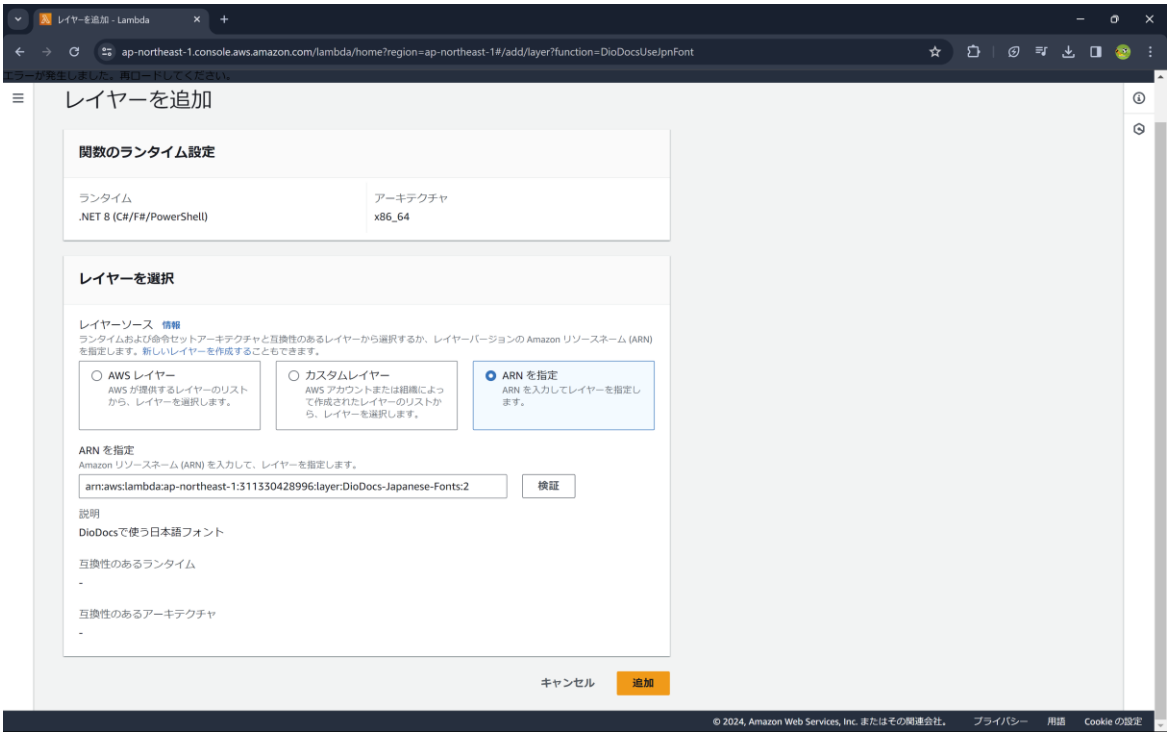
Lambda レイヤーが作成されます。「バージョン ARN」は、デプロイした Lambda 関数から Lambda レイヤーを追加する際に使用するのでコピーしておきます。



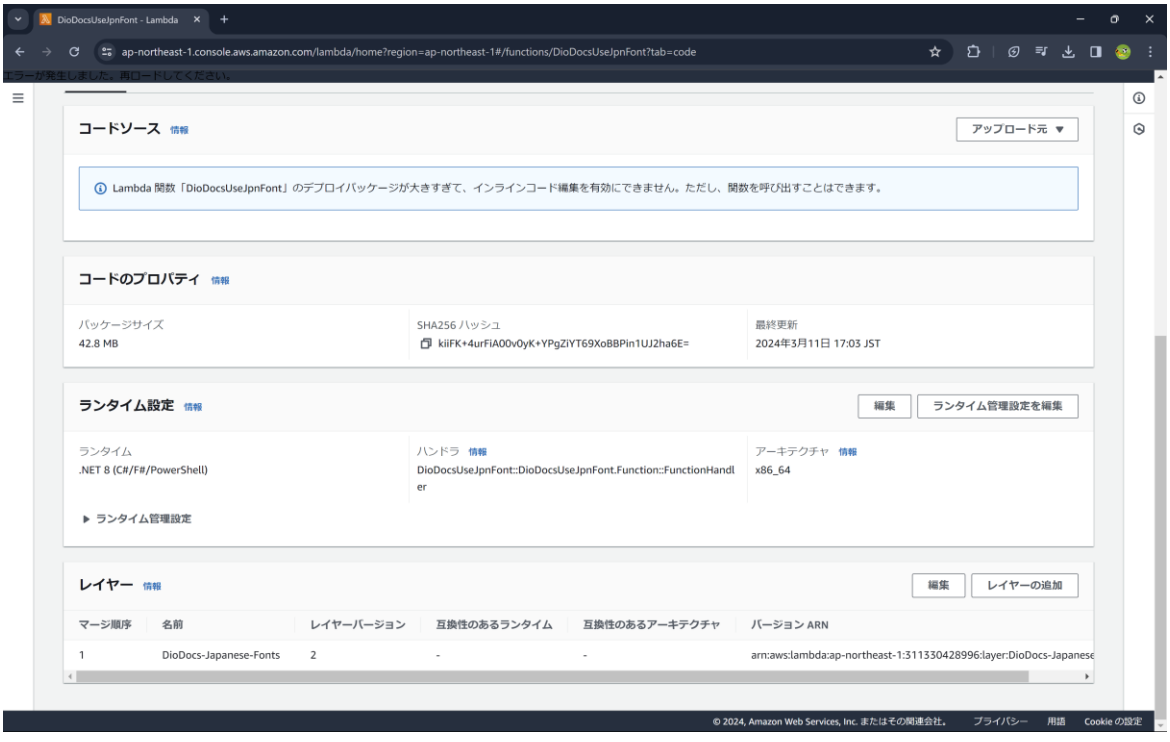
AWS コンソールから Lambda の［関数］をクリックし、デプロイした関数名をクリックします。表示された詳細画面の下部にある［レイヤーの追加］をクリックします。



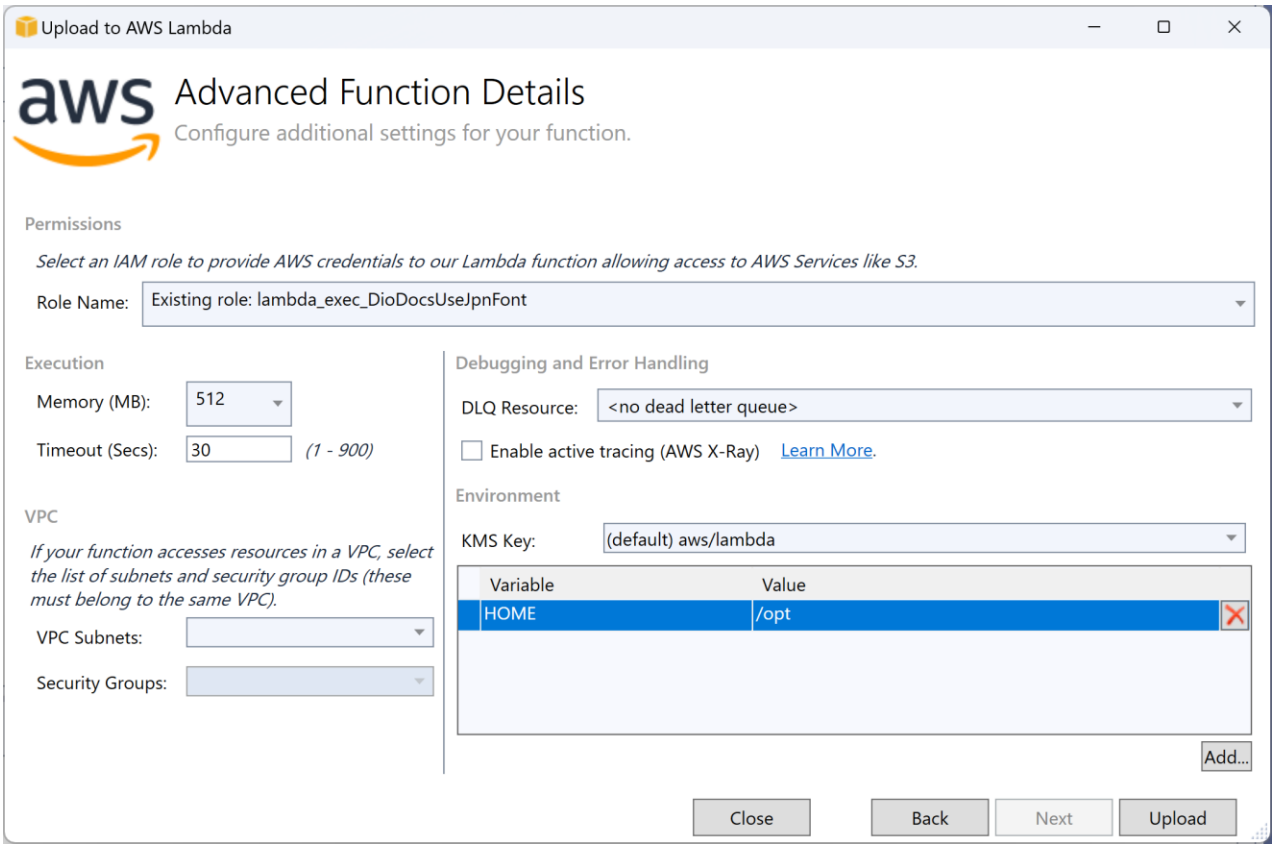
「レイヤーを追加」画面で「ARN を指定」を選択し、「ARN を指定」欄に先ほどコピーした Lambda レイヤーの ARN を入力して、［追加］をクリックします。



Lambda 関数に Lambda レイヤーが追加されます。



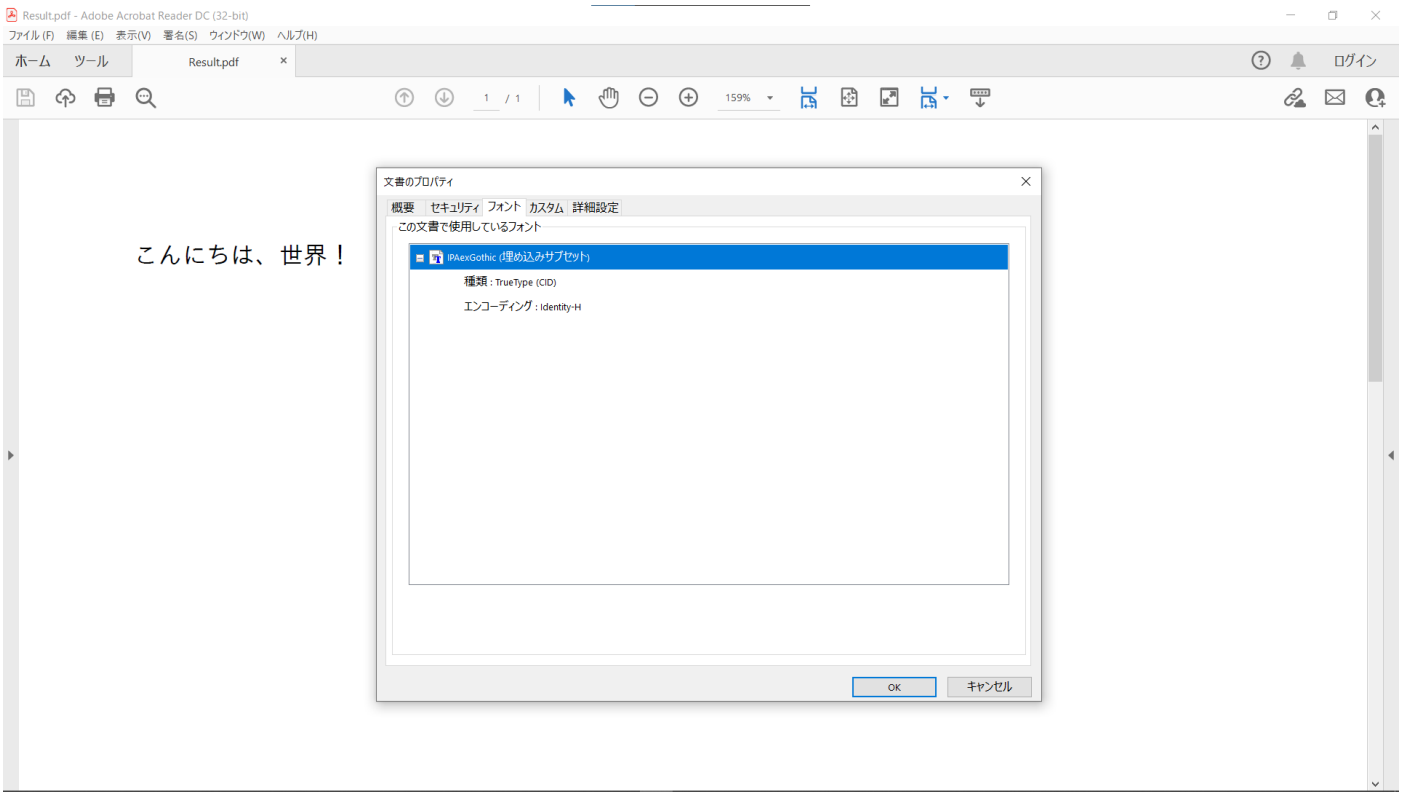
Visual Studio から Lambda 関数をデプロイする際に、以下の画面で［Add…］をクリックして環境変数を追加します。変数名は`HOME`で値は`/opt`を設定します。`/opt`は Lambda レイヤーが展開されるディレクトリです。この設定により Lambda 関数を実行するパス（ホームディレクトリ）が`/opt`になるので、Lambda レイヤーに格納した`.fonts`配下に含まれる日本語フォント「IPAex ゴシック」が利用できます。



この Lambda レイヤーで日本語フォントを追加した Lambda 関数で、以下のコードのように「IPAex ゴシック」を設定したセルに日本語の文字列を持つ Excel ワークブックを PDF ファイルへ出力すると、日本語が文字化けしたりすることなく「IPAex ゴシック」が設定されて正しく文字列が表示されていることが確認できます。

```
public APIGatewayProxyResponse FunctionHandler(APIGatewayProxyRequest input, ILambdaContext context)
{
    APIGatewayProxyResponse response;

    string? queryString;
    input.QueryStringParameters.TryGetValue("name", out queryString);

    string Message = string.IsNullOrEmpty(queryString)
        ? "こんにちは、世界！"
        : $"こんにちは、{queryString}！";

    Workbook workbook = new Workbook();

    workbook.Worksheets[0].Range["A1"].Font.Name = "IPAex ゴシック";

    workbook.Worksheets[0].Range["A1"].Value = Message;

    var base64String = "";
```

```
    using (var ms = new MemoryStream())
    {
        workbook.Save(ms, SaveFileFormat.Pdf);
        base64String = Convert.ToBase64String(ms.ToArray());
    }

    response = new APIGatewayProxyResponse
    {
        StatusCode = (int)HttpStatusCode.OK,
        Body = base64String,
        IsBase64Encoded = true,
        Headers = new Dictionary<string, string> {
            {"Content-Type", "application/pdf"},
            {"Content-Disposition", "attachment; filename=Result.pdf"},
        }
    };

    return response;
}
```



PDF ドキュメントを保存する際の日本語フォント（DioDocs for PDF）

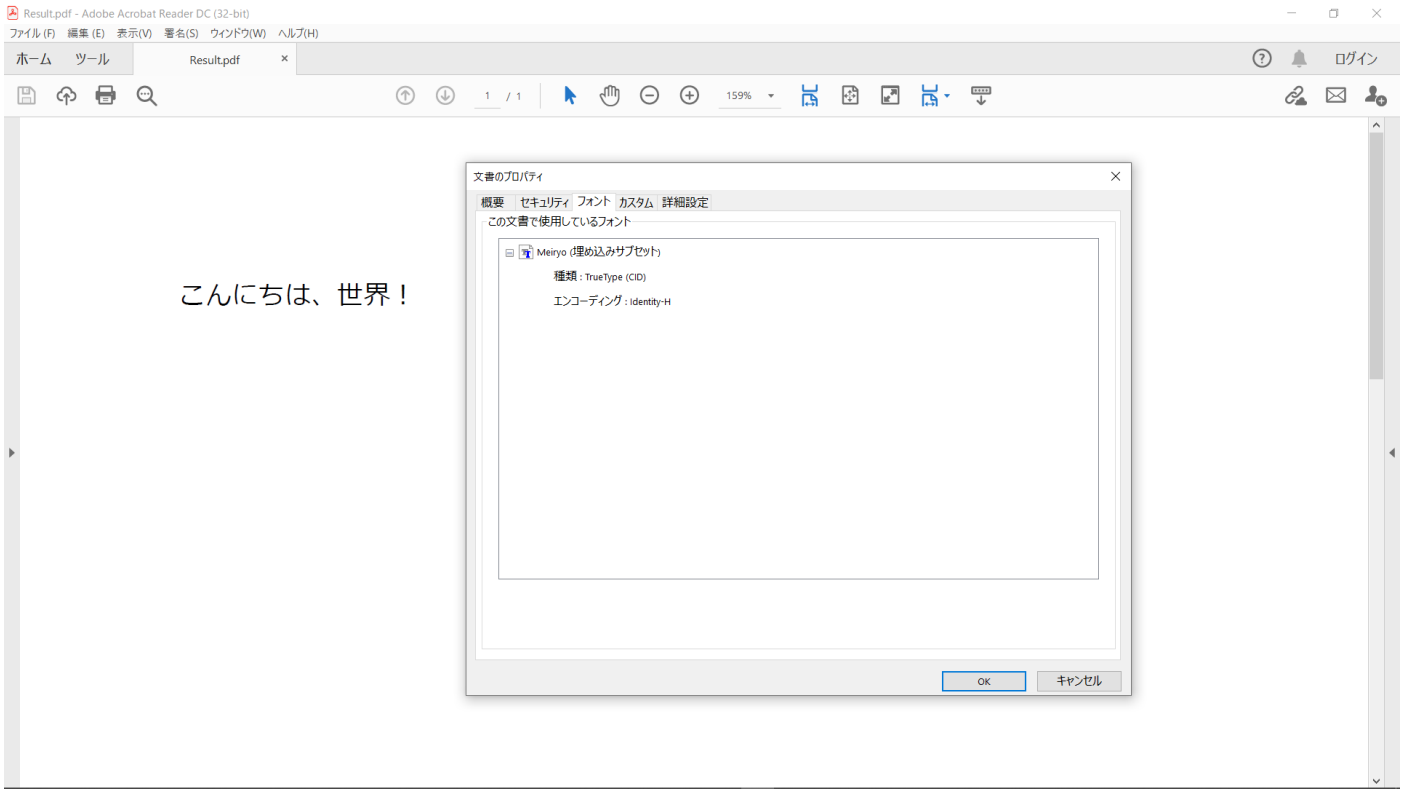
DioDocs for PDF で作成した PDF ドキュメントで日本語フォントを利用する場合も、「作成したワークシートを PDF 出力する際の日本語フォントを設定する（DioDocs for Excel）」と同じ手順で Lambda レイヤーを使用して

日本語フォントを追加します。

以下のコードのように「IPAex ゴシック」を設定した日本語の文字列を持つ PDF ドキュメントを出力すると、日本語が文字化けしたりすることなく「IPAex ゴシック」が設定されて正しく文字列が表示されていることが確認できます。

```
public APIGatewayProxyResponse FunctionHandler(APIGatewayProxyRequest input, ILambdaContext context)
{
    APIGatewayProxyResponse response;

    string? queryString;
    input.QueryStringParameters.TryGetValue("name", out queryString);

    string Message = string.IsNullOrEmpty(queryString)
    ? "こんにちは、世界！"
    : $"こんにちは、{queryString}！";

    GcPdfDocument doc = new GcPdfDocument();
    GcPdfGraphics g = doc.NewPage().Graphics;

    g.DrawString(Message,
        new TextFormat() { FontName = "IPAex ゴシック", FontSize = 12 },
        new PointF(72, 72));

    var base64String = "";

    using (var ms = new MemoryStream())
    {
        doc.Save(ms, false);
        base64String = Convert.ToBase64String(ms.ToArray());
    }

    response = new APIGatewayProxyResponse
    {
        StatusCode = (int)HttpStatusCode.OK,
        Body = base64String,
        IsBase64Encoded = true,
        Headers = new Dictionary<string, string> {
            { "Content-Type", "application/pdf" },
            { "Content-Disposition", "attachment; filename=Result.pdf"},
        }
    };
```

```
    return response;
}
```



本記事では AWS Lambda で DioDocs を利用する際に、日本語フォントを使用する Tips を紹介しました。

弊社 Web サイトでは、製品の機能を気軽に試せるデモアプリケーションやトライアル版も公開していますので、こちらもご確認いただければと思います。

- デモアプリケーション（DioDocs for Excel）を試す
- デモアプリケーション（DioDocs for PDF）を試す
- トライアル版をダウンロードして試す

また、ご導入前の製品に関するご相談やご導入後の各種サービスに関するご質問など、お気軽にお問合せください。

- 問合せ先を確認する
- 個別相談会（Web 会議）について確認する