



# Microsoft Cloud Workshop

Serverless architecture

Hands-on lab step-by-step

February 2020

## Contents

- Serverless architecture hands-on lab step-by-step
  - 学習目標
  - シナリオ
  - ソリューション アーキテクチャー
  - 必要条件
  - Exercise 1: 環境のセットアップ
    - Task 1: リソース グループの作成
    - Task 2: ストレージ アカウントのプロビジョニング
    - Task 3: Function Apps のプロビジョニング
    - Task 4: Event Grid トピックのプロビジョニング
    - Task 5: Azure Cosmos DB アカウントのプロビジョニング
    - Task 6: Computer Vision API のプロビジョニング
    - Task 7: Azure Key Vault のプロビジョニング
  - Exercise 2: 写真の処理を行う Azure Functions の開発と発行
    - Task 1: アプリケーション設定の構成
    - Task 2: Key Vault へアクセスするためのシステム割り当てマネージド ID の生成
    - Task 3: Key Vault へのアクセス許可を付与するアクセス ポリシーの作成
    - Task 4: ProcessImage 関数の開発
    - Task 5: Visual Studio から Function App の公開
  - Exercise 3: Azure ポータルでの Function App の開発
    - Task 1: ライセンス プレート データを Cosmos DB に保存する関数の作成
    - Task 2: 関数への Event Grid サブスクリプションの追加
    - Task 3: 関数への Cosmos DB 出力の追加
    - Task 4: run.csx へのコードの記述
    - Task 5: 手動検証データを Cosmos DB へ保存する関数の作成
    - Task 6: 関数への Event Grid サブスクリプションの追加
    - Task 7: 関数への Cosmos DB 出力の追加
    - Task 8: run.csx へのコードの記述
  - Exercise 4: Application Insights による Azure Functions の監視
    - Task 1: Application Insights インスタンスのプロビジョニング
    - Task 2: Function App での Application Insights の有効化
    - Task 3: ライブ メトリックス ストリーム を使用したリアルタイム監視

- Task 4: Azure Functions の動的スケーリングの監視
- Exercise 5: Azure Cosmos DB 内のデータ探索
  - Task 1: Azure Cosmos DB データ エクスプローラーの使用
- Exercise 6: データ エクスポート ワークフローの作成
  - Task 1: Logic App の作成
- Exercise 7: Function App の CI/CD 構成
  - Task 1: Azure DevOps 組織とプロジェクトの作成
  - Task 2: プロジェクトへのユーザーの追加
  - Task 3: Azure Repos へのソース コードの追加
  - Task 4: Azure Pipelines の作成
  - Task 5: ブランチの作成と ExportLicensePlates 関数の変更
  - Task 6: プルリクエストの作成
  - Task 7: ブランチのマージと新バージョンの Azure への展開
- Exercise 8: ワークフローの実行とエクスポート データの確認
  - Task 1: Logic App の実行
  - Task 2: エクスポートされた CSV ファイルの表示
- ワークショップの終了
  - Task 1: リソース グループの削除
  - Task 2: Azure DevOps プロジェクトの削除

## Serverless architecture hands-on lab step-by-step

### 学習目標

このワークショップでは、Microsoft Azure Functions, Cosmos DB, EventGrid および関連サービスに基づく提供されたサンプルを使用し、エンド ツー エンドのシナリオを実装することを課題としています。このシナリオでは、コンピューティング、ストレージ、ワークフロー、および監視など Microsoft Azure の様々なコンポーネントをします。ワークショップは一人で実装できますが、他のメンバーとペアを組んで、各メンバーがソリューション全体の専門知識を共有できるようにすることをお勧めします。

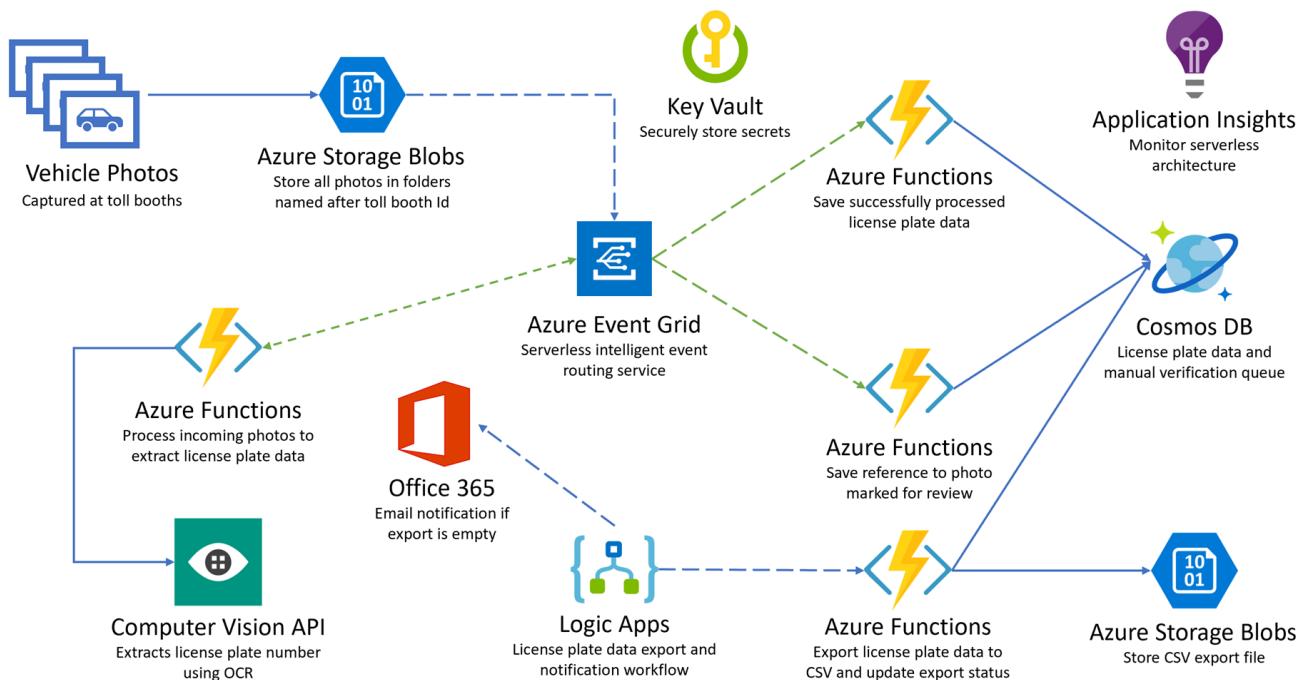
ワークショップを最後まで実施することで、回復力、拡張性、およびコスト効率の高いサーバーレス ソリューションの設計、開発、監視を行うことができるようになります。

### シナリオ

Contoso 社では、広いエリアで事業展開するために、料金所管理事業を拡大しています。これはオンライン決済サービスである彼らの主要ビジネスではないため、ストレージにアップロードされた車両の写真を使用して、多数の新しい料金所からナンバープレート情報を抽出するという今後の需要を満たすためのスケールアップに課題を抱えています。現在はこれらを手動で行うサードパーティーにバッチ処理で送信し、ナンバープレート情報を CSV ファイルへ抽出、Contoso 社に送り返してもらいオンライン処理システムにアップロードします。コスト効率と拡張性を備えた方法で、このプロセスを自動化したいと考えています。

### ソリューション アーキテクチャ

このワークショップで構築するソリューション アーキテクチャの図を次に示します。



このソリューションは、車両の写真が Azure Storage Blob コンテナーにアップロードされたことを検知した時点から始まります。Blob Storage の作成イベントに対して Event Grid サブスクリプションが作成され、写真を処理する **Azure Functions** が呼び出され、写真が **Cognitive Services Computer Vision API OCR** サービスに送信されナンバープレート番号が抽出されます。処理が成功し、ナンバープレート番号が返された場合、この Azure Functions は、新しい Event Grid イベントとデータを "savePlateData" というイベントタイプで Event Grid のトピックに送信します。処理が失敗した場合は、"queuePlateForManualCheckup" というイベントタイプで Event Grid イベントをトピックに送信します。新しいイベントが Event Grid トピックに追加されたとき起動する 2 つの Azure Functions が構成され、各イベントタイプでのフィルター処理を行います。15 分間隔で実行される **Logic Apps** は、Cosmos DB から新しいナンバープレート番号を取得し、番号を CSV ファイルへエクスポートした後 Blob Storage に保存する Azure Functions を HTTP トリガーを使用して起動します。新しいナンバープレート番号が見つからなかった場合、Logic Apps は、電子メール通知を送信します。

**Application Insights** は、データがサーバーレス アーキテクチャを通じて処理される際に、すべての Azure Functions をリアルタイムで監視するために使用します。このリアルタイム監視では、動的スケーリングを監視し、特定のイベントが発生した場合にアラートを構成できます。

**Azure Key Vault** は、接続文字列や API へのアクセスキーなどのシークレットを安全に格納するために使用します。Key Vault では Azure Functions に割り当てられたマネージド ID に対してポリシーを通じてアクセス許可を付与します。

## 必要条件

- Microsoft Azure サブスクリプション
- ノート PC
  - Visual Studio 2019 (Community or Professional or Enterprise)
    - <https://www.visualstudio.com/vs/>
  - Azure 開発ワークフロー
    - <https://docs.microsoft.com/azure/azure-functions/functions-develop-vs#prerequisites>
  - .NET Framework 4.7 以上のバージョンのランタイム
  - .NET Core 2.1 以上

- メール アカウント(Office 365 or Gmail or Outlook.com)

## Exercise 1: 環境のセットアップ<sup>°</sup>

所要時間 : 30分

ソリューション開発を行う前に Azure でいくつかのリソースをプロビジョニングします。

クリーンアップを容易に行うために、すべてのリソースが同じリソース グループを使用するようにします。

この演習では、ホット層を使用して Blob ストレージ アカウントをプロビジョニングし、アップロードされた写真と出力された CSV ファイルを保存するための 2 つのコンテナー (images, export) を作成します。次に 2 つの Function Apps インスタンスをプロビジョニングします。1 つは Visual Studio からデプロイし、もう 1 つは Azure ポータルを使用して管理します。そして新しい Event Grid トピックを作成します。その後、2 つのコレクション (Processed, NeedsManualReview) を持つ Azure Cosmos DB アカウントを作成します。最後に、ナンバープレートにオブジェクト文字認識 (OCR) を適用するための新しい Cognitive Services Computer Vision API サービスをプロビジョニングします。

Application Insights は、後の手順で追加しますので、Function Apps プロビジョニング時の Application Insights の設定は "無効" にしておいてください。

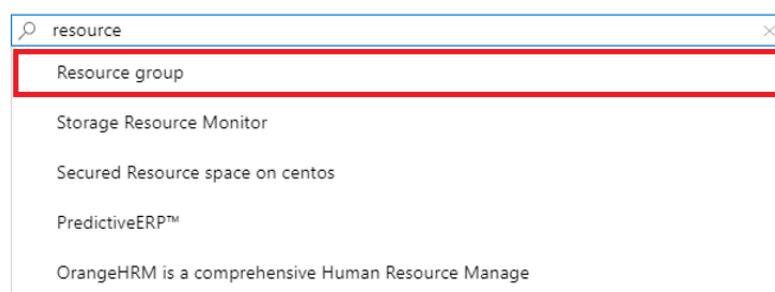
### 参考情報

- ストレージ アカウントの作成 (blob ホット層)  
<https://docs.microsoft.com/azure/storage/common/storage-create-storage-account?toc=%2Fazure%2Fstorage%2Fblobs%2Ftoc.json%23create-a-storage-account>
- Function App の作成  
<https://docs.microsoft.com/azure/azure-functions/functions-create-function-app-portal>
- Event Grid の概念  
<https://docs.microsoft.com/azure/event-grid/concepts>
- Cosmos アカウントの管理  
<https://docs.microsoft.com/azure/cosmos-db/manage-account>

### Task 1: リソース グループの作成

- Web ブラウザーの新しいタブまたはインスタンスを起動し、Azure ポータル (<https://portal.azure.com>) を開く
- 「+リソースの作成」をクリックし、画面上部の検索ボックスに **resource** と入力  
表示される候補から **Resource group** を選択

新規



- リソース グループの作成ブレードで「作成」ボタンをクリック

#### 4. リソース グループ作成の「基本」ブレード内で、次の構成オプションを指定

- リソース グループ名（任意、サブスクリプション内で一意）
- リージョン（任意、このワークショップで使用する地域）

**基本 タグ 確認および作成**

リソース グループ - Azure ソリューションの関連リソースを保持するコンテナー。リソース グループには、ソリューションのすべてのリソースを含めることも、グループとして管理したいリソースのみを含めることができます。組織にとって最も有用なことに基づいて、リソース グループにリソースを割り当てる方法を決めてください。[詳細情報](#)

プロジェクトの詳細

サブスクリプション \* ①

リソース グループ \* ①  ✓

リソースの詳細

リージョン \* ①  ✓

#### 5. 「確認および作成」をクリック

#### 6. エラーがないことを確認し「作成」をクリックし、新しいリソース グループを作成

### Task 2: ストレージ アカウントのプロビジョニング

#### 1. Azure ポータル (<https://portal.azure.com>) を開く

#### 2. 「+リソースの作成」をクリックし、「ストレージ」 - 「ストレージ アカウント」を選択

新規

Marketplace を検索

Azure Marketplace すべて表示 おすすめ すべて表示

作業の開始	ストレージ アカウント - Blob、File、Table、Queue クイックスタート チュートリアル
最近作成	Azure Stack Edge / Data Box Gateway 詳細情報
AI + Machine Learning	Data Lake Storage Gen1 クイックスタート チュートリアル
分析	Azure Data Box 詳細情報
ブロックチェーン	バックアップおよびサイトの回復 クイックスタート チュートリアル
Compute	AltaVault AVA-c4, version 4.4.1 (レビュー) プレビュー
コンテナー	Cloudian HyperCloud for Azure (レビュー) プレビュー
データベース	Veeam Cloud Connect for the Enterprise (レビュー) プレビュー
開発者ツール	
DevOps	
ID	
統合	
モノのインターネット (IoT)	
メディア	
Mixed Reality	
管理ツール	
ネットワーキング	
サービスとしてのソフトウェア (SaaS)	
セキュリティ	
ストレージ	
Web	

### 3. ストレージ アカウントの作成の「基本」ブレードで、次の構成オプションを指定

- リソース グループ（先の手順で作成したリソース グループを選択）
- ストレージ アカウント名（任意：小文字と数字で 3 から 24 文字で指定、一意の名前）
- 場所（リソース グループと同じ地域を指定）
- パフォーマンス（**Standard** を選択）
- アカウントの種類（**StorageV2** を選択）
- レプリケーション（**ローカル冗長ストレージ (LRS)** を選択）
- アクセス層（**ホット** を選択）

**基本** **ネットワーク** **詳細** **タグ** **確認および作成**

Azure Storage は、高可用性、セキュリティ、耐久性、スケーラビリティ、冗長性を備えたクラウド ストレージを提供する Microsoft が管理するサービスです。Azure Storage には、Azure BLOB (オブジェクト)、Azure Data Lake Storage Gen2、Azure Files、Azure Queues、Azure Tables が含まれます。ストレージ アカウントのコストは、使用量と、下で選ぶオプションに応じて決まります。Azure ストレージ アカウントの詳細 [☞](#)

**プロジェクトの詳細**  
デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソース グループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション \*

リソース グループ \*  [新規作成](#)

**インスタンスの詳細**  
既定の展開モデルは Resource Manager であり、これは最新の Azure 機能をサポートしています。代わりに、従来の展開モデルを使った展開も選択できます。クラシック展開モデルを選択します

ストレージ アカウント名 \* ①  [✓](#)

場所 \*  [▼](#)

パフォーマンス ①  Standard  Premium

アカウントの種類 ①  [▼](#)

レプリケーション ①  [▼](#)

アクセス層 (既定) ①  ケール  ホット

### 4. 「確認および作成」をクリック

### 5. エラーがないことを確認し「作成」をクリック

### 6. ストレージ アカウントのプロビジョニング完了後、「リソースに移動」をクリック

#### ✓ デプロイが完了しました

 デプロイ名: Microsoft.StorageAccount-20200207141312  
サブスクリプション: [\[遮断\]](#)  
リソース グループ: [HOL-2020-03-Serverless-RG](#)

▼ 展開の詳細 ([ダウンロード](#))

^ 次の手順

[リソースに移動](#)

### 7. ストレージ アカウントの管理ブレードが表示

### 8. ストレージ アカウントの「Blob service」メニュー配下の「コンテナー」をクリック

## 9. 以下の構成オプションを指定し、2つのコンテナーを作成

- a. 名前 (**images, export**)
- b. パブリック アクセス レベル (**プライベート**)

The screenshot shows the 'Containers' blade in the Azure portal. At the top, there are buttons for '+ Container', 'Access level', 'Update', and 'Delete'. Below that, a search bar contains '新しいコンテナー' (New container). A form is displayed for creating a new container:

- Name \***: images
- Public access level**: Private (匿名アクセスはありません)

At the bottom are 'OK' and 'Cancel' buttons.

作成後の画面には以下のコンテナーのリストが表示

The screenshot shows the 'Containers' blade with a search bar containing 'Prefixによるコンテナーの検索' (Search by prefix). The table lists two containers:

名前	最終変更日時	パブリック アク...	リース状態
export	2020/2/7 14:34:08	プライベート	利用可能
images	2020/2/7 14:33:16	プライベート	利用可能

## Task 3: Function Apps のプロビジョニング

1. Azure ポータル (<https://portal.azure.com>) を開く
2. 「+リソースの作成」をクリックし、画面上部の検索ボックスに **function** と入力  
表示される候補から **Function App** を選択



3. 関数アプリの作成ブレードで「**作成**」ボタンをクリック
4. 関数アプリの「基本」ブレード内で、以下の構成オプションを指定

- a. リソース グループ (先の手順で作成したリソース グループを選択)
- b. 関数アプリ名 (名前は 2 文字以上、一意の名前 ... TollBoothFunctionApp\*\*\* の名前で作成)
- c. 公開 (コードを選択)
- d. ランタイム スタック (.NET Core を選択)

## e. 地域（リソース グループと同じ地域を指定）

[基本](#) [ホスト中](#) [監視](#) [タグ](#) [確認および作成](#)

関数アプリを作成すると、関数を論理ユニットとしてグループ化できるため、リソースの管理、デプロイ、共有が容易になります。関数を使用するごとに VM を作成したり、Web アプリケーションを公開したりすることなく、サーバーレス環境でコードを実行できます。

**プロジェクトの詳細**

デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソース グループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション \* ①

リソース グループ \* ①  HOL-2020-03-Serverless-RG [新規作成](#)

**インスタンスの詳細**

関数アプリ名 \*  TollBoothFunctionApp01 [確認](#) .azurewebsites.net

公開 \* [コード](#) Docker コンテナー

ランタイム スタック \*  .NET Core

地域 \*  West US 2

[確認および作成](#) [次: ホスト中 >](#)

5. 「次: ホスト中 >」をクリック

6. 「ホスト中」ブレード内で、以下の構成オプションを指定

- ストレージ アカウント（新規作成）
- オペレーティング システム（Windows）
- プランの種類（従量課金プラン）

[基本](#) [ホスト中](#) [監視](#) [タグ](#) [確認および作成](#)

**Storage**

関数アプリを作成する場合は、Blob、キュー、テーブル ストレージをサポートする汎用 Azure Storage アカウントを作成するか、そのアカウントにリンクする必要があります。

ストレージ アカウント \*  (新規) storageaccounthol2084bb [新規作成](#)

**オペレーティング システム**

ランタイム スタックの選択に基づいて、オペレーティング システムが推奨されています。

オペレーティング システム \* [Linux](#) [Windows](#)

**プラン**

選択したプランによって、アプリの拡張方法、有効な機能、および価格の設定方法が決まります。[詳細情報](#)

プランの種類 \* ①  従量課金プラン

[確認および作成](#) [次: 監視 >](#)

7. 「次: 監視 >」をクリック

8. 「監視」ブレード内で「Application Insights を有効にする」を「いいえ」に設定

監視 タグ 確認および作成

Azure Monitor では、アプリケーション、インフラストラクチャ、ネットワークを完全に監視できます。 詳細情報

**Application Insights**

Application Insights を有効にする \*

いいえ はい

確認および作成 < 前へ 次: タグ >

9. 「確認および作成」をクリック

10. 「作成」をクリック

11. 1-10 のステップを繰り返し、新しい Function App をもう 1 つプロビジョニング

(2つ目の Function App の基本ブレード ... TollBoothEvent\*\*\* の名前で作成)

基本 ホスト中 監視 タグ 確認および作成

関数アプリを作成すると、関数を論理ユニットとしてグループ化できるため、リソースの管理、デプロイ、共有が容易になります。関数を使用すると、最初に VM を作成したり、Web アプリケーションを公開したりすることなく、サーバーレス環境でコードを実行できます。

プロジェクトの詳細

デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソース グループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション \* ①

リソース グループ \* ① HOL-2020-03-Serverless-RG 新規作成

インスタンスの詳細

関数アプリ名 \* TollBoothEvents01 .azurewebsites.net

公開 \*

ランタイム スタック \* .NET Core

地域 \* West US 2

## Task 4: Event Grid トピックのプロビジョニング

1. Azure ポータル (<https://portal.azure.com>) を開く
2. 「+リソースの作成」をクリックし、画面上部の検索ボックスに **event grid** と入力  
表示される候補から **Event Grid Topic** を選択

新規

event grid

Event Grid Domain

Event Grid Topic

Azure Event Grid on IoT Edge

3. Event Grid Topic ブレードで「作成」をクリック

4. 「トピックの作成」ブレード内で、以下の構成オプションを指定

- a. 名前（一意の名前）
- b. リソース グループ（先の手順で作成したリソース グループを選択）
- c. 場所（リソース グループと同じ地域を指定）
- d. イベント スキーマ（イベント グリッド スキーマ を選択）

トピックの作成

名前 \*

TollBoothEventGrid01

サブスクリプション \*

---

リソース グループ \*

HOL-2020-03-Serverless-RG

新規作成

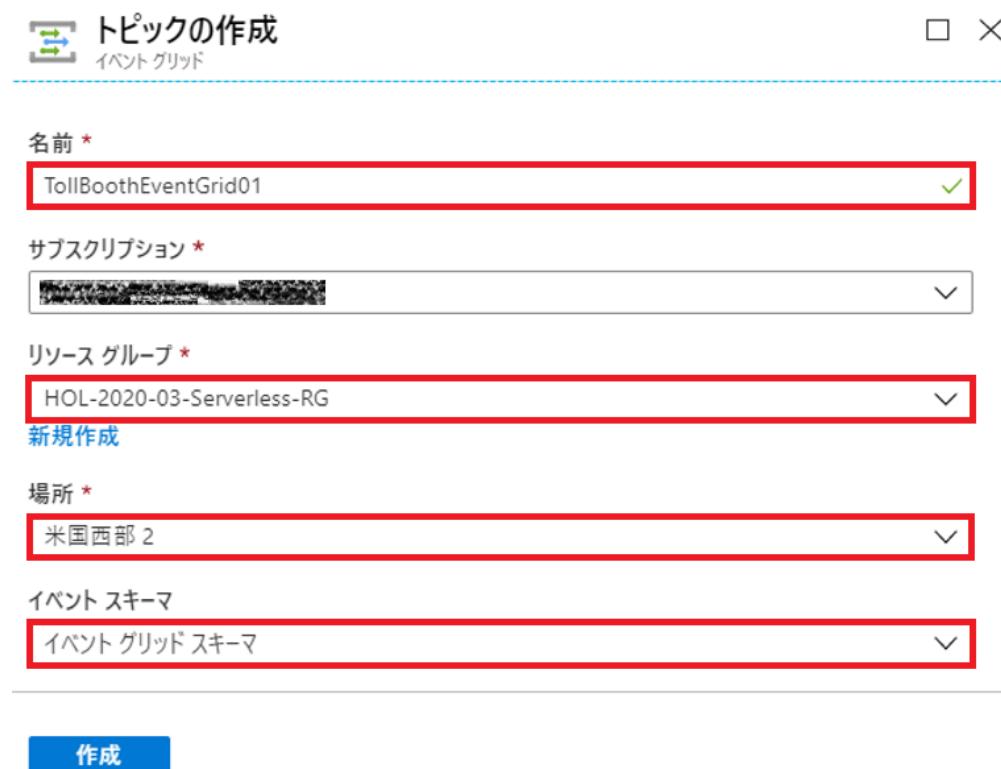
場所 \*

米国西部 2

イベント スキーマ

イベント グリッド スキーマ

作成



5. 「作成」をクリック

### Task 5: Azure Cosmos DB アカウントのプロビジョニング

1. Azure ポータル (<https://portal.azure.com>) を開く
2. 「+リソースの作成」をクリックし、「データベース」 - 「Azure Cosmos DB」を選択

## 新規

The screenshot shows the Azure Marketplace search results for databases. The search bar at the top contains the text "Marketplace を検索". Below it, there are two tabs: "Azure Marketplace" and "すべて表示" (All). Underneath are two more tabs: "おすすめ" (Recommended) and "すべて表示" (All). On the left, a sidebar lists categories: 作業の開始, 最近作成, AI + Machine Learning, 分析, ブロックチェーン, Compute, コンテナー, **データベース** (highlighted with a blue border), 開発者ツール, DevOps, ID, 統合, モノのインターネット (IoT), メディア, Mixed Reality, 管理ツール, ネットワーキング, and サービスとしてのソフトウェア (SaaS). The main content area displays several database services with their icons and names: Azure SQL Managed Instance (cloud icon), SQL Database (SQL icon), Azure Synapse Analytics (formerly SQL DW) (hexagon icon), Azure Database for MariaDB (boat icon), Azure Database for MySQL (MySQL icon), Azure Database for PostgreSQL (database icon), **Azure Cosmos DB** (globe icon) (highlighted with a red border), and SQL Server 2017 Enterprise Windows Server 2016 (monitor icon).

3. 「Azure Cosmos DB アカウントの作成」の「基本」ブレードで、以下の構成オプションを指定
  - a. リソース グループ（先の手順で作成したリソース グループを選択）
  - b. アカウント名（小文字、数字、'-'のみ、3 から 31 文字で一意の名前）
  - c. API (**コア (SQL)** を選択)
  - d. 場所（リソース グループと同じ地域を選択）
  - e. geo 変長性（**無効**を選択）
  - f. マルチ リージョン書き込み（**無効**を選択）
  - e. Availability Zones（**無効**を選択）

**基本 ネットワーク タグ 確認と作成**

Azure Cosmos DB は、フル マネージドでグローバル分散型のマルチモデル データベース サービスで、任意の数の Azure リージョン間でデータを透過的にレプリケートできます。スループットやストレージを弾力的にスケーリングし、お気に入りの API を使用して、99.999% の SLA で保証された 10 ミリ秒未満の高速なデータ アクセスをご利用いただけます。

### プロジェクトの詳細

デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソース グループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション \* [Redacted] ▾

リソース グループ \* HOL-2020-03-Serverless-RG ▾  
[新規作成](#)

### インスタンスの詳細

アカウント名 \* tollboothdb01 ✓

API \* ① コア (SQL) ▾

Apache Spark ① Notebooks (preview) Notebooks with Apache Spark (preview) [Sign up for Apache Spark preview](#) None

場所 \* (米国) 米国西部 2 ▾

geo 冗長性 ① 有効 無効

マルチ リージョン書き込み ① 有効 無効

Availability Zones ① 有効 無効

\*Up to 33% off multi-region writes is available to qualifying new accounts only. Accounts must be created between December 1, 2019 and February 29, 2020. Offer limited to accounts with both account locations and geo-redundancy, and applies only to multi-region writes in those same regions. Both Geo-Redundancy and Multi-region Writes must be enabled in account settings. Actual discount will vary based on number of qualifying regions selected.

4. 「確認と作成」をクリック
  5. エラーがないことを確認し「**作成**」をクリック
  6. Azure Cosmos DB アカウントのプロビジョニング完了後、「リソースに移動」をクリック



7. Azure Cosmos DB アカウントの管理ブレードが表示
  8. メニューの「データ エクスプローラー」 - 「**New Container**」をクリック

The screenshot shows the Azure portal interface for a database named 'tollboothdb01'. The left sidebar contains links for Overview, Activity Log, IAM, Tags, Diagnose and solve problems, Quick start, Notifications, and the Data Explorer blade, which is currently selected and highlighted with a grey background. The main content area is titled 'SQL API' and features a 'New Container' button, also highlighted with a red box. Other buttons include 'Enable Notebooks (Preview)' and a refresh icon.

## 9. 「Add Container」 ブレードで、以下の構成オプションを指定

- Database Id (**Create new / LicensePlates**)
- Provision database throughput(チェックを外す)
- Container Id (**Processed**)
- Partition Key (**/licensePlateText**)
- Throughput (**Manual / 5000**)

Add Container ×

**Start at \$24/mo per database, multiple containers included** [More details](#)

\* Database id ①  
 Create new  Use existing  
  
 Provision database throughput ①

\* Container id ①

\* Partition key ①  
  
 My partition key is larger than 100 bytes

\* Throughput (400 - 100,000 RU/s) ①  
 Autopilot (preview)  Manual

Estimated spend (USD): **\$0.40 hourly / \$9.60 daily** (1 region, 5000RU/s, \$0.00008/RU)

Unique keys ①  
+ Add unique key

OK

10. 「OK」をクリック

11. 再度「New Container」をクリックし、新しいコンテナーを作成

12. 「Add Container」ブレードで、以下の構成オプションを指定

- a. Database Id (**Use existing / LicensePlates**)
- b. Provision database throughput(チェックを外す)
- c. Container Id (**NeedsManualReview**)
- d. Partition Key (**fileName**)
- e. Throughput (**Manual / 5000**)

Add Container ×

**Start at \$24/mo per database, multiple containers included** [More details](#)

\* Database id ⓘ  
 Create new  Use existing  
LicensePlates

\* Container id ⓘ  
NeedsManualReview

\* Partition key ⓘ  
/fileName

My partition key is larger than 100 bytes

\* Throughput (400 - 100,000 RU/s) ⓘ  
 Autopilot (preview)  Manual  
5000

Estimated spend (USD): **\$0.40 hourly / \$9.60 daily** (1 region, 5000RU/s, \$0.00008/RU)

Unique keys ⓘ

+ Add unique key

**OK**

13. 「OK」をクリック

作成後、データ エクスプローラーは以下の画面を表示

## Task 6: Computer Vision API のプロビジョニング

1. Azure ポータル (<https://portal.azure.com>) を開く
2. 「+リソースの作成」をクリックし、画面上部の検索ボックスに **computer vision** と入力  
表示される候補から **Computer Vision** を選択

新規



3. Computer Visoin ブレードで「作成」をクリック
4. Computer Vision の「作成」ブレードで、以下の構成オプションを指定
  - a. 名前（英数字と '-' のみで 2 から 64 文字で一意の名前を指定）
  - b. 場所（リソース グループと同じ地域を選択）
  - c. 価格レベル（S1 を選択）
  - d. リソース グループ（先の手順で作成したリソース グループ）

名前 *	TollBoothVision01
サブスクリプション *	[redacted]
場所 *	(米国) 米国西部 2
価格レベル (価格の詳細を表示) *	S1 (10 1 秒あたりの呼び出し回数)
リソース グループ *	HOL-2020-03-Serverless-RG
<input style="background-color: #0072BC; color: white; font-weight: bold; padding: 5px 10px; margin-right: 10px;" type="button" value="Create"/> <a href="#">Automation options</a>	

## 5. 「作成」をクリック

### Task 7: Azure Key Vault のプロビジョニング

1. Azure ポータル (<https://portal.azure.com>) を開く
2. 「+リソースの作成」をクリックし、画面上部の検索ボックスに **key vault** と入力  
表示される候補から **Key Vault** を選択

新規



3. Key Vault ブレードで「作成」をクリック
4. キー コンテナーの作成の「基本」ブレードで、以下の構成オプションを指定
  - a. リソース グループ（先の手順で作成したリソース グループを選択）
  - b. Key Vault 名（英数字と '-' のみで 3 から 24 文字で一意の名前を指定）
  - c. 地域（リソース グループと同じ地域を選択）
  - d. 論理的な削除（無効化）

[基本](#) [アクセス ポリシー](#) [ネットワーク](#) [タグ](#) [確認および作成](#)

Azure Key Vault は、キー、シークレット、証明書を管理するために使用されるクラウド サービスです。Key Vault を使用すると、開発者がコードにセキュリティ情報を保存する必要がなくなります。これにより、アプリケーション シークレットのストレージを一元化できるため、シークレットが漏洩する可能性が大幅に減少します。Key Vault を使用すると、ハードウェア セキュリティモジュールまたは HSM でサポートされているシークレットとキーを安全に保存できます。使用されている HSM は、連邦情報処理規格 (FIPS) 140-2 レベル 2 が検証されています。さらに、キー コンテナーでは、コンプライアンスのための完全な監査証跡を確保できるように、シークレットのアクセスと使用のすべての試行に関するログが提供されます。 [詳細情報](#)

プロジェクトの詳細  
デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソース グループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション \*

リソース グループ \*  [新規作成](#)

インスタンスの詳細  
Key Vault 名 \* ①  ✓

地域 \*

価格レベル \* ①  ✓

論理的な削除 ① 無効化 有効化

[確認および作成](#) [次へ : アクセス ポリシー >](#) [< 前へ](#)

5. 「確認および作成」をクリック

6. エラーがないことを確認し「作成」をクリック

7. 展開完了後、「リソースに移動」をクリック

## ✓ デプロイが完了しました

デプロイ名: TollBoothVault01  
サブスクリプション:  [Redacted]  
リソース グループ: HOL-2020-03-Serverless-RG

▽ 展開の詳細 ([ダウンロード](#))

^ 次の手順

[リソースに移動](#)

8. キー コンテナーの管理ブレードが表示

9. メニューの「シークレット」 - 「**生成/インポート**」をクリック



The screenshot shows the 'TollBoothVault01 - シークレット' blade in the Azure portal. On the left, there's a navigation menu with items like '概要', 'アクティビティ ログ', 'アクセス制御 (IAM)', 'タグ', '問題の診断と解決', 'イベント (プレビュー)', '設定', 'キー', 'シークレット' (which is highlighted), and '証明書'. The main area has a search bar and buttons for '+ 生成/インポート' (highlighted with a red box), '更新', and 'バックアップの復元'. A message below the buttons says '利用可能なシークレットがありません。' (No secrets available).

(シークレット作成画面)

## シークレットの作成

アップロード オプション  
手動

名前 \* ⓘ  
blobStorageConnection

値 \* ⓘ  
.....

コンテンツの種類 (省略可能)

アクティベーションする日を設定しますか? ⓘ

有効期限を設定しますか? ⓘ

有効ですか? はい いいえ

**作成**

シークレット作成時の名前と値のペアは以下の表を参考

各シークレットは名前と値の指定のみで、他のフィールドはデフォルト値のまま作成

Name	Value
computerVisionApiKey	Computer Vision API Key
eventGridTopicKey	Event Grid Topic access Key
cosmosDBAuthorizationKey	Cosmos DB Primary Key
blobStorageConnection	Blob storage connection string

シークレット作成後の画面には以下のリストが表示

+ 生成/インポート ⟳ 更新 ⟲ バックアップの復元

名前	種類	状態
computerVisionApiKey		✓ 有効にする
eventGridTopicKey		✓ 有効にする
cosmosDBAuthorizationKey		✓ 有効にする
blobStorageConnection		✓ 有効にする

## Exercise 2: 写真の処理を行う Azure Functions の開発と発行

所要時間：45分

Visual Studio に統合された Azure Functions ツールを使用して、ローカル環境で関数を開発およびデバッグし、Azure に展開します。TollBooth ソリューションには、必要なコードのほとんどが含まれています。Azure に展開する前に不足しているコードを追加する必要があります。（コードを追加する箇所は TODO としてマークされています。）

Visual Studio から Azure に展開する前に Azure ポータルで Function App にアプリケーション設定を構成します。誤ってセキュリティ情報が漏洩しないようにアプリケーション設定には、Cosmos DB の接続文字列や Computer Vision API の API Key 直接記述するのではなく、Key Vault シークレットの URI を参照するように指定します。Function App が Key Vault にアクセスしてシークレットにアクセスできるように Function App ヘシステム割り当てマネージド ID を設定し、Key Vault 側にアクセス許可を与えるためのアクセス ポリシーを作成します。

注意：NuGet パッケージのバージョンは更新しないでください。

このソリューションは、現在定義されている NuGet パッケージのバージョンで動作するように構築されています。

## 参考情報

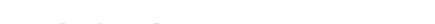
- App Service と Azure Functions の Key Vault 参照を使用する  
<https://docs.microsoft.com/azure/app-service/app-service-key-vault-references>
- App Service と Azure Functions でマネージドID を使用する方法  
<https://docs.microsoft.com/azure/app-service/overview-managed-identity>
- キー コンテナーへのアクセスをセキュリティで保護する  
<https://docs.microsoft.com/ja-jp/azure/key-vault/key-vault-secure-your-key-vault>
- マネージド ID で Key Vault の認証を提供する  
<https://docs.microsoft.com/azure/key-vault/managed-identity>
- Visual Studio を使用する Azure Functions の開発  
<https://docs.microsoft.com/azure/azure-functions/functions-develop-vs>
- Azure Functions Core Tools の操作  
<https://docs.microsoft.com/azure/azure-functions/functions-run-local?tabs=windows>

## Task 1: アプリケーション設定の構成

- Azure ポータル (<https://portal.azure.com>) を開く
- 先の手順で作成したリソース グループへ移動し、TollBoothFunctionApp\*\*\* を選択  
TollBoothFunctionApp\*\*\* の名前で作成していない場合は、どちらか片方を選択

	名前 ↑↓	種類 ↑↓
<input type="checkbox"/>	ASP-HOL202003ServerlessRG-9e11	App Service プラン
<input type="checkbox"/>	ASP-HOL202003ServerlessRG-acaa	App Service プラン
<input type="checkbox"/>	holserverlesstollbooth	ストレージ アカウント
<input type="checkbox"/>	storageaccounthol2082c6	ストレージ アカウント
<input type="checkbox"/>	storageaccounthol2084bb	ストレージ アカウント
<input type="checkbox"/>	tollboothdb01	Azure Cosmos DB アカウント
<input type="checkbox"/>	TollBoothEventGrid01	Event Grid トピック
<input type="checkbox"/>	TollBoothEvents01	App Service
<input checked="" type="checkbox"/>	TollBoothFunctionApp01	App Service
<input type="checkbox"/>	TollBoothVault01	キー コンテナー
<input type="checkbox"/>	TollBoothVision01	Cognitive Services

### 3. 概要パネルの**構成**をクリック

概要	プラットフォーム機能
 停止	 スワップ
 再開	 発行プロファイルの取得
状態  Running	サブスクリプション 
	サブスクリプション ID 

## 構成済みの機能

- ## 関数アプリの設定

4. +新しいアプリケーション設定をクリックし、以下のキー/値のペアを追加

The screenshot shows the Azure portal's application settings configuration. It includes sections for 'Application settings' and 'Connection strings'. The 'Application settings' section lists environment variables like 'AzureWebJobsStorage', 'FUNCTIONS\_EXTENSION\_VERSION', etc., each with a value and edit options. The 'Connection strings' section is currently empty.

名前	値	ソース	デプロイ スロットの設定	削除	編集
AzureWebJobsStorage	非表示の値。表示するには、上記の [値] アプリの構成				
FUNCTIONS_EXTENSION_VERSION	非表示の値。表示するには、上記の [値] アプリの構成				
FUNCTIONS_WORKER_RUNTIME	非表示の値。表示するには、上記の [値] アプリの構成				
WEBSITE_CONTENTAZUREFILECONNECTIONS	非表示の値。表示するには、上記の [値] アプリの構成				
WEBSITE_CONTENTSHARE	非表示の値。表示するには、上記の [値] アプリの構成				

名前	値	種類	デプロイス... 削除	編集
(表示する接続文字列がありません)				

computerVisionApiUrl		Computer Vision API のエンドポイントに <b>vision/v2.0/ocr</b> を追加して指定 <code>http://&lt;YOUR-SERVICE-NAME&gt;.cognitiveservices.azure.com/vision/v2.0/ocr</code>
computerVisionApiKey		<code>@Microsoft.KeyVault(SecretUri=referenceString)</code> referenceString に Key Vault の <b>computerVisionApiKey</b> シークレットの URI を指定
eventGridTopicEndpoint		Event Grid Topic のエンドポイント
eventGridTopicKey		<code>@Microsoft.KeyVault(SecretUri=referenceString)</code> referenceString に Key Vault の <b>eventGridTopicKey</b> シークレットの URI を指定
cosmosDBEndPointUrl		Cosmos DB URI
cosmosDBAuthorizationKey		<code>@Microsoft.KeyVault(SecretUri=referenceString)</code> referenceString に Key Vault の <b>cosmosDBAuthorizationKey</b> シークレットの URI を指定
cosmosDBDatabaseId		<b>LicensePlates</b> (Cosmos DB データベース Id)
cosmosDBCollectionId		<b>Processed</b> (Cosmos DB コレクション Id)
exportCsvContainerName		<b>export</b> (Blob ストレージのコンテナー)
blobStorageConnection		<code>@Microsoft.KeyVault(SecretUri=referenceString)</code> referenceString に Key Vault の <b>blobStorageConnection</b> シークレットの URI を指定

\*エンドポイントや API KEY, 接続文字列は各リソースの管理ブレードから取得  
(例 : Computer Vision API のエンドポイントとキーの取得)

TollBoothVision01 - キーとエンドポイント

検索 (Ctrl+ /) < キー 1 の再生成 キー 2 の再生成

名前: TollBoothVision01

エンドポイント: https://tollboothvision01.cognitiveservices.azure.com/

このサブスクリプションキーは、Cognitive Service APIにアクセスするために使用されます。キーを共有しないでください。Azure Key Vaultなどを使用して、安全に保管します。これらのキーを定期的に再生成することをお勧めします。API呼び出しを行うには、1つのキーのみが必要です。最初のキーを再生成する場合、2番目のキーを使用してサービスに引き続きアクセスできます。

キー 1: [REDACTED]

キー 2: [REDACTED]

\*アプリケーション設定の入力画面

#### アプリケーション設定の追加/編集

名前:	blobStorageConnection
値:	@Microsoft.KeyVault(SecretUri=https://[REDACTED].vault.azure.net/secrets/blobStorageConnection/[REDACTED])
<input type="checkbox"/> デプロイ スロットの設定	
<b>OK</b>	<b>キャンセル</b>

5. 「保存」をクリック



## Task 2: Key Vault へアクセスするためのシステム割り当てマネージド ID の生成

1. Function App の管理ブレードで「プラットフォーム機能」を開く
2. ID を開く

The screenshot shows the 'Platform Features' tab selected in the Azure Functions interface. On the left, there's a search bar labeled '機能の検索'. Below it are sections for 'General Settings' and 'Networking'. In the 'Networking' section, the 'ID' option is highlighted with a red box.

3. 「システム割り当て済み」タブで「状態」をオンに設定し、「保存」をクリック

The screenshot shows the 'Identity' blade in the Azure portal. It has tabs for 'System-assigned' and 'User-assigned'. The 'System-assigned' tab is selected. At the bottom, there are buttons for 'Save', 'Delete', 'Update', and a feedback link. Below that, there's a 'Status' section with a switch set to 'On'.

4. 「システム割り当てマネージド ID を有効化する」のメッセージが表示されるので「はい」をクリック

### Task 3: Key Vaultへのアクセス許可を付与するアクセス ポリシーの作成

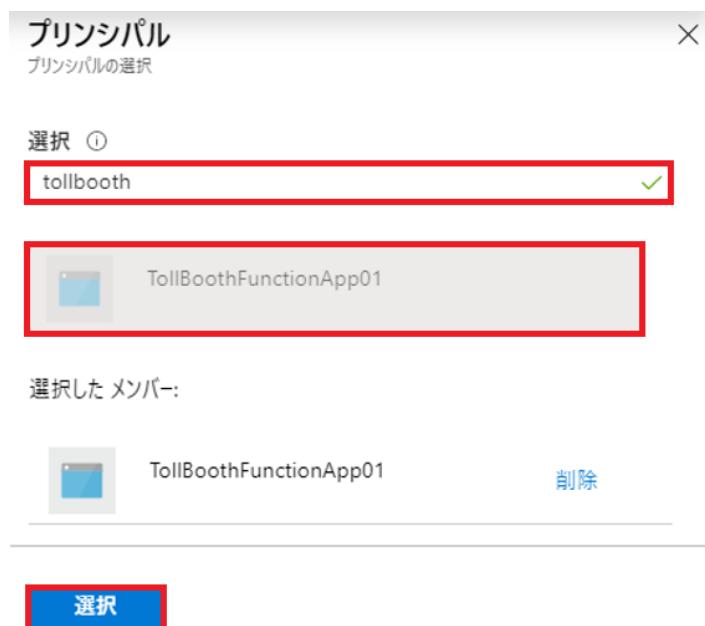
1. Key Vault の管理ブレードで「アクセス ポリシー」を開く
2. 「アクセス ポリシーの追加」をクリック

The screenshot shows the 'TollBoothVault01 - アクセス ポリシー' blade in the Azure portal. On the left, there's a sidebar with various navigation items like Overview, Activity Log, IAM, Tags, Diagnose & Solve, Preview, and more. Under 'Setting', 'Access Policies' is selected. The main area shows the 'Access Policy' configuration with sections for 'Effective access', 'Azure VMs (Deployment) (checkbox)', 'Azure Resource Manager (Template deployment) (checkbox)', and 'Azure Disk Encryption (Volume encryption) (checkbox)'. A red box highlights the '+ アクセス ポリシーの追加' (Add Access Policy) button. Below it, a table lists existing policies for 'User' (Hiroya Yamam...) with 9 items selected.

### 3. 「アクセス ポリシーの追加」フォームで「プリンシパルの選択」をクリック

The screenshot shows the 'Add Access Policy' form. It includes fields for 'Template composition (Optional)', 'Key access permission' (0 items selected), 'Secret access permission' (0 items selected), 'Certificate access permission' (0 items selected), and a 'Principal selection' section. This 'Principal selection' section is highlighted with a red box and contains a note '選択されていません' (Not selected). At the bottom, there's a 'Confirm application' section with a note '選択されていません' (Not selected) and a 'Add' button.

### 4. プリンシパル ブレードで先の手順で登録したサービス プリンシパルを検索し選択



## 5. 「アクセス ポリシーの追加」フォームに戻り「シークレットのアクセス許可」の「取得」を選択

### アクセス ポリシーの追加

アクセス ポリシーの追加

テンプレートからの構成 (省略可能)

キーのアクセス許可

0 項目が選択されました

シークレットのアクセス許可

取得

すべて選択

シークレットの管理操作

取得

一覧

セット

削除

回復

バックアップ

復元

特権シークレット操作

削除

## 6. 「追加」をクリックし

**アクセス ポリシーの追加**

テンプレートからの構成 (省略可能)

キーのアクセス許可

シークレットのアクセス許可

証明書のアクセス許可

プリンシパルの選択

★ TollBoothFunctionApp01 >

承認されているアプリケーション ①

選択されていません

**追加**

## 7. 「保存」をクリックし、これまでの設定を確定

TollBoothVault01 - アクセス ポリシー

キーコンテナー | ディレクトリ: Microsoft

検索 (Ctrl+ /) 保存 破棄 更新

変更をコミットするには、[保存] をクリックしてください。

アクセスの有効化:

- Azure Virtual Machines (展開用) ①
- Azure Resource Manager (テンプレートの展開用) ①
- Azure Disk Encryption (ポリューム暗号化用) ①

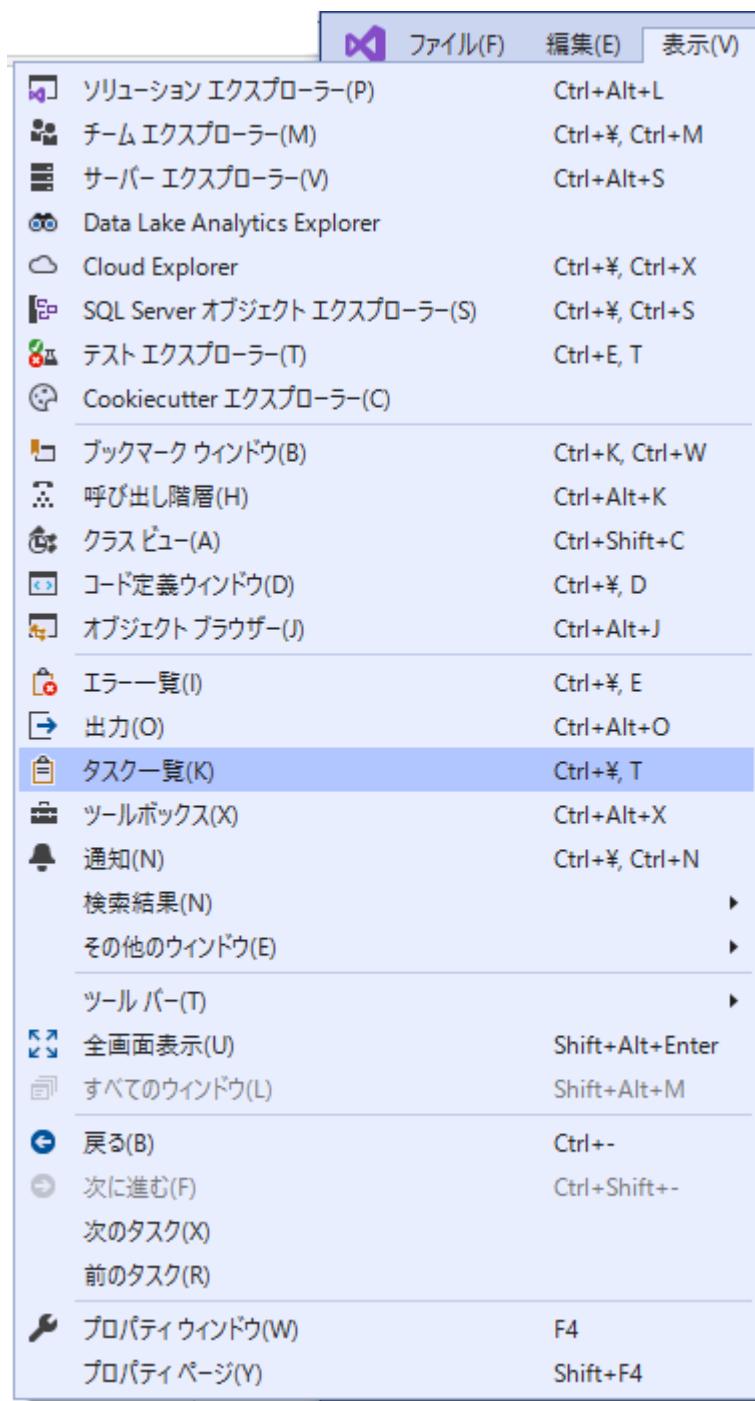
+ アクセス ポリシーの追加

現在のアクセス ポリシー

名前	メール	キーのアクセス許可
アプリケーション	TollBoothFuncti... 0 項目が選択されました	
ユーザー	Hiroya Yamam... 9 項目が選択されました	

## Task 4: ProcessImage 関数の開発

1. Visual Studio で「TollBooth」プロジェクトを開く
2. 「表示」メニューの「タスク一覧」を選択



### 3. TODO タスクの一覧が表示（各タスクは追加するコードの位置を示します。）

The screenshot shows the 'Tasks' window with the following content:

説明	プロジェクト	ファイル	行
TODO \$ exported の値が false である LicensePlateDataDocument オブジェクトのリストを collectionLink から取得	TollBooth	DatabaseMethods.cs	42
TODO \$ 以下の行を削除	TollBooth	DatabaseMethods.cs	44
TODO \$ メモリストリームから Blob を非同期にダウンロード	TollBooth	FileMethods.cs	60
TODO \$ 以下 2 つの変数に AppSettings プロパティを設定	TollBooth	FindLicensePlateText.cs	46
TODO 1: FindLicensePlateText.GetLicensePlate メソッドを実行し、licensePlateText 値を設定	TollBooth	ProcessImage.cs	63
TODO 3: ナンバープレートデータベースに保存するための eventType 値を含めて EventGrid へデータを送信	TollBooth	SendToEventGrid.cs	31
TODO 4: 手動確認の eventType 値を含めて EventGrid データを送信	TollBooth	SendToEventGrid.cs	36

### 4. TODO 1 をダブルクリックし、**ProcessImage.cs** のコードを追加する箇所を表示

"ProcessImage" は Azure Functions の関数のため `FunctionName` 属性で修飾されています。この関数は Event Grid サービスから HTTP 要求が送信されたことをトリガーされます。関数のトリガーは先の手順で作成したストレージ アカウントの images コンテナーにアップロードされる新しい Blob を監視します。Event Grid の通知から関数に渡されるデータには Blob の URL が含まれます。Blob の URL はアップロードされたイメージを取得するため入力

バインディングに渡されます。この後のタスクで Blob が作成されたイベントをサブスクライブする Event Grid サブスクリプションを作成します。

## 5. TODO 1 に以下のコードを追加

```
// TODO 1: FindLicensePlateText.GetLicensePlate メソッドを実行し、  
// licensePlateText 値を設定  
licensePlateText = await new FindLicensePlateText(log,  
    _client).GetLicensePlate(licensePlateImage);
```

## 6. タスク一覧から TODO 2 をダブルクリックし、**FindLicensePlateText.cs** を表示

FindLicensePlateText クラスは OCR を使用して写真からナンバープレートのテキストを検索して抽出します。このクラスは一時的なエラーを処理するためのオープンソースの .NET ライブラリ Polly を使用して復元パターンを実装しています。これはダウンストリーム サービス（今回は Computer Vision API）を過負荷にしないようにする際に役立ちます。

## 7. TODO 2 を以下のコードへ変更

```
// API URL と API キーの取得  
// TODO 2: 以下2つの変数に AppSettings プロパティを設定  
var uriBase = Environment.GetEnvironmentVariable("computerVisionApiUrl");  
var apiKey = Environment.GetEnvironmentVariable("computerVisionApiKey");
```

## 8. タスク一覧から TODO 3 をダブルクリックし、**SendToEventGrid.cs** を表示

SendToEventGrid クラスは、イベント タイプやナンバープレートを含むデータを Event Grid へ送信します。イベント リスナーはイベント タイプを使用してイベントをフィルタリングして処理します。ここで定義したイベント タイプは、先の手順で作成した 2 番目の Function App で使用します。

## 9. TODO 3, 4 に以下のコードを追加し、コードを完成

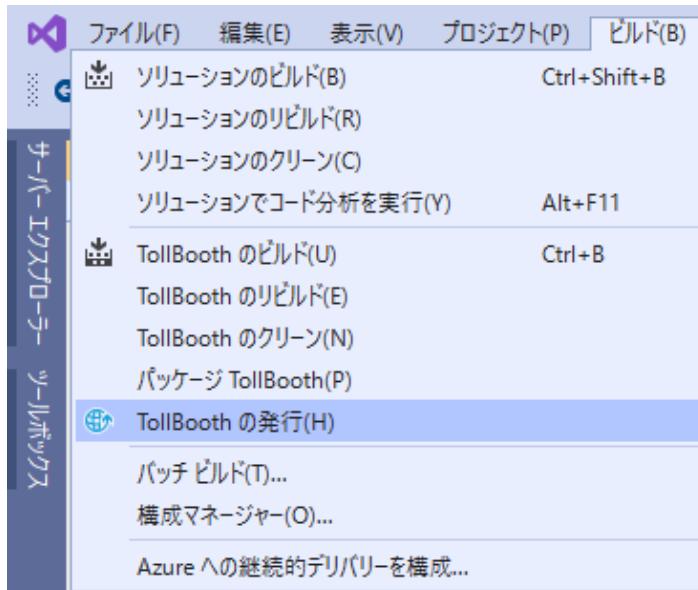
```
// 2つのルートのいずれかに送信  
// データベースに保存、もしくは手動での確認キューへ移動  
if (data.LicensePlateFound)  
{  
    // TODO 3: ナンバープレート データをデータベースに保存するための  
    // eventType 値を含めて EventGrid ヘデータを送信  
    await Send("savePlateData", "TollBooth/CustomerService", data);  
    // COMPLETE: await Send(...);  
}  
else  
{  
    // TODO 4: 手動確認の eventType 値を含めて EventGrid ヘデータを送信  
    await Send("queuePlateForManualCheckup", "TollBooth/CustomerService",  
    data);
```

```
// COMPLETE: await Send(...);  
}
```

TODO 5, 6, 7 は、とのタスクで完了します。

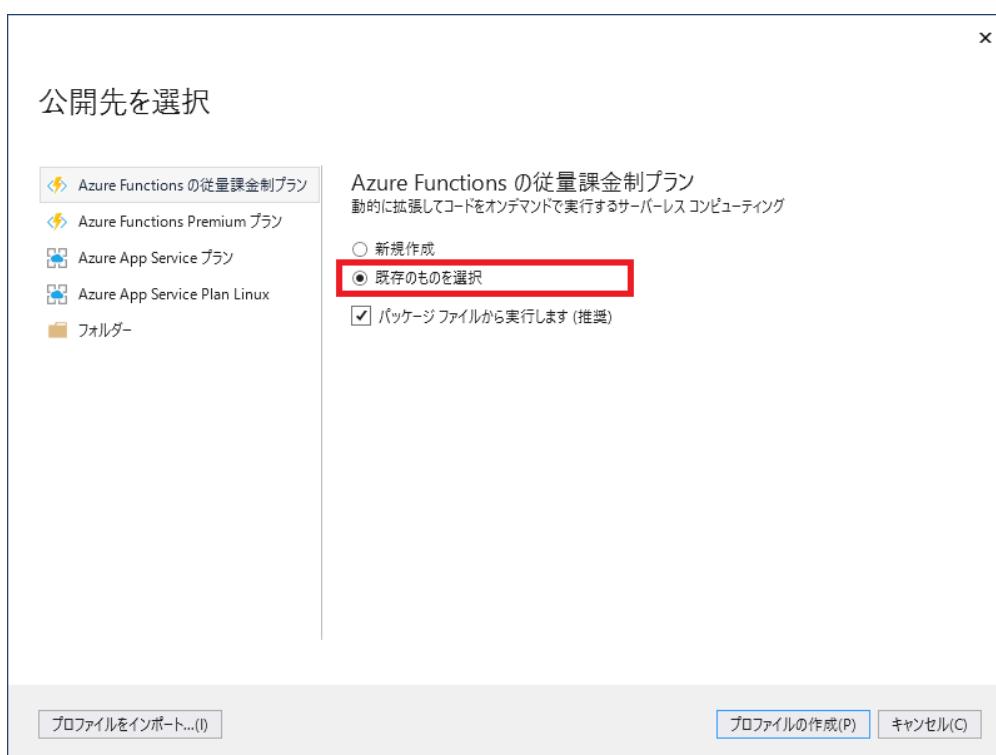
## Task 5: Visual Studio から Function App の公開

1. Visual Studio の「ビルド」メニューから「TollBooth の発行」を選択



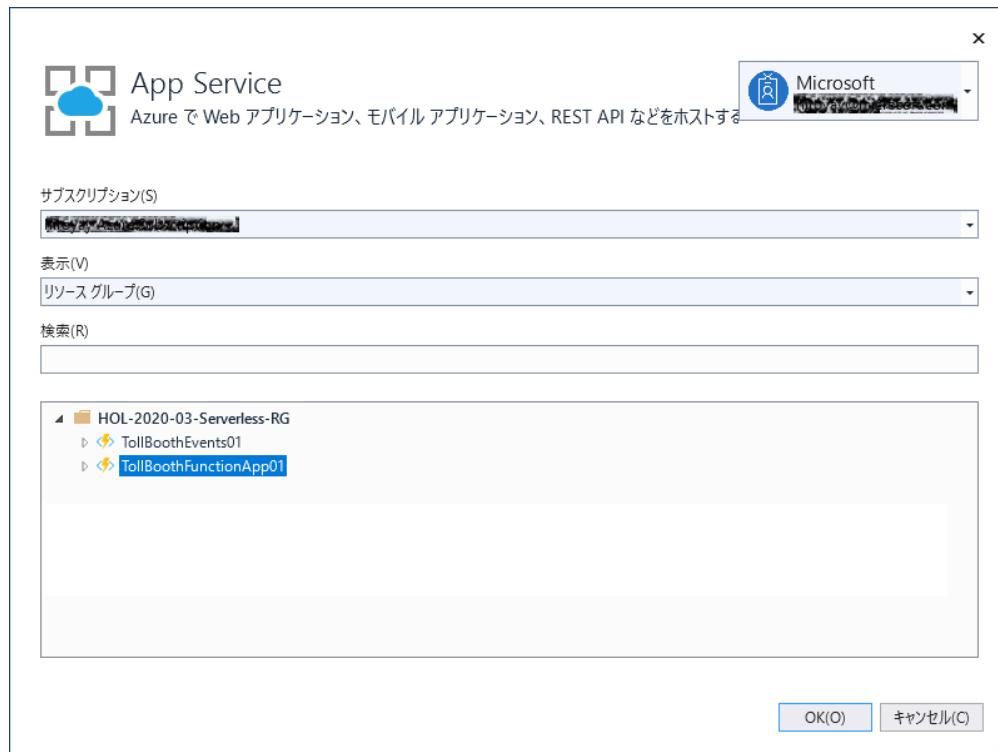
2. 「公開先の選択」画面が表示

「Azure Functions の従量課金制プラン」を選択し「既存のものを選択」を選択  
「パッケージファイルから実行します」にチェックを付けたまま「プロファイルの作成」をクリック



3. 「App Service」フォームでサブスクリプションを選択  
「表示」の下の「リソース グループ」を選択

先の手順で作成したリソース グループを展開し Function App を選択



4. 「OK」をクリック

5. 「発行」をクリックし、展開プロセスを開始

アプリの発行は Visual Studio の出力ウィンドウで確認できます。

完了すると ===== 公開: 1 正常終了, 0 失敗, 0 スキップ ===== のメッセージが表示

6. Azure ポータル (<https://portal.azure.com>) を開く

7. 先の手順でアプリを展開した Function App を選択

「関数（読み取り専用）」を展開し、Visual Studio から発行した関数が表示されることを確認

**TollBoothFunctionApp01**

関数アプリ

概要 プラットフォーム機能

停止 スワップ 再開 発行プロファイルの取得

状態 サブスクリプション  
✓ Running

サブスクリプション ID

構成済みの機能

関数アプリの設定 構成

## 8. 「ProcessImage」関数を選択し、「Event Grid サブスクリプションの追加」をクリック

**TollBoothFunctionApp01 - ProcessImage**

関数アプリ

function.json

保存 実行 Event Grid サブスクリプションの追加

```

1 {
2     "generatedBy": "Microsoft.NET.Sdk.Functions-1.0.29",
3     "configurationSource": "attributes",
4     "bindings": [
5         {
6             "type": "eventGridTrigger",
7             "name": "eventGridEvent"
8         }
9     ],
10    "disabled": false,
11    "scriptFile": "../bin/TollBooth.dll",
12    "entryPoint": "TollBooth.ProcessImage.Run"
13 }

```

パッケージファイルから実行しているため、アプリは現在読み取り専用モードです。変更を加えるには、Zip ファイルの内容と WEBSITE\_RUN\_FROM\_PACKAGE アプリ設定を更新します。

## 9. 「イベント サブスクリプションの作成」ブレードが表示されるので、以下の構成オプションを指定

- 名前 (processimagesub)
- イベントスキーマ (イベント グリッドスキーマを選択)
- トピックの種類 (先の手順で作成したストレージ アカウントを選択)

トピックの詳細

宛先にイベントをプッシュするトピックリソースを選択します。[詳細情報](#)

Topic Types	Storage Accounts
Subscription	[Redacted]
Resource Group	HOL-2020-03-Serverless-RG
Resource	holserverlesstollbooth

- イベントの種類 (Blob Create にチェック)



#### e. エンドポイントの詳細 (ProcessImage 関数)

A screenshot of the Azure portal's 'Event Grid Subscriptions' blade. It shows the configuration of a new subscription. The 'Name' field is set to 'processimagesub' and has a green checkmark. The 'Event schema' dropdown is set to 'Event Grid Schema' and has a red border. In the 'Topic' section, 'Storage Account' is selected and highlighted with a red border, and 'holserverlesstlbooth' is listed. In the 'Event type' section, 'Blob Created' is selected and highlighted with a red border. In the 'Endpoint' section, 'Azure Function' is selected and highlighted with a red border, and 'ProcessImage' is listed. At the bottom, there is a blue 'Create' button.

#### 10. 「作成」をクリック

## Exercise 3: Azure ポータルでの Function App の開発

所要時間：45分

Azure ポータルを使用して .NET Core (C#) で 2 つの新しい関数を作成します。これらは Event Grid によってトリガーされ、ProcessImage 関数によって実行されるライセンス プレート処理の結果を保存するために Cosmos DB へ出力されます。

### 参考情報

- Azure Portal で初めての関数を作成する  
<https://docs.microsoft.com/azure/azure-functions/functions-create-first-azure-function>

- Azure Functions と Cosmos DB を使用して非構造化データを格納する  
<https://docs.microsoft.com/azure/azure-functions/functions-integrate-store-unstructured-data-cosmosdb>

## Task 1: ライセンスプレートデータを Cosmos DB に保存する関数の作成

### 1. リソース グループからもう 1 つの Function App を選択

先の手順で Visual Studio からアプリを発行したものとは別の関数を選択

名前 ↑↓	種類 ↑↓
ASP-HOL202003ServerlessRG-9e11	App Service プラン
ASP-HOL202003ServerlessRG-acaa	App Service プラン
holserverlesstollbooth	ストレージ アカウント
storageaccounthol2082c6	ストレージ アカウント
storageaccounthol2084bb	ストレージ アカウント
tollboothdb01	Azure Cosmos DB アカウント
TollBoothEventGrid01	Event Grid トピック
<b>TollBoothEvents01</b>	<b>App Service</b>
TollBoothFunctionApp01	App Service
TollBoothVault01	キー コンテナー
TollBoothVision01	Cognitive Services

### 2. Function App の管理ブレードから「+新しい関数」をクリック



The screenshot shows the Azure portal's function app management interface for 'TollBoothEvents01'. On the left, there's a sidebar with 'TollBoothEvents01' selected. The main area has tabs for '概要' (Overview) and 'プラットフォーム機能' (Platform Features). Under '概要', it shows the app is 'Running' in the 'Running' state. It also displays the subscription ('hiroya Azure Subscription'), resource group ('HOL-2020-03-Serverless-RG'), URL ('https://tollboithevents01.azurewebsites.net'), and location ('West US 2'). Below this, there's a section for '構成済みの機能' (Configured Features) with a '新規機能' (New Function) button highlighted with a red box. A message at the bottom right says '関数アプリが作成されました。' (Function app created successfully.) and '次に、コードを追加します...' (Next, add code...).

### 3. 関数作成のガイダンスが表示されるので「ポータル内」を選択し「続行」をクリック

## .NET 用の Azure Functions - 作業の開始

クイック スタートのガイダンスに従って関数を作成し、発行します [詳細情報](#)



続行

- 「その他のテンプレート」を選択し「テンプレートの完了と表示」をクリック

## .NET 用の Azure Functions - 作業の開始

クイック スタートのガイダンスに従って関数を作成し、発行します [詳細情報](#)



戻る

テンプレートの完了と表示

- 検索ボックスに「event grid」と入力

表示されたテンプレートの候補から「Azure Event Grid trigger」をクリック

以下のテンプレートを選択します または [クイックスタートに移動します](#)

シナリオ: すべて

Azure Event Grid trigger	サーバーレス コミュニティ ライブラリ
イベント グリッドが新しいイベントを受信するたびに実行される関数	検索しているものが見つからない場合は Azure サーバーレス コミュニティ ライブラリをご確認ください。

## 6. 名前に **SavePlateData** と入力し「作成」をクリック

名前:  
SavePlateData

関数の作成後に Event Grid のサブスクリプションを追加する必要があります。

**作成**    [キャンセル](#)

## 7. **SavePlateData** 関数が作成されたことを確認

TollBoothEvents01 - SavePlateData  
関数アプリ

"TollBoothEvents01" [×](#)

hiroyay Azure Subscription

**関数アプリ**

**TollBoothEvents01**

**関数** +

**SavePlateData**

統合  
管理  
監視

## Task 2: 関数への Event Grid サブスクリプションの追加

### 1. 「Event Grid サブスクリプションの追加」をクリック

TollBoothEvents01 - SavePlateData

関数アプリ

run.csx

保存 実行 Event Grid サブスクリプションの追加

```

1 #r "Microsoft.Azure.EventGrid"
2 using Microsoft.Azure.EventGrid.Models;
3
4 public static void Run(EventGridEvent eventGridEvent, ILogger log)
5 {
6     log.LogInformation(eventGridEvent.Data.ToString());
7 }
8

```

hiroyay Azure Subscription

関数アプリ

TollBoothEvents01

関数

SavePlateData

統合 管理 監視

2. 「イベント サブスクリプションの作成」ブレードが表示されるので、以下の構成オプションを指定

- 名前 (**saveplatedatasub**)
- イベントスキーマ (イベント グリッドスキーマを選択)
- トピックの種類 (先の手順で作成した Event Grid トピックを選択)

トピックの詳細

宛先にイベントをプッシュするトピック リソースを選択します。[詳細情報](#)

Topic Types	Event Grid Topics
Subscription	<input type="text" value="████████████████"/>
Resource Group	HOL-2020-03-Serverless-RG
Resource	TollBoothEventGrid01

- イベントの種類 (イベントの種類のフィルターに **savePlateData** と入力)

イベントの種類

宛先にプッシュするイベントの種類を選択します。[詳細情報](#)

イベントの種類のフィルター  [+ イベントの種類の追加](#)

- エンドポイントの詳細 (SavePlateData 関数)

イベント サブスクリプションの作成

イベント グリッド

Basic フィルター 追加の機能

イベント サブスクリプションでは、トピック リソースから発生するイベントをリッスンし、それらのイベントをエンドポイント リソースに送信します。[詳細情報](#)

イベント サブスクリプションの詳細

名前: saveplatedatasub ✓

イベント スキーマ: イベント グリッド スキーマ ✓

トピックの詳細

宛先にイベントをプッシュするトピック リソースを選択します。[詳細情報](#)

トピックの種類: Event Grid トピック

トピックのリソース: TollBoothEventGrid01 (変更)

イベントの種類

宛先にプッシュするイベントの種類を選択します。[詳細情報](#)

イベントの種類のフィルター: savePlateData ✖ + イベントの種類の追加

エンドポイントの詳細

イベントを受信するイベント ハンドラーを選択します。[詳細情報](#)

エンドポイントのタイプ: Azure 関数 (変更)

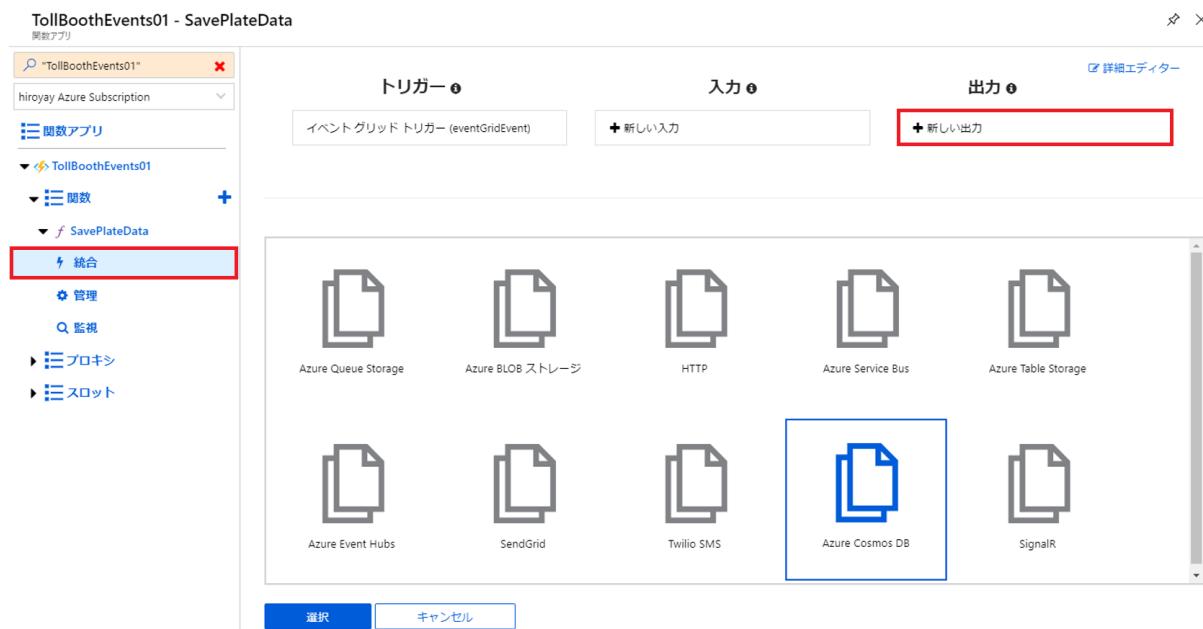
エンドポイント: SavePlateData (変更)

**作成**

### 3. 「作成」をクリック

#### Task 3: 関数への Cosmos DB 出力の追加

1. 「SavePlateData」関数を展開し「統合」をクリック
2. 「+ 新しい出力」をクリックし、出力リストから「Azure Cosmos DB」を選択し「選択」をクリック



3. データベース名に「**LicensePlate**」、コレクション名に「**Processed**」を入力

4. Azure Cosmos DB 出力フォームの「**Azure Cosmos DB アカウントの接続**」の「**新規**」をクリック



「接続」フォームで先の手順で作成した Cosmos DB アカウントを選択し「選択」をクリック



5. 「**保存**」をクリック

The screenshot shows the Azure Functions portal interface for the 'SavePlateData' function. The left sidebar shows the function app 'TollBoothEvents01' and its configuration sections: Management, Monitoring, Proxies, and Slots. The main area shows the trigger configuration for an 'eventGridEvent' trigger and the output binding configuration to an Azure Cosmos DB database.

- トリガー:** イベントグリッド トリガー (eventGridEvent)
- 入力:** + 新しい入力
- 出力:** + 新しい出力

**Azure Cosmos DB output**

- ドキュメントパラメータ名: outputDocument
- データベース名: LicensePlate (highlighted with a red box)
- コレクション名: Processed (highlighted with a red box)
- Azure Cosmos DB アカウント接続: tollboothdb01\_DOCUMENTDB (highlighted with a red box)
- パーティションキー (省略可能): パーティションキー (省略可能)
- Collection throughput (optional): Collection throughput (optional)

**操作ボタン:** 保存 (highlighted with a red box), キャンセル

## Task 4: run.csx へのコードの記述

1. **SavePlateData** をクリックし、**run.csx** を開く

The screenshot shows the Azure Functions portal interface with the 'SavePlateData' function selected in the list. The right sidebar provides management options for the function.

- 統合
- 管理 (highlighted with a blue box)
- 監視
- プロキシ
- スロット

2. 以下のコードを記述

```
#r "Microsoft.Azure.EventGrid"
#r "Newtonsoft.Json"

using Microsoft.Azure.EventGrid.Models;
```

```

using Newtonsoft.Json.Linq;

public static void Run(EventGridEvent eventGridEvent, out object
outputDocument, ILogger log)
{
    log.LogInformation(eventGridEvent.Data.ToString());

    var data = eventGridEvent.Data as JObject;

    var exportFlg = "exported";
    data[exportFlg] = false;

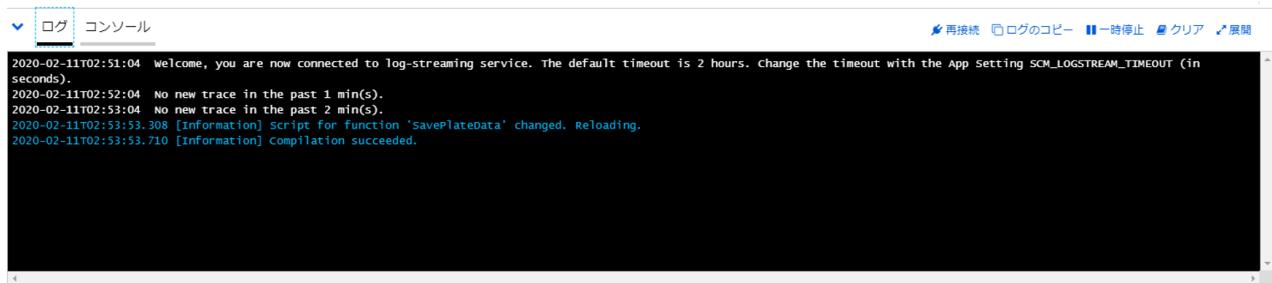
    outputDocument = data;

    log.LogInformation("license plate data to Azure Cosmos DB: " +
data.ToString());
}

```

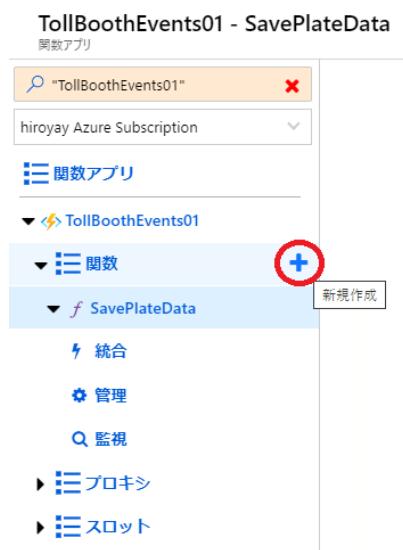
### 3. 「保存」をクリック

### 4. ログに **Comilation succeeded** メッセージが表示されることを確認



## Task 5: 手動検証データを Cosmos DB へ保存する関数の作成

### 1. 関数メニューの「+」をクリックし、新しい関数を追加

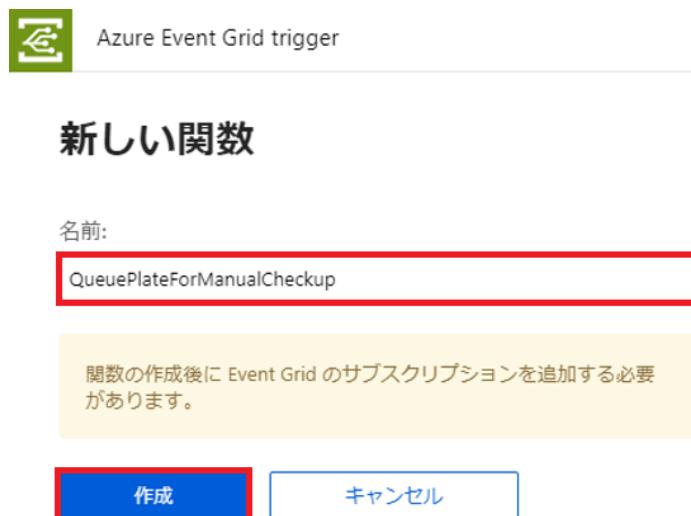


### 2. 先の手順と同じくテンプレートの選択画面が表示

検索ボックスに「**event grid**」と入力し、表示されたテンプレートの候補から「**Azure Event Grid**

trigger」をクリック

- 名前に **QueuePlateForManualCheckup** と入力し「作成」をクリック



- QueuePlateForManualCheckup** 関数が作成されたことを確認

## Task 6: 関数への Event Grid サブスクリプションの追加

- 「Event Grid サブスクリプションの追加」をクリック
- 「イベント サブスクリプションの作成」ブレードが表示されるので、以下の構成オプションを指定
  - 名前 (**queueplateformanualcheckupsub**)
  - イベント スキーマ (イベント グリッド キーマを選択)
  - トピックの種類 (先の手順で作成した Event Grid トピックを選択)
  - イベントの種類 (イベントの種類のフィルターに **QueuePlateForManualCheckup** と入力)
  - エンドポイントの詳細 (QueuePlateForManualCheckup 関数)

イベント サブスクリプションの作成

Basic フィルター 追加の機能

イベント サブスクリプションでは、トピック リソースから発生するイベントをリッスンし、それらのイベントをエンドポイント リソースに送信します。詳細情報

イベント サブスクリプションの詳細

名前: queueplateformmanualcheckupsub

イベント スキーマ: イベント グリッド スキーマ

トピックの詳細

宛先にイベントをプッシュするトピック リソースを選択します。詳細情報

トピックの種類: Event Grid トピック

トピックのリソース: TollBoothEventGrid01 (変更)

イベントの種類

宛先にプッシュするイベントの種類を選択します。詳細情報

イベントの種類のフィルター: queuePlateForManualCheckup (変更) や イベントの種類の追加

エンドポイントの詳細

イベントを受信するイベント ハンドラーを選択します。詳細情報

エンドポイントのタイプ: Azure 関数 (変更)

エンドポイント: QueuePlateForManualCheckup (変更)

作成

3. 「作成」をクリック

### Task 7: 関数への Cosmos DB 出力の追加

1. 「QueuePlateForManualCheckup」関数を展開し「統合」をクリック
2. 「+ 新しい出力」をクリックし、出力リストから「Azure Cosmos DB」を選択し「選択」をクリック
3. データベース名に「**LicensePlate**」、コレクション名に「**NeedsManualReview**」を入力
4. Azure Cosmos DB アカウント接続で SavePlateData 関数で作成した接続を選択

## Azure Cosmos DB output

ドキュメント パラメーター名 ⓘ  
outputDocument

データベース名 ⓘ  
LicensePlates

コレクション名 ⓘ  
NeedsManualReview

Azure Cosmos DB アカウント接続 ⓘ [値の表示](#) [新規](#)  
tollboothdb01\_DOCUMENTDB

Collection throughput (optional) ⓘ  
Collection throughput (optional)

パーティション キー (省略可能) ⓘ  
パーティション キー (省略可能)

**保存** **キャンセル**

**+ ドキュメント**

## 5. 「保存」をクリック

## Task 8: run.csx へのコードの記述

1. QueuePlateForManualCheckup をクリックし、run.csx を開く

2. 以下のコードを記述

```
#r "Microsoft.Azure.EventGrid"
#r "Newtonsoft.Json"

using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using Microsoft.Azure.EventGrid.Models;

public static void Run(EventGridEvent eventGridEvent, out object
outputDocument, ILogger log)
{
    log.LogInformation(eventGridEvent.Data.ToString());

    var data = eventGridEvent.Data as JObject;

    var resolvedFlg = "resolved";
    data[resolvedFlg] = false;

    outputDocument = data;

    log.LogInformation("manually verified to Azure Cosmos DB: " +
data.ToString());
}
```

## 3. 「保存」をクリック

4. ログに **Comlation succeeded** メッセージが表示されることを確認

## Exercise 4: Application Insights による Azure Functions の監視

所要時間 : 45分

Application Insights は Azure Functions と統合し、関数の堅牢な監視を提供します。

この演習では、新しい Application Insights をプロビジョニングし、テレメトリを送信するように Function App を構成します。

演習中に Computer Vision API のリクエストへのレート制限を行うよう価格レベルを変更します。意図的な遅延により、関数の応答時間が大幅に増加するため、動的スケーリングが開始されサーバーの割り当て数が増加します。ライブ メトリックス ビューを使用することで、これらすべてをリアルタイムで確認することができます。

### 参考情報

- Azure Functions の監視  
<https://docs.microsoft.com/azure/azure-functions/functions-monitoring>
- Live Metrics Stream での監視と診断  
<https://docs.microsoft.com/azure/azure-monitor/app/live-stream>

### Task 1: Application Insights インスタンスのプロビジョニング

1. Azure ポータル (<https://portal.azure.com>) を開く
2. 「+リソースの作成」をクリックし、「DevOps」 - 「Application Insights」を選択

## 新規

The screenshot shows the Azure Marketplace search results for 'Application Insights'. A red box highlights the first result, 'Application Insights', which has a purple icon featuring a lightbulb. The result includes the text 'クイック スタートとチュートリアル' (Quick Start and Tutorial). Other results listed include 'DevOps Project', 'Docker Enterprise - [17.06.2] (Preview)', 'IKAN ALM 5.8 demo (Preview)', 'Ansible Tower (Preview)', 'SaltStack Enterprise (Preview)', and 'Chef Automate (Preview)'. The left sidebar shows categories like '作業の開始', 'AI + Machine Learning', and 'DevOps', with 'DevOps' currently selected.

3. Application Insights ブレードで「**作成**」をクリック

4. Application Insights の「基本」ブレード内で、以下の構成オプションを指定

- リソース グループ（この演習で使用しているリソース グループ）
- 名前 (**TollBoothMonitor**)
- 地域（リソース グループと同じ地域を選択）

**Application Insights**  
Web アプリのパフォーマンスおよび使用状況の監視

基本 タグ 確認および作成

Application Insights リソースを作成して、ライブ Web アプリケーションを監視します。Application Insights を使用すると、複雑な分散アーキテクチャのすべてのコンポーネントと依存関係にわたって、アプリケーションを完全に監視できます。これには、問題を診断し、ユーザーがアプリで実際に何をしているかを把握するのに役立つ強力な分析ツールが含まれています。パフォーマンスと使いやすさを継続的に向上させるのに役立つように設計されています。.NET、Node.js、Java EE などの多様なプラットフォームのアプリや、オンプレミス、ハイブリッド、パブリック クラウドでホストされるアプリで機能します。[詳細情報](#)

プロジェクトの詳細  
デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソース グループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション \* ① [REDACTED] ▾  
リソース グループ \* ① **HOL-2020-03-Serverless-RG** ▾  
[新規作成](#)

インスタンスの詳細  
名前 \* ① **TollBoothMonitor** ✓  
地域 \* ① **(米国) 米国西部 2** ▾

**確認および作成** ▶ [« 前へ](#) [次: タグ >](#)

5. 「確認および作成」をクリック

6. 「作成」をクリック

7. プロビジョニング完了後、「リソースに移動」をクリック

## ✓ デプロイが完了しました

デプロイ名: Microsoft.AppInsights  
サブスクリプション: [\[REDACTED\]](#)  
リソース グループ: **HOL-2020-03-Serverless-RG**

▼ 展開の詳細 ([ダウンロード](#))

^ 次の手順

[リソースに移動](#)

## Task 2: Function App での Application Insights の有効化

1. Application Insights の管理ブレードへ移動

2. 「概要」ブレードで「インストルメンテーション キー」をコピー

**TollBoothMonitor**  
Application Insights | ディレクトリ: Microsoft

検索 (Ctrl+ /) ▶ [アプリケーション ダッシュボード](#) [作業の開始](#) [検索](#) [ログ](#) [リソース グループの監視](#) [フィードバック](#) [お気に入り](#) [クリップボードにコピー](#) [\[削除\]](#)

**概要**

リソース グループ ([変更](#)): **HOL-2020-03-Serverless-RG** インストルメンテーション キー: **2b49e8dd-8799-42f1-9c3a-86ebe1ee9e88** [\[クリップ\]](#)  
場所: 米国西部 2 接続文字列: [InstrumentationKey=2b49e8dd-8799-42f1-9c3a-86e...](#)  
サブスクリプション ([変更](#)): [\[REDACTED\]](#)

タグ ([変更](#)): タグを追加するにはここをクリック

3. 写真の処理を行う ProcessImage 関数を含む Function App へ移動

4. 概要ブレードで「構成」をクリック

The screenshot shows the Azure portal's function app overview for 'TollBoothFunctionApp01'. On the left, there's a navigation pane with 'hiroyay Azure Subscription' selected. Under 'Function App', 'TollBoothFunctionApp01' is expanded, showing 'Functions (Read-only)', 'Proxy (Read-only)', and 'Slots'. The 'Configuration' button is highlighted with a red box at the bottom of the main content area.

5. 「アプリケーション設定」の画面で「+ 新しいアプリケーション設定」をクリック

6. 以下の名前/値で設定を追加

- 名前 (**APPINSIGHTS\_INSTRUMENTATIONKEY**)
- 値 (コピーしたインストルメンテーション キー)

The screenshot shows the 'Application settings' blade. It has tabs for 'Application settings' and 'General settings'. Under 'Application settings', there's a table with one row for 'APPINSIGHTS\_INSTRUMENTATIONKEY'. The 'Value' column contains a placeholder: '表示するには、上記の [値] アプリの構成' (Show by referencing the [value] in the app's configuration). This row is highlighted with a red box.

7. 「保存」をクリック



### Task 3: ライブ メトリックス ストリームを使用したリアルタイム監視

1. Function App の「概要」ブレードで「Application Insights」をクリック

TollBoothFunctionApp01

概要

状態: Running

サブスクリプション: hiroyay Azure Subscription

プラットフォーム機能

停止 スワップ 再開 発行プロファイルの取得

構成済みの機能

関数アプリの設定 構成 Application Insights

2. Application Insights のメニューから「ライブ メトリックス ストリーム」をクリック

TollBoothMonitor

Application Insights | ディレクトリ: Microsoft

検索 (Ctrl+ /)

- 概要
- アクティビティ ログ
- アクセス制御 (IAM)
- タグ
- 問題の診断と解決

調査

- アプリケーション マップ
- スマート検出
- ライブ メトリックス ストリーム

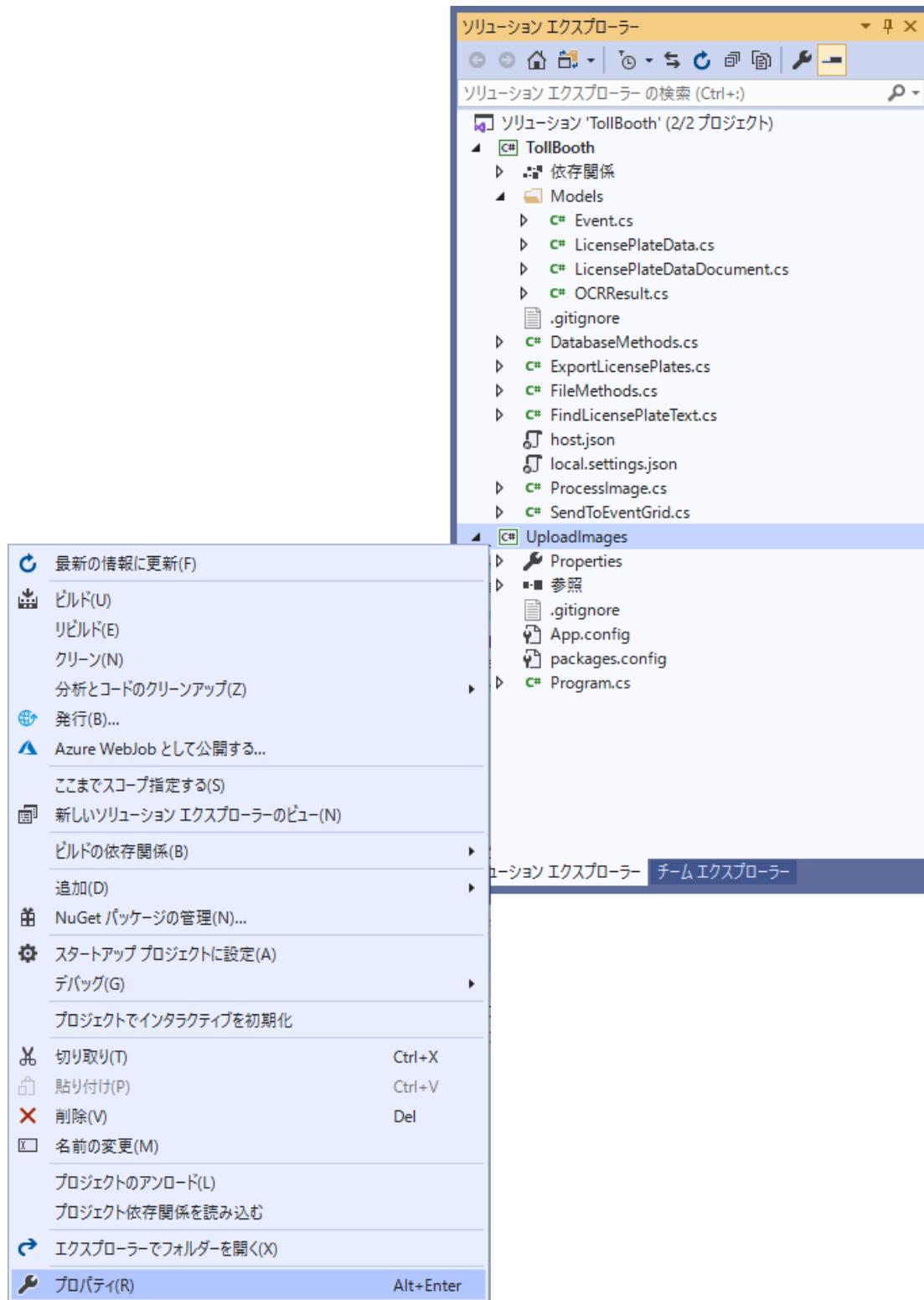
検索

可用性

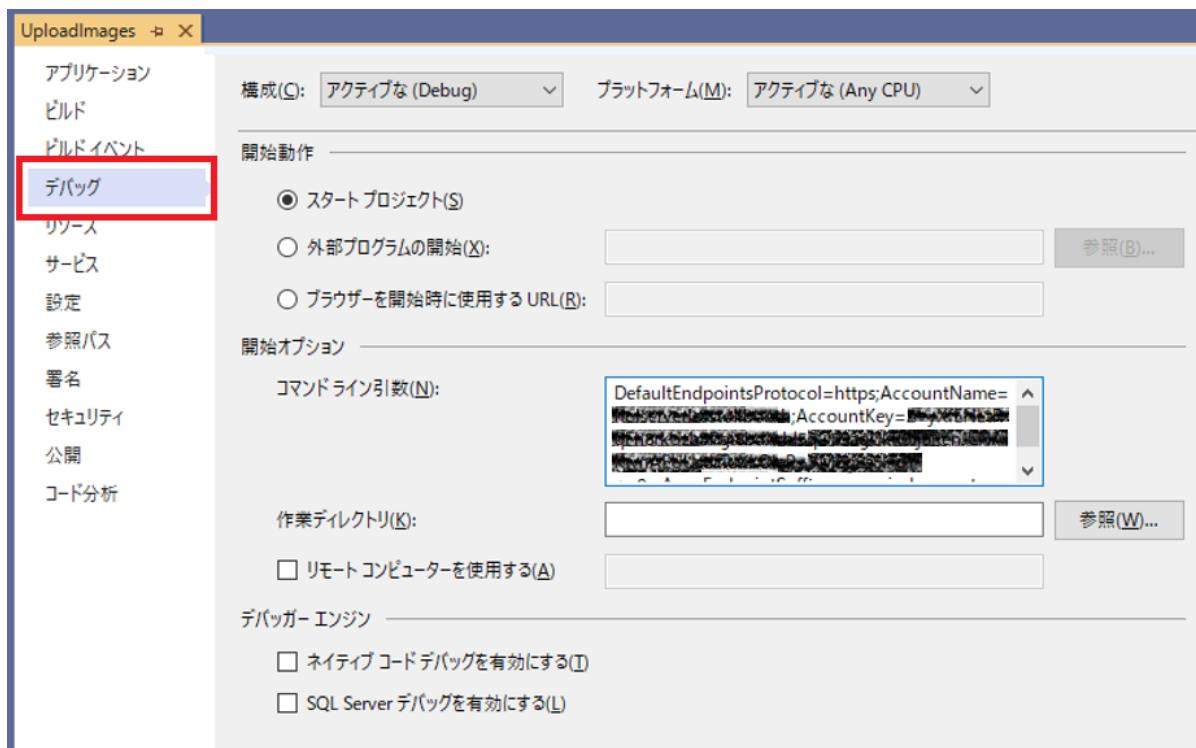
3. Visual Studio を起動し「TollBooth」ソリューションを開く

TollBooth ソリューションの UpdateImages プロジェクトは指定した Blob ストレージへ画像のアップロードを行うコンソール アプリケーションです。アプリケーションを実行するたびに必要な接続時文字列をパラメータとして追加するために、プロジェクトのプロパティで接続文字列を指定します。

#### 4. 「UploadImages」を右クリックし、表示されるメニューより「プロパティ」をクリック



#### 5. 「デバッグ」をクリックし、「コマンドライン引数」にストレージ アカウントへの接続文字列を指定



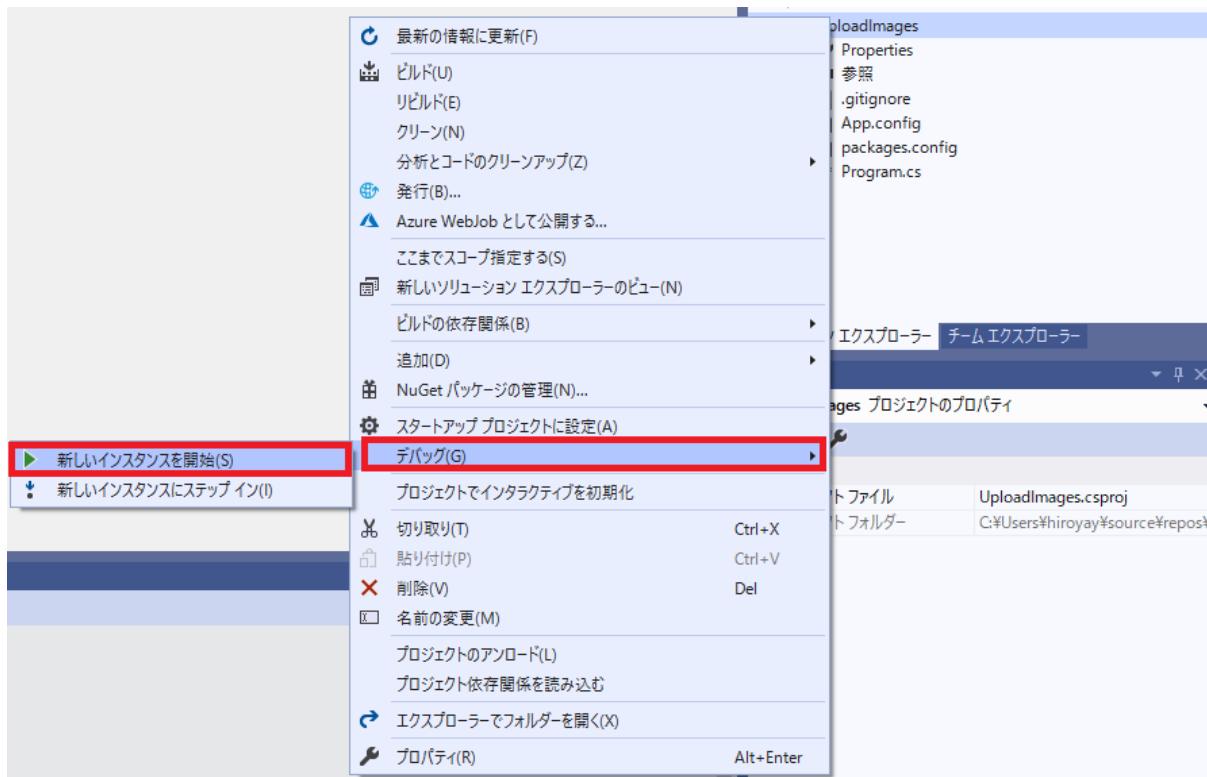
ストレージ アカウントへの接続文字列は「**アクセス キー**」メニューから取得  
(先の手順で作成した images コンテナーを含むストレージ アカウントを選択)



## 6. 変更を保存



## 7. 「UpdateImages」プロジェクトを右クリック メニューより「デバッグ」 - 「新しいインスタンスを開始」を選択

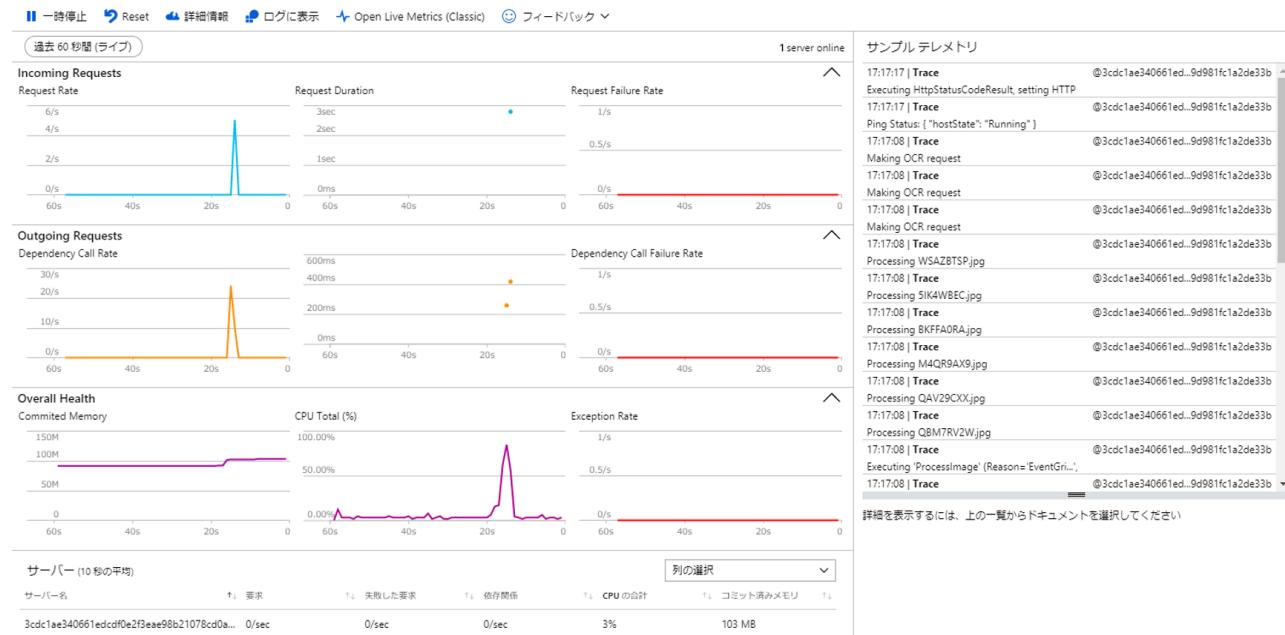


## 8. コンソール ウィンドウが表示されるので、「1」を入力し「Enter」キーを押下

```
C:\Users\hiroya\source\repos\MCW-Serverless-architecture\TollBooth\UploadImages\bin\Debug\UploadImages.exe
Enter one of the following numbers to indicate what type of image upload you want to perform:
  1 - Upload a handful of test photos
  2 - Upload 1000 photos to test processing at scale
1
Uploading images
Uploaded image 1: YU0Z107M.jpg
Uploaded image 2: Q74ANOX3.jpg
Uploaded image 3: ZS3J4FQU.jpg
Uploaded image 4: 3E7S90K3.jpg
Uploaded image 5: LYP08D7C.jpg
Uploaded image 6: PF7FD002.jpg
Uploaded image 7: C8HT4AAS.jpg
Uploaded image 8: E35XRONC.jpg
Uploaded image 9: WBYZ964W.jpg
Finished uploading images
```

指定したストレージ アカウントの images コンテナーに画像ファイルをアップロード  
ファイルをアップロード後はデバッグを終了し、コンソール アプリケーションを終了

## 9. ライブ メトリックス ストリームを表示しているブラウザへ画面を切替 テレメトリを受信しオンライン サーバー数、受信リクエストレート、CPU プロセス量などを表示



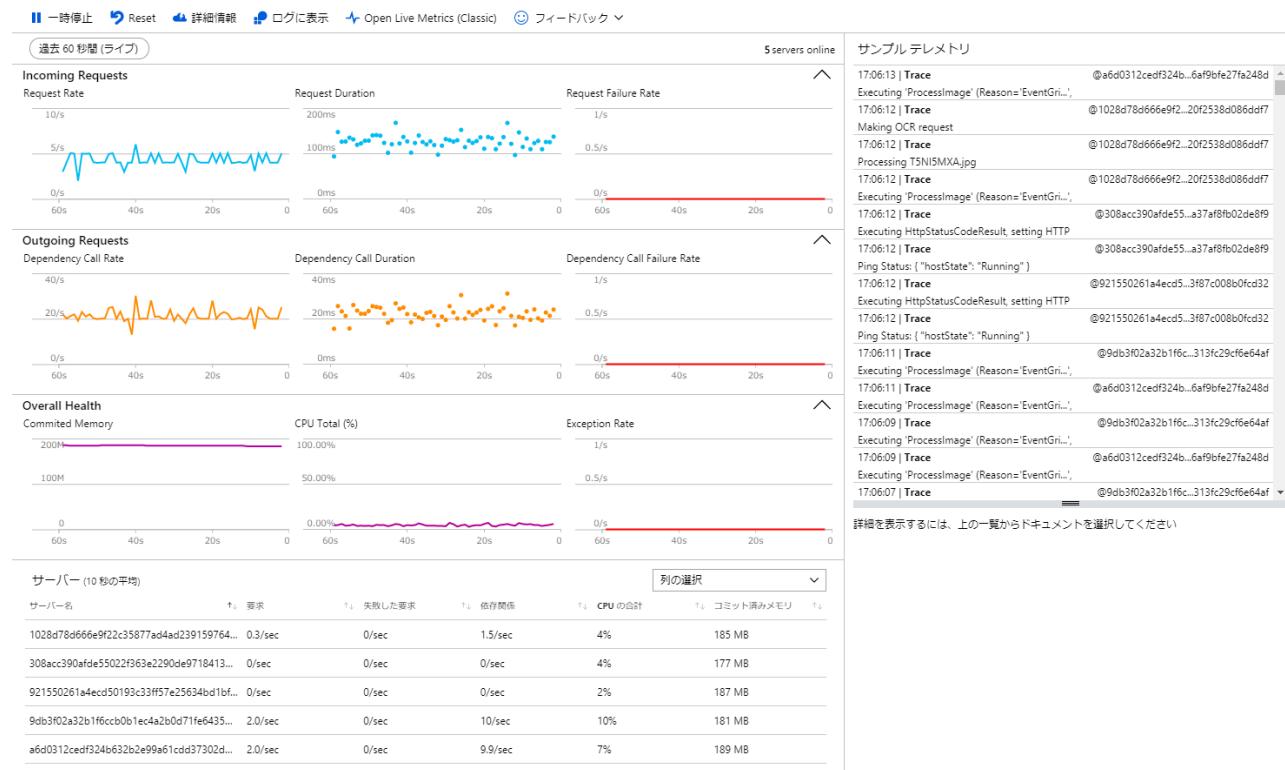
10. Visual Studio へ戻り、「**UpdateImages**」プロジェクトを右クリック  
メニューより「デバッグ」 - 「新しいインスタンスを開始」を選択

11. コンソール ウィンドウで、「2」 を入力し「Enter」キーを押下

```
C:\Users\hiroyay\source\repos\MCW-Serverless-architecture\TollBooth\UploadImages\bin\Debug\UploadImages.exe
Enter one of the following numbers to indicate what type of image upload you want to perform:
  1 - Upload a handful of test photos
  2 - Upload 1000 photos to test processing at scale
2
Uploading images
Uploaded image 1: PKVHQTH1.jpg
Uploaded image 2: 6DLFR1LG.jpg
Uploaded image 3: SMAHI4KC.jpg
Uploaded image 4: E5GIE7AV.jpg
Uploaded image 5: LP7MC0P.jpg
Uploaded image 6: TRRBJRWE.jpg
Uploaded image 7: GMUYT9BU.jpg
Uploaded image 8: ASDUN8X8.jpg
Uploaded image 9: D04B48EO.jpg
Uploaded image 10: UVXOMI3W.jpg
Uploaded image 11: JCC2ZMEL.jpg
Uploaded image 12: OMMTAHBV.jpg
Uploaded image 13: 4S8SXVII.jpg
Uploaded image 14: HFD8LK11.jpg
Uploaded image 15: YOLL64W0.jpg
Uploaded image 16: YSIWSMUON.jpg
Uploaded image 17: TC64KL9X.jpg
Uploaded image 18: 24VHID9E.jpg
Uploaded image 19: EU5T50Y9.jpg
Uploaded image 20: 6H3E81VW.jpg
Uploaded image 21: 00PJNORN.jpg
Uploaded image 22: RXSCUOSV.jpg
```

1,000 個の画像ファイルをアップロード

12. ライブ メトリックス ストリーム ウィンドウに戻りアクティビティを確認  
リクエスト モニターの安定したリズム、200 ミリ秒未満のホーリング リクエスト期間などプロセス  
が効率的に実行されることを確認



ファイルのアップロード後はデバッグを終了し、コンソール アプリケーションを終了

## Task 4: Azure Functions の動的スケーリングの監視

1. Computer Vision API サービスの管理ブレードへ移動
2. メニューの「価格レベル」を選択し、「F0 Free」を選択後、「選択」をクリック

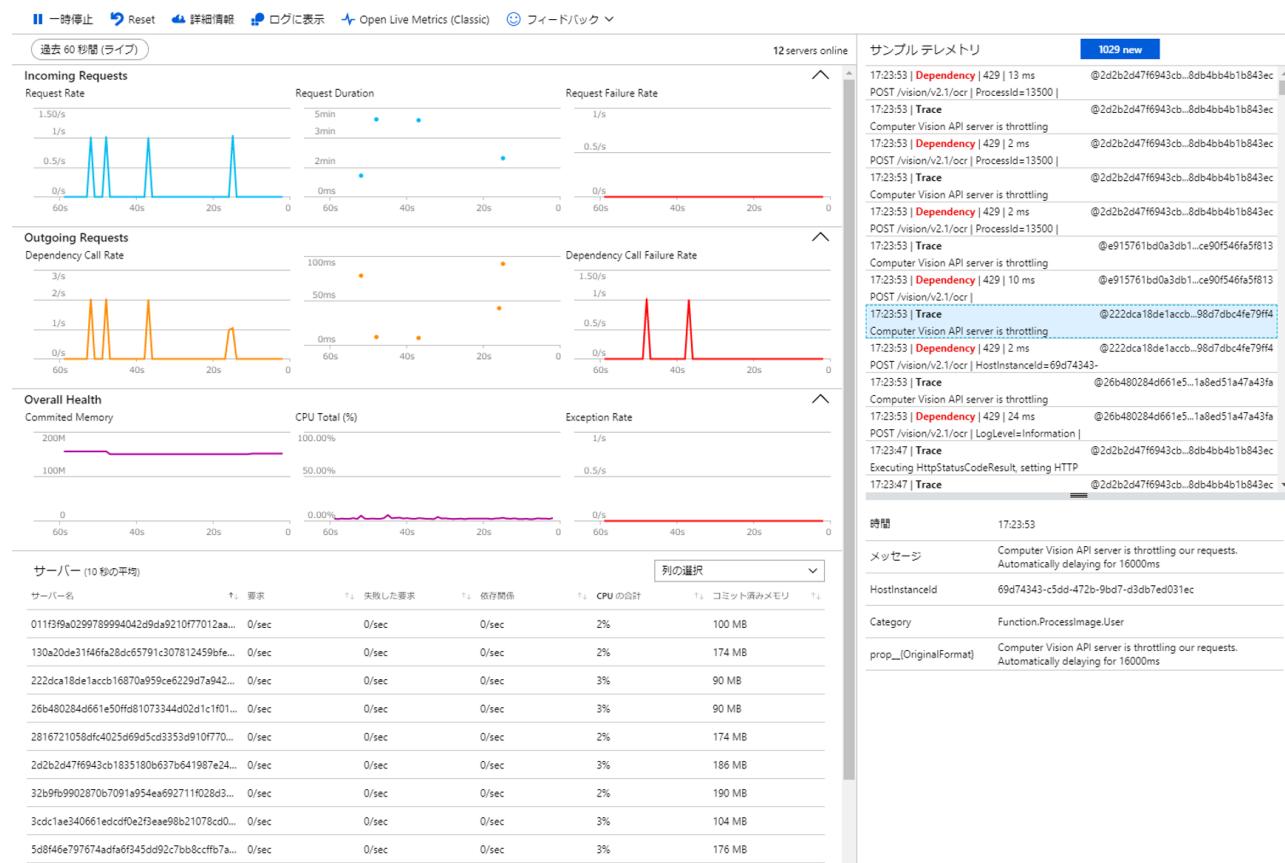
価格レベル	料金
F0 Free	0.00 1 回の呼び出しあたり USD (推定)
S1 Standard	112.00 JPY/1000 回の呼び出し (推定) の開始

**選択**

これ

により OCR サービスの呼び出しが 1 分あたり 10 に制限

3. Visual Studio に戻り 「UpdateImages」プロジェクトを右クリック  
メニューより「デバッグ」 - 「新しいインスタンスを開始」を選択
4. ライブ メトリックス ストリーム ウィンドウに戻りアクティビティを確認



リクエスト期間が時間とともに増加し、これと共にオンライン サーバー台数が増加  
サーバーが増加するたびにサンプル テレメトリに **Generateing 2 job function (s)** が表示  
Computer Vision API が要求を調整している復元ポリシーによって記録されたメッセージも表示  
(これは 429 から返された応答コードで認識可)

5. しばらく実行した後、UpdateImages コンソールを閉じて写真のアップロードを停止  
※ Ctrl+C を押下でアプリケーションは終了

6. Computer Vision API の管理ブレードへ戻り、価格レベルを **S1** に変更

## Exercise 5: Azure Cosmos DB 内のデータ探索

所要時間：15分

Azure ポータルからデータ エクスプローラを使用し Cosmos DB に保存されたデータを確認します。

Azure Cosmos DB SQL API アカウントでは、JSON クエリ言語として SQL クエリがサポートされています。

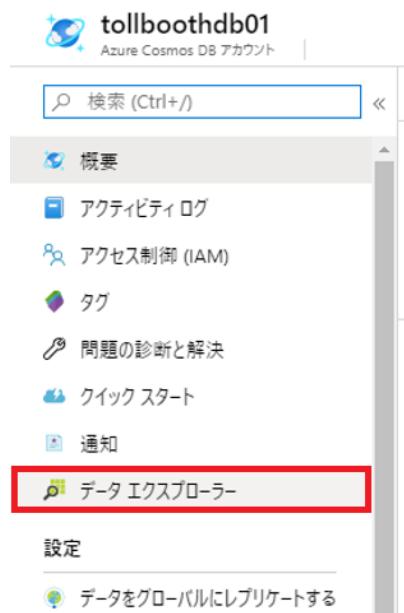
### 参考情報

- Azure Cosmos DB の概要  
<https://docs.microsoft.com/azure/cosmos-db/introduction>
- SQL クエリの使用を開始する  
<https://docs.microsoft.com/azure/cosmos-db/sql-query-getting-started>

### Task 1: Azure Cosmos DB データ エクスプローラの使用

## 1. Cosmos DB の管理ブレードへ移動

## 2. 左側のメニューより「データ エクスプローラー」をクリック



## 3. Processed コレクションを展開し、Items を選択、表示されるリストよりアイテムを選択

The screenshot shows the Azure portal interface for the 'LicensePlates' database. The 'Processed' collection is expanded, and the 'Items' sub-collection is selected, highlighted with a red box. A specific item, 'f51e83c3-cfee-41f4-95eb-c213d1fe5e56', is selected and highlighted with a red box. The details pane on the right shows the JSON representation of this item:

```

1  "fileName": "IJXK34LK.jpg",
2  "licensePlateText": "127 RFS",
3  "timeStamp": "2020-02-11T08:34:25.491919Z",
4  "licensePlateFound": true,
5  "exported": false,
6  "id": "f51e83c3-cfee-41f4-91c6-6245b0fc495d",
7  "_rid": "R+DTALhEk48CAAAAAAAA==",
8  "_self": "dbs/M+pTAA=/colls/M+pTALhEk48=/docs/M+pTALhEk48CAAAAAAAA==/",
9  "_etag": "\\"0600fe2e-0000-0800-0000-5e426a950000\\",
10  "_attachments": "attachments/",
11  "_ts": 1581410965
12
13

```

コレクションに追加された JSON ドキュメントを確認

## 4. 同様の手順で NeedsManualReview コレクション内のアイテムを確認

The screenshot shows the Azure portal interface for the 'LicensePlates' database. The 'NeedsManualReview' collection is expanded, and the 'Items' sub-collection is selected, highlighted with a red box. A specific item, '9bd320e7-0a00-41f4-95eb-c213d1fe5e56', is selected and highlighted with a red box. The details pane on the right shows the JSON representation of this item:

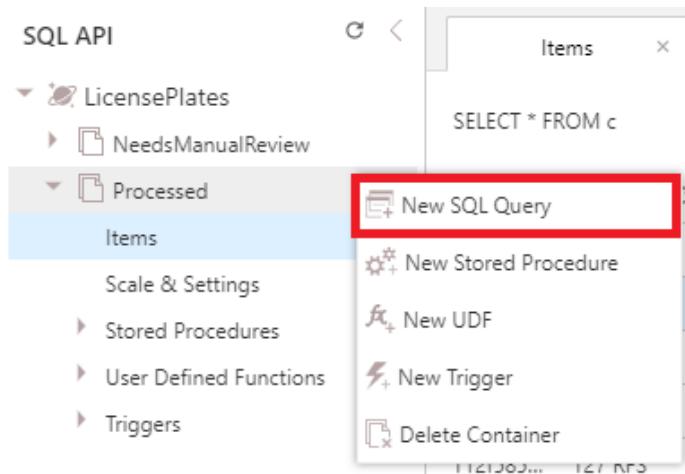
```

1  "fileName": "SUWY1208.jpg",
2  "licensePlateText": "",
3  "timeStamp": "2020-02-11T08:45:03.574562Z",
4  "licensePlateFound": false,
5  "resolved": false,
6  "id": "9bd320e7-0a00-41f4-95eb-c213d1fe5e56",
7  "_rid": "M+pTAO92zFQDAAAAAAA==",
8  "_self": "dbs/M+pTAA=/colls/M+pTAO92zFQ=/docs/M+pTAO92zFQDAAAAAAA==/",
9  "_etag": "\\"1d001b84-0000-0000-0000-5e4269900000\\",
10  "_attachments": "attachments/",
11  "_ts": 1581410704
12
13

```

## 5. Processed にマウスカーソルをホバーし表示される「...」をクリック

## 6. 「New SQL Query」を選択



## 7. クエリを実行し結果を確認

A screenshot of the Azure portal showing the results of a SQL query execution. The query is:

```
1 | SELECT * FROM c WHERE c.licensePlateText = "127 RFS"
```

The results pane shows a single JSON document:

```
{  
    "fileName": "IJJXK34LK.jpg",  
    "licensePlateText": "127 RFS",  
    "timeStamp": "2020-02-11T08:34:25.491919Z",  
    "licensePlateFound": true,  
    "exported": true,  
    "id": "f51e83c3-cfee-4136-91c6-6245b0fc495d",  
    "_rid": "M+pTALhEk48CAAAAAAAA==",  
    "_self": "dbs/M+pTAA=/colls/M+pTALhEk48=/docs/M+pTALhEk48CAAAAAAAA==/",  
    "_etag": "\"0900fce-0000-0800-0000-5e4503670000\"",  
    "_attachments": "attachments/",  
    "_ts": 1581581159  
},
```

WHERE 句に JSON メンバーを使用してフィルター条件を指定

## 8. 同様に New SQL Query を開き、クエリを実行し結果を確認

The screenshot shows the Azure Cosmos DB Query Explorer interface. At the top, there are tabs for 'Items', 'Query 1', and 'Query 2'. Below them, a query is entered: 'SELECT VALUE COUNT(1) FROM c WHERE c.exported = false'. The results tab is selected, showing a single row with the value '1017'.

Results    Query Stats

1 - 1

[	1017	]
---	------	---

コレクション内にあるエクスポートがされていないアイテム数をカウント  
この例では 1017 レコードが検出

## Exercise 6: データ エクスポート ワークフローの作成

所要時間：30分

データ エクスポートを行うワークフローを Logic App を使用して構成します。Logic App は定期的に実行され、ExportLicensePlates 関数を呼び出します。ExportLicensePlates 関数は Cosmos DB に格納されたアイテムの中から exported メンバーの値が `false` のものを CSV ファイルへ抽出し、Blob ストレージへ保存します。エクスポートするレコードがない場合は、Logic App からレコードがない旨を記載した電子メールを送信します。

### 参考情報

- Logic Apps を使用してワークフローを作成  
<https://docs.microsoft.com/azure/logic-apps/quickstart-create-first-logic-app-workflow>
- Logic Apps のコネクタ  
<https://docs.microsoft.com/azure/connectors/apis-list>
- Logic Apps の式で関数を使用するためのリファレンス ガイド  
<https://docs.microsoft.com/azure/logic-apps/workflow-definition-language-functions-reference>

### Task 1: Logic App の作成

- Azure ポータル (<https://portal.azure.com>) を開く
- 「+リソースの作成」をクリックし、「統合」 - 「Logic App」を選択

## 新規

The screenshot shows the 'New' blade in the Azure Marketplace. At the top, there's a search bar labeled 'Marketplace を検索'. Below it, there are two tabs: 'Azure Marketplace' and 'すべて表示' (All). Underneath are two more tabs: 'おすすめ' (Recommended) and 'すべて表示' (All). A red box highlights the first item in the recommended list: 'Logic App' with the sub-label 'クイック スタートとチュートリアル'. To the left of the main list, there's a sidebar with categories: '作業の開始', '最近作成', 'AI + Machine Learning', '分析', 'ブロックチェーン', 'Compute', 'コンテナー', 'データベース', '開発者ツール', 'DevOps', 'ID', '統合' (highlighted with a blue box), 'モノのインターネット (IoT)', 'メディア', 'Mixed Reality', '管理ツール', 'ネットワーキング', 'サービスとしてのソフトウェア (SaaS)', 'セキュリティ', 'ストレージ', and 'Web'. Each item has a small icon and a 'Quick Start and Tutorial' link.

### 3. Logic App の「基本」ブレードで、以下の構成オプションを指定

- リソース グループ（この演習で使用するリソース グループ）
- ロジック アプリ名（任意：英字, 数字, 括弧、ハイフン、アンダースコア、ピリオドの使用は可）
- 場所（リソース グループと同じ地域）
- Log Analytics（オフ）

## Logic App

基本 \* 確認および作成

**プロジェクトの詳細**

デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソース グループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション \*

リソース グループ \*  HOL-2020-03-Serverless-RG

[新規作成](#)

**インスタンスの詳細**

ロジック アプリ名 \*  TollBoothLogic

場所の選択

リージョン  統合サービス環境

場所 \*  米国西部 2

Log Analytics

[確認および作成](#) [Automation のテンプレートをダウンロードする](#)

4. 「確認および作成」をクリック
5. 「作成」をクリック
6. プロビジョニング完了後、「リソースに移動」をクリック

### ✔ デプロイが完了しました

 デプロイ名: Microsoft.EmptyWorkflow  
サブスクリプション:   
リソース グループ: HOL-2020-03-Serverless-RG

▽ 展開の詳細 ([ダウンロード](#))

へ 次の手順

[リソースに移動](#)

7. 作成した Logic App の **Logic App デザイナー**が表示

8. 「一般的なトリガーで開始する」セクションから「繰り返し」をクリック

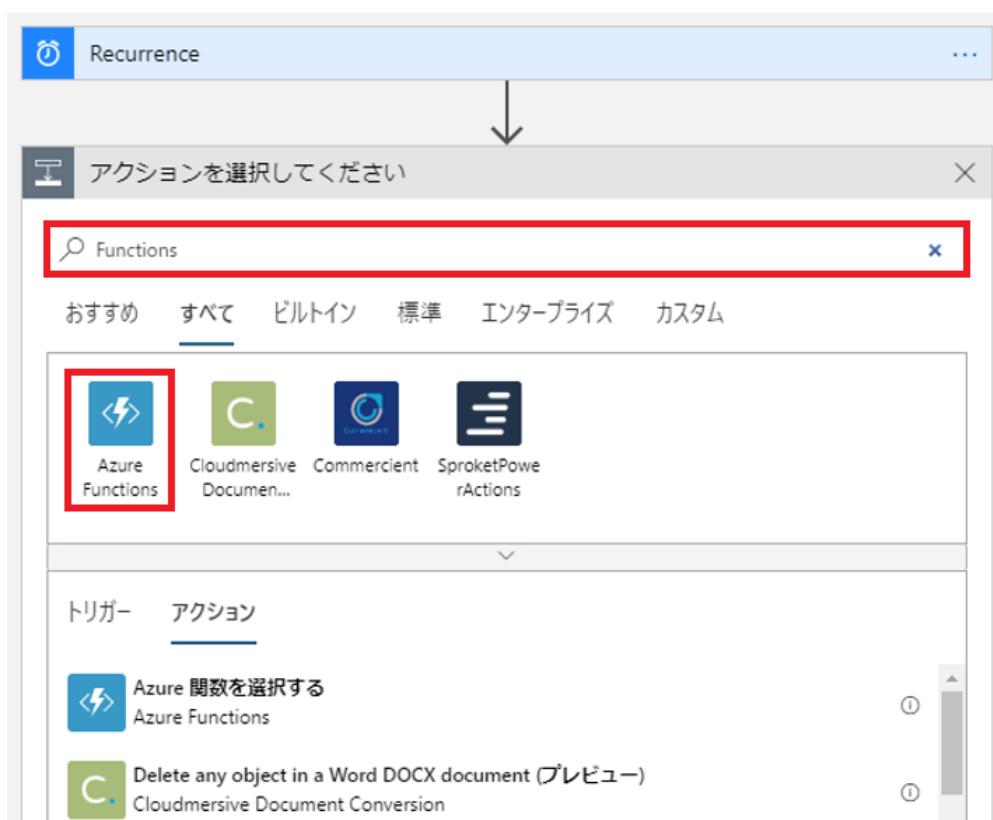


9. 「間隔」に「15」を入力、「頻度」を「分」に設定し、「新しいステップ」をクリック

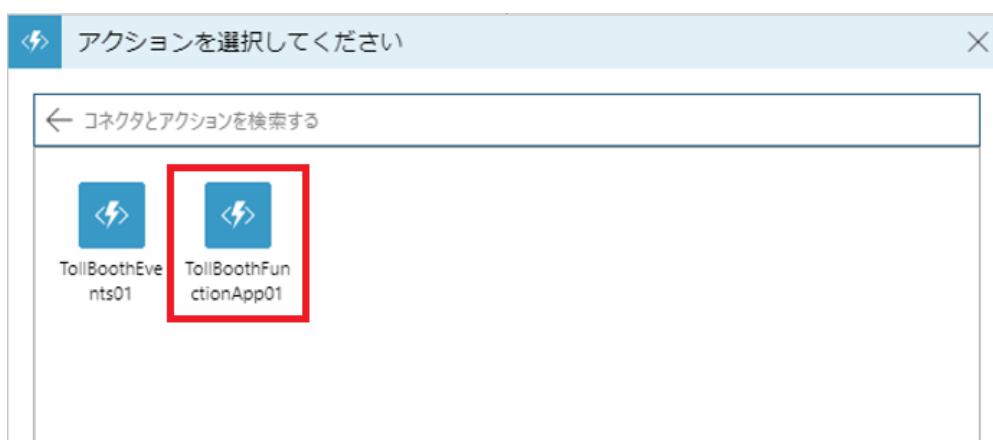


10. フィルター ボックスに **Function** と入力

表示される候補より「**Azure Functions**」コネクターを選択



11. ExportLicensePlates 関数を含む Function App を選択



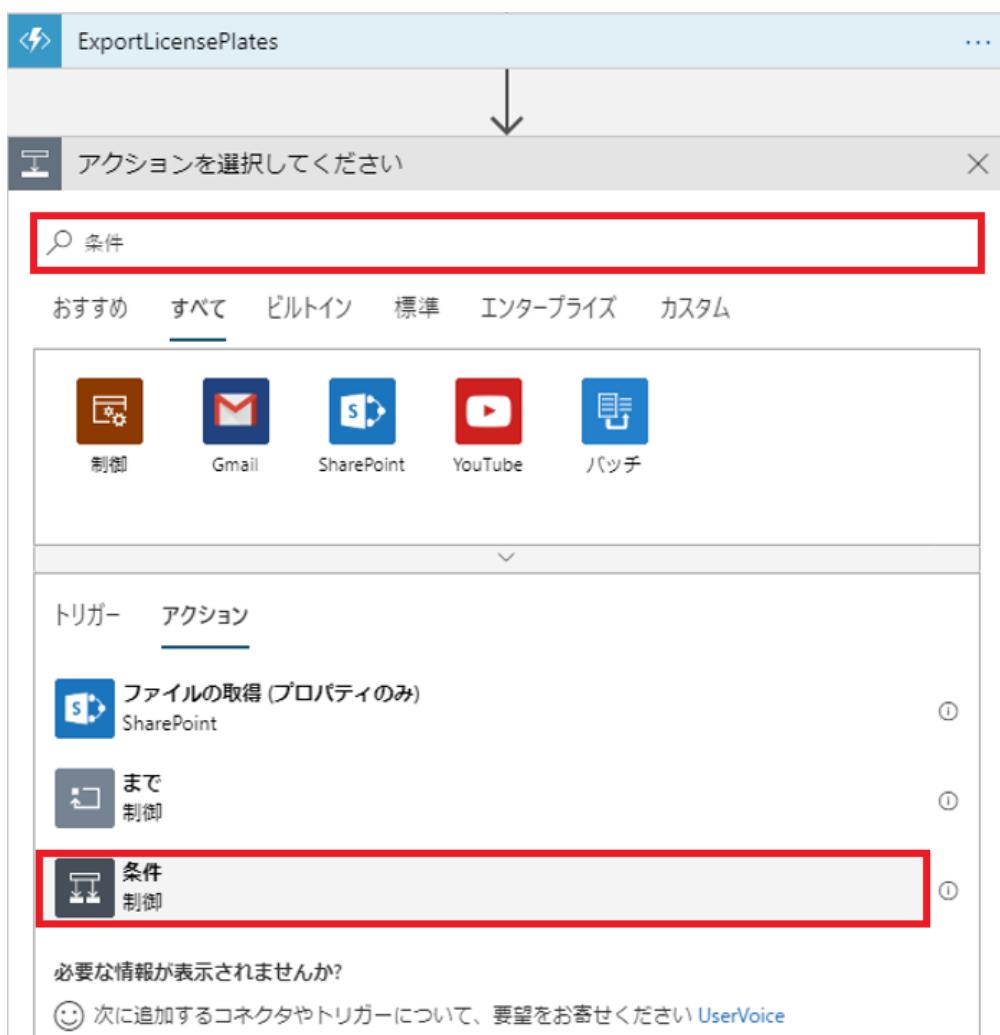
## 12. ExportLicensePlates を選択



## 13. 「新しいステップ」をクリック

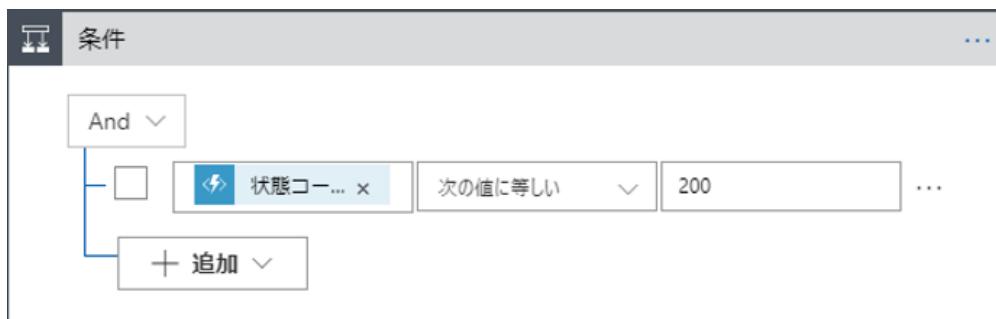
### 14. フィルター ボックスに「条件」と入力

表示される候補より「条件」を選択



## 15. 値フィールドで「状態コード」を選択

演算子で「次の値に等しい」を選択し、2 番目の値フィールドに「200」を入力



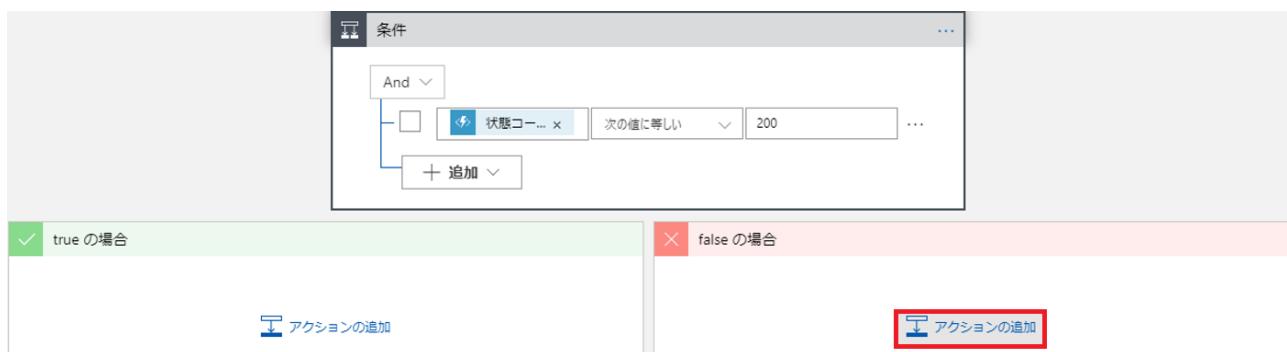
ExportLicensePlates 関数から返される状態コードを評価

関数でナンバー プレートが検出され、エクスポートされると 200,

エクスポートする必要があるナンバー プレートが検出されなかった場合 204 を送信

200 以外の状態コードが返された場合にメールを送信

16. ナンバー プレートが正常に返された場合は、実行する処理はないので **true の場合** は空白、  
**false の場合** の「アクションの追加」をクリック



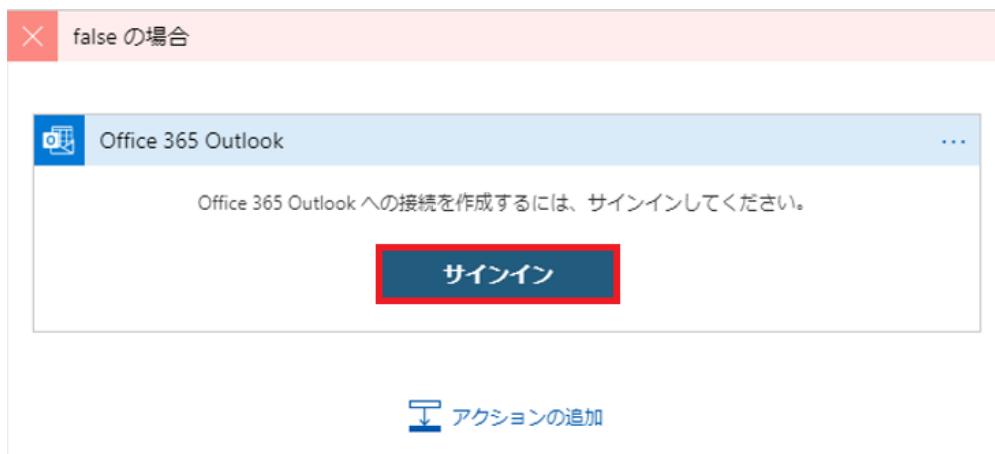
17. フィルター ボックスに「メールの送信」と入力

表示される候補より「**メールの送信(V2) Office 365 Outlook**」を選択

※ Gmail など他のアカウントでメール通知を行う場合は、該当のコントロールを選択

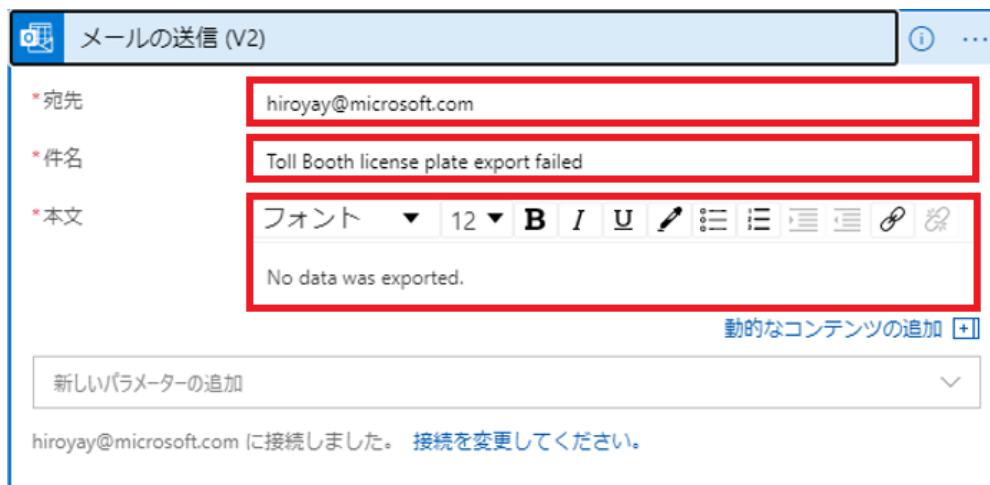


18. 「サインイン」をクリックし、選択したメール サービスへの接続を作成

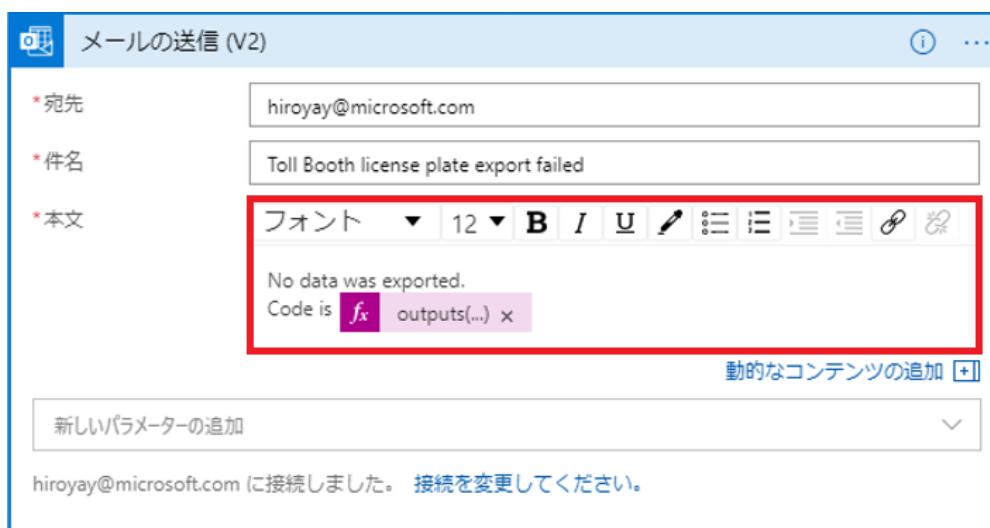


19. メール送信のフォームで、以下の構成オプションを指定

- 宛先（メールを通知を受信するユーザーのメール アドレス）
- 件名（**Toll Booth license plate export failed**）
- 本文（**No data was exported.**）

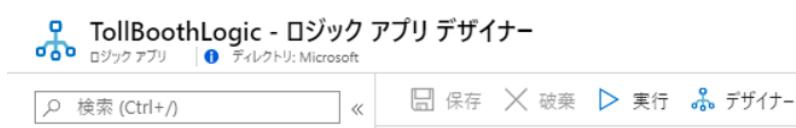


20. 「本文」で改行し「Code is」と入力し「動的なコンテンツの追加」をクリック  
「式」を選択し「outputs('ExportLicensePlates')['statusCode']」と入力し「OK」をクリック



動的なコンテンツを使用することで、他のコントロールから返されるパラメーターなどの取得  
が可

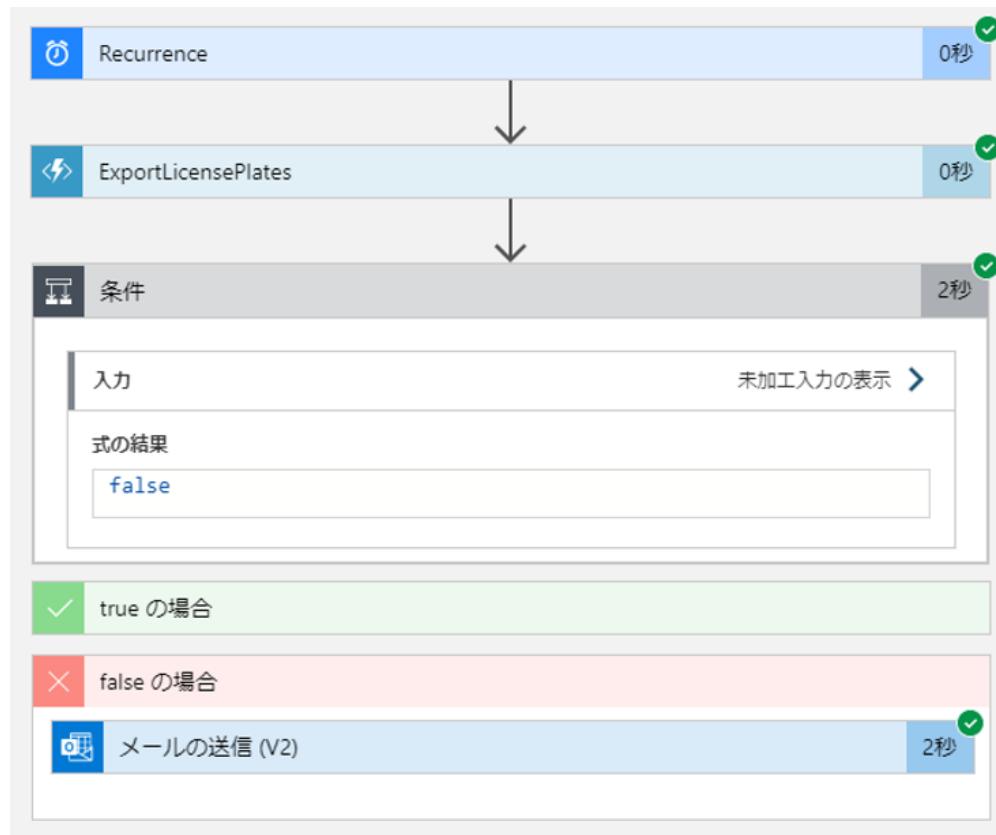
21. ツールバーの「保存」をクリック  
22. 「実行」をクリックし、Logic App を実行



### 23. Logic App デザイナー上でワークフローの各ステップの実行結果が表示

正常に実行された各ステップの横に緑色のチェックマークが表示

これを使用し、各ステップがどのように機能するかの確認が可



### 24. 関数にナンバー プレート検出のロジックが実装されていないためアラート メールを受信



### 25. Logic App の概要ブレードで「無効」をクリックしアプリを停止

## Exercise 7: Function App の CI/CD 構成

所要時間：45分

この演習では、CI/CD のために ProcessImage 関数を含む Function App を構成します。最初に Azure DevOps 組織とプロジェクトを作成し、Visual Studio からソリューションをソース管理に追加します。その後 Azure Pipelines から Azure Repos にコードがコミットされるとビルド、リリースを行うパイプラインを作成します。最後に新しいブランチを作成し、変更したコードをプッシュ、コード レビュー後にブランチをマージし、新しいバージョンの関数を Azure 環境に展開します。

### 参考情報

- Azure DevOps  
<https://azure.microsoft.com/services/devops/>
- What is Azure DevOps?  
<https://docs.microsoft.com/azure/devops/user-guide/what-is-azure-devops?view=azure-devops>
- What is Azure Repos?  
<https://docs.microsoft.com/azure/devops/repos/get-started/what-is-repos?view=azure-devops>
- What is Azure Pipelines?  
<https://docs.microsoft.com/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops>
- Use Azure Pipelines  
<https://docs.microsoft.com/azure/devops/pipelines/get-started/pipelines-get-started?view=azure-devops>
- Define approvals and checks  
<https://docs.microsoft.com/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-pass>
- Azure DevOps を使用した継続的デリバリー  
<https://docs.microsoft.com/azure/azure-functions/functions-how-to-azure-devops?tabs=csharp>
- YAML Schema reference  
<https://docs.microsoft.com/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema>

### Task 1: Azure DevOps 組織とプロジェクトの作成

Azure DevOps 組織を新規作成する場合

1. Web ブラウザーの新しいタブまたはインスタンスを起動  
「<https://azure.microsoft.com/services/devops/>」を開く
2. 「無料で始める >」をクリック

ホーム / サービス / Azure DevOps

# Azure DevOps

最新の一連の開発サービスを利用して、よりスマートに計画を立て、より効率的に共同作業を行い、より迅速に公開しましょう。

[無料で始める >](#)

[GitHub の使用を無料で開始する >](#)

アカウントを既にお持ちですか?

[Azure DevOps にサインイン >](#)

3. サインイン画面が表示されるので、Microsoft アカウントを使用してサインイン
4. 新規プロジェクト作成フォームで、以下の構成オプションを指定
  - a. Project name (**TollBooth**)
  - b. Project visibility (**Private**)
  - c. Country/region(**日本**)

The screenshot shows the 'Get started with Azure DevOps' setup page. At the top, it displays the Azure DevOps logo and the email address 'hiroya\_y@hotmail.com'. Below that, the heading 'Get started with Azure DevOps' is centered. The first input field is labeled 'Project name' with the value 'TollBooth'. The second input field is labeled 'Project visibility' with the value 'Private'. A note below these fields states: 'Choosing **Continue** means that you agree to our [Terms of Service](#), [Privacy Statement](#), and [Code of Conduct](#)'. There is a checkbox option: 'I would like information, tips, and offers about Azure DevOps and other Microsoft products and services. [Privacy Statement](#)' followed by a checked checkbox. The third input field is labeled 'Country/region' with the value '日本'. At the bottom right is a blue 'Continue' button.

5. 「Continue」をクリック
6. プロジェクトのサマリー画面が表示  
画面左上の「Azure DevOps」をクリックし、組織のトップページへ移動
7. 画面左下の「Organization Settings」をクリック

The screenshot shows the Azure DevOps organization settings interface. At the top left is the 'Azure DevOps' logo. Below it is a navigation bar with a blue icon and the text 'hiroyay'. The main content area has a header 'hiroyay0141' and tabs for 'Projects', 'My work items', and 'My pull requests'. A project card for 'TollBooth' is displayed. On the left, there's a 'What's new' section about Sprint 164 release notes and an 'Organization settings' button.

## 8. 「Users」を選択し「Add users」をクリック

The screenshot shows the 'Users' page under 'Organization Settings'. The left sidebar has a 'General' section with 'Overview', 'Projects', 'Users' (which is highlighted with a red box), 'Billing', 'Auditing', 'Global notifications', 'Usage', 'Extensions', and 'Azure Active Directory'. The main area shows a table of users. The first user listed is 'Hiroya Yamamoto' (hiroya@microsoft.com), with access level 'Early Adopter', date added '2017/3/29', and last accessed '2020/2/19'. There are buttons for 'Summary' and 'Add users' (which is highlighted with a red box). A notification bar at the top right says '1 session notification' with 'View all' and 'Dismiss all' buttons.

## 9. ユーザーの追加フォームでユーザー名、アクセス レベルを指定し「追加」をクリック

### Add new users

Users  
H [hiroyay@xxxxxxxxx](#) × +

Access level  
Basic

Add to projects  
  ▼

Send email invites

i You are inviting users from outside your directory. The user(s) [hiroya\\_y@hotmail.com](#) will need to click the link in their invitation e-mail to be able to access the organization and its resources. [Learn more](#)

Cancel Add

※組織外ユーザーの場合は、メールアドレスで指定

※組織外ユーザーは送信される招待メールからリンクをクリックしてアクセス



Azure DevOps

## You've been invited to Azure DevOps

Join your organization at [dev.azure.com/hiroyay](https://dev.azure.com/hiroyay)

Join now

### New to Azure DevOps?

Unlimited, free, private code repositories

Use IntelliJ, Visual Studio Code, Xcode, or your favorite editor

Track issues, user stories, feedback, and more

Enterprise grade services, priced to be small-team friendly

Develop with any language and OS

Continuous integration and delivery on Linux, macOS, and Windows

Get started

※アクセス レベルは Visual Studio サブスクリプションをお持ちの方は **Visual Studio Subscriber** でも可

## 10. ユーザーが追加されたことを確認

The screenshot shows the 'Users' page in the Azure Active Directory (AAD) portal. At the top, there are tabs for 'All users' and 'Group rules'. On the right, there is a 'Export users' button. Below the tabs, there is a search bar labeled 'Filter users' and dropdown menus for 'Access Level' and 'AAD User Type'. A 'Summary' button is highlighted in blue, and an 'Add users' button is also visible. The main table lists two users:

	Name	Access Level	Date Added	Last Accessed
<input type="checkbox"/>	Hiroya Yamamoto hiroyay@microsoft.com	Early Adopter	2017/3/29	2020/2/19
<input type="checkbox"/>	[Blurred Profile Picture] [Blurred Name]	Early Adopter	2020/2/19	Never

## 11. 画面左上の「Azure DevOps」をクリック 「TollBooth」をクリックし、プロジェクトのサマリ画面へ移動

既存の Azure DevOps 組織を使用し、プロジェクトを作成する場合

1. Web ブラウザーの新しいタブまたはインスタンスを起動  
[「https://azure.microsoft.com/services/devops/」を開く](https://azure.microsoft.com/services/devops/)
2. 「Azure DevOps にサインイン >」をクリック

ホーム / サービス / Azure DevOps

# Azure DevOps

最新の一連の開発サービスを利用して、よりスマートに計画を立て、より効率的に共同作業を行い、より迅速に公開しましょう。

[無料で始める >](#)

[GitHub の使用を無料で開始する >](#)

アカウントを既にお持ちですか?

[Azure DevOps にサインイン >](#)

3. 「+ New Project」をクリック

The screenshot shows the 'Projects - Home' page in the Azure DevOps portal. At the top, there is a search bar and a '+ New project' button. The main area displays a list of projects under the heading 'hiroyay'. There are three projects listed:

- Dotnet-Core-Aspnet-DemoApp (Team Foundation Version Control (TFVC) 使用)
- Dotnet-Core-Aspnet-HelloWorld-CSharp
- ASPNETCore-Sample-On-Container

4. プロジェクトの作成フォームで、以下の構成オプションを指定

- a. Project name (**TollBooth**)
- b. Visibility (**Private**)
- c. Version control (**Git**) d. Work item process (**Scrum**)

Create new project ×

Project name \* TollBooth ✓

Description

Visibility

Public ⓘ  
Anyone on the internet can view the project. Certain features like TFVC are not supported.

Enterprise  
[Members of your enterprise](#) can view the project.

Private  
Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

Advanced

Version control ⓘ Git

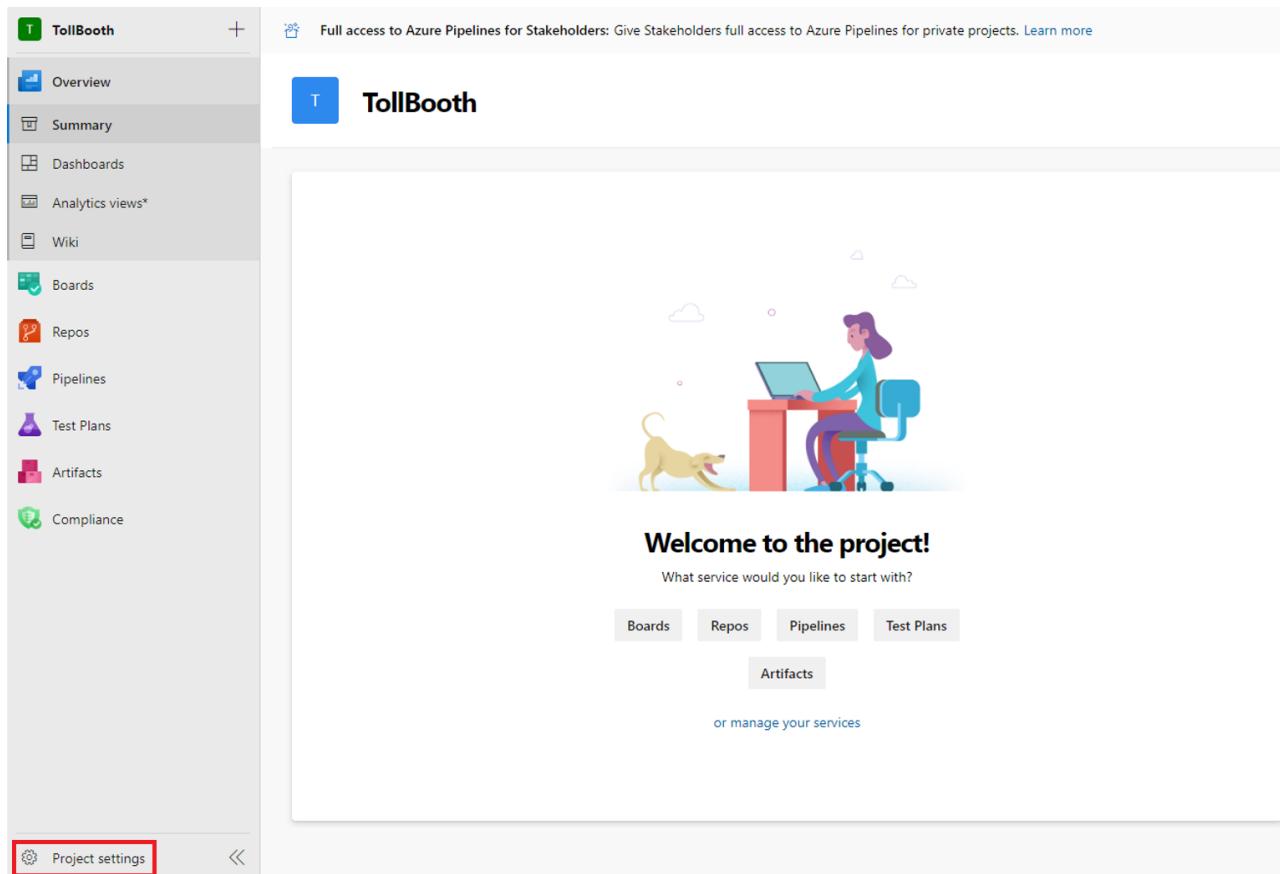
Work item process ⓘ Scrum

Cancel Create

5. 「Create」をクリック

**Task 2:** プロジェクトへのユーザーの追加

1. プロジェクトのサマリー画面の左下で「Project Settings」をクリック



## 2. 「Teams」を選択し「TollBooth」をクリック

The screenshot shows the 'Project Settings' page for the 'TollBooth' project, specifically the 'Teams' section. The 'Teams' option in the sidebar is highlighted with a red box. The main content area shows the 'TollBooth Team' settings. It includes a team icon, the team name, a description, and links to Notifications, Dashboards, and Iterations and Area Paths. Below this is a 'Members' table. The table has columns for Name, Type, and Username or scope. It shows one member: Hiroya Yamamoto (Admin, aad user, hiroyay@microsoft.com). There is a search bar at the top right and an 'Add' button at the bottom right, which is also highlighted with a red box.

## 3. 「Add」をクリックし、組織に追加したユーザーを入力し「Save」をクリック

**TollBooth Team**  
The default project team.  
Relevant links: [Notifications](#) | [Dashboards](#) | [Iterations and Area Paths](#)

Members Settings

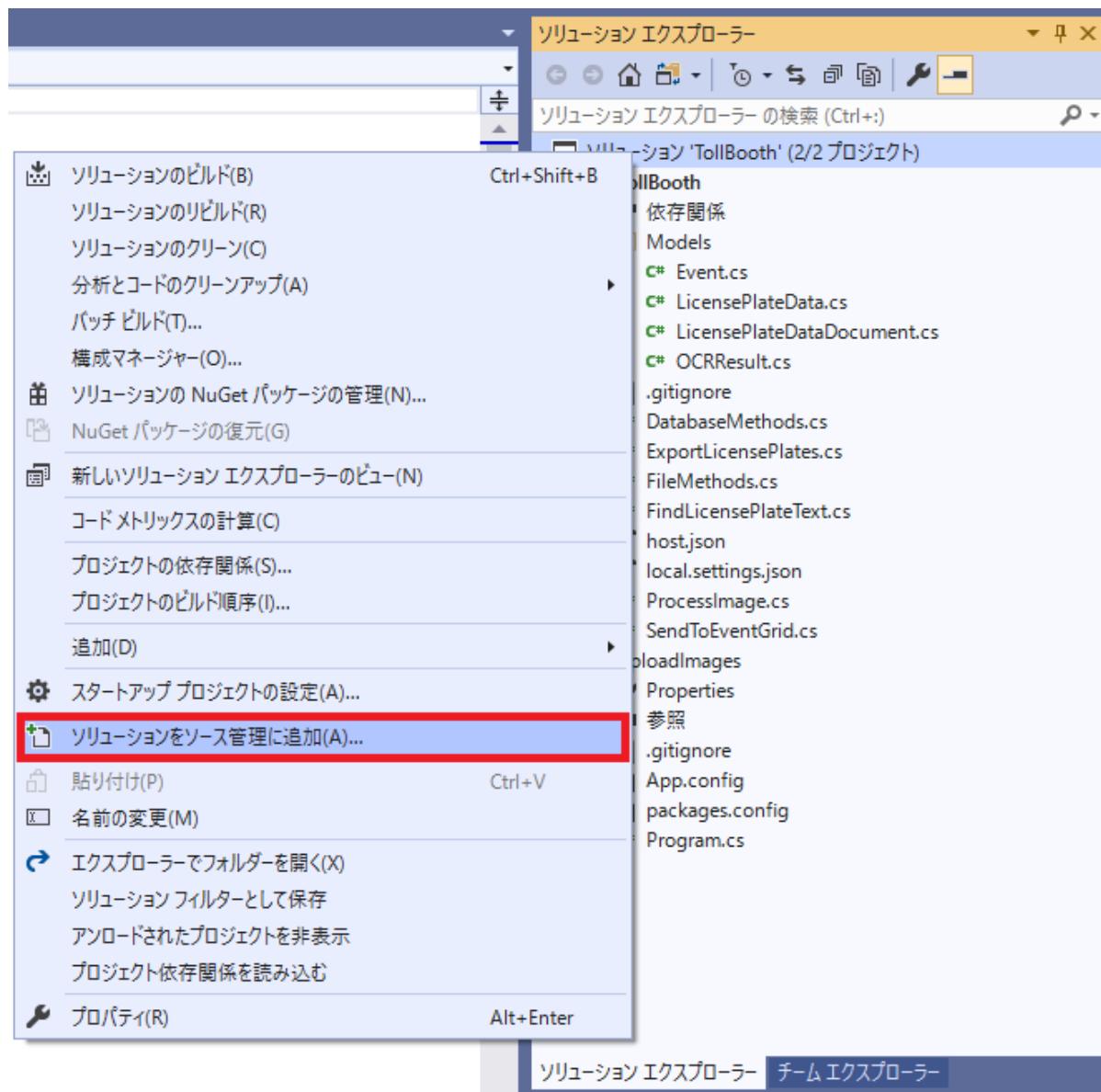
Total 2

<input type="checkbox"/>	Name	Type	Username or scope
<input type="checkbox"/>	HY Hiroya Yamamoto Admin hiroyay@microsoft.com	aad user	hiroyay@microsoft.com
<input type="checkbox"/>	H [REDACTED]	aad user	[REDACTED]

Direct Members

### Task 3: Azure Repos へのソース コードの追加

1. Visual Studio を起動し、「TollBooth」ソリューションを開く
2. ソリューション エクスプローラーで「TollBooth」ソリューションを右クリック  
表示されるメニューから「ソリューションをソース管理に追加」を選択



### 3. ローカルの Git リポジトリにソース コードが追加

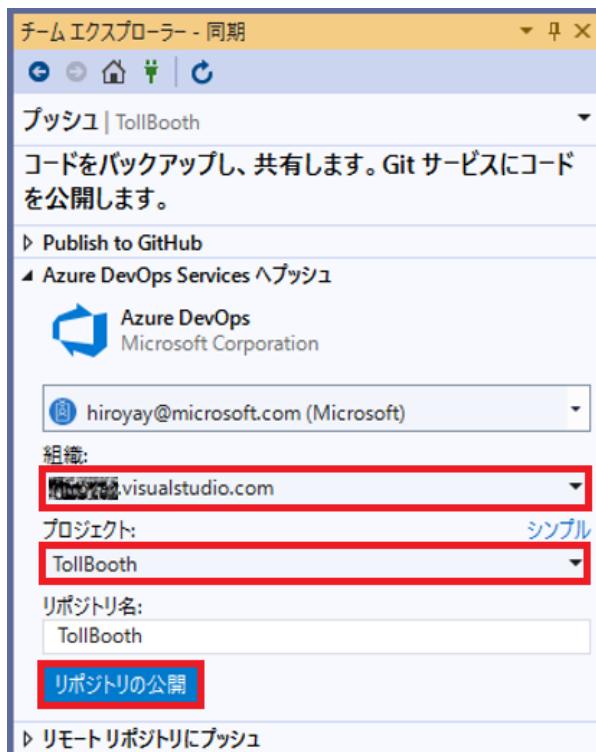
```
出力
出力元(S): ソース管理 - Git
C:\Users\hiroyay\source\repos\MCW-Serverless-architecture\TollBooth に新しい Git リポジトリが作成されました。
リポジトリを開いています:
C:\Users\hiroyay\source\repos\MCW-Serverless-architecture\TollBooth
コミット fe835593 がローカルでリポジトリ C:\Users\hiroyay\source\repos\MCW-Serverless-architecture\TollBooth に作成されました
```

### 4. 「表示」メニューから「チーム エクスプローラー」を選択

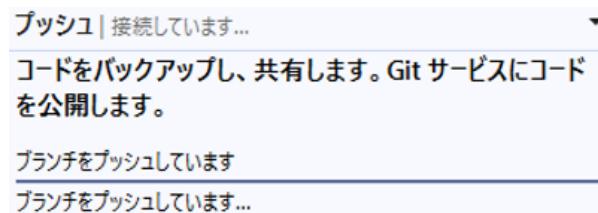
### 5. Azure DevOps Services ヘップシュの「Git リポジトリを公開」をクリック



## 6. 組織とプロジェクト(TollBooth)を選択し「リポジトリの公開」をクリック



## リモート リポジトリ (Azure Repos) へコードのプッシュが開始



### 7. プッシュ完了後 Web ブラウザーを移動

「**Repos**」を選択し、コードがプッシュされていることを確認

Azure DevOps hiroay / TollBooth / Repos / Files / TollBooth

TollBooth Overview Boards Repos Files Commits Pushes Branches Tags

Repos

Files

Name	Last change	Commits
TollBooth	木曜日	dbd2a373 Finished the ExportLicensePlates function Hiroya Yamamoto
UploadImages	2月11日	fe835593 プロジェクト ファイルを追加します。 Hiroya Yamamoto
.gitattributes	2月11日	fe835593 プロジェクト ファイルを追加します。 Hiroya Yamamoto
.gitignore	2月11日	fe835593 プロジェクト ファイルを追加します。 Hiroya Yamamoto
TollBooth.sln	2月11日	fe835593 プロジェクト ファイルを追加します。 Hiroya Yamamoto

## Task 4: Azure Pipelines の作成

### 1. 「Pipelines」を選択し、「Create Pipeline」をクリック

Azure DevOps hiroay / TollBooth / Pipelines

TollBooth Overview Boards Repos Pipelines Environments Releases Library Task groups Deployment groups Test Plans Artifacts

Create your first Pipeline

Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.

Create Pipeline

### 2. パイプライン作成のウィザードが開始

ソース コードを格納しているリポジトリの種類で「**Azure Repos Git**」を選択

Connect      Select      Configure      Review

New pipeline

## Where is your code?

-  Azure Repos Git YAML  
Free private Git repositories, pull requests, and code search
-  Bitbucket Cloud YAML  
Hosted by Atlassian
-  GitHub YAML  
Home to the world's largest community of developers
-  GitHub Enterprise Server YAML  
The self-hosted version of GitHub Enterprise
-  Other Git  
Any generic Git repository
-  Subversion  
Centralized version control by Apache

Use the classic editor to create a pipeline without YAML.

### 3. リポジトリで「TollBooth」を選択

✓ Connect      **Select**      Configure      Review

New pipeline

## Select a repository

TollBoothX

-  TollBooth

### 4. 「.NET Core Function App to Windows on Azure」テンプレートを選択

✓ Connect      ✓ Select      **Configure**      Review

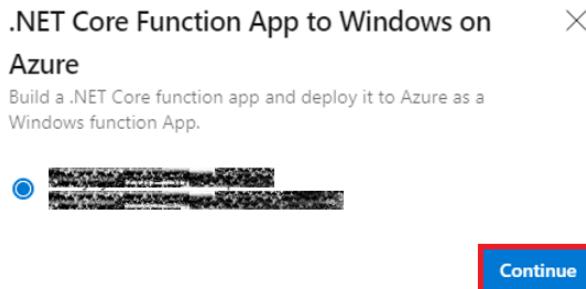
New pipeline

## Configure your pipeline

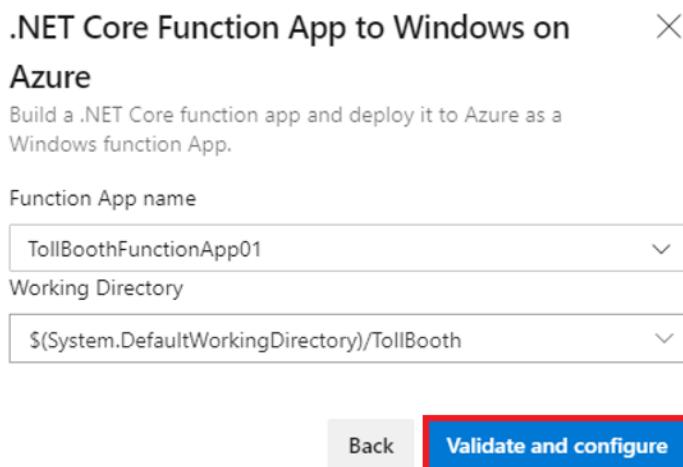
-  .NET Core Function App to Windows on Azure  
Build a .NET Core function app and deploy it to Azure as a Windows function App.
-  ASP.NET  
Build and test ASP.NET projects.
-  ASP.NET Core (.NET Framework)  
Build and test ASP.NET Core projects targeting the full .NET Framework.
-  .NET Desktop  
Build and run tests for .NET Desktop or Windows classic desktop solutions.

### 5. 展開先の Function App を選択するフォームが表示

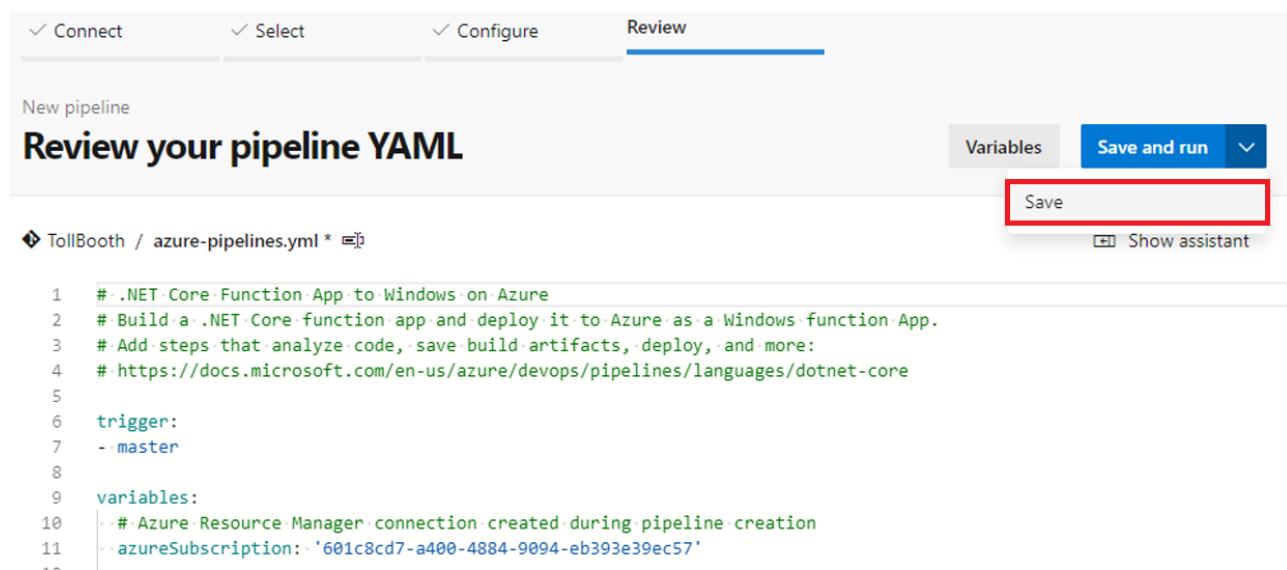
サブスクリプションを選択し、「Continue」をクリック



6. Function App name で Visual Studio から展開した Function App を選択  
**「Validate and configure」** をクリック

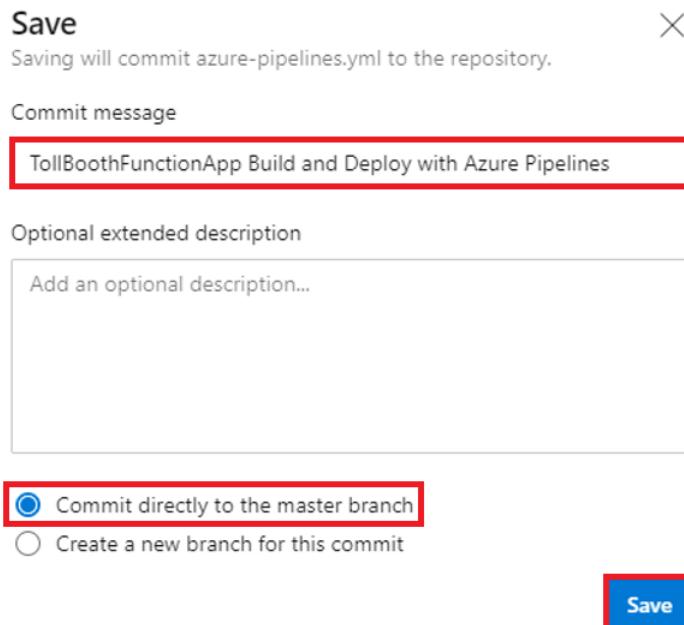


7. 今まで選択した内容で YAML が生成されるので内容を確認し 「Save」 をクリック



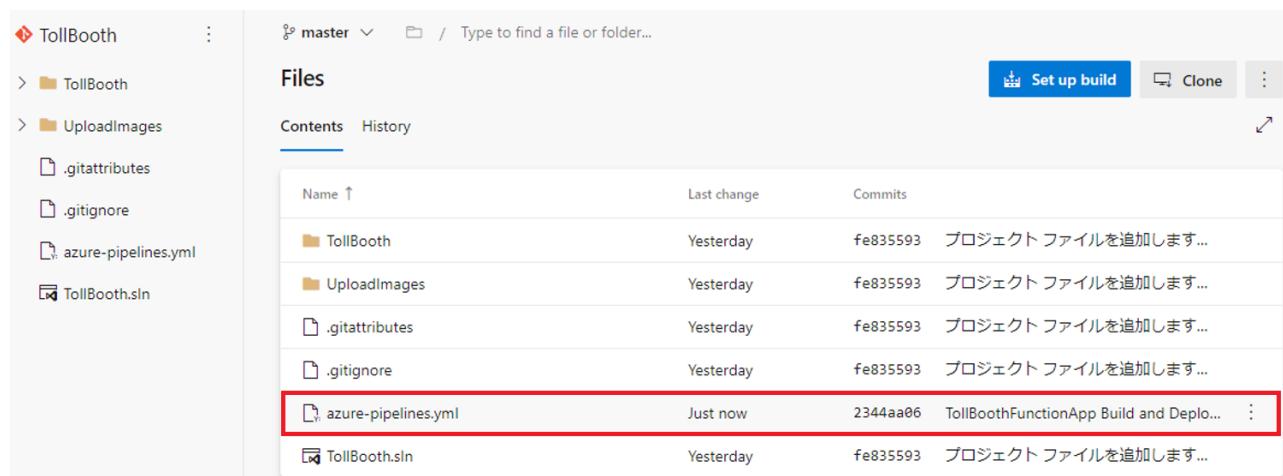
8. Save フォームで、以下の構成オプションを指定

- Commit message (**TollBoothFuncionApp Build and Deploy with Azure Pipelines**)
- Commit directly to the master branch** を選択



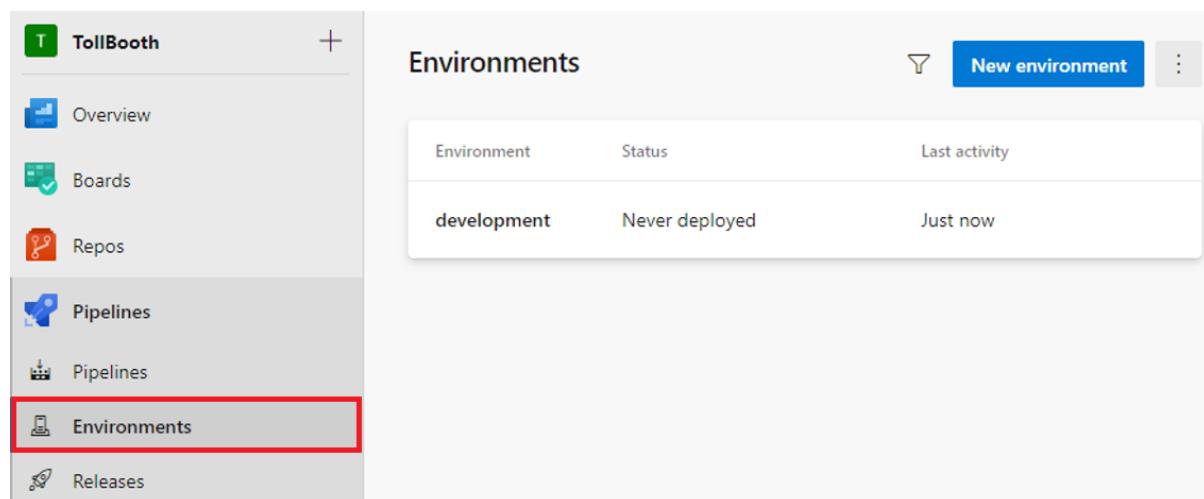
## 9. 「Save」をクリック

## 10. 「Repos」をクリックし、ディレクトリのトップに .yml ファイルが生成されていることを確認

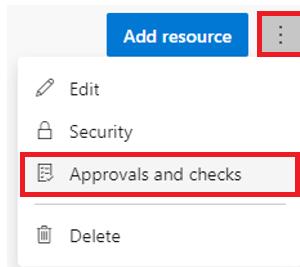


## リリース前の事前承認機能の追加

### 1. 「Pipelines」 - 「Environments」をクリック 「development」を選択



## 2. 「...」 - 「Approvals and checks」をクリック



## 3. 「Approvals」をクリック

A screenshot of the 'Add your first check' page. The 'Approvals' section, which contains the text 'Approvers should grant approval for deployment', is highlighted with a red box.

## 4. Approvals フォームのApprovers に承認を行うユーザーを入力

A screenshot of the 'Approvals' configuration form. A user named 'Hiroya Yamamoto' is listed in the 'Approvers' section, which is highlighted with a red box. Below the list is a text input field for 'Instructions to approvers (optional)' and two dropdown menus for 'Advanced' and 'Control options'. At the bottom are 'Cancel' and 'Create' buttons, with 'Create' being highlighted with a red box.

## 5. 「Create」をクリック

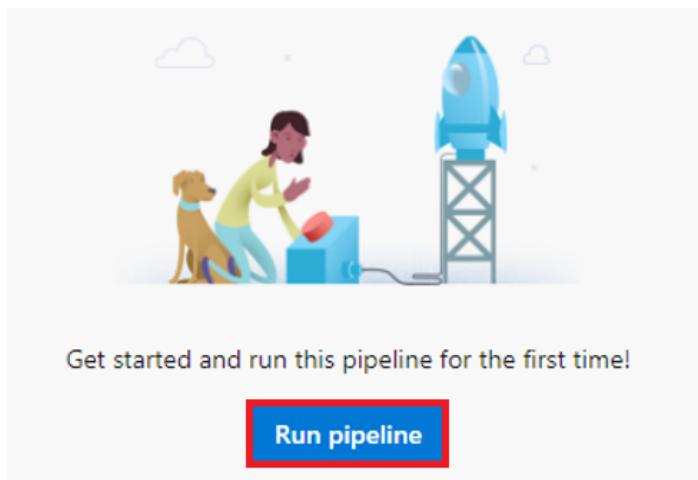
パイプラインの実行（ビルド&リリース）

## 6. メニューの「Pipelines」をクリック

先の手順で作成したパイプラインを選択

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with icons for Overview, Boards, Repos, Pipelines (which is selected and highlighted in blue), Environments, and Releases. The main area is titled 'Pipelines' and shows a table under 'Recently run pipelines'. The table has two columns: 'Pipeline' and 'Last run'. A single row is present for the 'TollBooth' pipeline, which is highlighted with a red box. The 'Last run' column for this pipeline says 'No runs yet'. At the top right, there are buttons for 'New pipeline' and 'Filter pipelines'.

## 7. 「Run pipeline」をクリック



## 8. 「Branch/tag」で「master」が選択されていることを確認し「Run」をクリック

The screenshot shows the 'Run pipeline' dialog box. At the top, it says 'Run pipeline' and 'Select parameters below and manually run the pipeline'. There's a close button 'X' on the right. Below that is a 'Branch/tag' section with a dropdown menu showing 'master'. A note says 'Select the branch, commit, or tag'. Under 'Advanced options', there are sections for 'Variables' (with a note 'This pipeline has no defined variables') and 'Stages to run' (with a note 'Run as configured'). At the bottom, there's a checkbox 'Enable system diagnostics', a 'Cancel' button, and a large blue 'Run' button which is highlighted with a red box.

## 9. 承認者に指定されたユーザーで Azure DevOps サイトにサインイン

The screenshot shows the 'Pipelines' page in a cloud-based development environment. At the top right are buttons for 'New pipeline' and a three-dot menu. Below is a navigation bar with 'Recent', 'All', and 'Runs' tabs, and a 'Filter pipelines' search bar. A section titled 'Recently run pipelines' lists the 'TollBooth' pipeline. The table has columns for 'Pipeline' (TollBooth), 'Last run' (#20200219.2), and time details ('11m ago' and '10m 58s'). The last run row is highlighted with a red border.

10. 「Pipelines」をクリック、先の手順で作成したパイプラインを選択

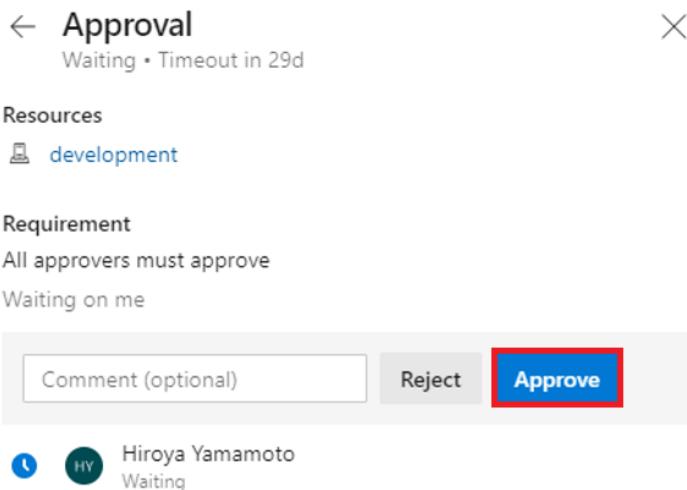
11. 実行中の項目を選択

The screenshot shows the 'TollBooth' pipeline run details. At the top right are 'Edit' and 'Run pipeline' buttons. Below is a navigation bar with 'Runs', 'Branches', and 'Analytics' tabs. The main area shows a table with a single row for the latest run. The table has columns for 'Description' (#20200219.2 Merged PR 6: Finished the ExportLicensePlates function) and 'Stages'. The run status is shown with a green checkmark and a blue clock icon. Time details ('13m ago' and '13m 6s') are also present. The entire run row is highlighted with a red border.

12. 「Review」をクリック

The screenshot shows a 'Review' dialog for the pipeline run. At the top right are 'Cancel' and a three-dot menu buttons. The main area has a 'Summary' tab selected. It shows the run was manually triggered by 'Hiroya Yamamoto' on 'TollBooth' branch 'master' at 'Today at 17:48'. Metrics include duration '15m 33s', tests '0', changes '0', work items '0', and artifacts '1 published'. A note says '1 approval needs your review before this run can continue to Deploy stage'. A large red-bordered 'Review' button is at the bottom right. Below is a 'Stages' tab showing two stages: 'Build stage' (green checkmark, 1 job completed, 2m 17s, 1 artifact) and 'Deploy stage' (blue waiting button, 0/1 checks passed).

13. 「Approve」をクリック



#### 14. Build, Deploy 各ステージが完了したことを確認

Build stage	Deploy stage
1 job completed	1 job completed
2m 15s	56s
1 artifact	1 check passed

※ステージをクリックすると、各ステップでのステータスを詳細に確認可

← Jobs in run #20200212.1

TollBooth

Build stage

- Build 2m 12s
- Initialize job 3s
- Checkout TollBooth@m... 6s
- Build 1m 19s
- Archive files 1s
- PublishPipelineArtifact 8s
- Component Detection... 32s
- Post-job: Checkout To... <1s
- Finalize Job <1s

Deploy stage

- Deploy 11s
- Initialize job 3s
- Download Artifact 7s
- Azure functions app deploy

Azure functions app deploy

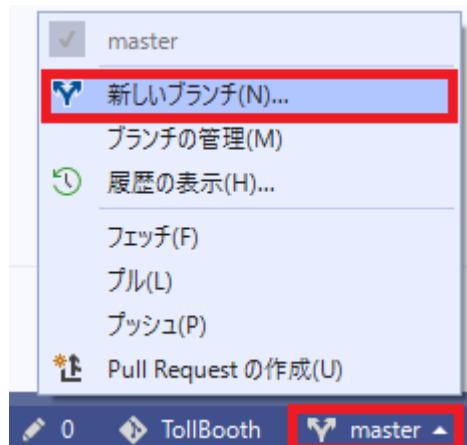
```

1 Starting: Azure functions app deploy
2 =====
3 Task      : Azure Functions
4 Description : Update a function app with .NET, Python, JavaScript, PowerShell, Java based web applications
5 Version   : 1.163.2
6 Author    : Microsoft Corporation
7 Help      : https://aka.ms/azurefunctiontroubleshooting
8 =====

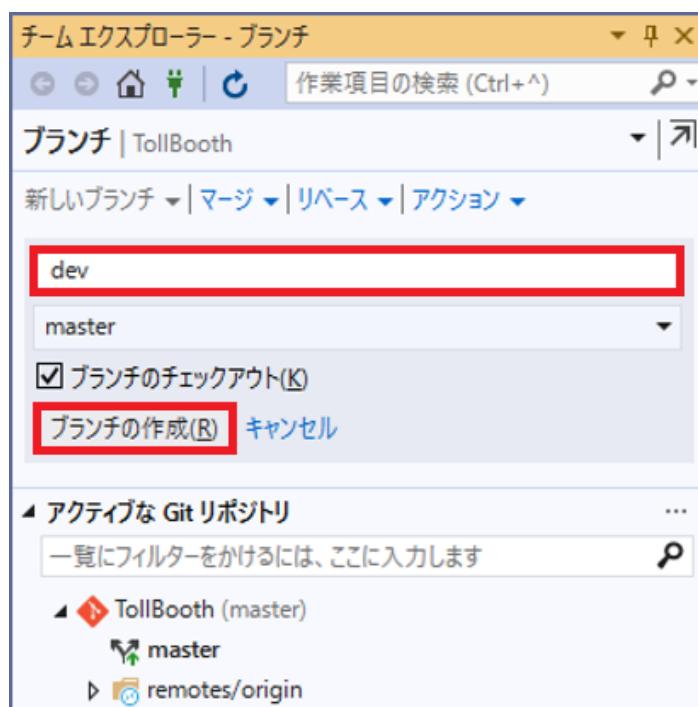
```

#### Task 5: ブランチの作成と ExportLicensePlates 関数の変更

- Visual Studio で右下の「master」をクリック  
表示されるメニューより「新しいブランチ」を選択



2. チーム エクスプローラーで、「dev」と入力し「ブランチの作成」をクリック



3. 「表示」メニューの「タスク一覧」を選択し ToDo リストを表示

4. 「TODO 5: ...」をダブルクリック

DatabaseMethods.cs ファイルのコードを追加する箇所が表示

5. TODO 5 に以下のコードを追加

```
// MaxItemCount 値は全てのレコードが返されるまでに一度に 100 個のドキュメントを取得するよう指示
// TODO 5: exported の値が false である
// LicensePlateDataDocument オブジェクトのリストを collectionLink から取得
licensePlates = _client.CreateDocumentQuery<LicensePlateDataDocument>
(collectionLink,
    new FeedOptions() { EnableCrossPartitionQuery = true, MaxItemCount = 100 })
    .Where(l => l.exported == false)
    .ToList();
// COMPLETE: licensePlates = _client.CreateDocumentQuery ...
```

6. TODO 6 の `licensePlates = new List<LicensePlateDataDocument>();` を削除

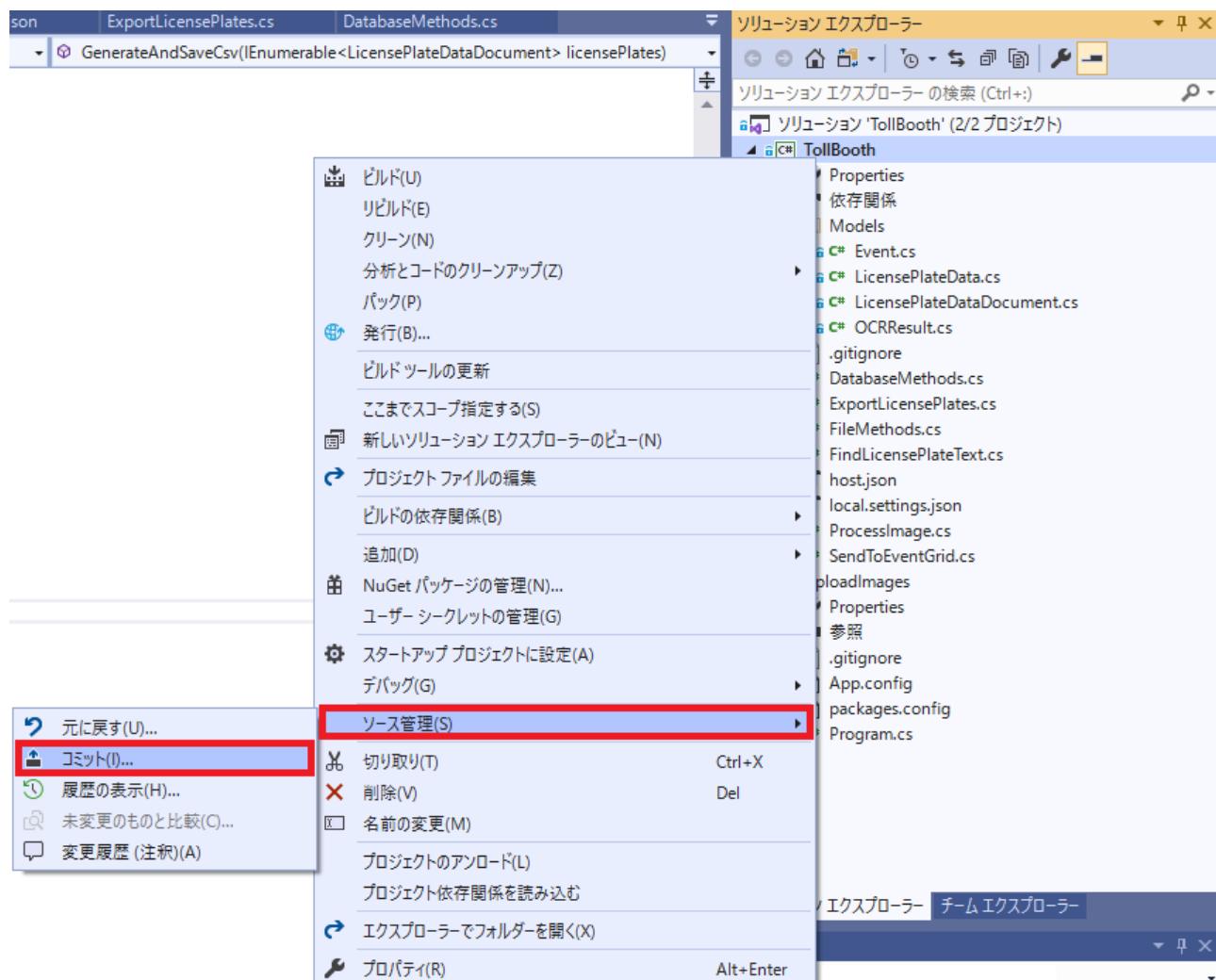
7. タスク一覧の「**TODO 7: ...**」をダブルクリック  
 FileMethods.cs ファイルのコードを追加する箇所が表示

8. TODO 7 に以下のコードを追加

```
// TODO 7: メモリ ストリームから Blob を非同期にアップロード
await blob.UploadFromStreamAsync(stream);
// COMPLETE: await blob...;
```

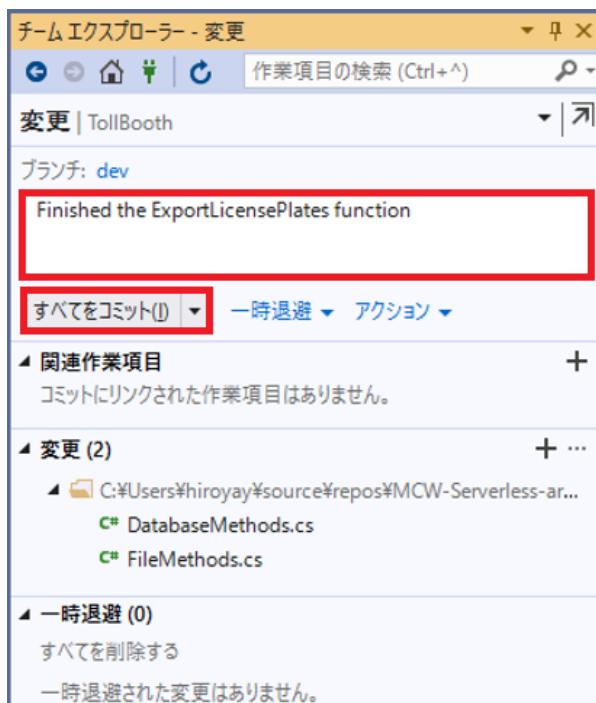
9. 「ファイル」メニューの「すべて保存」をクリックし、これまでの変更を保存

10. ソリューション エクスプローラーで「TollBooth」プロジェクトを右クリック  
 表示されるメニューより「ソース管理」 - 「コミット」を選択

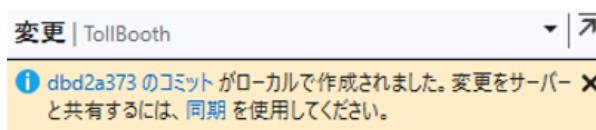


11. チーム エクスプローラーが表示

コメントに「**Finished the ExportLicensePlates function**」と入力  
 「すべてをコミット」をクリック



## 12. ローカル Git リポジトリへのコミット完了メッセージ内の「同期」をクリック



## 13. 出力方向のコミットの「プッシュ」をクリック



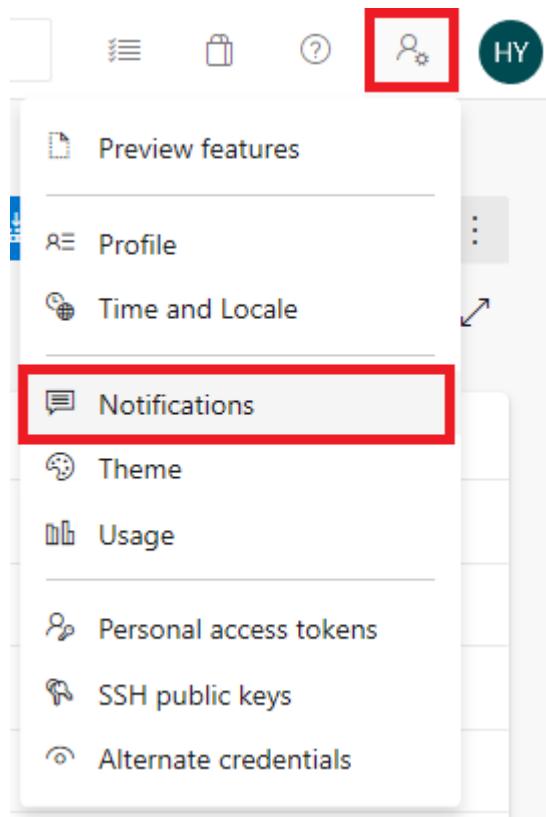
## 14. リモートリポジトリに正常にプッシュできたことを確認



## Task 6: プルリクエストの作成

### プルリクエストの通知設定

1. コード レビューを行うユーザーで Azure DevOps へサインイン
2. 画面右上の人型のアイコンをクリック  
表示されるメニューより「**Notifications**」を選択



3. 「\*\*+ New subscription」をクリック

The screenshot shows the 'New subscription' form in Azure DevOps. On the left, there is a sidebar with 'User settings' (Hiroya Yamamoto), 'Account' (Profile, Time and Locale), and 'Preferences' (Notifications, which is selected and highlighted with a grey background). The main area has tabs for 'Notifications' (selected) and 'Help'. Below the tabs, there is a 'Description' field, a 'Build' section with 'Build completes' (Notify when a build you queued or that was queued for you completes), a 'Code (Git)' section with 'Pull request reviewers added or removed' (Notify when you are added to a pull request or when a user is added or removed from a pull request you created), and a 'Help' section.

4. New subscription フォームで以下を選択 a. Category (**Code (Git)**) b. Template (**A pull request is created or updated**)

## New subscription

Category	Template
Build	A commit authored by me is pushed
Code (Git)	A commit is pushed by me
Code (TFVC)	A commit is pushed
Work	<b>A pull request is created or updated</b>
Artifacts	A comment is made on a pull request
Extension management	
Release	

**Next****Cancel**

## 5. 「**Next**」をクリック

## 6. A specific team project で「TollBooth」が選択されていることを確認

### New subscription

Description	Subscriber
A pull request is created or updated	Hiroya Yamamoto

Deliver to

Address

Preferred email: hiroyay@microsoft.com

Filter

Any team project    A specific team project   TollBooth

Filter criteria

+  And/Or  Field:  Operator:  Value:

+ Add new clause

Previous   **Finish**   Cancel

## 7. 「**Finish**」をクリックし、通知の作成を完了

### プルリクエストの作成

## 8. コード レビューアーとは別のユーザーで Azure DevOps へサインイン

## 9. 「Repos」 - 「Pull Request」を選択

The screenshot shows the 'TollBooth' repository in the Azure DevOps interface. The 'Pull requests' section is active. A message at the top says 'You updated dev 5m ago'. Below it, a diagram shows three people connected by arrows, one pointing to a document. The text 'Currently, no pull requests need your attention' is displayed. A 'New pull request' button is visible at the bottom right, which is also highlighted with a red box.

## 10. New Pull Request フォームでプルリクエストを作成

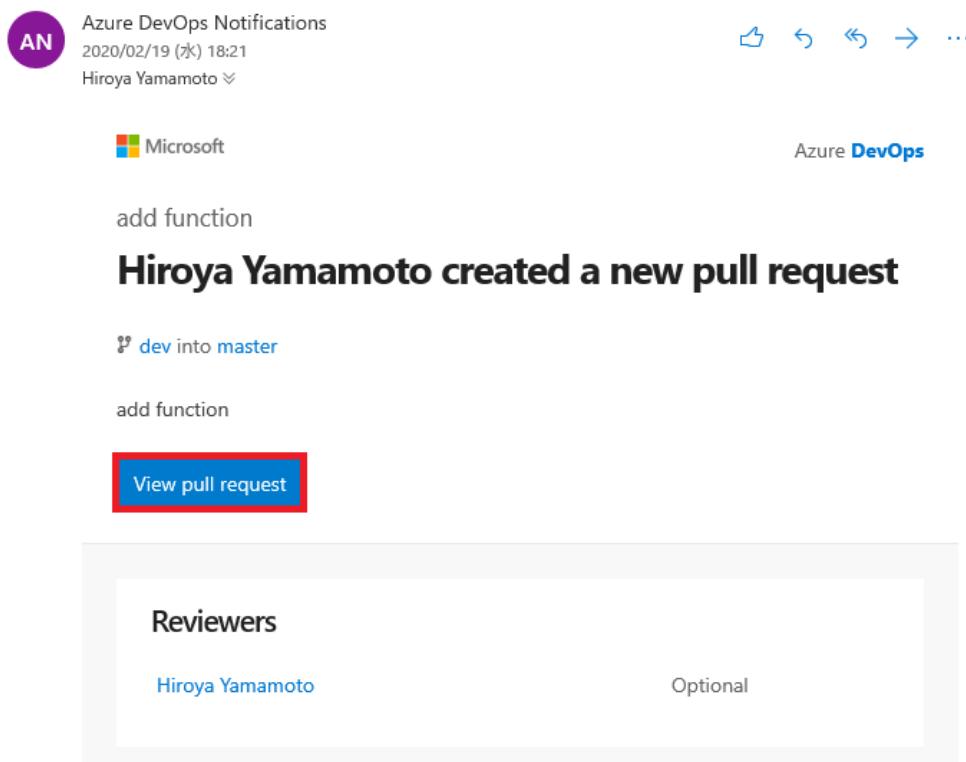
- 「dev」 into 「master」を確認
- 「Reviewers」にコードレビューを行うユーザーを入力

The screenshot shows the 'New Pull Request' form. It includes fields for 'From' (set to 'dev') and 'To' (set to 'master'), a 'Title' field containing 'Finished the ExportLicensePlates function', a 'Description' field containing 'Finished the ExportLicensePlates function', a 'Reviewers' field showing 'Hiroya Yamamoto' with a search bar, a 'Work Items' field with a search bar, and a 'Create' button at the bottom right.

## 11. 「Create」をクリック

## Task 7: ブランチのマージと新バージョンの Azure への展開

### 1. レビューアーがメール通知を受信

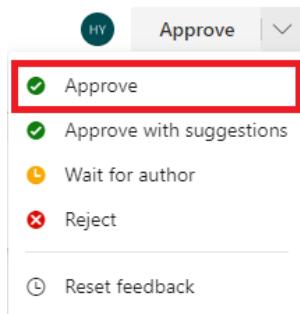


### 2. 「View pull request」をクリック

### 3. プルリクエストの確認フォームが Web ブラウザーで表示

The screenshot shows the Azure DevOps pull request review interface. At the top, it says "ACTIVE Finished the ExportLicensePlates function". Below that, there are sections for "Overview", "Files", "Updates", and "Commits". The "Description" section contains a comment input field with "Add a comment...". On the right side, there are sections for "Work Items", "Reviewers" (listing "Hiroya Yamamoto"), and "Labels".

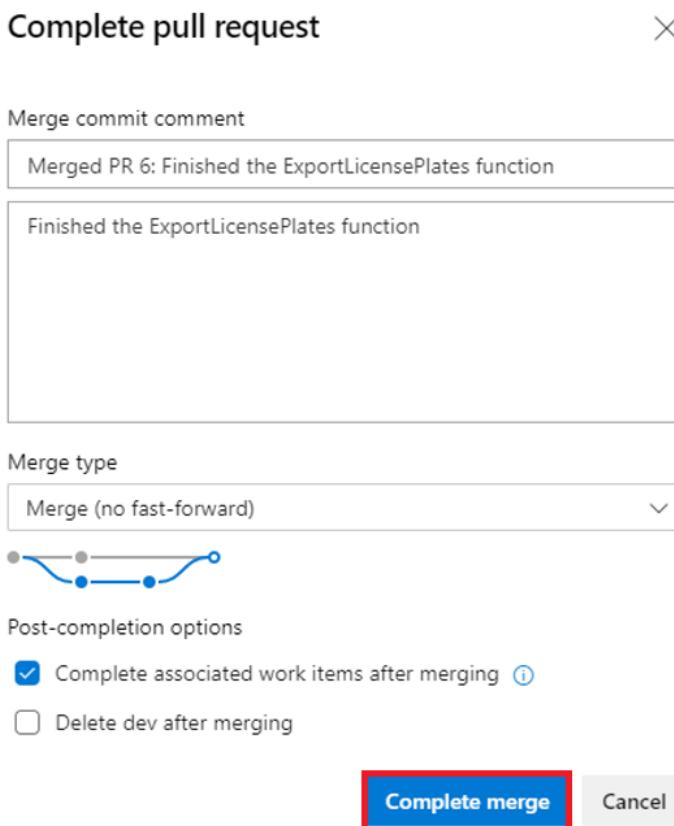
### 4. 「Approve」をクリック



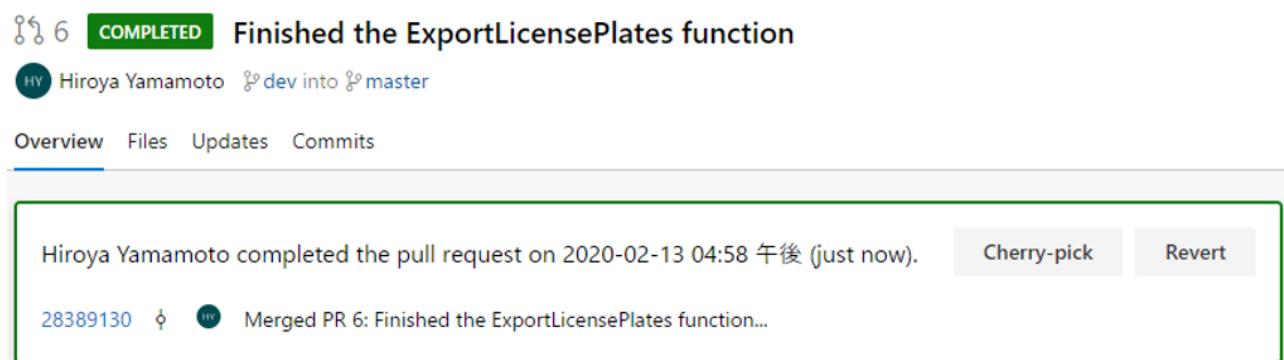
## 5. 「Complete」をクリック



## 6. 「Complete pull request」フォームで「Complete merge」をクリック

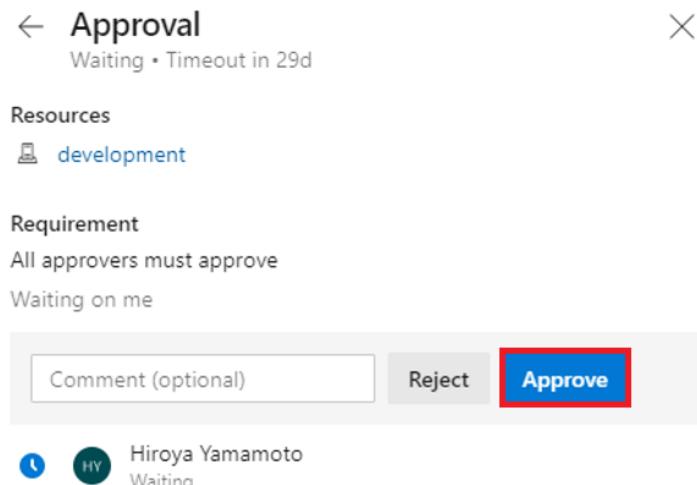


## 7. リクエストが COMPLETED とマークされ、マージが完了

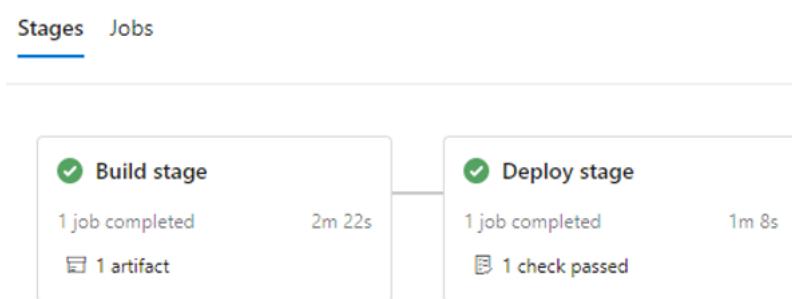


## 8. master ブランチに dev ブランチの変更が反映されたためパイプラインが起動

## 9. Deploy ステージの前に承認を実行



## 10. Function App への展開完了を確認



## Exercise 8: ワークフローの実行とエクスポートデータの確認

### Task 1: Logic App の実行

1. Azure ポータル (<https://portal.azure.com>) を開く
2. Logic App へ移動
3. 「概要」ブレードで「有効」をクリック



4. 「トリガーの実行」 - 「Recurrence」をクリック



※15分間隔で実行される Logic App を手動で即時実行

5. 実行の履歴から Logic App の処理が正常に完了したことを確認

実行履歴					
すべて	開始時刻を次の時刻より前にする	日付を選択	時刻の選択		
実行識別子を指定して直接モニタービューを開いてください					
状態	開始時刻	識別子	期間	静的な結果	
成功	2020/2/13 17:05	08586200257291608546387183715CU28	10.45 秒		
成功	2020/2/11 18:26	08586201937040008481889775020CU04	4.54 秒		
成功	2020/2/11 18:25	08586201937274780543955690552CU16	1.3 秒		

## Task 2: エクスポートされた CSV ファイルの表示

1. ストレージ アカウントへ移動
2. 「概要」ブレードで「コンテナー」をクリック

The screenshot shows the Azure Storage Explorer blade for the 'holserverlesstollbooth' storage account. On the left, there's a sidebar with options like 'アクティビティログ', 'アクセス制御 (IAM)', and 'タグ'. The main area displays container details: 'リソースグループ' (変更) : HOL-2020-03-Serverless-RG, '状態' : プライマリ: 利用可能, '場所' : 米国西部 2, 'サブスクリプション' (変更) : [REDACTED], 'サブスクリプション ID' : [REDACTED], and 'タグ (変更)' : タグを追加するにはここをクリック. A red box highlights the 'コンテナー' section, which contains a description: '非構造化データ用のスケーラブルでコスト効率の高いストレージ' and a '詳細情報' link. To the right, there's another section titled 'ファイル共有' with a 'サーバーレスの SMB ファイル共有' link.

3. 「export」コンテナーを選択

The screenshot shows the Azure Storage Explorer blade for the 'holserverlesstollbooth - コンテナー' container. The sidebar includes 'アクティビティログ', 'アクセス制御 (IAM)', and 'タグ'. The main area lists blobs: '名前' (Name), '最終変更日時' (Last modified), 'パブリック アク...', and 'リース状態' (Lease status). Two blobs are listed: 'export' (2020/2/7 14:34:08, プライベート, 利用可能) and 'images' (2020/2/11 17:33:57, プライベート, 利用可能). A red box highlights the 'export' blob.

4. ワークフローにより生成された CSV ファイルを選択

認証方法: アクセスキー (Azure AD のユーザー アカウントに切り替える)  
場所: export

プレフィックスによる BLOB の検索 (大文字と小文字を区別)

名前  
 2020-02-13T08:05:59.csv

5. 「ダウンロード」をクリック

2020-02-13T08:05:59.csv

保存 × 破棄 **ダウンロード** 最新の情報に更新 | 削除 | 履歴の変更

概要 スナップショット 編集 SAS の生成

プロパティ URL <https://holserverlesstollbooth.blob.core.windows.net/>

6. ダウンロードしたファイルを確認

A	B	C	D	
1	FileName	LicensePlateText	TimeStamp	LicensePlateFound
2	TZRPKRLE.jpg	THECAR	2/11/2020 8:48:43 AM	TRUE
3	IJXXK34LK.jpg	127 RFS	2/11/2020 8:49:25 AM	TRUE
4	EXJZQDU0.jpg	BUG 2145	2/11/2020 8:49:25 AM	TRUE
5	5ML4PJGA.jpg	ZOOMN65	2/11/2020 8:53:26 AM	TRUE
6	PW65T8YQ.jpg	127 RFS	2/11/2020 8:53:26 AM	TRUE
7	QJ0YXGR8.jpg	THECAR	2/11/2020 8:53:27 AM	TRUE
8	JOKROSCS.jpg	ZOOMN65	2/11/2020 8:53:28 AM	TRUE
9	X4SW3Y0F.jpg	127 RFS	2/11/2020 8:53:29 AM	TRUE
10	F7VFYGC4.jpg	THECAR	2/11/2020 8:53:30 AM	TRUE
11	JPE6CG5I.jpg	BUG 2145	2/11/2020 8:53:30 AM	TRUE
12	1H4UF24T.jpg	JP THRLS	2/11/2020 8:53:30 AM	TRUE
13	L4A2Y4FV.jpg	JP THRLS	2/11/2020 8:53:31 AM	TRUE
14	17K20CW2.jpg	ZOOMN65	2/11/2020 8:53:31 AM	TRUE
15	WXZ5C21Q.jpg	BUG 2145	2/11/2020 8:53:32 AM	TRUE
16	XFH8NFQA.jpg	127 RFS	2/11/2020 8:53:32 AM	TRUE
17	2X45O8OG.jpg	THECAR	2/11/2020 8:53:32 AM	TRUE
18	3MVHFCUK.jpg	ZOOMN65	2/11/2020 8:53:32 AM	TRUE
19	M3QRKW7P.jpg	BUG 2145	2/11/2020 8:53:32 AM	TRUE
20	1AST9DRV.jpg	JP THRLS	2/11/2020 8:53:33 AM	TRUE
21	VUAA9KHY.jpg	BUG 2145	2/11/2020 8:53:34 AM	TRUE
22	QJGDN00S.jpg	127 RFS	2/11/2020 8:53:34 AM	TRUE
23	0AFBUJ4T.jpg	THECAR	2/11/2020 8:53:34 AM	TRUE
24	HR543X86.jpg	JP THRLS	2/11/2020 8:53:34 AM	TRUE
25	CODBKZRZ.jpg	ZOOMN65	2/11/2020 8:53:34 AM	TRUE
26	GJPE1L3C.jpg	JP THRLS	2/11/2020 8:54:39 AM	TRUE

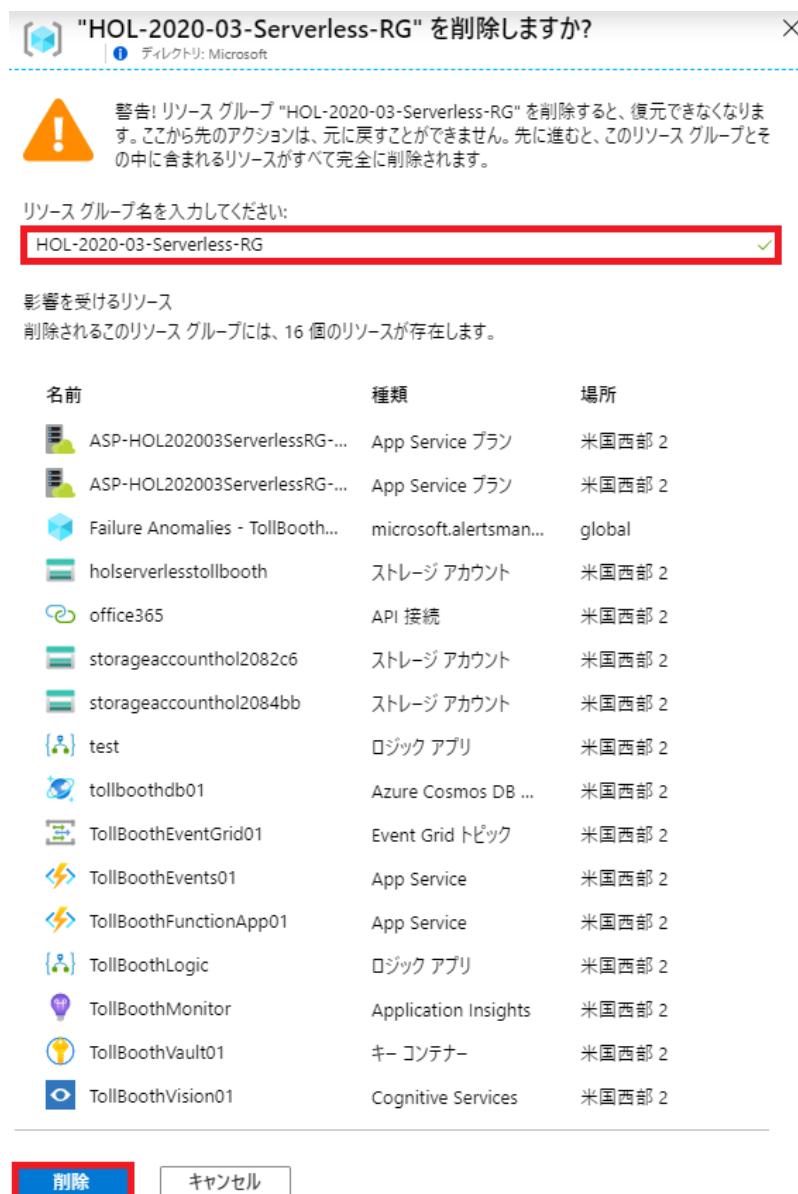
ExportLicensePlates 関数は Cosmos DB の Processed コレクション内の exported が false のアイテムを検出し、CSV ファイルへエクスポートします。エクスポートされたアイテムは exported が true へ更新されます。次のワークフローの実行では、最後のエクスポート以降にアップロードされたデータが検出されます。

## ワークショップの終了

ワークショップで使用した Azure リソースを削除します。

### Task 1: リソース グループの削除

1. Azure ポータル (<https://portal.azure.com>) を開く
2. 「リソース グループ」ブレードへ移動
3. 演習で使用したリソース グループを選択
4. 「概要」ブレードで 「Delete resource group」をクリック
5. リソース グループ名を入力し「削除」をクリック



## Task 2: Azure DevOps プロジェクトの削除

1. Azure DevOps サイトへ移動
2. 「TollBooth」プロジェクトを選択
3. 画面左下の「Project settings」をクリック
4. 「Delete」をクリック

Delete project  
This will affect all contents and members of this project.  
[Learn more about deleting projects](#)

**Delete**

5. Delete project フォームでプロジェクト名 (**TollBooth**) を入力

## Delete project

Are you sure you want to delete the "TollBooth" project?

You will have up to 28 days to recover this project. After, this project will be deleted resulting in a loss of all project artifacts including work items, repos, teams, and builds. [Learn more about deleting projects.](#)

To confirm this action, please type "TollBooth":

[Cancel](#)

[Delete](#)

### 6. 「Delete」をクリック

Azure DevOps 組織を削除する場合は、「Organization settings」から「Delete」を選択  
同様に組織名の入力を求められるので、名前を入力し「Delete」をクリック

---

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.