# Microsoft Azure への ASAv Auto Scale ソリューションの導入



# Azure での の Auto Scale ソリューション

## Auto Scale ソリューションについて

ASAv Auto Scale for Azure は、Azure が提供するサーバーレス インフラストラクチャ（Logic App、Azure 関数、ロードバランサ、セキュリティグループ、仮想マシンスケールセットなど）を使用する完全なサーバーレス導入です。

ASAv Auto Scale for Azure 導入の主な特徴は次のとおりです。

- Azure Resource Manager（ARM）テンプレートベースの展開。
- CPU および に基づくスケーリングメトリックのサポート：

（注） 詳細については、「Auto Scale ロジック （28 ページ）」を参照してください。

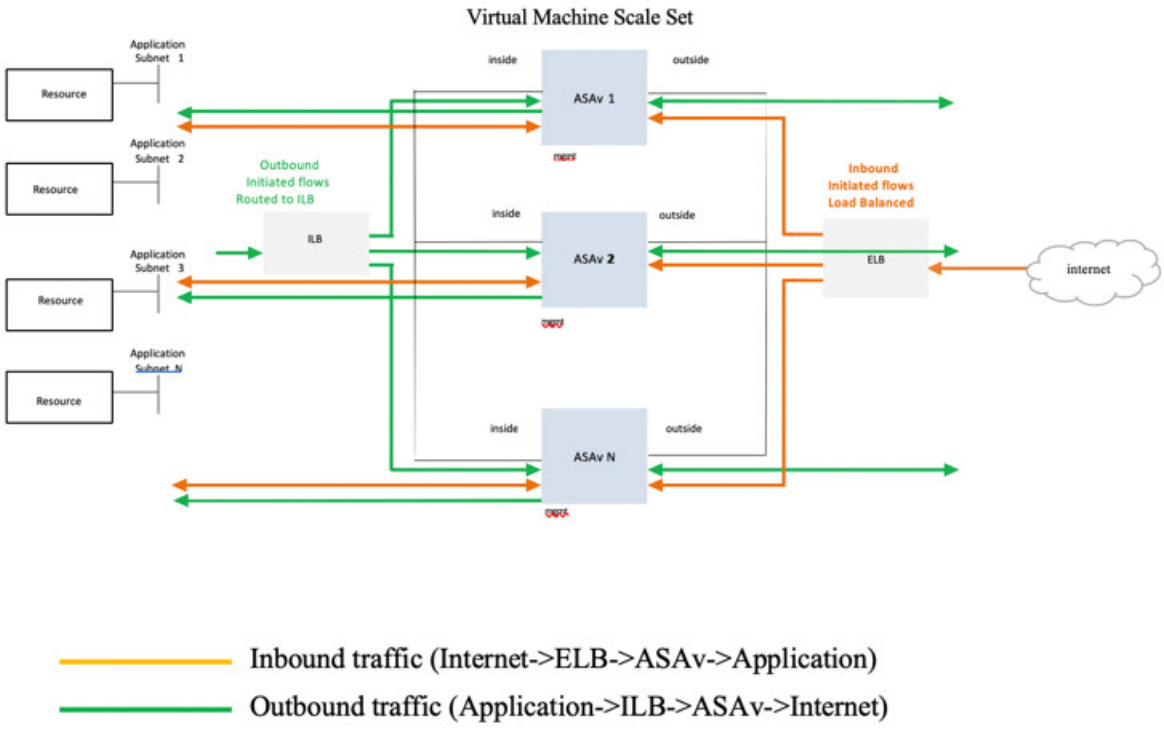
- ASAv 展開とマルチ可用性ゾーンのサポート。
- スケールアウトされた ASAv インスタンスに完全に自動化された構成を自動適用。
- ロードバランサとマルチ可用性ゾーンのサポート。
- Auto Scale 機能の有効化と無効化をサポート。
- シスコでは、導入を容易にするために、Auto Scale for Azure 導入パッケージを提供しています。

# Auto Scale の導入例

ASAv Auto Scale for Azure は、ASAv スケールセットを Azure の内部ロードバランサ（ILB）と Azure の外部ロードバランサ（ELB）の間に配置する自動水平スケーリングソリューションです。

- ELB は、インターネットからのトラフィックをスケールセット内の ASAv インスタンスに分散させます。その後、ファイアウォールがアプリケーションにトラフィックを転送します。
- ILB は、アプリケーションからのアウトバウンド インターネット トラフィックをスケールセット内の ASAv インスタンスに分散させます。その後、ファイアウォールがインターネットにトラフィックを転送します。
- ネットワークパケットが、単一の接続で両方（内部および外部）のロードバランサを通過することはありません。
- スケールセット内の ASAv インスタンスの数は、負荷条件に基づいて自動的にスケーリングおよび設定されます。

図 *1 : ASAv Auto Scale* の導入例




# スコープ

このドキュメントでは、ASAv Auto Scale for Azure ソリューションと、 のサーバーレスコンポーネントを展開する詳細な手順について説明します。

**重要**

- 導入を開始する前に、ドキュメント全体をお読みください。
- 導入を開始する前に、前提条件を満たしていることを確認します。
- ここに記載されている手順と実行順序に従っていることを確認します。

# 導入パッケージのダウンロード

ASAv Auto Scale for Azure ソリューションは、Azure が提供するサーバーレス インフラストラクチャ（Logic App、Azure 関数、ロードバランサ、仮想マシンスケールセットなど）を使用する Azure Resource Manager（ARM）テンプレートベースの展開です。

ASAv Auto Scale for Azure ソリューションの起動に必要なファイルをダウンロードします。該当するバージョン用の展開スクリプトとテンプレートは、GitHub リポジトリから入手できます。

**注目** Auto Scale 用のシスコ提供の導入スクリプトおよびテンプレートは、オープンソースの例として提供されており、通常の Cisco TAC サポートの範囲内ではカバーされないことに注意してください。更新と ReadMe の手順については、GitHub を定期的に確認してください。

ASM_Function.zip パッケージの作成方法については、「ソースコードからの Azure 関数の構築（31 ページ）」を参照してください。

# Auto Scale ソリューションのコンポーネント

ASAv Auto Scale for Azure ソリューションは、次のコンポーネントで構成されています。

### Azure 関数（Function App）

Function App とは一連の Azure 関数です。基本的な機能は次のとおりです。

- Azure メトリックを定期的に通信またはプローブします。
- ASAv の負荷をモニターし、スケールイン/スケールアウト操作をトリガーします。

関数は、圧縮された Zip パッケージの形式で提供されます（「Azure Function App パッケージの構築 （8 ページ）」を参照）。関数は、特定のタスクを実行するために可能な限り独立しており、拡張機能や新しいリリースのサポートのために必要に応じてアップグレードできます。

### Orchestrator（Logic App）

Auto Scale Logic App は、ワークフロー、つまり一連のステップの集合です。Azure 関数は独立したエンティティであり、相互に通信できません。この Orchestrator は、関数の実行を順序付けし、関数間で情報を交換します。

- Logic App は、Auto Scale Azure 関数間で情報をオーケストレーションおよび受け渡すために使用されます。
- 各ステップは、Auto Scale Azure 関数または組み込みの標準ロジックを表します。
- Logic App は JSON ファイルとして提供されます。
- Logic App は、GUI または JSON ファイルを使用してカスタマイズできます。

### 仮想マシンスケールセット（VMSS）

VMSS は、ASAv デバイスなどの同種の仮想マシンの集合です。

- VMSS では、新しい同一の VM をセットに追加できます。
- VMSS に追加された新しい VM は、ロードバランサ、セキュリティグループ、およびネットワーク インターフェイスに自動的に接続されます。
- VMSS には組み込みの Auto Scale 機能があり、ASAv for Azure では無効になっています。
- VMSS で ASAv インスタンスを手動で追加したり、削除したりしないでください。

#### Azure Resource Manager（ARM）テンプレート

ARM テンプレートは、ASAv Auto Scale for Azure ソリューションに必要なリソースを展開するために使用されます。

Auto Scale for Azure：ARM テンプレート **azure_asav_autoscale.json** は、以下を含む Auto Scale Manager コンポーネントへの入力情報を提供します。

- Azure Function App
- Azure Logic App
- 仮想マシンスケールセット（VMSS）
- 内部および外部ロードバランサ。
- 展開に必要なセキュリティグループおよびその他のコンポーネント。

**重要** ユーザー入力の検証に関しては、ARM テンプレートには限界があるため、展開時に入力を検証する必要があります。

# Auto Scale ソリューションの前提条件

## Azure のリソース

#### リソース グループ

このソリューションのすべてのコンポーネントを展開するには、既存または新しく作成されたリソースグループが必要です。

**（注）** 後で使用するために、リソースグループ名、リソースグループが作成されたリージョン、および Azure サブスクリプション ID を記録します。

### ネットワーキング

仮想ネットワークが使用可能または作成済みであることを確認します。 Auto Scale 展開では、ネットワークリソースの作成、変更、管理は行われません。

ASAv には 3 つのネットワークインターフェイスが必要なため、仮想ネットワークには次の 3 つのサブネットが必要です。

1. 管理トラフィック
2. 内部トラフィック
3. 外部トラフィック

サブネットが接続されているネットワーク セキュリティ グループで、次のポートを開く必要があります。

- SSH（TCP/22）

  ロードバランサと ASAv 間の正常性プローブに必要です。

  サーバーレス機能と ASAv 間の通信に必要です。

- アプリケーション固有のプロトコルまたはポート

  ユーザーアプリケーションに必要です（TCP/80 など）。

（注） 仮想ネットワーク名、仮想ネットワーク CIDR、 3 つすべてのサブネットの名前、および外部と内部のサブネットのゲートウェイ IP アドレスを記録します。

# ASA 構成ファイルの準備

ASAv 構成ファイルを準備し、ASAv インスタンスからアクセス可能な HTTP/HTTPS サーバーに保存します。これは標準の ASA 構成ファイル形式です。スケールアウトされた ASAv により、このファイルがダウンロードされて構成が更新されます。

ASA 構成ファイルでは、（少なくとも）次のことが必要になります。

- すべてのインターフェイスに DHCP IP 割り当てを設定します。
- GigabitEthernet0/1 は「内部」インターフェイスである必要があります。
- GigabitEthernet0/0 は「外部」インターフェイスである必要があります。
- ゲートウェイを内部インターフェイスと外部インターフェイスに設定します。
- 内部インターフェイスと外部インターフェイスで Azure ユーティリティ IP からの SSH を有効にします（ヘルスプローブ用）。
- 外部インターフェイスから内部インターフェイスにトラフィックを転送するための NAT 構成を作成します。

- 目的のトラフィックを許可するアクセスポリシーを作成します。
- 構成のライセンスを取得します。PAYG 課金はサポートされていません。

（注） 管理インターフェイスを特別に設定する必要はありません。

以下は、 の ASA 構成ファイルのサンプルです。

```
ASA Version 9.13(1)
!
interface GigabitEthernet0/1
nameif inside
security-level 100
ip address dhcp setroute
!
interface GigabitEthernet0/0
nameif outside
security-level 0
ip address dhcp setroute
!
route outside 0.0.0.0 0.0.0.0 10.12.3.1 2
!
route inside 0.0.0.0 0.0.0.0 10.12.2.1 3
!
ssh 168.63.129.0 255.255.255.0 outside
!
ssh 168.63.129.0 255.255.255.0 inside
!
object network webserver
host 10.12.2.5
object service myport
service tcp source range 1 65535 destination range 1 65535
access-list outowebaccess extended permit object myport any any log disable
access-group outowebaccess in interface outside
object service app
service tcp source eq www
nat (inside,outside) source static webserver interface destination static interface any
 service app app
object network obj-any
subnet 0.0.0.0 0.0.0.0
nat (inside,outside) source dynamic obj-any interface destination static obj-any obj-any
configure terminal
dns domain-lookup management
policy-map global_policy
class inspection_default
inspect icmp
call-home
profile License
destination transport-method http
destination address http https://tools.cisco.com/its/service/oddce/services/DDCEService
license smart
feature tier standard
throughput level 2G
license smart register idtoken <TOKEN>
: end
```

# Azure Function App パッケージの構築

ASAv Auto Scale ソリューションでは、*ASM_Function.zip* アーカイブファイルを作成する必要があります。このファイルから、圧縮された ZIP パッケージの形式で一連の個別の Azure 関数が提供されます。

ASM_Function.zip パッケージの作成方法については、「ソースコードからの Azure 関数の構築（31 ページ）」を参照してください。

関数は、特定のタスクを実行するために可能な限り独立しており、拡張機能や新しいリリースのサポートのために必要に応じてアップグレードできます。

# 入力パラメータ

次の表に、テンプレートパラメータおよび例を示します。各パラメータの値を決めたら、Azure サブスクリプションに ARM テンプレートを展開するときに、各パラメータを使用して ASAv デバイスを作成できます。「Auto Scale ARM テンプレートの展開 （13 ページ）」を参照してください。

**表 *1:* テンプレートパラメータ**

| パラメータ名 | 使用できる値/タイプ | 説明 | リソースの作成タイプ |
|---|---|---|---|
| resourceNamePrefix | 文字列*（3 ～ 10 文字） | すべてのリソースは、このプレフィックスを含む名前で作成されます。<br>注：小文字のみを使用してください。<br>例：asav | 新規作成 |
| virtualNetworkRg | 文字列 | 仮想ネットワークのリソースグループの名前。<br>例：cisco-virtualnet-rg | 既存 |
| virtualNetworkName | 文字列 | 仮想ネットワーク名（作成済み）<br>例：cisco-virtualnet | 既存 |
| mgmtSubnet | 文字列 | 管理サブネット名（作成済み）<br>例：cisco-mgmt-subnet | 既存 |

| パラメータ名 | 使用できる値/タイプ | 説明 | リソースの作成タイプ |
|---|---|---|---|
| insideSubnet | 文字列 | 内部サブネット名（作成済み）<br>例：cisco-inside-subnet | 既存 |
| internalLbIp | 文字列 | 内部サブネットの内部ロードバランサの IP アドレス（作成済み）。<br>例：1.2.3.4 | 既存 |
| outsideSubnet | 文字列 | 外部サブネット名（作成済み）<br>例：cisco-outside-subnet | 既存 |
| softwareVersion | 文字列 | ASAvバージョン（展開時にドロップダウンから選択）<br>デフォルト：914.1.0許可：914.1.0, 913.1.0 | 既存 |
| vmSize | 文字列 | ASAv インスタンスのサイズ（展開時にドロップダウンから選択） | 該当なし |
| asaAdminUserName | 文字列 * | ASAv「admin」ユーザーのユーザー名。<br>パスワードの長さは 12 ー 72 文字で、小文字、大文字、数字、特殊文字を使用する必要があります。また、文字の繰り返しは 2 回までにする必要があります。<br>これは「admin」には**できません**。VM 管理者ユーザー名のガイドラインについては、「Azure」を参照してください。<br>（注） テンプレートには、このパラメータのコンプライアンスチェック機能はありません。 | 新規作成 |

| パラメータ名 | 使用できる値/タイプ | 説明 | リソースの作成タイプ |
|---|---|---|---|
| asaAdminUserPassword | 文字列 * | ASAv 管理者ユーザのパスワード。<br>パスワードの長さは 12 ー 72 文字で、小文字、大文字、数字、特殊文字を使用する必要があります。また、文字の繰り返しは 2 回までにする必要があります。<br>（注） テンプレートには、このパラメータのコンプライアンスチェック機能はありません。 | 新規作成 |
| scalingPolicy | POLICY-1/POLICY-2 | **POLICY-1**：設定された期間に、いずれかの ASAv の平均負荷がスケールアウトしきい値を超えるとスケールアウトがトリガーされます。<br>**POLICY-2**：設定された期間に、Auto Scale グループ内のすべての ASAv デバイスの平均負荷がスケールアウトしきい値を超えるとスケールアウトがトリガーされます。<br>どちらの場合も、スケールインロジックは同じままです。設定された期間に、すべての ASAv デバイスの平均負荷がスケールインしきい値を下回るとスケールインがトリガーされます。 | 該当なし |
| scalingMetricsList | 文字列 | スケーリングの決定に使用されるメトリック。<br>許可：CPU<br>デフォルト：CPU | 該当なし |

| パラメータ名 | 使用できる値/タイプ | 説明 | リソースの作成タイプ |
|---|---|---|---|
| scaleInThreshold | 文字列 | スケールインしきい値（パーセント単位）。<br>デフォルト：10<br>ASAvメトリック（CPU 使用率）がこの値を下回ると、スケールインがトリガーされます。<br>「Auto Scale ロジック （28 ページ）」を参照してください。 | 該当なし |
| scaleOutThreshold | 文字列 | スケールアウトしきい値（パーセント単位）。<br>デフォルト：80<br>ASAvメトリック（CPU 使用率）がこの値を上回ると、スケールアウトがトリガーされます。<br>「scaleOutThreshold」は、常に「scaleInThreshold」より**大きく**する必要があります。<br>「Auto Scale ロジック （28 ページ）」を参照してください。 | 該当なし |
| minAsaCount | 整数 | 任意の時点でスケールセットで使用可能な最小 ASAv インスタンス数。<br>例：2。 | 該当なし |
| maxAsaCount | 整数 | スケールセットで許可される最大 ASAv インスタンス数。<br>例：10<br>（注） Auto Scale ロジックではこの変数の範囲はチェックされないため、慎重に入力してください。 | 該当なし |

| パラメータ名 | 使用できる値/タイプ | 説明 | リソースの作成タイプ |
|---|---|---|---|
| metricsAverageDuration | 整数 | ドロップダウンから選択します。<br>この数値は、メトリックが平均化される時間（分単位）を表します。<br>この変数の値が 5（5 分）の場合、Auto Scale Manager がスケジュールされると、メトリックの過去 5 分間の平均がチェックされ、その結果に基づいてスケーリングの判断が行われます。<br>（注） Azure の制限により、有効な数値は 1、5、15、および 30 だけです。 | 該当なし |
| initDeploymentMode | BULK/STEP | 主に最初の展開、またはスケールセットに ASAv インスタンスが含まれていない場合に適用されます。<br>BULK：Auto Scale Manager は、「minAsaCount」個の ASAv インスタンスを同時に展開しようとします。<br>STEP：Auto Scale Manager は、スケジュールされた間隔ごとに「minAsaCount」個の ASAv デバイスを 1 つずつ展開します。 | |
| configurationFile | 文字列 | ASAv 構成ファイルのファイルパス。<br>例：<br>https://myserver/asavconfig/asaconfig.txt | 該当なし |
| * Azure には、新しいリソースの命名規則に関する制限があります。制限を確認するか、またはすべて小文字を使用してください**スペースやその他の特殊文字は使用しないでください**。 | | | |

# Auto Scale の展開

## Auto Scale ARM テンプレートの展開

：ARM テンプレート **azure_asav_autoscale.json** を使用して、Azure 用 ASAv Auto Scale に必要なリソースを展開します。特定のリソースグループ内では、ARM テンプレートを展開することで次の内容が作成されます。

- 仮想マシンスケールセット（VMSS）
- 外部ロードバランサ
- 内部ロードバランサ
- Azure Function App
- Logic App
- セキュリティグループ（データインターフェイスおよび管理インターフェイス用）

### 始める前に

- GitHub リポジトリ（https://github.com/CiscoDevNet/cisco-asav/tree/master/autoscale/azure）から、ARM テンプレートをダウンロードします。

**ステップ 1** 複数の Azure ゾーンに ASAv インスタンスを展開する必要がある場合は、展開リージョンで使用可能なゾーンに基づいて、ARM テンプレートを編集します。

**例**：

```
"zones": [
    "1",
    "2",
    "3"
],
```

この例は、3 つのゾーンを持つ「Central US」リージョンを示しています。

**ステップ 2** 外部ロードバランサで必要なトラフィックルールを編集します。この「json」配列を拡張することで、任意の数のルールを追加できます。

**例**：

```
{
 "type": "Microsoft.Network/loadBalancers",
 "name": "[variables('elbName')]",
 "location": "[resourceGroup().location]",
 "apiVersion": "2018-06-01",
 "sku": {
   "name": "Standard"
```

```
        },
        "dependsOn": [
          "[concat('Microsoft.Network/publicIPAddresses/', variables('elbPublicIpName'))]"

        ],
        "properties": {
          "frontendIPConfigurations": [
            {
              "name": "LoadBalancerFrontEnd",
                "properties": {
                  "publicIPAddress": {
                    "id": "[resourceId('Microsoft.Network/publicIPAddresses/',
variables('elbPublicIpName'))]"
                  }
                }
            }
          ],
          "backendAddressPools": [
            {
              "name": "backendPool"
            }
          ],
          "loadBalancingRules": [
            {
              "properties": {
                "frontendIPConfiguration": {
                "Id": "[concat(resourceId('Microsoft.Network/loadBalancers', variables('elbName')),
 '/frontendIpConfigurations/LoadBalancerFrontend')]"
                },
                "backendAddressPool": {
                "Id": "[concat(resourceId('Microsoft.Network/loadBalancers', variables('elbName')),
 '/backendAddressPools/BackendPool')]"
                },
                "probe": {
                "Id": "[concat(resourceId('Microsoft.Network/loadBalancers', variables('elbName')),
 '/probes/lbprobe')]"
                },
                "protocol": "TCP",
                "frontendPort": "80",
                "backendPort": "80",
                "idleTimeoutInMinutes": "[variables('idleTimeoutInMinutes')]"
              },
              "Name": "lbrule"
            }
          ],
```

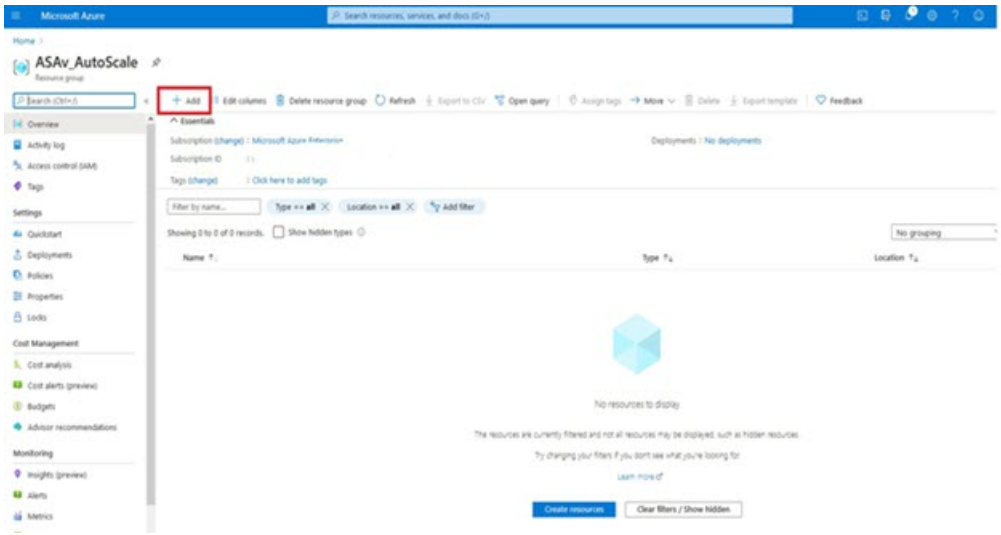
（注） このファイルを編集しない場合は、導入後に Azure ポータルから編集することもできます。

**ステップ 3** Microsoft アカウントのユーザー名とパスワードを使用して、Microsoft Azure ポータルにログインします。

**ステップ 4** [リソースグループ（Resource Groups）] ブレードにアクセスするには、サービスのメニューから [リソースグループ（Resource groups）] をクリックします。サブスクリプション内のすべてのリソースグループがブレードに一覧表示されます。

新しいリソースグループを作成するか、既存の空のリソースグループを選択します。たとえば、*ASAv_AutoScale*。

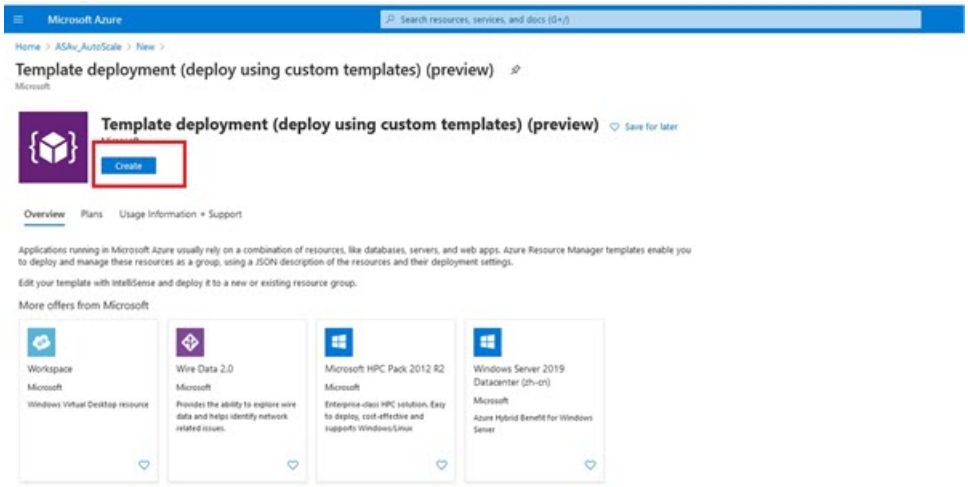
図 *2 : Azure* ポータル




**ステップ 5** [リソースの作成（+）（Create a resource (+)）] をクリックして、テンプレート展開用の新しいリソースを作成します。[リソースグループの作成（Create Resource Group）] ブレードが表示されます。

**ステップ 6** [マーケットプレイスの検索（Search the Marketplace）] で、「テンプレートの展開（カスタムテンプレートを使用した展開）（Template deployment (deploy using custom templates)）」と入力し、Enter を押します。

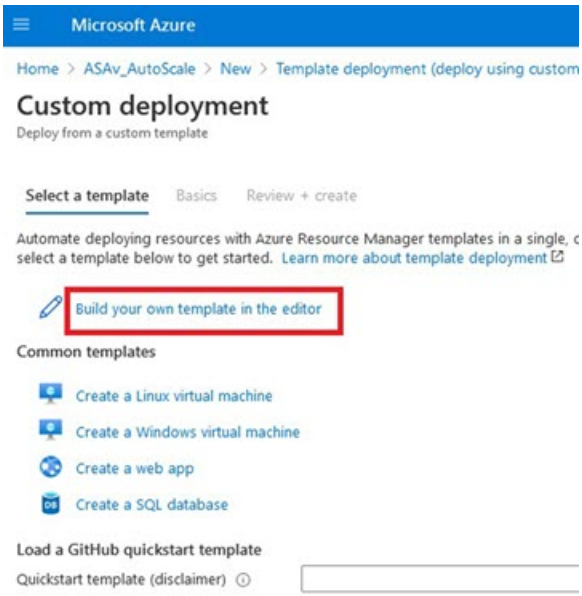
図 *3 :* カスタムテンプレートの展開




**ステップ 7** [作成（Create）] をクリックします。

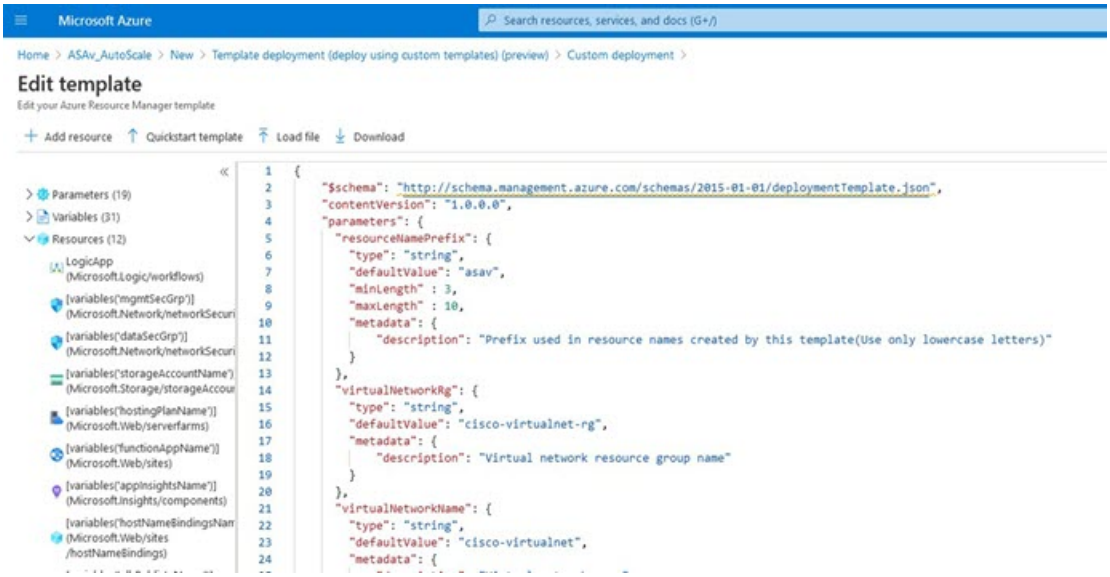
**ステップ 8** テンプレートを作成するためのオプションは複数あります。[エディタで独自のテンプレートを作成する（Build your own template in editor）] を選択します。

図 *4 :* 独自のテンプレートの作成



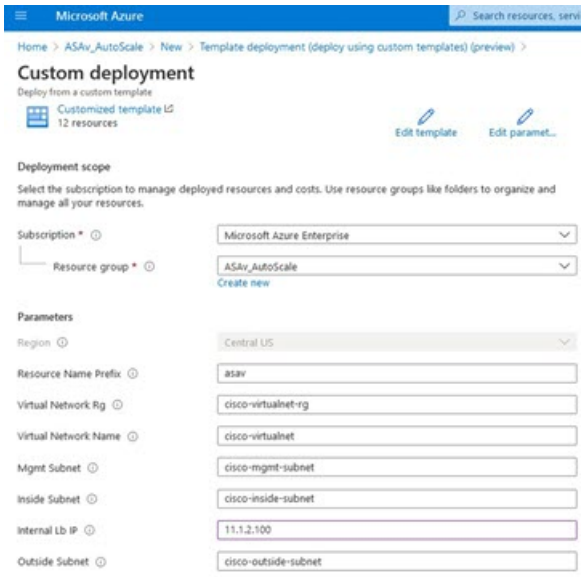

**ステップ 9** [テンプレートの編集（Edit template）] ウィンドウで、すべてのデフォルトコンテンツを削除し、更新した *azure_asav_autoscale.json* からコンテンツをコピーして、[保存（Save）] をクリックします。

図 *5 : Edit Template*




**ステップ 10** 次のセクションで、すべてのパラメータを入力します。各パラメータの詳細については、「入力パラメータ （8 ページ）」を参照してください。次に、[購入（Purchase）] をクリックします。

図 *6 : ARM* テンプレートパラメータ

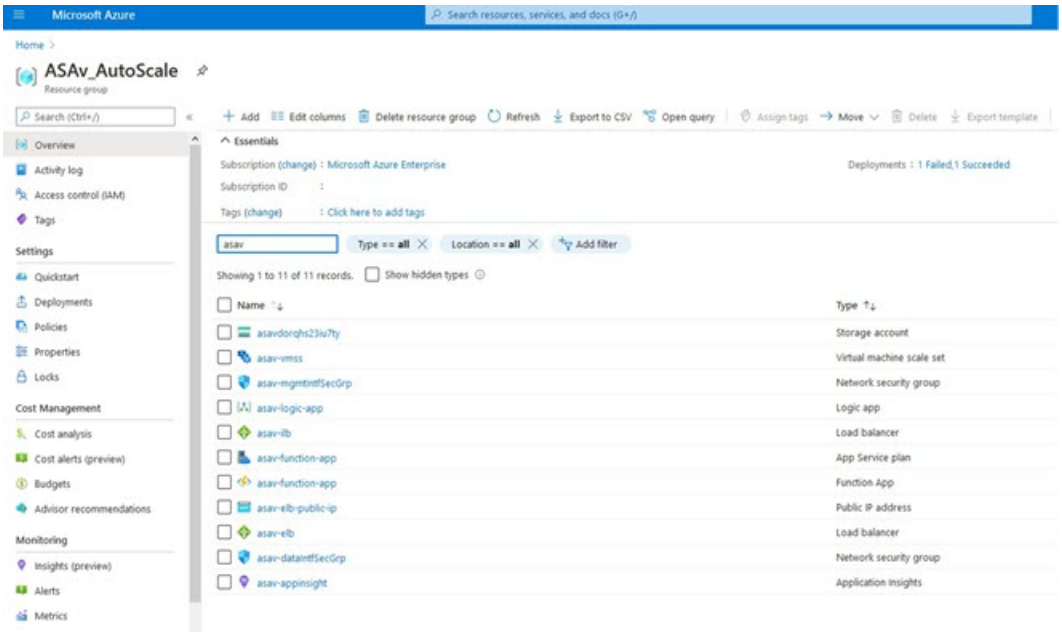



（注） [パラメータの編集（Edit Parameters）] をクリックして、JSON ファイルを編集するか、または事前入力されたコンテンツをアップロードできます。

ARM テンプレートの入力検証機能は限られているため、入力を検証するのはユーザーの責任です。

**ステップ 11** テンプレートの展開が成功すると、ASAv Auto Scale for Azure ソリューションに必要なすべてのリソースが作成されます。次の図のリソースを参照してください。[タイプ（Type）]列には、Logic App、VMSS、ロードバランサ、パブリック IP アドレスなどの各リソースが示されます。

図 *7: ASAv* 自動スケールテンプレートの展開




# Azure Function App の展開

ARM テンプレートを展開すると、Azure によってスケルトン Function App が作成されます。このアプリは、Auto Scale Manager ロジックに必要な関数を使用して手動で更新および設定する必要があります。

### 始める前に

- ASM_Function.zip パッケージをビルドします。「ソースコードからの Azure 関数の構築（31 ページ）」を参照してください。

**ステップ 1** ARM テンプレートを展開したときに作成した Function App に移動し、関数が存在しないことを確認します。ブラウザで次の URL にアクセスします。

https://<Function App Name>.scm.azurewebsites.net/DebugConsole

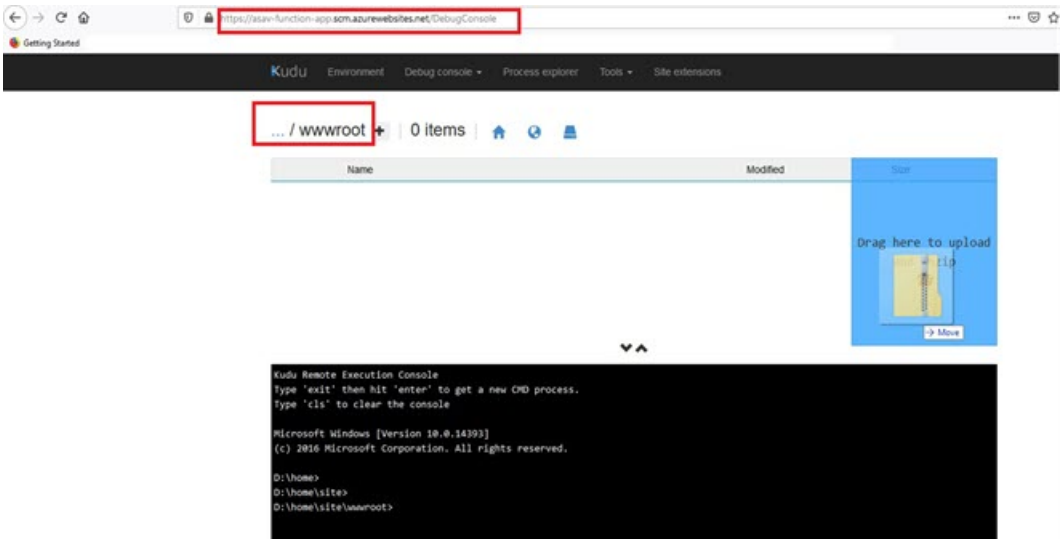
「Auto Scale ARM テンプレートの展開 （13 ページ）」の例の場合、次のようになります。

https://asav-function-app.scm.azurewebsites.net/DebugConsole

**ステップ 2** ファイルエクスプローラで、site/wwwroot に移動します。

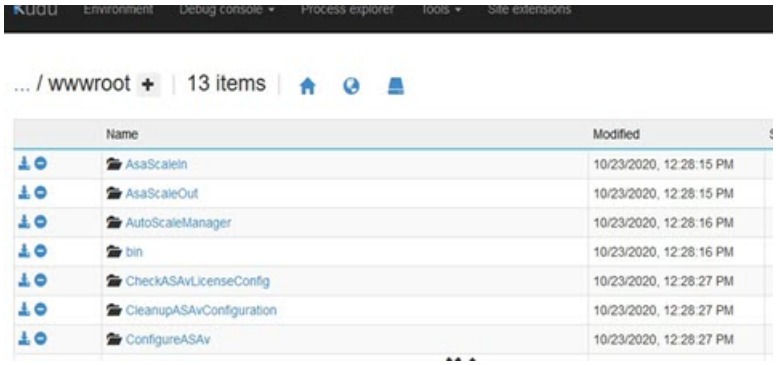
**ステップ 3** ASM_Function.zip をファイルエクスプローラの右隅にドラッグアンドドロップします。

図 *8 : ASAv Auto Scale* 機能のアップロード




**ステップ 4** アップロードが成功すると、すべてのサーバーレス関数が表示されます。

図 *9 : ASAv* のサーバーレス機能

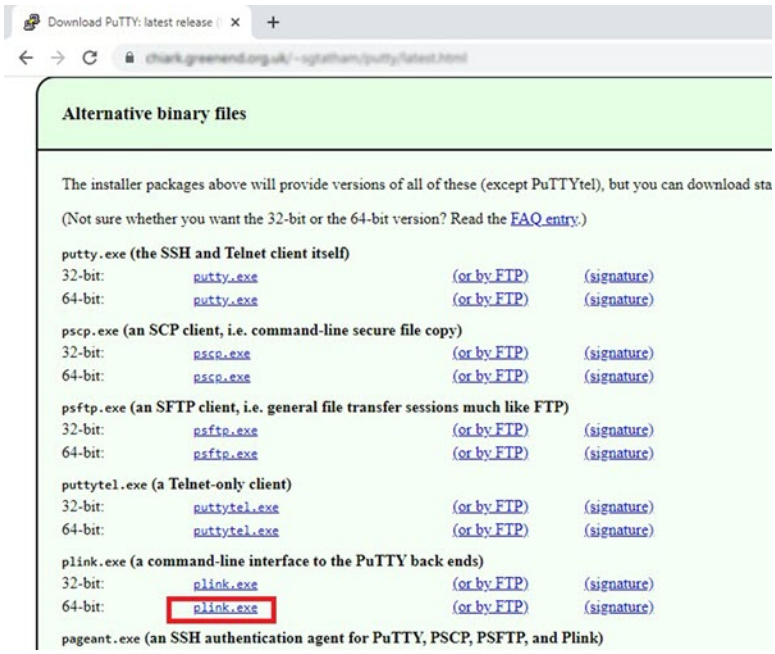



**ステップ 5** PuTTY SSH クライアントをダウンロードします。

Azure 関数は、SSH 接続を介して ASAv にアクセスする必要があります。ただし、サーバーレスコードで使用されるオープンソースライブラリは、ASAv で使用される SSH キー交換アルゴリズムをサポートしていません。したがって、事前に構築された SSH クライアントをダウンロードする必要があります。

www.putty.org から PuTTY コマンドライン インターフェイスを PuTTY バックエンド（plink.exe）にダウンロードします。

図 *10 : PuTTY* のダウンロード




**ステップ 6** SSH クライアントの実行ファイル **plink.exe** の名前を **asassh.exe** に変更します。

**ステップ 7** **asassh.exe** をファイルエクスプローラの右隅（前のステップで **ASM_Function.zip** をアップロードした場所）にドラッグアンドドロップします。

**ステップ 8** SSH クライアントが Function App とともに存在することを確認します。必要に応じてページを更新します。

# 設定の微調整

Auto Scale Manager を微調整したり、デバッグで使用したりするために使用できる設定がいくつかあります。これらのオプションは、ARM テンプレートには表示されませんが、Function App で編集できます。

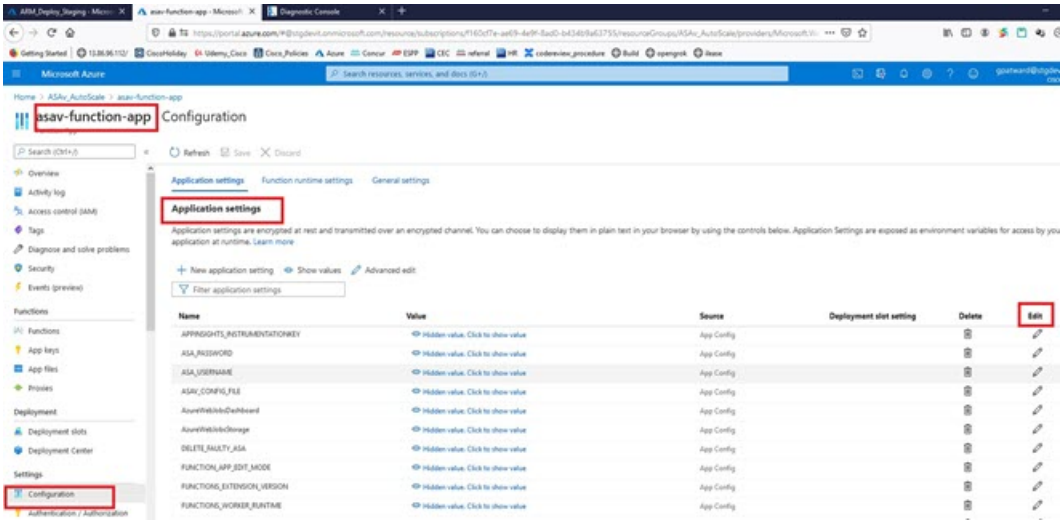
### 始める前に

（注） 設定はいつでも編集できます。設定を編集する場合は、次の手順に従います。

- Function App を無効にします。
- 既存のスケジュール済みタスクが終了するまで待ちます。
- 設定を編集して保存します。
- Function App を有効にします。

**ステップ 1** Azure ポータルで、ASAv Function App を検索して選択します。

**図 *11 : ASAv* 機能アプリケーション**




**ステップ 2** ここでは、ARM テンプレートを介して渡された設定も編集できます。変数名は、ARM テンプレートとは異なる場合がありますが、変数の目的は名前から簡単に識別できます。

ほとんどのオプションは、名前を見ればわかります。次に例を示します。

- [構成名（Configuration Name）] :「DELETE_FAULTY_ASA」（[デフォルト値]（Default value ）] : YES）

  スケールアウト中に、新しい ASAv インスタンスが起動し、 構成ファイルを介して設定されますFMC に登録されます。設定が失敗した場合、このオプションに基づいて、Auto Scale Manager がその ASAv インスタンスを保持するか、削除するかを決定します。（[はい（Yes）] : 障害のある ASAv を削除します。[いいえ（No）] : 設定が失敗した場合でも、ASAv インスタンスを保持します）。

- Function App 設定では、Azure サブスクリプションにアクセスできるユーザーは、すべての変数（「password」などのセキュアな文字列を含んでいる変数を含む）をクリアテキスト形式で表示できます。

  この点に関するセキュリティ上の懸念がある場合（たとえば、Azure サブスクリプションが組織内の低い権限を持つユーザー間で共有されている場合）、ユーザーは Azure の Key Vault サービスを使用してパスワードを保護できます。この設定をすると、関数の設定でクリアテキストの「password」を入力する代わりに、ユーザーは、パスワードが保存されている Key Vault によって生成された、セキュアな識別子を入力する必要があります。

  （注） Azure のドキュメントを検索して、アプリケーションデータを保護するためのベストプラクティスを見つけてください。

# 仮想マシンスケールセットでの IAM ロールの設定

Azure Identity and Access Management（IAM）は、Azure Security and Access Control の一部として使用され、ユーザーの ID を管理および制御します。Azure リソースのマネージド ID は、Azure Active Directory で自動的にマネージド ID が Azure サービスに提供されます。

これにより、明示的な認証ログイン情報がなくても、Function App が仮想マシンスケールセット（VMSS）を制御できます。

**ステップ 1** Azure ポータルで、VMSS に移動します。

**ステップ 2** [アクセス制御（IAM）（Access control (IAM)）] をクリックします。

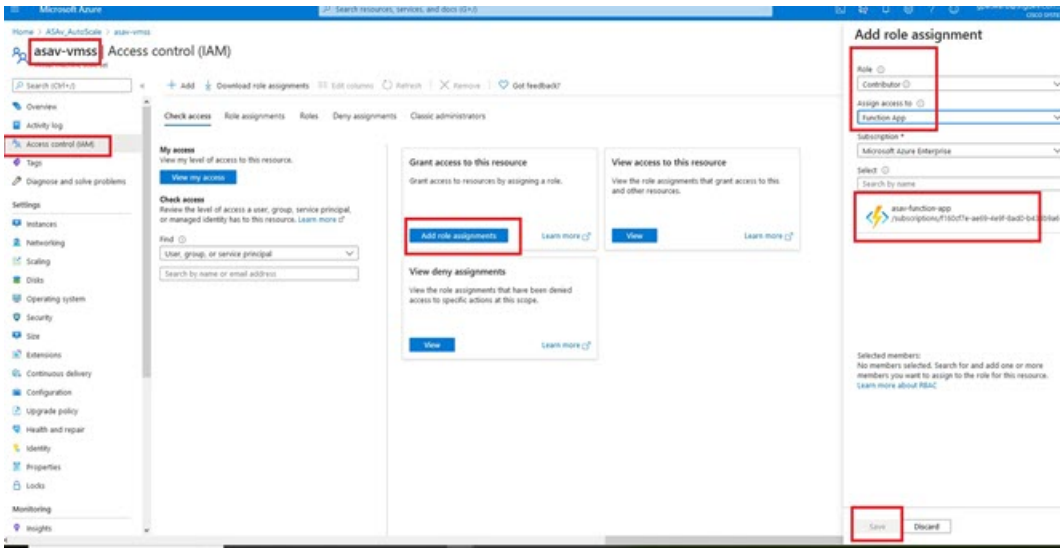
**ステップ 3** [追加（Add）] をクリックしてロールの割り当てを追加します。

**ステップ 4** [ロール割り当ての追加（Add role assignment）] ドロップダウンから、[共同作成者（Contributor）] を選択します。

**ステップ 5** [アクセスの割り当て先（Assign access to）] ドロップダウンから、[Function App] を選択します。

**ステップ 6** ASAv Function App を選択します。

**図 *12 : AIM* ロールの割り当て**




**ステップ 7** [保存（Save）] をクリックします。

（注） まだ ASAv インスタンスが起動していないことも確認する必要があります。

# Azure セキュリティグループの更新

ARM テンプレートは、管理インターフェイス用とデータインターフェイス用の 2 つのセキュリティグループを作成します。管理セキュリティグループは、ASAv 管理アクティビティに必

要なトラフィックのみを許可します。ただし、データインターフェイスのセキュリティグループはすべてのトラフィックを許可します。

展開のトポロジとアプリケーションのニーズに基づいてセキュリティグループのルールを微調整します。

（注） データインターフェイスのセキュリティグループは、少なくともロードバランサからの SSH トラフィックを許可する必要があります。

# Azure Logic App の更新

Logic App は、Auto Scale 機能の Orchestrator として機能します。ARM テンプレートによってスケルトン Logic App が作成されます。このアプリケーションを手動で更新して、Auto Scale Orchestrator として機能するために必要な情報を提供する必要があります。

**ステップ 1** リポジトリから、LogicApp.txt ファイルをローカルシステムに取得し、次のように編集します。

**重要** 手順をすべて読んで理解してから続行してください。

手動の手順は、ARM テンプレートでは自動化されないため、Logic App のみ後で個別にアップグレードできます。

a) 必須: すべての「SUBSCRIPTION_ID」を検索し、サブスクリプション ID 情報に置き換えます。
b) 必須: すべての「RG_NAME」を検索し、リソースグループ名に置き換えます。
c) 必須: すべての「FUNCTIONAPPNAME」を検索し、Function App 名に置き換えます。

次の例は、LogicApp.txt ファイルの行の一部を示しています。

```
  "AutoScaleManager": {
      "inputs": {
          "function": {
              "id":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RG_NAME/providers/Microsoft.Web/sites/FUNCTIONAPPNAME/functions/AutoScaleManager"

          }
.
.
                      },
                      "Deploy_Changes_to_ASA": {
                          "inputs": {
                              "body": "@body('AutoScaleManager')",
                              "function": {
                                  "id":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RG_NAME/providers/Microsoft.Web/sites/FUNCTIONAPPNAME/functions/DeployConfiguration"

                              }
.
.
                      "DeviceDeRegister": {
                          "inputs": {
                              "body": "@body('AutoScaleManager')",
```

```
                                "function": {
                                    "id":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RG_NAME/providers/Microsoft.Web/sites/FUNCTIONAPPNAME/functions/DeviceDeRegister"
                                }
                            },
                            "runAfter": {
                                "Delay_For_connection_Draining": [
```

d) （任意）トリガー間隔を編集するか、デフォルト値（5）のままにします。これは、Auto Scale 機能が定期的にトリガーされる時間間隔です。次の例は、LogicApp.txt ファイルの行の一部を示しています。

```
"triggers": {
    "Recurrence": {
        "conditions": [],
        "inputs": {},
        "recurrence": {
            "frequency": "Minute",
            "interval": 5
        },
```

e) （任意）ドレインする時間を編集するか、デフォルト値（5）のままにします。これは、スケールイン操作中にデバイスを削除する前に、ASAv から既存の接続をドレインする時間間隔です。次の例は、LogicApp.txt ファイルの行の一部を示しています。

```
"actions": {
     "Branch_based_on_Scale-In_or_Scale-Out_condition": {
         "actions": {
             "Delay_For_connection_Draining": {
                 "inputs": {
                     "interval": {
                         "count": 5,
                         "unit": "Minute"
                     }
```

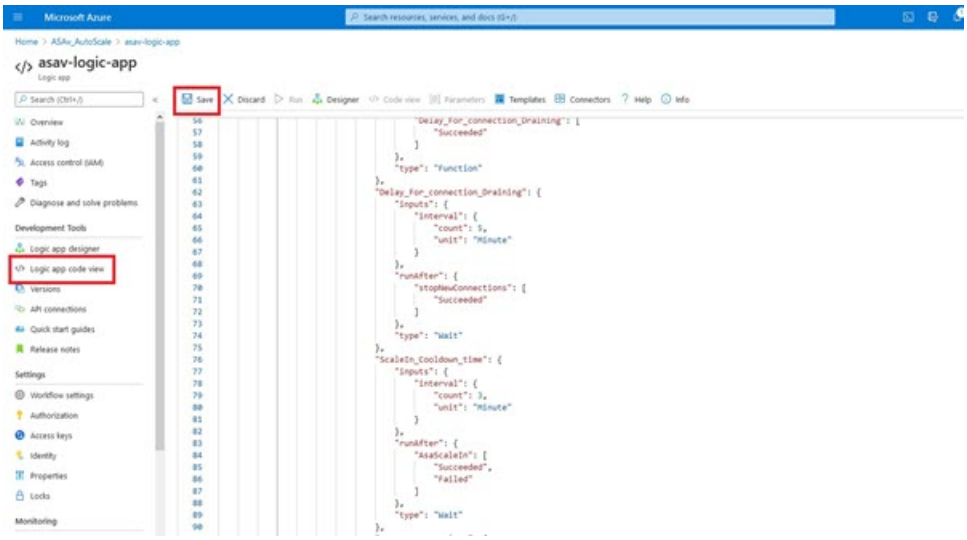
f) （任意）クールダウン時間を編集するか、デフォルト値（10）のままにします。これは、スケールアウト完了後に NO ACTION を実行する時間です。次の例は、LogicApp.txt ファイルの行の一部を示しています。

```
"actions": {
    "Branch_based_on_Scale-Out_or_Invalid_condition": {
        "actions": {
            "Cooldown_time": {
                "inputs": {
                    "interval": {
                        "count": 10,
                        "unit": "Second"
                    }
```

（注） これらの手順は、Azure ポータルからも実行できます。詳細については、Azure のドキュメントを参照してください。

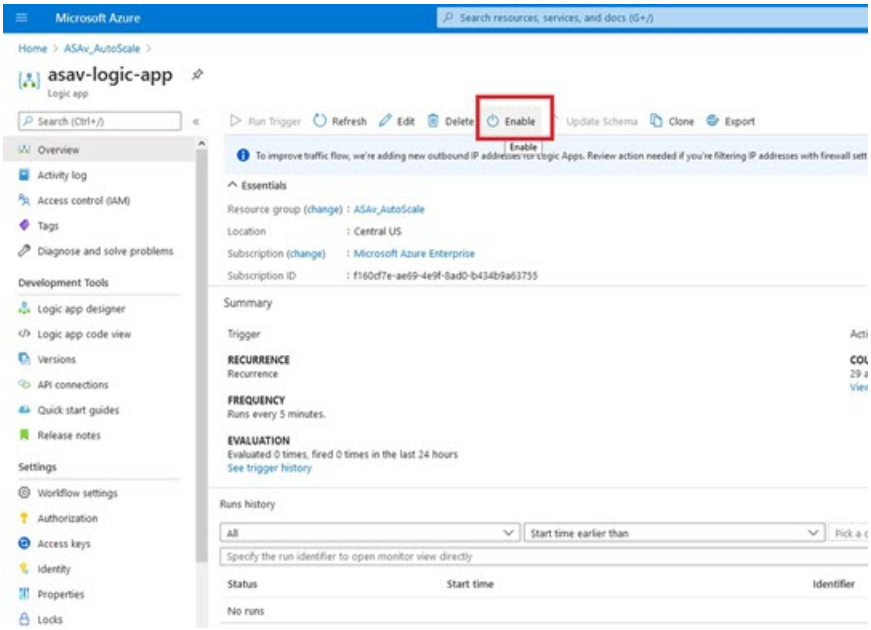
**ステップ 2** [Logic Appコードビュー（Logic App code view）] に移動し、デフォルトの内容を削除して、編集した LogicApp.txt ファイルの内容を貼り付け、[保存（Save）] をクリックします。

図 *13 : Logic App* コードビュー



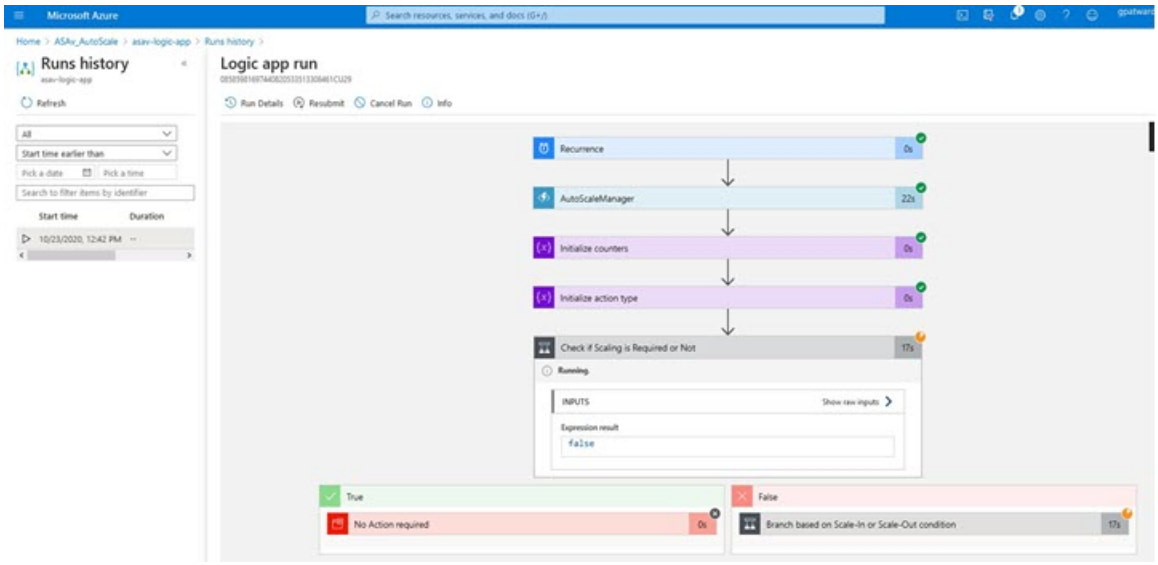

**ステップ 3** Logic App を保存すると、[無効（Disabled）] 状態になります。Auto Scale Manager を起動する場合は、[有効化（Enable）] をクリックします。

図 *14 : Logic App* の有効化




**ステップ 4** 有効にすると、タスクの実行が開始されます。[実行中（Running）] ステータスをクリックしてアクティビティを表示します。

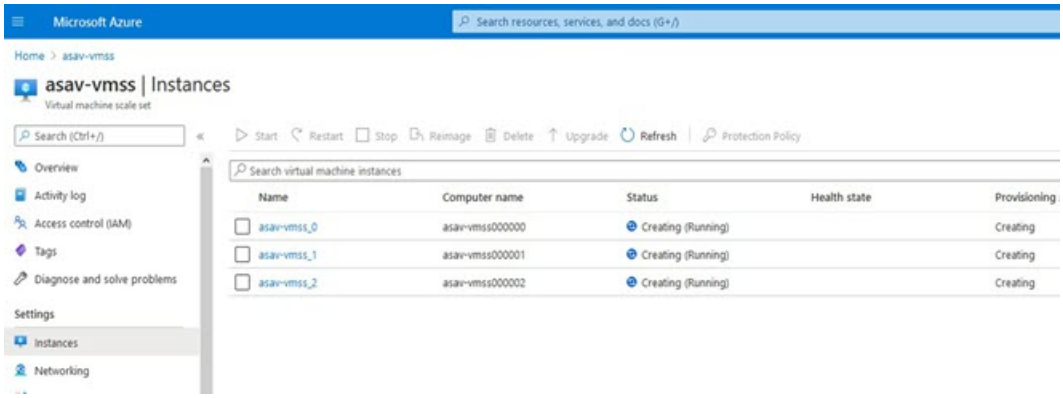
図 *15 : Logic App* の実行ステータス




**ステップ 5** Logic App が起動すると、導入関連のすべての手順が完了します。

**ステップ 6** ASAv インスタンスが作成されていることを VMSS で確認します。

図 *16 :* 稼働中の*ASAv* インスタンス




この例では、ARM テンプレートの展開で「'minAsaCount'」が「3」に設定され、「initDeploymentMode」が「BULK」に設定されているため、3 つの ASAv インスタンスが起動されます。

# ASAvのアップグレード

ASAv アップグレードは、仮想マシンスケールセット（VMSS）のイメージアップグレードの形式でのみサポートされます。したがって、ASAv は Azure REST API インターフェイスを介してアップグレードします。

（注） 任意の REST クライアントを使用して ASAv をアップグレードできます。

**始める前に**

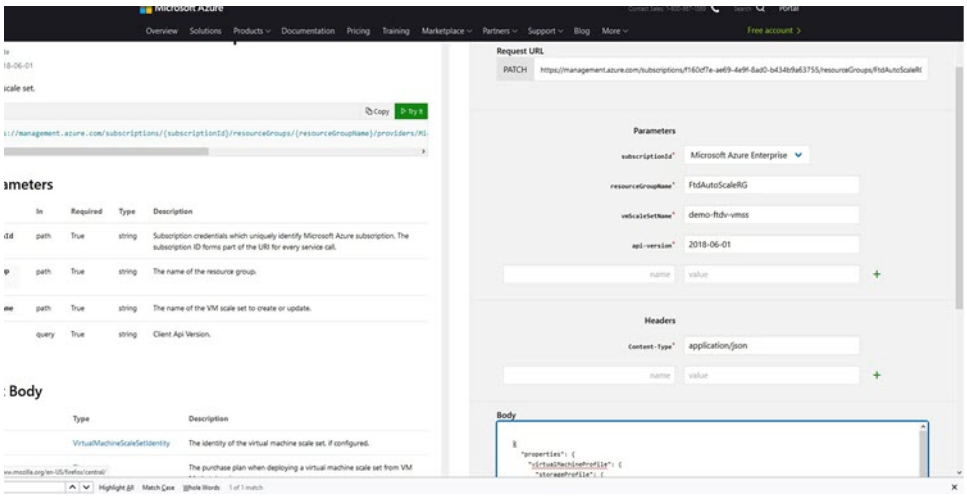
- 市場で入手可能な新しい ASAv イメージバージョンを取得します（例：914.001）。
- 元のスケールセットの展開に使用する SKU を取得します（例：asav-azure-byol）。
- リソースグループと仮想マシンスケールセット名を取得します。

**ステップ 1** ブラウザで次の URL にアクセスします。

https://docs.microsoft.com/en-us/rest/api/compute/virtualmachinescalesets/update#code-try-0

**ステップ 2** [パラメータ（Parameters）] セクションに詳細を入力します。

**図 *17 : ASAv*のアップグレード**




**ステップ 3** 新しい ASAv イメージバージョン、SKU、トリガー RUN を含む JSON 入力を **[本文（Body）]** セクションに入力します。

```
{
 "properties": {
        "virtualMachineProfile": {
              "storageProfile": {
                      "imageReference": {
                           "publisher": "cisco",
                           "offer": "cisco-asav",
                           "sku": "asav-azure-byol",
                            "version": "650.32.0"
                      }
                  },
            }
      }
```

```
}
```

**ステップ 4** VMSS が変更を受け入れると、Azure から成功の応答が返ってきます。

新しいイメージは、スケールアウト操作の一環として起動される新しい ASAv インスタンスで使用されます。

- 既存の ASAv インスタンスは、スケールセットに存在している間、古いソフトウェアイメージを使用し続けます。
- 前述の動作を上書きし、既存の ASAv インスタンスを手動でアップグレードできます。これを行うには、VMSS の [アップグレード（Upgrade）] ボタンをクリックします。選択した ASAv インスタンスが再起動されて、アップグレードされます。アップグレードされた ASAv インスタンスは手動で再登録および再設定する必要があります。**この方法は推奨されません。**

# Auto Scale ロジック

### スケールアウトロジック

- **POLICY-1**：設定された期間に、**いずれか** ASAvの平均負荷がスケールアウトしきい値を超えるとスケールアウトがトリガーされます。
- **POLICY-2**：設定された期間に、**すべて**の ASAv デバイスの平均負荷がスケールアウトしきい値を超えるとスケールアウトがトリガーされます。「

### スケールインロジック

- 設定された期間に、**すべて**の ASAv デバイスの CPU 使用率が設定されたスケールインしきい値を下回った場合。

### 注意

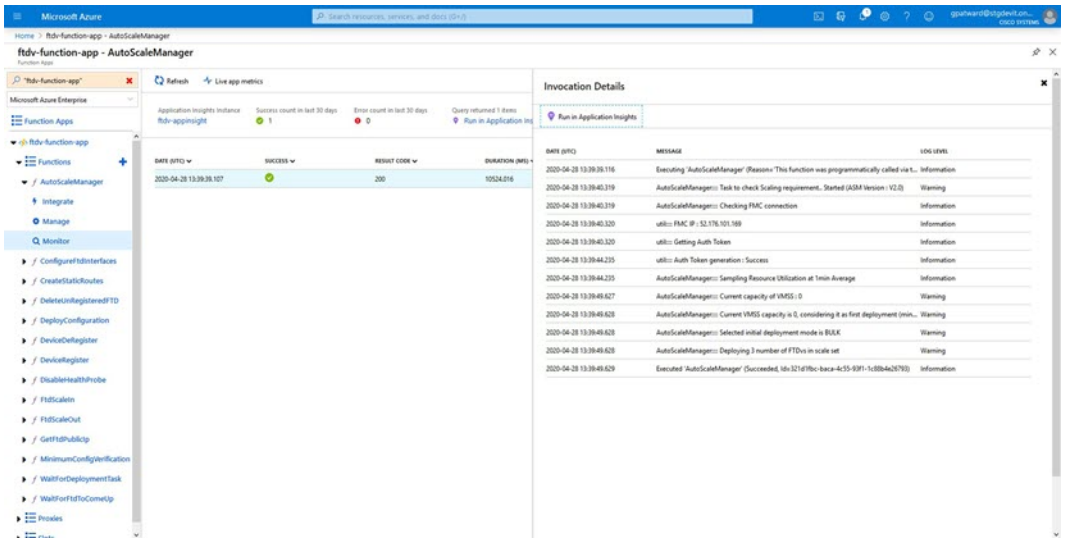
- スケールイン/スケールアウトは 1 つずつ行われます（つまり、一度に 1 つの ASAv だけがスケールインまたはスケールアウトされます）。
- 上記のロジックは、ロードバランサがすべての ASAv デバイスに接続を均等に分散しようとし、平均してすべての ASAv デバイスが均等にロードされるという前提に基づいています。

# Auto Scale のロギングとデバッグ

サーバーレスコードの各コンポーネントには、独自のロギングメカニズムがあります。また、ログはアプリケーションインサイトにパブリッシュされます。

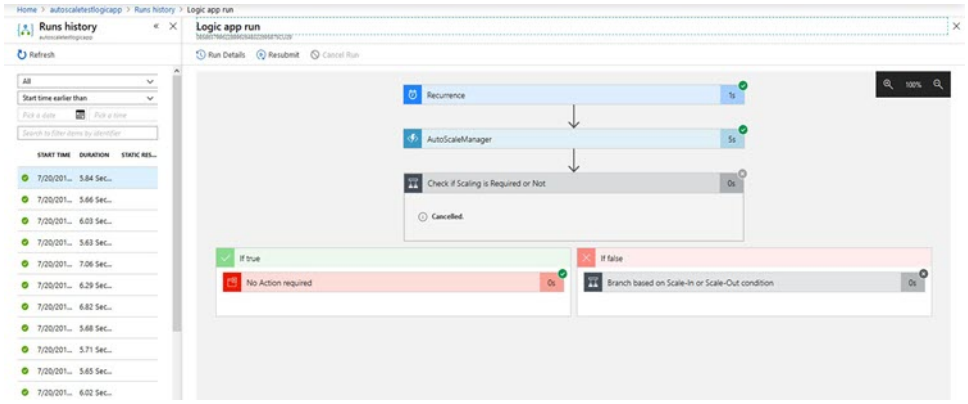
- 個々の Azure 関数のログを表示できます。

**図 *18 : Azure* 関数ログ**




- Logic App とその個々のコンポーネントの実行ごとに同様のログを表示できます。

**図 *19 : Logic App* の実行ログ**




- 必要な場合は、Logic App で実行中のタスクをいつでも停止または終了できます。ただし、現在実行中の ASAv デバイスが起動または終了すると、一貫性のない状態になります。
- 各実行または個々のタスクにかかった時間は、Logic App で確認できます。
- Function App は、新しい zip をアップロードすることでいつでもアップグレードできます。Logic App を停止し、すべてのタスクの完了を待ってから、Function App をアップグレードします。

# Auto Scale のガイドラインと制約事項

ASAv Auto Scale for Azure を導入する場合は、次のガイドラインと制限事項に注意してください。

- スケーリングの決定は、CPU 使用率に基づきます。
- ASAv 管理インターフェイスは、パブリック IP アドレスを持つように設定されます。
- IPv4 だけがサポートされます。
- ARM テンプレートの入力検証機能は限られているため、入力を正しく検証するのはユーザーの責任です。
- Azure 管理者は、Function App 環境内の機密データ（管理者ログイン情報やパスワードなど）をプレーンテキスト形式で確認できます。Azure Key Vault サービスを使用して、センシティブデータを保護できます。
- 設定の変更は、すでに実行中のインスタンスには自動的に反映されません。変更は新しいデバイスにのみ反映されます。このような変更は、既存のデバイスに手動でプッシュする必要があります。
- 既存のインスタンスの設定を手動で更新しているときに問題が発生した場合は、それらのインスタンスをスケーリンググループから削除し、新しいインスタンスに置き換えることを推奨します。

# Auto Scale のトラブルシューティング

次に、ASAv Auto Scale for Azure の一般的なエラーシナリオとデバッグのヒントを示します。

- ASAv に SSH 接続できない：複雑なパスワードがテンプレートを介して ASAv に渡されているか確認します。セキュリティグループで SSH 接続が許可されているか確認します。
- ロードバランサのヘルスチェックエラー：ASAv がデータインターフェイスの SSH に応答しているか確認します。セキュリティグループの設定を確認します。
- トラフィックの問題：ロードバランサルール、ASAv で設定された NAT ルールおよびスタティックルートを確認します。テンプレートとセキュリティグループルールで提供される Azure 仮想ネットワーク/サブネット/ゲートウェイの詳細を確認します。
- Logic App が VMSS にアクセスできない：VMSS の IAM ロール設定が正しいか確認します。
- Logic App の実行時間が長すぎる：スケールアウトされた ASAv デバイスで SSH アクセスを確認します。Azure VMSS で ASAv デバイスの状態を確認します。
- サブスクリプション ID 関連の Azure 関数のスローエラー：アカウントでデフォルトのサブスクリプションが選択されていることを確認します。

- スケールイン操作の失敗：Azure でのインスタンスの削除には長時間かかることがあります。このような状況では、スケールイン操作がタイムアウトし、エラーが報告されますが、最終的にはインスタンスが削除されます。
- 設定を変更する前に、Logic App を無効にし、実行中のすべてのタスクが完了するまで待ちます。

# ソースコードからの Azure 関数の構築

### システム要件

- Microsoft Windows デスクトップ/ラップトップ。
- Visual Studio（Visual Studio 2019 バージョン 16.1.3 でテスト済み）

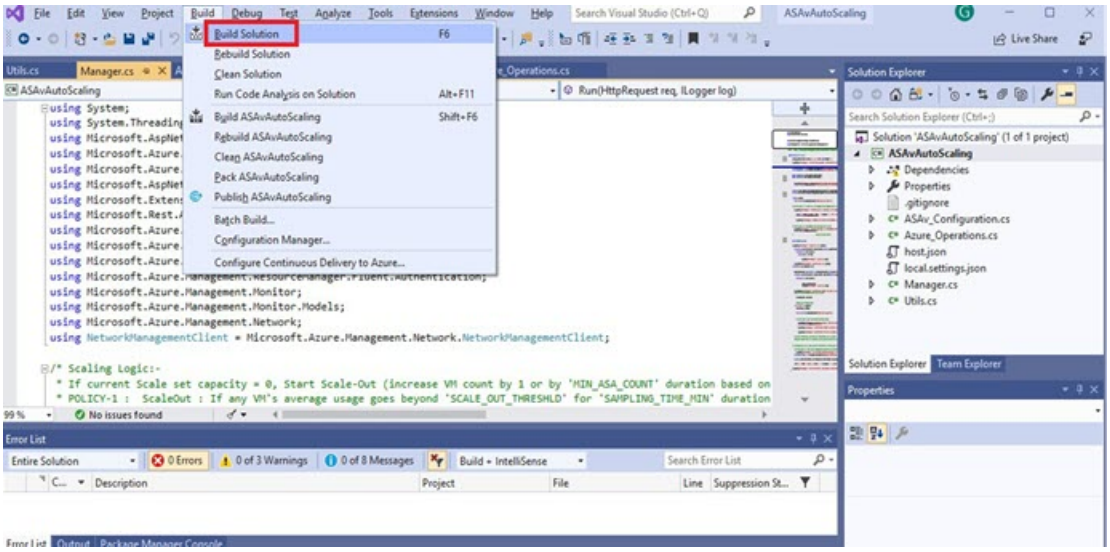
（注） Azure 関数は C# を使用して記述されます。

- 「Azure 開発」ワークロードを Visual Studio にインストールする必要があります。

### Visual Studio を使用したビルド

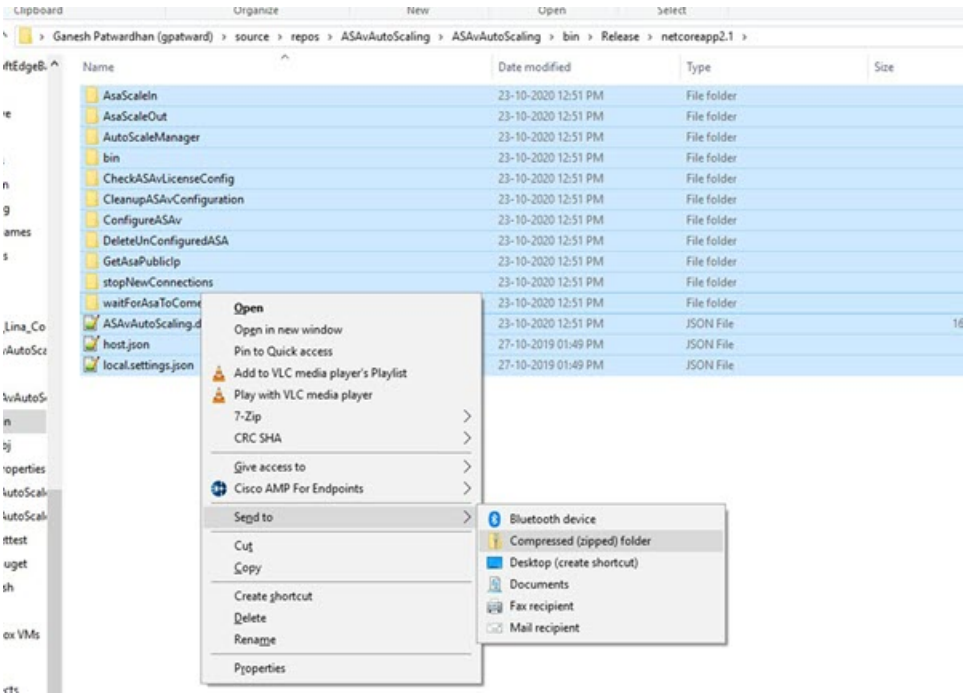
1. 「code」フォルダをローカルマシンにダウンロードします。
2. 「ASAAutoScaling」フォルダに移動します。
3. Visual Studio でプロジェクトファイル「ASAAutoScaling」を開きます。
4. クリーンアップしてビルドするには、Visual Studio の標準手順を使用します。

図 *20 : Visual Studio* ビルド




5. ビルドが正常にコンパイルされたら、**\bin\Release\netcoreapp2.1** フォルダに移動します。
6. すべての内容を選択し、**[送信先（Send to）]**>**[圧縮（ZIP）フォルダ（Compressed (zipped) folder）]** の順にクリックして、ZIP ファイルを ASM_Function.zip として保存します。

図 *21 : ASM_Function.zip* のビルド



翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。