GCP への ASA 仮想 Auto Scale ソリューションの展開

GCP 上の ASA 仮想 向けの Auto Scale ソリューション

以下のセクションでは、Auto Scale ソリューションのコンポーネントが GCP の ASA 仮想 でどのように機能するかについて説明します。

Auto Scale ソリューションについて

ASA 仮想 Auto Scale for GCP は、GCP によって提供されるサーバーレス インフラストラクチャ（クラウド機能、ロードバランサ、Pub/Sub、インスタンスグループなど）を利用した完全なサーバーレス導入です。

ASA 仮想 Auto Scale for GCP 導入の主な特徴は次のとおりです。

- GCP Deployment Manager のテンプレートをベースとした導入。
- CPU に基づくスケーリングメトリックのサポート。
- ASA 仮想 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 ASA 仮想 インスタンスに完全に自動化された構成を自動適用。

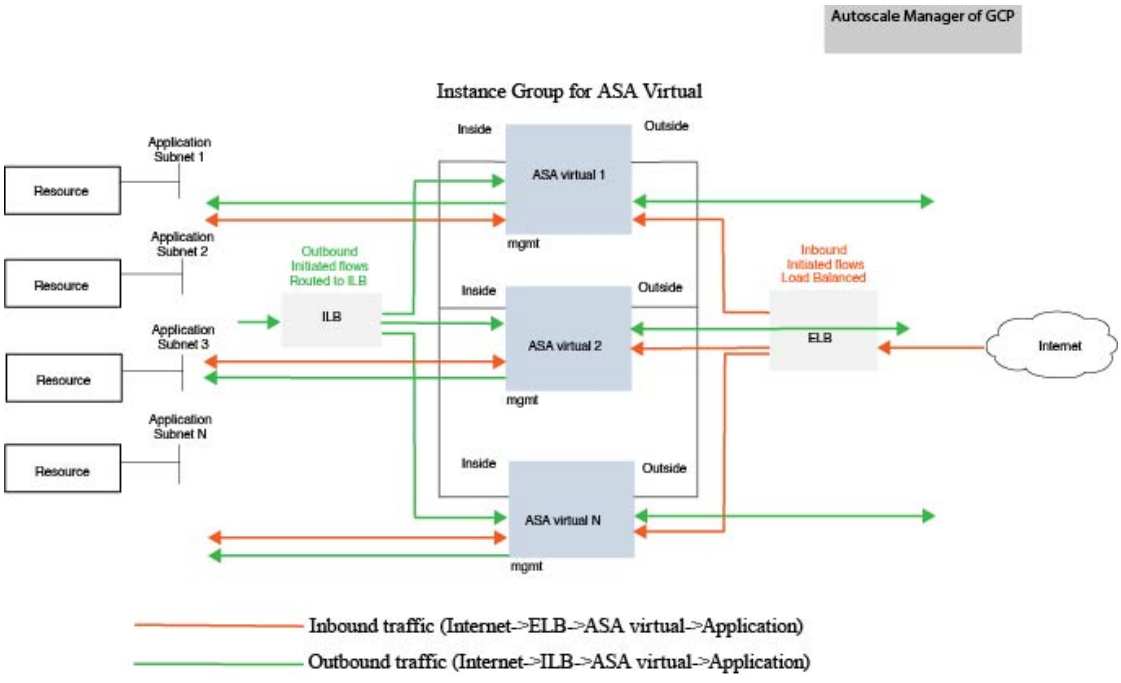
- ロードバランサとマルチ可用性ゾーンのサポート。
- シスコでは、導入を容易にするために、Auto Scale for GCP の導入パッケージを提供しています。

Auto Scale の導入例

ASA 仮想 Auto Scale for GCP は、ASA 仮想 インスタンスグループを GCP の内部ロードバランサ（ILB）と GCP の外部ロードバランサ（ELB）の間に配置する水平方向の自動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インスタンスグループ内の ASA 仮想 インスタンスに分散させます。その後、ファイアウォールからアプリケーションにトラフィックが転送されます。
- ILB は、アプリケーションからのインターネットトラフィックをインスタンスグループ内の ASA 仮想 インスタンスに分散させます。その後、ファイアウォールからインターネットにトラフィックが転送されます。
- 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 ASA 仮想 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 : ASA* 仮想 自動スケールのユースケース



スコープ

このドキュメントでは、ASA 仮想 Auto Scale for GCP ソリューション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

ASA 仮想 Auto Scale for GCP は GCP Deployment Manager のテンプレートをベースとした導入であり、GCP によって提供されるサーバーレス インフラストラクチャ（クラウド機能、ロードバランサ、Pub/Sub、インスタンスグループなど）を利用します。

ASA 仮想 Auto Scale for GCP 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 ASA 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。

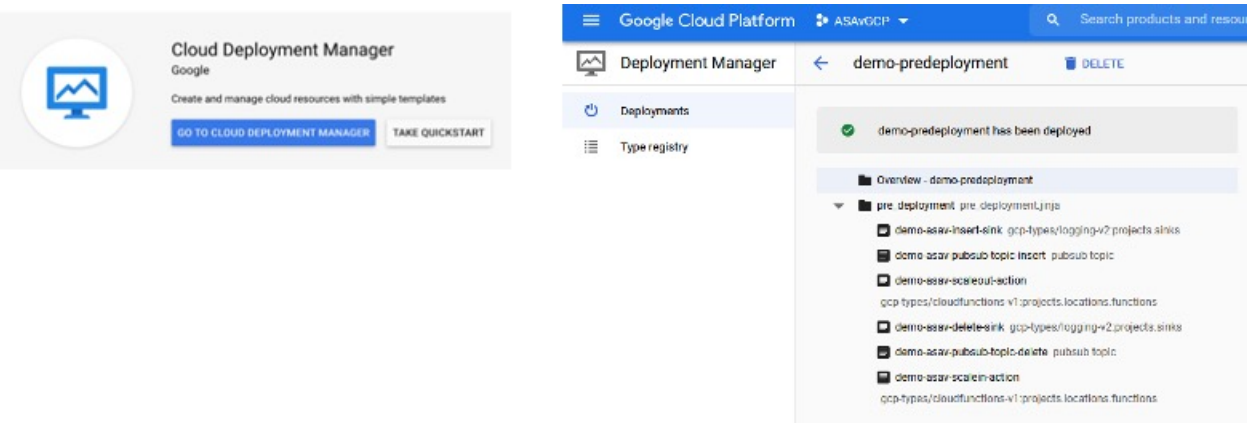
Auto Scale ソリューションのコンポーネント

ASA 仮想 Auto Scale for GCP 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。

導入マネージャ

- 構成をコードとして扱い、反復可能な展開を実行します。Google Cloud Deployment Manager では、YAML を使用して、アプリケーションに必要なすべてのリソースを宣言形式で指定できます。また、Python または Jinja2 テンプレートを使用して構成をパラメータ化し、一般的な導入パラダイムを再利用できます。
- リソースを定義する構成ファイルを作成します。リソースを作成するプロセスを繰り返し実行することで、一貫した結果を得ることができます。詳細については、https://cloud.google.com/deployment-manager/docsを参照してください。

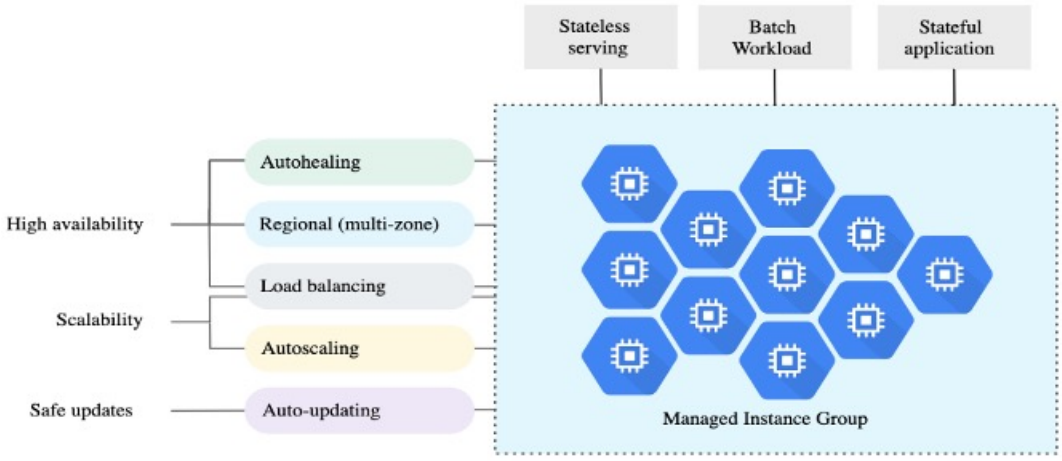
図 *2:* 導入マネージャビュー



GCP のマネージド インスタンス グループ

マネージドインスタンス グループ（MIG）は、指定したインスタンステンプレートとオプションのステートフル構成に基づいて、各マネージドインスタンスを作成します。詳細については、https://cloud.google.com/compute/docs/instance-groupsを参照してください。

図 *3:* インスタンスグループの機能

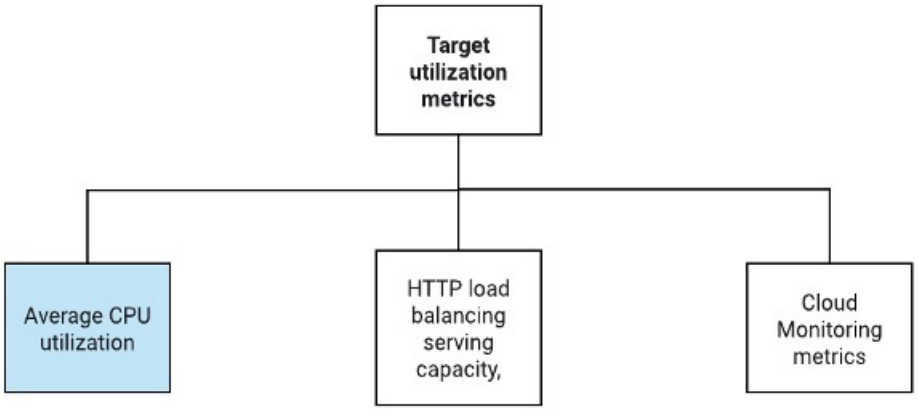


ターゲット使用率メトリック

- 次の図は、ターゲット使用率のメトリックを示しています。自動スケーリングを決定する際、平均 CPU 使用率メトリックのみが使用されます。
- オートスケーラは、選択された使用率メトリクスに基づいて使用状況の情報を継続的に収集し、実際の使用率を希望するターゲット使用率と比較します。次に、この情報を使用して、グループがインスタンスを削除する必要があるか（スケールイン）またはインスタンスを追加する必要があるか（スケールアウト）を判断します。

- ターゲット使用率レベルとは、仮想マシン（VM）インスタンスをどのレベルで維持するかを示します。たとえば、CPU使用率に基づいてスケーリングする場合、ターゲット使用率レベルを75%に設定すると、オートスケーラは指定されたインスタンス グループで75%またはそれに近いCPU使用率を維持します。各メトリックの使用率レベルは、自動スケーリング ポリシーに基づいてさまざまに解釈されます。詳細については、https://cloud.google.com/compute/docs/autoscalerを参照してください。

図 *4:* ターゲット使用率メトリック



サーバーレスクラウド機能

Instance Group Manager でインスタンスが起動したときに、サーバーレスの Google Cloud 機能を使用して、SSH パスワードの設定、パスワードの有効化、ホスト名の変更を行います。

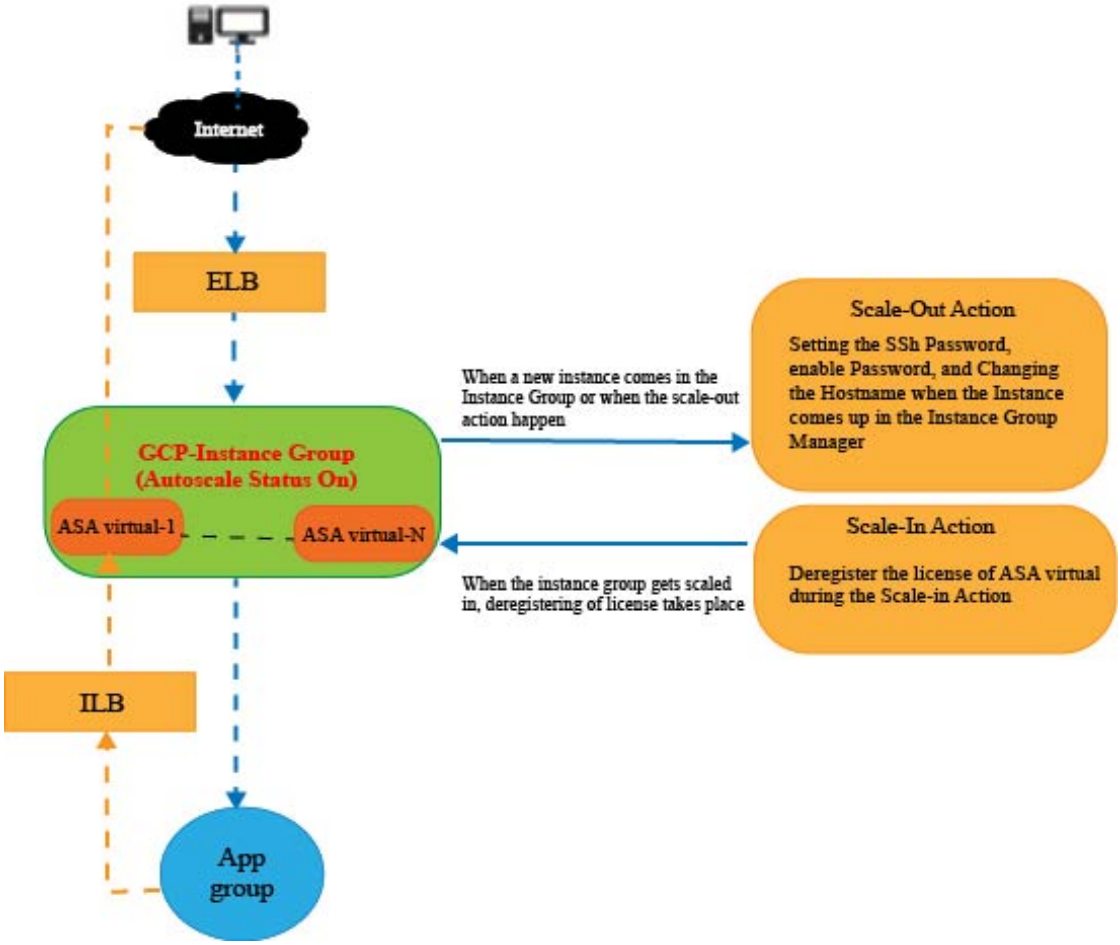
- スケールアウト中に新しい ASA 仮想 インスタンスがインスタンスグループに追加された場合、スケールアウトプロセスを常に監視できないため、SSHパスワードを設定し、パスワードを有効にして、ホスト名を変更する必要があります。
- クラウド機能は、スケールアウトプロセス中に クラウドのパブリック/サブトピックを介してトリガーされます。また、スケールアウト時のインスタンス追加専用のフィルタを備えたログシンクもあります。

クラウド機能を使用したサーバーレスのライセンス登録解除

- スケールイン時のインスタンス削除中に、ASA 仮想 インスタンスからライセンスの登録を解除する必要があります。
- クラウド機能は、クラウドのパブリック/サブトピックを介してトリガーされます。特に削除プロセスについては、スケールイン時のインスタンス削除専用のフィルタを備えたログシンクがあります。
- クラウド機能は、トリガーされると削除対象の ASA 仮想 インスタンスに SSH で接続し、ライセンス登録解除のコマンドを実行します。

Auto Scale ソリューションの大まかな概要

図 *5 : Auto Scale* ソリューションの概要



Auto Scale ソリューションの前提条件

GCP リソース

GCP プロジェクト

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プロジェクトが必要です。

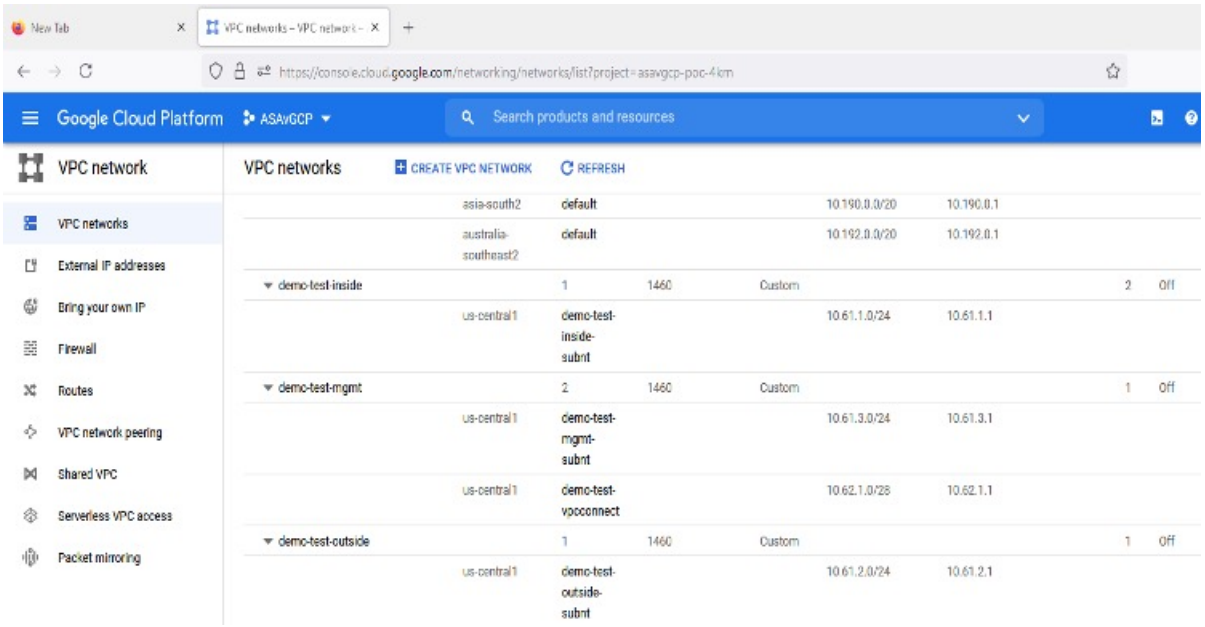
ネットワーキング

3 つの VPC が使用可能または作成されていることを確認してください。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。

ASA 仮想 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。

- 管理トラフィック
- 内部トラフィック
- 外部トラフィック

図 *6 : VPC* ネットワークビュー



Firewall

VPC 間通信を許可し、正常性プローブも許可するファイアウォールルールを作成する必要があります。Deployment Manager テンプレートで後に使用されるファイアウォールタグに注意する必要があります。

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 ASA 仮想 間の正常性プローブに必要です。サーバーレス機能と ASA 仮想 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）。

ASA 構成ファイルの準備

Deployment Manager jinja 構成ファイルに含める ASA 仮想 構成ファイルを準備します。この構成は、プロジェクト内の ASA 仮想 のインスタンステンプレートで起動スクリプトとして使用されます。

構成ファイルに最低限必要な内容は以下のとおりです。

- 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。
- GCP ロードバランサはトラフィックを Nic0 にのみ転送するため、Nic0 は「外部」としてマークする必要があります。
- Nic0 は IP 転送のみをサポートしているため、ASA 仮想 への SSH 接続に使用されます。
- ASA 設定の外部インターフェイスで SSH を有効にします。
- 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。
- リソースの正常性ステータスについては、適切な NAT ルールを使用して、リソースの正常性プローブをメタデータサーバーにリダイレクトする必要があります。

参考用に ASA 構成ファイルの例を次に示します。

```
!ASA Version 9.15.1.10
!Interface Config
interface G0/0
nameif inside
security-level 100
ip address dhcp setroute
no shutdown


interface G0/1
nameif management
security-level 50
ip address dhcp setroute
no shutdown

interface M0/0
no management-only
nameif outside
security-level 0
ip address dhcp setroute
no shutdown
!
same-security-traffic permit inter-interface
!
!Due to some constraints in GCP,
!"GigabitEthernet0/0" will be used as a Management interface
!"Management0/0" will be used as a data interface
crypto key generate rsa modulus 2048
ssh 0.0.0.0 0.0.0.0 management
ssh version 2
ssh timeout 60
```

```
aaa authentication ssh console LOCAL
ssh authentication publickey {{ properties["publicKey"] }}
username admin privilege 15
username admin attributes
service-type admin

! required config end
dns domain-lookup management
dns server-group DefaultDNS
name-server 8.8.8.8
!
access-list all extended permit ip any any
access-list out standard permit any4
access-group all global
! Objects
object network metadata
host 169.254.169.254
object network ilb
host $(ref.{{ properties["resourceNamePrefix"] }}-ilb-ip.address)
object network hc1
subnet 35.191.0.0 255.255.0.0
object network hc2
subnet 130.211.0.0 255.255.63.0
object network elb
host $(ref.{{ properties["resourceNamePrefix"] }}-elb-ip.address)
object network appServer
host 10.61.2.3
object network defaultGateway
subnet 0.0.0.0 0.0.0.0
! Nat Rules
nat (inside,outside) source dynamic hc1 ilb destination static ilb metadata
nat (inside,outside) source dynamic hc2 ilb destination static ilb metadata
nat (inside,outside) source dynamic defaultGateway interface
 !
 object network appServer
nat (inside,outside) static $(ref.{{ properties["resourceNamePrefix"] }}-elb-ip.address)
object network defaultGateway
nat (outside,inside) dynamic interface
! Route Add
route inside 0.0.0.0 0.0.0.0 10.61.1.1 2
route management 0.0.0.0 0.0.0.0 10.61.3.1 3
license smart register idtoken <licenseIDToken>
```

GCP クラウド機能パッケージの構築

ASA 仮想 GCP Auto Scale ソリューションでは、圧縮された ZIP パッケージの形式でクラウド機能を提供する 2 つのアーカイブファイルを作成する必要があります。

- `scalein-action.zip`
- `scaleout-action.zip`

`scalein-action.zip` および `scaleout-action.zip` パッケージの作成方法については、Auto Scale の導入手順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、GCP プロジェクトに GCP Deployment Manager を展開するときに、各パラメータを使用して ASA 仮想 デバイスを作成できます。

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	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。 例：demo-test	新規作成（New）
region	GCP でサポートされている有効なリージョン [String]	プロジェクトが展開されるリージョン名。 例：us-central1	
serviceAccountMailId	文字列 [Email Id]	サービスアカウントを識別するメールアドレス。	
vpcConnectorName	文字列	サーバーレス環境と VPC ネットワーク間のトラフィックを処理するコネクタの名前。 例：demo-test-vpc-connector	
bucketName	文字列	クラウド機能の ZIP パッケージをアップロードする GCP ストレージバケットの名前。 例：demo-test-bkt	
cpuUtilizationTarget	10 進数 (0,1]	オートスケーラーが維持する必要があるインスタンスグループ内の VM の平均 CPU 使用率。 例：0.5	

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
healthCheckFirewallRuleName	文字列	ヘルスチェックプローブの IP 範囲からのパケットを許可するファイアウォールルールのタグ。 例： demo-test-healthallowall	既存
insideFirewallRuleName	文字列	内部 VPC での通信を許可するファイアウォールルールのタグ。 例： demo-test-inside-allowall	既存
insideVPCName	文字列	内部 VPC の名前。 例：demo-test-inside	既存
insideVPCSubnet	文字列	内部サブネットの名前。 例： demo-test-inside-subnt	既存
マシン タイプ	文字列	ASA 仮想 VM のマシンタイプ。 例：e2-standard-4	
maxASACount	整数	インスタンスグループで許可される ASA 仮想インスタンスの最大数。 例：3	
mgmtFirewallRuleName	文字列	管理 VPC での通信を許可するファイアウォールルールのタグ。 例： demo-test-mgmt-allowall	
mgmtVPCName	文字列	管理 VPC の名前。 例：demo-test-mgmt	

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
mgmtVPCSubnet	文字列	管理サブネットの名前。 例： demo-test-mgmt-subnt	
minASACount	整数	任意の時点でインスタンスグループで使用可能な ASA 仮想 インスタンスの最小数。 例 1	
outsideFirewallRuleName	文字列	外部 VPC での通信を許可するファイアウォールルールのタグ。 例： demo-test-outside-allowall	
outsideVPCName	文字列	外部 VPC の名前。 例：demo-test-outside	
outsideVPCSubnet	文字列	外部サブネットの名前。 例： demo-test-outside-subnt	
publicKey	文字列	ASA 仮想 VM の SSH キー。	
sourceImageURL	文字列	プロジェクトで使用する ASA 仮想 のイメージ。 例： https://www.googleapis.com/compute/v1/projects/cisco-public/global/images/cisco-asav-9-15-1-15	
アプリケーションサーバーの IP アドレス	文字列	内部 Linux マシンの内部 IP アドレス。 例：10.61.1.2	

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
内部 VPC ゲートウェイの IP アドレス	文字列	内部 VPC のゲートウェイ。 例：10.61.1.1	
管理 VPC ゲートウェイの IP アドレス	文字列	管理 VPC のゲートウェイ。 例：10.61.3.1	

Auto Scale ソリューションの展開

ステップ 1 Git リポジトリをローカルフォルダに複製します。

```
git clone git_url -b branch_name
```

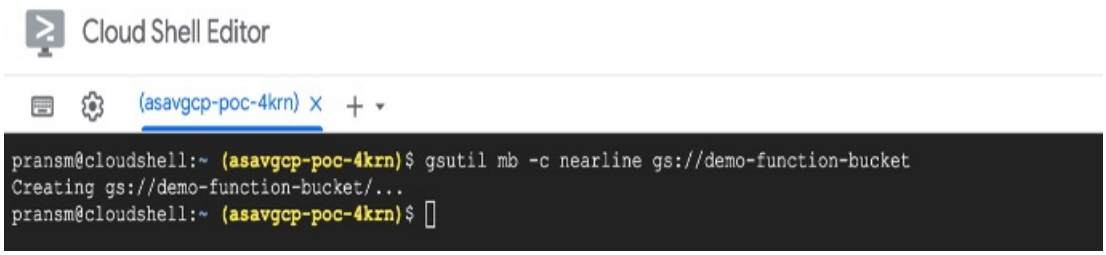
例：

```
Last login: Thu Jun  3 13:01:32 on ttys002
(base) pransm@PRANSM-M-F9KA ~ % git clone https://bitbucket-eng-bgl1.cisco.com/bitbucket/scm/vcb/cloud_autoscale.git -b saaarwar_asa_autoscale_public_key
Cloning into 'cloud_autoscale'...
remote: Enumerating objects: 1604, done.
remote: Counting objects: 100% (1604/1604), done.
remote: Compressing objects: 100% (1507/1507), done.
remote: Total 1604 (delta 759), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (1604/1604), 58.35 MiB | 8.54 MiB/s, done.
Resolving deltas: 100% (759/759), done.
(base) pransm@PRANSM-M-F9KA ~ %
```

ステップ 2 gcloud CLI でバケットを作成します。

```
gsutil mb -c nearline gs://bucket_name
```

例：



ステップ 3 ZIP 形式の圧縮パッケージを作成します。

a) `scalein_action` および `scaleout_action` フォルダから、以下のファイルで構成される Zip 形式の圧縮パッケージを作成します。

- main.py
- basic_functions.py

requirements.txt

b) Zip 形式の圧縮パッケージの名前を `scaleout-action.zip` および `scalein-action.zip` に変更します。

（注） フォルダー内を移動し、ファイルを選択して右クリックし、「圧縮 | 」を選択します。archive' を使用して、GCP が読み取れる .zip を作成します。

ステップ 4 Zip 形式の圧縮パッケージ（`scaleout-action.zip` および `scalein-action.zip`）を Cloud Editor ワークスペースにアップロードします。

ステップ 5 以下のファイルを Deployment Manager テンプレートから Cloud Editor ワークスペースにアップロードします。

- asav_autoscale.jinja
- asav_autoscale_params.yaml
- pre_deployment.jinja
- pre_deployment.yaml

ステップ 6 ZIP 形式の圧縮パッケージをバケットストレージにコピーします。

- `gsutil cp scaleout-action.zip gs://bucket_name`
- `gsutil cp scalein-action.zip gs://bucket_name`

例：

```
pransm@cloudshell:~ (asavgcp-poc-4krn)$ gsutil cp scaleout-action.zip gs://demo-function-bucket
Copying file://scaleout-action.zip [Content-Type=application/zip]...
/ [1 files][  3.3 KiB/  3.3 KiB]
Operation completed over 1 objects/3.3 KiB.
pransm@cloudshell:~ (asavgcp-poc-4krn)$ gsutil cp scalein-action.zip gs://demo-function-bucket
Copying file://scalein-action.zip [Content-Type=application/zip]...
/ [1 files][  3.3 KiB/  3.3 KiB]
Operation completed over 1 objects/3.3 KiB.
pransm@cloudshell:~ (asavgcp-poc-4krn)$
```

ステップ 7 内部、外部、および管理インターフェイス用の VPC とサブネットを作成します。

管理 VPC では、/28 サブネット（10.8.2.0/28 など）が必要です。

ステップ 8 内部、外部、および管理インターフェイス用に 3 つのファイアウォールルールが必要です。また、ファイアウォールルールではヘルスチェックプローブを許可する必要があります。

ステップ 9 事前展開および ASA 仮想 Auto Scale の Jinja ファイルと YAML ファイルのパラメータを更新します。

a) `asav_autoscale_params.yaml` ファイルを開き、以下のパラメータを更新します。

- **resourceNamePrefix**: <resourceNamePrefix>
- **region**: <region>
- **serviceAccountMailId**: <serviceAccountMailId>
- **publicKey**: <publicKey>
- **insideVPCName**: <Inside-VPC-Name>

- **insideVPCSubnet**: <Inside-VPC-Subnet>
- **outsideVPCName**: <Outside-VPC-Name>
- **outsideVPCSubnet**: <Outside-VPC-Subnet>
- **mgmtVPCName**: <Mgmt-VPC-Name>
- **mgmtVPCSubnet**: <Mgmt-VPC-Subnet>
- **insideFirewallRuleName**: <Inside-Network-Firewall-Tag>
- **outsideFirewallRuleName**: <Outside-Network-Firewall-Tag>
- **mgmtFirewallRuleName**: <Mgmt-Network-Firewall-Tag>
- healthCheckFirewallRuleName :<HealthCheck-IP-Firewall-Tag>
- **machineType**: <machineType>

（注） ASA 仮想 Auto Scale の場合、**cpuUtilizationTarget: 0.5** パラメータが設定されており、必要に応じて編集できます。

この値は、すべての ASA 仮想 インスタンスグループの CPU 使用率が 50% であることを示します。

b) `asav_autoscale.jinja` ファイルを開き、以下のパラメータを更新します。

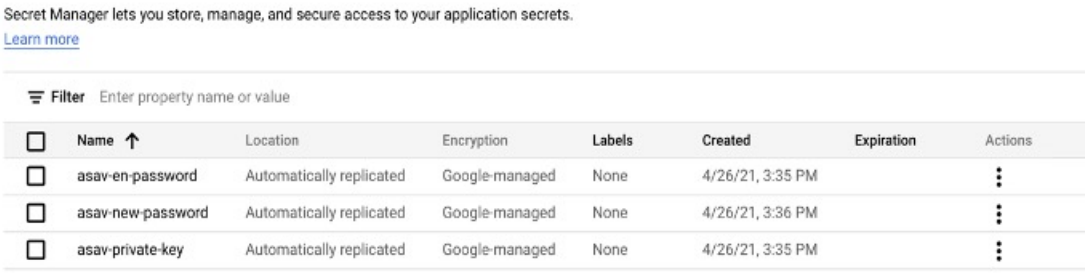
- **host**: <Application server IP address>
- **route inside 0.0.0.0 0.0.0.0**: <Inside VPC Gateway IP address> 2
- **route management 0.0.0.0 0.0.0.0**: <Management VPC Gateway IP address> 3
- **license smart register idtoken**: <licenseIDToken>

c) `pre_deployment.yaml` ファイルを開き、以下のパラメータを更新します。

- **resourceNamePrefix**: <resourceNamePrefix>
- **region**: <region>
- **serviceAccountMailId**: <serviceAccountMailId>
- **vpcConnectorName**: <VPC-Connector-Name>
- **bucketName**: <bucketName>

ステップ 10 Secret Manager GUI を使用して、次の 3 つのシークレットを作成します。「https://console.cloud.google.com/security/secret-manager」を参照してください。

- asav-en-password
- asav-new-password
- asav-private-key


Secret Manager lets you store, manage, and secure access to your application secrets.
Learn more

Filter Enter property name or value

Name	Location	Encryption	Labels	Created	Expiration	Actions
asav-en-password	Automatically replicated	Google-managed	None	4/26/21, 3:35 PM		
asav-new-password	Automatically replicated	Google-managed	None	4/26/21, 3:36 PM		
asav-private-key	Automatically replicated	Google-managed	None	4/26/21, 3:35 PM		

ステップ 11 VPC コネクタを作成します。

```
gcloud beta compute networks vpc-access connectors create <vpc-connector-name>
--region <region> --subnet=</28 subnet name>
```

例：

```
gcloud beta compute networks vpc-access connectors create demo-vpc-connector
--region us-central1 --subnet=outside-connect-28
Create request issued for: [demo-vpc-connector]
Waiting for operation [projects/asavgcp-poc-4krn/locations/us-central1/operations/
10595de7-837f-4c19-9396-0c22943ecf15] to complete...done.
Created connector [demo-vpc-connector].
```

ステップ 12 事前展開の YAML 構成を展開します。

```
gcloud deployment-manager deployments create <pre-deployment-name>
--config pre_deployment.yaml
```

例：

```
gcloud deployment-manager deployments create demo-predeployment
--config pre_deployment.yaml

The fingerprint of the deployment is b'9NOy0gsTPgg16SqUEVsBjA=='
Waiting for create [operation-1624383045917-5c55e266e596d-4979c5b6-66d1025c]...done.
Create operation operation-1624383045917-5c55e266e596d-4979c5b6-66d1025c
completed successfully
```

NAME	TYPE	STATE
demo-asav-delete-sink	gcp-types/logging-v2:projects.sinks	COMPLETED
demo-asav-insert-sink	gcp-types/logging-v2:projects.sinks	COMPLETED
demo-asav-pubsub-topic-delete	pubsub.v1.topic	COMPLETED
demo-asav-pubsub-topic-insert	pubsub.v1.topic	COMPLETED
demo-asav-scalein-action	gcp-types/cloudfunctions-v1:projects.locations.functions	COMPLETED
demo-asav-scaleout-action	gcp-types/cloudfunctions-v1:projects.locations.functions	COMPLETED

ステップ 13 ASA 仮想 Auto Scale の展開を作成します。

```
gcloud deployment-manager deployments create <deployment-name>
--config asav_autoscale_params.yaml
```

例：

```
gcloud deployment-manager deployments create demo-asav-autoscale
--config asav_autoscale_params.yaml
The fingerprint of the deployment is b'1JCQi7Il-laWOY7vOLza0g=='
Waiting for create [operation-1624383774235-5c55e51d79d01-1a3acf92-4f3daf16]...done.
Create operation operation-1624383774235-5c55e51d79d01-1a3acf92-4f3daf16
completed successfully.
```

NAME	TYPE	STATE
demo-asav-autoscaler	compute.v1.regionAutoscaler	COMPLETED
demo-asav-backend-service-elb	compute.v1.regionBackendService	COMPLETED
demo-asav-backend-service-ilb	compute.v1.regionBackendService	COMPLETED
demo-asav-fr-elb	compute.v1.forwardingRule	COMPLETED
demo-asav-fr-ilb	compute.v1.forwardingRule	COMPLETED
demo-asav-hc-elb	compute.v1.regionHealthChecks	COMPLETED
demo-asav-hc-ilb	compute.v1.healthCheck	COMPLETED
demo-asav-health-check	compute.v1.healthCheck	COMPLETED
demo-asav-instance-group	compute.v1.regionInstanceGroupManager	COMPLETED
demo-asav-instance-template	compute.v1.instanceTemplate	COMPLETED
demo-elb-ip	compute.v1.address	COMPLETED

ステップ 14 内部アプリケーションからインターネットにパケットを転送する ILB のルートを作成します。

```
gcloud beta compute routes create <ilb-route-name>
--network=<inside-vpc-name> --priority=1000 --destination-range=0.0.0.0/0
--next-hop-ilb=<ilb-forwarding-rule-name> --next-hop-ilb-region=<region>
```

例：

```
gcloud beta compute routes create demo-ilb --network=sdt-test-asav-inside
--priority=1000 --destination-range=0.0.0.0/0 --next-hop-ilb=demo-asav-fr-ilb
--next-hop-ilb-region=us-central1
Created [https://www.googleapis.com/compute/beta/projects/asavgcp-poc-4krn/global
/routes/demo-ilb].
```

NAME	NETWORK	DEST_RANGE	NEXT_HOP	PRIORITY
demo-ilb	sdt-test-asav-inside	0.0.0.0/0	10.7.1.60	1000

ステップ 15 Cloud Router と Cloud NAT を作成します。

```
gcloud compute routers create <cloud-router-name>
--project=<project-name> --region <region> --network=<outside-vpc-name>
--advertisement-mode=custom
```

```
gcloud compute routers nats create <cloud-nat-name>
--router=<cloud-router-name> --nat-all-subnet-ip-ranges --auto-allocate-nat-external-ips
--region=<region>
```

例：

```
gcloud compute routers create demo-cloud-router --project=asavgcp-poc-4krn
--region us-central1 --network=sdt-test-asav-outside --advertisement-mode=custom
Creating router [demo-cloud-router]...done.
```

NAME	REGION	NETWORK
demo-cloud-router	us-central1	sdt-test-asav-outside

```
gcloud compute routers nats create demo-cloud-nat
--router=demo-cloud-router --nat-all-subnet-ip-ranges
--auto-allocate nat-external-ips --region=us-central1
Creating NAT [demo-cloud-nat] in router [demo-cloud-router]...done.
```

Auto Scale ロジック

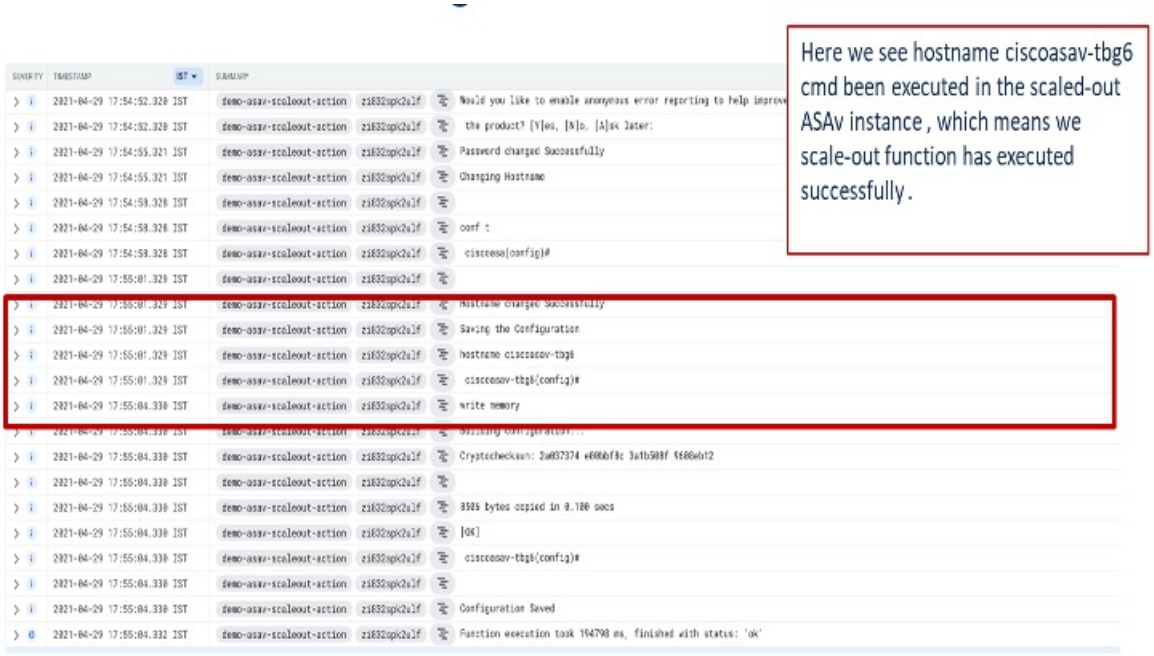
- オートスケーラは、ターゲット CPU 使用率レベルを、インスタンスグループ内の一定期間にわたるすべての vCPU の平均使用量の一部として扱います。
- 合計 vCPU の平均使用率がターゲット使用率を超えると、オートスケーラによって VM インスタンスが追加されます。合計 vCPU の平均使用率がターゲット使用率よりも低い場合、オートスケーラはインスタンスを削除します。
- たとえば、0.75 のターゲット使用率を設定すると、オートスケーラはインスタンスグループ内のすべての vCPU の平均使用率を 75% に維持するように指示されます。
- スケーリングの決定では、CPU 使用率メトリックのみが使用されます。
- このロジックは、ロードバランサが、すべての ASA に接続を均等に分散しようとし、平均してすべての ASA 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

表示できるクラウド機能のログは以下のとおりです。

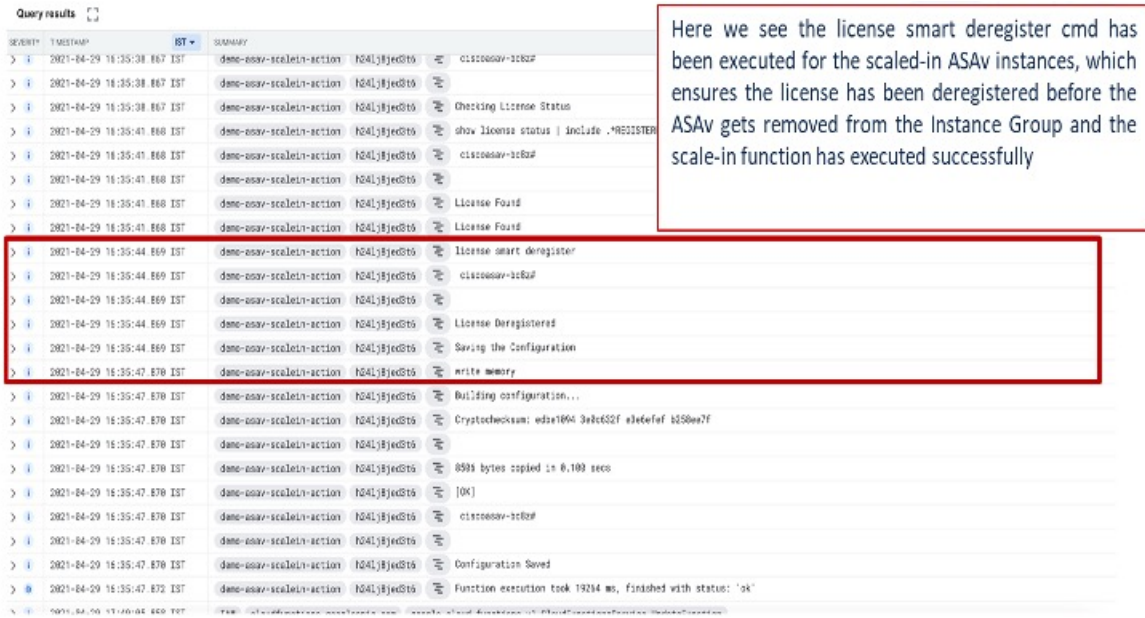
- スケールアウト機能のログ

図 7: スケールアウト機能のログ



- スケールイン機能のログ

図 *8 :* スケールイン機能のログ



Auto Scale のガイドラインと制約事項

- IPv4 だけがサポートされます。
- サポートされているライセンスは BYOL のみです。PAYG は GCP 上の ASA 仮想 では利用できません。
- 外部ロードバランサはテンプレートによって作成されるため、ロードバランサのパブリック IP に関する特定の DNS 要件は範囲外です。
- アプリケーションはユーザーが作成したロードバランサの背後にあると想定され、ASA 仮想 は（トラフィックを特定のアプリケーション IP に直接送信する代わりに）すべてのトラフィックをこのロードバランサにルーティングします。
- TAG、冗長性、およびロードバランサアフィニティ構成の必要性に関する詳細は考慮されていません。
- ASA 仮想 ログイン情報は次の状況で表示されます。
  - サーバーレスコードのクリアテキスト。
  - インスタンスグループ内のすべてのインスタンス。
  - インスタンステンプレート（共有 GCP アカウントを使用している場合）。

  このような機密データは、GCP の公開キーサービスを使用して保護できます。

重要 シスコでは、ライセンスサーバーへの ASA 仮想 の登録を定期的に追跡して、スケールアウトされた ASA が期待どおりにライセンスサーバーに登録されているか、スケールインされた ASA 仮想 インスタンスがライセンスサーバーから削除されているか確認することを推奨しています。

Auto Scale のトラブルシューティング

次に、ASA 仮想 Auto Scale for GCP 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。

- `main.py` が見つからない：Zip パッケージがファイルからのみ構成されていることを確認します。クラウド機能に移動してファイルツリーを確認できます。フォルダがあってはいけません。
- テンプレートの導入中のエラー：「<>」内のすべてのパラメータ値が .jinja と .yaml で入力されていること、および同じ導入名がすでに存在することを確認します。
- Google 関数が ASA 仮想 に到達できない：VPC コネクタが作成されており、YAML パラメータファイルで同じ名前が指定されていることを確認します。
- ASA 仮想 に SSH 接続中に認証に失敗：公開キーと秘密キーのペアが正しいことを確認します。
- ライセンスの登録に失敗：ライセンス ID トークンが正しいことを確認します。また、Cloud NAT が作成されており、ASA 仮想 が tools.cisco.com にアクセスできることを確認します。

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。