Deploy the ASAv Auto Scale Solution on GCP

Auto Scale Solution for ASAv on GCP

The following sections describe how the components of the auto scale solution work for the ASAv on GCP.

About the Auto Scale Solution

ASAv auto scale for GCP is a complete serverless implementation that makes use of serverless infrastructure provided by GCP (Cloud Functions, Load Balancers, Pub/Sub, Instance Groups, etc.).

Some of the key features of the ASAv auto scale for GCP implementation include:

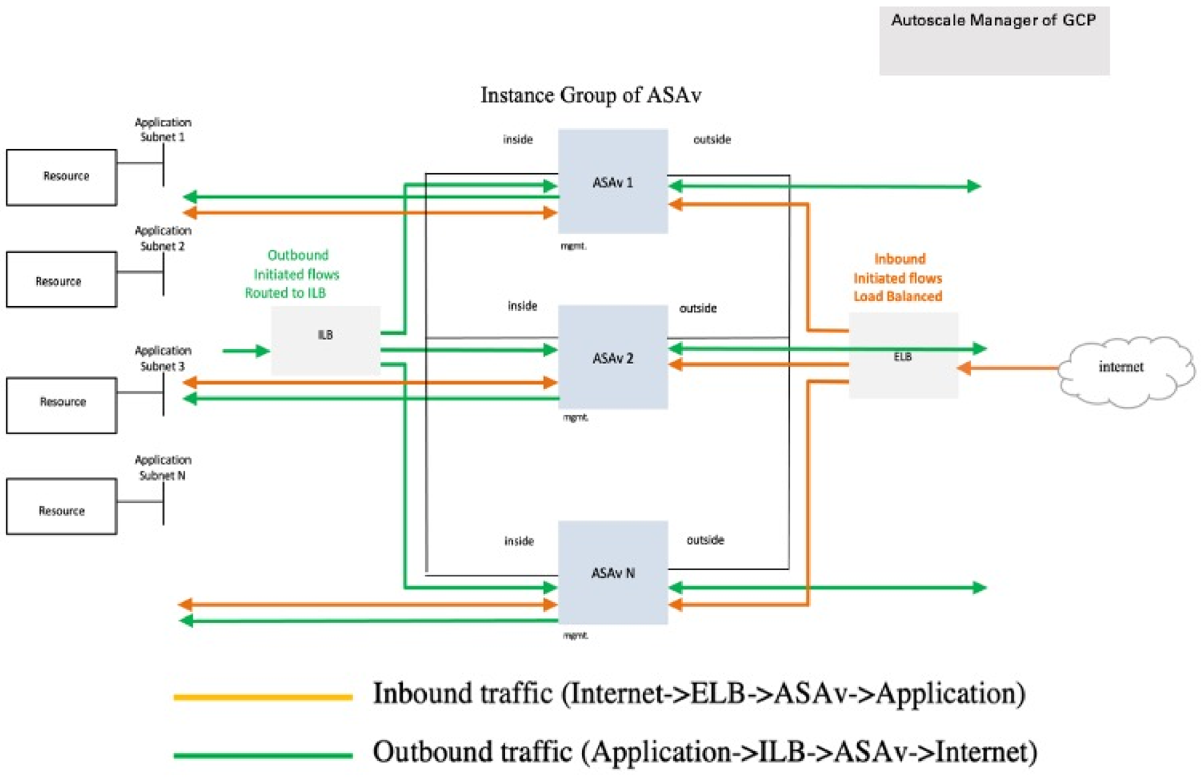
- GCP Deployment Manager template-based deployment.
- Support for scaling metrics based on CPU.
- Support for ASAv deployment and multi-availability zones.
- Completely automated configuration automatically applied to scaled-out ASAv instances.
- Support for Load Balancers and multi-availability zones.
- Cisco provides an auto scale for GCP deployment package to facilitate the deployment.

Auto Scale Use Case

The ASAv auto scale for GCP is an automated horizontal scaling solution that positions an ASAv instance group sandwiched between a GCP Internal load balancer (ILB) and a GCP External load balancer (ELB).

- The ELB distributes traffic from the Internet to ASAv instances in the instance group; the firewall then forwards traffic to the application.
- The ILB distributes outbound Internet traffic from an application to ASAv instances in the instance group; the firewall then forwards traffic to the Internet.
- A network packet will never pass through both (internal & external) load balancers in a single connection.
- The number of ASAv instances in the scale set will be scaled and configured automatically based on load conditions.

Figure 1: ASAv Auto Scale Use Case



Scope

This document covers the detailed procedures to deploy the serverless components for the ASAv Auto Scale for GCP solution.

Important

- Read the entire document before you begin your deployment.
- Make sure the prerequisites are met before you start deployment.
- Make sure you follow the steps and order of execution as described herein.

Download the Deployment Package

The ASAv Auto Scale for GCP solution is a GCP Deployment Manager template-based deployment that makes use of the serverless infrastructure provided by GCP (Cloud Functions, Load Balancers, Pub/Sub, Instance Groups, etc.).

From the download location specified by your beta manager, download the files required to launch the ASAv auto scale for GCP solution. Deployment scripts and templates for your ASA version are available at the specified download location.

Attention Note that Cisco-provided deployment scripts and templates for auto scale are provided as open source examples, and are not covered within the regular Cisco TAC support scope.

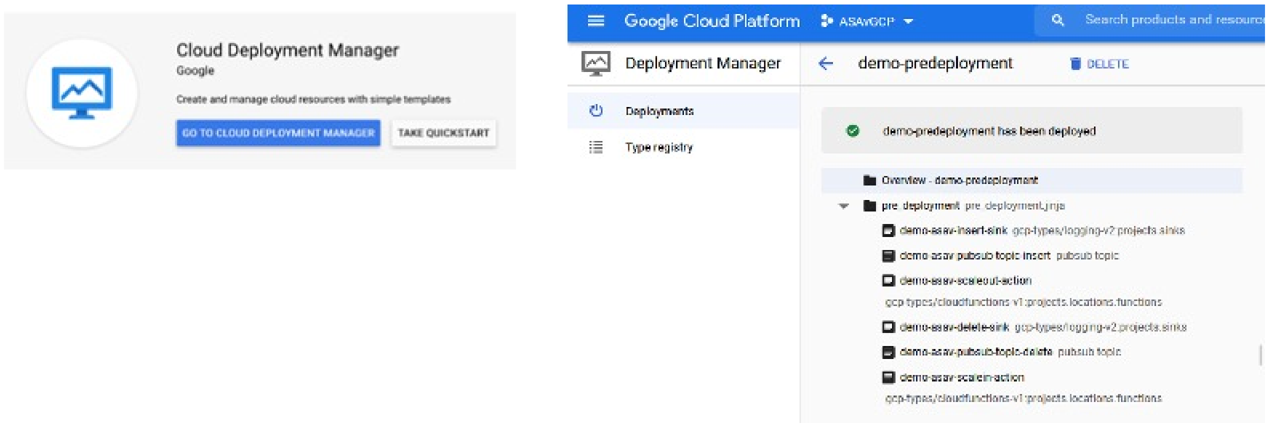
Auto Scale Solution Components

The following components make up the ASAv auto scale for GCP solution.

Deployment Manager

- Treat your configuration as code and perform repeatable deployments. Google Cloud Deployment Manager allows you to specify all the resources needed for your application in a declarative format using YAML. You can also use Python or Jinja2 templates to parameterize the configuration and allow the reuse of common deployment paradigms.
- Create configuration files that define the resources. The process of creating those resources can be repeated over and over with consistent results. See https://cloud.google.com/deployment-manager/docs for more information.

Figure 2: Deployment Manager View



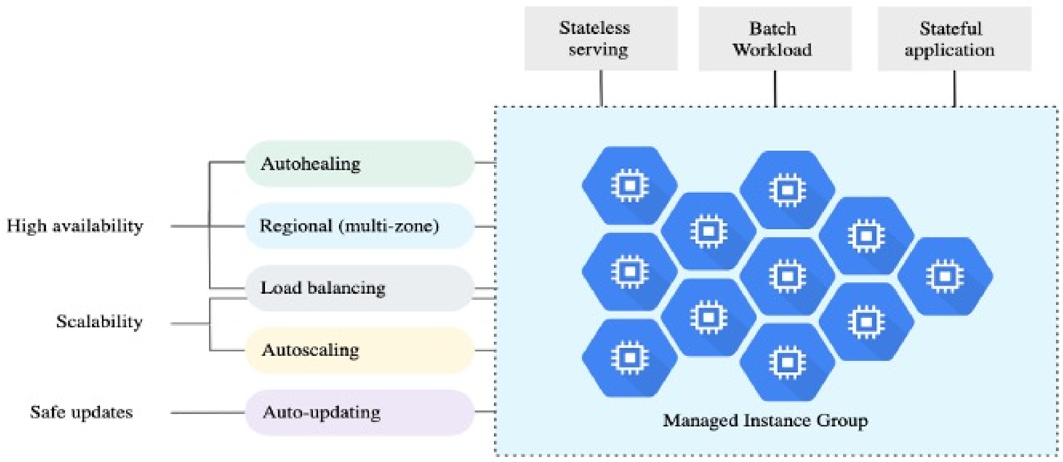
demo-predeployment has been deployed

Overview - demo-predeployment
pre_deployment pre_deployment.jinja
demo-asav-insert-sink gcp-types/logging-v2:projects.sinks
demo-asav-pubsub-topic-insert pubsub.topic
demo-asav-scaleout-action gcp-types/cloudfunctions-v1:projects.locations.functions
demo-asav-delete-sink gcp-types/logging-v2:projects.sinks
demo-asav-pubsub-topic-delete pubsub.topic
demo-asav-scalein-action gcp-types/cloudfunctions-v1:projects.locations.functions

Managed Instance Group in GCP

A Managed Instance Group (MIG) creates each of its managed instances based on the instance template and optional stateful configuration that you specify. See https://cloud.google.com/compute/docs/instance-groups for more information.

Figure 3: Instance Group Features

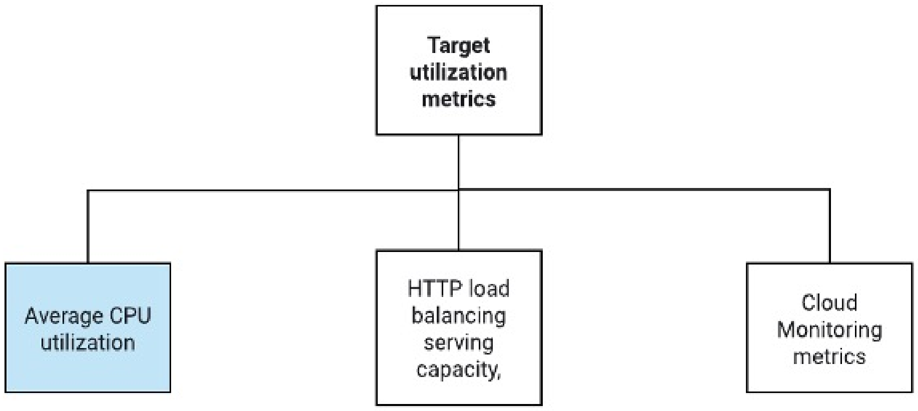


Target Utilization Metrics

- The following diagram alongside shows the target utilization metrics. Only average CPU utilization metrics are used in making autoscaling decisions.
- The autoscaler continuously collects usage information based on the selected utilization metric, compares actual utilization to your desired target utilization, and uses this information to determine whether the group needs to remove instances (Scale In) or add instances (Scale Out).
- The target utilization level is the level at which you want to maintain your virtual machine (VM) instances. For example, if you scale based on CPU utilization, you can set your target utilization level at 75% and

the autoscaler will maintain the CPU utilization of the specified group of instances at or close to 75%. The utilization level for each metric is interpreted differently based on the autoscaling policy. See https://cloud.google.com/compute/docs/autoscaler for more information.

Figure 4: Target Utilization Metrics


Target utilization metrics
Average CPU utilization
HTTP load balancing serving capacity,
Cloud Monitoring metrics

Serverless Cloud Functions

You use serverless Google Cloud functions for setting the SSH Password, enable Password, and Changing the Hostname when the instance comes up in the Instance Group Manager.

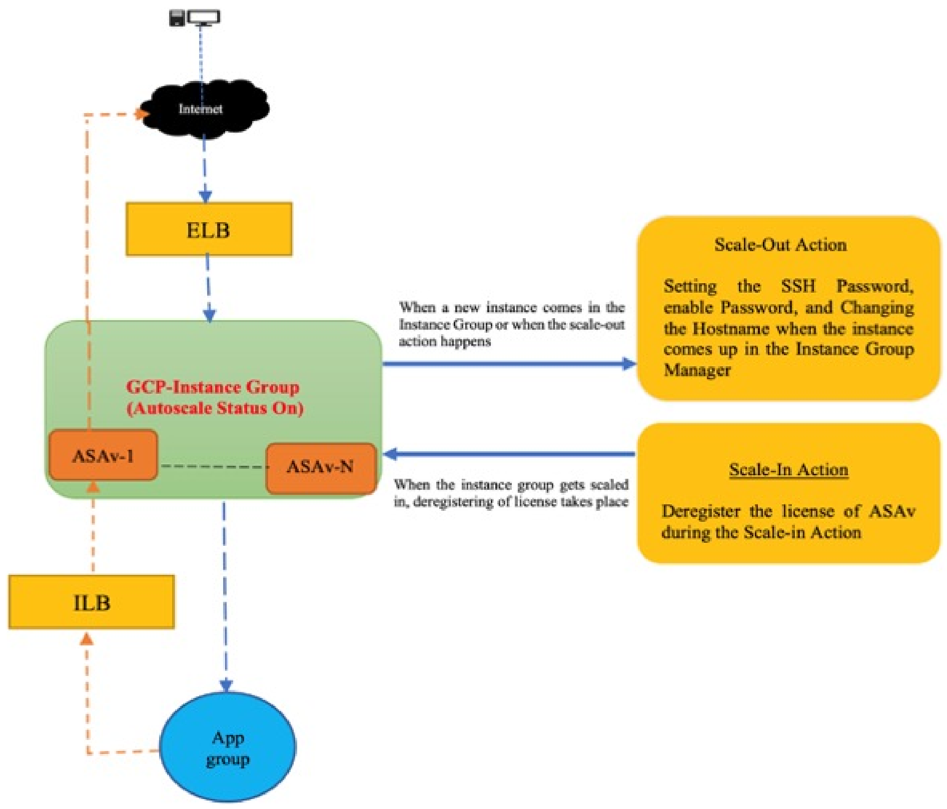
- When a new ASAv instance comes up in the instance group during Scale Out, you need to set the SSH Password, enable Password, and change the Hostname because you cannot always monitor the Scale Out process.
- Cloud functions are triggered through a Cloud Pub/Sub Topic during the Scale Out process. You also have a Log Sink with a filter that is exclusive to the addition of instances while Scale Out.

Serverless License Deregistering using Cloud Functions

- While the instances are getting deleted during Scale In, you need to deregister the license from the ASAv instance.
- Cloud functions are triggered through a Cloud Pub/Sub Topic. Particularly for the deletion process, you have a Log Sink with a filter that is exclusive to the deletion of instances while Scale In.
- Cloud Function, when triggered, will SSH into the deleting ASAv instance and run the command for license deregistration.

High-Level Overview of Autoscale Solution

Figure 5: Autoscale Solution Overview


Internet
ELB
GCP-Instance Group (Autoscale Status On)
ASAv-1
ASAv-N
ILB
App group
When a new instance comes in the Instance Group or when the scale-out action happens
When the instance group gets scaled in, deregistering of license takes place
Scale-Out Action
Setting the SSH Password, enable Password, and Changing the Hostname when the instance comes up in the Instance Group Manager
Scale-In Action
Deregister the license of ASAv during the Scale-in Action

Auto Scale Solution Prerequisites

GCP Resources

GCP Project

An existing or newly created project is required to deploy all the components of this solution.

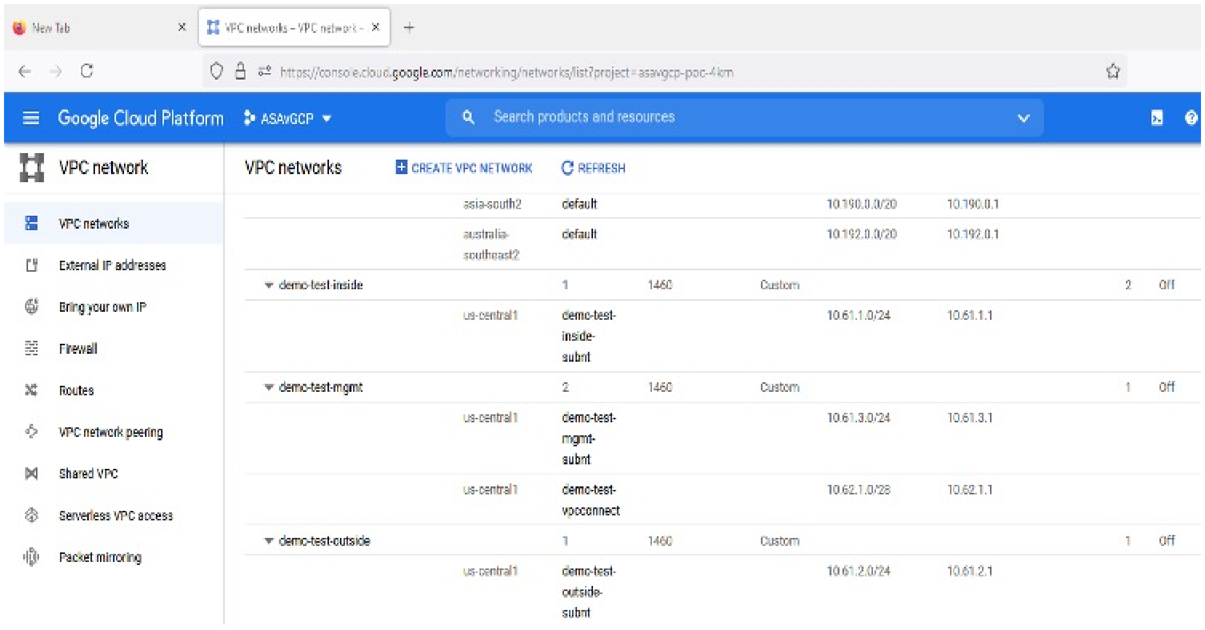
Networking

Make sure three VPCs are available/created. An auto scale deployment will not create, alter, or manage any networking resources.

The ASAv requires 3 network interfaces, thus your virtual network requires 3 subnets for:

- Management traffic
- Inside traffic
- Outside traffic

Figure 6: VPC Network View



VPC networks

	asia-south2	default			10.190.0.0/20	10.190.0.1		
	australia-southeast2	default			10.192.0.0/20	10.192.0.1		
demo-test-inside		1	1460	Custom			2	Off
	us-central1	demo-test-inside-subnt			10.61.1.0/24	10.61.1.1		
demo-test-mgmt		2	1460	Custom			1	Off
	us-central1	demo-test-mgmt-subnt			10.61.3.0/24	10.61.3.1		
	us-central1	demo-test-vpcconnect			10.62.1.0/28	10.62.1.1		
demo-test-outside		1	1460	Custom			1	Off
	us-central1	demo-test-outside-subnt			10.61.2.0/24	10.61.2.1		

Firewall

Firewall rules that allow inter VPC communication and also allow health probes are required to be created. You must note the firewall tags which are used later in the deployment manager template.

The following ports should be open in the Network Security Group to which the subnets are connected:

- SSH(TCP/22) — Required for the health probe between the Load Balancer and ASAv. Required for communication between the serverless functions and ASAv.
- Application-specific protocol/ports — Required for any user applications (for example, TCP/80, etc.).

Prepare the ASA Configuration File

Prepare an ASAv configuration file which will be put into the deployment manager jinja configuration file. This configuration will be used as a startup script in the instance template for ASAv in the project.

The configuration file should have the following (at a minimum):

- Set DHCP IP assignment to all the interfaces.
- Nic0 should be marked as 'outside' because GCP Load Balancer forwards traffic only to nic0.
- Nic0 will be used to SSH to ASAv as it only supports IP forwarding.
- Enable SSH on the outside interface in ASA configuration.
- Create NAT configuration to forward traffic from outside to inside interface.
- Create Access policy to allow desired traffic.
- For the health status of resources, their health probes should be redirected to the metadata server using proper NAT rules.

The following is a sample ASA configuration file for reference only.

```
!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>
```

Build the GCP Cloud Function Package

The ASAv GCP auto scale solution requires that you build two archive files that deliver the cloud functions in the form of a compressed ZIP package.

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

See the auto scale deployment instructions for information on how to build the `scalein-action.zip` and `scaleout-action.zip` packages.

These functions are as discrete as possible to carry out specific tasks and can be upgraded as needed for enhancements and new release support.

Input Parameters

The following table defines the template parameters and provides an example. Once you decide on these values, you can use these parameters to create the ASAv device when you deploy the GCP Deployment Manager template into your GCP project.

Table 1: Template Parameters

Parameter Name	Allowed Values/Type	Description	Resource Creation Type
resourceNamePrefix	String	All the resources are created with name containing this prefix. Example: demo-test	New
region	Valid regions supported by GCP [String]	Name of the region where project will be deployed. Example: us-central1	
serviceAccountMailId	String [Email Id]	Email address that identifies the service account.	

Parameter Name	Allowed Values/Type	Description	Resource Creation Type
vpcConnectorName	String	Name of the connector that handles the traffic between your serverless environment and your VPC network. Example: demo-test-vpc-connector	
bucketName	String	Name of the GCP storage bucket where the cloud function ZIP package will be uploaded. Example: demo-test-bkt	
cpuUtilizationTarget	Decimal (0,1]	The average CPU utilization of the VMs in the instance group the autoscaler should maintain. Example: 0.5	
healthCheckFirewallRuleName	String	Tag of the firewall rule that allows packets from health check probe IP ranges. Example: demo-test-healthallowall	Existing
insideFirewallRuleName	String	Tag of the firewall rules that allows communication in Inside VPC. Example: demo-test-inside-allowall	Existing
insideVPCName	String	Name of Inside VPC. Example: demo-test-inside	Existing
insideVPCSubnet	String	Name of Inside subnet. Example: demo-test-inside-subnt	Existing
machineType	String	Machine type for the ASAv VM. Example: e2-standard-4	

Parameter Name	Allowed Values/Type	Description	Resource Creation Type
maxASACount	Integer	The maximum ASAv instances allowed in the instance group. Example: 3	
mgmtFirewallRuleName	String	Tag of the firewall rules which allows communication in Management VPC. Example: demo-test-mgmt-allowall	
mgmtVPCName	String	Name of Management VPC. Example: demo-test-mgmt	
mgmtVPCSubnet	String	Name of Management Subnet. Example: demo-test-mgmt-subnt	
minASACount	Integer	The minimum ASAv instances available in the Instance Group at any given time. Example: 1	
outsideFirewallRuleName	String	Tag of the firewall rules which allows communication in outside VPC. Example: demo-test-outside-allowall	
outsideVPCName	String	Name of Outside VPC. Example: demo-test-outside	
outsideVPCSubnet	String	Name of Outside Subnet. Example: demo-test-outside-subnt	
publicKey	String	SSH key of the ASAv VM.	

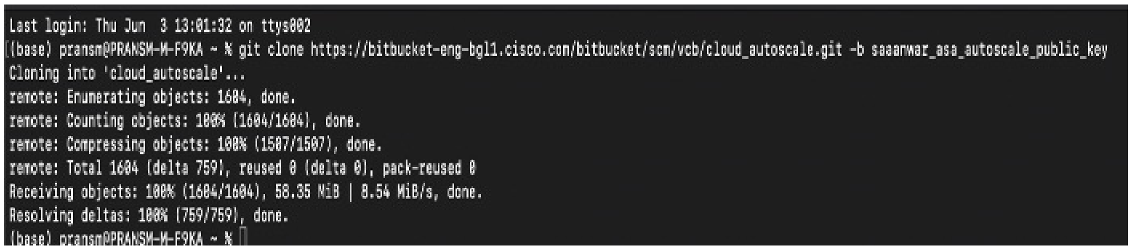
Parameter Name	Allowed Values/Type	Description	Resource Creation Type
sourceImageURL	String	Image of the ASAv which is to be used in the project. Example: https://www.googleapis.com/compute/v1/projects/cisco-public/global/images/cisco-asav-9-15-1-15	
Application server IP address	String	Internal IP address of the inside Linux machine. Example: 10.61.1.2	
Inside VPC Gateway IP address	String	Gateway of Inside VPC. Example: 10.61.1.1	
Management VPC Gateway IP address	String	Gateway of Management VPC. Example: 10.61.3.1	

Deploy the Auto Scale Solution

Step 1 Clone the Git repository to a local folder.

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

Example:

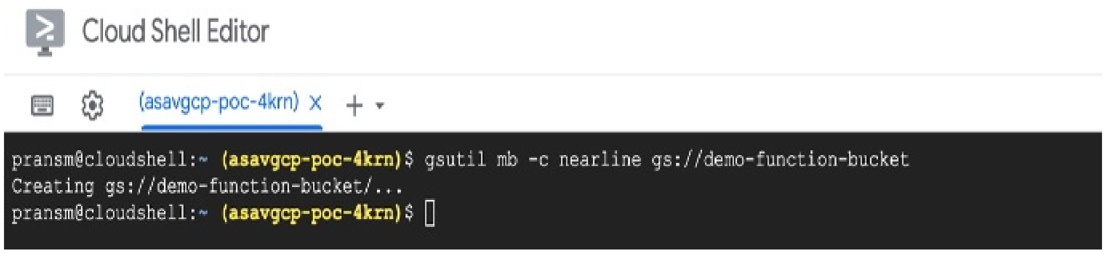


```
Last login: Thu Jun  3 13:01:32 on ttys002
[(base) pransm@PRANSM-M-F9KA ~ % git clone https://bitbucket-eng-bgl1.cisco.com/bitbucket/scm/vcb/cloud_autoscale.git -b saaarwar_asa_autoscale_public_key
Cloning into 'cloud_autoscale'...
remote: Enumerating objects: 1604, done.
remote: Counting objects: 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 ~ %
```

Step 2 Create the bucket in gcloud CLI.

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

Example:



```
pransm@cloudshell:~ (asavgcp-poc-4krn)$ gsutil mb -c nearline gs://demo-function-bucket
Creating gs://demo-function-bucket/...
pransm@cloudshell:~ (asavgcp-poc-4krn)$ 
```

Step 3 Build compressed Zip packages:

a) Create compressed Zip packages consisting of the following files from the folders `scalein_action` and `scaleout_action`.

- main.py
- basic_functions.py

  requirements.txt

b) Rename the compressed Zip packages to `scaleout-action.zip` and `scalein-action.zip`.

Note Navigate inside the folder, select the files, right-click, and select 'compress | archive' to make a .zip that GCP can read.

Step 4 Upload the compressed Zip packages (`scaleout-action.zip` and `scalein-action.zip`) to the Cloud Editor workspace.

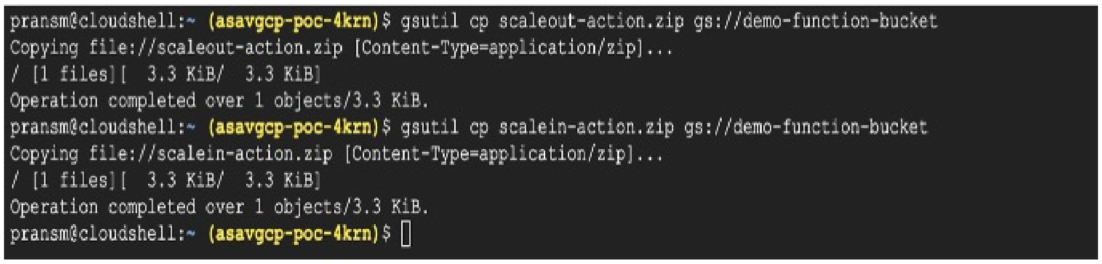
Step 5 Upload the following files from the deployment manager template to the Cloud Editor workspace.

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

Step 6 Copy the compressed Zip packages to the Bucket Storage.

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

Example:



```
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)$ 
```

Step 7 Create VPC and Subnet for inside, outside, and management interfaces.

In the management VPC, you need to have /28 subnet, for example, 10.8.2.0/28.

Step 8 You need three firewall rules for the interfaces inside, outside, and management. Also, you should have a firewall rule to allow the health check probes.

Step 9 Update the parameters in the Jinja and YAML files for the Pre-Deployment and ASAv Autoscale deployment.

a) Open the `asav_autoscale_params.yaml` file and update the following parameters:

- **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>

Note For the ASAv auto scale, the **cpuUtilizationTarget: 0.5** parameter is set and you can edit it according to your requirements.

This value signifies 50% CPU usage of all the ASAv Instance Group.

b) Open the `asav_autoscale.jinja` file and update the following parameters.

- **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>

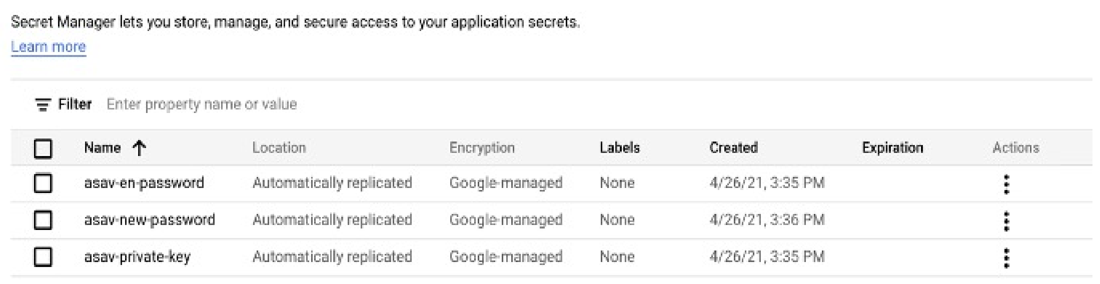
c) Open the `pre_deployment.yaml` file and update the following parameters.

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

- **bucketName**: <bucketName>

Step 10 Create three secrets for the following using the Secret Manager GUI. See https://console.cloud.google.com/security/secret-manager.

- 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		⋮

Step 11 Create the VPC connector.

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

Example:

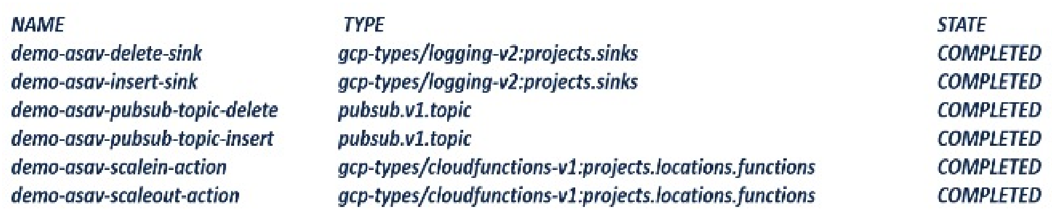
```
gcloud beta compute networks vpc-access connectors create demo-vpc-connector
--region us-central1 --subnet=outside-connect-28
Create request issued for: [demo-vpc-connector]
Waiting for operation [projects/asavgcp-poc-4krn/locations/us-central1/operations/
10595de7-837f-4c19-9396-0c22943ecf15] to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

Step 12 Deploy the pre-deployment YAML configuration.

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

Example:

```
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
```



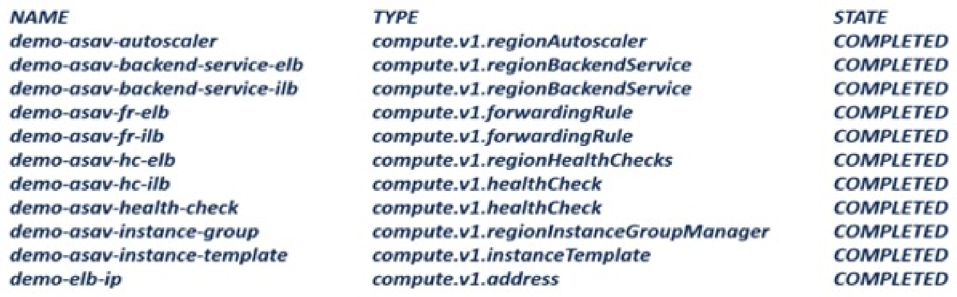
```
NAME                            TYPE                                               STATE
demo-asav-delete-sink           gcp-types/logging-v2:projects.sinks                COMPLETED
demo-asav-insert-sink           gcp-types/logging-v2:projects.sinks                COMPLETED
demo-asav-pubsub-topic-delete   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
```

Step 13 Create the ASAv auto scale deployment.

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

Example:

```
gcloud deployment-manager deployments create demo-asav-autoscale
--config asav_autoscale_params.yaml
The fingerprint of the deployment is b'1JCQi7I1-laWOY7vOLza0g=='
Waiting for create [operation-1624383774235-5c55e51d79d01-1a3acf92-4f3daf16]...done.
Create operation operation-1624383774235-5c55e51d79d01-1a3acf92-4f3daf16
completed successfully.
```



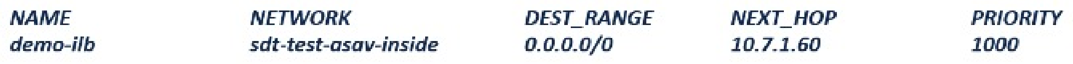
```
NAME                              TYPE                                        STATE
demo-asav-autoscaler              compute.v1.regionAutoscaler                 COMPLETED
demo-asav-backend-service-elb     compute.v1.regionBackendService             COMPLETED
demo-asav-backend-service-ilb     compute.v1.regionBackendService             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
```

Step 14 Create a route for ILB to forward the packets from the inside application to the Internet.

```
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>
```

Example:

```
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
```

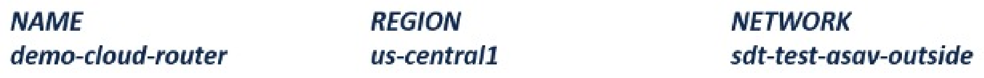
Step 15 Create Cloud Router and Cloud NAT.

```
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>
```

Example:

```
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 Logic

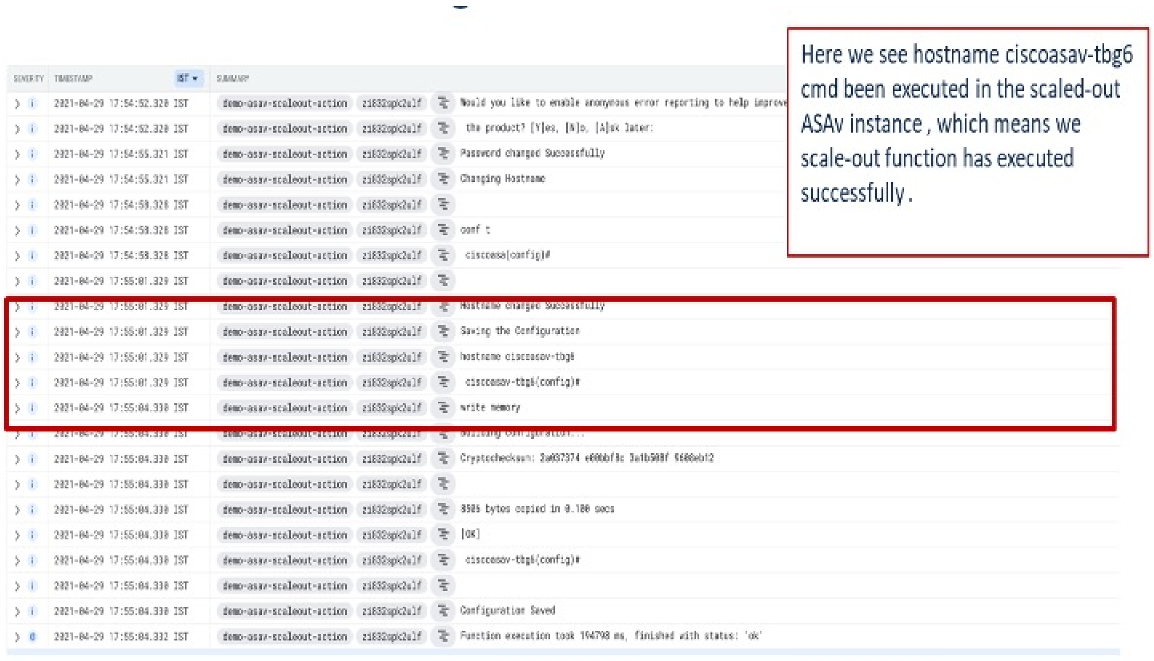
- The autoscaler treats the target CPU utilization level as a fraction of the average use of all vCPUs over time in the instance group.
- If the average utilization of your total vCPUs exceeds the target utilization, the autoscaler adds more VM instances. If the average utilization of your total vCPUs is less than the target utilization, the autoscaler removes instances.
- For example, setting a 0.75 target utilization tells the autoscaler to maintain an average utilization of 75% among all vCPUs in the instance group.
- Only CPU utilization metrics are used in scaling decisions.
- This logic is based on the assumption that load balancer will try to equally distribute connections across all ASAs, and on average, all ASAs should be loaded equally.

Auto Scale Logging and Debugging

Logs of cloud functions can be viewed as follows.

- Scale Out function logs

Figure 7: Scale Out Function Logs

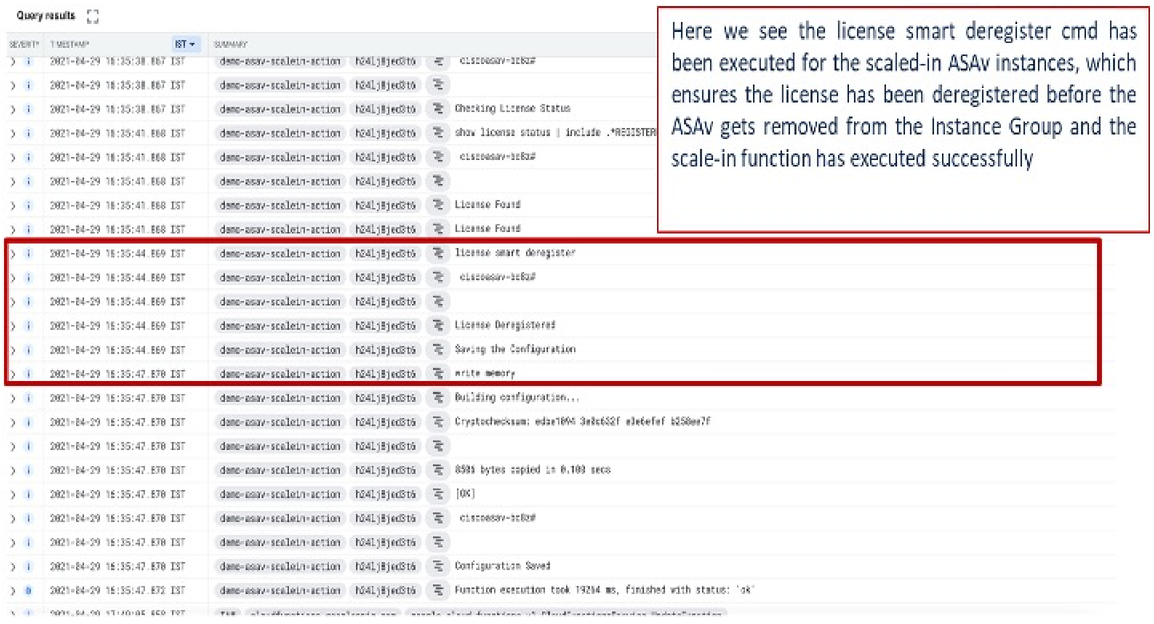


```
SEVERITY  TIMESTAMP                  IST   SUMMARY
>  i  2021-04-29 17:54:52.328 IST  demo-asav-scaleout-action  z1832spk2alf  Would you like to enable anonymous error reporting to help improve
>  i  2021-04-29 17:54:52.328 IST  demo-asav-scaleout-action  z1832spk2alf   the product? [Y]es, [N]o, [A]sk later:
>  i  2021-04-29 17:54:55.321 IST  demo-asav-scaleout-action  z1832spk2alf  Password changed Successfully
>  i  2021-04-29 17:54:55.321 IST  demo-asav-scaleout-action  z1832spk2alf  Changing Hostname
>  i  2021-04-29 17:54:58.328 IST  demo-asav-scaleout-action  z1832spk2alf
>  i  2021-04-29 17:54:58.328 IST  demo-asav-scaleout-action  z1832spk2alf  conf t
>  i  2021-04-29 17:54:58.328 IST  demo-asav-scaleout-action  z1832spk2alf   ciscoasa(config)#
>  i  2021-04-29 17:55:01.329 IST  demo-asav-scaleout-action  z1832spk2alf
>  i  2021-04-29 17:55:01.329 IST  demo-asav-scaleout-action  z1832spk2alf  Hostname changed Successfully
>  i  2021-04-29 17:55:01.329 IST  demo-asav-scaleout-action  z1832spk2alf  Saving the Configuration
>  i  2021-04-29 17:55:01.329 IST  demo-asav-scaleout-action  z1832spk2alf  hostname ciscoasav-tbg6
>  i  2021-04-29 17:55:01.329 IST  demo-asav-scaleout-action  z1832spk2alf   ciscoasav-tbg6(config)#
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf  write memory
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf  Building configuration...
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf  Cryptochecksum: 3a037374 e00bbf3c 3a1b508f 5608eb12
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf  8505 bytes copied in 0.100 secs
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf  [OK]
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf   ciscoasav-tbg6(config)#
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf
>  i  2021-04-29 17:55:04.330 IST  demo-asav-scaleout-action  z1832spk2alf  Configuration Saved
>  @  2021-04-29 17:55:04.332 IST  demo-asav-scaleout-action  z1832spk2alf  Function execution took 194798 ms, finished with status: 'ok'
```

Here we see hostname ciscoasav-tbg6 cmd been executed in the scaled-out ASAv instance , which means we scale-out function has executed successfully .

- Scale In function log

Figure 8: Scale In Function Log



Query results

```
SEVERITY  TIMESTAMP  IST      SUMMARY
> i  2021-04-29 16:35:38.867 IST  demo-asav-scalein-action  h24lj8jed3t6  ciscoasav-bc8z#
> i  2021-04-29 16:35:38.867 IST  demo-asav-scalein-action  h24lj8jed3t6
> i  2021-04-29 16:35:38.867 IST  demo-asav-scalein-action  h24lj8jed3t6  Checking License Status
> i  2021-04-29 16:35:41.868 IST  demo-asav-scalein-action  h24lj8jed3t6  show license status | include .*REGISTER
> i  2021-04-29 16:35:41.868 IST  demo-asav-scalein-action  h24lj8jed3t6  ciscoasav-bc8z#
> i  2021-04-29 16:35:41.868 IST  demo-asav-scalein-action  h24lj8jed3t6
> i  2021-04-29 16:35:41.868 IST  demo-asav-scalein-action  h24lj8jed3t6  License Found
> i  2021-04-29 16:35:41.868 IST  demo-asav-scalein-action  h24lj8jed3t6  License Found
> i  2021-04-29 16:35:44.869 IST  demo-asav-scalein-action  h24lj8jed3t6  license smart deregister
> i  2021-04-29 16:35:44.869 IST  demo-asav-scalein-action  h24lj8jed3t6  ciscoasav-bc8z#
> i  2021-04-29 16:35:44.869 IST  demo-asav-scalein-action  h24lj8jed3t6
> i  2021-04-29 16:35:44.869 IST  demo-asav-scalein-action  h24lj8jed3t6  License Deregistered
> i  2021-04-29 16:35:44.869 IST  demo-asav-scalein-action  h24lj8jed3t6  Saving the Configuration
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6  write memory
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6  Building configuration...
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6  Cryptochecksum: eda01094 3e8c632f e1e6efef b258ee7f
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6  8585 bytes copied in 0.100 secs
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6  [OK]
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6  ciscoasav-bc8z#
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6
> i  2021-04-29 16:35:47.870 IST  demo-asav-scalein-action  h24lj8jed3t6  Configuration Saved
> i  2021-04-29 16:35:47.872 IST  demo-asav-scalein-action  h24lj8jed3t6  Function execution took 19214 ms, finished with status: 'ok'
```

Here we see the license smart deregister cmd has been executed for the scaled-in ASAv instances, which ensures the license has been deregistered before the ASAv gets removed from the Instance Group and the scale-in function has executed successfully

Auto Scale Guidelines and Limitations

- Only IPv4 is supported.
- Supported licensing is BYOL only. PAYG is not available for ASAv on GCP.
- The external Load Balancer is created by the template, and therefore, any specific DNS requirements for Load Balancer's public IP are out of the scope.
- An assumption is made that the application is behind a user-created Load Balancer, and the ASAv will route all traffic to this Load Balancer (instead of directly sending traffic to a specific application IP).
- Details about the need for TAGs, redundancy, and Load Balancer affinity configurations are not considered.
- ASAv credentials are visible to you as:
  - Clear text in the serverless code.
  - In all the instances in the Instance Group.
  - In the Instance Template, if you are using a shared GCP account.

  Such sensitive data can be protected using the public key service in GCP.

Important Cisco recommends tracking the ASAv registration with the licensing server periodically to check if the scaled-out ASAs are registering with the licensing server as expected, and scaled-in ASAv instances are getting removed from the license server).

Auto Scale Troubleshooting

The following are common error scenarios and debugging tips for ASAv auto scale for GCP:

- `main.py` not found—Make sure that the Zip package is made only from the files. You can go to cloud functions and check the file tree. There should not be any folder.
- Error while deploying the template—Make sure that all the parameters values within "<>" are filled in. jinja and .yaml as well, or the deployment by the same name exists already.
- Google Function cannot reach ASAv—Make sure that the VPC connector is created and the same name is mentioned in the YAML parameter file.
- Authentication Failed while SSH-ing ASAv—Make sure that the Public and Private key pair is correct.
- License Registration Failed—Make sure that the License ID token is correct. Also, ensure that the Cloud NAT is created and ASAv is able to reach tools.cisco.com.