

# Deploy the ASA Virtual Auto Scale Solution on GCP

- Overview, on page 1
- Download the Deployment Package, on page 3
- Auto Scale Solution Components, on page 3
- Prerequisites, on page 6
- Deploy the Auto Scale Solution, on page 12
- Auto Scale Logic, on page 17
- Logging and Debugging, on page 17
- Guidelines and Limitations, on page 18
- Troubleshooting, on page 19

# **Overview**

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

### **About the Auto Scale Solution**

ASA virtual 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 ASA virtual auto scale for GCP implementation include:

- GCP Deployment Manager template-based deployment.
- Support for scaling metrics based on CPU.
- Support for ASA virtual deployment and multi-availability zones.
- Completely automated configuration automatically applied to scaled-out ASA virtual 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 ASA virtual auto scale for GCP is an automated horizontal scaling solution that positions an ASA virtual 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 ASA virtual instances in the instance group; the firewall then forwards traffic to the application.
- The ILB distributes outbound Internet traffic from an application to ASA virtual 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 ASA virtual instances in the scale set will be scaled and configured automatically based on load conditions.

### Figure 1: ASA Virtual Auto Scale Use Case



### Scope

This document covers the detailed procedures to deploy the serverless components for the ASA virtual 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 ASA virtual 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.).

Download the files required to launch the ASA virtual auto scale for GCP solution. Deployment scripts and templates for your ASA version are available in the GitHub repository.

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 ASA virtual 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 <a href="https://cloud.google.com/deployment-manager/docs">https://cloud.google.com/deployment-manager/docs</a> for more information.

#### Figure 2: Deployment Manager View

		=	Google Cloud Platform	🕽 ASAVGCP 👻	Q Search products and resource			
	Cloud Deployment Manager Google	M	Deployment Manager	demo-predeployment	DELETE			
	Create and manage cloud resources with simple templates	Ċ	Deployments					
	GO TO CLOUD DEPLOYMENT MANAGER TAKE QUICKSTART	:=	i≣ Type registry	demo-predeployment has been deployed				
				Overview - demo-predeployment				
				👻 🖿 pre deployment pre deployment,jinja				
				<ul> <li>demo-saar-insert-sink gcp-types/logging-v2 projects ainks</li> <li>demo asar pubsub topic insert, pubsub topic</li> </ul>				
				demo-ssav-scaleout-action ocn immediate via actions of projects locations of	miestis locations functions			
				demo-sasv-delete-sink gcp	-types/logging-v2.projects.sinks			
				📕 demo-asav-pubsub-topio-di	alete pubsub topic			
				demo-asay-scale in-action				
				gcp-types/cloudfunctions-vilg	rojects 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 <a href="https://cloud.google.com/compute/docs/instance-groups">https://cloud.google.com/compute/docs/instance-groups</a> 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 <a href="https://cloud.google.com/compute/docs/autoscaler">https://cloud.google.com/compute/docs/autoscaler</a> for more information.

#### Figure 4: Target Utilization 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 ASA virtual 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 ASA virtual 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 ASA virtual instance and run the command for license deregistration.

### **High-Level Overview of Autoscale Solution**

Figure 5: Autoscale Solution Overview



# **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 ASA virtual requires 3 network interfaces, thus your virtual network requires 3 subnets for:

L

- Management traffic
- Inside traffic
- Outside traffic

#### Figure 6: VPC Network View

😸 Nev	Tab × 🖬 v	PC networks – VPC network – IX	+									
←	o c	음 후 https://console.clou	d.google.com/networking/netw	vorks/list?project	=asavgcp-poc-4	km				습		
≡	Google Cloud Platform	🕽 ASAVGCP 👻	Q. Search p	roducts and re	sources				~		2.	0
#	VPC network	VPC networks	CREATE VPC NETWORK	C REFRESH	6							
			asia-south2	default			10.190.0.0/20	10.190.0.1				
2	VPC networks		australia- southeast2	default			10.192.0.0/20	10.192.0.1				
	External in addresses			1	1460	Custom				2	Off	6
6	Bring your own IP		us-central 1	demo-test-			10.61.1.0/24	10.61.1.1				
22	Firewall			subnt								
x	Routes	▼ demo-test-mgmt		2	1460	Custom				1	Off	
φ	VPC network peering		us-pentral 1	demo-test- mgmt- subnt			10.61.3.0/24	10.61.3.1				
×	Shared VPC		us-central 1	demo-test-			10.62.1.0/28	10.62.1.1				
\$	Serverless VPC access			vpcconnect								
(D)	Packet mirroring			1	1460	Custom				1	Off	
цр.			us-central1	demo-test- outside- subrit			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 ASA virtual. Required for communication between the serverless functions and ASA virtual.
- Application-specific protocol/ports Required for any user applications (for example, TCP/80, etc.).

### **Prepare the ASA Configuration File**

Prepare an ASA virtual 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 ASA virtual 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 ASA virtual 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
1
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 hcl
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
 1
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 ASA virtual 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 ASA virtual device when you deploy the GCP Deployment Manager template into your GCP project.

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	

**Table 1: Template Parameters** 

I

Parameter Name	Allowed Values/Type	Description	Resource Creation Type
serviceAccountMailId	String [ Email Id]	Email address that identifies the service account.	
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

Parameter Name	Allowed Values/Type	Description	Resource Creation Type
machineType	String	Machine type for the ASA virtual VM.	
		Example: e2-standard-4	
maxASACount	Integer	The maximum ASA virtual 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 ASA virtual 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	

Parameter Name	Allowed Values/Type	Description	Resource Creation Type
publicKey	String	SSH key of the ASA virtual VM.	
sourceImageURL	String	Image of the ASA virtual 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**

Procedure

**Step 1** Clone the Git repository to a local folder.

git clone git\_url -b branch\_name

### Example:

Last login: Thu Jun 3 13:81:32 on ttys002 ((base) pransm@PRANSM-M-F9KA ~ % git clone https://bitbucket-eng-bgl1.cisco.com/bitbucket/scm/vcb/cloud\_autoscale.git -b saaanwar\_asa\_autoscale\_public\_key Cloning into 'cloud\_autoscale'... remote: Enumerating objects: 160% (d04/1604), done. remote: Compressing objects: 100% (1507/1507), done. remote: Total 1604 (delta 759), reused 0 (delta 0), pack-reused 0 Receiving ubjects: 100% (1644/1604), 58.35 MiB | 8.54 MiB/s, done. Resolving deltas: 100% (759/759), done.

**Step 2** Create the bucket in gcloud CLI.

gsutil mb -c nearline gs://bucket\_name

### Example:

	Cloud Shell	Editor
	🖃 😥 (asavgcp	p-poc-4krn) × + -
	pransm@cloudshell:~ Creating gs://demo- pransm@cloudshell:~	<pre>(asavgcp-poc-4krn)\$ gsutil mb -c nearline gs://demo-function-bucket function-bucket/ (asavgcp-poc-4krn)\$ []</pre>
Step 3	Build compressed 2 a) Create compression scaleout_a	Zip packages: ssed Zip packages consisting of the following files from the folders scalein_action and ction.
	• hasic fun	tions by
	requireme	nts.txt
	b) Rename the co	mpressed 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 compress workspace.	ssed Zip packages (scaleout-action.zip and scalein-action.zip) to the Cloud Editor
Step 5	Upload the following	ng files from the deployment manager template to the Cloud Editor workspace.
	<ul> <li>asav_autoscale</li> </ul>	e.jinja
	• asav_autoscal	e_params.yaml
	• pre_deployme	nt.jinja
	<ul> <li>pre_deployme</li> </ul>	nt.yaml
Step 6	Copy the compress	ed Zip packages to the Bucket Storage.
	•gsutil cp sc	aleout-action.zip gs://bucket_name
	•gsutil cp sc	alein-action.zip gs://bucket_name
	Example:	
	<pre>pransm@cloudshell:~ Copying file://scalea / [1 files][ 3.3 Ki] Operation completed pransm@cloudshell:~ Copying file://scalea / [1 files][ 3.3 Ki] Operation completed o pransm@cloudshell:~</pre>	<pre>(asavgcp-poc-4krn)\$ gsutil cp scaleout-action.zip gs://demo-function-bucket but-action.zip [Content-Type=application/zip] 8/ 3.3 KiB] bver 1 objects/3.3 KiB. (asavgcp-poc-4krn)\$ gsutil cp scalein-action.zip gs://demo-function-bucket in-action.zip [Content-Type=application/zip] 8/ 3.3 KiB] over 1 objects/3.3 KiB. (asavgcp-poc-4krn)\$ []</pre>

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 ASA virtual 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 ASA virtual 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 ASA virtual 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

∓ Fi	Iter Enter property nam	e or value					
	Name 个	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

ТҮРЕ	STATE
gcp-types/logging-v2:projects.sinks	COMPLETED
gcp-types/logging-v2:projects.sinks	COMPLETED
pubsub.v1.topic	COMPLETED
pubsub.v1.topic	COMPLETED
gcp-types/cloudfunctions-v1:projects.locations.functions	COMPLETED
gcp-types/cloudfunctions-v1:projects.locations.functions	COMPLETED
	TYPE gcp-types/logging-v2:projects.sinks gcp-types/logging-v2:projects.sinks pubsub.v1.topic pubsub.v1.topic gcp-types/cloudfunctions-v1:projects.locations.functions gcp-types/cloudfunctions-v1:projects.locations.functions

**Step 13** Create the ASA virtual 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'1JCQi7II-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

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-centrall --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

L

```
--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.

# Logging and Debugging

Logs of cloud functions can be viewed as follows.

Scale Out function logs

Figure 7: Scale Out Function Logs

		-					Here we see hostname ciscoasav-tbg6
SIMERTY	TIMESTAMP	101.4	\$3,000,059			cmd been executed in the scaled-out	
> 0	2821-84-29 17:54:52.328	IST	femo-asav-scaleout-action	zi832spk2ulf	1	Nould you like to enable anonymous error reporting to help improve	464
> 0	2821-84-29 17:54:52.328	IST	<pre>femo-assy-scaleout-action</pre>	zi832spk2ulf	2	the product? [Y es, [N]o, [A]sk later:	ASAV Instance, which means we
> 0	2821-04-29 17:54:55.321	IST	femo-assy-scaleout-action	zi832spk2ulf	2	Password charged Successfully	scale-out function has executed
> 0	2821-04-29 17:54:55.321	IST	femo-assy-scaleout-action	ziE02spk2ulf	R	Changing Hostname	au e e e e fullu
> 1	2821-84-29 17:54:58.328	IST	femo-assy-scaleout-action	zi652spk2slf	1h		successfully.
> 0	2821-84-29 17:54:58.328	IST	femo-asav-scaleout-action	zi852spk2ulf	1	comf t	
> 0	2821-84-29 17:54:58.328	IST	femo-asay-scaleout-action	zi832spk2ulf	1	cistessa(config)#	
> 0.	2821-84-29 17:55:81.329	IST	femo-asav-scaleout-action	zi832spic2ulf	2		
2.1	2821-84-29 17:55:81.329	151	femo-assy-scaleout-action	21832spk2e1f	s.	Hastname changed Successfully	
> 0	2821-84-29 17:55:81.329	IST	femo-assy-scaleout-action	zi832spk2ulf	2	Saving the Configuration	
> 1	2821-84-29 17:55:81.329	IST	femo-assy-scaleout-action	ziE02spk2ulf	R.	hostneme classesev-tbg8	
> 0	2821-84-29 17:55:81.329	IST	femo-asav-scaleout-action	zi652spk2slf	R.	discessor-thg8(config)#	
> 0	2821-84-29 17:55:84.338	IST	femo-asav-scaleout-action	zi832spk2ulf	14	write memory	
2.9	2821-84-29 17:55:84.338	151	1000-3517-55210001-355100	218523pc2011	5	antend on theaton	
> 0	2821-84-29 17:55:84.338	IST	femo-asav-scaleout-action	zi832spk2ulf	3	Crystochecksun: 2a037374 e000bf8c 3a1b508f \$680eb12	
> 0	2821-84-29 17:55:84.338	IST	femo-assy-scaleout-action	21832spk2ulf	2		
> ;	2821-84-29 17:55:84.338	IST	femo-assy-scaleout-action	ziE32spk2slf	2	3525 bytes copied in 0.180 secs	
> 1	2821-84-29 17:55:84.338	IST	femo-assy-scaleout-action	ziE02spk2wlf	R	[as]	
> 1	2821-84-29 17:55:84.338	IST	femo-asav-scaleout-action	zi652spk2wlf	1	discessor-thg8(config)#	
> 0	2821-84-29 17:55:84.338	IST	femo-asav-scaleout-action	zi832spk2slf	14		
> 0	2821-84-29 17:55:84.338	IST	femo-asay-scaleout-action	zi832spk2ulf	2	Configuration Seved	
1.0	2821-84-29 17:55:84.332	IST	demo-assy-scaleout-action	zi832spk2alf	2	Function execution took 194798 ms, finished with status: 'ok'	

• Scale In function log

Figure 8: Scale In Function Log

Query	results []					Here we see the licence smart deveristor and he	
EVENT?	T MESTAMP IST -	SUMMARY				Here we see the license smart deregister crid ha	
× 1	2021-84-29 16:35:38.867 IST	demo-asay-scalein-action	h24lj#jed0to	4	CISTORGAV-ICB2#	been executed for the scaled-in ASAv instances, whic	
) (	2921-84-29 18:35:38.867 IST	demo-asav-scalein-action	h24lj#jed8t6	5		ensures the license has been deregistered before th	
> 1	2021-04-29 10:25:38.067 IST	demo-asav-scalein-action	h241j8jed3t6	1	Checking License Status	citates the license has been deregistered before th	
) (	2021-04-29 18:25:41.000 IST	demo-asav-scalein-action	h241;8jed3t6	5	show license status   include .*REGISTER	ASAV gets removed from the Instance Group and th	
) (	2021-04-29 16:25:41.868 IST	demo-assy-scalein-action	h241j8jed8t6	5	ciscoesav-brBz#	scale-in function has executed successfully	
•	2021-04-29 18:25:41.868 IST	demo-asav-scalein-action	h241j8jed3t6	2			
i (	2021-04-29 18:35:41.868 IST	demo-asav-scalein-action	h241;8jed3t6	2	License Found		
1	2821-84-29 18:85:41.868 IST	demo-assy-scalein-action	h241j8jedSt6	3	License Found		
) 1	2021-84-29 15:35:44.869 IST	demo-asav-scalein-action	h241;8jed3t6	S.	license smart deregister		
) (i	2021-84-29 18:35:44.869 IST	demo-asav-scalein-action	h241;8jed3t6	2	ciscossav-br8z#		
) (I	2021-04-29 15:35:44.869 IST	demo-asay-scalein-action	h241j8jed3t6	2			
) (	2021-84-29 15:35:44.869 IST	demo-asay-scalein-action	h241;8jed3t6	2	E License Deregistered		
) (	2021-84-29 16:35:44.869 IST	demo-assy-scalein-action	h241j8jed3t6	R Saving the Configuration			
1	2021-84-29 15:35:47.870 IST	demo-asay-scalein-action	h241j8jedSt6	2	write memory		
> ()	2021-84-29 15:35:47.870 IST	demo-asay-scalein-action	h241j8jed3t6	1	Building configuration		
) (	2021-84-29 15:35:47.870 IST	dane-asay-scalein-action	h241j8jedSt6	7	🛬 Cryptochecksum: edua1894 3edotS2f eletefef b258ee7f		
) (	2021-84-29 15:35:47.870 IST	danc-asay-scalein-action	h241j8jedSt6	7			
) (	2821-84-29 15:35:47.878 IST	demo-asav-scalein-action	h241j8jedStá	1	8595 bytes copied in 8.109 seco		
) (	2021-84-29 15:35:47.870 IST	dame-asay-scalein-action	h241j8jedSt6	7	[0K]		
	2021-84-29 15:35:47.870 IST	danc-asay-scalein-action	h241j8jedStá	7	ciscossav-bc8z#		
) (	2921-84-29 15:35:47.878 IST	dans-asay-scalein-action	h241j8jedSt6	7			
) [	2021-84-29 15:35:47.870 IST	dame-asav-scalein-action	h241j8jedSt6	1	Configuration Saved		
) 0	2021-84-29 15:35:47.872 IST	demo-asav-scalein-action	h241j8jedSt6	The Function execution took 19264 ms, finished with status: 'ok'			
	1011.54.10 11-40-65 550 TOT				land freekdaan of Mandteastanetanian m	adaka Taraké ny	

# **Guidelines and Limitations**

- Only IPv4 is supported.
- Supported licensing is BYOL only. PAYG is not available for ASA virtual 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 ASA virtual 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.
- ASA virtual 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.

L

¢

Important

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

# Troubleshooting

The following are common error scenarios and debugging tips for ASA virtual 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 ASA virtual—Make sure that the VPC connector is created and the same name is mentioned in the YAML parameter file.
- Authentication Failed while SSH-ing ASA virtual—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 ASA virtual is able to reach tools.cisco.com.