

## VM-Series for GCP

---



# GCP Terraform Template Deployment Guide

Deploys an External Load Balancer, multiple VM-Series NGFW, GKE Cluster with Internal Load Balancer and Cluster Nodes in different zone. This deployment model is commonly referred to as a Load Balancer Sandwich.

<https://www.paloaltonetworks.com>

## Table of Contents

---

<b>Version History .....</b>	<b>3</b>
<b>1. About Terraform Templates.....</b>	<b>4</b>
<b>2. Support Policy .....</b>	<b>5</b>
<b>3. Instances used.....</b>	<b>5</b>
<b>4. Prerequisites .....</b>	<b>5</b>
4.1    Create GCP account .....	5
4.2    Install the Google Cloud SDK .....	6
4.3    Accept the EULA (If Required).....	6
4.4    Create a Project .....	6
4.5    Enable the API .....	7
4.6    Create a Bootstrap Bucket .....	9
4.7    Download the Terraform Template Files .....	11
4.8    Gather Information and Update the Template File .....	12
<b>5. Launch the Template.....</b>	<b>12</b>
<b>6. Review what was created .....</b>	<b>14</b>
<b>7. Container Deployment .....</b>	<b>19</b>
<b>8. Access the firewall .....</b>	<b>21</b>
<b>9. Access the Webservers via ELB.....</b>	<b>25</b>
<b>10. Cleanup.....</b>	<b>26</b>
10.1    Delete the deployment.....	26
<b>11. Conclusion .....</b>	<b>26</b>
<b>Appendix A.....</b>	<b>27</b>
Troubleshooting tips.....	27

## Version History

Version number	Comments
1.0	Initial Draft

## 1. About Terraform Templates

GCP Terraform Templates, are files that can deploy, configure, and launch GCP resources such as VPC networks & subnets, security groups, firewall rules, route tables, load balancers, GKE Clusters, and more. These templates are used for ease of deployment and are key to any cloud deployment model.

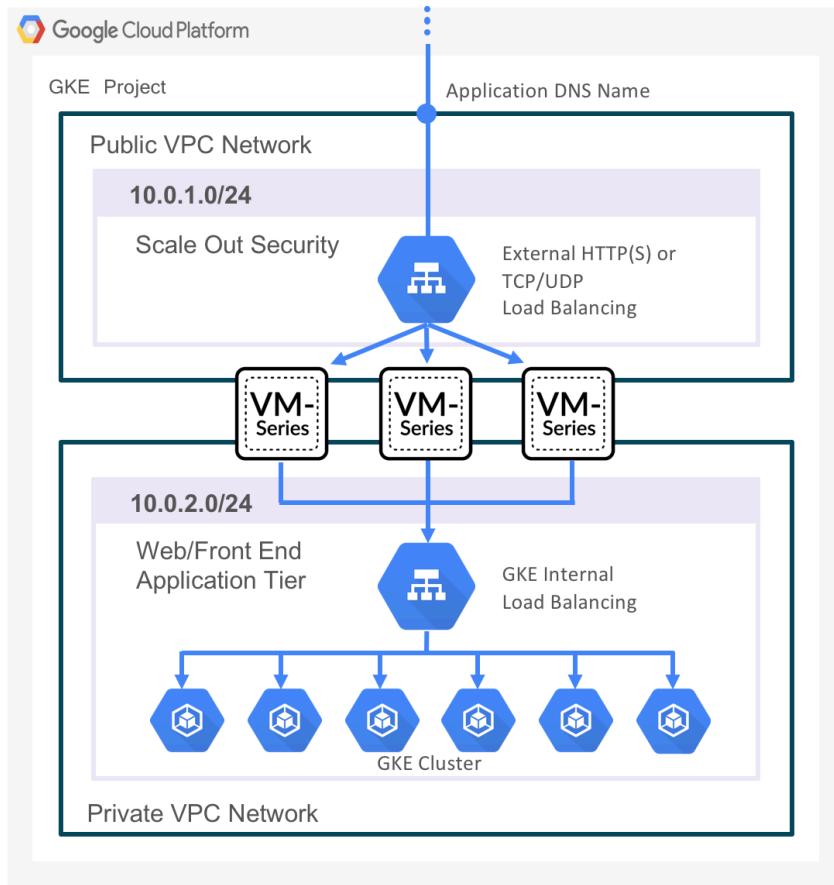
For more information on Templates refer to Google's documentation

<https://cloud.google.com/community/tutorials/managing-gcp-projects-with-terraform>

There are also many Terraform template s available here:

<https://github.com/GoogleCloudPlatform/terraform-google-examples>

This document will explain how to deploy a Terraform template that launches everything that is shown below in the diagram. This includes, an HTTP ELB, multiple VM-Series firewall, GKE Cluster and the subnets. In addition, the Terraform template performs a native bootstrapping feature on the VM-Series firewall that allows for additional configuration of the VM-Series firewall (such as routes, security policies, NAT rules, management interface swap, etc.) Once the Terraform template has been deployed, the network topology will align with the following diagram:



## 2. Support Policy

This template is released under an as-is, best effort, support policy. These scripts should be seen as community supported and Palo Alto Networks will contribute our expertise as and when possible. We do not provide technical support or help in using or troubleshooting the components of the project through our normal support options such as Palo Alto Networks support teams, or ASC (Authorized Support Centers) partners and backline support options. The underlying product used (the VM-Series firewall) by the scripts or templates are still supported, but the support is only for the product functionality and not for help in deploying or using the template or script itself.

Unless explicitly tagged, all projects or work posted in our GitHub repository (at <https://github.com/PaloAltoNetworks/googlecloud>) or sites other than our official Downloads page on <https://support.paloaltonetworks.com> are provided under the best effort policy.

## 3. Instances used

When using this Terraform template the following machine types are used:

Instances	Machine Types
GKE Nodes	f1-micro
VM Series Firewall	n1-standard-4

**Note: There are costs associated with each machine type launched load balancers and other components, please refer to the Google instance pricing page <https://cloud.google.com/compute/pricing>**

## 4. Prerequisites

Here are the prerequisites required to successfully launch this template:

- Terraform installed

### 4.1 Create GCP account

If you do not have a GCP account already, go to <https://cloud.google.com/free/> and create an account.

\*If you are using a free GCP account, the Compute Engine API CPU Quota is 8 per region. To increase the CPU quota, you will need to upgrade your account. The template will deploy multiple firewalls which can hit the maximum CPU quota. If you are using the free trial account, you can modify the highlighted option below to deploy a single FW in the main.tf file:

```
// Create a new PAN-VM instance
resource "google_compute_instance" "firewall" {
  name      = "${var.firewall_name}-${count.index + 1}"
  machine_type = "${var.machine_type_fw}"
  zone      = "${element(var.zone_fw, count.index)}"

  # min_cpu_platform      = "${var.machine_cpu_fw}"
  can_ip_forward        = true
  allow_stopping_for_update = true
  count                = 2
```

## 4.2 Install the Google Cloud SDK

Template installations in GCP are performed from the CLI. Install the SDK/CLI by selecting the relevant platform from the following link and following the installation instructions:

<https://cloud.google.com/sdk/>

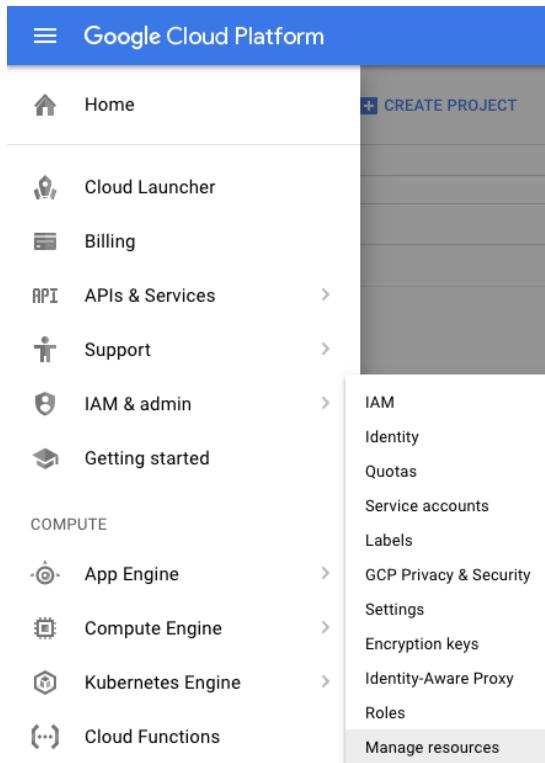
After installing Cloud SDK, install the kubectl command-line tool by running the following command:

```
gcloud components install kubectl
```

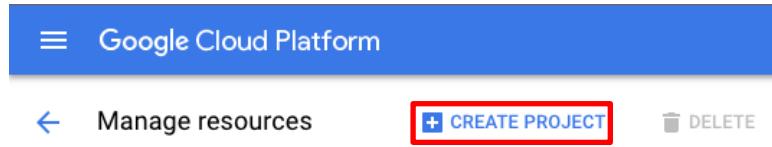
## 4.3 Accept the EULA (If Required)

## 4.4 Create a Project

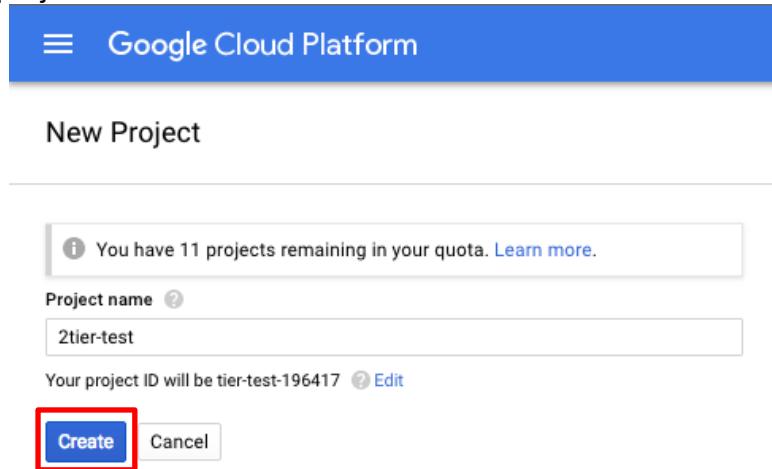
All GCP resources are deployed to a GCP Project. A GCP Project is an organizational boundary that separates users, resources, billing information, etc. A GCP Project is similar to an AWS VPC or an Azure Resource Group. By default, GCP will create a Project upon creation of an account. If that is not the case or to manually create a dedicated project, use the drop-down on the left and select **IAM & admin > Manage Resources**:



Click **Create Project**:



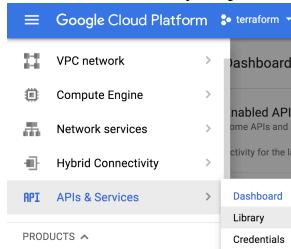
Specify a name for the project and click **Create**:



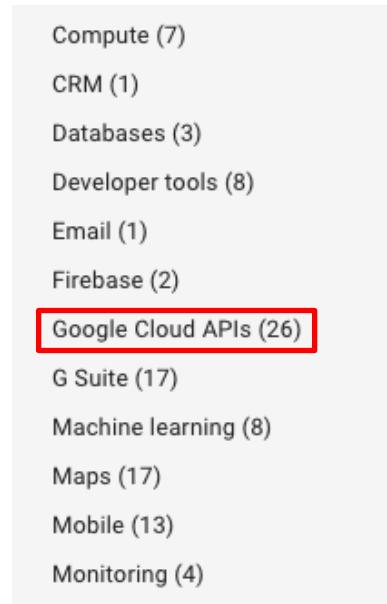
**Note: GCP Project creation will take a few minutes.**

## 4.5 Enable the API

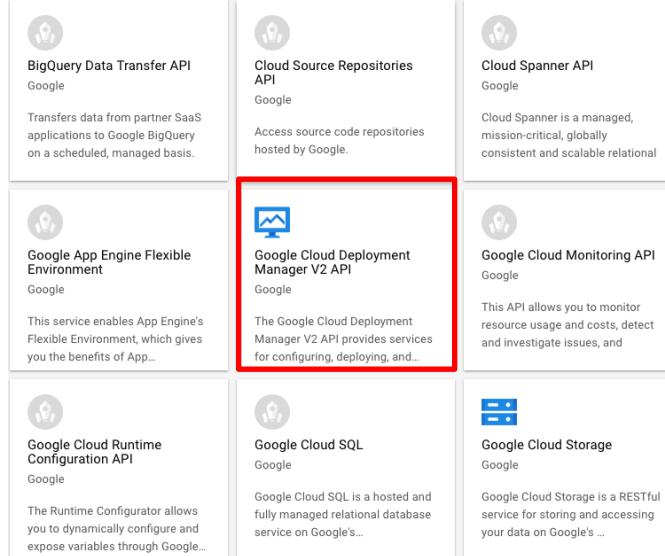
Deploying a template requires the API be enable on the project. Navigate to **APIs & Services > Library**:



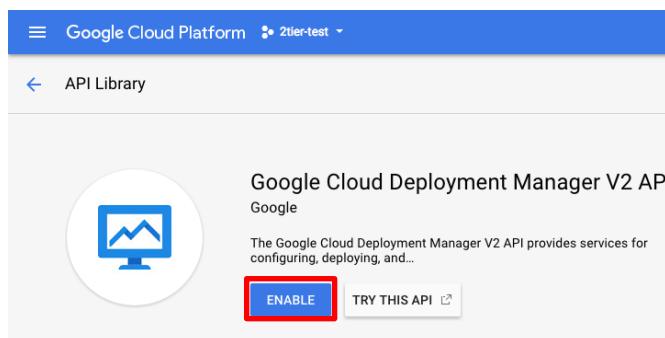
Select Google Cloud APIs on the left-hand-side:



Select Google Cloud Deployment Manager V2 API:



Select Enable:

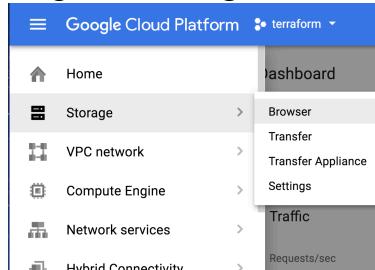


Note: Enabling the API for the project will take a few minutes to complete.

## 4.6 Create a Bootstrap Bucket

Bootstrapping is a feature of the VM-Series firewall that allows you to load a pre-defined configuration into the firewall during boot-up. This ensures that the firewall is configured and ready at initial boot-up, thereby removing the need for manual configuration. The bootstrapping feature also enables automating deployment of the VM-Series firewall.

In order to create a Bootstrap bucket, navigate to **Storage > Browser**:



Click **Create Bucket**:

A screenshot of the 'Cloud Storage Buckets' creation page. The title is 'Cloud Storage Buckets'. The text explains that Cloud Storage lets you store unstructured objects in containers called buckets. It offers options to 'Create bucket' or 'Take the quickstart'. The 'Create bucket' button is highlighted with a red box.

Specify a globally-unique bucket name and regional settings and click **Create**:

# Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich

[← Create a bucket](#)

Name [?](#)  
Must be unique across Cloud Storage. If you're [serving website content](#), enter the website domain as the name.

Default storage class [?](#)  
[Compare storage classes](#)

Multi-Regional  
 Regional  
 Nearline  
 Coldline

Location

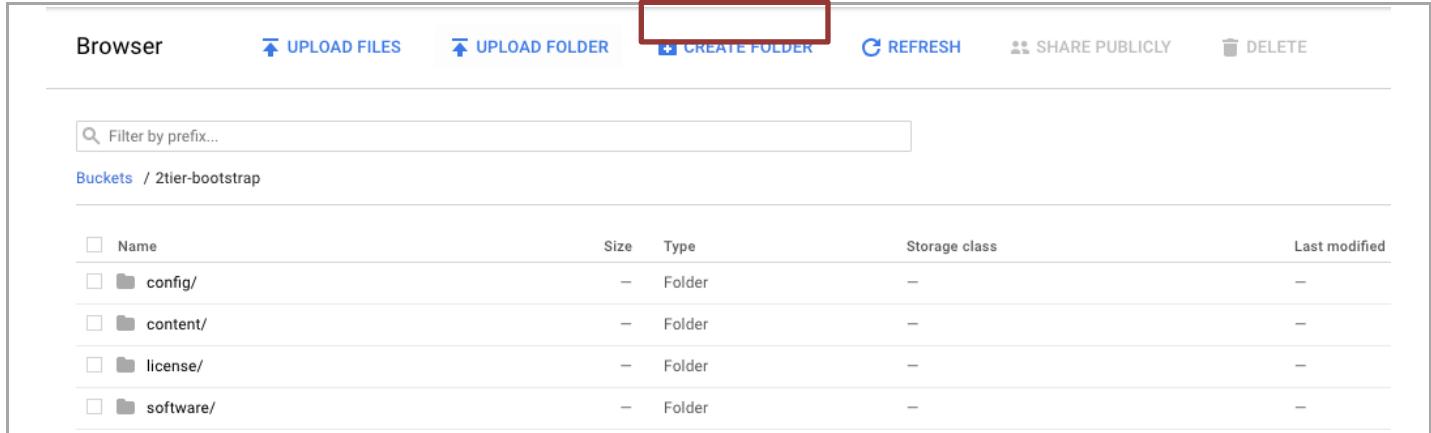
United States

Storage cost	Retrieval cost	Class A operations <a href="#">?</a>	Class B operations <a href="#">?</a>
\$0.026 per GB-month	Free	\$0.005 per 1,000 ops	\$0.0004 per 1,000 ops

[Specify labels](#)

[Create](#) [Cancel](#)

You will need to enter a globally unique bucket name. GCP will warn you if the name is not unique. Once the bucket is created, click on the newly created bucket and add four folders called **config**, **license**, **software** and **content** by clicking on **Create Folder**:



Browser    [UPLOAD FILES](#)    [UPLOAD FOLDER](#)    [CREATE FOLDER](#)    [REFRESH](#)    [SHARE PUBLICLY](#)    [DELETE](#)

Filter by prefix...

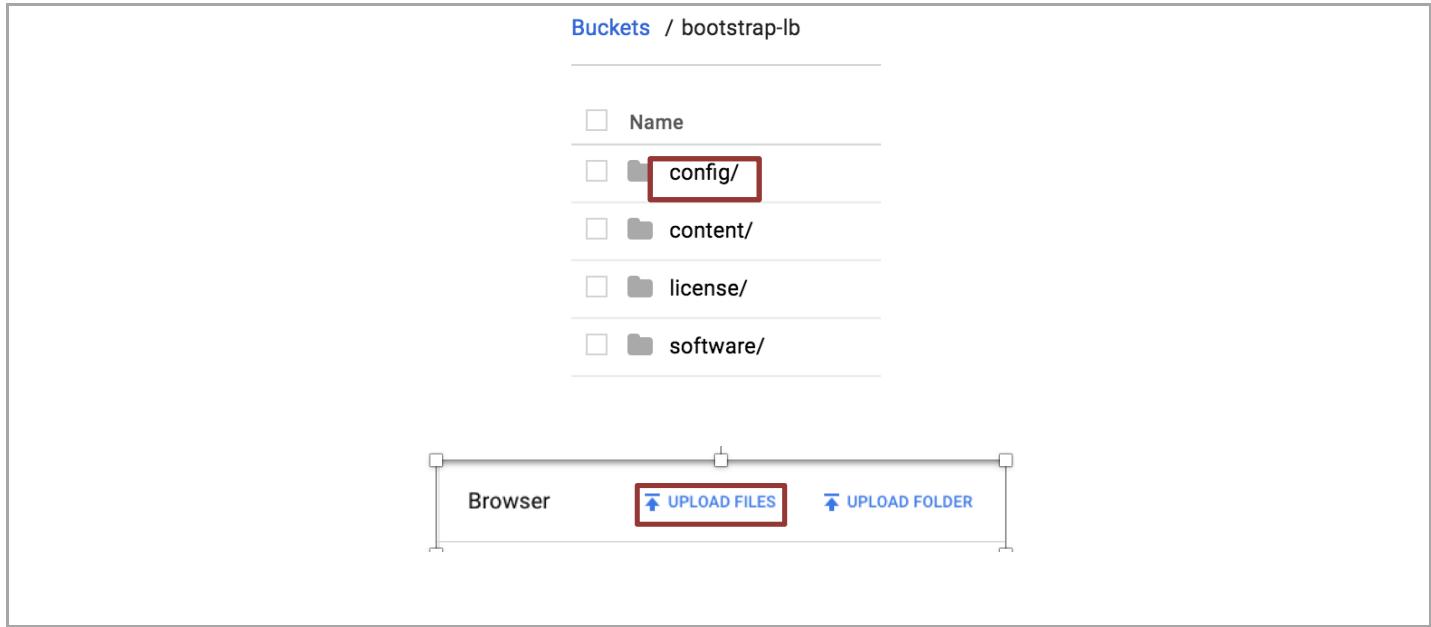
Buckets / 2tier-bootstrap

<input type="checkbox"/>	Name	Size	Type	Storage class	Last modified
<input type="checkbox"/>	config/	—	Folder	—	—
<input type="checkbox"/>	content/	—	Folder	—	—
<input type="checkbox"/>	license/	—	Folder	—	—
<input type="checkbox"/>	software/	—	Folder	—	—

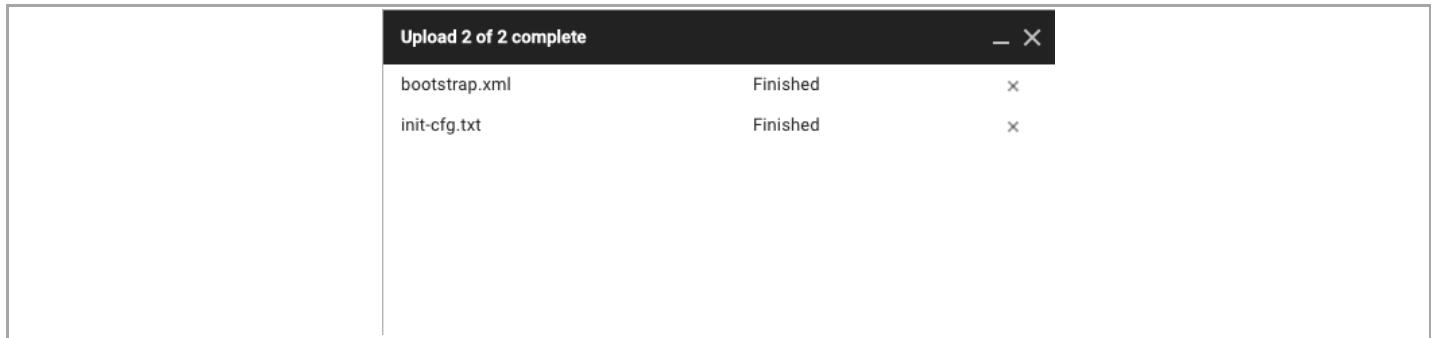
Download the following files using the links provided and save the files in a known location:

<https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/GKE-LB-Sandwich/bootstrap.zip>

Now click on the **config** folder in the console and click **UPLOAD FILES**:



Select the two files (bootstrap.xml and init-cft.txt) downloaded previously and click **Open**:



**NOTE:** All four folders must be created for the bootstrapping process to occur.

However, all folders DO NOT need to contain files.

**NOTE:** Please create the folders using the GUI or GCP CLI console. Creating folders locally on your machine and uploading them may not work as expected.

## 4.7 Download the Terraform Template Files

Download and save all of the template files to a known location by selecting **Clone or download**:

<https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/GKE-LB-Sandwich/config.yaml>

<https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/GKE-LB-Sandwich/main.tf>

<https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/GKE-LB-Sandwich/variables.tf>

<https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/GKE-LB-Sandwich/output.tf>

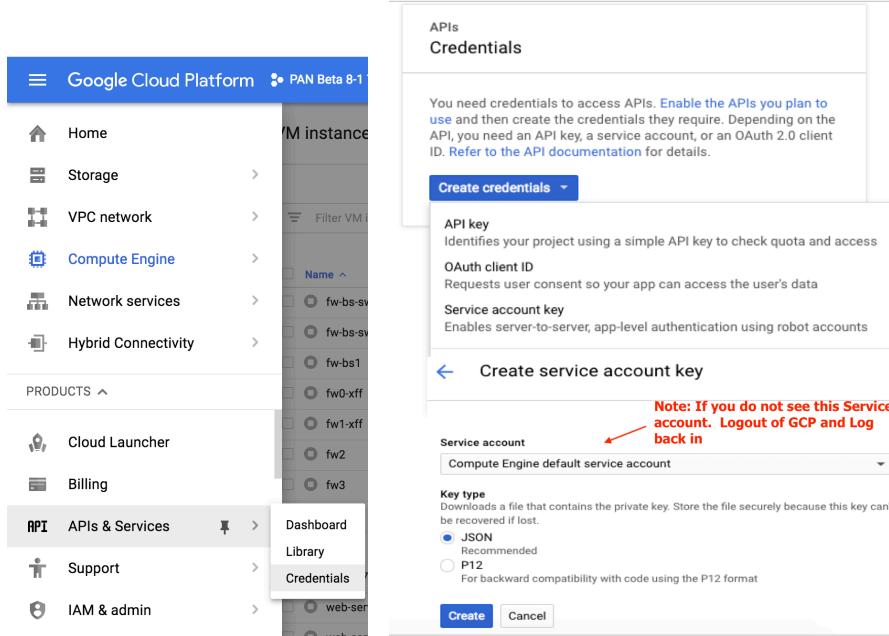
## 4.8 Gather Information and Update the Template File

Deploying the Terraform template in GCP requires modification of the template Main and Variable files to include deployment-specific information. The minimum required information is:

```
credentials = "${file("Your_Project_Credentials.json")}"
project  = "${Your_Project_ID}"
region   = "${Your_Project_Region}"
zone     = "${Your_Project_Zone}"
sshKey   = "${Your_Public_SSHKey}"
```

- The master\_auth password needs to be changed in the main.tf file

To create the credentials to access the APIs in JSON format. In GCP console go to (APIs & Services > Credentials > Create Credentials > Service Account Key), and download the file (client\_secrets.json). Put the .json credential file in your Terraform template folder.



Once the information has been gathered, update the Main and Variable files with the information. Save the Files.

## 5. Launch the Template

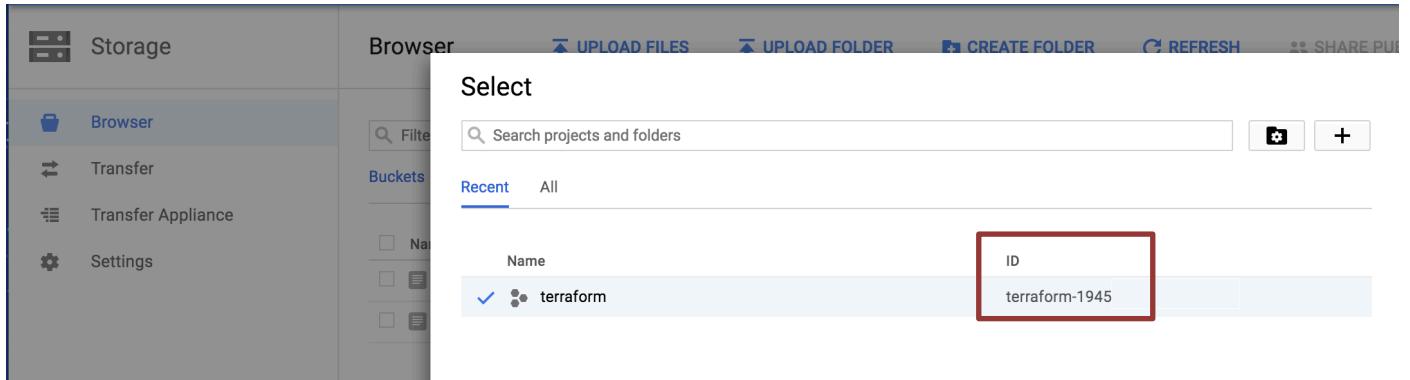
Navigate to a command shell navigate to the directory containing the downloaded template files:

Authenticate to the GCP environment from the command line with the command:

**\$ gcloud auth login**

- Copy/paste the link into a browser and select the account to authenticate if a browser does not automatically launch:
- Review the requested permissions and click **Allow**:
- Copy the one-time verification code:
- Paste it into the window to complete the authentication request (ignore the warning):

Get the Project ID:



Set the target project for template deployment via command line:

```
$ gcloud config set project my_Project_id
```

Run Terraform Commands:

Initiate template deployment using command “**terraform init**”. You should get a green Message indicating success.

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Once the terraform init has completed run the command “**terraform plan**”. If there are no errors you will see a number of components to be added. Scroll up to see detailed information on additions.

```
Plan: 12 to add, 0 to change, 0 to destroy.

-----
Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.
```

Now run the “**terraform apply**” command and say **yes** when prompt.

```
Plan: 12 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

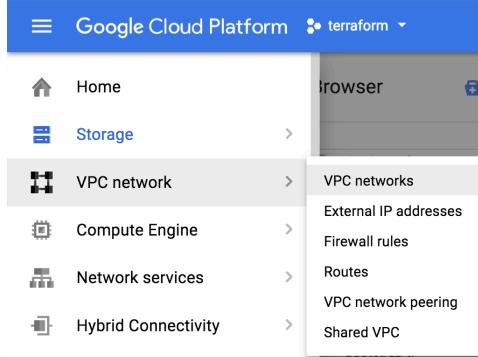
Enter a value: yes

Apply complete! Resources: 12 added, 0 changed, 0 destroyed.
```

If all goes well, Terraform will report success (“Apply Complete!” and no errors):

## 6. Review what was created

Let's review what the template has launched. The newly created networks can be viewed via **VPC Networks > VPC Network**:



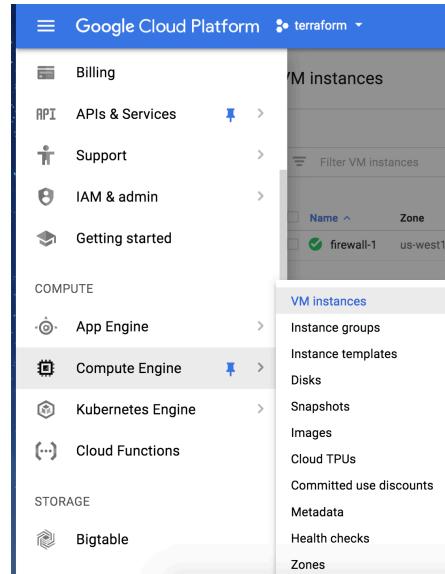
The template creates four networks: management-network, trust-network, and untrust-network.

VPC networks							
Name	Region	Subnets	Mode	IP addresses ranges	Gateways	Firewall Rules	Global dynamic routing
management	us-west1	management-sub	Custom	10.0.0.0/24	10.0.0.1	1	Off
trust	us-west1	trust-sub	Custom	10.0.2.0/24	10.0.2.1	1	Off
untrust	us-west1	untrust-sub	Custom	10.0.1.0/24	10.0.1.1	1	Off

**Note: A default network is automatically created when a GCP Project is instantiated. This default network can be ignored or deleted.**

Deployed VM-Series Firewall and GKE Cluster Nodes can be viewed by navigating to **Compute Engine > VM Instances**:

## Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich



High-level information regarding the deployed instances are available with the default view:

VM instances					
		CREATE INSTANCE		REFRESH	
		IMPORT VM		START STOP RESET	
Filter VM instances					
Name	Zone	Recommendation	Internal IP	External IP	Connect
firewall-1	us-west1-a		10.0.1.2	35.230.40.4	SSH
firewall-2	us-west1-b		10.0.1.3	35.227.148.205	SSH
gke-gkecluster-default-pool-54d503e3-fd3v	us-west1-c		10.0.2.4	35.227.182.201	SSH
gke-gkecluster-default-pool-7037fb42-t6lr	us-west1-a		10.0.2.5	35.230.87.182	SSH
gke-gkecluster-default-pool-a337a4d4-4s7t	us-west1-b		10.0.2.6	35.185.245.102	SSH

**Note: The GKE nodes were created with External IP Addresses. At the time of the development of this Terraform template sample Terraform cannot deploy private GKE Clusters. The External IP Addresses can manually be deleted by editing the nodes.**

Also note the order in which the networks are attached to the firewalls. Click on firewall-1 and scroll down to see the network order.

# Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich

VM instance details

[EDIT](#) [RESET](#) [CLONE](#) [STOP](#) [DELETE](#)

firewall-1

CPU utilization ▾

**CPU**

% CPU

Mar 6, 10:00 AM Mar 6, 10:05 AM Mar 6, 10:10 AM Mar 6, 10:15 AM Mar 6, 10:20 AM Mar 6, 10:25 AM

CPU: 6.472

**Remote access**

[SSH](#) ▾ [Connect to serial console](#) ▾

Enable connecting to serial ports ⓘ

**Logs**

Stackdriver Logging  
Serial port 1 (console)  
More

**Machine type**

n1-standard-4 (4 vCPUs, 15 GB memory)

**In use by**

fw-ig

**CPU platform**

Intel Broadwell

**Zone**

us-west1-a

**Labels**

None

**Creation time**

Mar 6, 2018, 9:44:20 AM

Network interfaces					
Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	IP forwarding
untrust	untrust-sub	10.0.1.2	—	35.230.119.56 (ephemeral)	On
management	management-sub	10.0.0.2	—	35.185.216.43 (ephemeral)	
trust	trust-sub	10.0.2.2	—	None	

Public DNS PTR Record

None

Network tags

None

**NOTE: The untrust network is first. The GCP Load Balancers only communicate with the lowest numbered interface on a VM. During the bootstrap phase of deployment, the init-config.txt told the VM-Series firewall to perform a management interface swap. Therefore, we must have the GCP networks in this order.**

## Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich

The screenshot shows the Google Cloud Platform interface for managing VPC networks. The left sidebar has 'VPC network' selected. A sub-menu for 'Routes' is open, showing options like 'Routes', 'VPC network peering', and 'Shared VPC'. Below this, the main content area displays a table of routes:

Name	Destination IP ranges	Priority	Instance tags	Next hop	Network
default-route-354346117ff5059a	10.0.0.0/24	1000	None	Virtual network	management
default-route-4ec4d794bfa971a0	10.0.2.0/24	1000	None	Virtual network	trust
default-route-6726ace17905c873	0.0.0.0/0	1000	None	Default internet gateway	untrust
default-route-7832e335985081ec	10.0.1.0/24	1000	None	Virtual network	untrust
default-route-85c1ffd65b473093	0.0.0.0/0	1000	None	Default internet gateway	trust
default-route-dde47d271469593f	0.0.0.0/0	1000	None	Default internet gateway	management
trust-route	0.0.0.0/0	100	None	firewall-1 (Zone us-west1-a)	trust

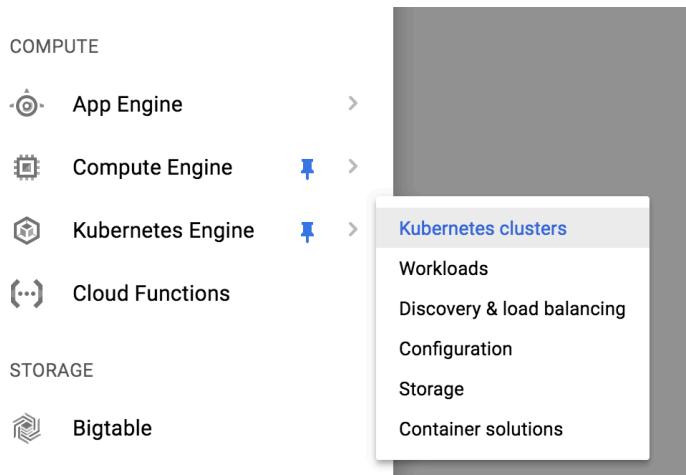
Check your newly deployed External Load Balancer by navigating to Network Services then select Load Balancing.

The screenshot shows the Google Cloud Platform interface for managing network services. The left sidebar has 'Network services' selected. A sub-menu for 'Load balancing' is open, showing options like 'Cloud DNS', 'Cloud CDN', and 'Networking solutions'. Below this, the main content area displays a table of networking solutions:

Name
default-route-354346117ff5059a

**NOTE: You should see one Load Balancer configured (HTTP-ELB) showing that the backend service is unhealthy. The Internal GKE LB will be deployed via the config.yaml file. You will create NAT rules on the VM-Series Firewalls to achieve a healthy state for the HTTP-ELB backend services in the next session.**

The GKE Cluster will have also deployed. That can be seen by clicking on Kubernetes Engine then selecting Kubernetes Cluster.



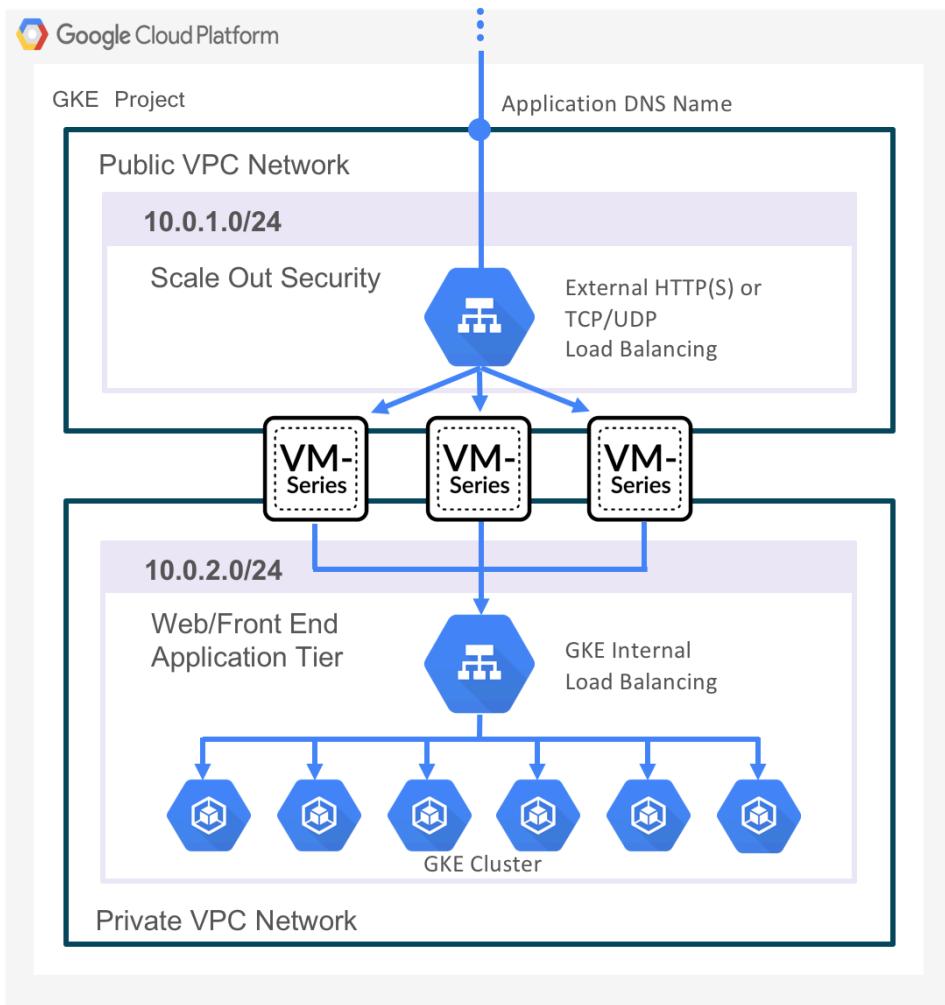
You will see your GKE Cluster Healthy with a Green checkmark. Note the Connect button to the right of your GKE Cluster. We will use this to connect to the GKE Cluster and complete a few more setup steps in the next section.

A screenshot of the 'Kubernetes clusters' page. At the top, there are buttons for 'CREATE CLUSTER', 'REFRESH', and 'DELETE'. Below is a search bar labeled 'Filter by label or name'. A table lists a single cluster:

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels
<input checked="" type="checkbox"/> gkecluster	us-west1-a	3	3 vCPUs	1.80 GB		

The 'Connect' button, located to the right of the cluster row, is highlighted with a red box.

All of this matches the topology shown previously:



## 7. Container Deployment

This deployment is based on the google hello-server quick-start guide. Located here.

<https://cloud.google.com/kubernetes-engine/docs/quickstart>

Once the Terraform deployment is complete with no errors and you have reviewed the items build in your GCP Project complete the following steps.

To connect to the GKE Cluster click the connect button then the Copy/Paste option to copy the login string to your clipboard. Now paste the string into the GCP Cloud Shell Console.

# Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich

## Connect to the cluster

You can connect to your cluster via command-line or using a dashboard.

### Command-line access

Configure `kubectl` command line access by running the following command:

```
$ gcloud container clusters get-credentials gkecluster --zone us-west1-a --project terraform-1945
```

[Copy](#)

[Run in Cloud Shell](#)

### Cloud Console dashboard

You can view the workloads running in your cluster in the Cloud Console [Workloads dashboard](#).

[Open workloads dashboard](#)

```
32@terraform-1945 ~$  
32@terraform-1945 ~$ gcloud container clusters get-credentials gkecluster --zone us-west1-a --project terraform-1945  
Fetching cluster endpoint and auth data.  
kubeconfig entry generated for gkecluster.
```

Once connected we will create the deployment via GCP Cloud Shell Console. Paste the following command into your GCP Cloud Shell Console and select enter. A response of <deployment “hello-server” created> will confirm the command was successful.

```
$ kubectl run hello-server --image gcr.io/google-samples/hello-app:1.0 --port 8080
```

```
32@terraform-1945 :~$  
32@terraform-1945 :~$ kubectl run hello-server --image gcr.io/google-samples/hello-app:1.0 --port 8080  
deployment "hello-server" created  
32@terraform-1945 :~$
```

You will also see the hello-server now under Workloads under the Kubernetes Engine section.

Name	Status	Type	Pods	Namespace	Cluster
hello-server	OK	Deployment	1/1	default	gkecluster

Now expose your service by creating the Internal Load Balancer for the GKE Cluster by launching the Config.yaml files by running the following command. If you have the file on your local machine and you have GCP SDK installed you can run the command from your local machine. A confirmation of <Service “hello-server” created>

```
kubectl apply -f config.yaml
```

```
1TG8WL:terraform $ kubectl apply -f config.yaml  
service "hello-server" created
```

Once the hello-server Service is created you will see an entry in the Discovery & load balancing section under Kubernetes Engine.

The screenshot shows the Kubernetes Engine interface under the 'Discovery & load balancing' section. A table lists services, with the 'Endpoints' column for the 'hello-server' service highlighted with a red box.

Name	Status	Service Type	Endpoints	Pods	Namespace	Cluster
hello-server	Ok	Load balancer	10.0.2.7:9000	0 / 1	default	gkecluster

Note the Endpoints area. You will see the internal IP address of your Internal Load Balancer and the port it is listening on. This information can also be seen by running the command

`$ kubectl get service`

The output will look like this.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
hello-server	LoadBalancer	10.7.242.21	10.0.2.7	9000:31154/TCP	13m
kubernetes	ClusterIP	10.7.240.1	<none>	443/TCP	1h

In the next section we will modify the NAT settings in the VM-Series firewall with the Endpoints information.

## 8. Access the firewall

**NOTE:** Bootstrapping a VM-Series firewall takes approximately 9 minutes. Be patient ☺ Once the template has been deployed successfully, it may be a while before the VM-Series firewall is up and you are able to log into the VM-Series firewall by browsing to the Management public IP Address. Recall we swapped the Management interface so you will need to click on the VM Series to get the Public IP address.

# Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich

[VM instance details](#) [EDIT](#) [RESET](#) [CLONE](#) [STOP](#) [DELETE](#)

**firewall-1**

CPU utilization ▾

**CPU**

% CPU

The chart displays CPU usage in percent over a five-minute period. The Y-axis ranges from 2% to 6%. The X-axis shows times from Mar 6, 10:30 AM to Mar 6, 10:50 AM. The usage fluctuates between approximately 5.5% and 6.5%.

Mar 6, 10:30 AM Mar 6, 10:35 AM Mar 6, 10:40 AM Mar 6, 10:45 AM Mar 6, 10:50 AM

CPU: 5.449

**Remote access**

SSH ▾ Connect to serial console ▾

Enable connecting to serial ports ⓘ

**Logs**

Stackdriver Logging

Serial port 1 (console)

More

**Machine type**

n1-standard-4 (4 vCPUs, 15 GB memory)

**In use by**

fw-ig

**CPU platform**

Intel Broadwell

**Zone**

us-west1-a

**Labels**

None

**Creation time**

Mar 6, 2018, 9:44:20 AM

**Network interfaces**

Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	IP forwarding
untrust	untrust-sub	10.0.1.2	—	35.230.119.56 (ephemeral)	On
management	management-sub	10.0.0.2	—	35.185.216.43 (ephemeral)	
trust	trust-sub	10.0.2.2	—	None	

**Public DNS PTR Record**

None

**Network tags**

None

You should now be able to browse to the VM-Series firewall and login using the **username: paloalto** and **password: PaloAlt0@123**

# Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich

The screenshot shows the Palo Alto Networks Management Console interface. The top navigation bar includes tabs for Dashboard, ACC, Monitor, Policies, Objects, Network, and Device. The main content area has three tabs open:

- General Information:** Displays device details such as Device Name (sample-cft-fw), MGT IP Address (10.5.0.4 (DHCP)), and various software and hardware versions.
- System Logs:** Shows a log of system events, including failed password attempts and user logins.
- System Resources:** Monitors CPU usage (Management CPU at 1%, Data Plane CPU at 0%), and session counts (3 / 819200).

At the bottom, there are links for 'Logout' and 'Last Login Time: 02/27/2018 12:31:46', along with a 'Commit' button and a search bar.

Here are the interfaces to zone mappings.

The screenshot shows the Palo Alto Networks Management Console with the Network tab selected. On the left, a sidebar lists various network-related profiles and global protect settings. The main pane displays a table of interfaces:

Interface	Interface Type	Management Profile	Link State	IP Address	Virtual Router	Tag	VLAN / Virtual-Wire	Security Zone	Features	Comment
ethernet1/1	Layer3	mgmt-untrust	Up	Dynamic-DHCP Client	default	Untagged	none	untrust		
ethernet1/2	Layer3	mgmt-trust	Up	Dynamic-DHCP Client	default	Untagged	none	trust		
ethernet1/3			Up	none	none	Untagged	none	none		
ethernet1/4			Up	none	none	Untagged	none	none		
ethernet1/5			Up	none	none	Untagged	none	none		
ethernet1/6			Up	none	none	Untagged	none	none		
ethernet1/7			Up	none	none	Untagged	none	none		

In the policies tab you can review the security policies:

## Palo Alto Networks GCP Terraform Template Deployment Guide GKE LB Sandwich

Name	Tags	Type	Source			Destination			Rule Usage			Application	Service	Action
			Zone	Address	User	Zone	Address	Hit Count	Last Hit	First Hit				
1 allow-all-out	none	universal	trust	any	any	any	untrust	any	38	2018-03-06 10:38:43	2018-03-06 10:27:57	any	any	Allow
2 allow-web-ssh-in	none	universal	untrust	any	any	any	trust	any	36829	2018-03-06 18:26:08	2018-03-06 10:00:05	ssh	application-default	Allow
3 intrazone-default	none	intrazone	any	any	any	any	(intrazone)	any	347	2018-03-06 18:22:29	2018-03-06 12:03:44	any	any	Allow
4 interzone-default	none	interzone	any	any	any	any	any	any	0	-	-	any	any	Deny

Under the policies tab select NAT on the left side. We need to have a health check from the ELB through to the backend Nodes of the GKE ILB. In order for the ELB health checks to flow through the VM-Series firewall then to and through the GKE ILB to the backend Nodes we need to add the Terraform outputs to the highlighted objects in the ELB Health Check NAT statement in the next step. There is also a rule to NAT all traffic out from web servers to the outside world.

Name	Tags	Original Packet							Translated Packet			Hit Count
		Source Zone	Destination Zone	Destination Interface	Source Address	Destination Address	Service	Source Translation	Destination Translation			
1 ELB Health Check	none	untrust	untrust	any	any	Untrust IP	service-ssh	dynamic-ip-and-port	destination-translation	0		
2 nat-out	none	trust	untrust	any	any	any	any	dynamic-ip-and-port	address: GKE-ILB port: 9000	0		
3 nat-web-ssh	none	untrust	untrust	any	any	Untrust IP	any	dynamic-ip-and-port	destination-translation	0		

Make note of the outputs from your Terraform for the Untrust IP of the VM-Series firewall. The GKE-ILB IP address can be found under Endpoints. Your outputs may be different.

```
firewall-untrust-ips-for-nat-healthcheck = [
  10.0.1.2,
  10.0.1.3,
  10.0.1.4
]
internal-lb-ip-for-nat-healthcheck = 10.0.2.9
```

Name	Status	Service Type	Endpoints	Pods	Namespace	Cluster
hello-server	Ok	Load balancer	10.0.2.7:9000	0 / 1	default	gkecluster

Click on the Objects Tab and up the Untrust IP and GKE-ILB on each VM-Series firewall.

Name	Location	Type	Address	Tags
GKE-ILB		IP Netmask	10.0.2.9	
Untrust IP		IP Netmask	10.0.1.2	
Web Server IP		IP Netmask	10.0.2.4	

Repeat this step for all of the Firewalls and **COMMIT** these changes on the VM-Series Firewalls Then go back to your GCP Load Balancers and check the Health Status.

**Frontend**

Protocol	IP:Port	Certificate
HTTP	35.227.197.98:80	—

**Host and path rules**

Hosts	Paths	Backend
All unmatched (default)	All unmatched (default)	fw-backend

**Backend**

**Backend services**

**1. fw-backend**

Endpoint protocol: **HTTP** Named port: **http** Timeout: **30 seconds** Health check: **elb-health-check** Session affinity: **None** Cloud CDN: **disabled**

**Advanced configurations**

Instance group	Zone	Healthy	Autoscaling	Balancing mode	Capacity
fw-ig	us-west1-a	3 / 3	Off	Max CPU: 80%	100%

## 9. Access the Webservers via ELB

Open a browser and browse to the IP address of the ELB. The IP of the ELB can be found under load balancers then expand the ELB. You will get the following screen.

**Load balancing**

- Cloud DNS
- Cloud CDN

**Load balancers**

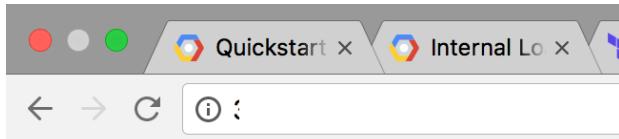
**Load balancer**

Protocol
HTTP

**Host and path rules**

Hosts	Paths	Backend
All unmatched (default)	All unmatched (default)	fw-backend

**Backend**



You have now successfully deployed a Terraform template with a VM-Series firewall in GCP.

## 10. Cleanup

### 10.1 Delete the deployment

Once done, cleanup as follows:

- If you licensed the VM-Series firewall perform the De-License function.
  - [https://www.paloaltonetworks.com/documentation/71/virtualization/virtualization/license-the-vm-series-firewall/deactivate-vm#\\_87329](https://www.paloaltonetworks.com/documentation/71/virtualization/virtualization/license-the-vm-series-firewall/deactivate-vm#_87329)
- Delete the GKE Load Balancer service with the following command from the Cloud Shell.
  - kubectl delete service hello-server
- Delete the GKE hello-server Workloads with the following command from the Cloud Shell.
  - kubectl delete deployment hello-server
- Last step from the Terraform CLI, issue the command “**terraform destroy**”
  - This will delete all the resources created via the Terraform template.

## 11. Conclusion

You have successfully deployed a Terraform template in GCP and demonstrated how the Palo Alto Next Generation VM-Series firewall can be deployed via Terraform automation to not only secure traffic throughout your GCP Project, but throughout your Enterprise Google Cloud Infrastructure.

## Appendix A

### Troubleshooting tips

#### **1. Bootstrapping not working**

If the VM-Series firewall is up and you are able to access the login page, but unable to login using the username/password: paloalto/PaloAlt0@123, then chances are bootstrapping has failed. There could be several reasons:

##### *a. Corrupt configuration files*

Please ensure that the bootstrap.xml and init-cft.txt files mentioned in [Section 4.6](#) are not corrupted.

##### *b. Incorrect bootstrap bucket-name*

Another reason for bootstrapping to fail is that the bootstrap bucket name (Parameter: bootstrapbucket) was incorrectly entered in the template file. Please make sure the bucket name created in [Section 4.6](#) is mentioned when launching the template.

#### **2. Container Deployment failed**

##### *a. Corrupt configuration files*

Delete the Service and Workload for the GKE Cluster. Step through the install process of that section again.