VM-Series for GCP

GCP Terraform Template Deployment Guide

Deploys an External Load Balancer multiple VM-Series NGFW Internal Load Balancers and Web Servers in different zone. This deployment model is commonly referred to as a Load Balancer Sandwich.

https://www.paloaltonetworks.com

Last Update: May 2016

Table of Contents

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, and more. These templates are used for ease of deployment and are key to any cloud deployment model.

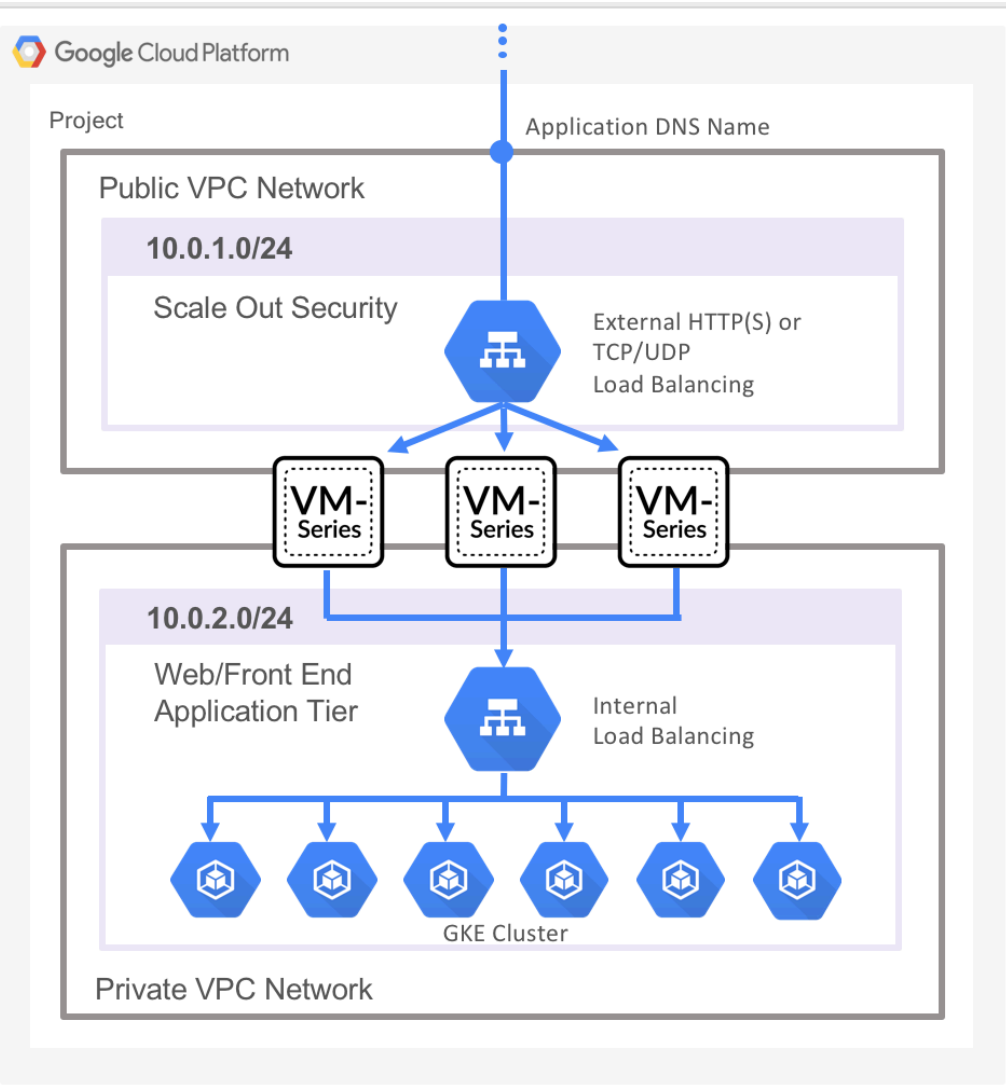
For more information on Templates refer to Google's documentation

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, multiple apache web server, multiple VM-Series firewall and the subnets, an HTTP ELB, and a TCP ILB. 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, 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
Apache Web Servers	f1-micro
VM Series Firewall	n1-standard-4

Note: There are costs associated with each machine type launched, 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.

4.2 Install the Google Cloud SDK

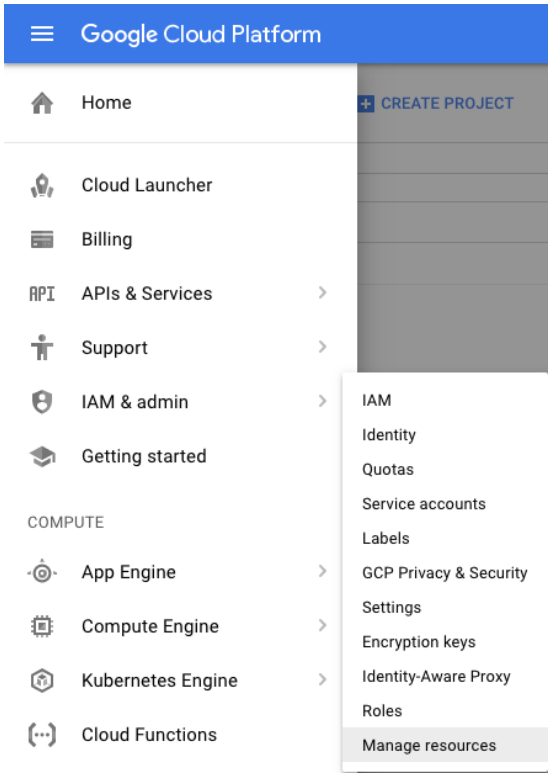
Template installations in GCP are performed from the CLI. Install the SDK/CLI by selecting the relevant platform from the following link and following the installation instructions:

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

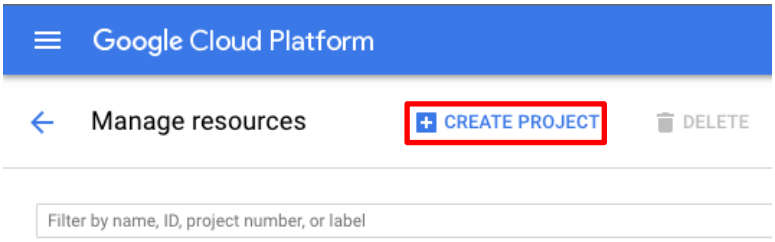
4.3 Accept the EULA (If Required)

4.4 Create a Project

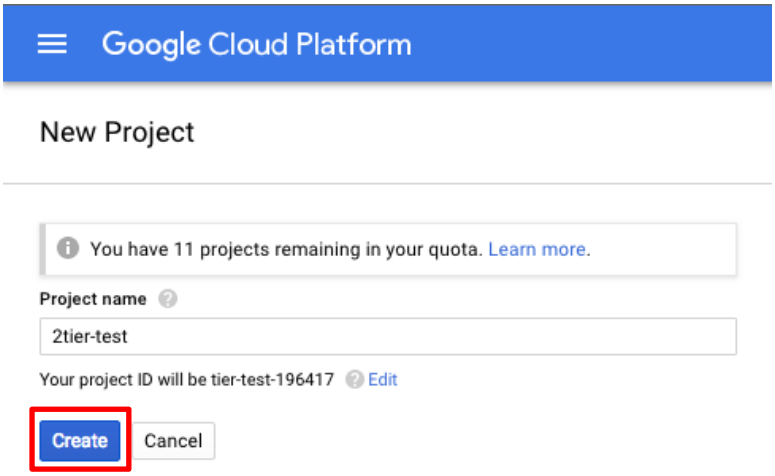
All GCP resources are deployed to a GCP Project. A GCP Project is an organizational boundary that separates users, resources, billing information, etc. A GCP Project is similar to an AWS VPC or an Azure Resource Group. By default, GCP will create a Project upon creation of an account. If that is not 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**:



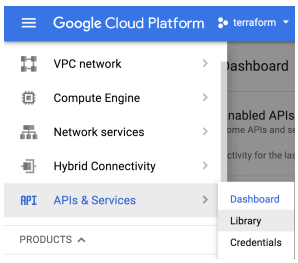
Specify a name for the project and click **Create**:



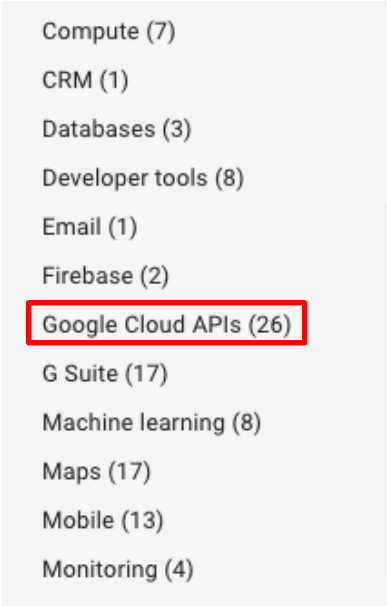
Note: GCP Project creation will take a few minutes.

4.5 Enable the API

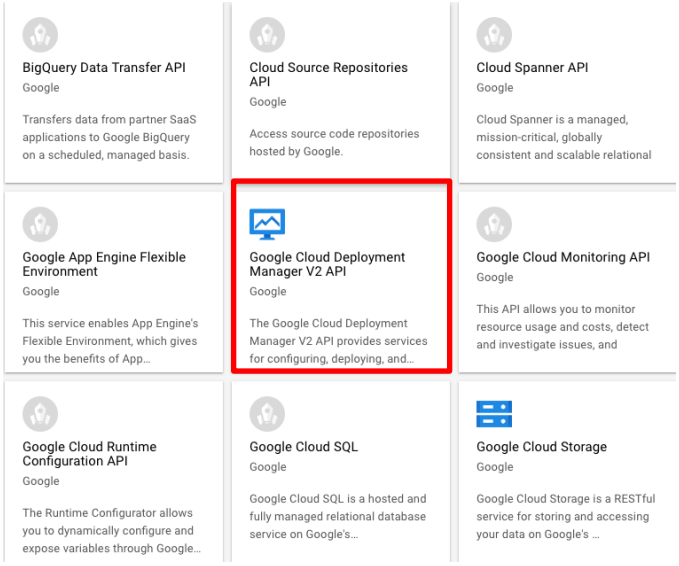
Deploying a template requires the API be enable on the project. Navigate to **APIs & Services > Library**:



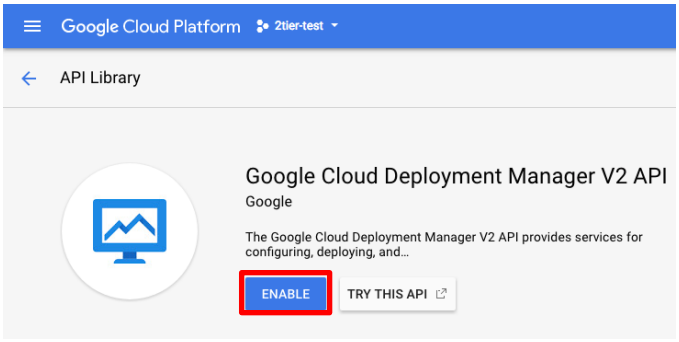
Select Google Cloud APIs on the left-hand-side:



Select Google **Cloud Deployment Manager V2 API**:



Select **Enable**:

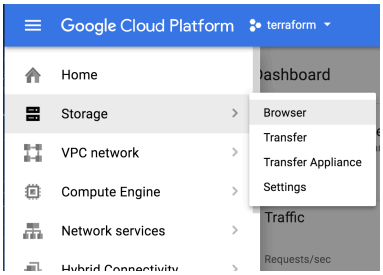


Note: Enabling the API for the project will take a few minutes to complete.

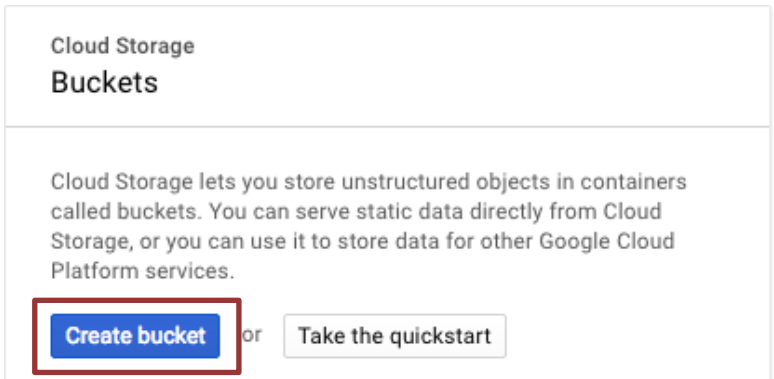
4.6 Create a Bootstrap Bucket

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.

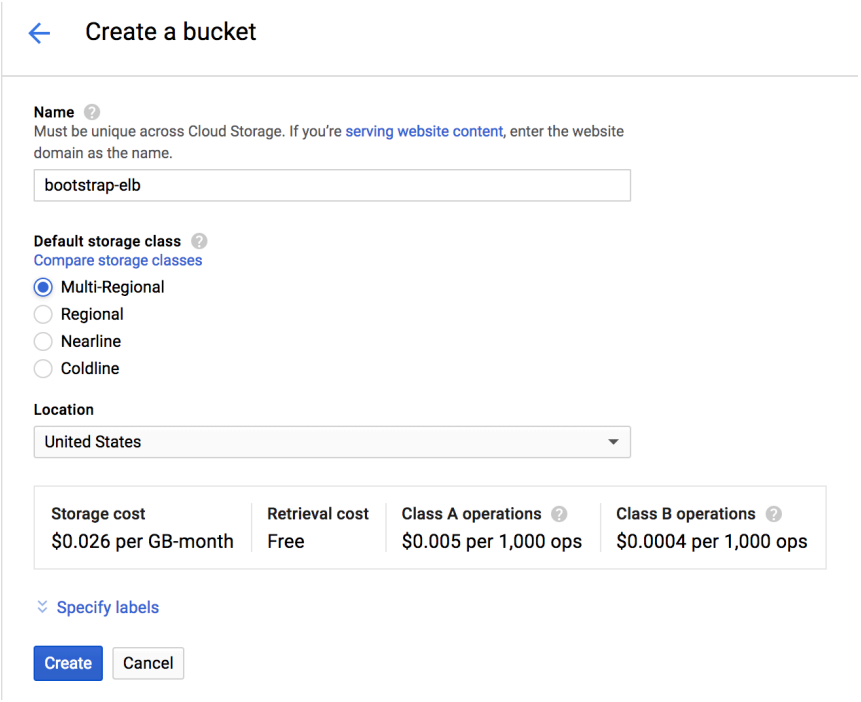
In order to create a Bootstrap bucket, navigate to **Storage > Browser**:



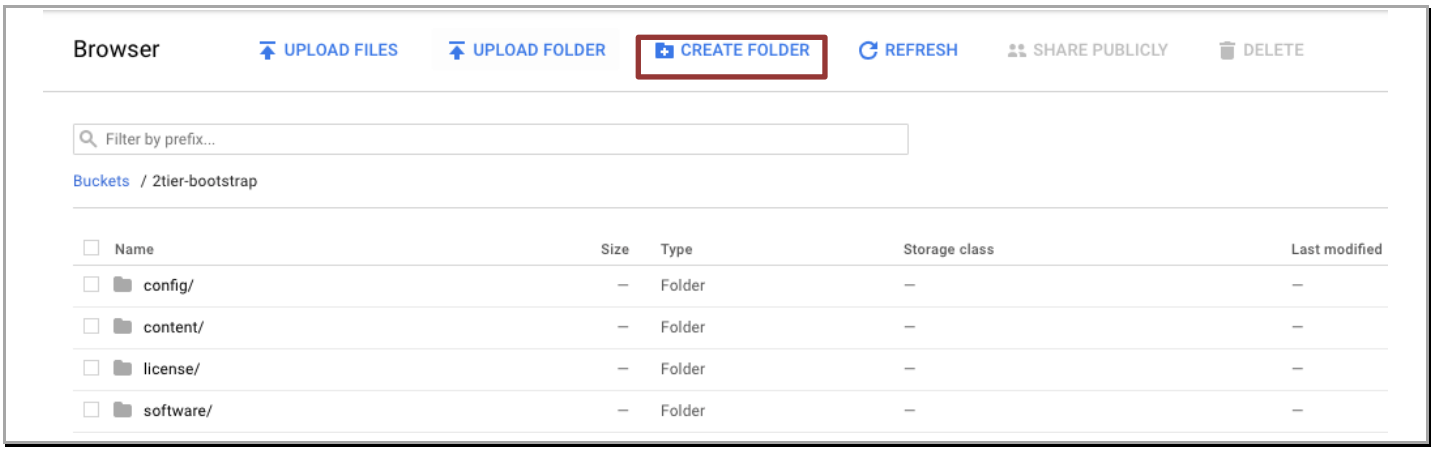
Click **Create Bucket**:



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



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**:

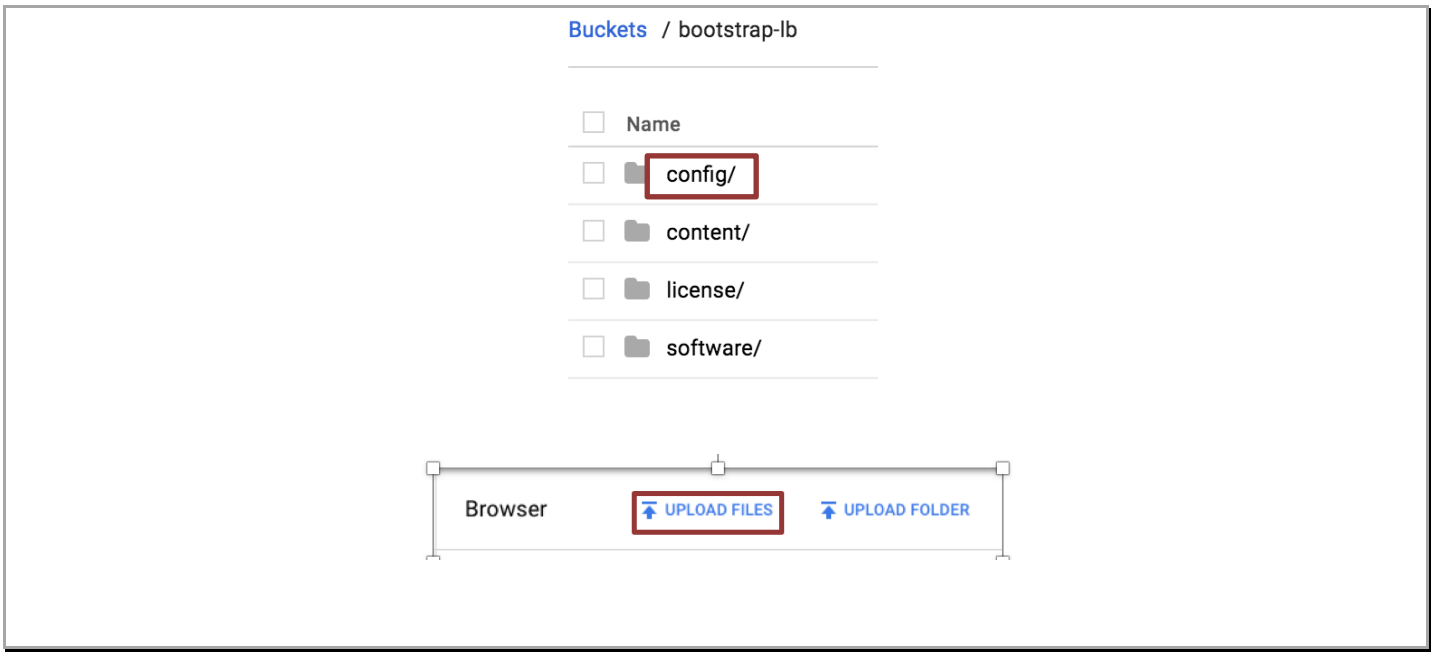


Download the following files using the links provided and save the files in a known location:

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

https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/LB-Sandwich/startup.sh

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/LB-Sandwich/Main.tf

https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/LB-Sandwich/Variables.tf

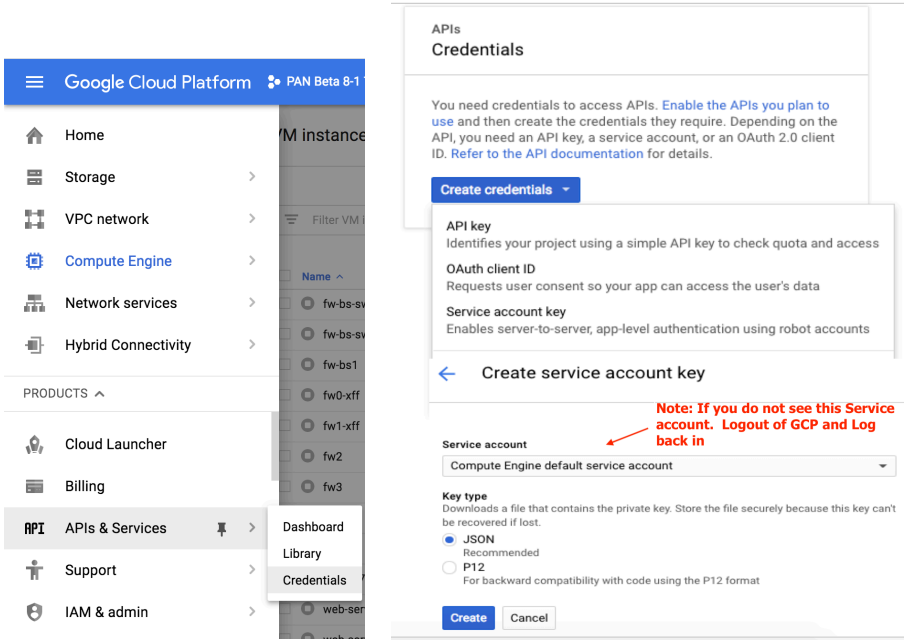
https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/LB-Sandwich/Output.tf

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")}"

metadata_ startup_script= "${file("location/startup.sh")}"

project     = "${Your_Project_ID}"

region      = "${Your_Project_Region}"

zone        = "${Your_Project_Zone}"

2nd zone        = "${Your_Zone2}"

sshKey      = "${Your_Public_SSHKey}"
```

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

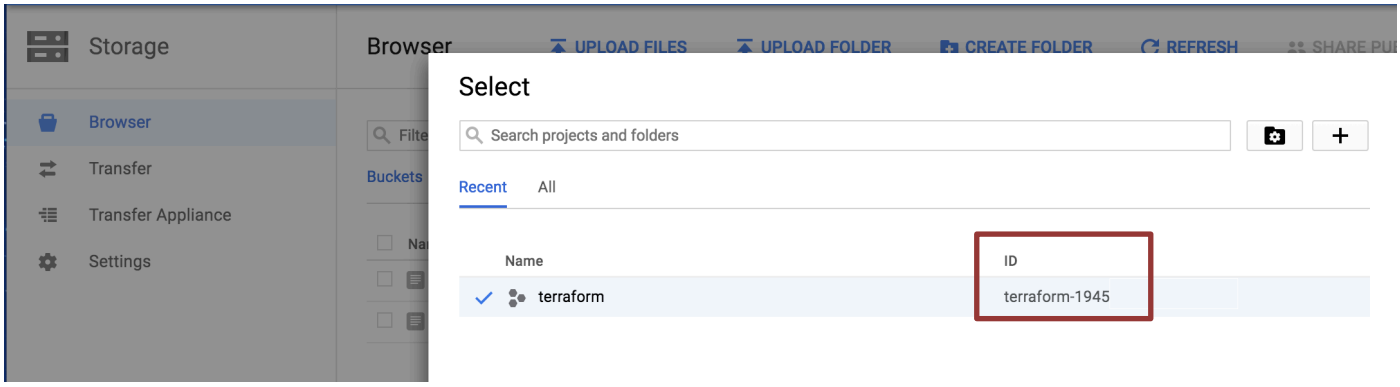
Navigate to a command shell navigate to the directory containing the downloaded template files:

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”.

```
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”.

```
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.
```

You will see if there are any errors and what terraform will be deploying. Now run the “terraform apply” command and say yes when prompt.

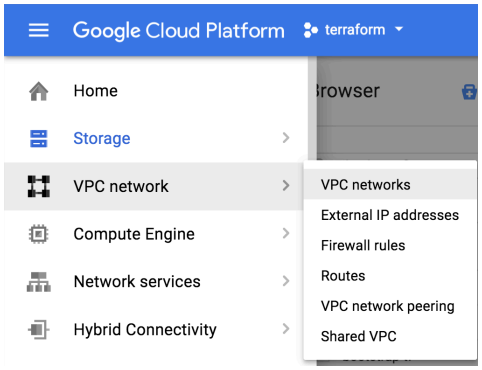
```
Plan: 12 to add, 0 to change, 0 to destroy.

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
```

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

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

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**:



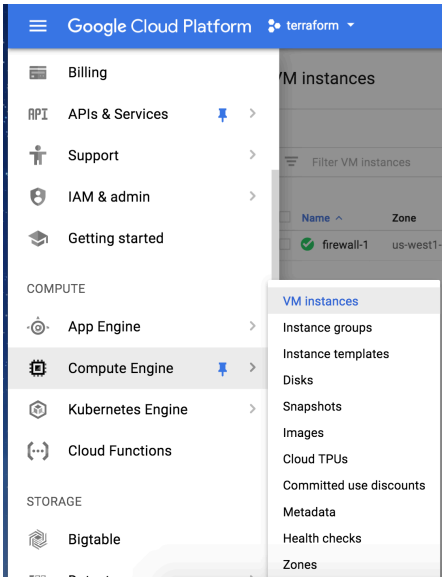
The template creates four networks: management-network, trust-network, and untrust-network.

VPC networks CREATE VPC NETWORK REFRESH

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

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

Deployed hosts can be viewed by navigating to **Compute Engine > VM Instances**:



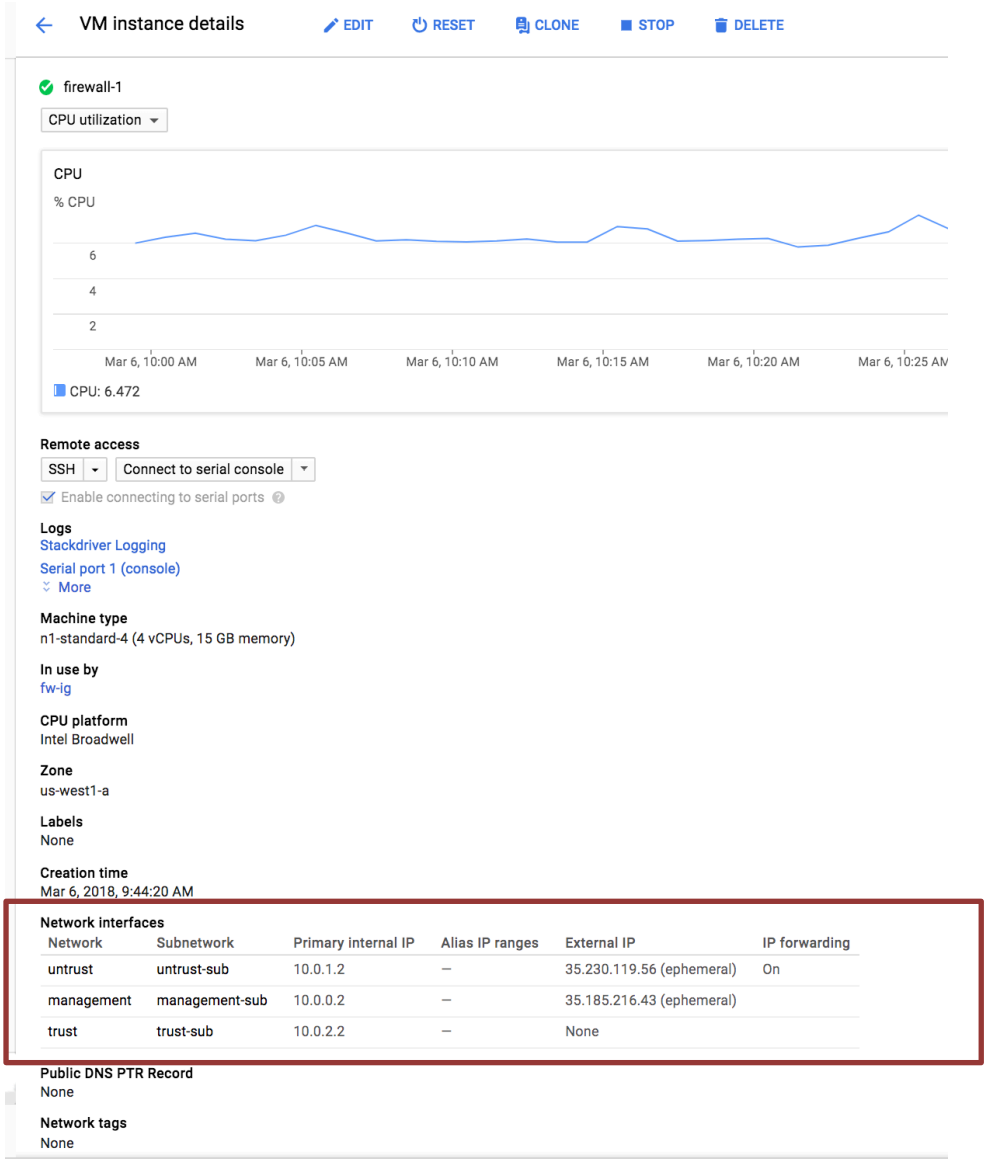
High-level information regarding the deployed instances are available with the default view:

VM instances CREATE INSTANCE IMPORT VM REFRESH

Filter VM instances

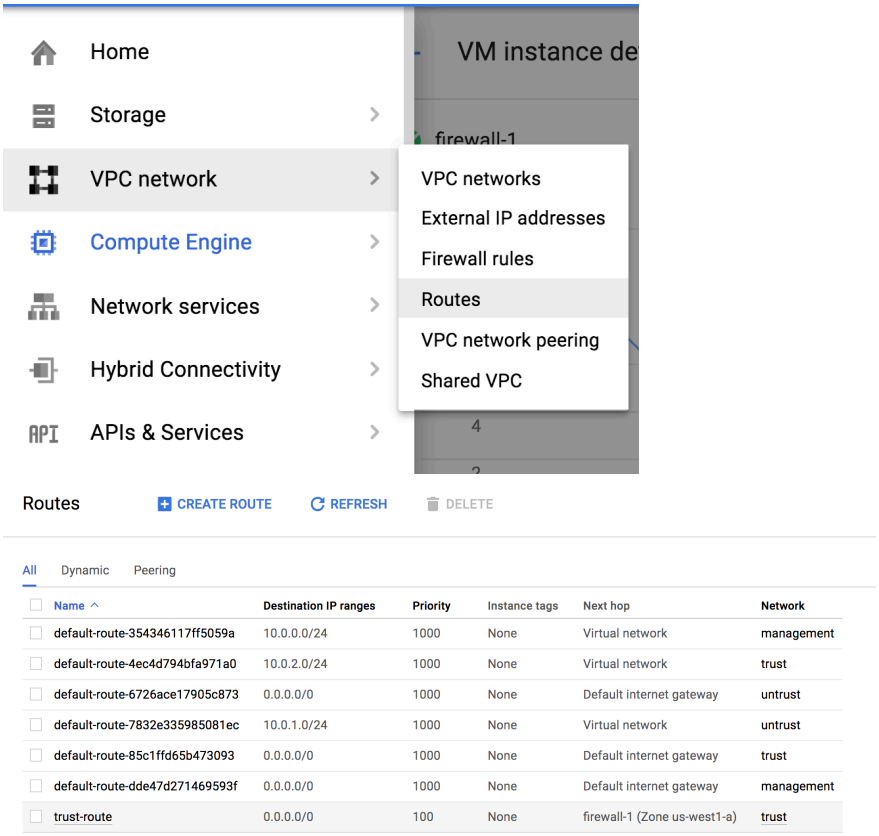
Name	Zone	Recommendation	Internal IP	External IP	Connect
firewall-1	us-west1-a		10.0.1.2	35.230.119.56	SSH
firewall-2	us-west1-a		10.0.1.3	35.230.69.250	SSH
firewall-3	us-west1-a		10.0.1.4	35.185.245.54	SSH
webserver-1	us-west1-a		10.0.2.3	None	SSH
webserver-2	us-west1-b		10.0.2.7	None	SSH
webserver-3	us-west1-a		10.0.2.6	None	SSH
webserver-4	us-west1-b		10.0.2.8	None	SSH

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.

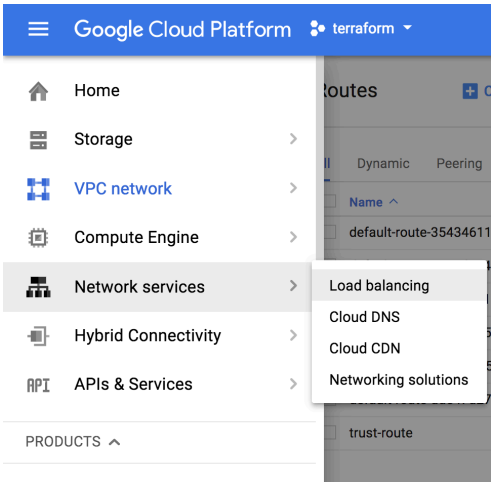


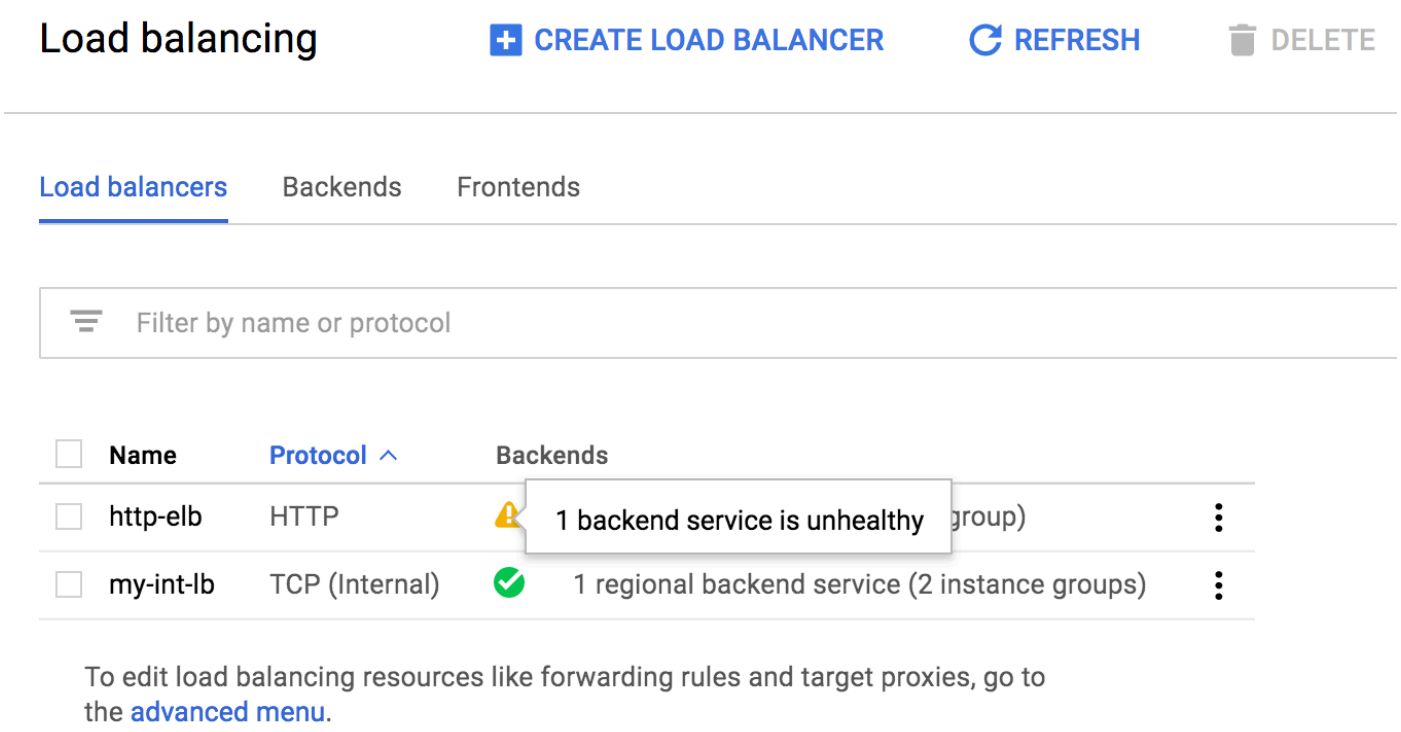
NOTE: The untrust network is first. The GCP Load Balancers only communicate with the lowest numbered interface on a VM. During the bootstrap phase of deployment the init-cfg.txt told the VM-Series firewall to perform a management interface swap. Therefore, we must have the GCP networks in this order.

In order for the webservers to download apache we have added a default route through the first VM-Series Firewall instance. Navigate to routes via the VPN Network dropdown. See below.



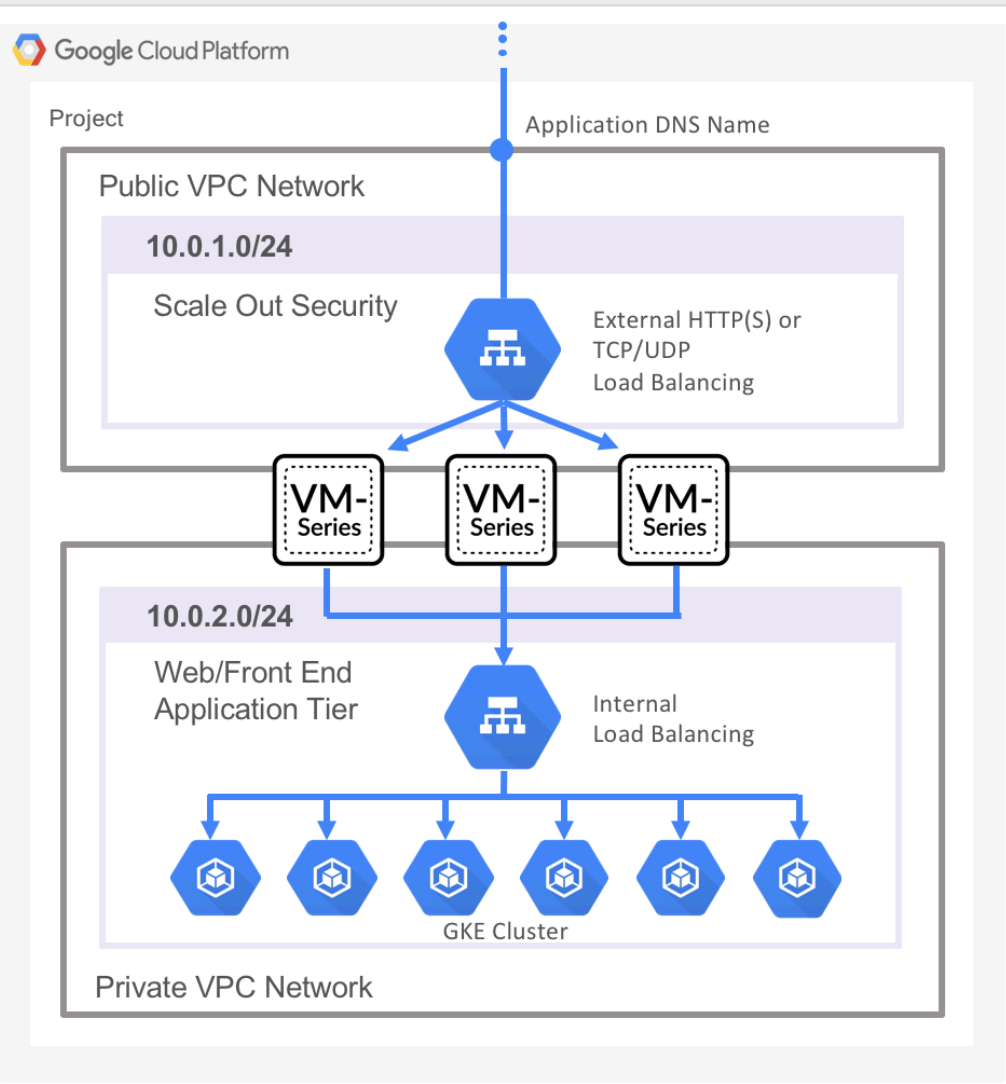
Lastly check your newly deployed Load Balancers by navigating to Network Services then select Load Balancing.





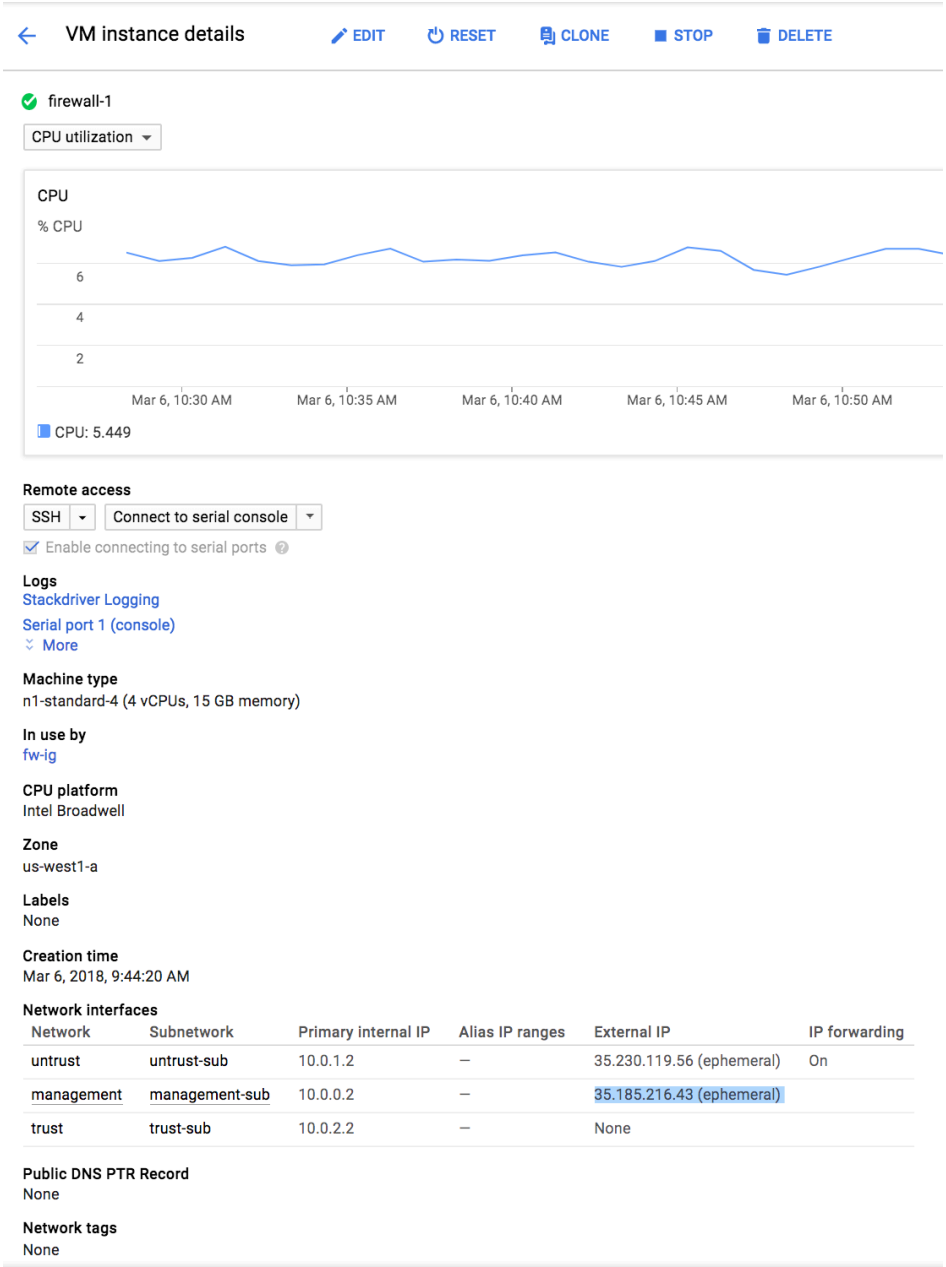
NOTE: You should see two load balancers configured with the HTTP-ELB showing that the backend service is unhealthy. The Internal TCP LB shows all backend services as healthy. We need create NAT rules on the VM-Series Firewall to achieve a healthy state for the HTTP-ELB backend services in the next session.

All of this matches the topology shown previously:

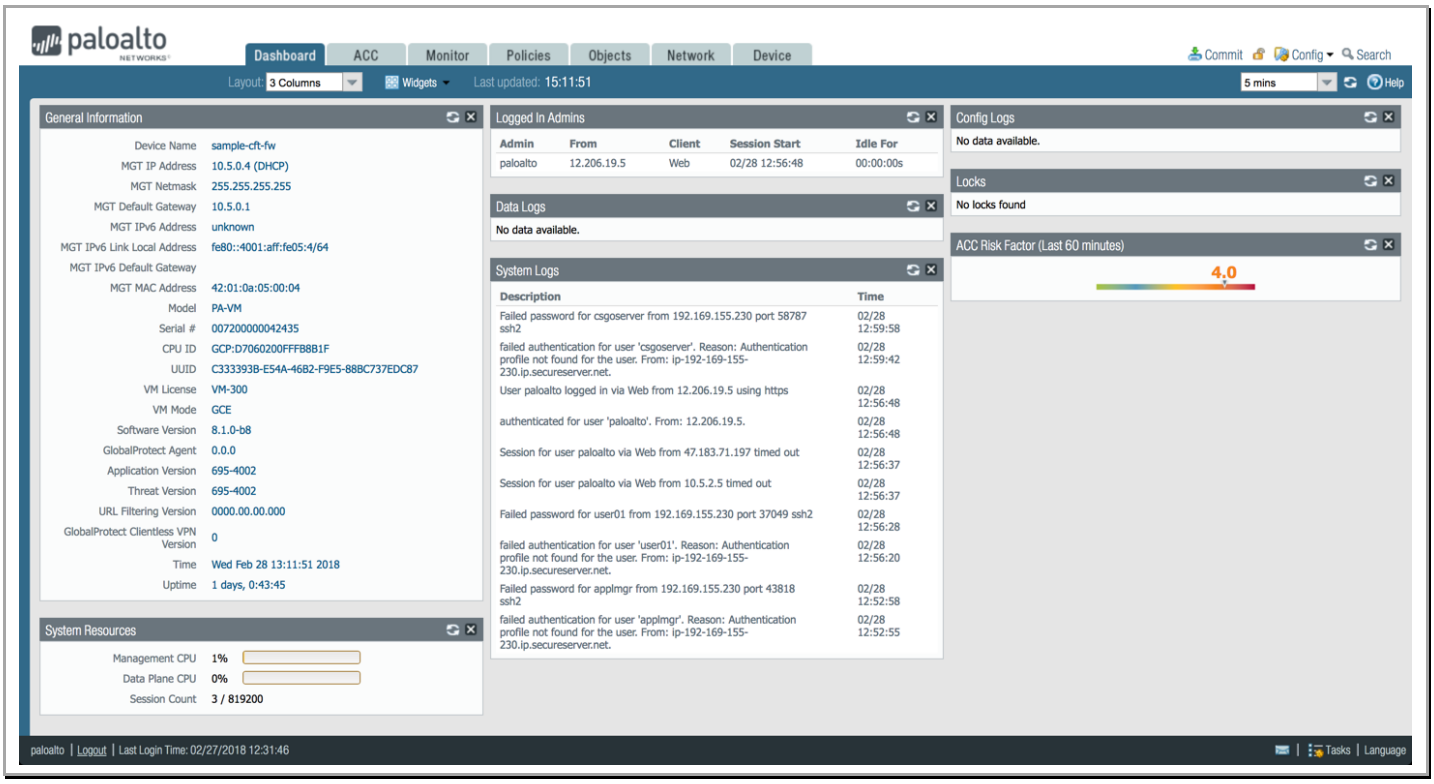


7. Access the firewall

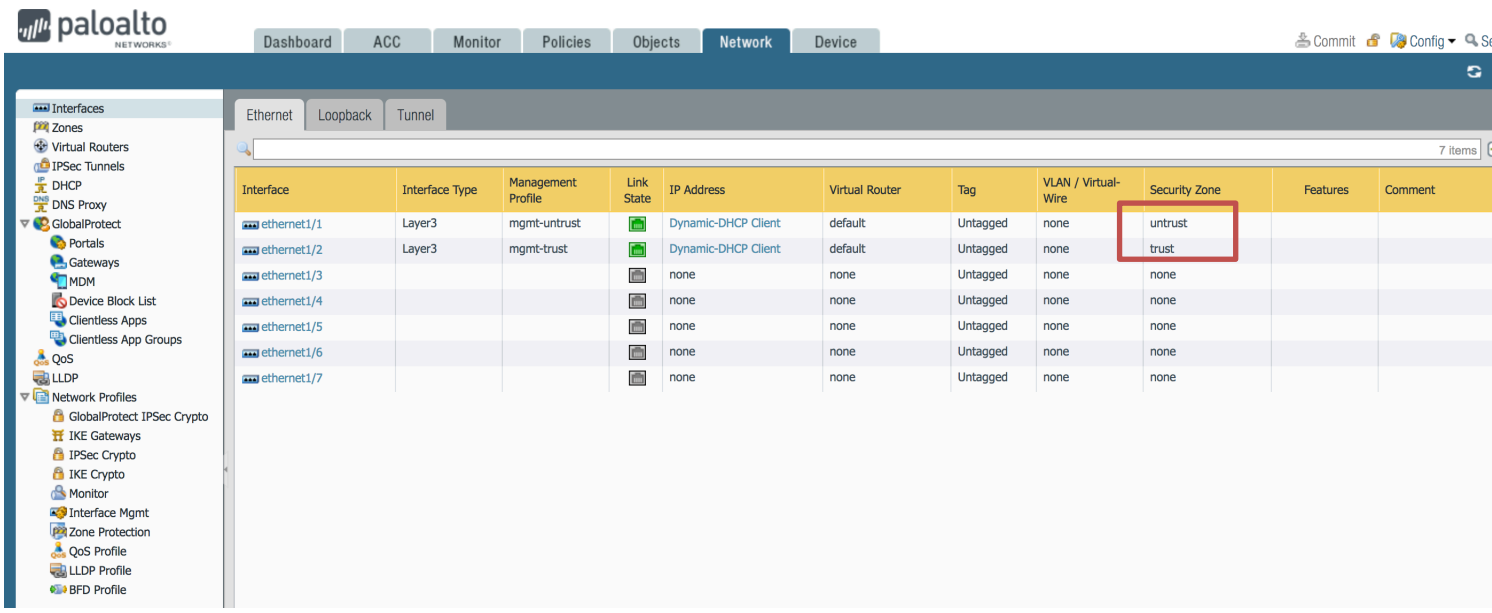
NOTE: Bootstrapping a VM-Series firewall takes approximately 9 minutes. Be patient☺ Once the template has been deployed successfully, it may be a while before the VM-Series firewall is up and you are able to log into the VM-Series firewall by browsing to the Management public IP Address. Recall we swapped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



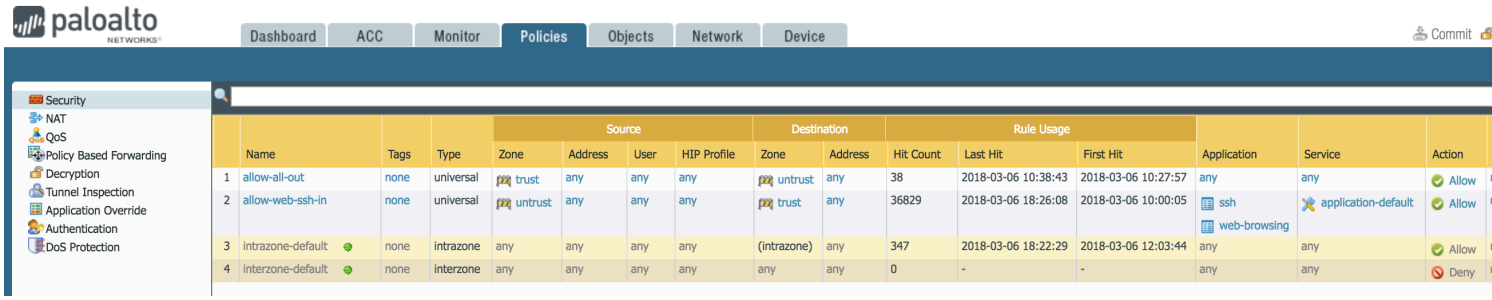
You should now be able to browse to the VM-Series firewall and login using the **username: paloalto** and password: **Pal0Alt0@123**



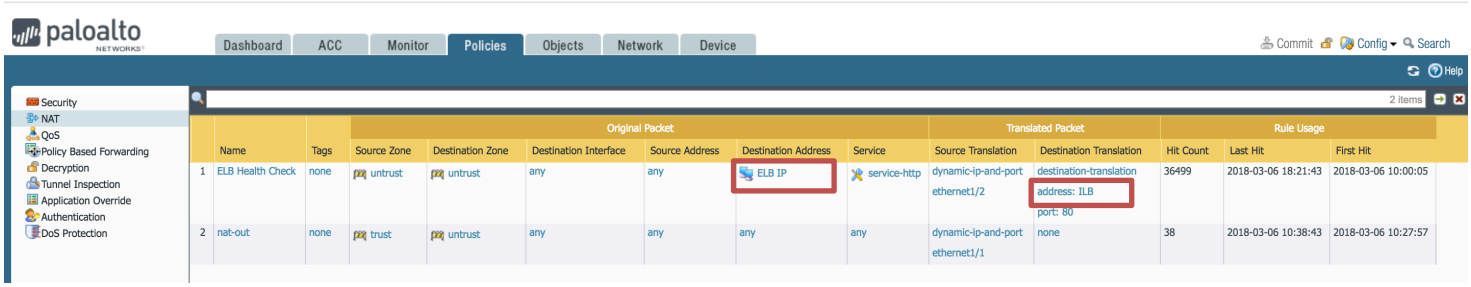
Here are the interfaces to zone mappings.



In the policies tab you can review the security policies:



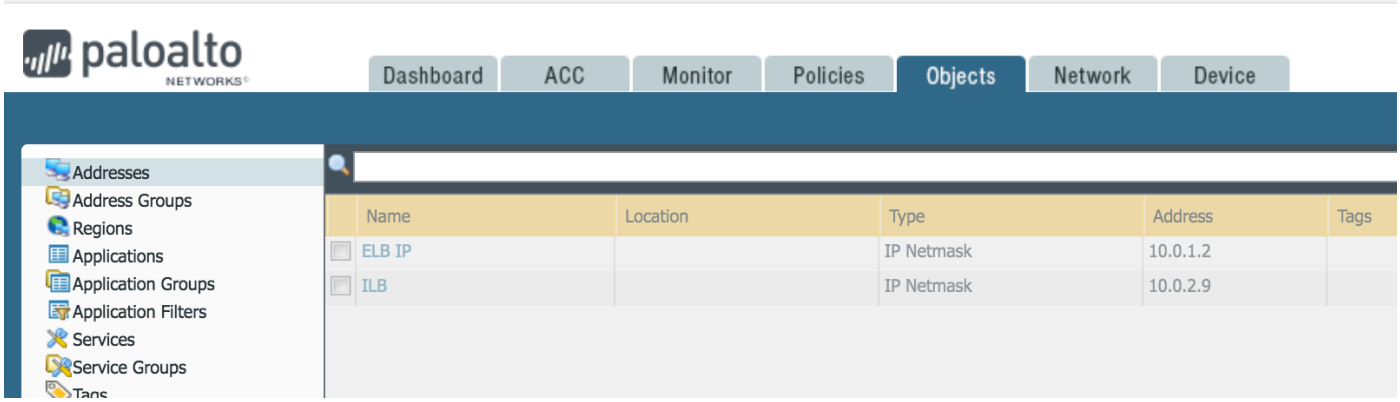
Under the polices tab select NAT on the left side. We need to have a health check from the ELB through to the backend web server of the ILB. In order for the ELB health checks to flow through the VM-Series firewall then to and through the ILB to the web servers we need to add the Terraform outputs to 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.



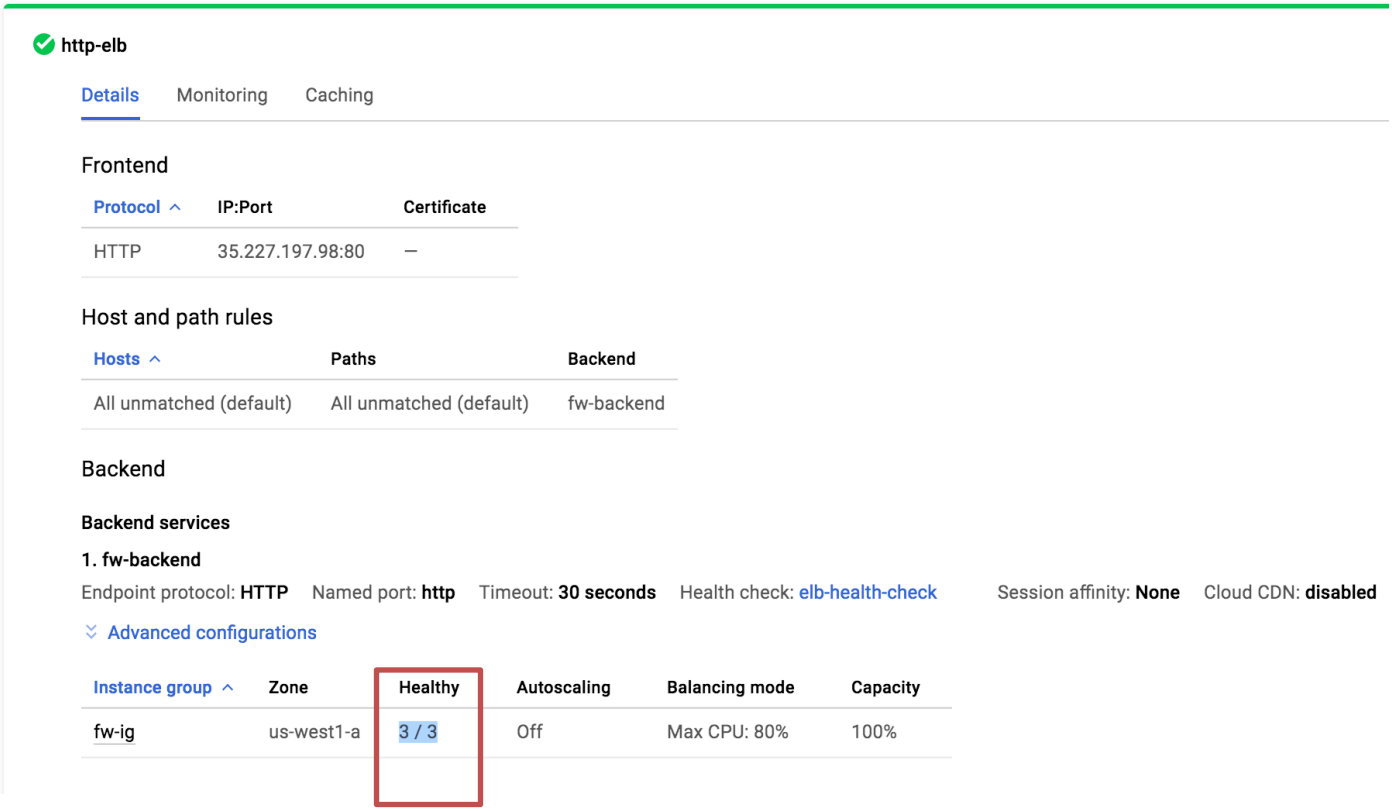
Make note of the outputs from your Terraform apply. Your outputs may be different.

```
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
```

Click on the Objects Tab and up the ELB IP and ILB on each VM-Series Firewall.

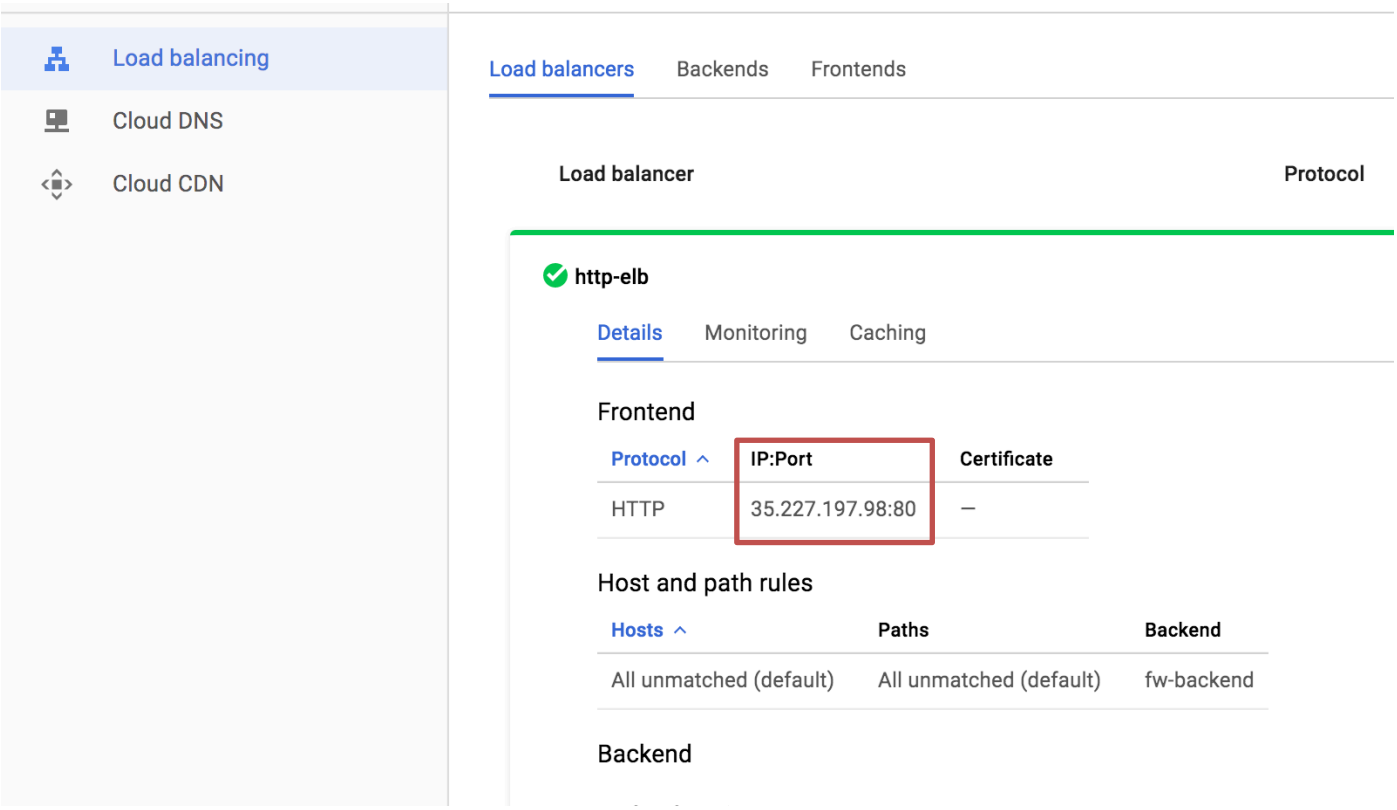


Repeat this step for all of the Firewalls and **COMMIT** these changes on the VM-Series Firewalls Then go back to your GCP Load Balancers and check the Health Status.



8. 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. Each Webserver will provide their name in the browser upon reloading the browser. This lets you know the Load Balancer is working.



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

9. Cleanup

9.1 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
- From the CLI, issue the command "terraform destroy"
  - This will delete all the resources created via the Terraform template.

10. 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. Unable to access the webserver or web page not visible

If the VM-Series firewall is up and accessible but you are unable to access the webserver (or the web page is not visible), then chances are that the startup scripts did not get downloaded from the bootstrap bucket or were corrupted during (or prior to) the upload. Ensure that the files webserver-startup.sh and dbserver-startup.sh are in the bootstrap bucket. If they are extant, replace them with new copies downloaded from the GitHub repository.

2. 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/Pal0Alt0@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.