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

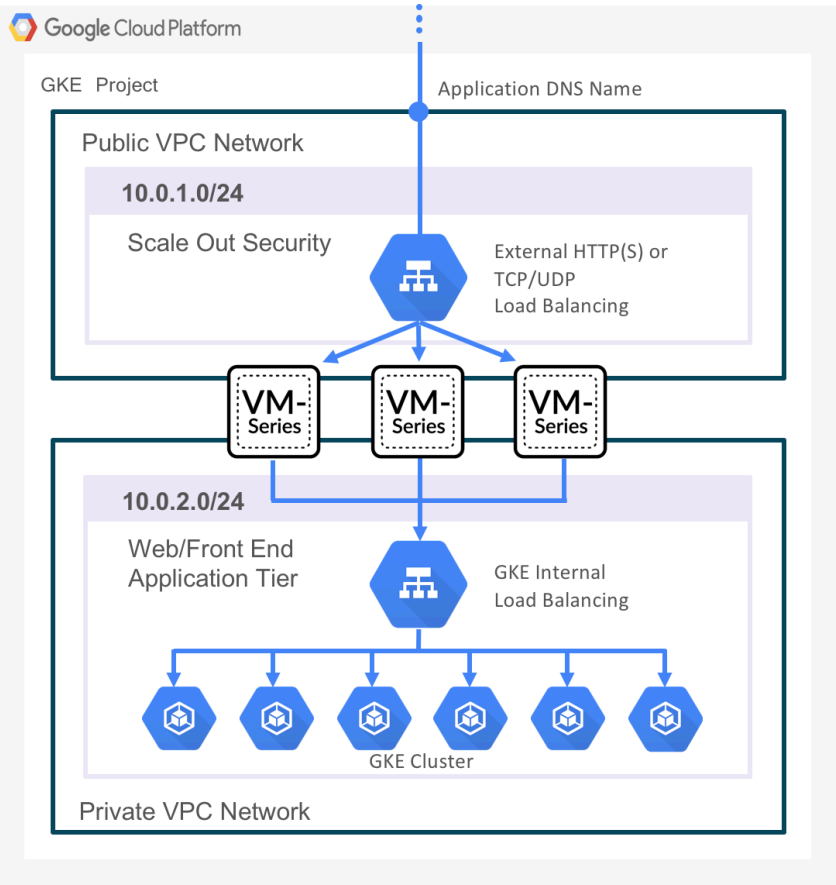
Version History **3**
1. About Terraform Templates **4**
2. Support Policy **5**
3. Instances used **5**
4. Prerequisites **5**
4.1 Create GCP account 5
4.2 Install the Google Cloud SDK 6
4.3 Accept the EULA (If Required) 6
4.4 Create a Project 6
4.5 Enable the API 7
4.6 Create a Bootstrap Bucket 9
4.7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

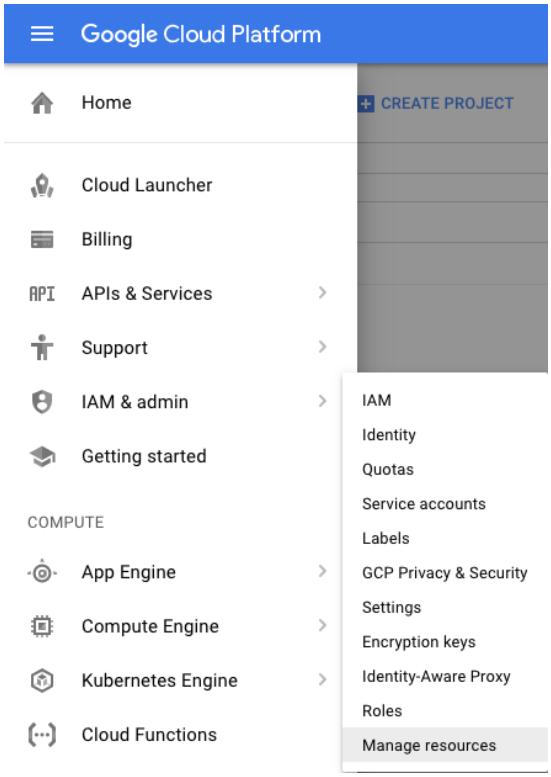
Template installations in GCP are performed from the CLI. Install the SDK/CLI by selecting the relevant platform from the following link and following the installation instructions:
https://cloud.google.com/sdk/
After installing Cloud SDK, install the `kubectl` command-line tool by running the following command:

```
gcloud components install kubectl
```

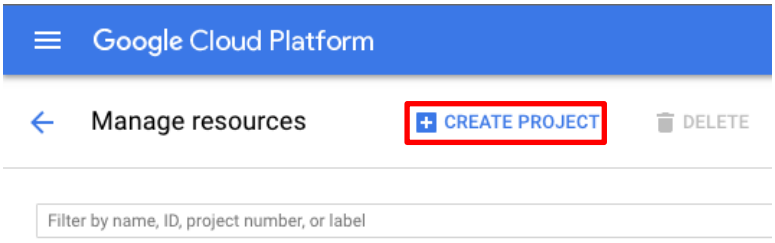
4.3 Accept the EULA (If Required)

4.4 Create a Project

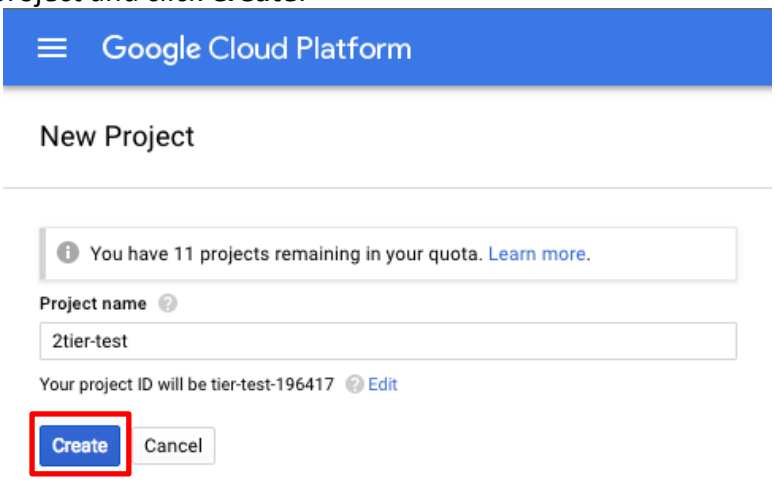
All GCP resources are deployed to a GCP Project. A GCP Project is an organizational boundary that separates users, resources, billing information, etc. A GCP Project is similar to an AWS VPC or an Azure Resource Group. By default, GCP will create a Project upon creation of an account. If that is not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



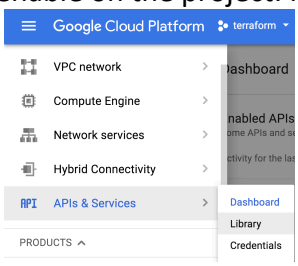
Specify a name for the project and click **Create**:



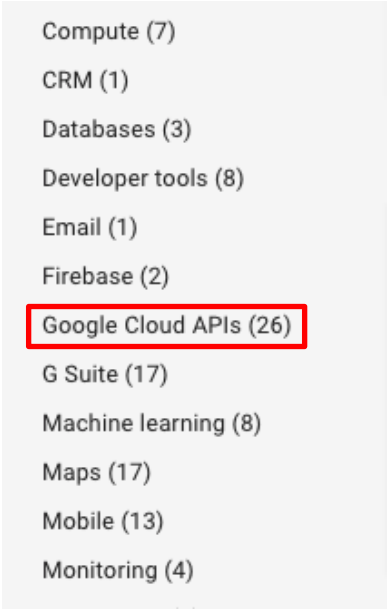
Note: GCP Project creation will take a few minutes.

4.5 Enable the API

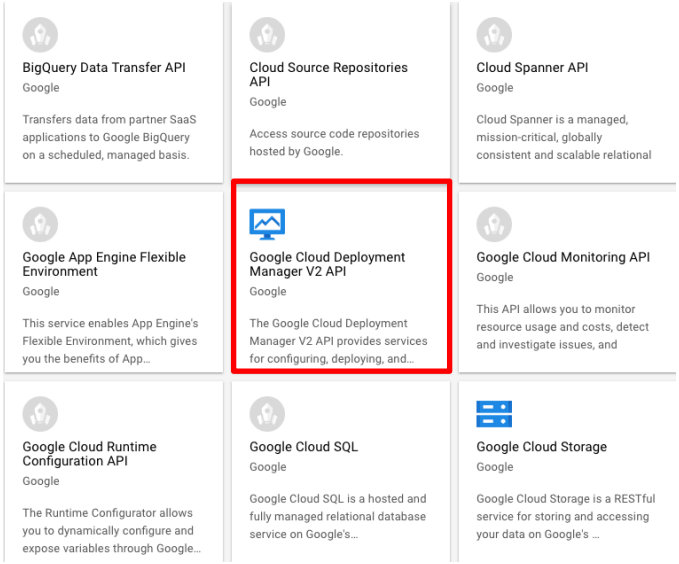
Deploying a template requires the API be enable on the project. Navigate to **APIs & Services > Library**:



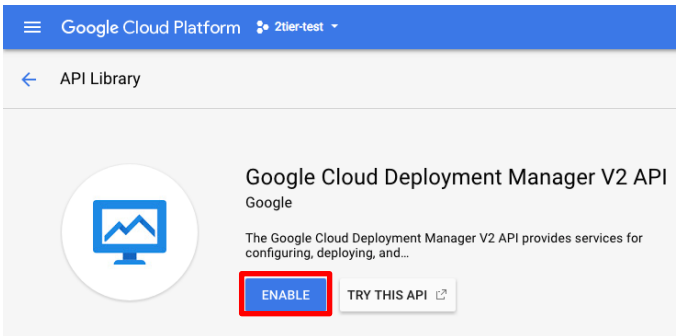
Select Google Cloud APIs on the left-hand-side:



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



Select **Enable**:

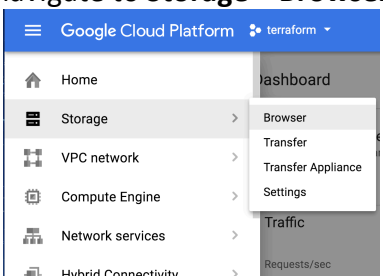


Note: Enabling the API for the project will take a few minutes to complete.

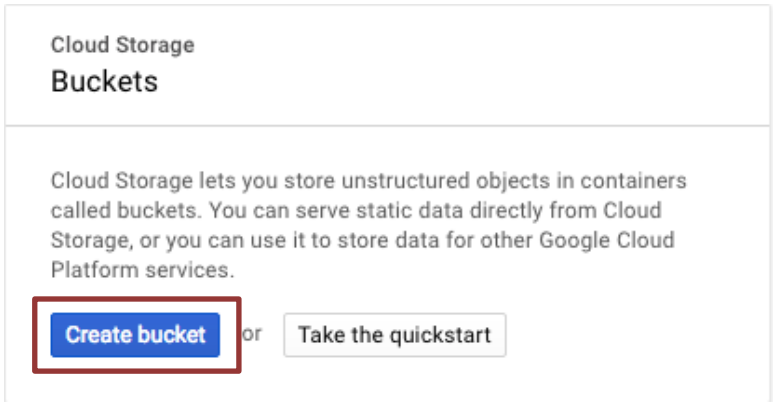
4.6 Create a Bootstrap Bucket

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.

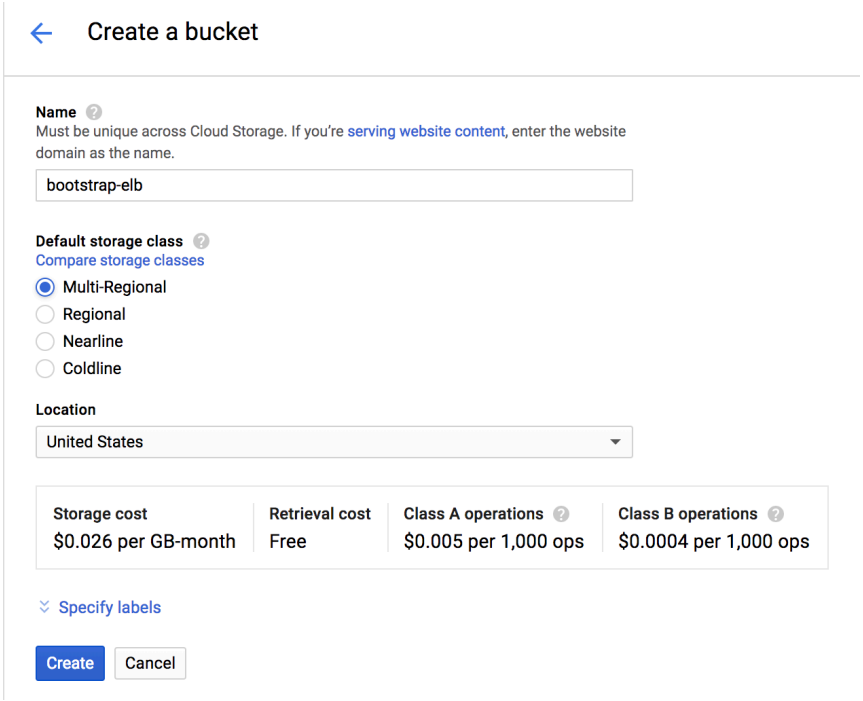
In order to create a Bootstrap bucket, navigate to **Storage > Browser**:



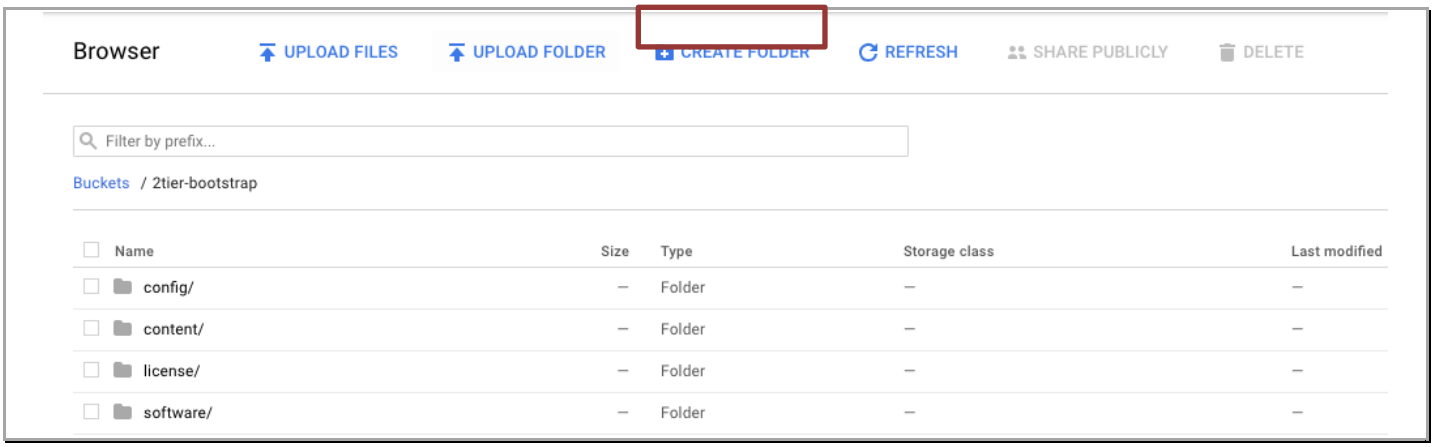
Click **Create Bucket**:



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**:

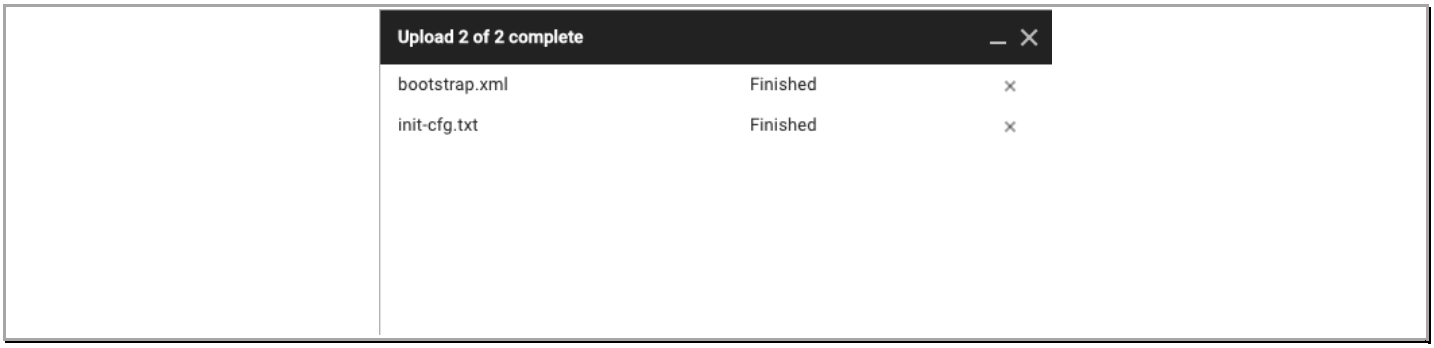


Download the following files using the links provided and save the files in a known location:

https://github.com/PaloAltoNetworks/GCP-Terraform-Samples/blob/master/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

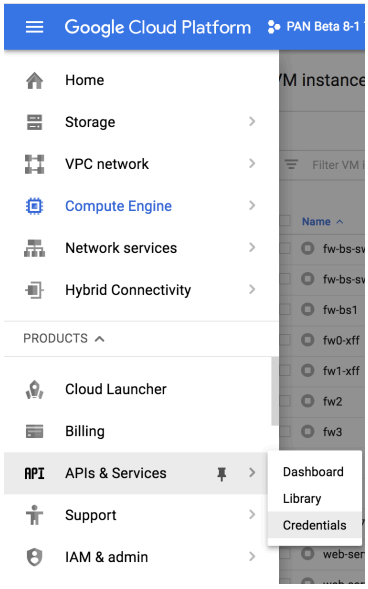
4.8 Gather Information and Update the Template File

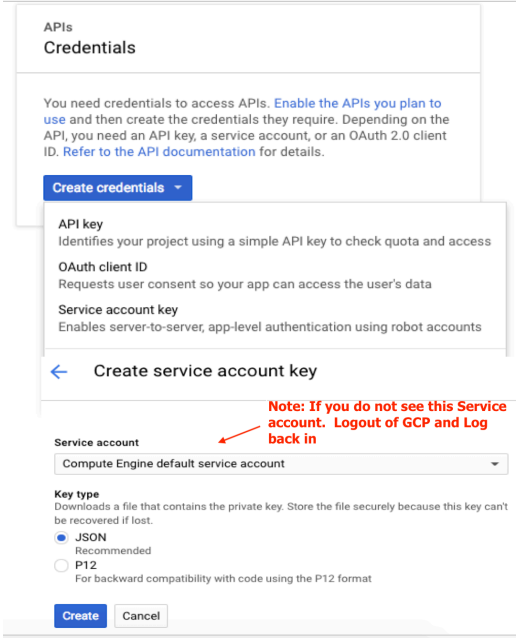
Deploying the Terraform template in GCP requires modification of the template Main and Variable files to include deployment-specific information. The minimum required information is:

```
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.

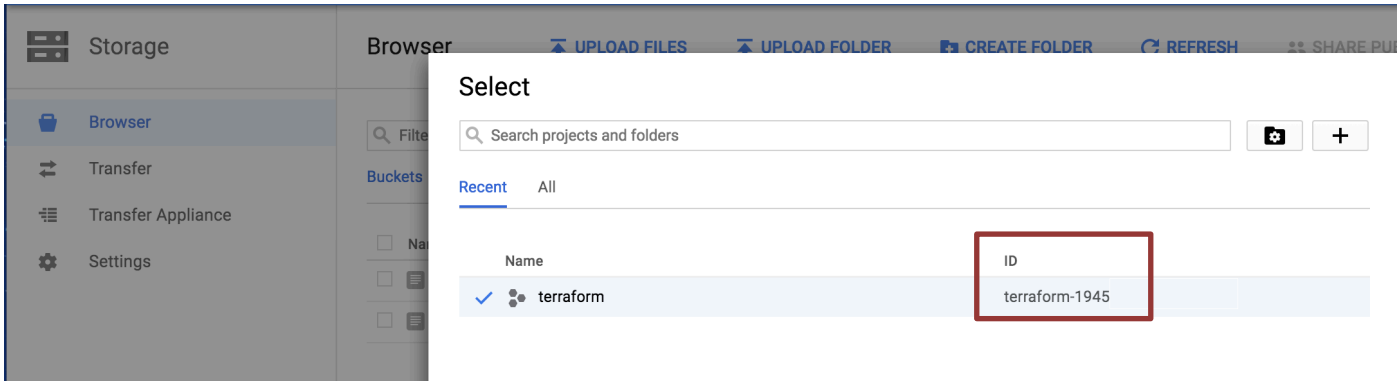
5. Launch the Template

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.

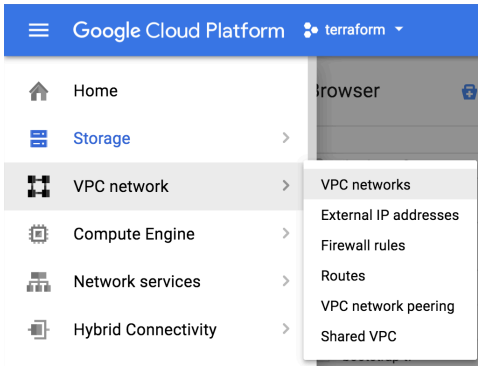
```
Plan: 12 to add, 0 to change, 0 to destroy.

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
```

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**:



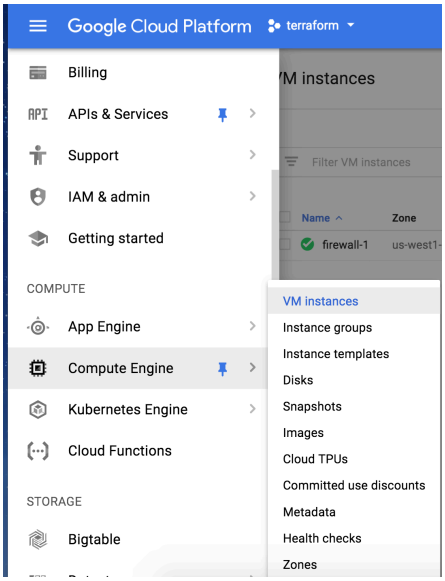
The template creates four networks: management-network, trust-network, and untrust-network.

VPC networks CREATE VPC NETWORK REFRESH

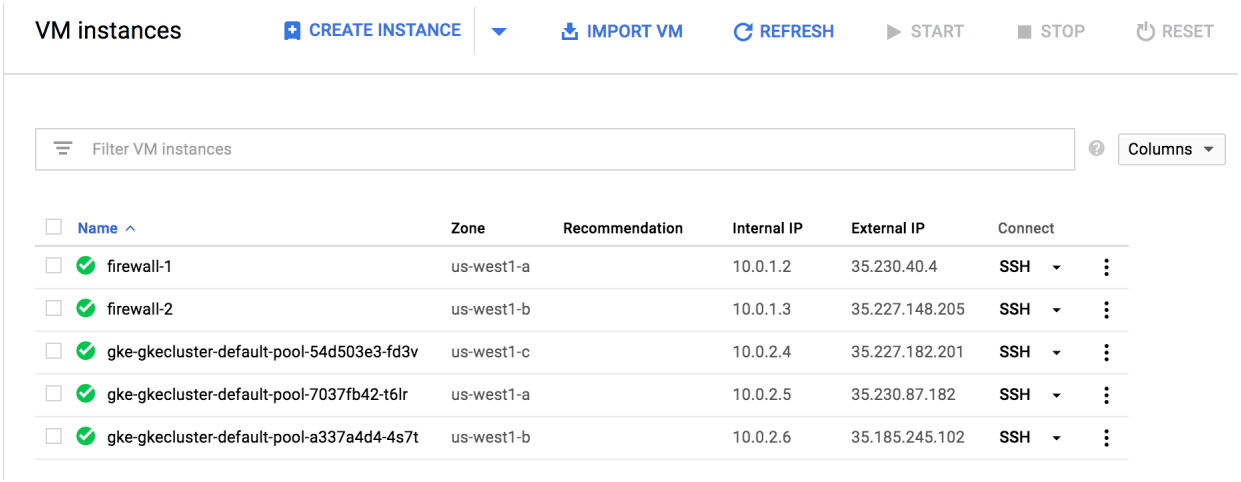
Name	Region	Subnets	Mode	IP addresses ranges	Gateways	Firewall Rules	Global dynamic routing
management		1	Custom			1	Off
	us-west1	management-sub		10.0.0.0/24	10.0.0.1		
trust		1	Custom			1	Off
	us-west1	trust-sub		10.0.2.0/24	10.0.2.1		
untrust		1	Custom			1	Off
	us-west1	untrust-sub		10.0.1.0/24	10.0.1.1		

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

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

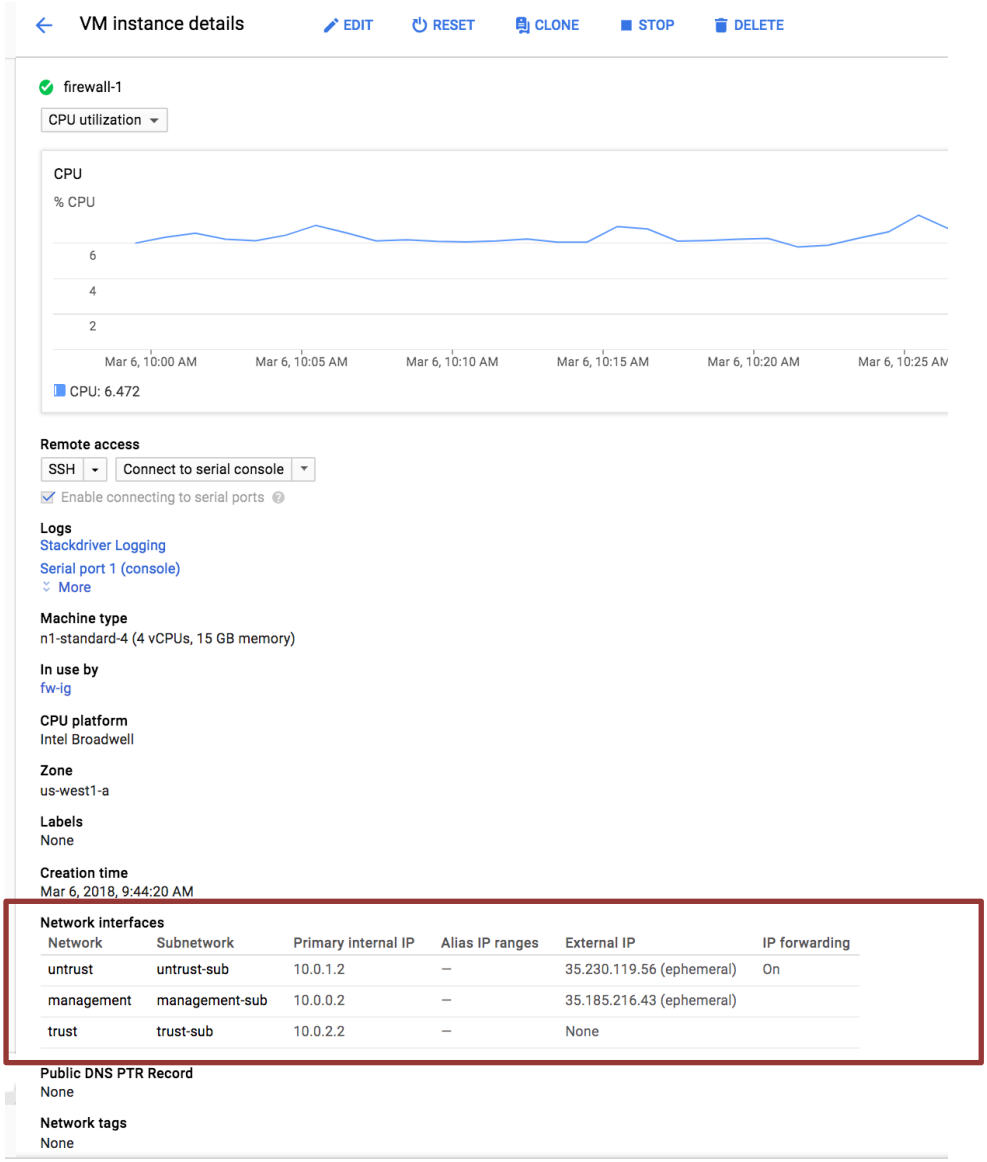


High-level information regarding the deployed instances are available with the default view:

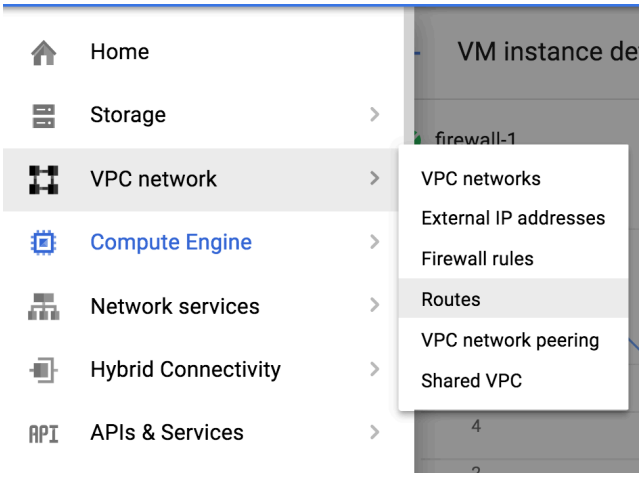


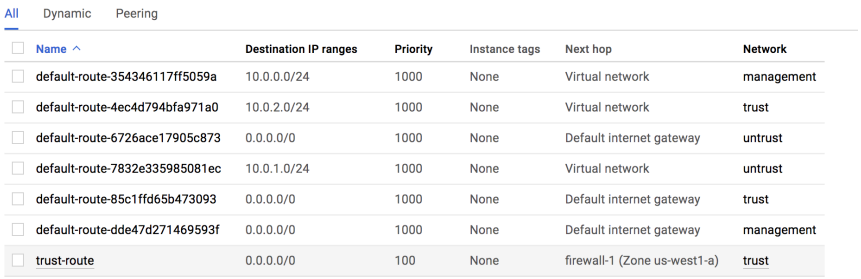
Note: The GKE nodes were created with External IP Addresses. At the time of the development of this Terraform template sample Terraform cannot deploy private GKE Clusters. The External IP Addresses can manually be deleted by editing the nodes.

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.



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.

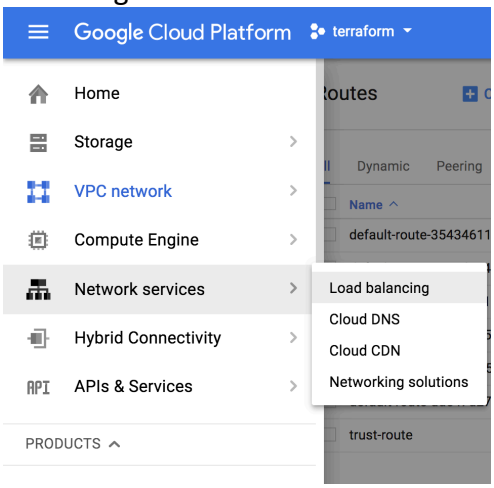




All | Dynamic | Peering

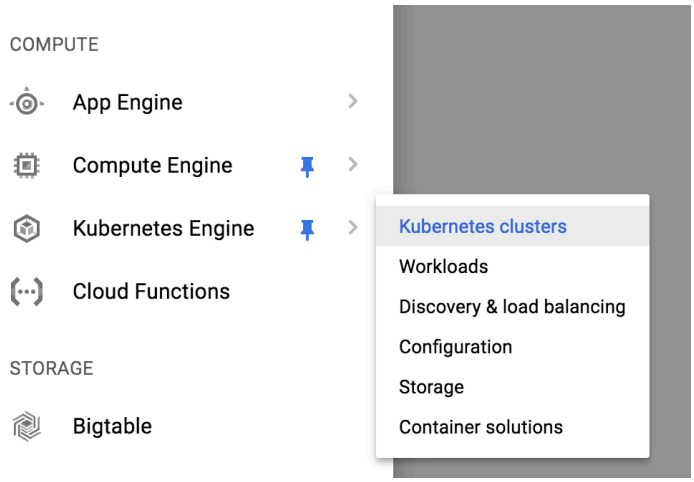
Name	Destination IP ranges	Priority	Instance tags	Next hop	Network
default-route-354346117ff5059a	10.0.0.0/24	1000	None	Virtual network	management
default-route-4ec4d794bfa971a0	10.0.2.0/24	1000	None	Virtual network	trust
default-route-6726ace17905c873	0.0.0.0/0	1000	None	Default internet gateway	untrust
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.

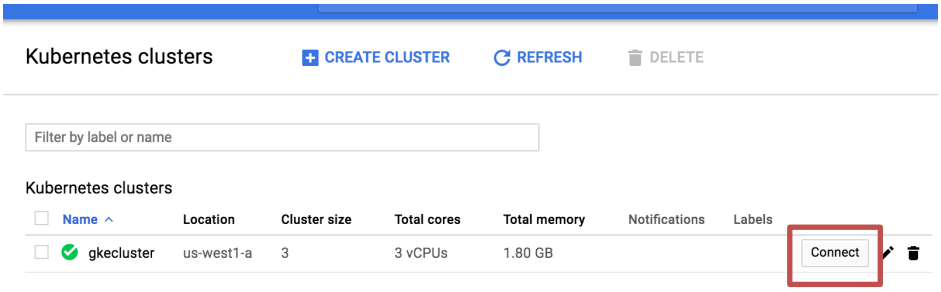


NOTE: You should see one Load Balancer configured (HTTP-ELB) showing that the backend service is unhealthy. The Internal GKE LB will be deployed via the config.yaml file. You will create NAT rules on the VM-Series Firewalls to achieve a healthy state for the HTTP-ELB backend services in the next session.

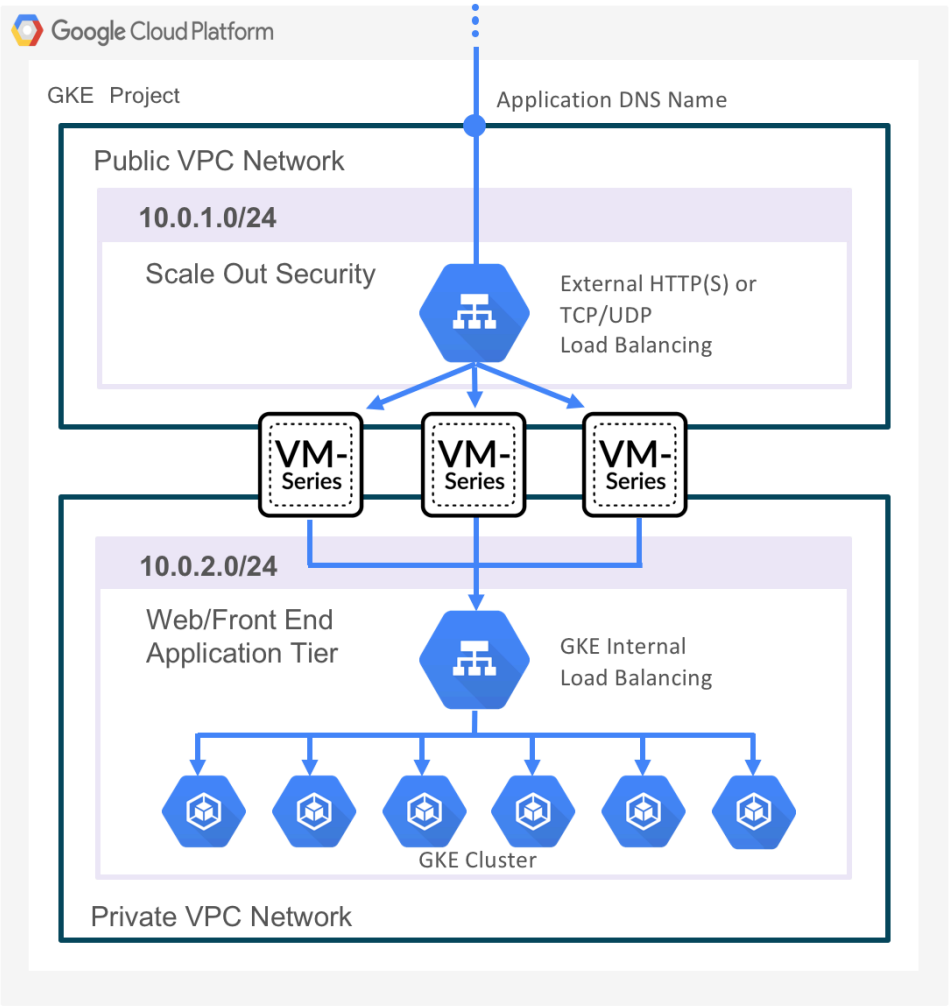
The GKE Cluster will have also deployed. That can be seen by clicking on Kubernetes Engine then selecting Kubernetes Cluster.



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

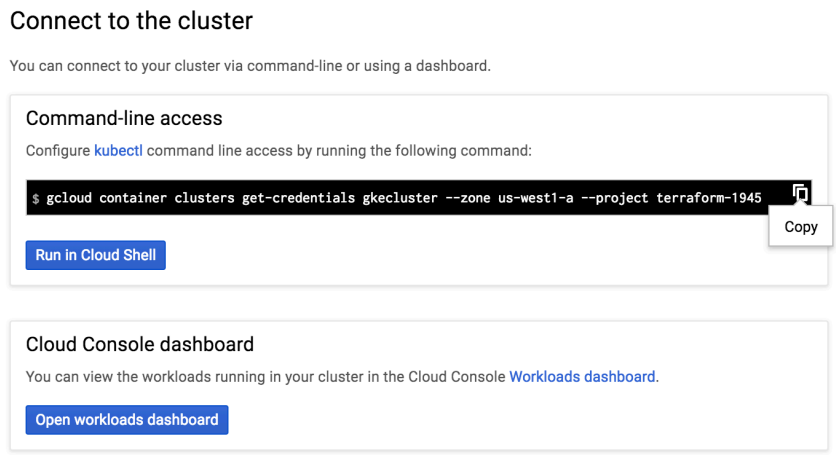


All of this matches the topology shown previously:



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.



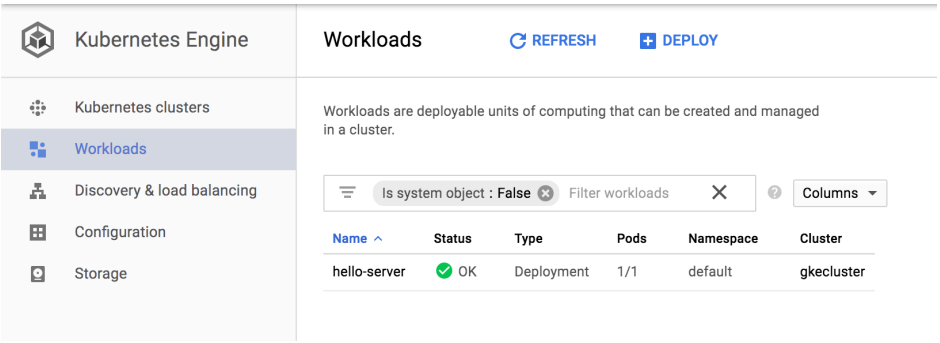

```
       32@terraform-1945   ~$
       32@terraform-1945   ~$ gcloud container clusters get-credentials gkecluster --zone us-west1-a --project terraform-1945
Fetching cluster endpoint and auth data.
kubeconfig entry generated for gkecluster.
```

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.

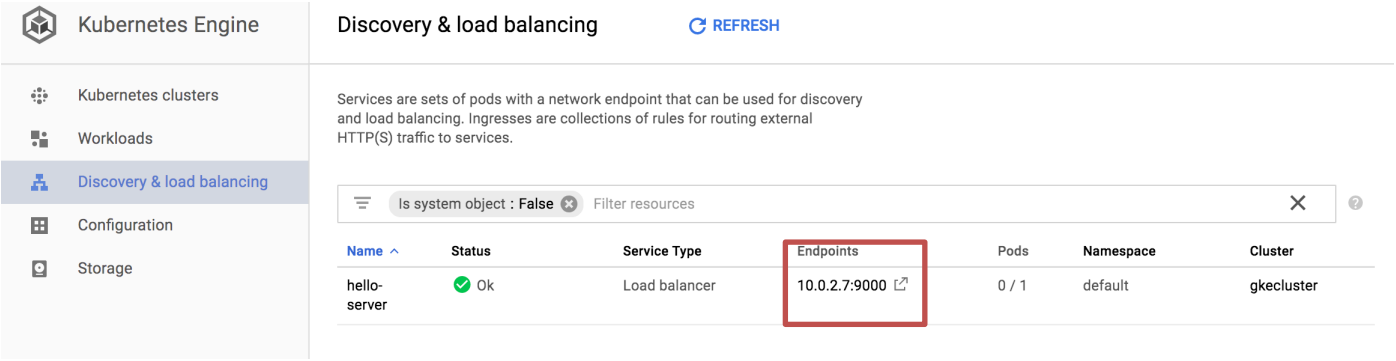


Now expose your service by creating the Internal Load Balancer for the GKE Cluster by launching the Config.yaml files by running the following command. If you have the file on your local machine and you have GCP SDK installed you can run the command from your local machine. A confirmation of <Service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



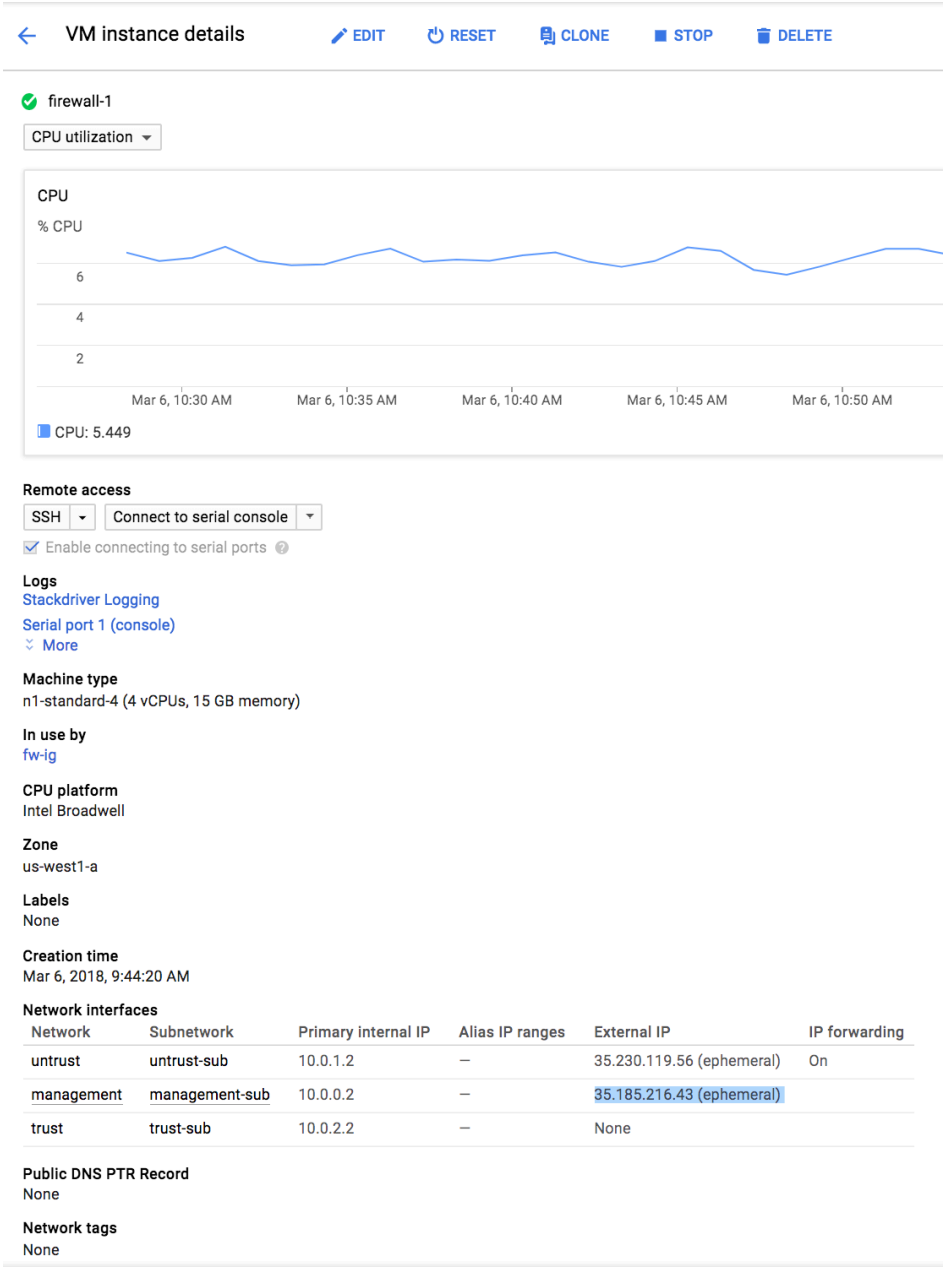
Note the Endpoints area. You will see the internal IP address of your Internal Load Balancer and the port it is listening on. This information can also be seen by running the command
$ kubectl get service
The output will look like this.

```
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
```

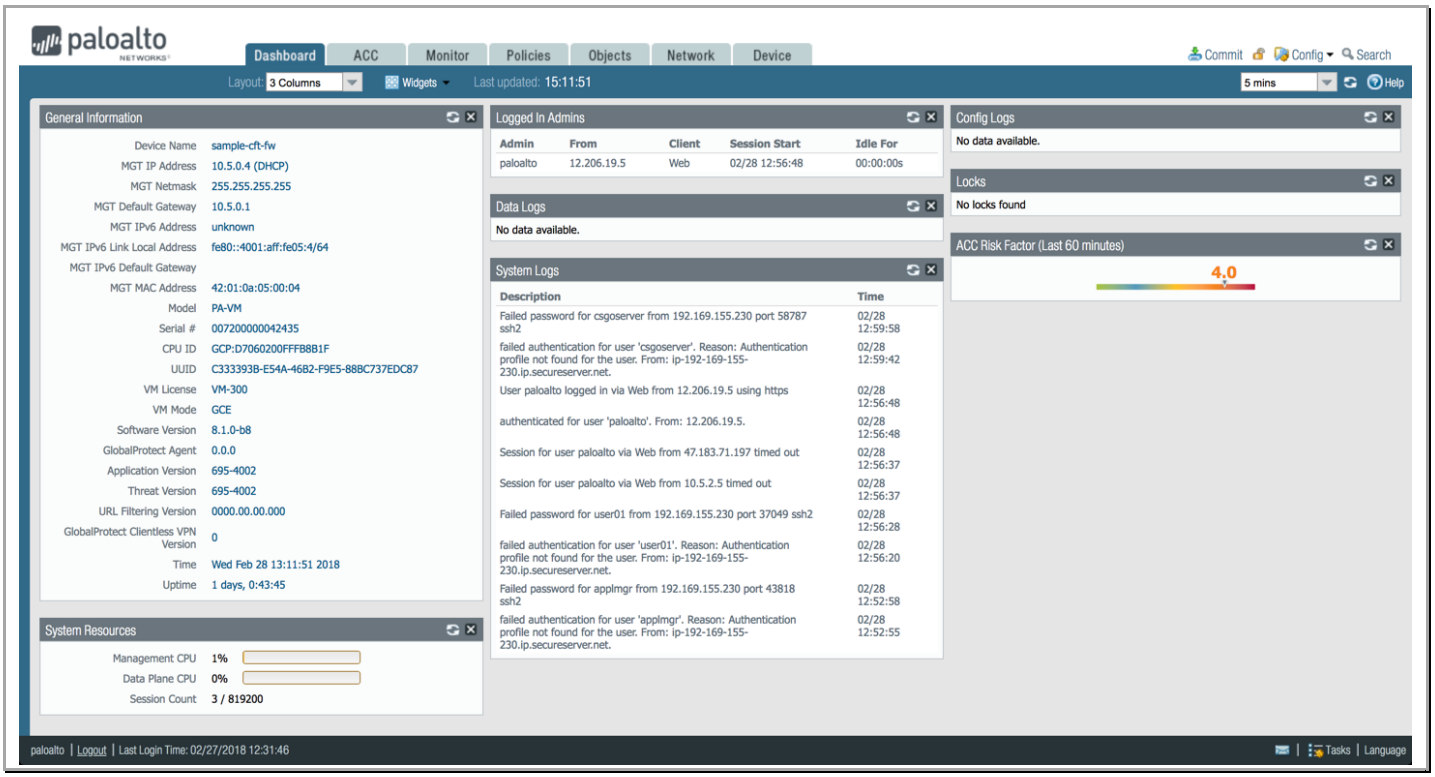
In the next section we will modify the NAT settings in the VM-Series firewall with the Endpoints information.

8. Access the firewall

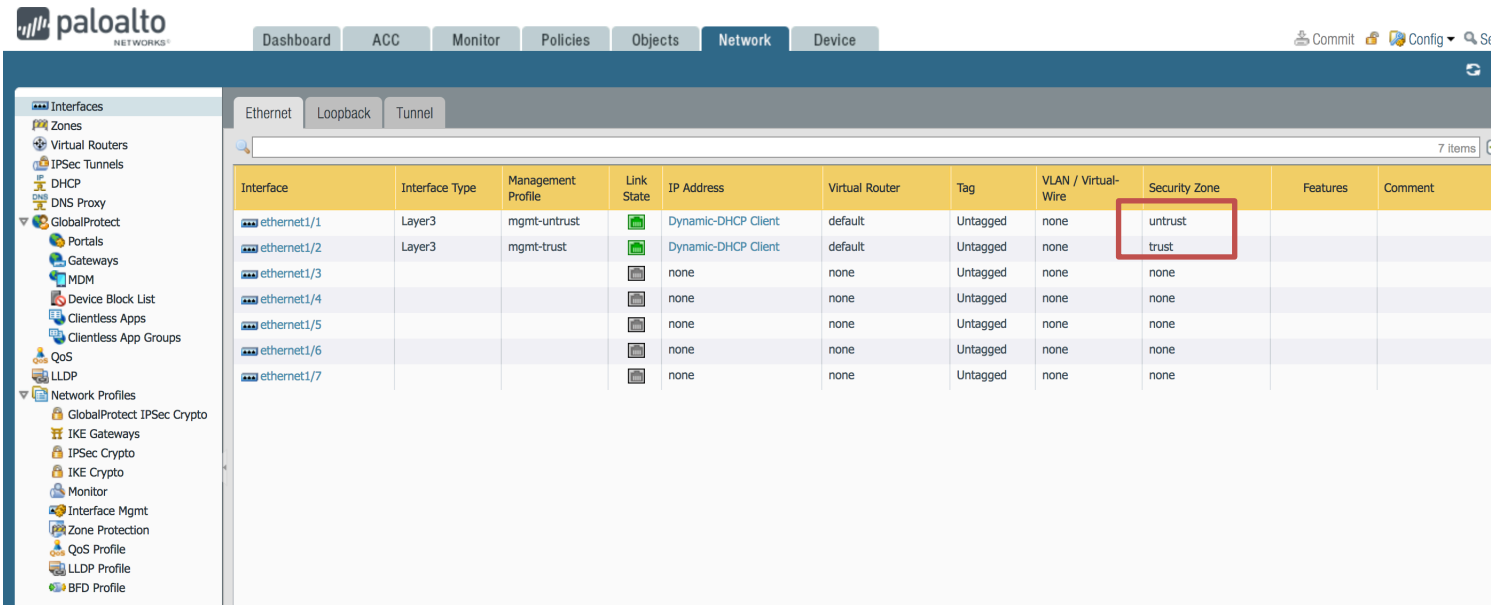
NOTE: Bootstrapping a VM-Series firewall takes approximately 9 minutes. Be patient☺ Once the template has been deployed successfully, it may be a while before the VM-Series firewall is up and you are able to log into the VM-Series firewall by browsing to the Management public IP Address. Recall we swapped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



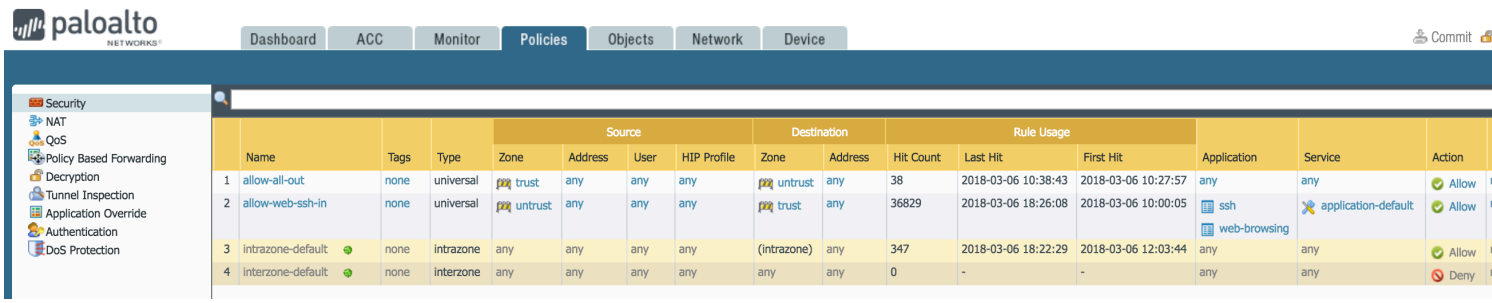
You should now be able to browse to the VM-Series firewall and login using the **username: paloalto** and password: **Pal0Alt0@123**



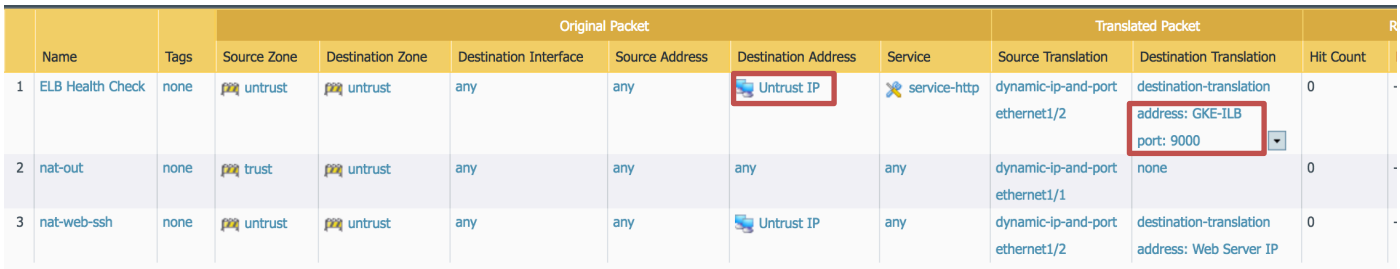
Here are the interfaces to zone mappings.



In the policies tab you can review the security policies:

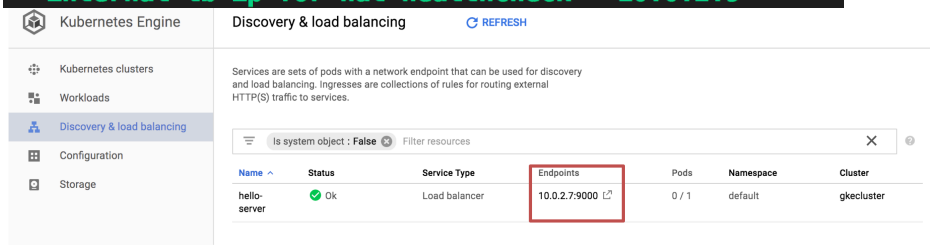


Under the polices tab select NAT on the left side. We need to have a health check from the ELB through to the backend Nodes of the GKE ILB. In order for the ELB health checks to flow through the VM-Series firewall then to and through the GKE ILB to the backend Nodes we need to add the Terraform outputs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

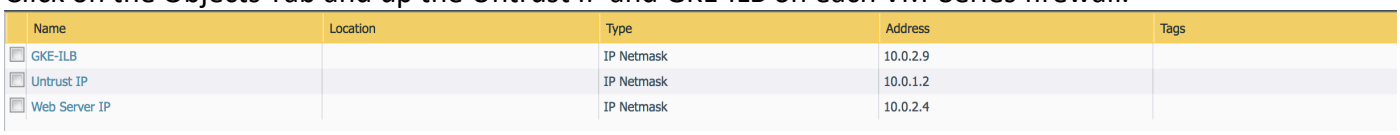


Make note of the outputs from your Terraform for the Untrust IP of the VM-Series firewall. The GKE-ILB IP address can be found under Endpoints. Your outputs may be different.

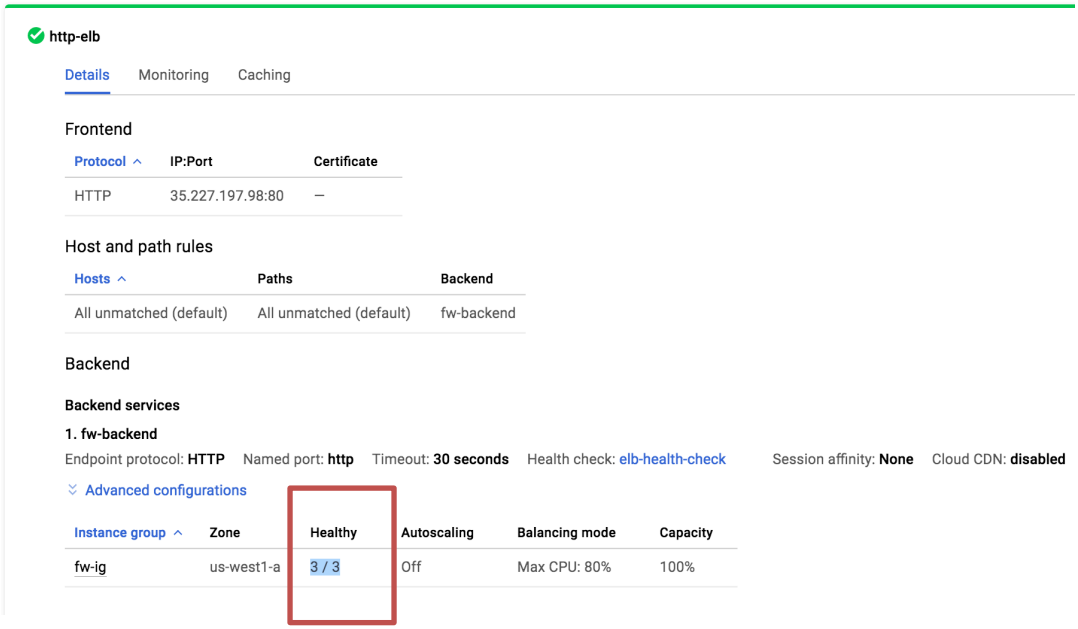
```
firewall-untrust-ips-for-nat-healthcheck = [
    10.0.1.2,
    10.0.1.3,
    10.0.1.4
]
internal-lb-ip-for-nat-healthcheck = 10.0.2.9
```



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

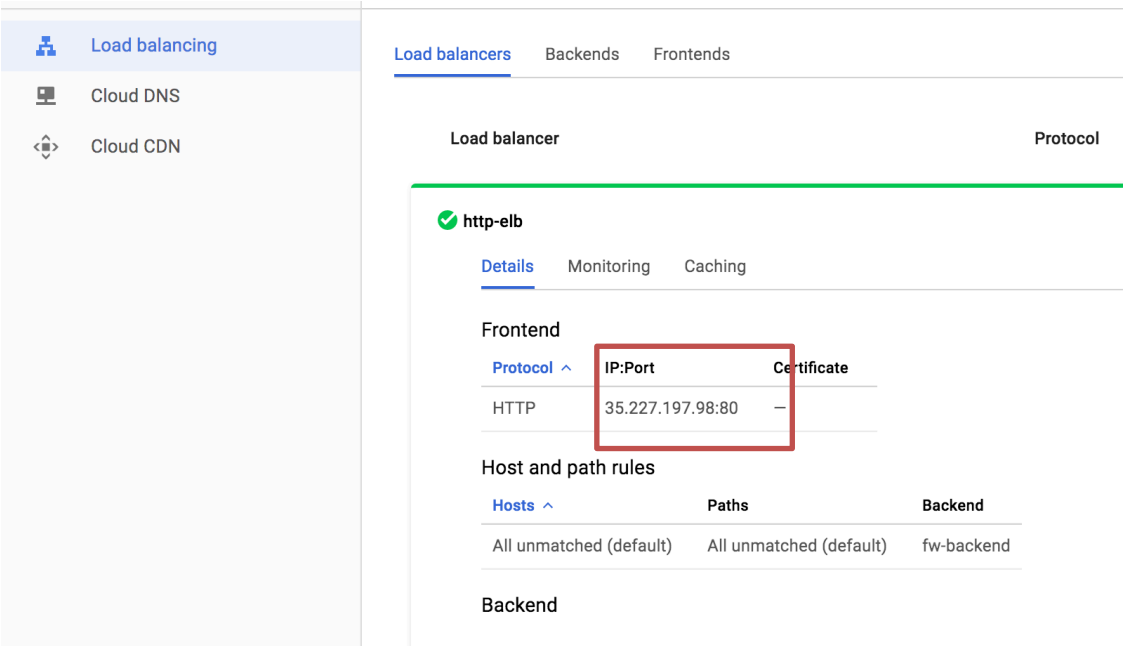


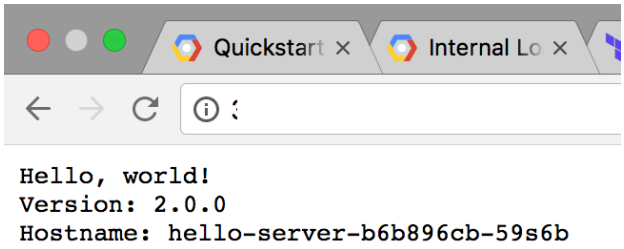
Repeat this step for all of the Firewalls and **COMMIT** these changes on the VM-Series Firewalls Then go back to your GCP Load Balancers and check the Health Status.



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.





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
- 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/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.

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.