

# GCP Transit VPC with Advanced Peering

## Terraform Build Guide

Matt McLimans, Public Cloud Consultant Engineer

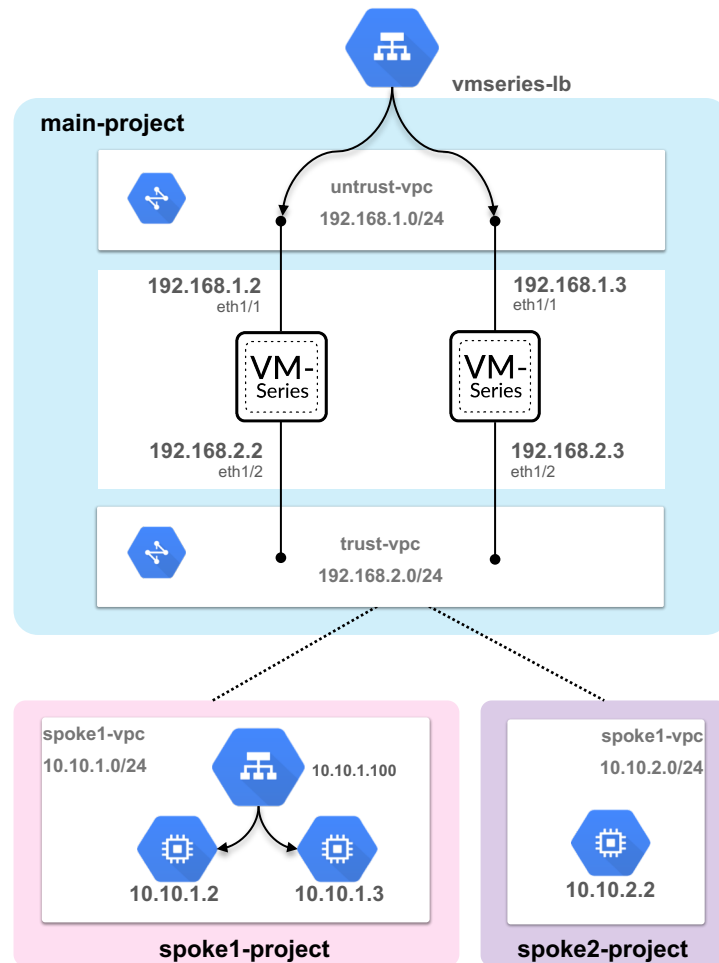


# SUPPORT POLICY

This 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>) or sites other than our official Downloads page on <https://support.paloaltonetworks.com> are provided under the best effort policy.

# DEPLOYMENT OVERVIEW

- Terraform builds 2 VM-Series firewalls and two peered VPCs.
- Spoke1 VPC has 1 internal load balancer and 2 backend Web servers (configured with Apache)
- Spoke2 VPC has 1 Linux host.
- spoke1-vpc & spoke2-vpc can be deployed into the same project as the VM-Series or in different projects.



# CONFIGURE GCP API & RETRIEVE API CREDENTIALS

# STEP 1. CREATE A PROJECT

1. Create a GCP Project
2. Record the Project ID.

Select from **MRM.WORLD**

NEW PROJECT

Search projects and folders

RECENT ALL

Name	ID
✓ [icon] [redacted] ?	[redacted]

CANCEL OPEN

### New Project

**Project name \***  
host-project ?

**Project ID** host-project-242119 It cannot be changed later. [EDIT](#)

**Organization**  
mrm.world ?

This project will be attached to mrm.world.

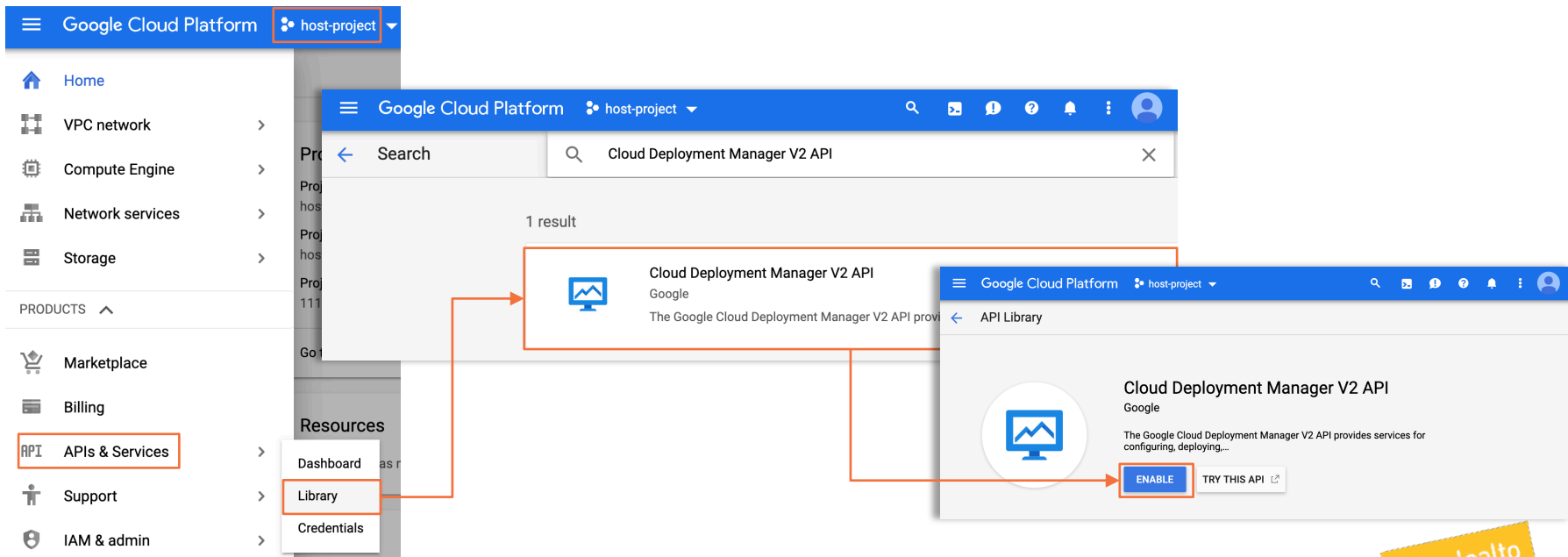
**Location \***  
[icon] mrm.world [redacted] [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

## STEP 2. ENABLE GOOGLE COMPUTE API

1. Go to **API & Services** → **Library**
2. Search for **Cloud Deployment Manager V2 API** and click **Enable**



## STEP 3. RETRIEVE API CREDENTIALS

1. Go to **API & Services** → **Credentials** → **Create Credentials** → **Service account key**
2. Select **Compute Engine default service account** and select **JSON** as the key type

The image shows two screenshots from the Google Cloud Platform interface. The left screenshot shows the 'APIs & Services' page with the 'Credentials' tab selected. A red box highlights the 'Credentials' link in the left sidebar. The right screenshot shows the 'Create service account key' dialog. A red box highlights the 'Service account' dropdown menu, which is set to 'Compute Engine default service account'. Another red box highlights the 'JSON' radio button under 'Key type', which is marked as 'Recommended'. A red arrow points from the 'Create credentials' button in the left screenshot to the 'Create service account key' dialog in the right screenshot.

**Google Cloud Platform** host-project

**APIs & Services** Credentials

Dashboard Library Credentials

**APIs Credentials**

You need credentials to access APIs. [Enable the APIs you plan to use](#) and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. For more information, see the [authentication documentation](#).

Create credentials

**API key**  
Identifies your project using a simple API key to check quota and access

**OAuth client ID**  
Requests user consent so your app can access the user's data

**Service account key**  
Enables server-to-server, app-level authentication using robot accounts

**Help me choose**  
Asks a few questions to help you decide which type of credential to use

**Google Cloud Platform** host-project

Create service account key

**Service account**  
Compute Engine default service account

**Key type**  
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

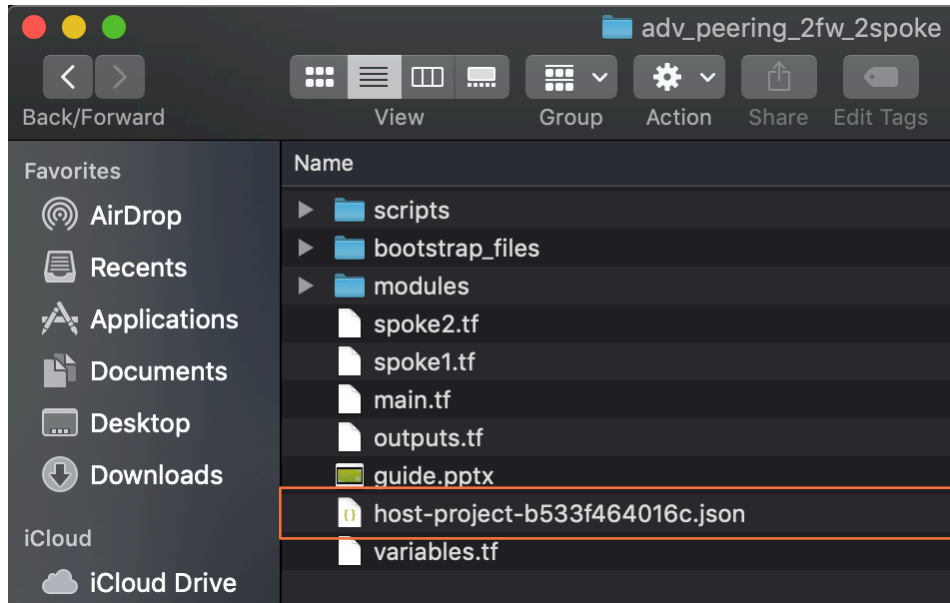
☒ **JSON** Recommended

☐ **P12**  
For backward compatibility with code using the P12 format

Create Cancel

## STEP 4. RETRIEVE API CREDENTIALS

1. Move the downloaded key into the main directory of the Terraform build.
2. Download the Terraform build from



**Repeat STEPS1-4 if you want Spoke1  
& Spoke2 to reside in different  
projects than the VM-Series**



# DELETE DEFAULT NETWORK

Every new project has a default VPC. Each project has a soft maximum of 5 VPCs.

If you are deploying everything to the same project, make sure you either delete the default VPC in the project or ask for a quota increase.

The screenshot shows the Google Cloud Platform interface for VPC networks. The top navigation bar includes the Google Cloud Platform logo, the project name 'host-project', and a search icon. The left sidebar shows the 'VPC network' section with a list of resources: 'VPC networks', 'External IP addresses', 'Firewall rules', 'Routes', 'VPC network peering', 'Shared VPC', and 'Serverless VPC access'. The 'VPC networks' resource is highlighted. The main content area displays a table of VPC networks. The 'default' network is highlighted with a red box. The table has columns for Name, Region, Subnets, Mode, IP addresses ranges, Gateways, and Firewall Rule. The 'default' network is in the 'us-central1' region, has 20 subnets, and is in 'Auto' mode. Below the table, the 'VPC network details' for the 'default' network are shown. The details include a description, subnet creation mode (Auto subnets), dynamic routing mode (Regional), and DNS server policy (None). A red box highlights the 'DELETE VPC NETWORK' button in the top right corner of the details panel.

Name	Region	Subnets	Mode	IP addresses ranges	Gateways	Firewall Rule
default	us-central1	20	Auto	10.128.0.0/20	10.128.0.1	4
	europe-west1	default		10.132.0.0/20	10.132.0.1	
	us-west1	default		10.138.0.0/20	10.138.0.1	
	asia-east1	default		10.140.0.0/20	10.140.0.1	
	us-east1	default		10.142.0.0/20	10.142.0.1	

**VPC network details**

**default**

**Description**  
Default network for the project

**Subnet creation mode**  
Auto subnets

**Dynamic routing mode**  
Regional

**DNS server policy**  
None

**Subnets** | Static internal IP addresses | Firewall rules | Routes | VPC Network Peering | Private service



# EDIT VARIABLES.TF

## STEP 5. ADJUST VARIABLES.TF

1. Open **variables.tf** in a text editor.
2. Enter the project ID for each project in:
  - **main\_project**
  - **spoke1\_project**
  - **spoke2\_project**
3. Enter the corresponding key file for the projects in:
  1. **main\_project\_authfile**
  2. **spoke1\_project\_authfile**
  3. **spoke2\_project\_authfile**
4. In this example, we are deploying everything to the same project (host-project-242119), so the project ID and authfile value will be the same for main, spoke1, and spoke2 environments.

```
#####
# main.tf PROJECT ID & AUTHFILE
#####
1 references
variable "main_project" {
  description = "Existing project ID for main project (all resources deployed in main.tf)"
  default     = "host-project-242119"
}

1 references
variable "main_project_authfile" {
  description = "Authentication file for main project (all resources deployed in main.tf)"
  default     = "host-project-b533f464016c.json"
}

#####
# spoke1.tf PROJECT ID & AUTHFILE
#####
1 references
variable "spoke1_project" {
  description = "Existing project for spoke1 (can be the same as main project and can be same as main project)."
  default     = "host-project-242119"
}

1 references
variable "spoke1_project_authfile" {
  description = "Authentication file for spoke1 project (all resources deployed in spoke1.tf)"
  default     = "host-project-b533f464016c.json"
}

#####
# spoke2.tf PROJECT ID & AUTHFILE
#####
1 references
variable "spoke2_project" {
  description = "Existing project for spoke2 (can be the same as main project and can be same as main project)."
  default     = "host-project-242119"
}

1 references
variable "spoke2_project_authfile" {
  description = "Authentication file for spoke2 project (all resources deployed in spoke2.tf and can be same as main project)"
  default     = "host-project-b533f464016c.json"
}
```

## STEP 6. SSH KEY FOR UBUNTU VM & VM-SERIES LICENSE TYPE

1. Create an SSH key for instances in the Spoke VPCs.

```
$ ssh-keygen -t rsa -f ~/.ssh/ubuntukey -C ubuntu  
  
<enter passphrase x 2>  
  
$ chmod 600 ~/.ssh/ubuntukey  
  
$ cat ~/.ssh/ubuntukey.pub
```

```
#####  
# UBUNTU SSH KEY  
#####  
2 references  
variable "ubuntu_ssh_key" {  
  default = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDBAmjFRPLEwSvNH41yU/7ouw7vB0BJzprcMss  
}
```

2. Copy CAT output and paste it as the default value for **ubuntu\_ssh\_key** inside **variables.tf**

3. Uncomment the **vmseries\_image** to the license SKU that you want.

```
1 references  
variable "vmseries_image" {  
  # default = "https://www.googleapis.com/compute/v1/projects/paloaltonetworksgcp-public/global/images/vmseries-byol-814"  
  default = "https://www.googleapis.com/compute/v1/projects/paloaltonetworksgcp-public/global/images/vmseries-bundle1-814"  
  # default = "https://www.googleapis.com/compute/v1/projects/paloaltonetworksgcp-public/global/images/vmseries-bundle2-814"  
}
```

**SAVE VARIABLES.TF**



# RUN TERRAFORM

# STEP 7. RUN TERRAFORM

## 1. terraform init

```
adv_peering_2fw_2spoke mmclimans$ 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.

## 2. terraform apply

```
adv_peering_2fw_2spoke mmclimans$ terraform apply
```

```
...  
...  
...
```

Plan: 49 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**

**You will receive this output once the deployment has completed.**

**Apply complete! Resources: 49 added, 0 changed, 0 destroyed.**

**Outputs:**

```

===== IMPORTANT!! PLEASE READ!! =====
Before proceeding, you must enable import/export custom routes on all peering links,
and remove the default (0.0.0.0/0) route from TRUST, SPOKE1, and SPOKE2 VPCs,
=====
]
GLB-ADDRESS      = http://35.244.207.26
MGMT-URL-FW1     = https://35.245.168.131
MGMT-URL-FW2     = https://35.199.45.71
SSH-SPOKE1-FW1   = ssh ubuntu@35.230.184.204 -p 221 -i <INSERT KEY>
SSH-SPOKE1-FW2   = ssh ubuntu@35.194.81.140 -p 221 -i <INSERT KEY>
SSH-SPOKE2-FW1   = ssh ubuntu@35.230.184.204 -p 222 -i <INSERT KEY>
SSH-SPOKE2-FW2   = ssh ubuntu@35.194.81.140 -p 222 -i <INSERT KEY>
```

# STEP 8. ENABLE IMPORT/EXPORT CUSTOM ROUTES

Go to: **VPC Network** → **VPC network peering**

- For **EACH PEER**, enable **Import custom routes** & **Export custom routes**

The screenshot shows the Google Cloud Platform interface for VPC network peering. On the left, the 'VPC network peering' option is selected in the sidebar. The main panel shows the 'spoke1-to-trust' peering connection details. Under the 'Exchange custom routes' section, both 'Import custom routes' and 'Export custom routes' are checked. A red box highlights these two options. An inset window titled 'VPC Network Peering' shows a table of all peering connections, with a red box highlighting the 'Exchange custom routes' column for each row.

Name	Your VPC network	Peered VPC network	Peered project ID	Status	Exchange custom routes
spoke1-to-trust	spoke1-vpc	trust-vpc	host-project-242119	Connected.	Import & Export custom routes
spoke2-to-trust	spoke2-vpc	trust-vpc	host-project-242119	Connected.	Import & Export custom routes
trust-to-spoke1	trust-vpc	spoke1-vpc	host-project-242119	Connected.	Import & Export custom routes
trust-to-spoke2	trust-vpc	spoke2-vpc	host-project-242119	Connected.	Import & Export custom routes

When done, your peering connections should look like this.

# STEP 9. DELETE TRUST & SPOKE DEFAULT INTERNET ROUTES

Go to: **VPC Network → Routes**

- Delete **spoke1-vpc**, **spoke2-vpc**, & **trust-vpc** default route to the internet.

Routes

+ CREATE ROUTE

REFRESH

DELETE

One or more VPC networks in this project has been configured to import custom routes using VPC Network Peering. Any imported custom dynamic routes are omitted from this list, and some route conflicts might complete list of imported custom routes, and the [routing order](#) for information about how GCP resolves conflicts.

Filter resources

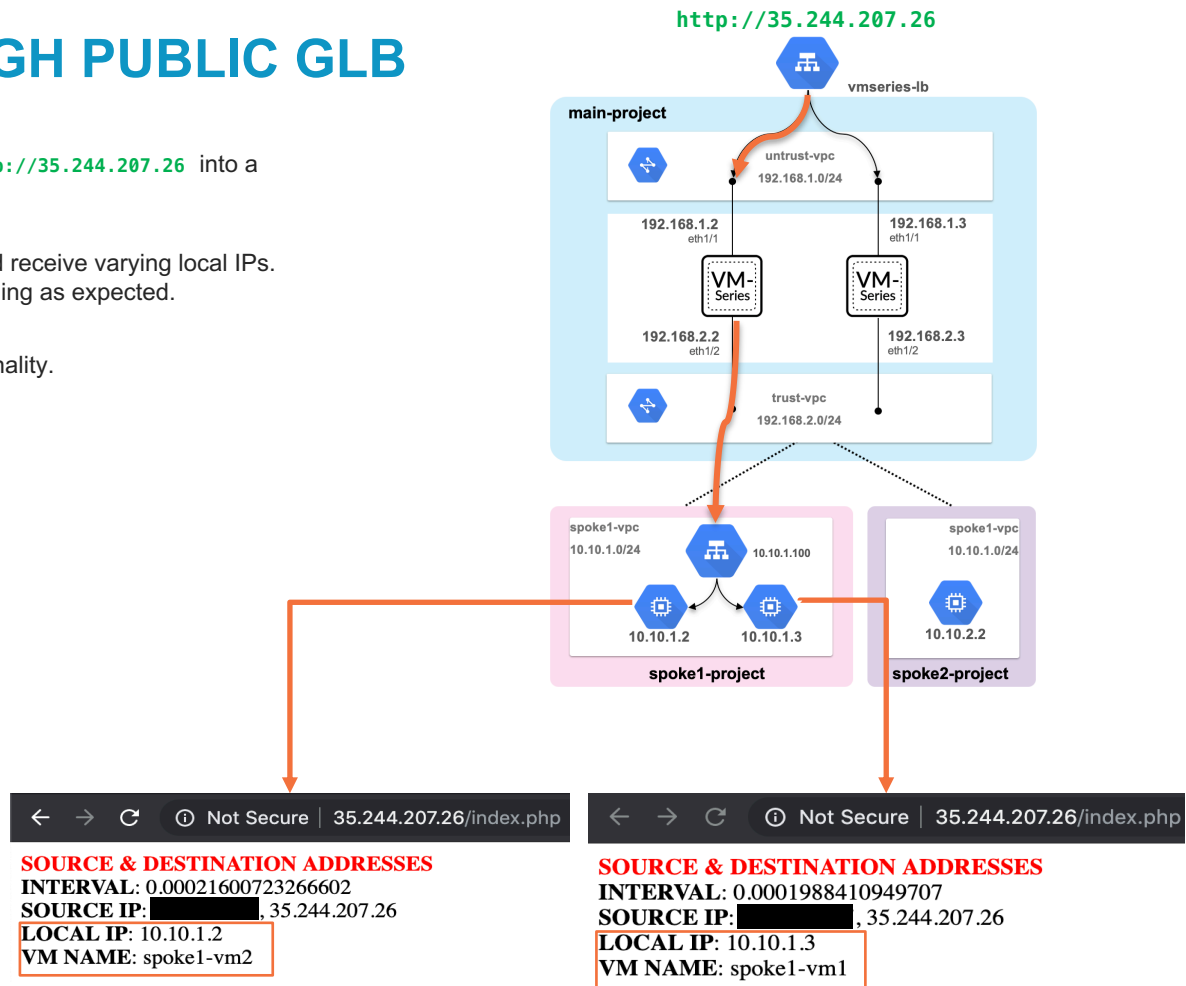
	Name	Description	Destination IP range ^	Priority	Instance tags	Next hop	Network
<input type="checkbox"/>	default-to-vmseries01		0.0.0.0/0	100	None	Instance vmseries01 (zone us-east4-a)	trust-vpc
<input type="checkbox"/>	default-to-vmseries02		0.0.0.0/0	100	None	Instance vmseries02 (zone us-east4-b)	trust-vpc
	peering-route-d475ceb4173e9483	Auto generated route via peering [spoke1-to-trust].	0.0.0.0/0	100	None	Network peering spoke1-to-trust	spoke1-vpc
	peering-route-e13564b53e9f2e09	Auto generated route via peering [spoke1-to-trust].	0.0.0.0/0	100	None	Network peering spoke1-to-trust	spoke1-vpc
	peering-route-15ad69d2a49fb174	Auto generated route via peering [spoke2-to-trust].	0.0.0.0/0	100	None	Network peering spoke2-to-trust	spoke2-vpc
	peering-route-f6da3dad37748741	Auto generated route via peering [spoke2-to-trust].	0.0.0.0/0	100	None	Network peering spoke2-to-trust	spoke2-vpc
<input type="checkbox"/>	default-route-20628ec1d28a95c6	Default route to the Internet.	0.0.0.0/0	1000	None	Default internet gateway	mgmt-vpc
<input checked="" type="checkbox"/>	default-route-df8a810aaf3353d8	Default route to the Internet.	0.0.0.0/0	1000	None	Default internet gateway	spoke1-vpc
<input checked="" type="checkbox"/>	default-route-28c163fa61f105cd	Default route to the Internet.	0.0.0.0/0	1000	None	Default internet gateway	spoke2-vpc
<input checked="" type="checkbox"/>	default-route-3b1dd6054bde92ae	Default route to the Internet.	0.0.0.0/0	1000	None	Default internet gateway	trust-vpc
<input type="checkbox"/>	default-route-f6ad36676090d8c6	Default route to the Internet.	0.0.0.0/0	1000	None	Default internet gateway	untrust-vpc
	peering-route-4671025c54e80b82	Auto generated route via peering [trust-to-spoke1].	10.10.1.0/24	1000	None	Network peering trust-to-spoke1	trust-vpc
<input type="checkbox"/>	default-route-54d2fe18ad71c197	Default local route to the subnetwork 10.10.1.0/24.	10.10.1.0/24	1000	None	Virtual network spoke1-vpc	spoke1-vpc



# TEST TRAFFIC FLOWS

# TEST INBOUND THROUGH PUBLIC GLB

1. From the Terraform output, copy **GLB-ADDRESS** = **http://35.244.207.26** into a web browser.
2. Once the page resolves, on each refresh you should receive varying local IPs. This indicates that ingress load balancing is functioning as expected.
3. View the firewall logs to view load balancing functionality.



# TEST OUTBOUND

1. SSH through either firewall to a backend server.
2. Test egress connectivity (i.e. `sudo apt-get update` ).
3. View the firewall logs. The egress request should flow through both firewalls since we are leveraging ECMP.

## FW1 Egress Traffic

	Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
	05/29 17:50:09	end	trust-zone	untrust-zone	10.10.1.3		91.189.88.161	80	apt-get	allow
	05/29 17:50:08	end	trust-zone	untrust-zone	10.10.1.2		91.189.88.162	80	apt-get	allow
	05/29 17:49:52	start	trust-zone	untrust-zone	10.10.1.3		91.189.88.161	80	apt-get	allow
	05/29 17:49:52	start	trust-zone	untrust-zone	10.10.1.3		91.189.88.161	80	web-browsing	allow

## FW2 Egress Traffic

	Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application
	05/29 17:50:08	end	trust-zone	untrust-zone	10.10.1.2		91.189.88.162	80	apt-get
	05/29 17:50:07	end	trust-zone	untrust-zone	10.10.1.3		91.189.88.161	80	apt-get
	05/29 17:49:53	start	trust-zone	untrust-zone	10.10.1.2		91.189.88.162	80	apt-get
	05/29 17:49:53	start	trust-zone	untrust-zone	10.10.1.2		91.189.88.162	80	web-browsing

```

GLB-ADDRESS      = http://35.244.207.26
MGMT-URL-FW1     = https://35.245.168.131
MGMT-URL-FW2     = https://35.199.45.71
SSH-SPOKE1-FW1   = ssh ubuntu@35.230.184.204 -p 221 -i <INSERT KEY>
SSH-SPOKE1-FW2   = ssh ubuntu@35.194.81.140 -p 221 -i <INSERT KEY>
SSH-SPOKE2-FW1   = ssh ubuntu@35.230.184.204 -p 222 -i <INSERT KEY>
SSH-SPOKE2-FW2   = ssh ubuntu@35.194.81.140 -p 222 -i <INSERT KEY>
    
```

