GCP Transit VPC

VPC Peering & Internal Load Balancer as Next Hop

Matt McLimans
Public Cloud Consultant Engineer

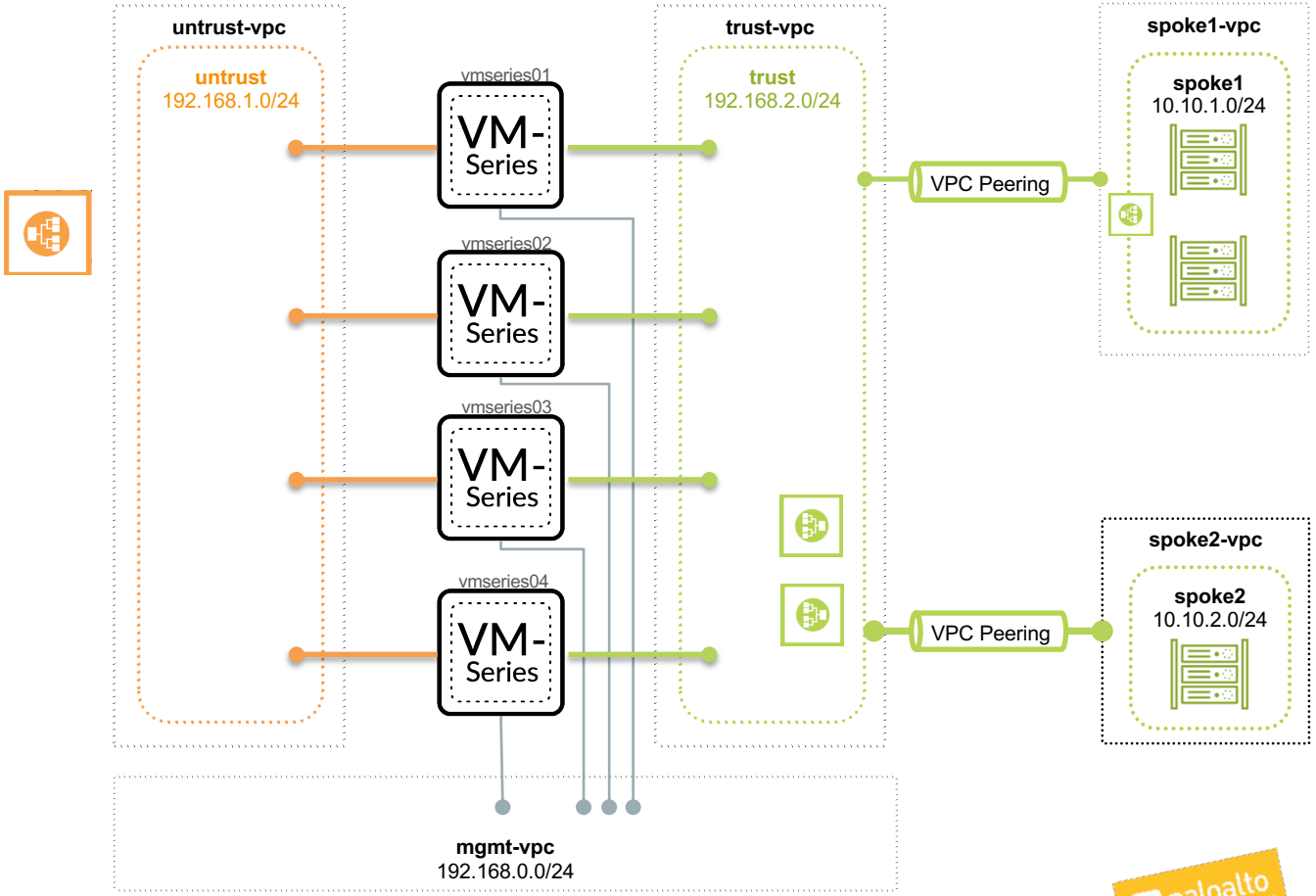
paloalto NETWORKS®

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.

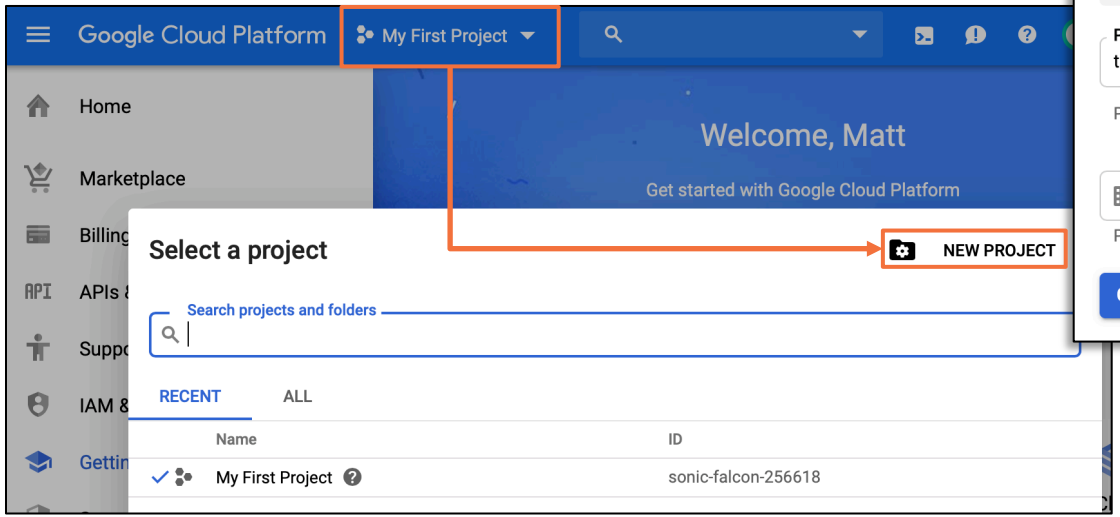
OVERVIEW

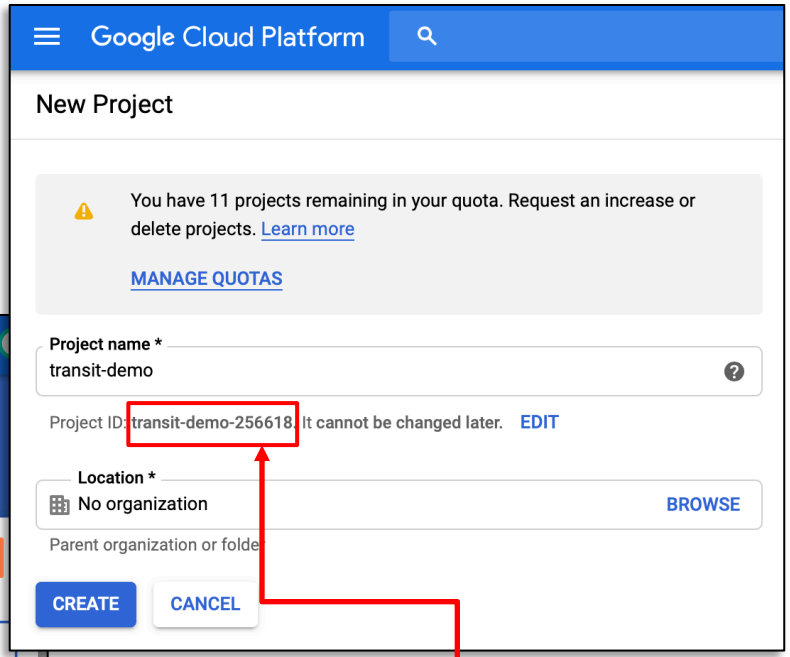
- Terraform builds 4 VM-Series & 2 spoke VPCs.
  - **vmseries01/02** handle inbound internet traffic from public LB.
  - **vmseries03/04** handle outbound & east/west traffic via separate internal LBs.
- spoke1-vpc has 2 web VMs with a frontend internal LB.
- spoke2-vpc has 1 Linux VM.
- Egress traffic flows over peering links with the VM-Series internal LB as the next hop.



PART 1
SETUP PROJECT

STEP 1. CREATE A PROJECT





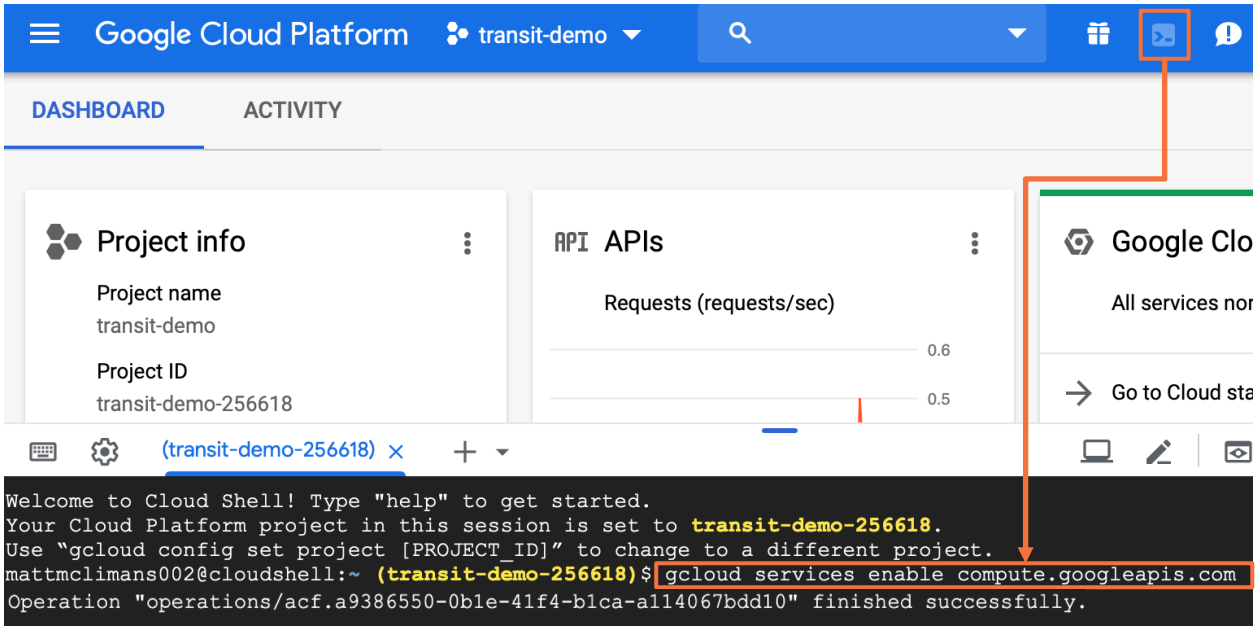
Copy the Project ID to a text editor

ALL REMAINING STEPS WILL BE PERFORMED INSIDE GCP CLOUD TERMINAL

STEP 2. ENABLE COMPUTE ENGINE API

1. Inside your project, open a cloud shell terminal
2. Enable the Compute Engine API

```
$ gcloud services enable compute.googleapis.com
```



STEP 3. CREATE SSH KEY & DOWNLOAD REPO

Create an SSH Key to enable console access to the VMs.

1. Create a SSH key (password protected key is optional)

```
$ ssh-keygen -f ~/.ssh/gcp-demo -t rsa -C gcpdemo
```

2. Copy the ***public*** key's full file path to a text editor

```
$ ls -l ~/.ssh/*
```

```
(transit-demo-256618)$ ls -l ~/.ssh/*
mattmclimans002 1675 Oct 21 15:01 /home/mattmclimans002/.ssh/gcp-demo
mattmclimans002  389 Oct 21 15:01 /home/mattmclimans002/.ssh/gcp-demo.pub
```

3. Clone the Github repo & change directories to adv_peering_4fw_2spoke/

```
$ git clone https://github.com/wwce/terraform; cd terraform/gcp/adv_peering_4fw_2spoke
```

STEP 4. UPDATE TERRAFORM.TFVARS

1. Open the terraform.tfvars file

   ```
   $ nano terraform.tfvars
   ```

2. Uncomment & set values for:
   - `project_id` (your project ID)
   - `public_key_path` (local path to public key)
   - `fw_panos` (license type and PAN-OS version)

3. Save file and exit (`ctrl-x` then `y`)

```
#project_id      = ""
#public_key_path = "~/.ssh/gcp-demo.pub"

#fw_panos        = "byol-904"
#fw_panos        = "bundle1-904"
#fw_panos        = "bundle2-904"
```

Your terraform.tfvars should look like this

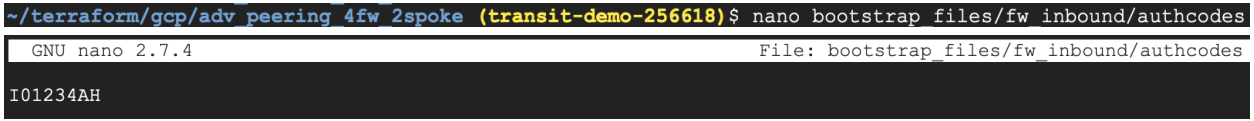
```
project_id      = "transit-demo-256618"
public_key_path = "~/.ssh/gcp-demo.pub"

#fw_panos        = "byol-904"
fw_panos         = "bundle1-904"
#fw_panos        = "bundle2-904"
```

(OPTIONAL) BYOL LICENSES ONLY

If using BYOL firewall licenses, the authorization codes must be typed into **/bootstrap_files/fw_inbound/authcodes** and **bootstrap_files/fw_outbound/authcodes.** **The firewalls can also be manually licensed post-deployment.**

1. BYOL for Inbound Firewalls

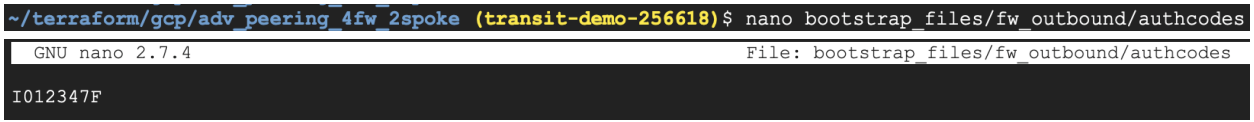
```
$ nano bootstrap_files/fw_inbound/authcodes
```



2. BYOL for Outbound Firewalls

```
$ nano bootstrap_files/fw_outbound/authcodes
```

3. Make sure both files are saved and the authcode is registered with the Palo Alto Networks support site before proceeding.



STEP 5. APPLY TERRAFORM

Initialize and apply the Terraform build (make sure you are in the `adv_peering_4fw_2spoke` directory).

```
$ terraform init
```

```
~/terraform/gcp/adv_peering_4fw_2spoke (transit-demo-256618)$ terraform init

* provider.google: version = "~> 2.17"
* provider.null: version = "~> 2.1"
* provider.random: version = "~> 2.2"

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.
```

```
$ terraform apply
```

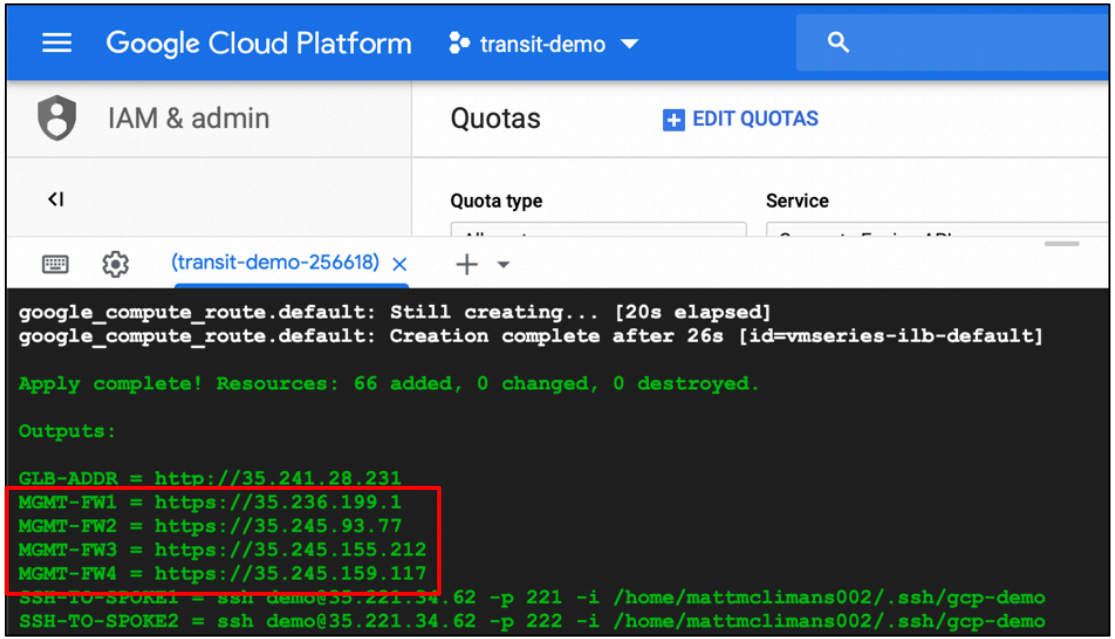
```
~/terraform/gcp/adv_peering_4fw_2spoke (transit-demo-256618)$ terraform apply

Plan: 66 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
```

PART 3
TEST ENVIRONMENT

Please wait 10-15 minutes
to allow the deployment to finish.

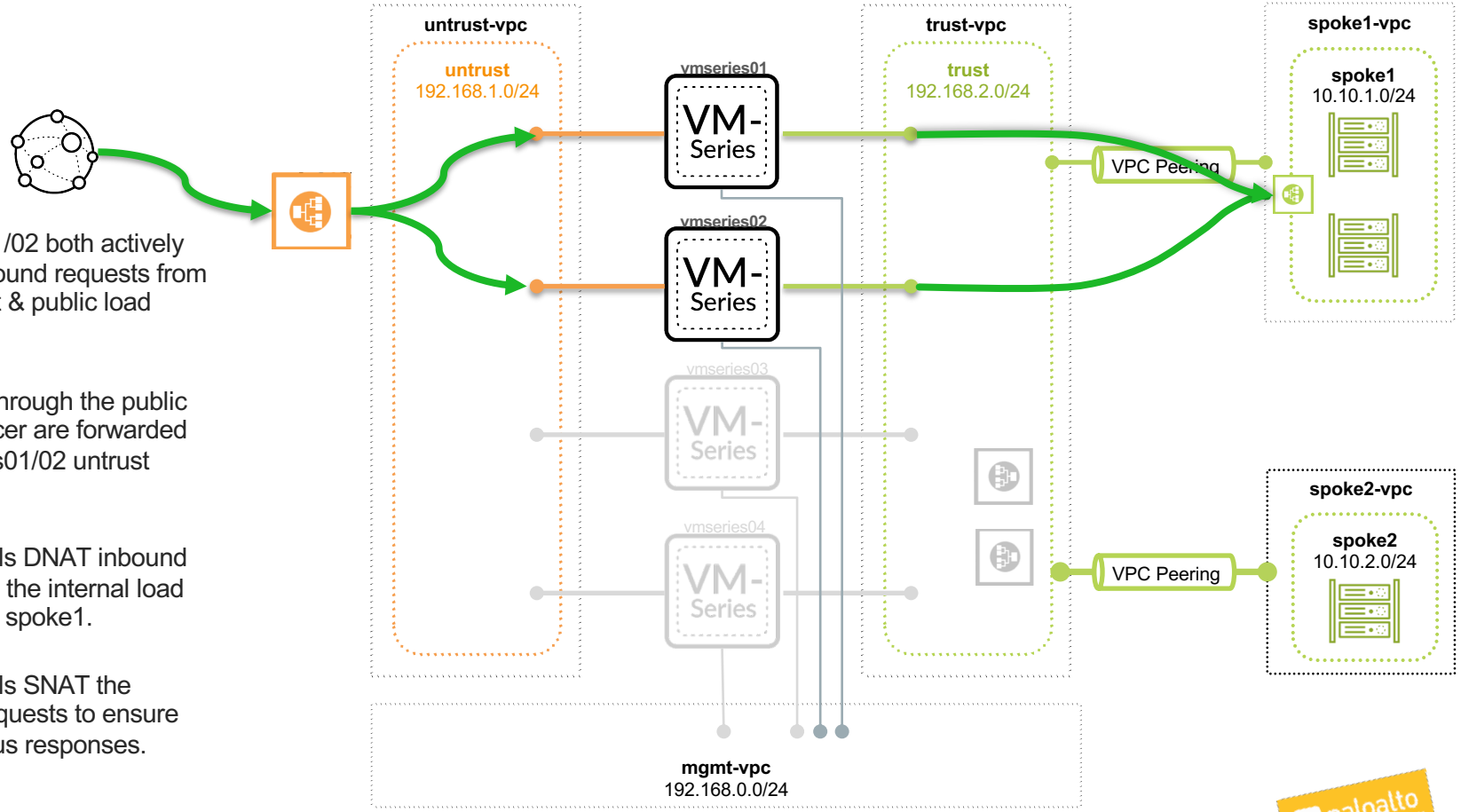
LOG INTO THE FIREWALLS

- From the Terraform output, into each firewall using the values in the `MGMT-FW[1-4]` outputs
  - **UN:** `paloalto`
  - **PW:** `Pal0Alto0@123`



INBOUND FLOW

- vmseries01/02 both actively handle inbound requests from the internet & public load balancer.
- Requests through the public load balancer are forwarded to vmseries01/02 untrust interfaces.
- The firewalls DNAT inbound requests to the internal load balancer in spoke1.
- The firewalls SNAT the inbound requests to ensure synchronous responses.

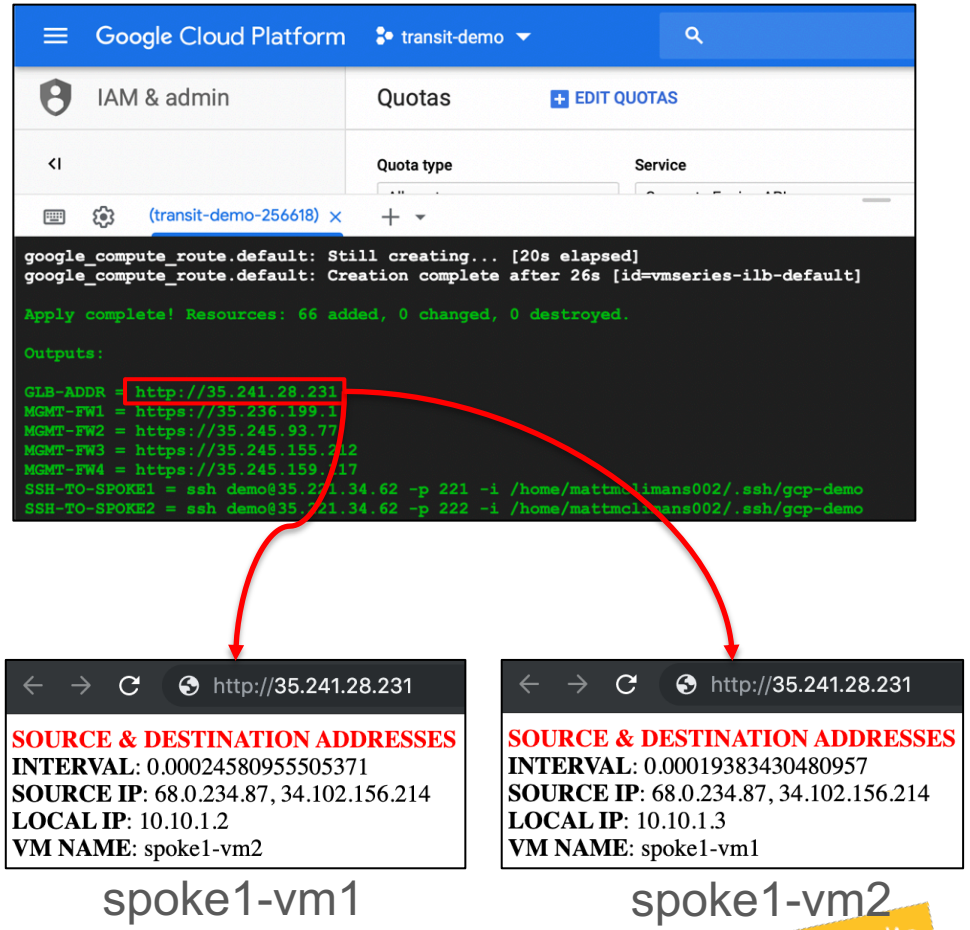


TEST INBOUND FLOW

1. Copy the **GLB-ADDR** output into a web browser.

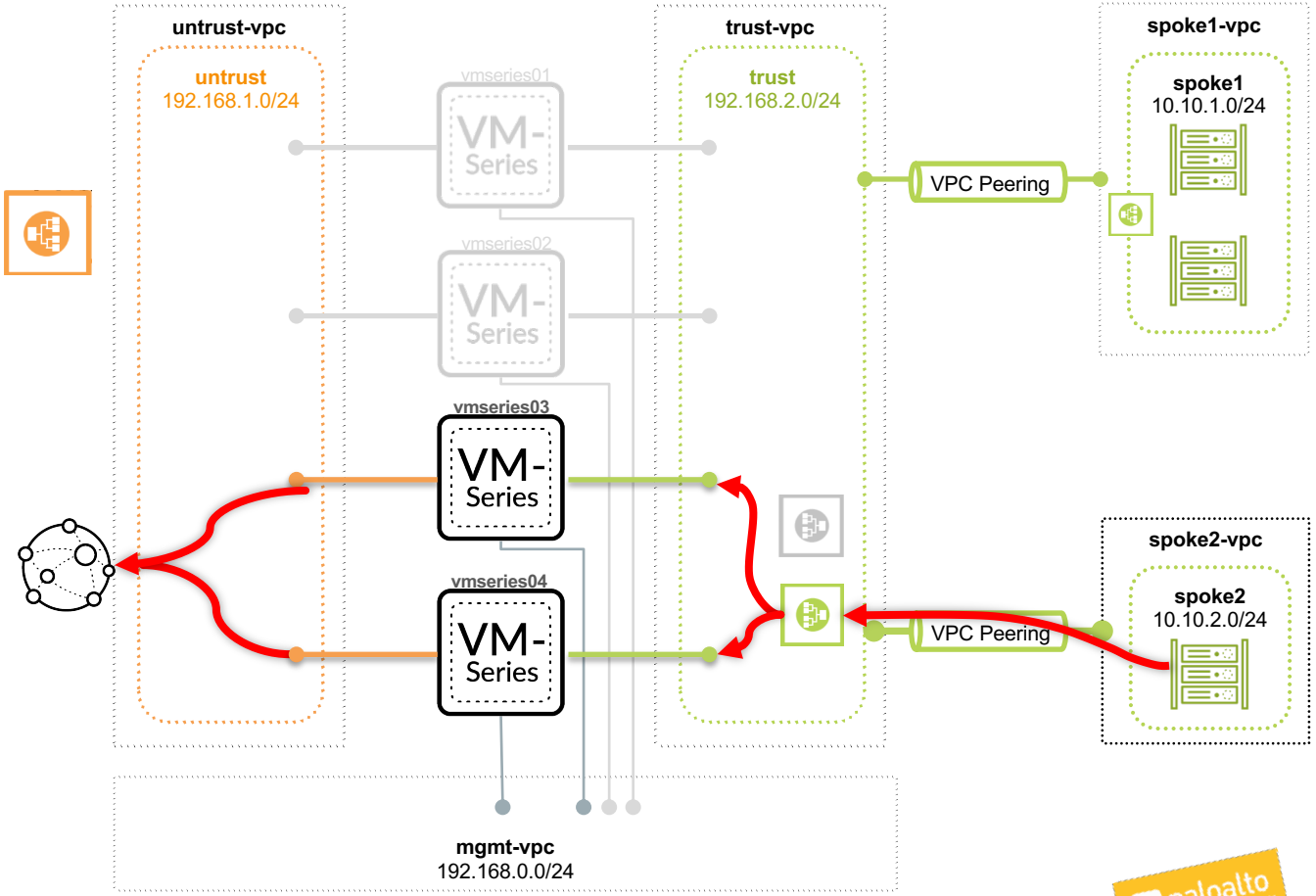
2. Observe inbound load balancing by refreshing the page. The value of **LOCAL IP** and **VM NAME** will change between the two web-servers in Spoke1.

3. View the firewall logs (MGMT-FW1, MGMT-FW2) to view the traffic.



OUTBOUND FLOW

- Over the peering link, the trust VPC exports a static default route to the spoke VPCs. The route uses the firewall's internal load balancer as its next hop.
- vmseries03 & 04 will both actively handle outbound requests to the internet.
- ***Note***: The internal load balancer will forward all TCP and UDP traffic. ICMP traffic is not supported (i.e. pings are not a good test since they will always fail).



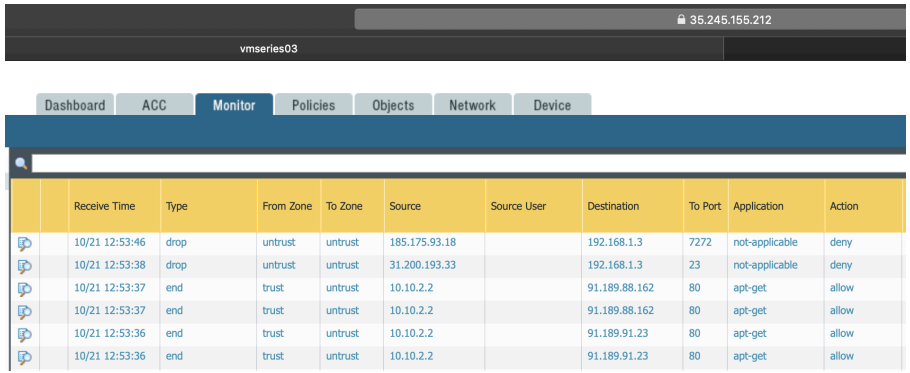
TEST OUTBOUND FLOW

1. Copy the **SSH-TO-SPOKE2** output and paste it into your terminal to log into the VM in spoke2

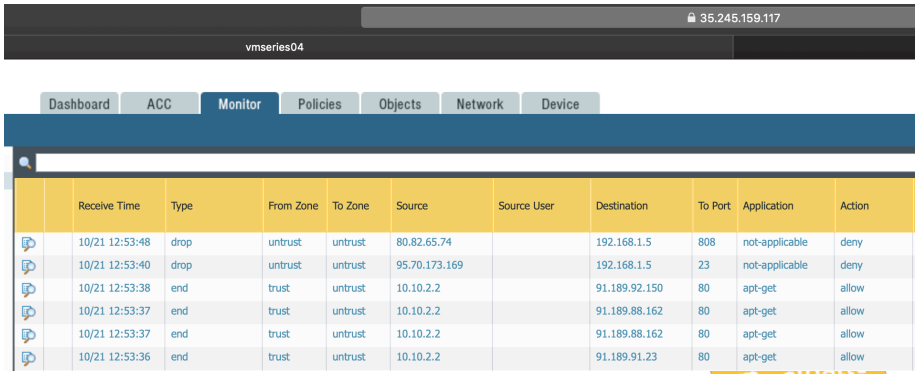
```
Outputs:

GLB-ADDR = http://35.241.28.231
MGMT-FW1 = https://35.236.199.1
MGMT-FW2 = https://35.245.93.77
MGMT-FW3 = https://35.245.155.212
MGMT-FW4 = https://35.245.159.117
SSH-TO-SPOKE1 = ssh demo@35.221.34.62 -p 221 -i /home/mattmclimans002/.ssh/gcp-demo
SSH-TO-SPOKE2 = ssh demo@35.221.34.62 -p 222 -i /home/mattmclimans002/.ssh/gcp-demo
(transit-demo-256618)$ ssh demo@35.221.34.62 -p 222 -i /home/mattmclimans002/.ssh/gcp-demo
```

2. From the VM, test egress internet connectivity by running **`$ sudo apt-get update`**

3. Log into vmseries03/04 to view the outbound traffic flow under **Monitor → Traffic.** You should see multiple apt-get requests through both firewalls.



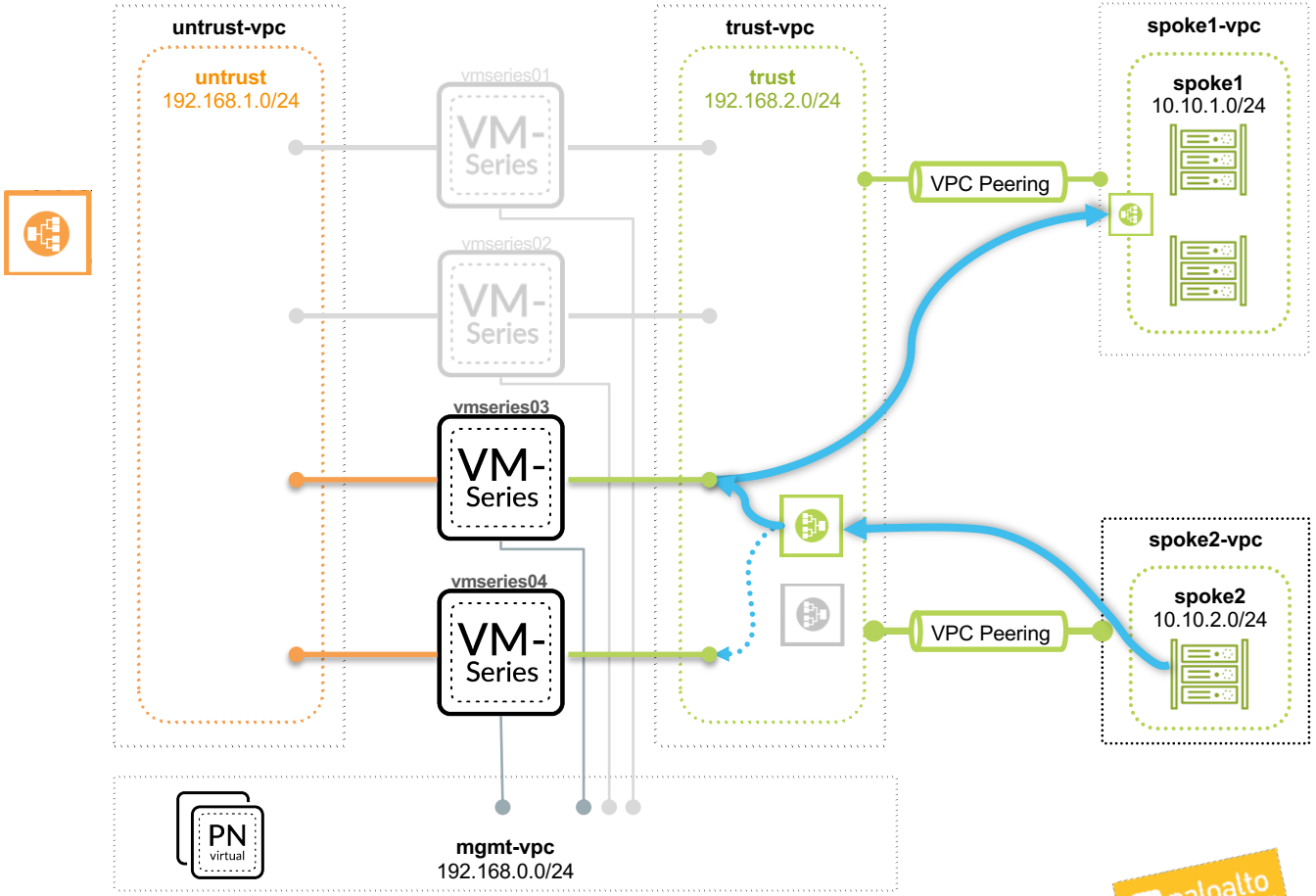
Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
10/21 12:53:46	drop	untrust	untrust	185.175.93.18		192.168.1.3	7272	not-applicable	deny
10/21 12:53:38	drop	untrust	untrust	31.200.193.33		192.168.1.3	23	not-applicable	deny
10/21 12:53:37	end	trust	untrust	10.10.2.2		91.189.88.162	80	apt-get	allow
10/21 12:53:37	end	trust	untrust	10.10.2.2		91.189.88.162	80	apt-get	allow
10/21 12:53:36	end	trust	untrust	10.10.2.2		91.189.91.23	80	apt-get	allow
10/21 12:53:36	end	trust	untrust	10.10.2.2		91.189.91.23	80	apt-get	allow



Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
10/21 12:53:48	drop	untrust	untrust	80.82.65.74		192.168.1.5	808	not-applicable	deny
10/21 12:53:40	drop	untrust	untrust	95.70.173.169		192.168.1.5	23	not-applicable	deny
10/21 12:53:38	end	trust	untrust	10.10.2.2		91.189.92.150	80	apt-get	allow
10/21 12:53:37	end	trust	untrust	10.10.2.2		91.189.88.162	80	apt-get	allow
10/21 12:53:37	end	trust	untrust	10.10.2.2		91.189.88.162	80	apt-get	allow
10/21 12:53:36	end	trust	untrust	10.10.2.2		91.189.91.23	80	apt-get	allow

EAST-WEST FLOW

- Over the peering link, the trust VPC exports a static 10.10.0.0/16 route to the spoke VPCs. The route uses the firewall's internal load balancer as its next hop.
- vmseries03 handles all east/west traffic. If vmseries03 fails, the internal load balancer will reroute east/west traffic to vmseries04.
- ***Note***: Both firewalls can actively handle east/west traffic if a SNAT policy is applied. SNAT is required to ensure synchronous responses.

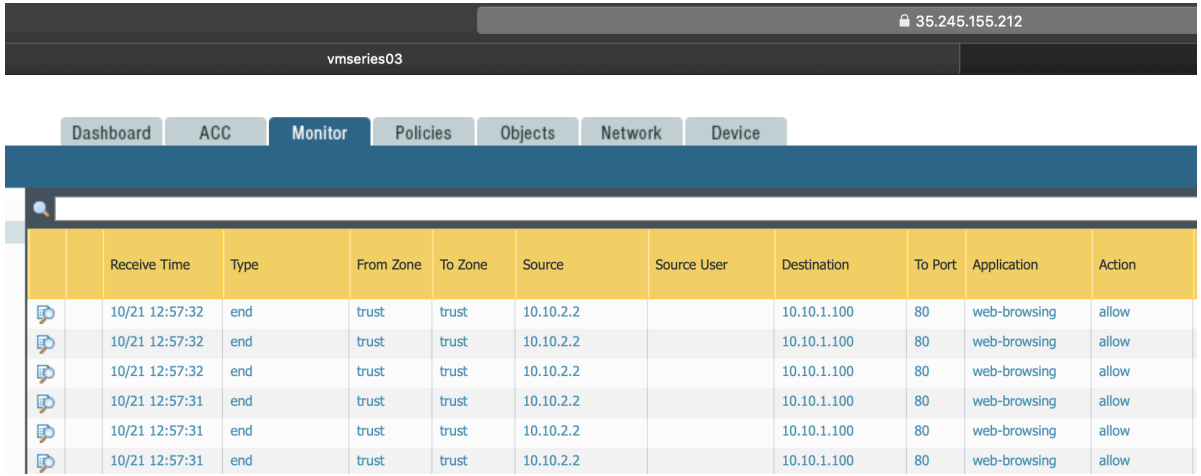


TEST EAST-WEST FLOW

1. Stay inside Spoke2-VM.
2. Run a curl command to the web server's internal load balancer in Spoke1

```
$ curl http://10.10.1.100/?[1-1000]
```

3. View the logs on FW3. The request from Spoke2 to Spoke1 will flow only through FW3 because we are leveraging Active/Passive backend pools.



		Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
		10/21 12:57:32	end	trust	trust	10.10.2.2		10.10.1.100	80	web-browsing	allow
		10/21 12:57:32	end	trust	trust	10.10.2.2		10.10.1.100	80	web-browsing	allow
		10/21 12:57:32	end	trust	trust	10.10.2.2		10.10.1.100	80	web-browsing	allow
		10/21 12:57:31	end	trust	trust	10.10.2.2		10.10.1.100	80	web-browsing	allow
		10/21 12:57:31	end	trust	trust	10.10.2.2		10.10.1.100	80	web-browsing	allow
		10/21 12:57:31	end	trust	trust	10.10.2.2		10.10.1.100	80	web-browsing	allow

DESTROY ENVIRONMENT

DESTROY BUILD

Perform the following to destroy the deployment.

1. Verify you are in the `adv_peering_4fw_2spoke` directory and run the following. Enter `yes` to confirm.

```
$ terraform destroy
```

```
~/terraform/gcp/adv_peering_4fw_2spoke (transit-demo-256618)$ terraform destroy

Plan: 0 to add, 0 to change, 65 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```

2. Once the destroy is complete, you will receive the following output:

```
module.vpc_untrust.google_compute_network.default: Still destroying... [id=untrust, 20s elapsed]
module.vpc_trust.google_compute_network.default: Still destroying... [id=trust, 20s elapsed]
module.vpc_untrust.google_compute_network.default: Destruction complete after 26s
module.vpc_trust.google_compute_network.default: Destruction complete after 26s

Destroy complete! Resources: 65 destroyed.
```

DEPLOYMENT CHEAT SHEET

Part 1: Setup Project

```
$ gcloud services enable compute.googleapis.com
$ ssh-keygen -f ~/.ssh/<keyname> -t rsa -C <comment>
$ git clone https://github.com/wwce/terraform; cd terraform/gcp/adv_peering_4fw_2spoke
```

Part 2: Run Build

Edit terraform.tfvars to match project ID, SSH Key, and PAN-OS version and license.

```
$ terraform init
$ terraform apply
```

Part 3: Destroy Build

```
$ terraform destroy
```

