

GCP Transit VPC

VPC Peering & Internal Load Balancer as Next Hop

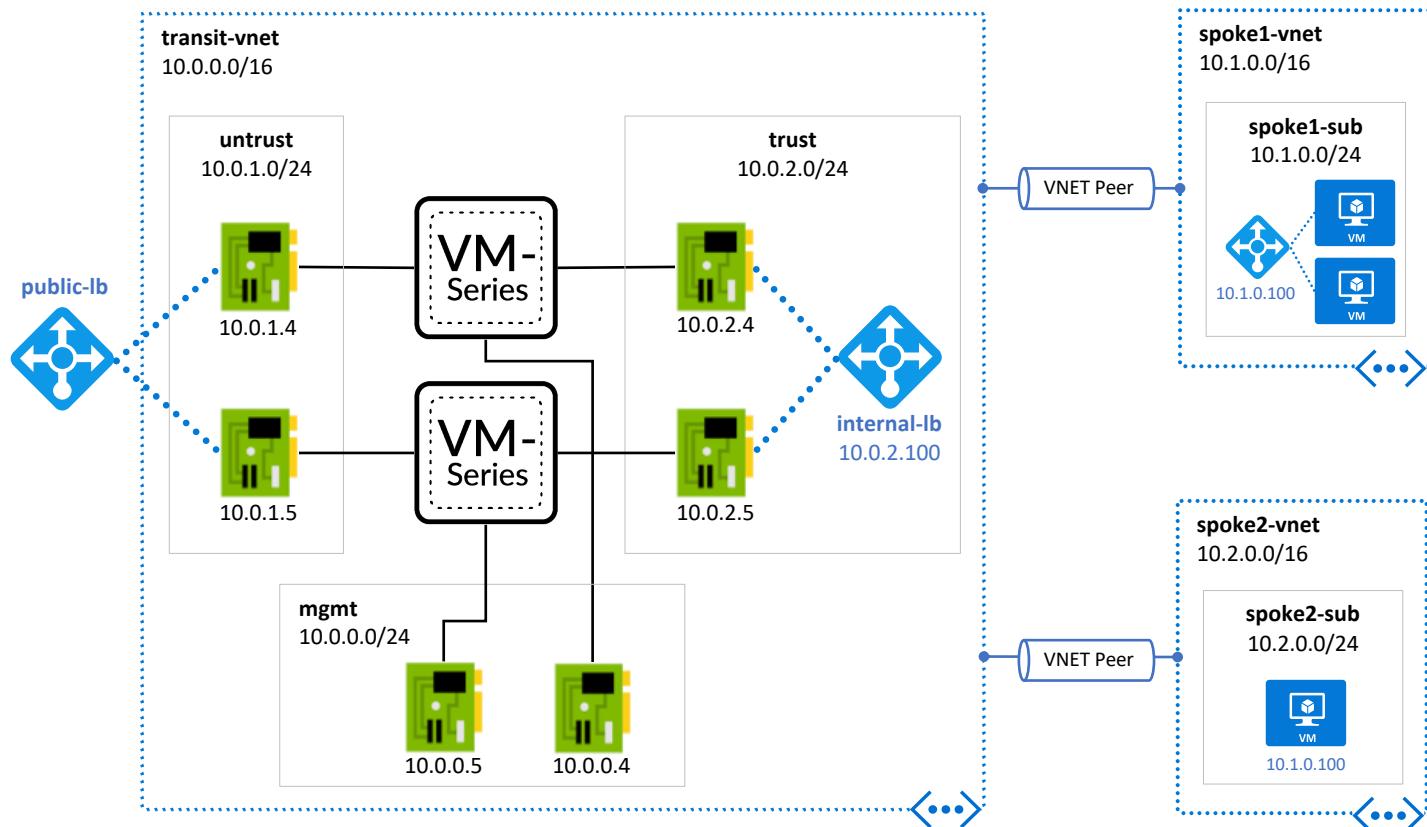
Matt McLimans
Public Cloud Consultant Engineer



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.

- VM-Series deployed between public & internal load balancers to secure spoke VNET traffic:
 - Inbound
 - Outbound
 - East-west
- The spoke VNETs connect to the transit VNET via peering.
- Spoke1 contains Web VMs frontended by an internal load balancer.
- Spoke2 contains a Linux VM.



Requirements

1. A valid Azure subscription
2. A valid VM-Series authcode (if using BYOL)

PART 1

BUILD ENVIRONMENT

Step 1. Open Azure Cloud Shell

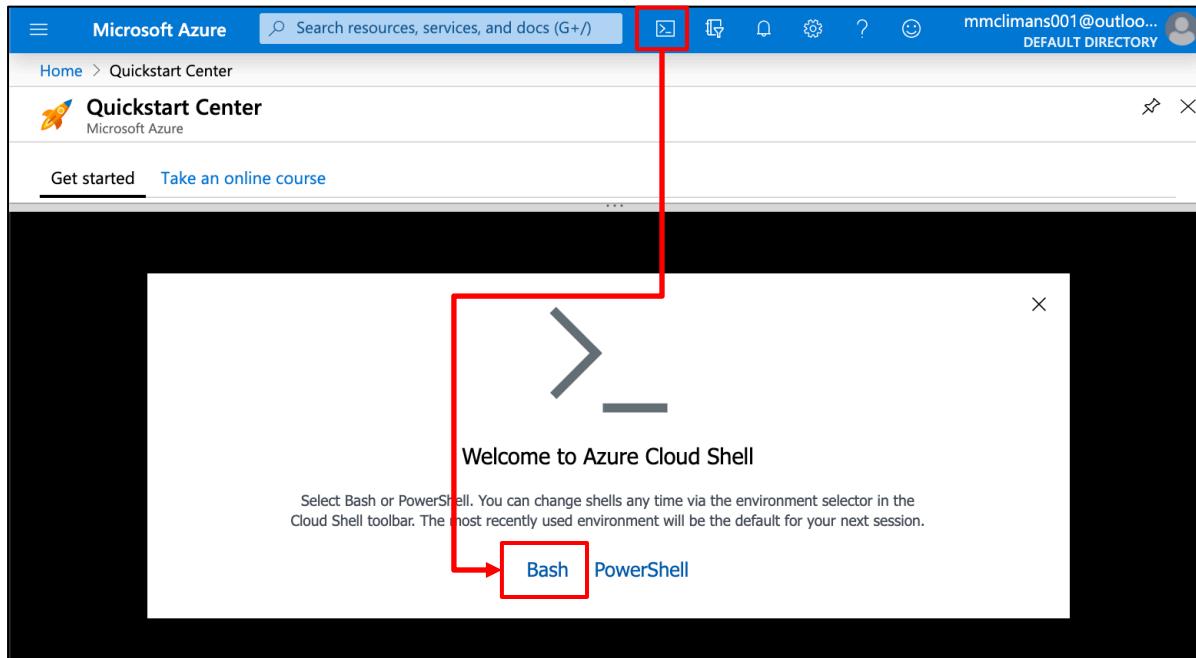
1. Log into Azure Portal

<https://portal.azure.com>

2. Click  to open Cloud Shell.

3. Select **Bash**

Note, you will be prompted to create a storage account if you do not have one.



Step 2. Accept VM-Series EULA

You must accept Marketplace EULAs for the desired VM-Series license SKU:

1. BYOL

```
az vm image terms accept --urn paloaltonetworks:vmseries1:byol:9.0.1
```

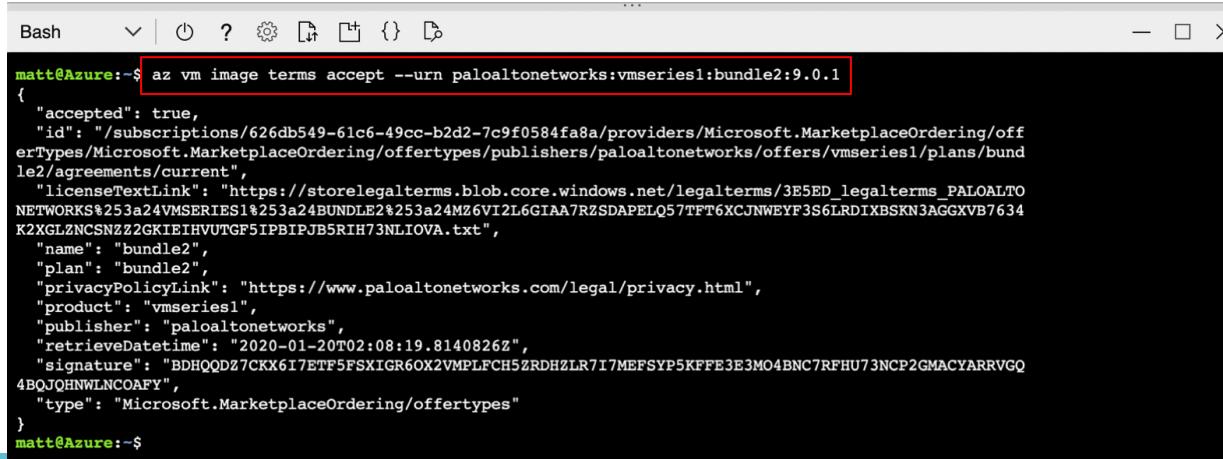
2. Bundle1

```
az vm image terms accept --urn paloaltonetworks:vmseries1:bundle1:9.0.1
```

3. Bundle2

```
az vm image terms accept --urn paloaltonetworks:vmseries1:bundle2:9.0.1
```

For the desired offering/license, copy & paste the command it into your cloud shell.



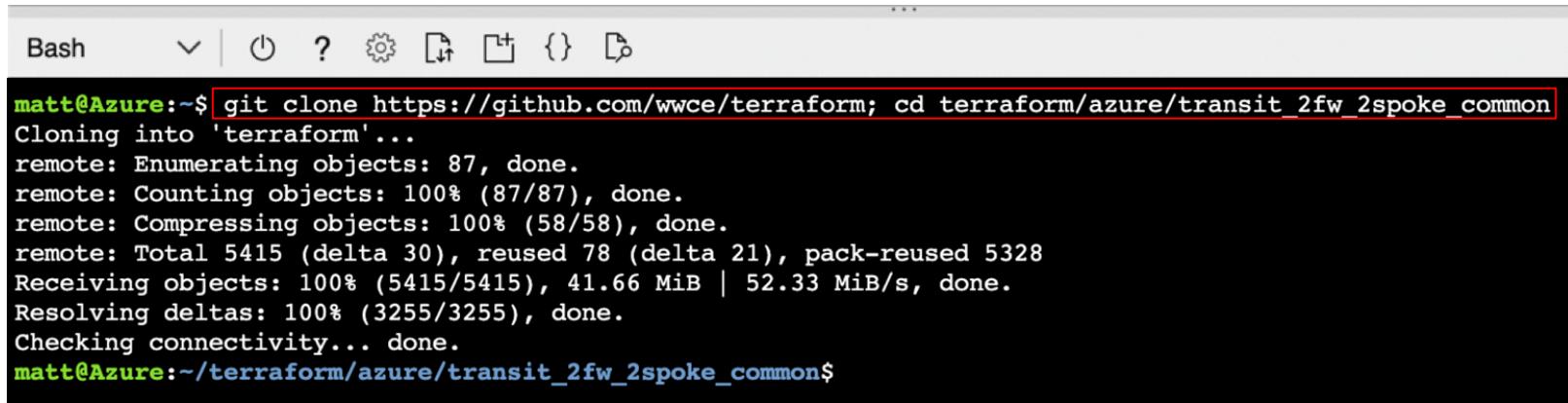
```
matt@Azure:~$ az vm image terms accept --urn paloaltonetworks:vmseries1:bundle2:9.0.1
{
    "accepted": true,
    "id": "/subscriptions/626db549-61c6-49cc-b2d2-7c9f0584fa8a/providers/Microsoft.MarketplaceOrdering/offertypes/Microsoft.MarketplaceOrdering/offertypes/publishers/paloaltonetworks/offers/vmseries1/plans/bundle2/agreements/current",
    "licenseTextLink": "https://storelegalterms.blob.core.windows.net/legalterms/3E5ED_legalterms_PALOALTONETWORKS%253a%24VMSERIES1%253a%24BUNDLE2%253a%24MZ6V12L6GIAA7RZSDAPELQ57TFT6XCJNWEYF3S6LRDIXBSKN3AGGXVB7634K2XGLZNCSNZ22GKIEIHVUTGF5IPBIPJB5RIH73NLIOVA.txt",
    "name": "bundle2",
    "plan": "bundle2",
    "privacyPolicyLink": "https://www.paloaltonetworks.com/legal/privacy.html",
    "product": "vmseries1",
    "publisher": "paloaltonetworks",
    "retrieveDatetime": "2020-01-20T02:08:19.8140826Z",
    "signature": "BDHQDZ7CKX617ETF5FSXIGR60X2VMPLFCH5ZRDHZLR7I7MEFSYP5KFFE3E3M04BNC7RFHU73NCP2GMACYARRVGQ4BQJQHNWLNOAFY",
    "type": "Microsoft.MarketplaceOrdering/offertypes"
}
```



Step 3. Download Github Repo

1. Clone the Github repo & change directories to transit_2fw_2spoke_common

```
$ git clone https://github.com/wwce/terraform; cd terraform/azure/transit_2fw_2spoke_common
```



The screenshot shows a terminal window titled "Bash". The command `git clone https://github.com/wwce/terraform; cd terraform/azure/transit_2fw_2spoke_common` is entered and highlighted with a red border. The terminal then displays the progress of the cloning process, including object enumeration, counting, compressing, and receiving objects, followed by a successful resolution of deltas and connectivity check.

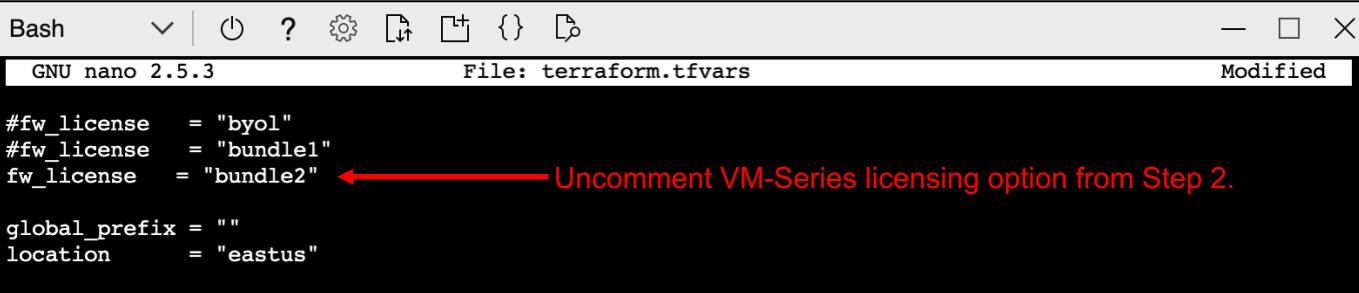
```
matt@Azure:~$ git clone https://github.com/wwce/terraform; cd terraform/azure/transit_2fw_2spoke_common
Cloning into 'terraform'...
remote: Enumerating objects: 87, done.
remote: Counting objects: 100% (87/87), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 5415 (delta 30), reused 78 (delta 21), pack-reused 5328
Receiving objects: 100% (5415/5415), 41.66 MiB | 52.33 MiB/s, done.
Resolving deltas: 100% (3255/3255), done.
Checking connectivity... done.
matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$
```

Step 4. Update terraform.tfvars

1. Open the terraform.tfvars file

```
$ nano terraform.tfvars
```

2. Uncomment **only one** value for **fw_license** (delete # to uncomment)



Bash GNU nano 2.5.3 File: terraform.tfvars Modified

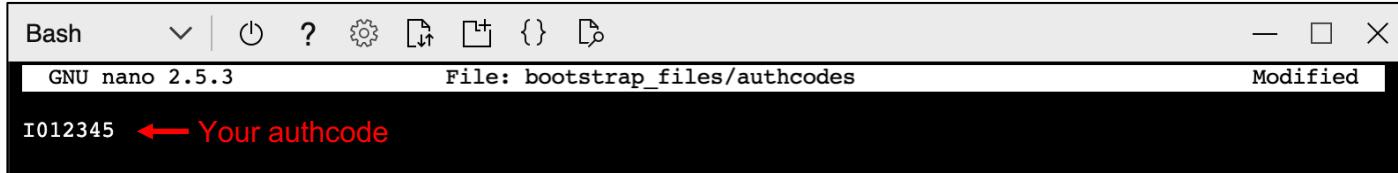
```
#fw_license = "byol"
#fw_license = "bundle1"
fw_license = "bundle2" ← Uncomment VM-Series licensing option from Step 2.
global_prefix =
location = "eastus"
```

3. Save file and exit: **ctrl-x** → **y** → **enter**

BYOL ONLY

- You can manually license the firewalls post deployment.
- If you want to license the firewalls during deployment (via bootstrapping):
 1. Ensure authcode is registered with the Palo Alto Networks support site.
 2. Open bootstrap_files/authcodes & enter your authcode.

```
$ nano bootstrap_files/authcodes
```



A screenshot of a terminal window titled "Bash". The title bar includes icons for a dropdown menu, power, help, settings, copy, paste, brace matching, and close. The status bar shows "GNU nano 2.5.3", "File: bootstrap_files/authcodes", and "Modified". The main area of the terminal contains the command "\$ nano bootstrap_files/authcodes" followed by the text "I012345" with a red arrow pointing to it. The terminal has a dark background with white text and a light gray header.

3. Save file and exit: **ctrl-x** → **y** → **enter**

STEP 5. DEPLOY BUILD

Initialize and apply the Terraform build

```
$ terraform init
```

```
matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$ 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.

matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$
```

```
$ terraform apply
```

```
matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$ terraform apply

Plan: 71 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.

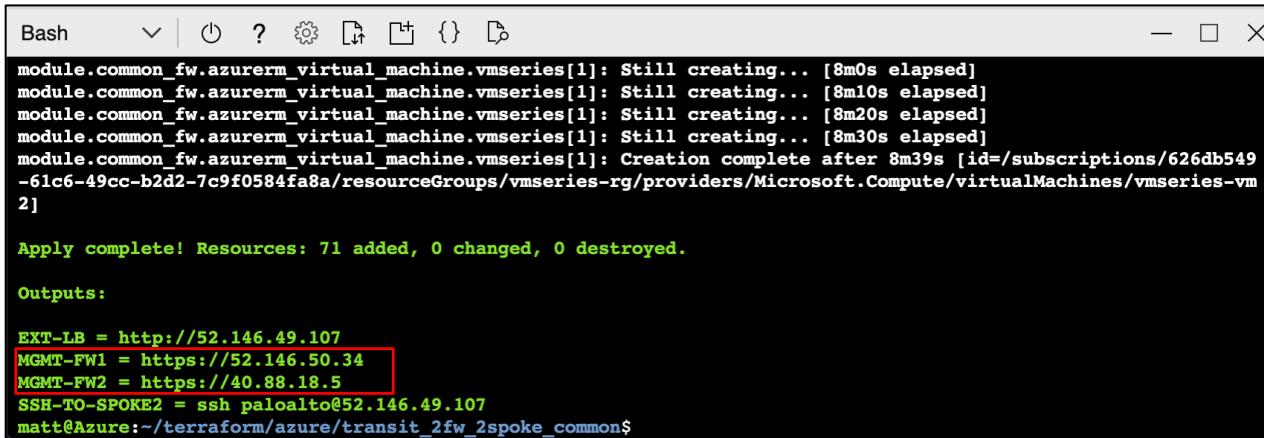
Enter a value: yes ← Enter yes to start build
```



Step 6. Build Completion & Firewall Login

- Once the build completes, the following output will be displayed

It may take up to 15 minutes for the firewalls to fully boot.



The screenshot shows a terminal window titled "Bash". The terminal displays the output of a Terraform apply command. It shows the creation of a virtual machine named "vmseries[1]" over a period of approximately 8 minutes. Once complete, it lists 71 resources added, 0 changed, and 0 destroyed. The "Outputs:" section provides URLs for various components: EXT-LB (http://52.146.49.107), MGMT-FW1 (https://52.146.50.34), MGMT-FW2 (https://40.88.18.5), and SSH-TO-SPOKE2 (ssh paloalto@52.146.49.107). The user "matt" is at the prompt in the Azure terraform directory.

```
Bash      ~ | ⌂ ? ⚙️ ⌂ ⌂ {} ⌂
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m0s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m10s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m20s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m30s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Creation complete after 8m39s [id=/subscriptions/626db549-61c6-49cc-b2d2-7c9f0584fa8a/resourceGroups/vmseries-rg/providers/Microsoft.Compute/virtualMachines/vmseries-vm2]

Apply complete! Resources: 71 added, 0 changed, 0 destroyed.

Outputs:

EXT-LB = http://52.146.49.107
MGMT-FW1 = https://52.146.50.34
MGMT-FW2 = https://40.88.18.5
SSH-TO-SPOKE2 = ssh paloalto@52.146.49.107
matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$
```

- Copy the **MGMT-FW1** & **MGMT-FW2** outputs into a web-browser.
 - UN:** paloalto
 - PW:** Pal0Alt0@123

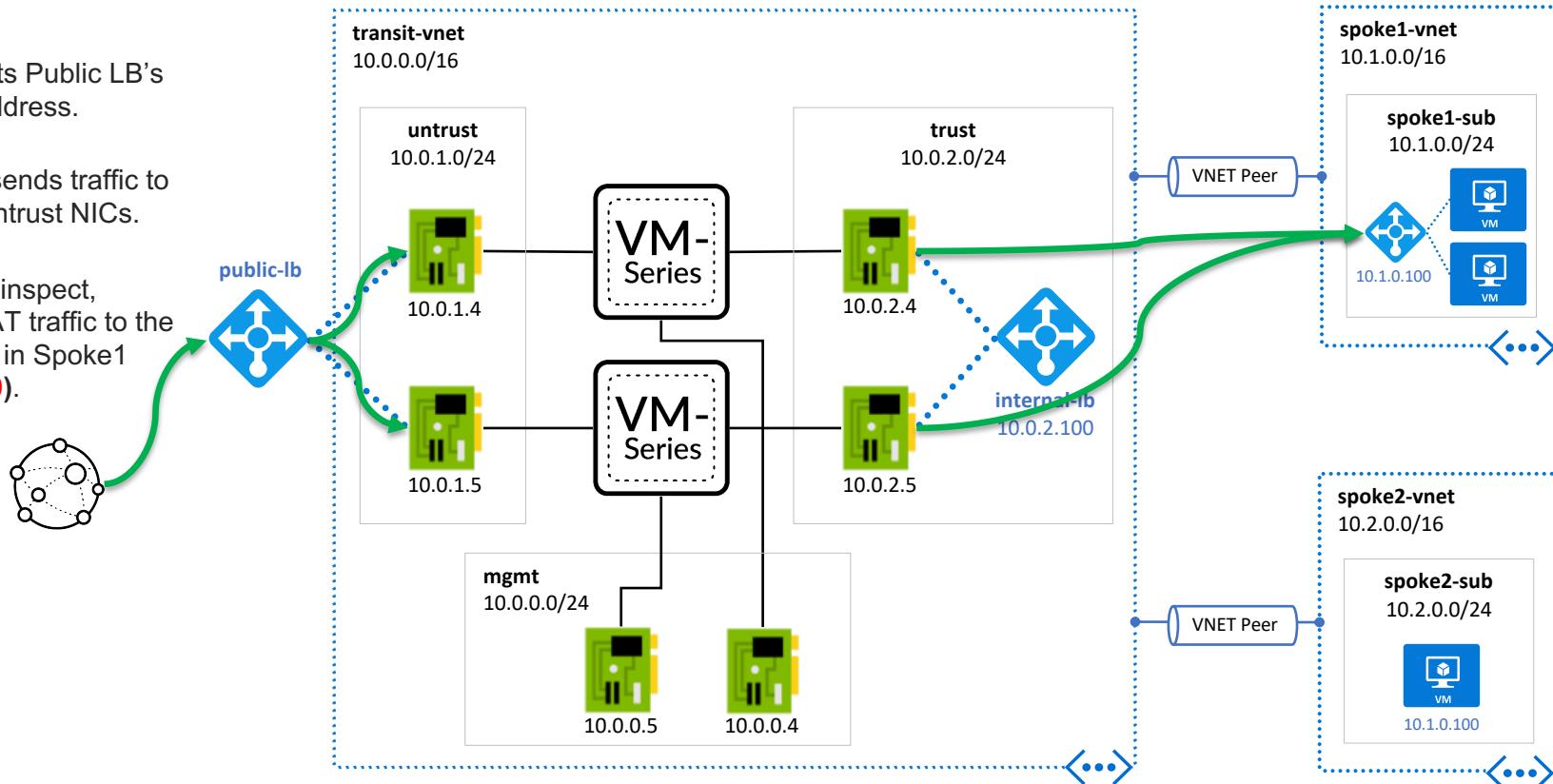


PART 2

TEST ENVIRONMENT

Inbound Flow

1. Request hits Public LB's frontend address.
2. Public LB sends traffic to firewall's untrust NICs.
3. VM-Series inspect, SNAT/DNAT traffic to the internal LB in Spoke1 (**10.1.0.100**).



Inbound Flow Test

1. Copy the **EXT-LB** output into a web browser.
2. Observe inbound load balancing by refreshing the page. The following values will change:

SOURCE IP

- FW trust IP that is handling request.

LOCAL IP

- IP Address of the VM in Spoke1 that received the request.

```
Bash ? { } x
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m0s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m10s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m20s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Still creating... [8m30s elapsed]
module.common_fw.azurerm_virtual_machine.vmseries[1]: Creation complete after 8m39s [id=/subscriptions/626db549-61c6-49cc-b2d2-7c9f0584fa8a/resourceGroups/vmseries-rg/providers/Microsoft.Compute/virtualMachines/vmseries-vm2]

Apply complete! Resources: 71 added, 0 changed, 0 destroyed.

Outputs:
EXT-LB = http://52.146.49.107
MGMT-FW1 = https://52.146.50.34
MGMT-FW2 = https://10.88.18.5
SSH-TO-SPOKE2 = ssh paloalto@52.146.49.107
matt@Azure:~/terraform/azure/transit_zin_2spoke_commons
```

The image shows two side-by-side browser windows. Both windows have the address bar set to `http://52.146.49.107`. The top window's title bar says "52.146.49.107". The bottom window's title bar also says "52.146.49.107".

The left browser window (labeled "spoke1-vm1" below it) displays the following information:
SOURCE & DESTINATION ADDRESSES
INTERVAL: 0.00023484230041504
SOURCE IP: 10.0.2.4
LOCAL IP: 10.1.0.5
VM NAME: spoke1-vm1

HEADER INFORMATION
HTTP_HOST: 52.146.49.107
HTTP_CONNECTION: keep-alive
HTTP_CACHE_CONTROL: max-age=0
HTTP_UPGRADE_INSECURE_REQUESTS: 1

The right browser window (labeled "spoke1-vm2" below it) displays the following information:
SOURCE & DESTINATION ADDRESSES
INTERVAL: 0.00018501281738281
SOURCE IP: 10.0.2.5
LOCAL IP: 10.1.0.4
VM NAME: spoke1-vm2

HEADER INFORMATION
HTTP_HOST: 52.146.49.107
HTTP_CONNECTION: keep-alive
HTTP_CACHE_CONTROL: max-age=0
HTTP_UPGRADE_INSECURE_REQUESTS: 1

spoke1-vm1

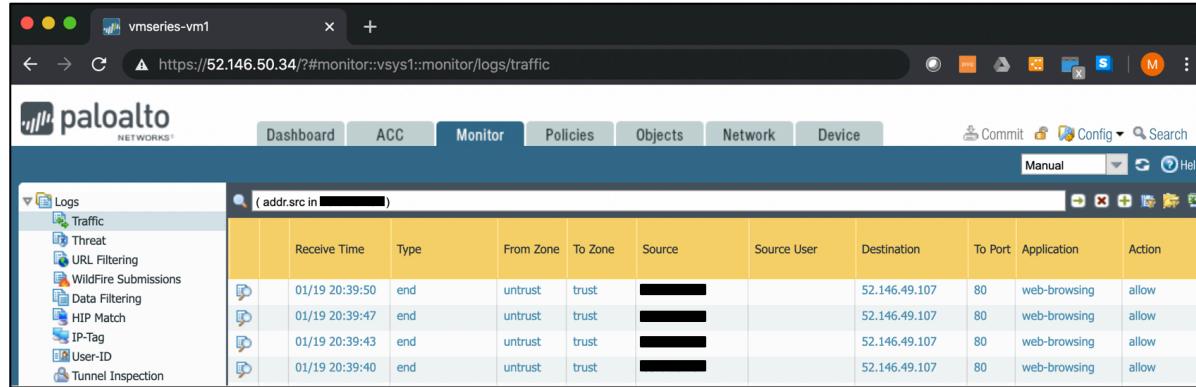
spoke1-vm2



Inbound Flow Test

- View the inbound traffic on both firewalls (Monitor → Traffic)

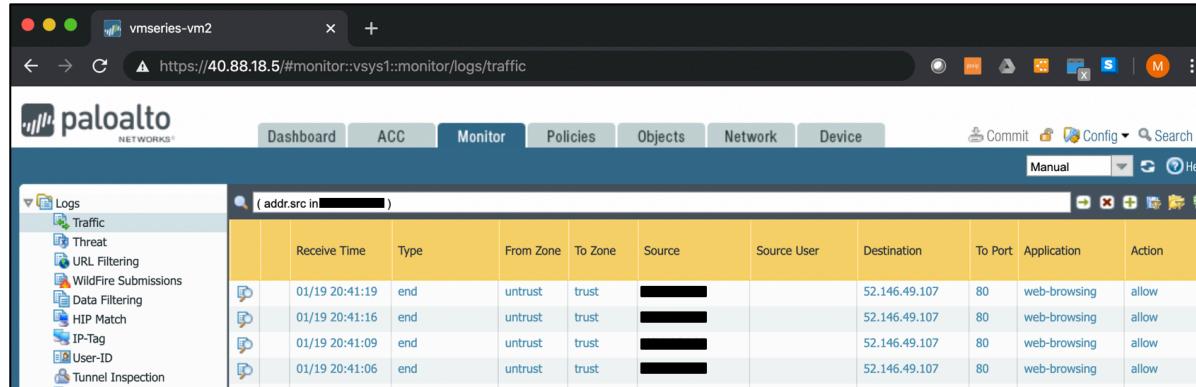
vmseries01



The screenshot shows the Palo Alto Networks Firewall interface for vmseries-vm1. The left sidebar has a 'Logs' section with 'Traffic' selected. The main area displays a table of traffic logs with the following columns: Receive Time, Type, From Zone, To Zone, Source, Source User, Destination, To Port, Application, and Action. The logs show four entries from 'untrust' to 'trust' to '52.146.49.107' port 80, application 'web-browsing', and action 'allow'. A search bar at the top right contains the query '(addr.src in [REDACTED])'.

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
01/19 20:39:50	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow
01/19 20:39:47	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow
01/19 20:39:43	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow
01/19 20:39:40	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow

vmseries02

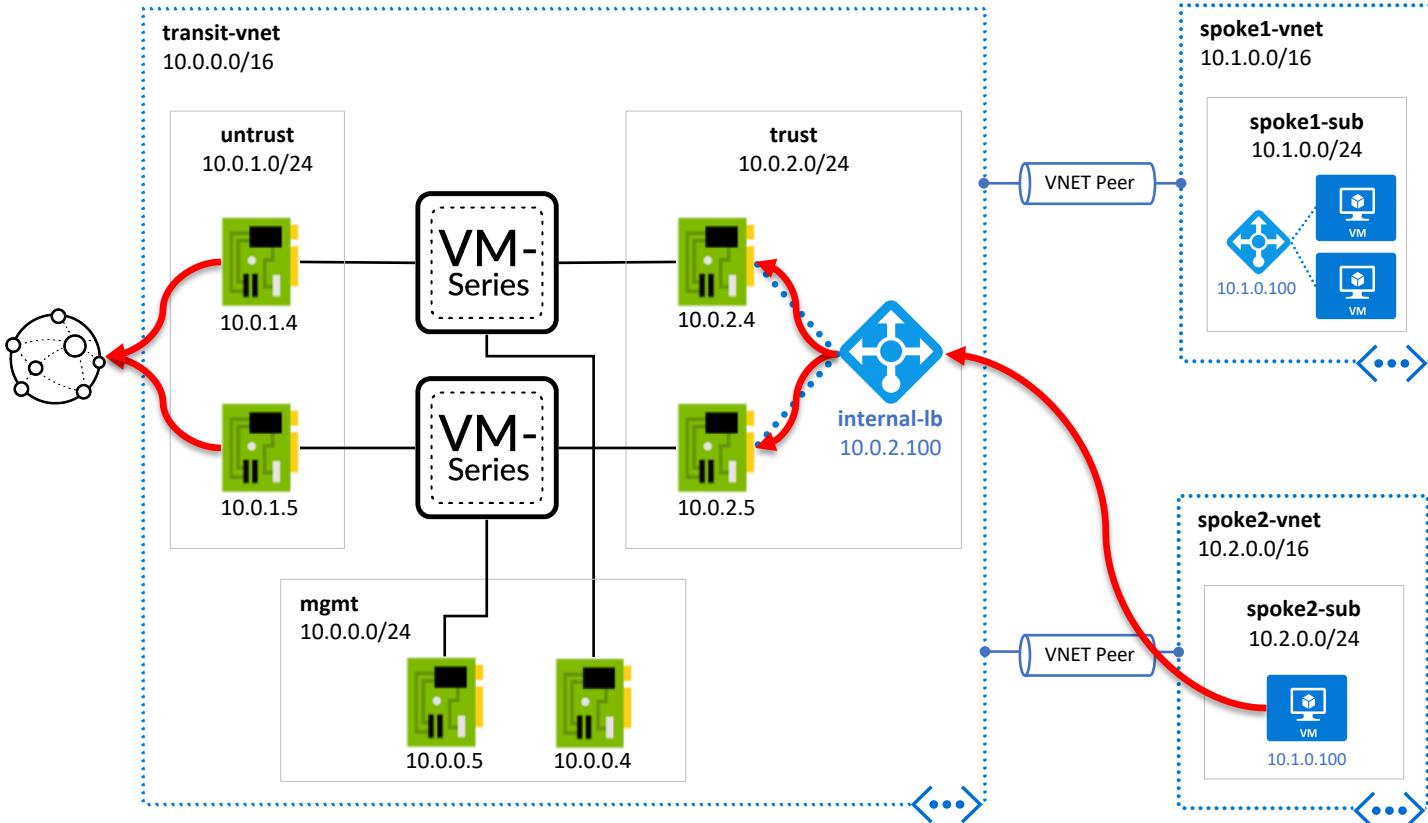


The screenshot shows the Palo Alto Networks Firewall interface for vmseries-vm2. The left sidebar has a 'Logs' section with 'Traffic' selected. The main area displays a table of traffic logs with the following columns: Receive Time, Type, From Zone, To Zone, Source, Source User, Destination, To Port, Application, and Action. The logs show four entries from 'untrust' to 'trust' to '52.146.49.107' port 80, application 'web-browsing', and action 'allow'. A search bar at the top right contains the query '(addr.src in [REDACTED])'.

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
01/19 20:41:19	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow
01/19 20:41:16	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow
01/19 20:41:09	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow
01/19 20:41:06	end	untrust	trust	[REDACTED]		52.146.49.107	80	web-browsing	allow

Outbound Flow

1. VM in spoke2-subnet sends a request to internet.
2. UDR within spoke2's route table sends traffic to internal LB **10.0.2.100**.
3. LB forwards traffic to firewall's trust interfaces.
4. VM-Series inspects, SNATs to untrust interface.
5. Untrust NIC sends traffic out attached public IP to internet.

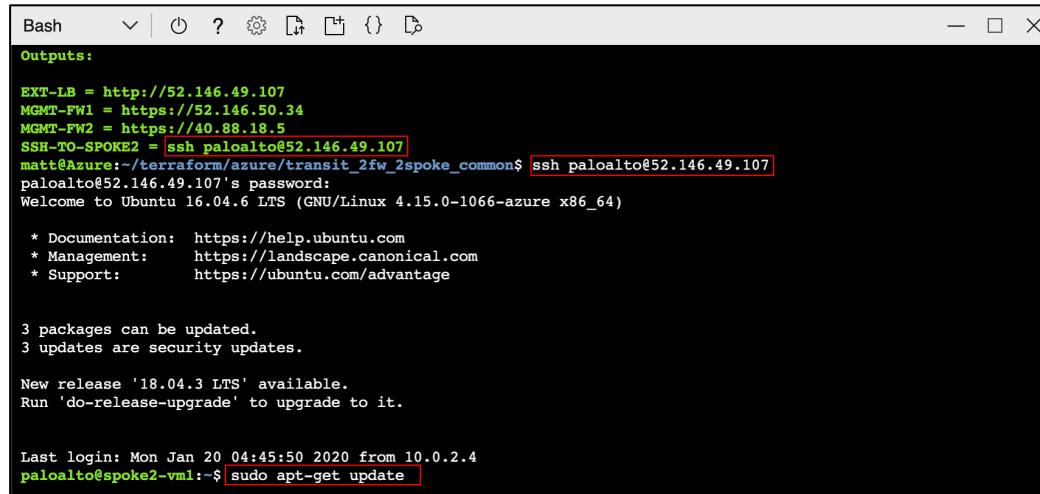


Outbound Flow Test

1. Log into spoke2-vm. Copy the **SSH-T0-SPOKE2** output and paste it into your terminal
2. From the spoke2-vm, test outbound connectivity:

```
$ sudo apt-get update
```

3. View the outbound traffic on both firewalls
(Monitor → Traffic)



```
Bash | ? | ⌂ | Outputs:  
EXT-LB = http://52.146.49.107  
MGMT-FW1 = https://52.146.50.34  
MGMT-FW2 = https://40.88.18.5  
SSH-T0-SPOKE2 = ssh paloalto@52.146.49.107  
matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$ ssh paloalto@52.146.49.107  
paloalto@52.146.49.107's password:  
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1066-azure x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
3 packages can be updated.  
3 updates are security updates.  
  
New release '18.04.3 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Jan 20 04:45:50 2020 from 10.0.2.4  
paloalto@spoke2-vm1:~$ sudo apt-get update
```

vmseries01



	Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
1	01/19 20:55:36	end	trust	untrust	10.2.0.4		52.168.50.79	80	apt-get	allow
2	01/19 20:55:21	end	trust	untrust	10.2.0.4		52.168.50.79	80	apt-get	allow

vmseries02

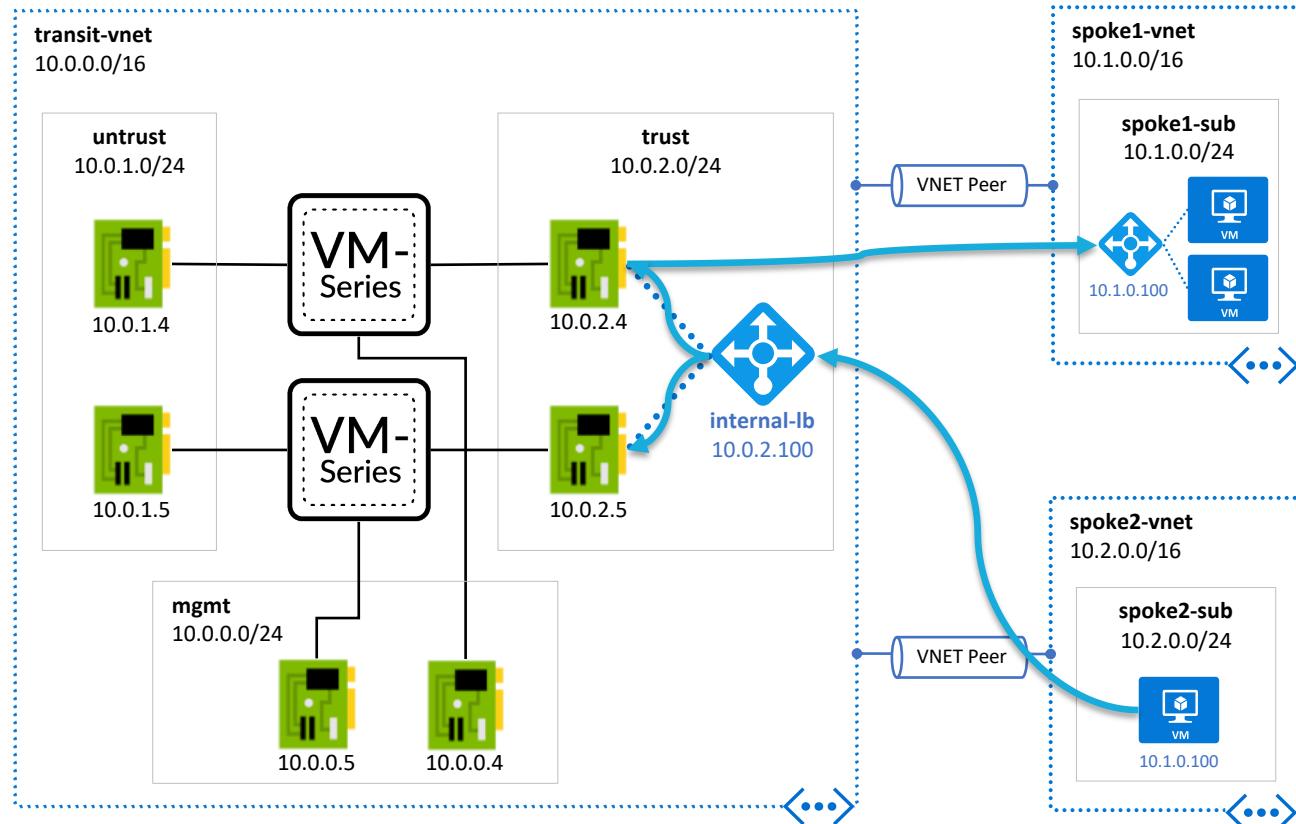
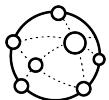


	Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
1	01/19 20:46:18	end	trust	untrust	10.2.0.4		91.189.88.174	80	apt-get	allow
2	01/19 20:46:18	end	trust	untrust	10.2.0.4		52.168.50.79	80	apt-get	allow



East-West Flow

1. VM in spoke2-subnet sends a request to VM in spoke1-subnet.
2. UDR within spoke2's route table sends traffic to internal LB **10.0.2.100**.
3. LB forwards traffic to firewall's trust interfaces.
4. VM-Series inspects and routes traffic to spoke1-subnet.
5. Response back from spoke1-subnet takes same path back.



East-West Flow Test

1. Stay inside spoke2-vm1 (**10.2.0.4**)
2. Run a curl command to the web server's internal load balancer in spoke1 (**10.1.0.100**)

```
$ curl http://10.1.0.100/?[1-1000]
```

```
paloalto@spoke2-vm1:~$ curl http://10.1.0.100/?[1-1000]
```

3. View the east-west traffic on the firewalls (Monitor → Traffic)

Logs

Traffic

Threat

URL Filtering

Wildfire Submissions

Data Filtering

HIP Match

IP-Tag

User-ID

Tunnel Inspection

(addr[src in 10.2.0.4])

	Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
1	01/19 21:03:40	end	trust	trust	10.2.0.4		10.1.0.100	80	web-browsing	allow
2	01/19 21:03:40	end	trust	trust	10.2.0.4		10.1.0.100	80	web-browsing	allow
3	01/19 21:03:40	end	trust	trust	10.2.0.4		10.1.0.100	80	web-browsing	allow
4	01/19 21:03:39	end	trust	trust	10.2.0.4		10.1.0.100	80	web-browsing	allow

Logs

Traffic

Threat

URL Filtering

Wildfire Submissions

Data Filtering

HIP Match

IP-Tag

User-ID

Tunnel Inspection

(addr[dst in 10.1.0.100])

	Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
1	01/19 21:04:22	end	trust	trust	10.2.0.4		10.1.0.100	80	web-browsing	allow
2	01/19 21:04:21	end	trust	trust	10.2.0.4		10.1.0.100	80	web-browsing	allow
3	01/19 21:04:21	end	trust	trust	10.2.0.4		10.1.0.100	80	web-browsing	allow
4	01/19 21:04:20	end	trust	trust	10.2.0.4		10.1.0.100	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
```

```
matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$ terraform destroy
Plan: 0 to add, 0 to change, 71 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:

```
azurerm_resource_group.common_fw: Destruction complete after 3m16s

Destroy complete! Resources: 71 destroyed.
matt@Azure:~/terraform/azure/transit_2fw_2spoke_common$
```



DEPLOYMENT CHEAT SHEET

Part 1: Setup Build

```
$ az vm image terms accept --urn paloaltonetworks:vmseries1:<byol/bundle1/bundle2>:9.0.1  
$ git clone https://github.com/wwce/terraform; cd terraform/azure/transit_vnet_2fw_2spoke_common
```

Part 2: Run Build

Edit `terraform.tfvars` to match PAN-OS license.

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

Part 3: Destroy Build

```
$ terraform destroy
```

