Azure Transit VNet Deployment Guide

How to deploy a Transit VNet solution in Azure

http://www.paloaltonetworks.com

© 2018 Palo Alto Networks. Proprietary and Confidential

Last Update: Feb 2018

Table of Contents

Version History

Version number	Comments
1.0	Initial GitHub check-in

1. About

This document will guideline how to deploy a Transit VNet solution in Azure. For more details about the advantages of the hub and spoke topology please refer to this link:

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/hybrid-networking/hub-spoke

Note: The Azure Transit VNet solution is considered advanced. It requires familiarity with Azure and Palo alto Networks. For a more entry level solution please refer to the following Azure Two Tier solution.

https://github.com/PaloAltoNetworks/azure/tree/master/two-tier-sample

Palo Alto Networks provides Microsoft Azure ARM Templates to deploy a Transit VNet solution of VM-Series firewalls used in conjunction with, Application Gateways, Standard Load Balancers, Basic Load Balancers, and User Defined Route Tables. This solution does not provide native bootstrapping, so you will be provided firewall configuration files for both the Hub and Spoke firewall pairs as well as detailed steps on how to apply the configuration files.

The Transit VNet solution allows you to provide centralized secured outbound internet access for all your Azure Virtual Networks. This secured outbound internet access is provided by two VM-Series firewall pairs positioned behind an Azure Standard any port load balancer in the Hub VNet. All outbound traffic originating from your Azure Virtual Networks will be provided with a secure single point of exit from your cloud architecture by way of the Hub Virtual Network. User Define Routes are used to route Spoke originating traffic to the Hub internal load balancer for packet forwarding to the Hub VM-Series Firewalls. With layer-7 security being provided by Palo Alto Networks Next Generation Firewalls, you will be able to utilize layer-7 application visibility, content identification, data exfiltration prevention, anti-spyware, anti-malware and many other security benefits.

2. Topology

The transit VNet solution deploys a classic hub-and-spoke. The Hub is deployed in a separate VNet, and each spoke is deployed in a separate VNet as well.

VNet Peering

For the different virtual networks to talk to each other, they must be peered in both the directions. VNet Peering works under the assumption that the peering networks **do not have overlapping subnets.** In this topology, when a VNet spoke is deployed, we will dynamically peer the spoke's VNet and the hub's VNet enabling traffic to flow between them.

For additional information on VNet Peering please reference the link below

https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview

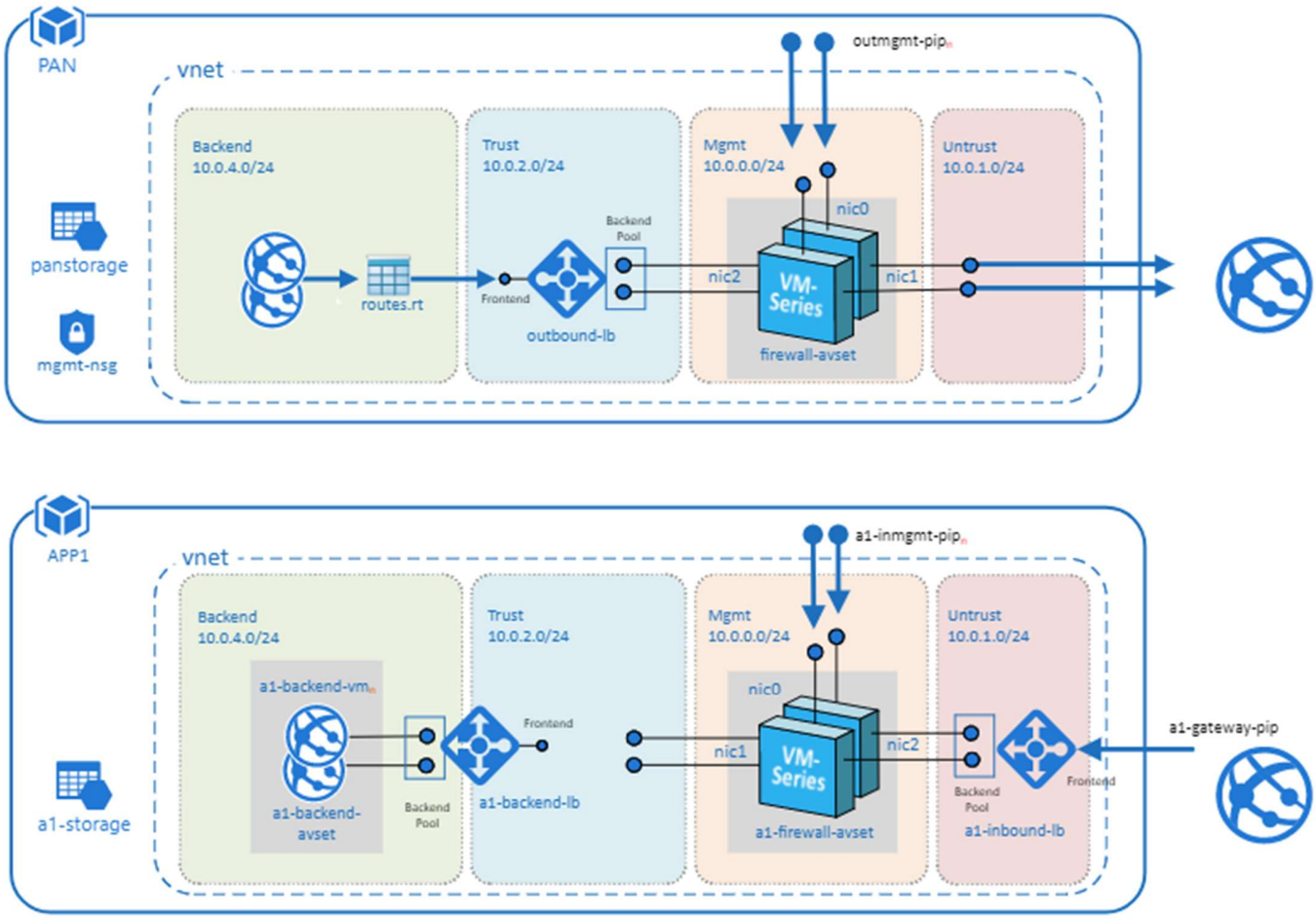


Figure 1

Hub Topology

In Figure 1 **PAN** represents the Hub VNet. The Hub VNet consists of Mgmt , Untrust and Trust subnets. An Azure internal LB[Outbound-LB] used for outbound traffic and a pair of VM Series FWs in an availability set. The Hub topology serves as the exit point of all non-return traffic for the Hub and Spoke topology.

The Hub topology consists of

- 2 VM-Series Firewalls
- 1 Standard internal Loadbalancer

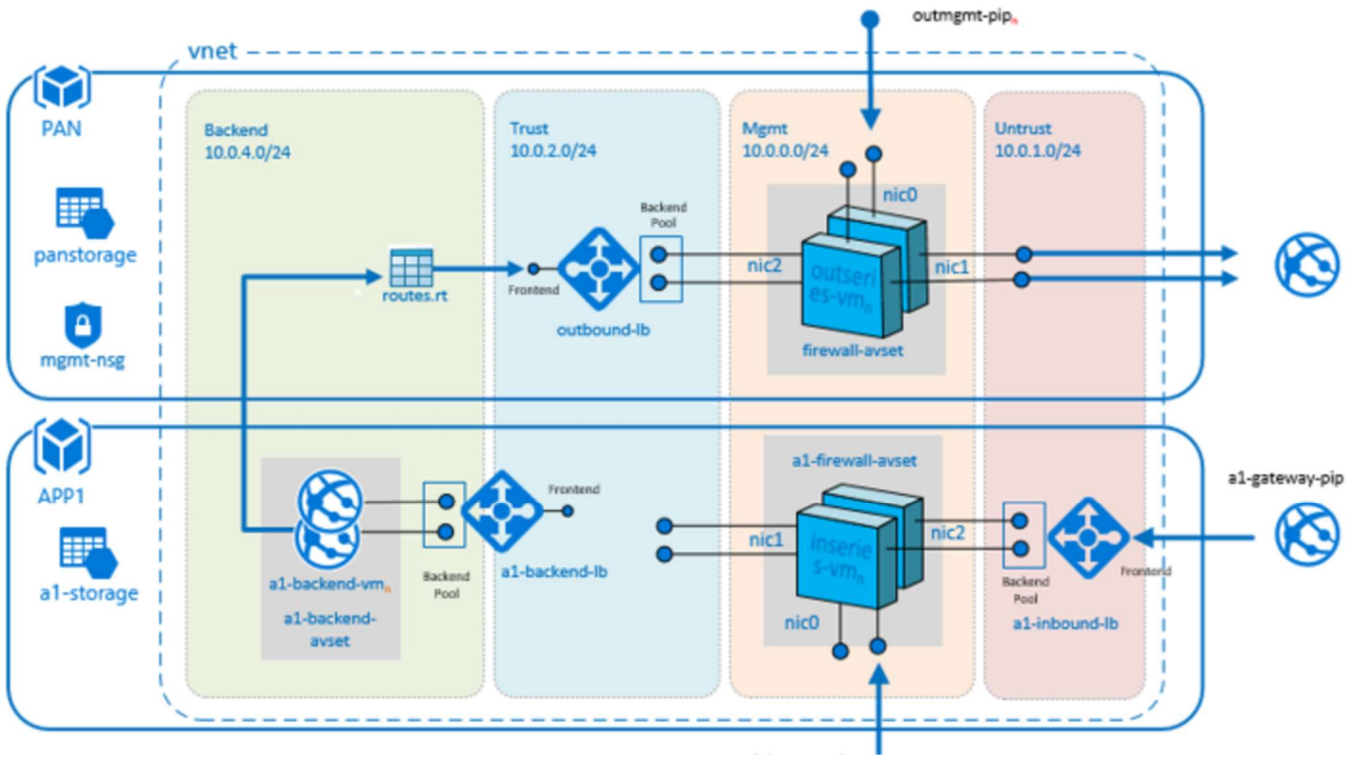
Spoke Topology

In Figure 1 **APP1** represents the Spoke VNet. The spoke VNet allows an ingress point for all traffic destined to public facing resources. The subnets consist of Mgmt, Untrust, Trust and Backend Subnets for the application servers. An Application Gateway doubles as a public facing load balancer and sits on the front end. A pair of VM Series FWs in an availability set receive traffic from the public facing LB. An Internal LB sits behind the firewalls and sends traffic to the backend application servers. All return traffic egresses this same path. When a spoke subscribes to a hub, a UDR is also defined which has a default route to the Hub's Interal Load Balancer. This is so all packets that are not destined to the spoke's virtual network gets forwarded to the Hub Internal LB for routing.

The Spoke topology consists of

- 1 Application Gateway listening on port 80. The App Gateway also functions as a public facing external load balancer
- 2 VM-Series Firewalls
- 1 Internal Loadbalancer
- 2 Linux Web servers
- 1 UDR sending all default route traffic to the Hub vnet Standard Loadbalancer.

Hub & Spoke Topology



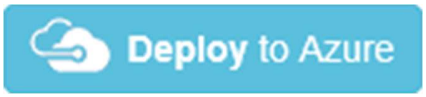
3. Support Policy

This solution 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.

4. Prerequisites

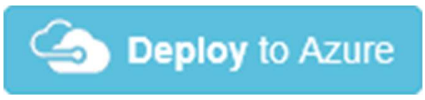
Here are the prerequisites required to successfully launch this template:

1. AZURE account with appropriate permissions.
2. Clone or download the files from the following GitHub repository on to your local machine: https://github.com/PaloAltoNetworks/Azure-transit-VNet
3. If the GitHub Repository has a deploy button you can deploy your templates using the button.

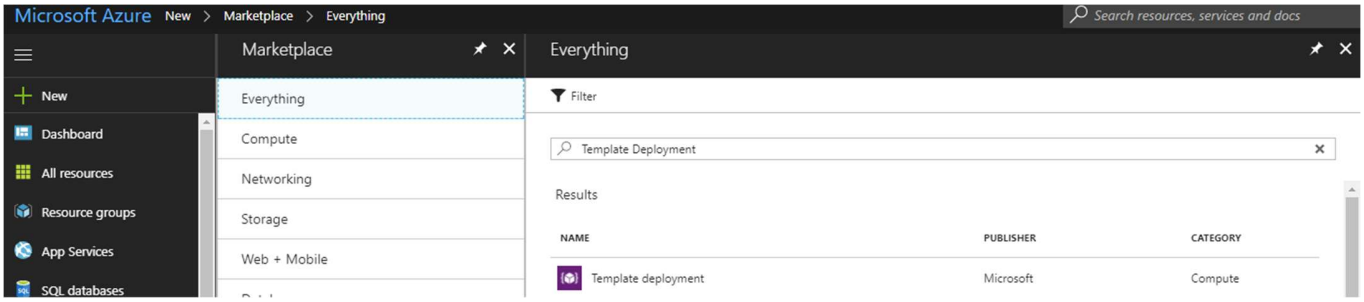


5. Launch the Transit VNet Hub Template

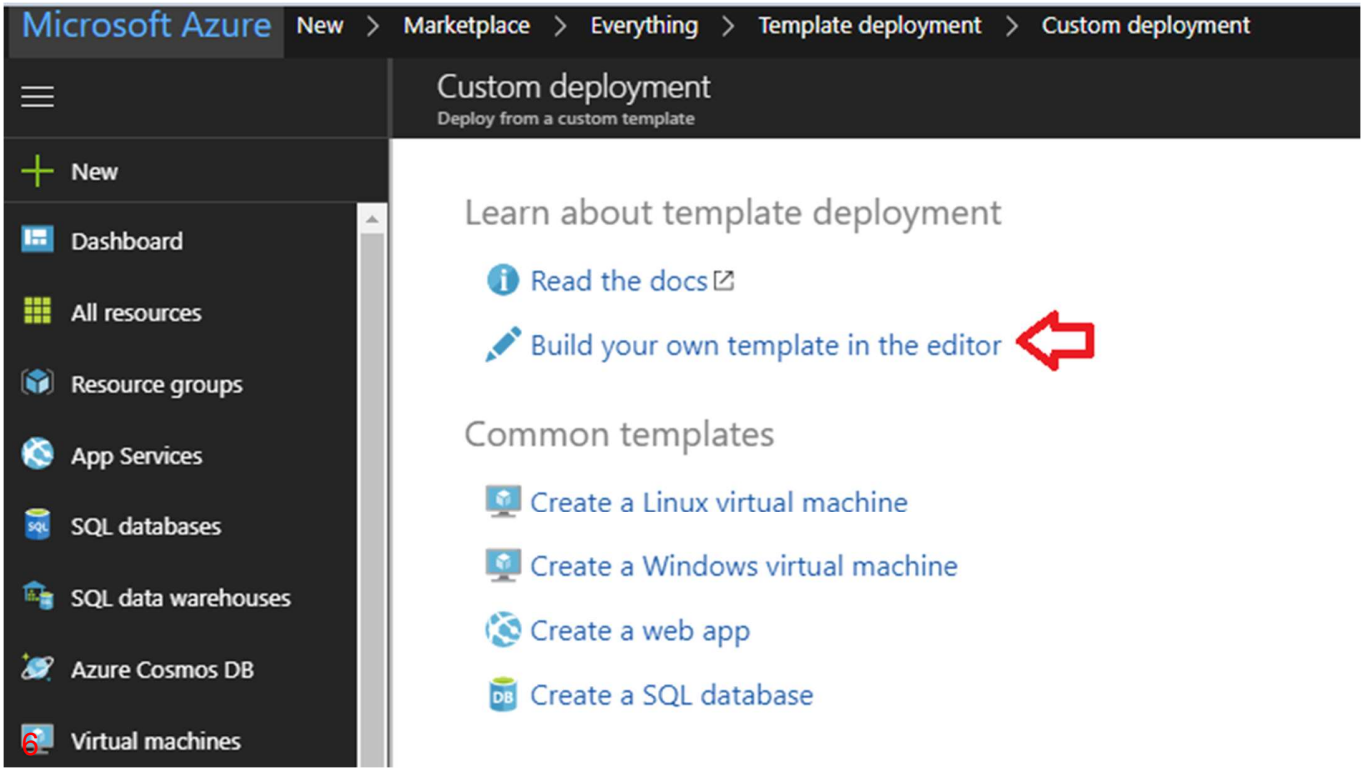
There are multiple ways to deploy your template. You can use Azure CLI, PowerShell, Deploy to Azure button or you can deploy the template manually. If the GitHub Repository has a **Deploy to Azure** button you can deploy your template by clicking the deploy button for each template. Below I will walk you through how to launch your ARM template manually.



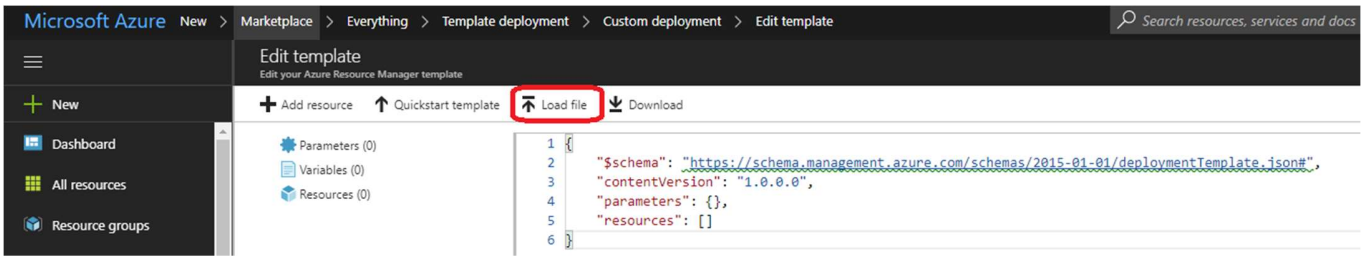
In the Azure Resource Manager console you can launch the **azureDeployInfra.json** file directly from the Azure Portal. To do this click **"New"** then search **"Template Deployment"**, click the Template Deployment icon an select **"Create".**



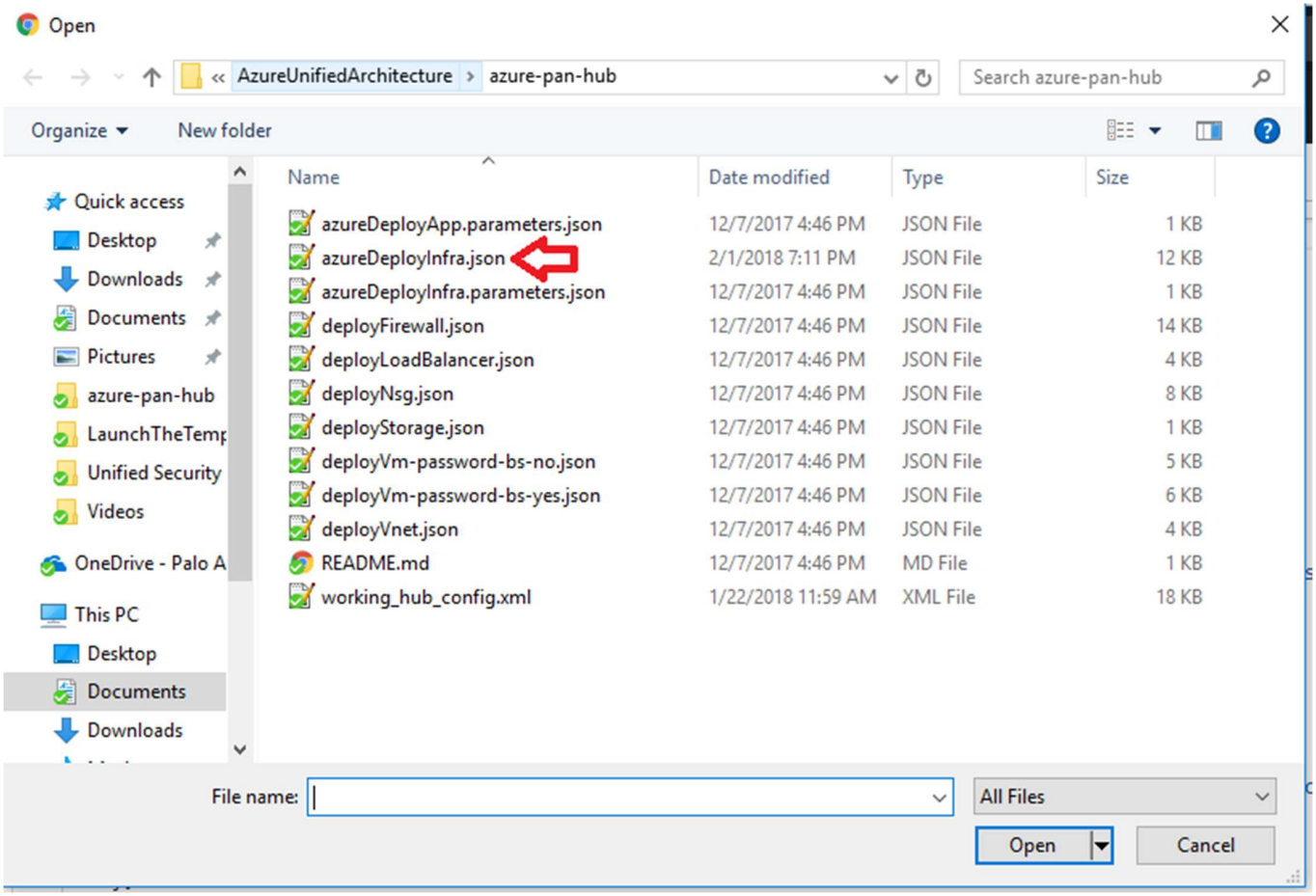
In the next screen click **“Build your own template in the editor”**



Select **“Load File”**



Select **“azureDeployInfra.json“** file from the Azure-Transit-VNet/azure-pan-hub directory that you cloned from GitHub, then click **“Save”** to bring up the parameters.



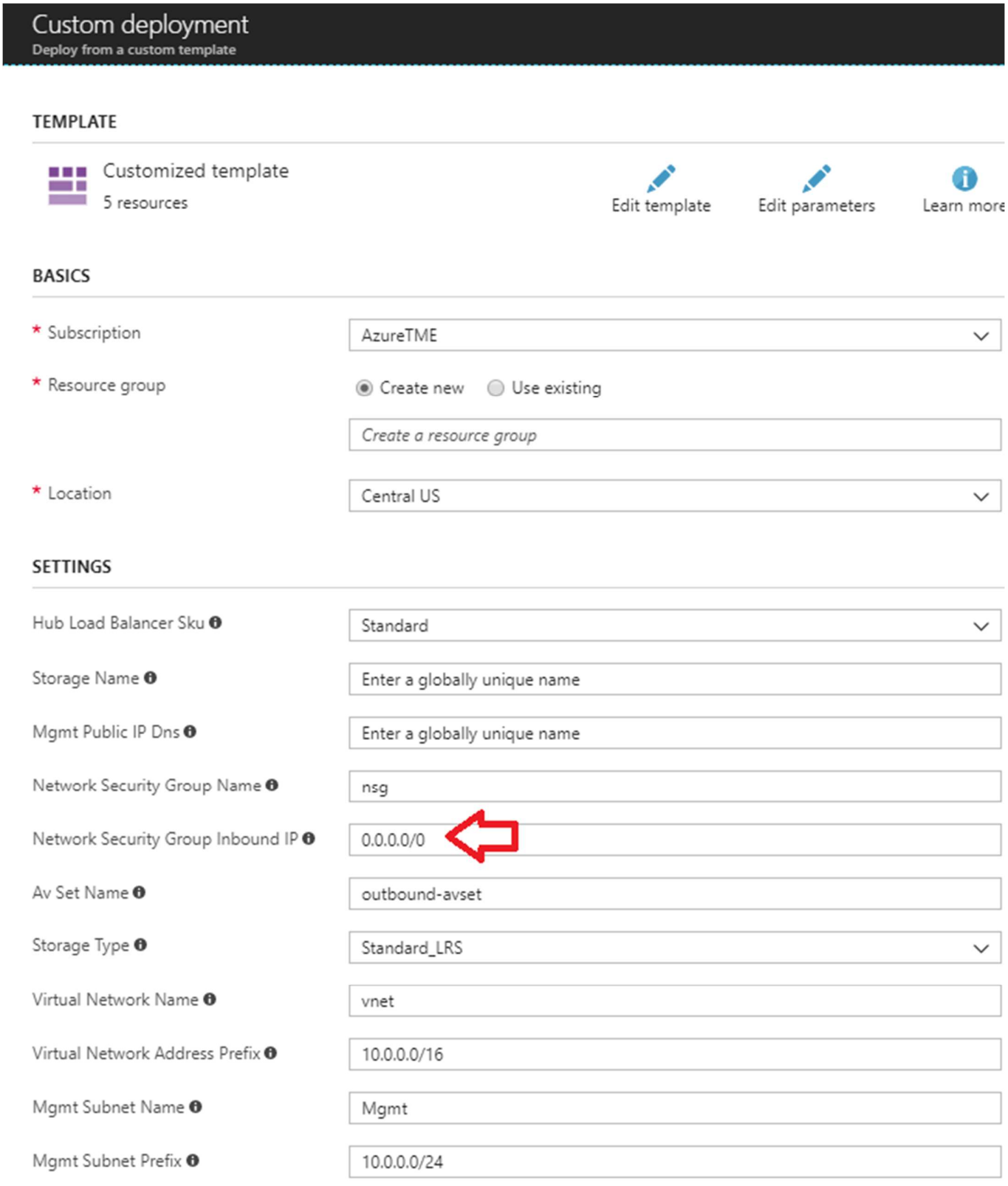
a. Most of the **parameters** are self-explanatory and should be left at the defaults
b. **Resource Group** – Always create a new resource Group. The hub template does not work in an existing resource group
c. **Location** – Use a location where Standard Load Balancer Preview feature is enabled. List of regions is found here - https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-standard-overview#region-availability
d. **Hub Load Balancer Sku** – Use the **Standard** SKU type which will load balance all outbound TCP traffic.
e. **Network Security Group Name** is the NSG that is attached to the Firewall's Management subnet
f. **Network Security Group Inbound Src IP** – This is the IP you will allow explicit access to the management interface of the virtual machines.
g. **Virtual Network Name** – Use a unique name which will not be used in the spoke VNet deployment. Remember this name since it is an input parameter for the spoke template.

h. For security purposes be sure to set **Security Group Inbound IP** for mgmt access to the firewall.

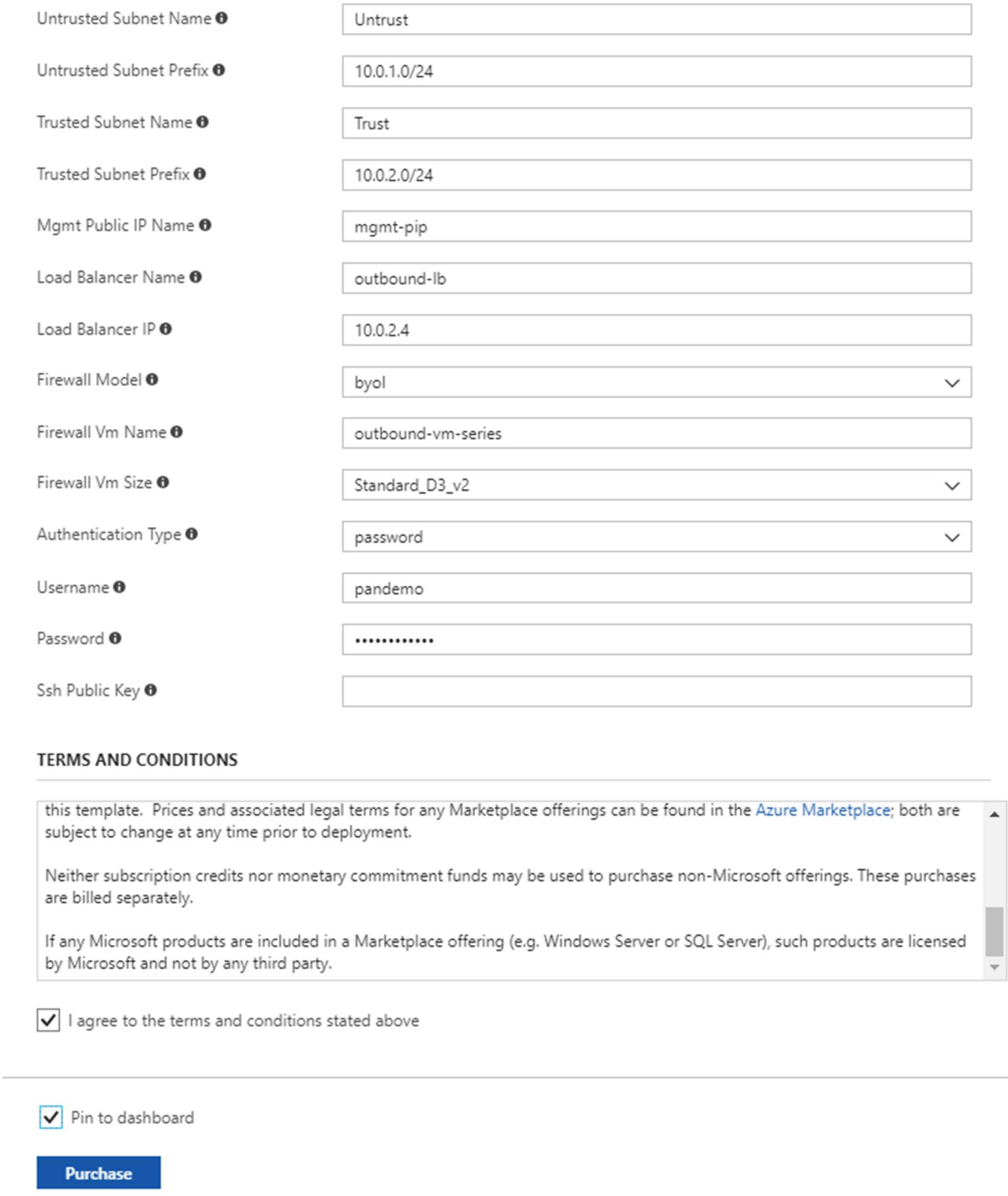
i. **Virtual Network Address Prefix** – Use a network address which will not be used in the spoke deployment.
j. **Virtual Network Name** – Use a unique name which will not be used in the spoke VNet deployment. Remember this name since it is an input parameter for the spoke template.
k. **Virtual Network Address Prefix** – Use a network address which will not be used in the spoke deployment.
l. **Load Balancer IP** – Use a static IP for Load Balancer in the Trust network. Remember this address since it is used as an input parameter for the spoke template.
m. **Firewall Model** – The template default will work unless a larger size is required.
n. It could take up to 5 minutes to complete the launch or It could take longer depending on Azure.

o. **Username** and **password** that is entered by default for the devices is

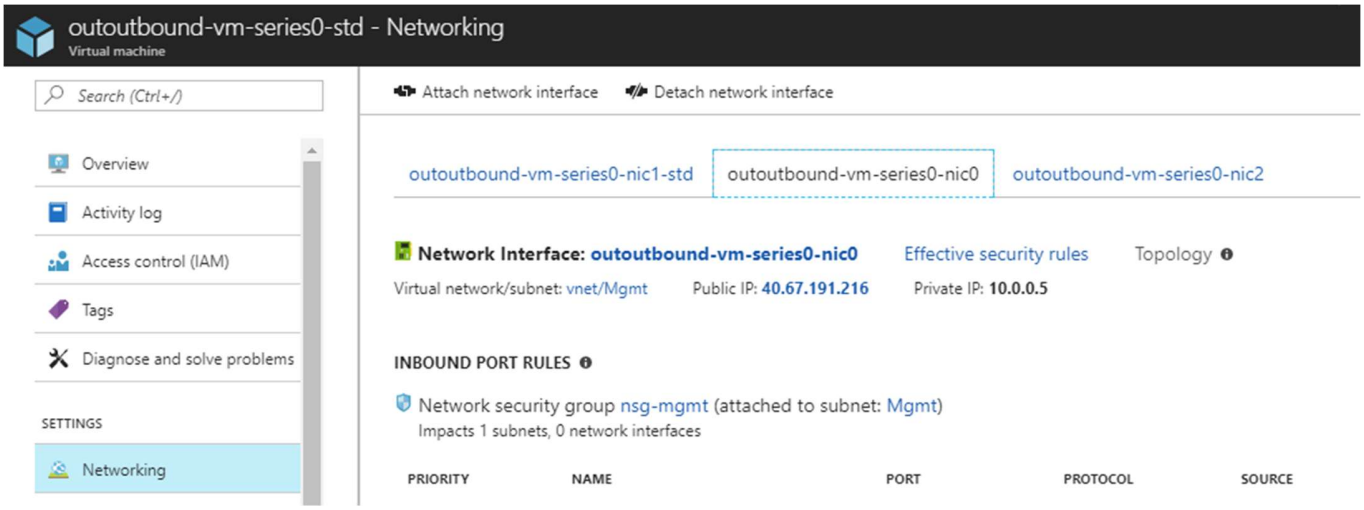
   user:pandemo **password:**Dem0pa$$w0rd


Custom deployment
Deploy from a custom template
TEMPLATE
Customized template
5 resources
Edit template
Edit parameters
Learn more
BASICS
* Subscription
AzureTME
* Resource group
Create new
Use existing
Create a resource group
* Location
Central US
SETTINGS
Hub Load Balancer Sku
Standard
Storage Name
Enter a globally unique name
Mgmt Public IP Dns
Enter a globally unique name
Network Security Group Name
nsg
Network Security Group Inbound IP
0.0.0.0/0
Av Set Name
outbound-avset
Storage Type
Standard_LRS
Virtual Network Name
vnet
Virtual Network Address Prefix
10.0.0.0/16
Mgmt Subnet Name
Mgmt
Mgmt Subnet Prefix
10.0.0.0/24

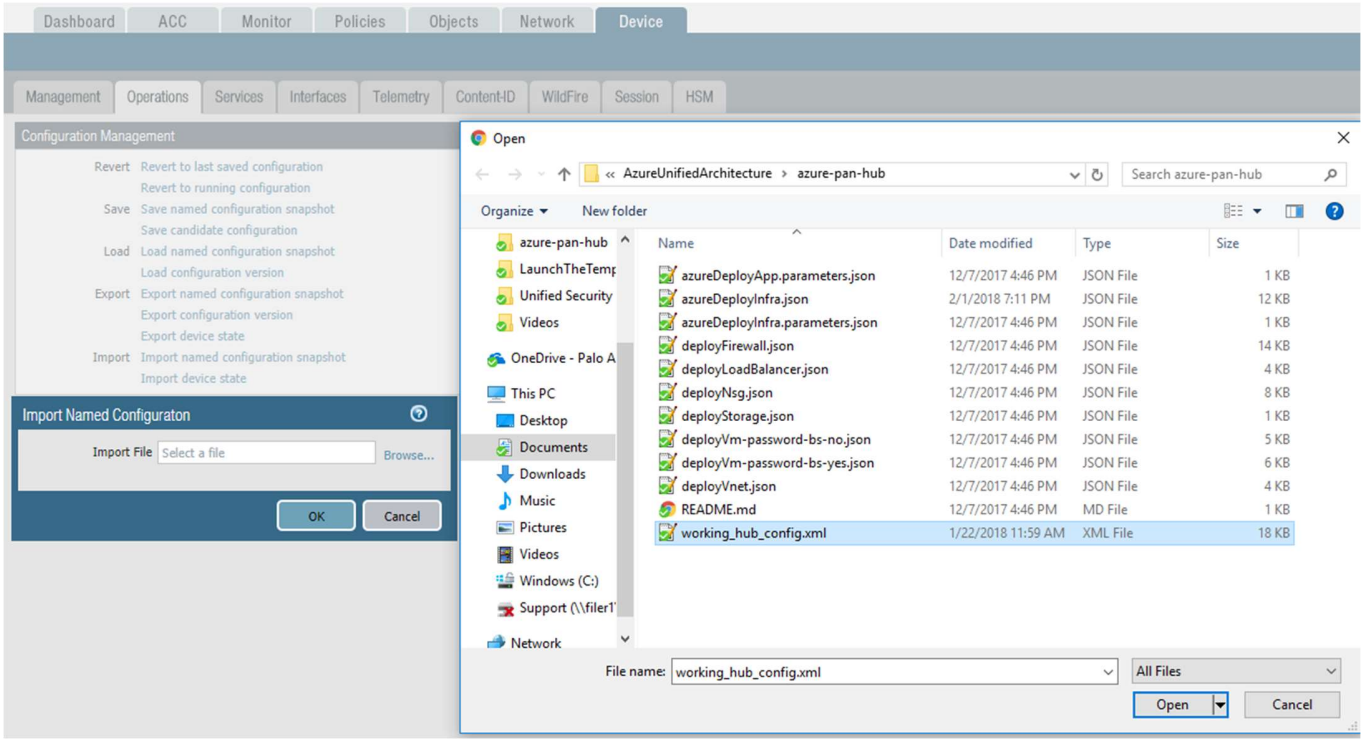
Acknowledge the terms and conditions and click "**Purchase**"



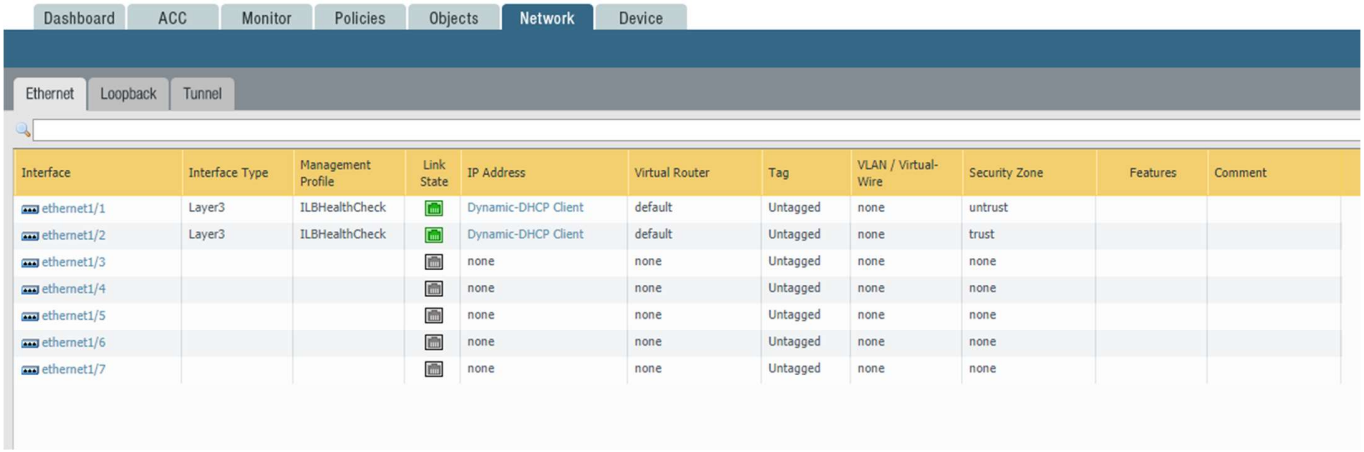
Once the firewalls have launched, locate the **Management** interface public IP address in Azure.



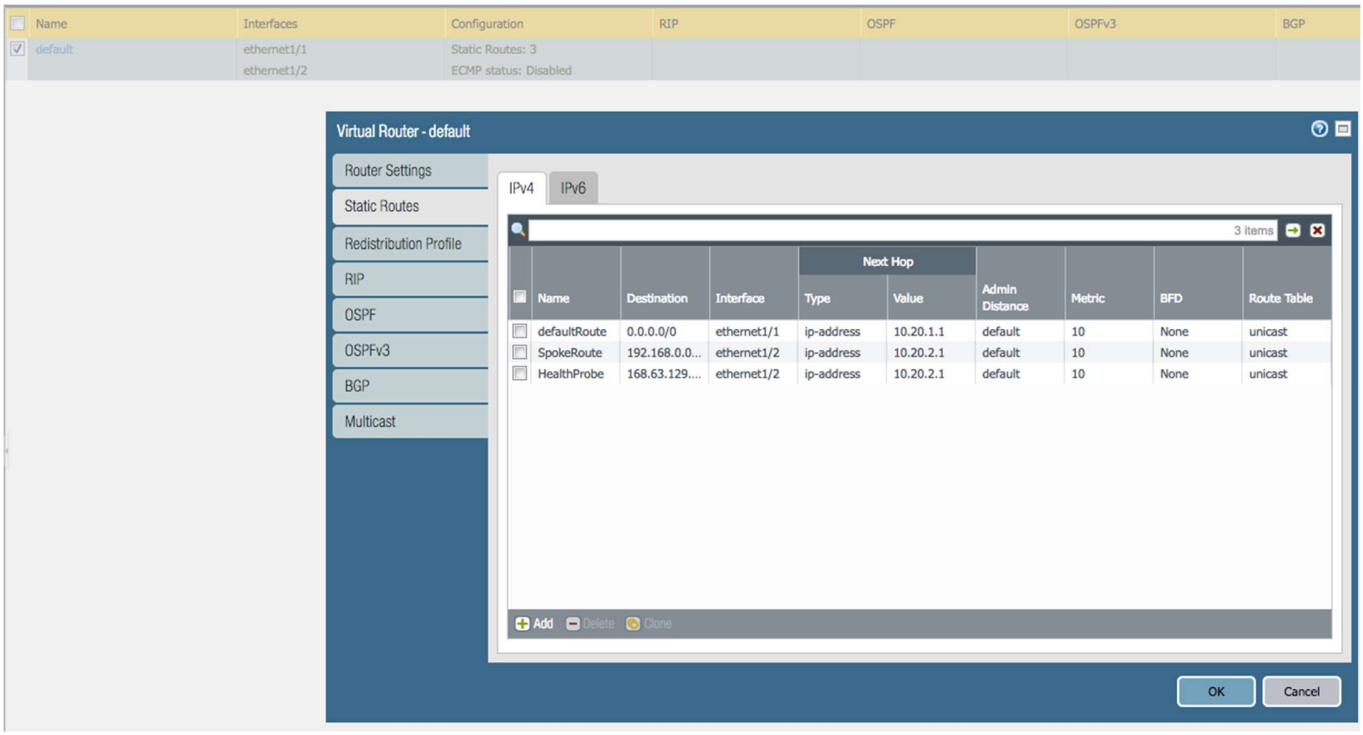
Log into the hub firewalls using **HTTPS.** Locate the **working_hub_config.xml** configuration snapshot and import this configuration into both firewalls. This is in the Hub directory that you exported from GitHub.



Once you load the configuration and commit the changes make sure your ethernet1/1 and Ethernet1/2 interfaces now show green.



Verify the **virtual router** has the following configuration.

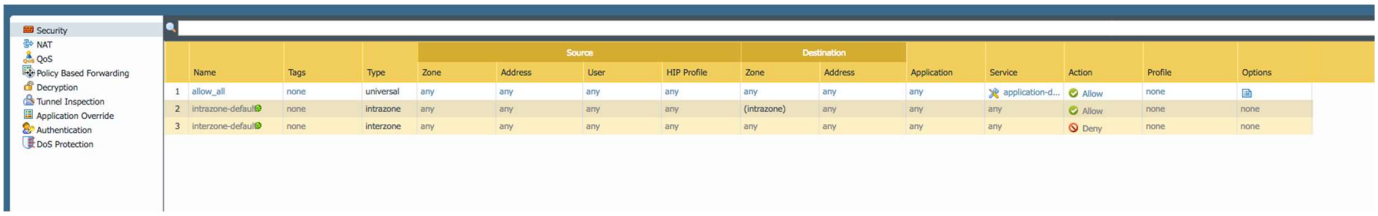


DefaultRoute: is to forward all outbound traffic to the untrust interface so that it egresses out of the Azure network.

SpokeRoute: is to forward all the inbound traffic and inter-spoke traffic back to the Trust interface so that it reaches the appropriate Spoke (application server). Note that the Network address of the all the spokes VNets should be part of this network address. If a new spoke is added whose network address is not part of this network address, then a new route needs to be added in the config to forward that traffic to the Trust interface.

HealthProbe: is to respond to the health probe packets generated by the Internal Load Balancer. For this lab the health check is configured to port 22 on the firewall Trust interface.

An **allow-all** security policy is created to forward all traffic. This should be modified to accommodate your policy preferences.



Verify that you have a **NAT rule** on the hub firewall for outbound traffic

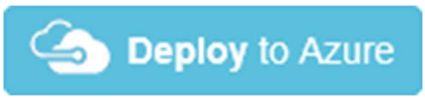
6. Launch the Transit VNet Spoke Template

Spoke Template Options

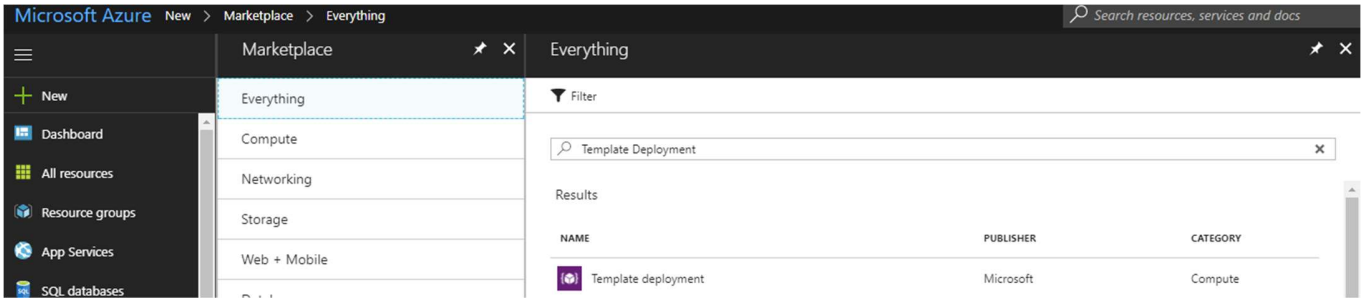
Azuredeploy.json – This launches the spoke template with VM-Series firewalls sandwiched between an external and internal load balancer. This provides secured external access to public facing workloads with return traffic egressing the spoke VNet. All internal originating traffic will be forwarded to the Hub VNet as the exit route to provide secure outbound access.

Azuredeploy-no-firewall.json – Launches the spoke template with no firewalls but still launches application servers. This scenario would NOT provide security using the VM-Series for public facing workloads. All internal originating traffic will be forwarded to the Hub VNet as the exit route to provide secure outbound access.

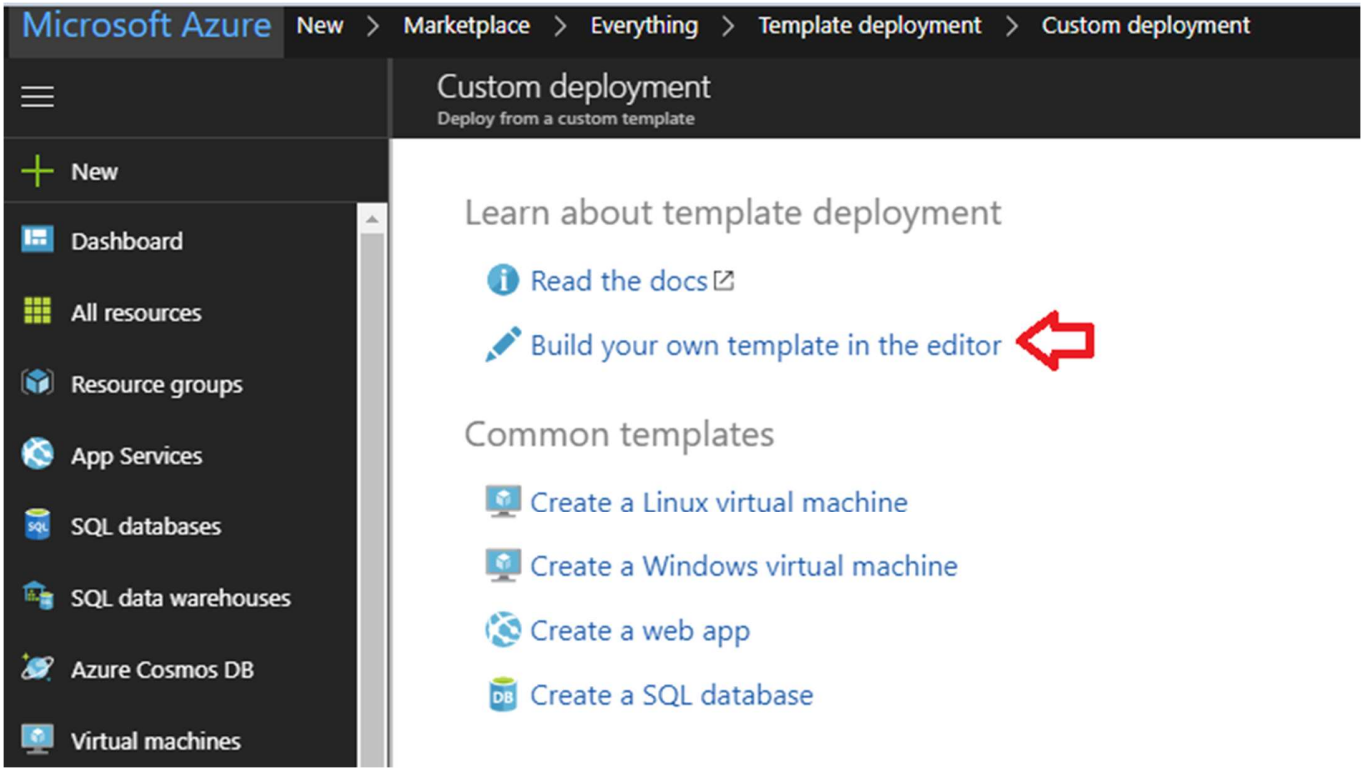
There are multiple ways to deploy your template. You can use Azure CLI, PowerShell, Deploy to Azure button or you can deploy the template manually. If the GitHub Repository has a **Deploy to Azure** button you can deploy your template by clicking the deploy button for each template. Below I will walk you through how to launch your ARM template manually.



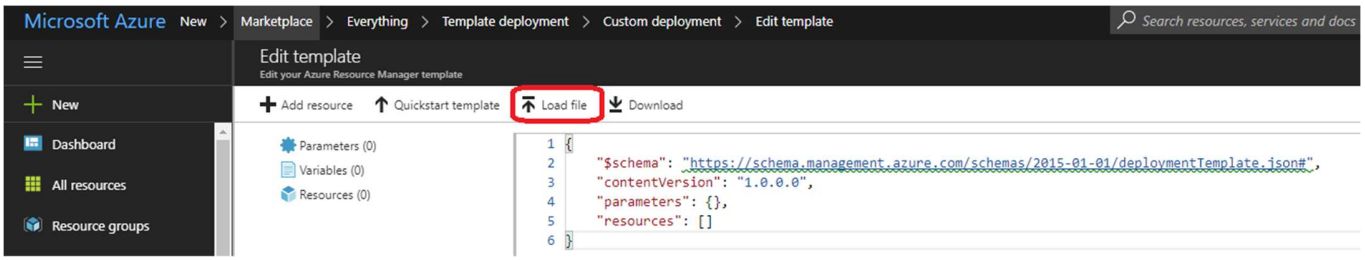
From the Azure-Transit-VNet/azure-pan-spoke GitHub repository that you cloned, launch the **azuredeploy.json** file directly from the Azure Portal. You may need to bring up two azure portal browsers in order to locate information needed to fill out the parameters when launching this template. To do this click **"New"** then search **"Template Deployment"**, click the Template Deployment icon an select **"Create".**



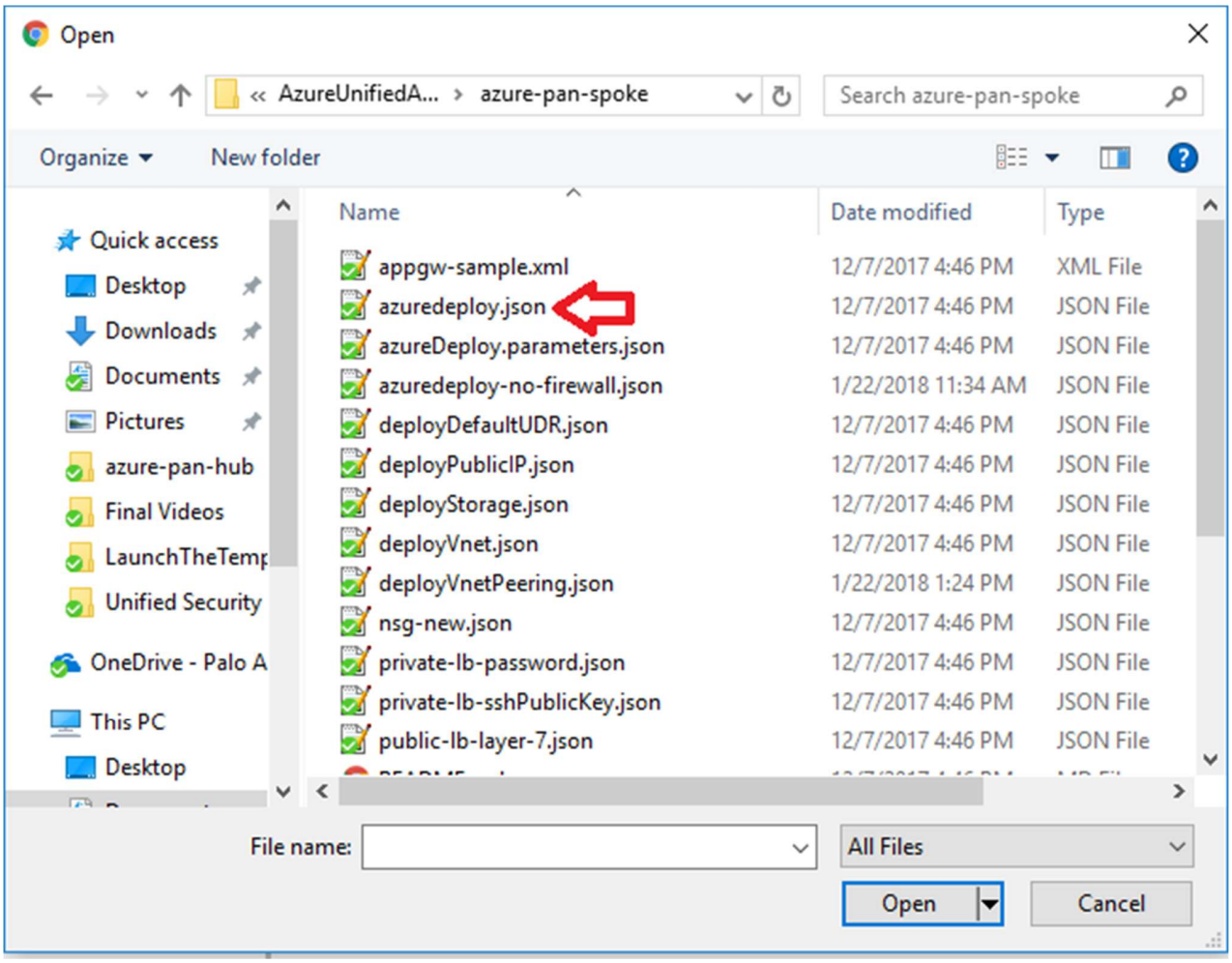
In the next screen click **“Build your own template in the editor”**



Select **“Load File”**



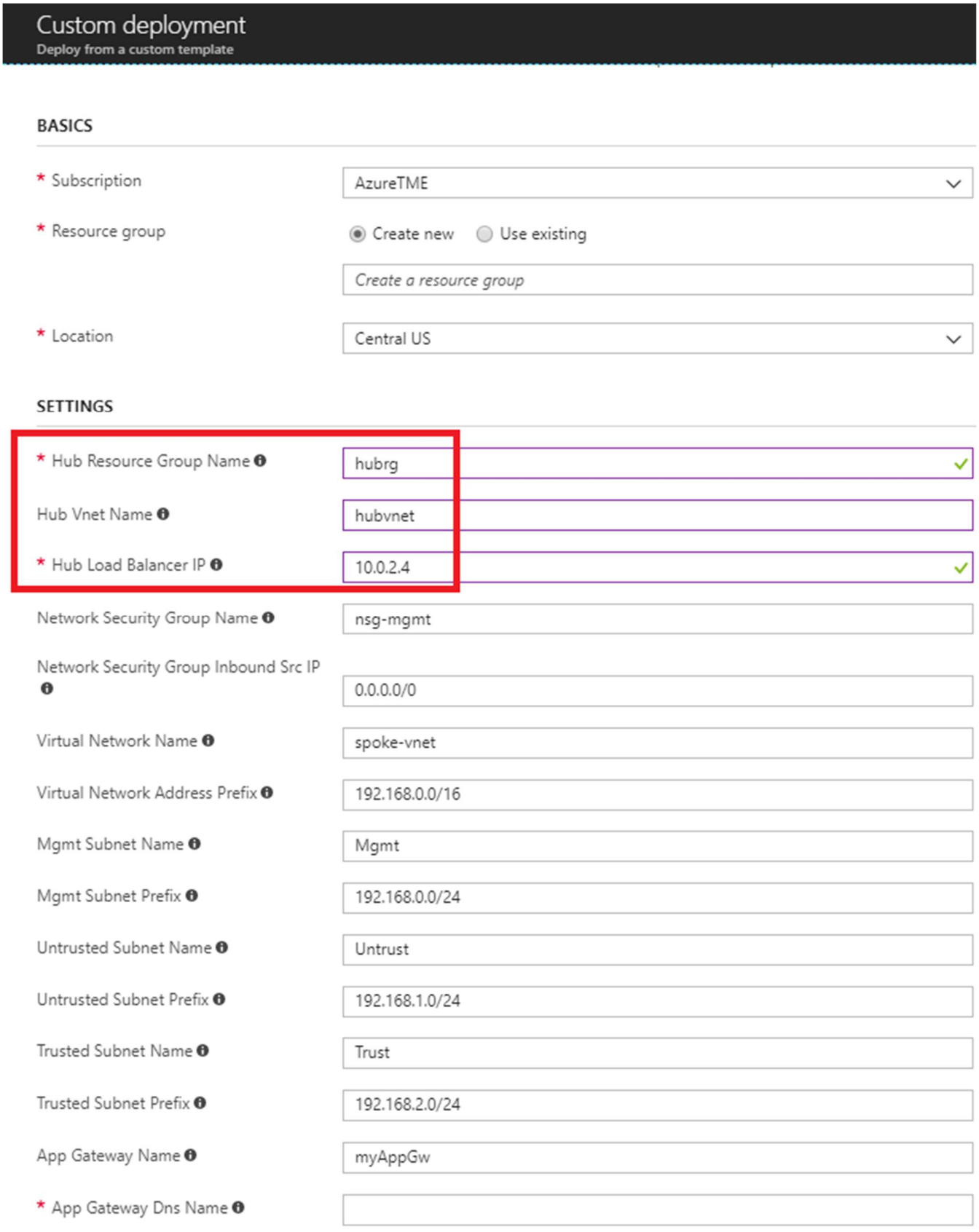
Select **“azuredeploy.json“** file from the Azure-Transit-VNet/azure-pan-spoke directory that you cloned from GitHub, then click **“Save”** to bring up the parameters.



a. Most of the **parameters** are self-explanatory and should be left at the defaults

b. **Resource Group** – Create a new Resource Group. This template does not work with existing resource groups.
c. **Location** – It should be the same location as the hub since VNet peering does not work well across regions.
d. **Hub Resource Group Name** – Give the Resource Group name of the hub created resource group.
e. **Hub VNet Name** – Use the exact VNet name of the hub created earlier.
f. **Hub Load Balancer IP** – Use the static IP given to the Load Balancer in the created in the hub template. You can find this information in the load balancer settings
g. **Network Security Group Name –** The security group name for mgmt access
h. **Network Security Group Inbound Src IP** – This is the IP you will allow explicit access to the management interface of the virtual machines.
i. **Virtual Network Address Prefix** – This network address should be the subnet of the network address given in the "SpokeRoute" in the hub's firewall configuration.
j. **Mgmt, Trust and Untrust** subnets should be subnets of the VNet subnet created in the previous step.
k. **Firewall VM Size** - Choose the Firewall Model and Size based on requirements. Use Standard D3 or D3 v2.

l. **SSH Public Key** – If using a password then leave this section blank.


Custom deployment
Deploy from a custom template
BASICS
* Subscription
AzureTME
* Resource group
Create new
Use existing
Create a resource group
* Location
Central US
SETTINGS
* Hub Resource Group Name
hubrg
Hub Vnet Name
hubvnet
* Hub Load Balancer IP
10.0.2.4
Network Security Group Name
nsg-mgmt
Network Security Group Inbound Src IP
0.0.0.0/0
Virtual Network Name
spoke-vnet
Virtual Network Address Prefix
192.168.0.0/16
Mgmt Subnet Name
Mgmt
Mgmt Subnet Prefix
192.168.0.0/24
Untrusted Subnet Name
Untrust
Untrusted Subnet Prefix
192.168.1.0/24
Trusted Subnet Name
Trust
Trusted Subnet Prefix
192.168.2.0/24
App Gateway Name
myAppGw
* App Gateway Dns Name

App Gateway Subnet Name	AppGWSubnet
App Gateway Subnet Prefix	192.168.3.0/24
Internal Load Balancer Name	myPrivateLB
Backend Subnet Name	backendSubnet
Backend Subnet Prefix	192.168.4.0/24
Backend Vm Size	Standard_D1
Firewall Model	byol
Firewall Vm Name	VM-Series
Firewall Vm Size	Standard_D3
* Mgmt Public IP Address Name	
* Storage Account Name	
Storage Account Type	Standard_LRS
* Username	
Authentication Type	password
Password	
Ssh Public Key	

TERMS AND CONDITIONS

Azure Marketplace Terms | Azure Marketplace

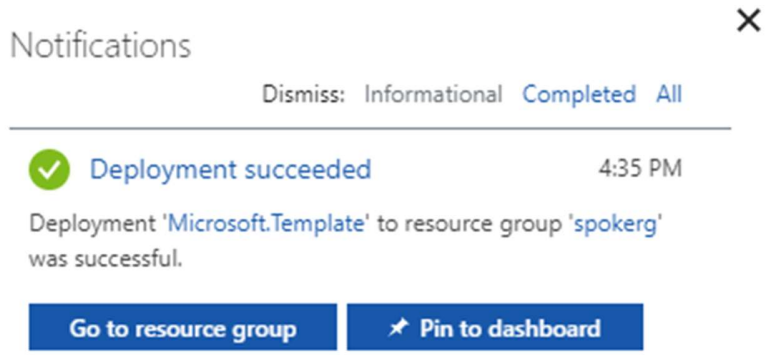
By clicking "Purchase," I (a) agree to the applicable legal terms associated with the offering; (b) authorize Microsoft to charge or bill my current payment method for the fees associated the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s); and (c) agree that, if the deployment involves 3rd party offerings, Microsoft may share my contact information and other details of such deployment with the publisher of that offering.

☐ I agree to the terms and conditions stated above

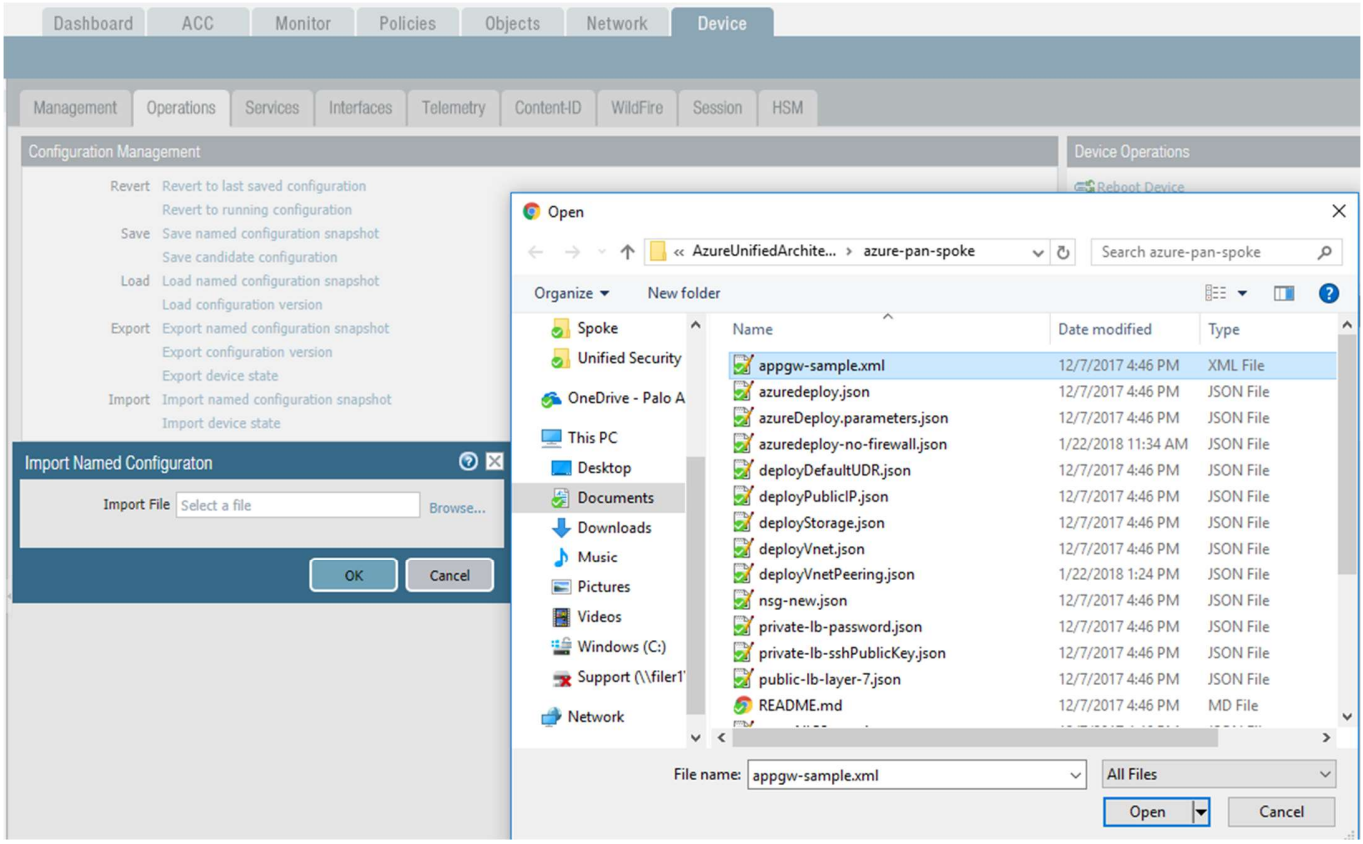
☐ Pin to dashboard

Purchase

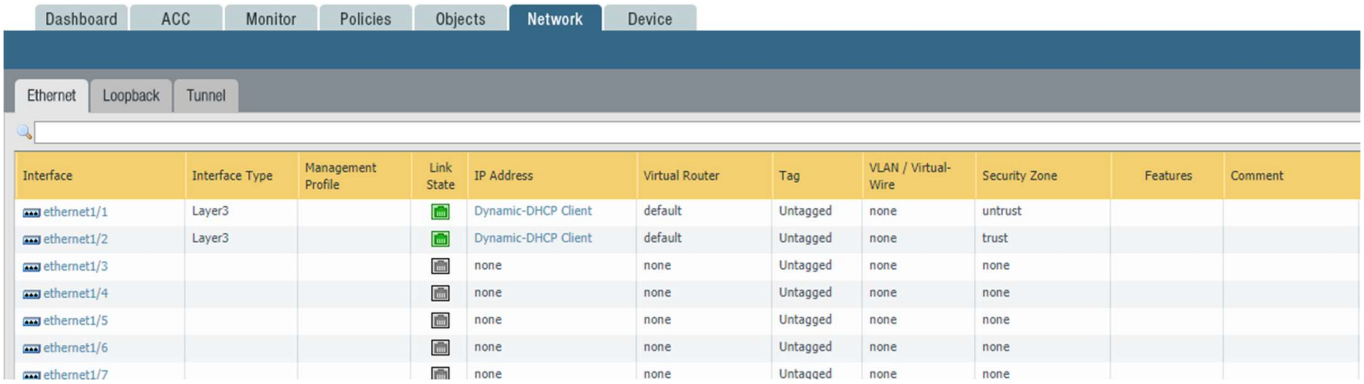
Once the Spoke template has successfully launched you will see Deployment Succeeded.



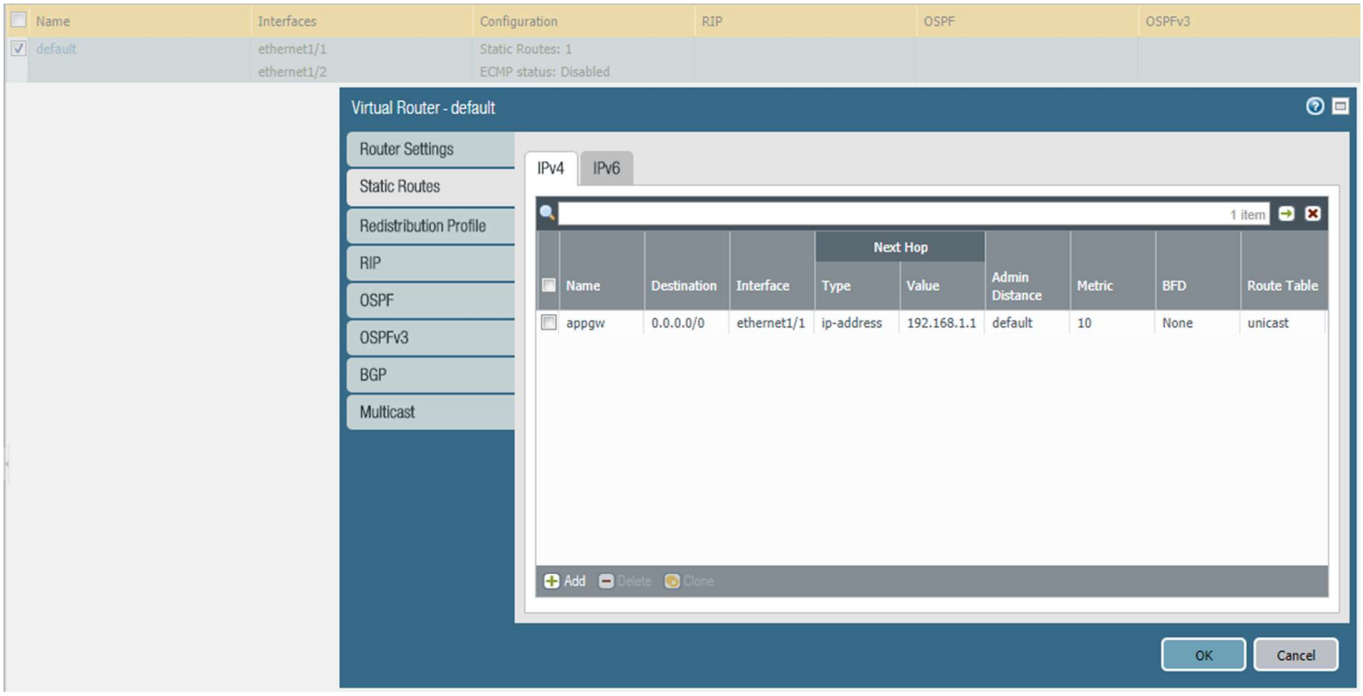
Log into the spoke firewalls using **HTTPS.** Locate the **appgw-sample.xml** configuration snapshot and import this configuration into both firewalls. This is in the Spoke directory that you exported from GitHub.



Once you load the configuration and commit the changes. Once you have committed the changes make sure your ethernet1/1 and Ethernet1/2 interfaces now show green



Verify the spoke firewall **virtual router** has the following configuration.



appgw: is to forward all traffic originating from the firewall to the untrust interface. Traffic originating from spoke resources behind the firewall will egress through the Hub VNet.

An **allow-all** security policy on the firewall is created to receive all traffic although the application gateway load balancer only listens for port 80. This should be modified to accommodate your policy preferences.

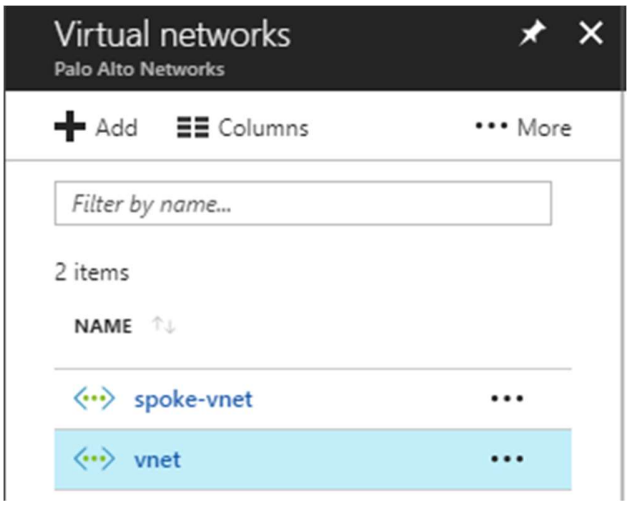
	Name	Tags	Type	Source Zone	Source Address	Source User	Source HIP Profile	Destination Zone	Destination Address	Application	Service	Action	Profile	Options
1	allow_all	none	universal	any	any	any	any	any	any	any	application-d...	Allow	none	
2	intrazone-default	none	intrazone	any	any	any	any	(intrazone)	any	any	any	Allow	none	none
3	interzone-default	none	interzone	any	any	any	any	any	any	any	any	Deny	none	none

Verify that you have a **NAT rule** on the spoke firewall for inbound traffic

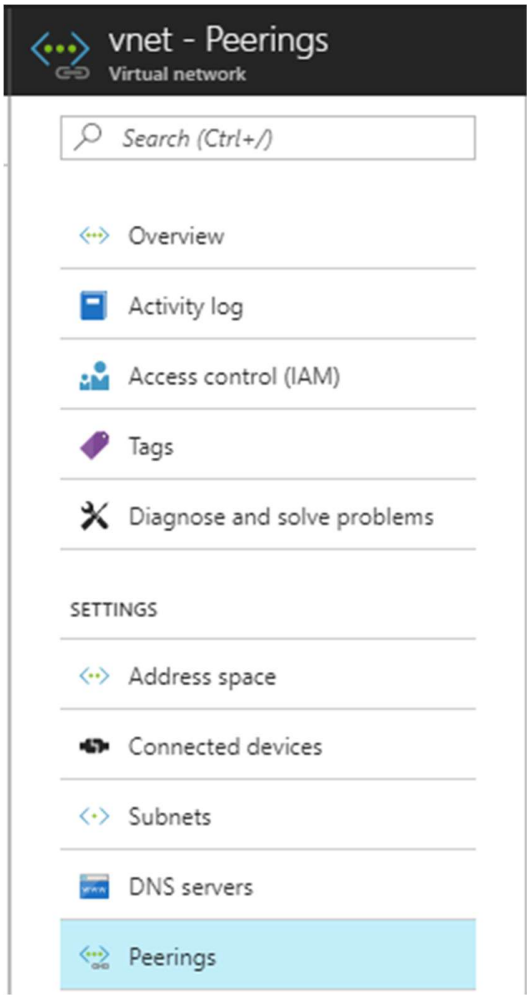
	Name	Tags	Original Packet: Source Zone	Original Packet: Destination Zone	Original Packet: Destination Interface	Original Packet: Source Address	Original Packet: Destination Address	Original Packet: Service	Translated Packet: Source Translation	Translated Packet: Destination Translation
1	ilb	none	any	untrust	any	any	firewall-untrust...	any	dynamic-ip-and-port ethernet1/2	address: internal-load-balancer-IP

7. VNet Peering Verification

Within Azure Portal verify that **VNet Peering** has been configured automatically between the Hub VNet and Spoke VNet. To check this in Azure navigate to Virtual Networks > select the virtual network **name.**



Then select **Peerings**



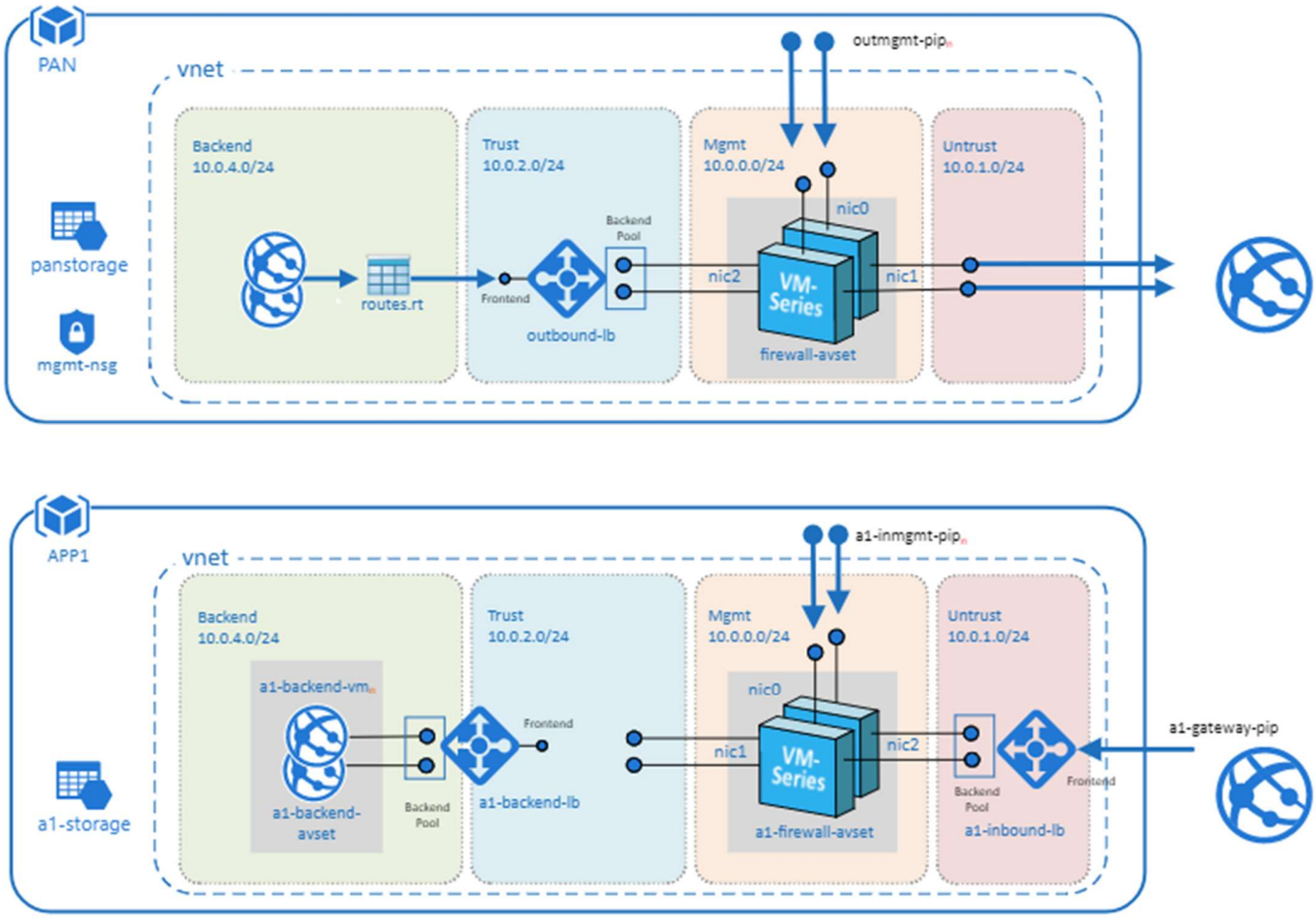
Here you should see the name of the peer **VNet** with a status of connected. **Gateway Transit** should be disabled. Check this on both the hub and spoke VNet.

8. Inbound and Outbound Traffic Tests

Once you have confirmed that both the Hub and Spoke templates were successfully deployed, you have imported and loaded the firewall configuration and confirmed VNet Peering, you will want to test your proof of concept with live traffic.

Outbound Traffic Test

As per the diagram all traffic originating from within the Azure virtual networks will exit through the Hub VNet.



One way to test this setup is to originate traffic from a backend Linux VM deployed in the spoke to www.google.com by using wget www.google.com. From there check the traffic logs of the Hub firewalls

for www.google.com traffic or web-browsing traffic if using another port 80 based website for wget tests. You will need a license to see logs in the traffic logs or you can edit the template to use PAYG1 or PAYG2.

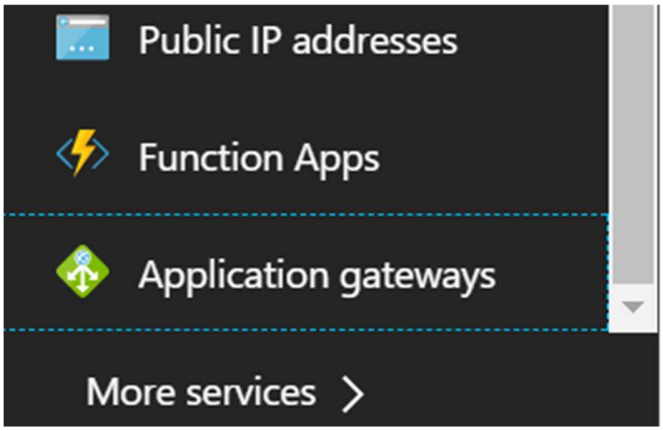
By default you will not be able to access the Linux servers in the spoke. To access the Linux devices you will need to add a public IP address to one of the Spoke backend Linux servers. Then add a route on the UDR named "**defaultBackendUDR**" for mgmt traffic, that will allow your public IP address with a next hop of "**Internet**"

Another way to accomplish this would be to install a **Bastion Host** or **Jump Box** into the Backend Subnet and SSH from that device.

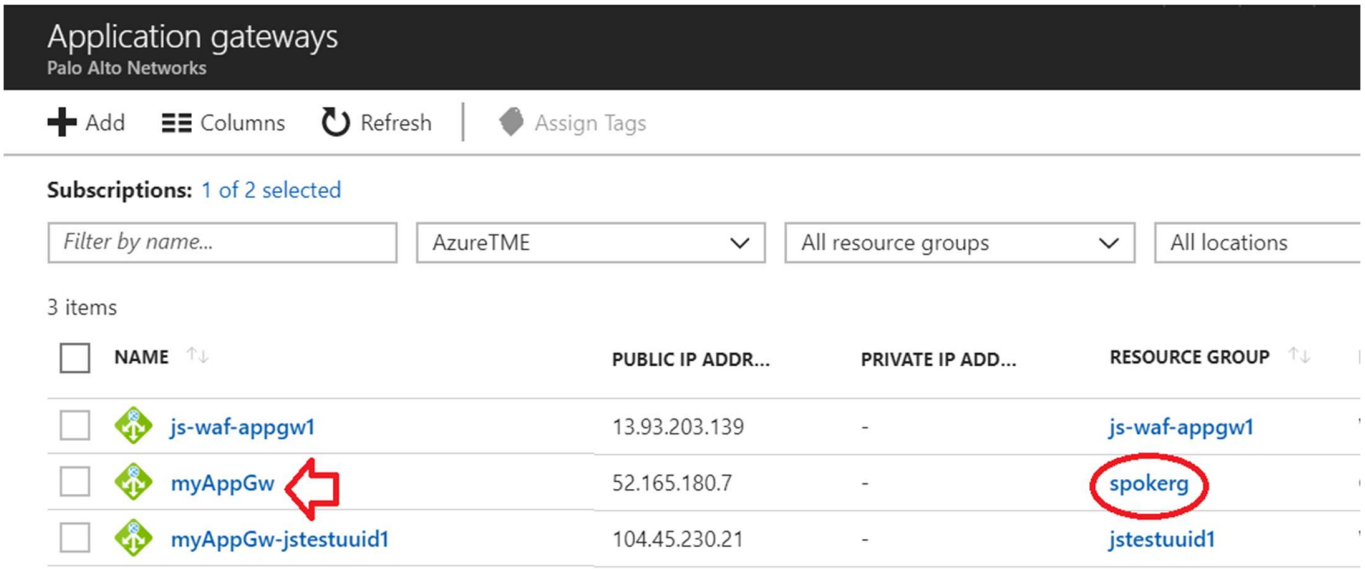
Inbound Traffic Test

When launching the spoke template with firewalls, the spoke VNet will have an Application Gateway (External LB), A set of firewalls and an internal Load balancer. This allows the spoke to host its own public facing workloads. Once you have launched the Spoke template with firewalls you can test access to the public facing workload by

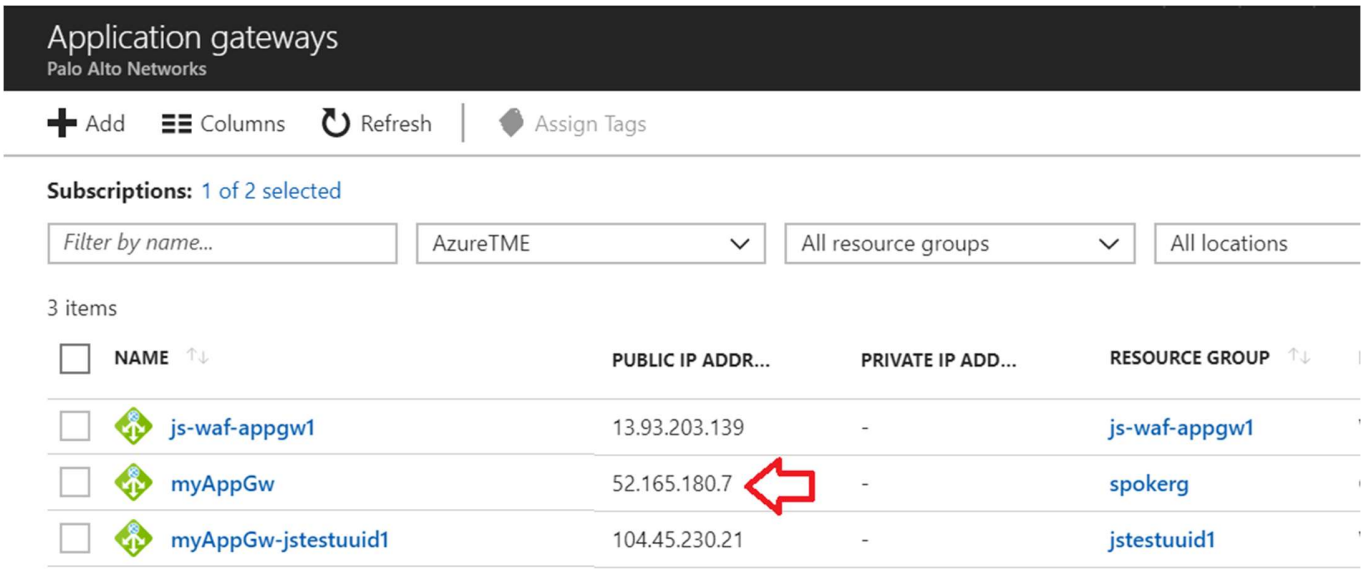
Navigating to **"Application gateways"** within the Azure Portal



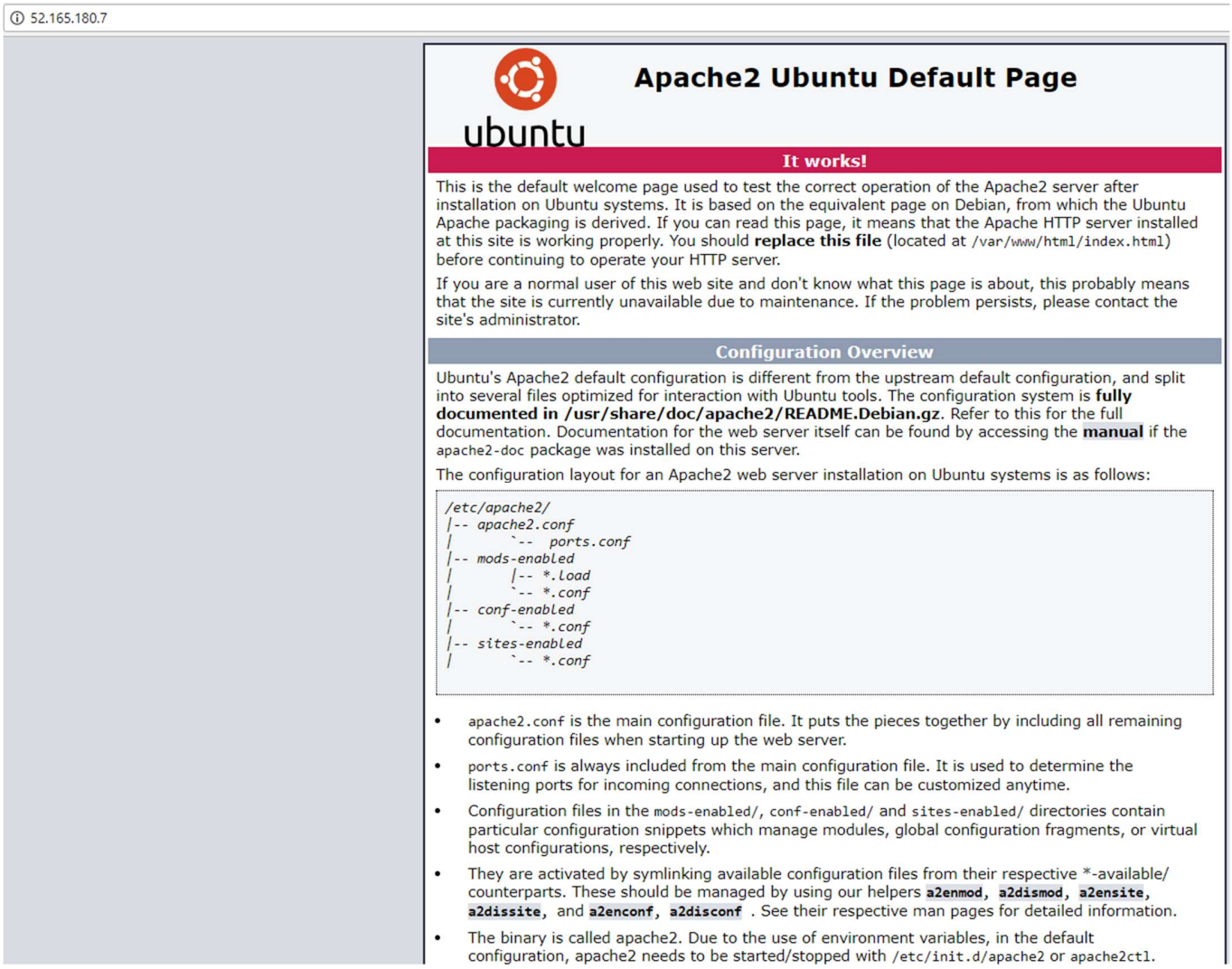
Selecting the name of your **Application Gateway** that was created when you launched the Transit VNet Spoke template. You can find the name of your **Resource Group** to help you differentiate from any other Application Gateways.



Locale the **Public IP address** for your Application Gateway.



Place the **Public IP address** in your web browser. This IP address is the public facing IP of the Application Gateway Load Balancer. You will see the default Ubuntu Page.



9. Cleanup

You can clean up the setup by deleting the **resource groups** for both the hub and spoke deployments. Once you have deleted the resource groups for both the hub and spoke you will have successfully deleted all resources created in this deployment.