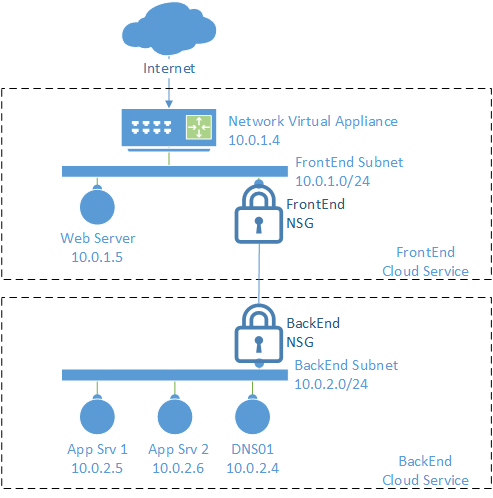
# Example 2 – Build a DMZ to protect applications with a Firewall and NSGs

[Return to the Security Boundary Best Practices Page](/documentation/articles/best-practices-network-security)

This example will create a DMZ with a firewall, four windows servers, and Network Security Groups. It will also walk through each of the relevant commands to provide a deeper understanding of each step. There is a also a Traffic Scenario section to provide a in-depth step-by-step how traffic proceeds through the layers of defense in the DMZ. Finally, in the references section is the complete code and instruction to build this environment to test and experiment with various scenarios.



Inbound DMZ with NVA and NSG

## Environment Description

In this example there is a subscription that contains the following:

* Two cloud services: “FrontEnd001” and “BackEnd001”
* A Virtual Network, “CorpNetwork”, with two subnets; “FrontEnd” and “BackEnd”
* A single Network Security Group that is applied to both subnets
* A network virtual appliance, in this example a Barracuda NG Firewall, connected to the Frontend subnet
* A Windows Server that represents an application web server (“IIS01”)
* Two windows servers that represent application back end servers (“AppVM01”, “AppVM02”)
* A Windows server that represents a DNS server (“DNS01”)

[AZURE.NOTE] Although this example uses a Barracuda NG Firewall, many of the different Network Virtual Appliances could be used for this example.

In the references section below there is a PowerShell script that will build most of the environment described above. Building the VMs and Virtual Networks, although are done by the example script, are not described in detail in this document.

To build the environment;

1. Save the network config xml file included in the references section (updated with names, location, and IP addresses to match the given scenario)
2. Update the user variables in the script to match the environment the script is to be run against (subscriptions, service names, etc)
3. Execute the script in PowerShell

**Note**: The region signified in the PowerShell script must match the region signified in the network configuration xml file.

Once the script runs successfully the following post-script steps may be taken;

1. Set up the firewall rules, this is covered in the section below titled: Firewall Rules.
2. Optionally in the references section are two scripts to set up the web server and app server with a simple web application to allow testing with this DMZ configuration.

The next section explains most of the scripts statements relative to Network Security Groups.

## Network Security Groups (NSG)

For this example, a NSG group is built and then loaded with six rules.

[AZURE.TIP] Generally speaking, you should create your specific “Allow” rules first and then the more generic “Deny” rules last. The assigned priority dictates which rules are evaluated first. Once traffic is found to apply to a specific rule, no further rules are evaluated. NSG rules can apply in either in the inbound or outbound direction (from the perspective of the subnet).

Declaratively, the following rules are being built for inbound traffic:

1. Internal DNS traffic (port 53) is allowed
2. RDP traffic (port 3389) from the Internet to any VM is allowed
3. HTTP traffic (port 80) from the Internet to the NVA (firewall) is allowed
4. Any traffic (all ports) from IIS01 to AppVM1 is allowed
5. Any traffic (all ports) from the Internet to the entire VNet (both subnets) is Denied
6. Any traffic (all ports) from the Frontend subnet to the Backend subnet is Denied

With these rules bound to each subnet, if a HTTP request was inbound from the Internet to the web server, both rules 3 (allow) and 5 (deny) would apply, but since rule 3 has a higher priority only it would apply and rule 5 would not come into play. Thus the HTTP request would be allowed to the firewall. If that same traffic was trying to reach the DNS01 server, rule 5 (Deny) would be the first to apply and the traffic would not be allowed to pass to the server. Rule 6 (Deny) blocks the Frontend subnet from talking to the Backend subnet (except for allowed traffic in rules 1 and 4), this protects the Backend network in case an attacker compromises the web application on the Frontend, the attacker would have limited access to the Backend “protected” network (only to resources exposed on the AppVM01 server).

There is a default outbound rule that allows traffic out to the internet. For this example, we’re allowing outbound traffic and not modifying any outbound rules. To lock down traffic in both directions, User Defined Routing is required, this is explored in a different example that can found in the [main security boundary document](/documentation/articles/best-practices-network-security).

The above discussed NSG rules are very similar to the NSG rules in [Example 1 - Build a Simple DMZ with NSGs](./virtual-networks-dmz-nsg-asm.md). Please review the NSG Description in that document for a detailed look at each NSG rule and it’s attributes.

## Firewall Rules

A management client will need to be installed on a PC to manage the firewall and create the configurations needed. See the documentation from your firewall (or other NVA) vendor on how to manage the device. The remainder of this section will describe the configuration of the firewall itself, through the vendors management client (i.e. not the Azure Management Portal or PowerShell).

Instructions for client download and connecting to the Barracuda used in this example can be found here: [Barracuda NG Admin](https://techlib.barracuda.com/NG61/NGAdmin)

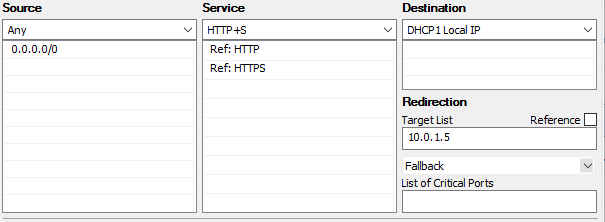
On the firewall, forwarding rules will need to be created. Since this example only routes internet traffic in-bound to the firewall and then to the web server, only one forwarding NAT rule is needed. On the Barracuda NG firewall used in this example the rule would be a Destination NAT rule (“Dst NAT”) to pass this traffic.

To create the following rule (or verify existing default rules), starting from the Barracuda NG Admin client dashboard, navigate to the configuration tab, in the Operational Configuration section click Ruleset. A grid called, “Main Rules” will show the existing active and deactivated rules on the firewall. In the upper right corner of this grid is a small, green “+” button, click this to create a new rule (Note: your firewall may be “locked” for changes, if you see a button marked “Lock” and you are unable to create or edit rules, click this button to “unlock” the ruleset and allow editing). If you wish to edit an existing rule, select that rule, right-click and select Edit Rule.

Create a new rule and provide a name, such as “WebTraffic”.

The Destination NAT rule icon looks like this: Destination NAT Icon

The rule itself would look something like this:



Firewall Rule

Here any inbound address that hits the Firewall trying to reach HTTP (port 80 or 443 for HTTPS) will be sent out the Firewall’s “DHCP1 Local IP” interface and redirected to the Web Server with the IP Address of 10.0.1.5. Since the traffic is coming in on port 80 and going to the web server on port 80 no port change was needed. However, the Target List could have been 10.0.1.5:8080 if our Web Server listened on port 8080 thus translating the inbound port 80 on the firewall to inbound port 8080 on the web server.

A Connection Method should also be signified, for the Destination Rule from the Internet, “Dynamic SNAT” is most appropriate.

Although only one rule has been created it’s important that its priority is set correctly. If in the grid of all rules on the firewall this new rule is on the bottom (below the “BLOCKALL” rule) it will never come into play. Ensure the newly created rule for web traffic is above the BLOCKALL rule.

Once the rule is created, it must be pushed to the firewall and then activated, if this is not done the rule change will not take effect. The push and activation process is described in the next section.

## Rule Activation

With the ruleset modified to add this rule, the ruleset must be uploaded to the firewall and activated.



Firewall Rule Activation

In the upper right hand corner of the management client are a cluster of buttons. Click the “Send Changes” button to send the modified rules to the firewall, then click the “Activate” button.

With the activation of the firewall ruleset this example environment build is complete. Optionally, the post build scripts in the References section can be run to add an application to this environment to test the below traffic scenarios.

[AZURE.IMPORTANT] It is critical to realize that you will not hit the web server directly. When a browser requests an HTTP page from FrontEnd001.CloudApp.Net, the HTTP endpoint (port 80) passes this traffic to the firewall not the web server. The firewall then, according to the rule created above, NATs that request to the Web Server.

## Traffic Scenarios

#### (Allowed) Web to Web Server through Firewall

1. Internet user requests HTTP page from FrontEnd001.CloudApp.Net (Internet Facing Cloud Service)
2. Cloud service passes traffic through open endpoint on port 80 to firewall local interface on 10.0.1.4:80
3. Frontend subnet begins inbound rule processing:
4. NSG Rule 1 (DNS) doesn’t apply, move to next rule
5. NSG Rule 2 (RDP) doesn’t apply, move to next rule
6. NSG Rule 3 (Internet to Firewall) does apply, traffic is allowed, stop rule processing
7. Traffic hits internal IP address of the firewall (10.0.1.4)
8. Firewall forwarding rule see this is port 80 traffic, redirects it to the web server IIS01
9. IIS01 is listening for web traffic, receives this request and starts processing the request
10. IIS01 asks the SQL Server on AppVM01 for information
11. No outbound rules on Frontend subnet, traffic is allowed
12. The Backend subnet begins inbound rule processing:
13. NSG Rule 1 (DNS) doesn’t apply, move to next rule
14. NSG Rule 2 (RDP) doesn’t apply, move to next rule
15. NSG Rule 3 (Internet to Firewall) doesn’t apply, move to next rule
16. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
17. AppVM01 receives the SQL Query and responds
18. Since there are no outbound rules on the Backend subnet the response is allowed
19. Frontend subnet begins inbound rule processing:
20. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
21. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
22. The IIS server receives the SQL response and completes the HTTP response and sends to the requestor
23. Since this is a NAT session from the firewall, the response destination (initially) is for the Firewall
24. The firewall receives the response from the Web Server and forwards back to the Internet User
25. Since there are no outbound rules on the Frontend subnet the response is allowed, and the Internet User receives the web page requested.

#### (Allowed) RDP to Backend

1. Server Admin on internet requests RDP session to AppVM01 on BackEnd001.CloudApp.Net:xxxxx where xxxxx is the randomly assigned port number for RDP to AppVM01 (the assigned port can be found on the Azure Management Portal or via PowerShell)
2. Since the Firewall is only listening on the FrontEnd001.CloudApp.Net address, it is not involved with this traffic flow
3. Backend subnet begins inbound rule processing:
4. NSG Rule 1 (DNS) doesn’t apply, move to next rule
5. NSG Rule 2 (RDP) does apply, traffic is allowed, stop rule processing
6. With no outbound rules, default rules apply and return traffic is allowed
7. RDP session is enabled
8. AppVM01 prompts for user name password

#### (Allowed) Web Server DNS lookup on DNS server

1. Web Server, IIS01, needs a data feed at www.data.gov, but needs to resolve the address.
2. The network configuration for the VNet lists DNS01 (10.0.2.4 on the Backend subnet) as the primary DNS server, IIS01 sends the DNS request to DNS01
3. No outbound rules on Frontend subnet, traffic is allowed
4. Backend subnet begins inbound rule processing:
5. NSG Rule 1 (DNS) does apply, traffic is allowed, stop rule processing
6. DNS server receives the request
7. DNS server doesn’t have the address cached and asks a root DNS server on the internet
8. No outbound rules on Backend subnet, traffic is allowed
9. Internet DNS server responds, since this session was initiated internally, the response is allowed
10. DNS server caches the response, and responds to the initial request back to IIS01
11. No outbound rules on Backend subnet, traffic is allowed
12. Frontend subnet begins inbound rule processing:
13. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
14. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed
15. IIS01 receives the response from DNS01

#### (Allowed) Web Server access file on AppVM01

1. IIS01 asks for a file on AppVM01
2. No outbound rules on Frontend subnet, traffic is allowed
3. The Backend subnet begins inbound rule processing:
4. NSG Rule 1 (DNS) doesn’t apply, move to next rule
5. NSG Rule 2 (RDP) doesn’t apply, move to next rule
6. NSG Rule 3 (Internet to Firewall) doesn’t apply, move to next rule
7. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
8. AppVM01 receives the request and responds with file (assuming access is authorized)
9. Since there are no outbound rules on the Backend subnet the response is allowed
10. Frontend subnet begins inbound rule processing:
11. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
12. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
13. The IIS server receives the file

#### (Denied) Web direct to Web Server

Since the Web Server, IIS01, and the Firewall are in the same Cloud Service they share the same public facing IP address. Thus any HTTP traffic would be directed to the firewall. While the request would be successfully served, it cannot go directly to the Web Server, it passed, as designed, through the Firewall first. See the first Scenario in this section for the traffic flow.

#### (Denied) Web to Backend Server

1. Internet user tries to access a file on AppVM01 through the BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for file share, this would not pass the Cloud Service and wouldn’t reach the server
3. If the endpoints were open for some reason, NSG rule 5 (Internet to VNet) would block this traffic

#### (Denied) Web DNS lookup on DNS server

1. Internet user tries to lookup an internal DNS record on DNS01 through the BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for DNS, this would not pass the Cloud Service and wouldn’t reach the server
3. If the endpoints were open for some reason, NSG rule 5 (Internet to VNet) would block this traffic (Note: that Rule 1 (DNS) would not apply for two reasons, first the source address is the internet, this rule only applies to the local VNet as the source, also this is an Allow rule, so it would never deny traffic)

#### (Denied) Web to SQL access through Firewall

1. Internet user requests SQL data from FrontEnd001.CloudApp.Net (Internet Facing Cloud Service)
2. Since there are no endpoints open for SQL, this would not pass the Cloud Service and wouldn’t reach the firewall
3. If endpoints were open for some reason, the Frontend subnet begins inbound rule processing:
4. NSG Rule 1 (DNS) doesn’t apply, move to next rule
5. NSG Rule 2 (RDP) doesn’t apply, move to next rule
6. NSG Rule 2 (Internet to Firewall) does apply, traffic is allowed, stop rule processing
7. Traffic hits internal IP address of the firewall (10.0.1.4)
8. Firewall has no forwarding rules for SQL and drops the traffic

## Conclusion

This is a relatively straight forward way of protecting your application with a firewall and isolating the back end subnet from inbound traffic.

More examples and an overview of network security boundaries can be found [here](/documentation/articles/best-practices-network-security).

## References

### Main Script and Network Config

Save the Full Script in a PowerShell script file. Save the Network Config into a file named “NetworkConf2.xml”. Modify the user defined variables as needed. Run the script, then follow the Firewall rule setup instruction above.

#### Full Script

This script will, based on the user defined variables;

1. Connect to an Azure subscription
2. Create a new storage account
3. Create a new VNet and two subnets as defined in the Network Config file
4. Build 4 windows server VMs
5. Configure NSG including:

* Creating a NSG
* Populating it with rules
* Binding the NSG to the appropriate subnets

This PowerShell script should be run locally on an internet connected PC or server.

[AZURE.IMPORTANT] When this script is run, there may be warnings or other informational messages that pop in PowerShell. Only error messages in red are cause for concern.

<#   
 .SYNOPSIS  
 Example of DMZ and Network Security Groups in an isolated network (Azure only, no hybrid connections)  
  
 .DESCRIPTION  
 This script will build out a sample DMZ setup containing:  
 - A default storage account for VM disks  
 - Two new cloud services  
 - Two Subnets (FrontEnd and BackEnd subnets)  
 - A Network Virtual Appliance (NVA), in this case a Barracuda NG Firewall  
 - One server on the FrontEnd Subnet (plus the NVA on the FrontEnd subnet)  
 - Three Servers on the BackEnd Subnet  
 - Network Security Groups to allow/deny traffic patterns as declared  
   
 Before running script, ensure the network configuration file is created in  
 the directory referenced by $NetworkConfigFile variable (or update the  
 variable to reflect the path and file name of the config file being used).  
  
 .Notes  
 Security requirements are different for each use case and can be addressed in a  
 myriad of ways. Please be sure that any sensitive data or applications are behind  
 the appropriate layer(s) of protection. This script serves as an example of some  
 of the techniques that can be used, but should not be used for all scenarios. You  
 are responsible to assess your security needs and the appropriate protections  
 needed, and then effectively implement those protections.  
  
 FrontEnd Service (FrontEnd subnet 10.0.1.0/24)  
 myFirewall - 10.0.1.4  
 IIS01 - 10.0.1.5  
   
 BackEnd Service (BackEnd subnet 10.0.2.0/24)  
 DNS01 - 10.0.2.4  
 AppVM01 - 10.0.2.5  
 AppVM02 - 10.0.2.6  
  
#>  
  
# Fixed Variables  
 $LocalAdminPwd = Read-Host -Prompt "Enter Local Admin Password to be used for all VMs"  
 $VMName = @()  
 $ServiceName = @()  
 $VMFamily = @()  
 $img = @()  
 $size = @()  
 $SubnetName = @()  
 $VMIP = @()  
  
# User Defined Global Variables  
 # These should be changes to reflect your subscription and services  
 # Invalid options will fail in the validation section  
  
 # Subscription Access Details  
 $subID = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
  
 # VM Account, Location, and Storage Details  
 $LocalAdmin = "theAdmin"  
 $DeploymentLocation = "China North"  
 $StorageAccountName = "vmstore02"  
  
 # Service Details  
 $FrontEndService = "FrontEnd001"  
 $BackEndService = "BackEnd001"  
  
 # Network Details  
 $VNetName = "CorpNetwork"  
 $FESubnet = "FrontEnd"  
 $FEPrefix = "10.0.1.0/24"  
 $BESubnet = "BackEnd"  
 $BEPrefix = "10.0.2.0/24"  
 $NetworkConfigFile = "C:\Scripts\NetworkConf2.xml"  
  
 # VM Base Disk Image Details  
 $SrvImg = Get-AzureVMImage | Where {$\_.ImageFamily -match 'Windows Server 2012 R2 Datacenter'} | sort PublishedDate -Descending | Select ImageName -First 1 | ForEach {$\_.ImageName}  
 $FWImg = Get-AzureVMImage | Where {$\_.ImageFamily -match 'Barracuda NG Firewall'} | sort PublishedDate -Descending | Select ImageName -First 1 | ForEach {$\_.ImageName}  
  
 # NSG Details  
 $NSGName = "MyVNetSG"  
  
# User Defined VM Specific Config  
 # Note: To ensure proper NSG Rule creation later in this script:  
 # - The Web Server must be VM 1  
 # - The AppVM1 Server must be VM 2  
 # - The DNS server must be VM 4  
 #  
 # Otherwise the NSG rules in the last section of this  
 # script will need to be changed to match the modified  
 # VM array numbers ($i) so the NSG Rule IP addresses  
 # are aligned to the associated VM IP addresses.  
  
 # VM 0 - The Network Virtual Appliance (NVA)  
 $VMName += "myFirewall"  
 $ServiceName += $FrontEndService  
 $VMFamily += "Firewall"  
 $img += $FWImg  
 $size += "Small"  
 $SubnetName += $FESubnet  
 $VMIP += "10.0.1.4"  
  
 # VM 1 - The Web Server  
 $VMName += "IIS01"  
 $ServiceName += $FrontEndService  
 $VMFamily += "Windows"  
 $img += $SrvImg  
 $size += "Standard\_D3"  
 $SubnetName += $FESubnet  
 $VMIP += "10.0.1.5"  
  
 # VM 2 - The First Appliaction Server  
 $VMName += "AppVM01"  
 $ServiceName += $BackEndService  
 $VMFamily += "Windows"  
 $img += $SrvImg  
 $size += "Standard\_D3"  
 $SubnetName += $BESubnet  
 $VMIP += "10.0.2.5"  
  
 # VM 3 - The Second Appliaction Server  
 $VMName += "AppVM02"  
 $ServiceName += $BackEndService  
 $VMFamily += "Windows"  
 $img += $SrvImg  
 $size += "Standard\_D3"  
 $SubnetName += $BESubnet  
 $VMIP += "10.0.2.6"  
  
 # VM 4 - The DNS Server  
 $VMName += "DNS01"  
 $ServiceName += $BackEndService  
 $VMFamily += "Windows"  
 $img += $SrvImg  
 $size += "Standard\_D3"  
 $SubnetName += $BESubnet  
 $VMIP += "10.0.2.4"  
  
# ----------------------------- #  
# No User Defined Varibles or #  
# Configuration past this point #  
# ----------------------------- #  
  
 # Get your Azure accounts  
 Add-AzureAccount  
 Set-AzureSubscription –SubscriptionId $subID -ErrorAction Stop  
 Select-AzureSubscription -SubscriptionId $subID -Current -ErrorAction Stop  
  
 # Create Storage Account  
 If (Test-AzureName -Storage -Name $StorageAccountName) {   
 Write-Host "Fatal Error: This storage account name is already in use, please pick a diffrent name." -ForegroundColor Red  
 Return}  
 Else {Write-Host "Creating Storage Account" -ForegroundColor Cyan   
 New-AzureStorageAccount -Location $DeploymentLocation -StorageAccountName $StorageAccountName}  
  
 # Update Subscription Pointer to New Storage Account  
 Write-Host "Updating Subscription Pointer to New Storage Account" -ForegroundColor Cyan   
 Set-AzureSubscription –SubscriptionId $subID -CurrentStorageAccountName $StorageAccountName -ErrorAction Stop  
  
# Validation  
$FatalError = $false  
  
If (-Not (Get-AzureLocation | Where {$\_.DisplayName -eq $DeploymentLocation})) {  
 Write-Host "This Azure Location was not found or available for use" -ForegroundColor Yellow  
 $FatalError = $true}  
  
If (Test-AzureName -Service -Name $FrontEndService) {   
 Write-Host "The FrontEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow  
 $FatalError = $true}  
Else { Write-Host "The FrontEndService service name is valid for use." -ForegroundColor Green}  
  
If (Test-AzureName -Service -Name $BackEndService) {   
 Write-Host "The BackEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow  
 $FatalError = $true}  
Else { Write-Host "The BackEndService service name is valid for use." -ForegroundColor Green}  
  
If (-Not (Test-Path $NetworkConfigFile)) {   
 Write-Host 'The network config file was not found, please update the $NetworkConfigFile variable to point to the network config xml file.' -ForegroundColor Yellow  
 $FatalError = $true}  
Else { Write-Host "The network config file was found" -ForegroundColor Green  
 If (-Not (Select-String -Pattern $DeploymentLocation -Path $NetworkConfigFile)) {  
 Write-Host 'The deployment location was not found in the network config file, please check the network config file to ensure the $DeploymentLocation varible is correct and the netowrk config file matches.' -ForegroundColor Yellow  
 $FatalError = $true}  
 Else { Write-Host "The deployment location was found in the network config file." -ForegroundColor Green}}  
  
If ($FatalError) {  
 Write-Host "A fatal error has occured, please see the above messages for more information." -ForegroundColor Red  
 Return}  
Else { Write-Host "Validation passed, now building the environment." -ForegroundColor Green}  
  
# Create VNET  
 Write-Host "Creating VNET" -ForegroundColor Cyan   
 Set-AzureVNetConfig -ConfigurationPath $NetworkConfigFile -ErrorAction Stop  
  
# Create Services  
 Write-Host "Creating Services" -ForegroundColor Cyan  
 New-AzureService -Location $DeploymentLocation -ServiceName $FrontEndService -ErrorAction Stop  
 New-AzureService -Location $DeploymentLocation -ServiceName $BackEndService -ErrorAction Stop  
  
# Build VMs  
 $i=0  
 $VMName | Foreach {  
 Write-Host "Building $($VMName[$i])" -ForegroundColor Cyan  
 If ($VMFamily[$i] -eq "Firewall")   
 {   
 New-AzureVMConfig -Name $VMName[$i] -ImageName $img[$i] –InstanceSize $size[$i] | `  
 Add-AzureProvisioningConfig -Linux -LinuxUser $LocalAdmin -Password $LocalAdminPwd | `  
 Set-AzureSubnet –SubnetNames $SubnetName[$i] | `  
 Set-AzureStaticVNetIP -IPAddress $VMIP[$i] | `  
 New-AzureVM –ServiceName $ServiceName[$i] -VNetName $VNetName -Location $DeploymentLocation  
 # Set up all the EndPoints we'll need once we're up and running  
 # Note: Web traffic goes through the firewall, so we'll need to set up a HTTP endpoint.  
 # Also, the firewall will be redirecting web traffic to a new IP and Port in a  
 # forwarding rule, so the HTTP endpoint here will have the same public and local  
 # port and the firewall will do the NATing and redirection as declared in the  
 # firewall rule.  
 Add-AzureEndpoint -Name "MgmtPort1" -Protocol tcp -PublicPort 801 -LocalPort 801 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM  
 Add-AzureEndpoint -Name "MgmtPort2" -Protocol tcp -PublicPort 807 -LocalPort 807 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM  
 Add-AzureEndpoint -Name "HTTP" -Protocol tcp -PublicPort 80 -LocalPort 80 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM  
 # Note: A SSH endpoint is automatically created on port 22 when the appliance is created.  
 }  
 Else  
 {  
 New-AzureVMConfig -Name $VMName[$i] -ImageName $img[$i] –InstanceSize $size[$i] | `  
 Add-AzureProvisioningConfig -Windows -AdminUsername $LocalAdmin -Password $LocalAdminPwd | `  
 Set-AzureSubnet –SubnetNames $SubnetName[$i] | `  
 Set-AzureStaticVNetIP -IPAddress $VMIP[$i] | `  
 Set-AzureVMMicrosoftAntimalwareExtension -AntimalwareConfiguration '{"AntimalwareEnabled" : true}' | `  
 Remove-AzureEndpoint -Name "PowerShell" | `  
 New-AzureVM –ServiceName $ServiceName[$i] -VNetName $VNetName -Location $DeploymentLocation  
 # Note: A Remote Desktop endpoint is automatically created when each VM is created.  
 }  
 $i++  
 }  
  
# Configure NSG  
 Write-Host "Configuring the Network Security Group (NSG)" -ForegroundColor Cyan  
   
 # Build the NSG  
 Write-Host "Building the NSG" -ForegroundColor Cyan  
 New-AzureNetworkSecurityGroup -Name $NSGName -Location $DeploymentLocation -Label "Security group for $VNetName subnets in $DeploymentLocation"  
  
 # Add NSG Rules  
 Write-Host "Writing rules into the NSG" -ForegroundColor Cyan  
 Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable Internal DNS" -Type Inbound -Priority 100 -Action Allow `  
 -SourceAddressPrefix VIRTUAL\_NETWORK -SourcePortRange '\*' `  
 -DestinationAddressPrefix $VMIP[4] -DestinationPortRange '53' `  
 -Protocol \*  
  
 Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable RDP to $VNetName VNet" -Type Inbound -Priority 110 -Action Allow `  
 -SourceAddressPrefix INTERNET -SourcePortRange '\*' `  
 -DestinationAddressPrefix VIRTUAL\_NETWORK -DestinationPortRange '3389' `  
 -Protocol \*  
  
 Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable Internet to $($VMName[0])" -Type Inbound -Priority 120 -Action Allow `  
 -SourceAddressPrefix Internet -SourcePortRange '\*' `  
 -DestinationAddressPrefix $VMIP[0] -DestinationPortRange '\*' `  
 -Protocol \*  
  
 Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable $($VMName[1]) to $($VMName[2])" -Type Inbound -Priority 130 -Action Allow `  
 -SourceAddressPrefix $VMIP[1] -SourcePortRange '\*' `  
 -DestinationAddressPrefix $VMIP[2] -DestinationPortRange '\*' `  
 -Protocol \*  
   
 Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Isolate the $VNetName VNet from the Internet" -Type Inbound -Priority 140 -Action Deny `  
 -SourceAddressPrefix INTERNET -SourcePortRange '\*' `  
 -DestinationAddressPrefix VIRTUAL\_NETWORK -DestinationPortRange '\*' `  
 -Protocol \*  
  
 Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Isolate the $FESubnet subnet from the $BESubnet subnet" -Type Inbound -Priority 150 -Action Deny `  
 -SourceAddressPrefix $FEPrefix -SourcePortRange '\*' `  
 -DestinationAddressPrefix $BEPrefix -DestinationPortRange '\*' `  
 -Protocol \*  
  
 # Assign the NSG to the Subnets  
 Write-Host "Binding the NSG to both subnets" -ForegroundColor Cyan  
 Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $FESubnet -VirtualNetworkName $VNetName  
 Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $BESubnet -VirtualNetworkName $VNetName  
  
# Optional Post-script Manual Configuration  
 # Configure Firewall  
 # Install Test Web App (Run Post-Build Script on the IIS Server)  
 # Install Backend resource (Run Post-Build Script on the AppVM01)  
 Write-Host  
 Write-Host "Build Complete!" -ForegroundColor Green  
 Write-Host  
 Write-Host "Optional Post-script Manual Configuration Steps" -ForegroundColor Gray  
 Write-Host " - Configure Firewall" -ForegroundColor Gray  
 Write-Host " - Install Test Web App (Run Post-Build Script on the IIS Server)" -ForegroundColor Gray  
 Write-Host " - Install Backend resource (Run Post-Build Script on the AppVM01)" -ForegroundColor Gray  
 Write-Host

#### Network Config File

Save this xml file with updated location and add the link to this file to the $NetworkConfigFile variable in the script above.

<NetworkConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">  
 <VirtualNetworkConfiguration>  
 <Dns>  
 <DnsServers>  
 <DnsServer name="DNS01" IPAddress="10.0.2.4" />  
 <DnsServer name="Level3" IPAddress="209.244.0.3" />  
 </DnsServers>  
 </Dns>  
 <VirtualNetworkSites>  
 <VirtualNetworkSite name="CorpNetwork" Location="China North">  
 <AddressSpace>  
 <AddressPrefix>10.0.0.0/16</AddressPrefix>  
 </AddressSpace>  
 <Subnets>  
 <Subnet name="FrontEnd">  
 <AddressPrefix>10.0.1.0/24</AddressPrefix>  
 </Subnet>  
 <Subnet name="BackEnd">  
 <AddressPrefix>10.0.2.0/24</AddressPrefix>  
 </Subnet>  
 </Subnets>  
 <DnsServersRef>  
 <DnsServerRef name="DNS01" />  
 <DnsServerRef name="Level3" />  
 </DnsServersRef>  
 </VirtualNetworkSite>  
 </VirtualNetworkSites>  
 </VirtualNetworkConfiguration>  
</NetworkConfiguration>

#### Sample Application Scripts

If you wish to install a sample application for this, and other DMZ Examples, one has been provided at the following link: [Sample Application Script](./virtual-networks-sample-app.md)