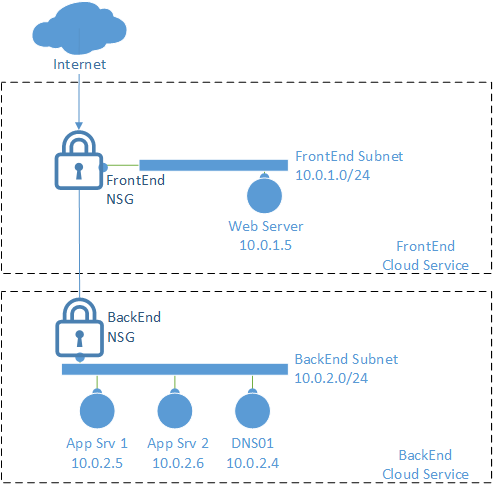
# Example 1 – Build a Simple DMZ with NSGs

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

This example will create a simple DMZ with 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 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 Network Security Group that is applied to both subnets
* 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”)

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 additional optional steps may be taken, 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 following sections will provide a detailed description of Network Security Groups and how they function for this example by walking through key lines of the PowerShell script.

## 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 web server (IIS01) 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 web server. 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 “Example 3” below.

Each rule is discussed in more detail as follows (Note; any item in the below list in beginning with a dollar sign (e.g.: $NSGName) is a user defined variable from the script in the reference section of this document):

1. First a Network Security Group must be built to hold the rules:

* New-AzureNetworkSecurityGroup -Name $NSGName `  
   -Location $DeploymentLocation `  
   -Label "Security group for $VNetName subnets in $DeploymentLocation"

1. The first rule in this example will allow DNS traffic between all internal networks to the DNS server on the backend subnet. The rule has some important parameters:

* “Type” signifies in which direction of traffic this rule will take effect; this is from the perspective of the subnet or Virtual Machine (depending on where this NSG is bound). Thus if Type is “Inbound” and traffic is entering the subnet, the rule would apply and traffic leaving the subnet would not be affected by this rule.
* “Priority” sets the order in which a traffic flow will be evaluated. The lower the number the higher the priority. As soon as a rule applies to a specific traffic flow, no further rules are processed. Thus if a rule with priority 1 allows traffic, and a rule with priority 2 denies traffic, and both rules apply to traffic then the traffic would be allowed to flow (since rule 1 had a higher priority it took effect and no further rules were applied).
* “Action” signifies if traffic affected by this rule is blocked or allowed.
* 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 \*

1. This rule will allow RDP traffic to flow from the internet to the RDP port on any server on either subnet in the VNET. This rule uses two special types of address prefixes; “VIRTUAL\_NETWORK” and “INTERNET”. This is an easy way to address a larger category of address prefixes.

* 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 \*

1. This rule allows inbound internet traffic to hit the web server. This doesn’t change the routing behavior; it only allows traffic destine for IIS01 to pass. Thus if traffic from the Internet had the web server as its destination this rule would allow it and stop processing further rules. (In the rule at priority 140 all other inbound internet traffic is blocked). If you’re only processing HTTP traffic, this rule could be further restricted to only allow Destination Port 80.

* 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 \*

1. This rule allows traffic to pass from the IIS01 server to the AppVM01 server, a later rule blocks all other Frontend to Backend traffic. To improve this rule, if the port is known that should be added. For example, if the IIS server is hitting only SQL Server on AppVM01, the Destination Port Range should be changed from “\*” (Any) to 1433 (the SQL port) thus allowing a smaller inbound attack surface on AppVM01 should the web application ever be compromised.

* 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 \*

1. This rule denies traffic from the internet to any servers on the network. In combination with the rule at priority 110 and 120, allows only inbound internet traffic to the firewall and RDP ports to other servers and blocks everything else.

* 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 \*

1. The final rule denies traffic from the Frontend subnet to the Backend subnet. Since this is an Inbound only rule, reverse traffic is allowed (from the Backend to the Frontend).

* 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 \*

## Traffic Scenarios

#### (*Allowed*) Web to Web Server

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 towards IIS01 (the web server)
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 IIS01) does apply, traffic is allowed, stop rule processing
7. Traffic hits internal IP address of the web server IIS01 (10.0.1.5)
8. IIS01 is listening for web traffic, receives this request and starts processing the request
9. IIS01 asks the SQL Server on AppVM01 for information
10. No outbound rules on Frontend subnet, traffic is allowed
11. The Backend subnet begins inbound rule processing:
12. NSG Rule 1 (DNS) doesn’t apply, move to next rule
13. NSG Rule 2 (RDP) doesn’t apply, move to next rule
14. NSG Rule 3 (Internet to Firewall) doesn’t apply, move to next rule
15. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
16. AppVM01 receives the SQL Query and responds
17. Since there are no outbound rules on the Backend subnet the response is allowed
18. Frontend subnet begins inbound rule processing:
19. 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
20. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
21. The IIS server receives the SQL response and completes the HTTP response and sends to the requestor
22. 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. Backend subnet begins inbound rule processing:
3. NSG Rule 1 (DNS) doesn’t apply, move to next rule
4. NSG Rule 2 (RDP) does apply, traffic is allowed, stop rule processing
5. With no outbound rules, default rules apply and return traffic is allowed
6. RDP session is enabled
7. 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 IIS01) 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 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 3 (Internet to IIS01) does apply, traffic is allowed, stop rule processing
7. Traffic hits internal IP address of the IIS01 (10.0.1.5)
8. IIS01 isn’t listening on port 1433, so no response to the request

## Conclusion

This is a relatively simple and straight forward way of 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 “NetworkConf1.xml”. Modify the user defined variables as needed. Run the script, then follow the Firewall rule setup instruction contained in the Example 1 section 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 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)  
 - One server 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)  
 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\NetworkConf1.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}  
   
 # 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 0  
 # - The AppVM1 Server must be VM 1  
 # - The DNS server must be VM 3  
 #  
 # 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 Web Server  
 $VMName += "IIS01"  
 $ServiceName += $FrontEndService  
 $VMFamily += "Windows"  
 $img += $SrvImg  
 $size += "Standard\_D3"  
 $SubnetName += $FESubnet  
 $VMIP += "10.0.1.5"  
  
 # VM 1 - The First Application Server  
 $VMName += "AppVM01"  
 $ServiceName += $BackEndService  
 $VMFamily += "Windows"  
 $img += $SrvImg  
 $size += "Standard\_D3"  
 $SubnetName += $BESubnet  
 $VMIP += "10.0.2.5"  
  
 # VM 2 - The Second Application Server  
 $VMName += "AppVM02"  
 $ServiceName += $BackEndService  
 $VMFamily += "Windows"  
 $img += $SrvImg  
 $size += "Standard\_D3"  
 $SubnetName += $BESubnet  
 $VMIP += "10.0.2.6"  
  
 # VM 3 - 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  
 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++  
 }  
 # Add HTTP Endpoint for IIS01  
 Get-AzureVM -ServiceName $ServiceName[0] -Name $VMName[0] | Add-AzureEndpoint -Name HTTP -Protocol tcp -LocalPort 80 -PublicPort 80 | Update-AzureVM  
  
# 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[3] -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[0]) to $($VMName[1])" -Type Inbound -Priority 130 -Action Allow `  
 -SourceAddressPrefix $VMIP[0] -SourcePortRange '\*' `  
 -DestinationAddressPrefix $VMIP[1] -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  
 # 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 " - 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)