How to load balance Windows virtual machines in Azure to create a highly available application

08/

Load balancing provides a higher level of availability by spreading incoming requests across multiple virtual machines. In this tutorial, you learn about the different components of the Azure load balancer that distribute traffic and provide high availability. You learn how to:

* Create an Azure load balancer
* Create a load balancer health probe
* Create load balancer traffic rules
* Use the Custom Script Extension to create a basic IIS site
* Create virtual machines and attach to a load balancer
* View a load balancer in action
* Add and remove VMs from a load balancer

This tutorial requires the Azure PowerShell module version 3.6 or later. Run Get-Module -ListAvailable AzureRM to find the version. If you need to upgrade, see [Install Azure PowerShell module](https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps).

Azure load balancer overview

An Azure load balancer is a Layer-4 (TCP, UDP) load balancer that provides high availability by distributing incoming traffic among healthy VMs. A load balancer health probe monitors a given port on each VM and only distributes traffic to an operational VM.

You define a front-end IP configuration that contains one or more public IP addresses. This front-end IP configuration allows your load balancer and applications to be accessible over the Internet.

Virtual machines connect to a load balancer using their virtual network interface card (NIC). To distribute traffic to the VMs, a back-end address pool contains the IP addresses of the virtual (NICs) connected to the load balancer.

To control the flow of traffic, you define load balancer rules for specific ports and protocols that map to your VMs.

Create Azure load balancer

This section details how you can create and configure each component of the load balancer. Before you can create your load balancer, create a resource group with [New-AzureRmResourceGroup](https://docs.microsoft.com/en-us/powershell/module/azurerm.resources/new-azurermresourcegroup). The following example creates a resource group named *myResourceGroupLoadBalancer* in the *EastUS* location:

PowerShellCopy

New-AzureRmResourceGroup `

-ResourceGroupName myResourceGroupLoadBalancer `

-Location EastUS

Create a public IP address

To access your app on the Internet, you need a public IP address for the load balancer. Create a public IP address with [New-AzureRmPublicIpAddress](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermpublicipaddress). The following example creates a public IP address named *myPublicIP* in the *myResourceGroupLoadBalancer* resource group:

PowerShellCopy

$publicIP = New-AzureRmPublicIpAddress `

-ResourceGroupName myResourceGroupLoadBalancer `

-Location EastUS `

-AllocationMethod Static `

-Name myPublicIP

Create a load balancer

Create a frontend IP address with [New-AzureRmLoadBalancerFrontendIpConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermloadbalancerfrontendipconfig). The following example creates a frontend IP address named *myFrontEndPool*:

PowerShellCopy

$frontendIP = New-AzureRmLoadBalancerFrontendIpConfig `

-Name myFrontEndPool `

-PublicIpAddress $publicIP

Create a backend address pool with [New-AzureRmLoadBalancerBackendAddressPoolConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermloadbalancerbackendaddresspoolconfig). The following example creates a backend address pool named *myBackEndPool*:

PowerShellCopy

$backendPool = New-AzureRmLoadBalancerBackendAddressPoolConfig -Name myBackEndPool

Now, create the load balancer with [New-AzureRmLoadBalancer](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermloadbalancer). The following example creates a load balancer named *myLoadBalancer* using the *myPublicIP* address:

PowerShellCopy

$lb = New-AzureRmLoadBalancer `

-ResourceGroupName myResourceGroupLoadBalancer `

-Name myLoadBalancer `

-Location EastUS `

-FrontendIpConfiguration $frontendIP `

-BackendAddressPool $backendPool

Create a health probe

To allow the load balancer to monitor the status of your app, you use a health probe. The health probe dynamically adds or removes VMs from the load balancer rotation based on their response to health checks. By default, a VM is removed from the load balancer distribution after two consecutive failures at 15-second intervals. You create a health probe based on a protocol or a specific health check page for your app.

The following example creates a TCP probe. You can also create custom HTTP probes for more fine grained health checks. When using a custom HTTP probe, you must create the health check page, such as *healthcheck.aspx*. The probe must return an **HTTP 200 OK** response for the load balancer to keep the host in rotation.

To create a TCP health probe, you use [Add-AzureRmLoadBalancerProbeConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/add-azurermloadbalancerprobeconfig). The following example creates a health probe named *myHealthProbe* that monitors each VM:

PowerShellCopy

Add-AzureRmLoadBalancerProbeConfig `

-Name myHealthProbe `

-LoadBalancer $lb `

-Protocol tcp `

-Port 80 `

-IntervalInSeconds 15 `

-ProbeCount 2

Update the load balancer with [Set-AzureRmLoadBalancer](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/set-azurermloadbalancer):

PowerShellCopy

Set-AzureRmLoadBalancer -LoadBalancer $lb

Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the front-end IP configuration for the incoming traffic and the back-end IP pool to receive the traffic, along with the required source and destination port. To make sure only healthy VMs receive traffic, you also define the health probe to use.

Create a load balancer rule with [Add-AzureRmLoadBalancerRuleConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/add-azurermloadbalancerruleconfig). The following example creates a load balancer rule named *myLoadBalancerRule* and balances traffic on port *80*:

PowerShellCopy

$probe = Get-AzureRmLoadBalancerProbeConfig -LoadBalancer $lb -Name myHealthProbe

Add-AzureRmLoadBalancerRuleConfig `

-Name myLoadBalancerRule `

-LoadBalancer $lb `

-FrontendIpConfiguration $lb.FrontendIpConfigurations[0] `

-BackendAddressPool $lb.BackendAddressPools[0] `

-Protocol Tcp `

-FrontendPort 80 `

-BackendPort 80 `

-Probe $probe

Update the load balancer with [Set-AzureRmLoadBalancer](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/set-azurermloadbalancer):

PowerShellCopy

Set-AzureRmLoadBalancer -LoadBalancer $lb

Configure virtual network

Before you deploy some VMs and can test your balancer, create the supporting virtual network resources. For more information about virtual networks, see the [Manage Azure Virtual Networks](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-virtual-network)tutorial.

Create network resources

Create a virtual network with [New-AzureRmVirtualNetwork](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermvirtualnetwork). The following example creates a virtual network named *myVnet* with *mySubnet*:

PowerShellCopy

# Create subnet config

$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig `

-Name mySubnet `

-AddressPrefix 192.168.1.0/24

# Create the virtual network

$vnet = New-AzureRmVirtualNetwork `

-ResourceGroupName myResourceGroupLoadBalancer `

-Location EastUS `

-Name myVnet `

-AddressPrefix 192.168.0.0/16 `

-Subnet $subnetConfig

Create a network security group rule with [New-AzureRmNetworkSecurityRuleConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermnetworksecurityruleconfig), then create a network security group with [New-AzureRmNetworkSecurityGroup](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermnetworksecuritygroup). Add the network security group to the subnet with [Set-AzureRmVirtualNetworkSubnetConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/set-azurermvirtualnetworksubnetconfig) and then update the virtual network with [Set-AzureRmVirtualNetwork](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/set-azurermvirtualnetwork).

The following example creates a network security group rule named *myNetworkSecurityGroup*and applies it to *mySubnet*:

PowerShellCopy

# Create security rule config

$nsgRule = New-AzureRmNetworkSecurityRuleConfig `

-Name myNetworkSecurityGroupRule `

-Protocol Tcp `

-Direction Inbound `

-Priority 1001 `

-SourceAddressPrefix \* `

-SourcePortRange \* `

-DestinationAddressPrefix \* `

-DestinationPortRange 80 `

-Access Allow

# Create the network security group

$nsg = New-AzureRmNetworkSecurityGroup `

-ResourceGroupName myResourceGroupLoadBalancer `

-Location EastUS `

-Name myNetworkSecurityGroup `

-SecurityRules $nsgRule

# Apply the network security group to a subnet

Set-AzureRmVirtualNetworkSubnetConfig `

-VirtualNetwork $vnet `

-Name mySubnet `

-NetworkSecurityGroup $nsg `

-AddressPrefix 192.168.1.0/24

# Update the virtual network

Set-AzureRmVirtualNetwork -VirtualNetwork $vnet

Virtual NICs are created with [New-AzureRmNetworkInterface](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermnetworkinterface). The following example creates three virtual NICs. (One virtual NIC for each VM you create for your app in the following steps). You can create additional virtual NICs and VMs at any time and add them to the load balancer:

PowerShellCopy

for ($i=1; $i -le 3; $i++)

{

New-AzureRmNetworkInterface `

-ResourceGroupName myResourceGroupLoadBalancer `

-Name myNic$i `

-Location EastUS `

-Subnet $vnet.Subnets[0] `

-LoadBalancerBackendAddressPool $lb.BackendAddressPools[0]

}

Create virtual machines

To improve the high availability of your app, place your VMs in an availability set.

Create an availability set with [New-AzureRmAvailabilitySet](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/new-azurermavailabilityset). The following example creates an availability set named *myAvailabilitySet*:

PowerShellCopy

$availabilitySet = New-AzureRmAvailabilitySet `

-ResourceGroupName myResourceGroupLoadBalancer `

-Name myAvailabilitySet `

-Location EastUS `

-Managed `

-PlatformFaultDomainCount 3 `

-PlatformUpdateDomainCount 2

Set an administrator username and password for the VMs with [Get-Credential](https://msdn.microsoft.com/powershell/reference/5.1/microsoft.powershell.security/Get-Credential):

PowerShellCopy

$cred = Get-Credential

Now you can create the VMs with [New-AzureRmVM](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/new-azurermvm). The following example creates three VMs:

PowerShellCopy

for ($i=1; $i -le 3; $i++)

{

$vm = New-AzureRmVMConfig `

-VMName myVM$i `

-VMSize Standard\_D1 `

-AvailabilitySetId $availabilitySet.Id

$vm = Set-AzureRmVMOperatingSystem `

-VM $vm `

-Windows `

-ComputerName myVM$i `

-Credential $cred `

-ProvisionVMAgent `

-EnableAutoUpdate

$vm = Set-AzureRmVMSourceImage `

-VM $vm `

-PublisherName MicrosoftWindowsServer `

-Offer WindowsServer `

-Skus 2016-Datacenter `

-Version latest

$vm = Set-AzureRmVMOSDisk `

-VM $vm `

-Name myOsDisk$i `

-DiskSizeInGB 128 `

-CreateOption FromImage `

-Caching ReadWrite

$nic = Get-AzureRmNetworkInterface `

-ResourceGroupName myResourceGroupLoadBalancer `

-Name myNic$i

$vm = Add-AzureRmVMNetworkInterface -VM $vm -Id $nic.Id

New-AzureRmVM `

-ResourceGroupName myResourceGroupLoadBalancer `

-Location EastUS `

-VM $vm

}

It takes a few minutes to create and configure all three VMs.

Install IIS with Custom Script Extension

In a previous tutorial on [How to customize a Windows virtual machine](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-automate-vm-deployment), you learned how to automate VM customization with the Custom Script Extension for Windows. You can use the same approach to install and configure IIS on your VMs.

Use [Set-AzureRmVMExtension](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/set-azurermvmextension) to install the Custom Script Extension. The extension runs powershell Add-WindowsFeature Web-Server to install the IIS webserver and then updates the *Default.htm* page to show the hostname of the VM:

PowerShellCopy

for ($i=1; $i -le 3; $i++)

{

Set-AzureRmVMExtension `

-ResourceGroupName myResourceGroupLoadBalancer `

-ExtensionName IIS `

-VMName myVM$i `

-Publisher Microsoft.Compute `

-ExtensionType CustomScriptExtension `

-TypeHandlerVersion 1.4 `

-SettingString '{"commandToExecute":"powershell Add-WindowsFeature Web-Server; powershell Add-Content -Path \"C:\\inetpub\\wwwroot\\Default.htm\" -Value $($env:computername)"}' `

-Location EastUS

}

Test load balancer

Obtain the public IP address of your load balancer with [Get-AzureRmPublicIPAddress](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/get-azurermpublicipaddress). The following example obtains the IP address for *myPublicIP* created earlier:

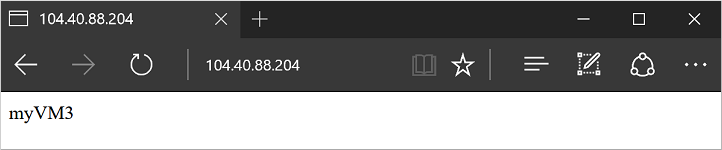
PowerShellCopy

Get-AzureRmPublicIPAddress `

-ResourceGroupName myResourceGroupLoadBalancer `

-Name myPublicIP | select IpAddress

You can then enter the public IP address in to a web browser. The website is displayed, including the hostname of the VM that the load balancer distributed traffic to as in the following example:



To see the load balancer distribute traffic across all three VMs running your app, you can force-refresh your web browser.

Add and remove VMs

You may need to perform maintenance on the VMs running your app, such as installing OS updates. To deal with increased traffic to your app, you may need to add additional VMs. This section shows you how to remove or add a VM from the load balancer.

Remove a VM from the load balancer

Get the network interface card with [Get-AzureRmNetworkInterface](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/get-azurermnetworkinterface), then set the *LoadBalancerBackendAddressPools* property of the virtual NIC to *$null*. Finally, update the virtual NIC.:

PowerShellCopy

$nic = Get-AzureRmNetworkInterface `

-ResourceGroupName myResourceGroupLoadBalancer `

-Name myNic2

$nic.Ipconfigurations[0].LoadBalancerBackendAddressPools=$null

Set-AzureRmNetworkInterface -NetworkInterface $nic

To see the load balancer distribute traffic across the remaining two VMs running your app you can force-refresh your web browser. You can now perform maintenance on the VM, such as installing OS updates or performing a VM reboot.

Add a VM to the load balancer

After performing VM maintenance, or if you need to expand capacity, set the *LoadBalancerBackendAddressPools* property of the virtual NIC to the *BackendAddressPool* from [Get-AzureRMLoadBalancer](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/get-azurermloadbalancer):

Get the load balancer:

PowerShellCopy

$lb = Get-AzureRMLoadBalancer `

-ResourceGroupName myResourceGroupLoadBalancer `

-Name myLoadBalancer

$nic.IpConfigurations[0].LoadBalancerBackendAddressPools=$lb.BackendAddressPools[0]

Set-AzureRmNetworkInterface -NetworkInterface $nic