Create a Virtual Machine Scale Set and deploy a highly available app on Windows

A virtual machine scale set allows you to deploy and manage a set of identical, auto-scaling virtual machines. You can scale the number of VMs in the scale set manually, or define rules to autoscale based on resource usage such as CPU, memory demand, or network traffic. In this tutorial, you deploy a virtual machine scale set in Azure. You learn how to:

* Use the Custom Script Extension to define an IIS site to scale
* Create a load balancer for your scale set
* Create a virtual machine scale set
* Increase or decrease the number of instances in a scale set
* Create autoscale rules

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).

Scale Set overview

A virtual machine scale set allows you to deploy and manage a set of identical, auto-scaling virtual machines. VMs in a scale set are distributed across logic fault and update domains in one or more *placement groups*. These are groups of similarly configured VMs, similar to [availability sets](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-availability-sets).

VMs are created as needed in a scale set. You define autoscale rules to control how and when VMs are added or removed from the scale set. These rules can trigger based on metrics such as CPU load, memory usage, or network traffic.

Scale sets support up to 1,000 VMs when you use an Azure platform image. For workloads with significant installation or VM customization requirements, you may wish to [Create a custom VM image](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-custom-images). You can create up to 300 VMs in a scale set when using a custom image.

Create an app to scale

Before you can create a scale set, 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 *myResourceGroupAutomate* in the *EastUS*location:

PowerShellCopy

New-AzureRmResourceGroup -ResourceGroupName myResourceGroupScaleSet -Location EastUS

In an earlier tutorial, you learned how to [Automate VM configuration](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-automate-vm-deployment) using the Custom Script Extension. Create a scale set configuration then apply a Custom Script Extension to install and configure IIS:

PowerShellCopy

# Create a config object

$vmssConfig = New-AzureRmVmssConfig `

-Location EastUS `

-SkuCapacity 2 `

-SkuName Standard\_DS2 `

-UpgradePolicyMode Automatic

# Define the script for your Custom Script Extension to run

$publicSettings = @{

"fileUris" = (,"https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/automate-iis.ps1");

"commandToExecute" = "powershell -ExecutionPolicy Unrestricted -File automate-iis.ps1"

}

# Use Custom Script Extension to install IIS and configure basic website

Add-AzureRmVmssExtension -VirtualMachineScaleSet $vmssConfig `

-Name "customScript" `

-Publisher "Microsoft.Compute" `

-Type "CustomScriptExtension" `

-TypeHandlerVersion 1.8 `

-Setting $publicSettings

Create scale load balancer

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. For more information, see the next tutorial on [How to load balance Windows virtual machines](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-load-balancer).

Create a load balancer that has a public IP address and distributes web traffic on port 80:

PowerShellCopy

# Create a public IP address

$publicIP = New-AzureRmPublicIpAddress `

-ResourceGroupName myResourceGroupScaleSet `

-Location EastUS `

-AllocationMethod Static `

-Name myPublicIP

# Create a frontend and backend IP pool

$frontendIP = New-AzureRmLoadBalancerFrontendIpConfig `

-Name myFrontEndPool `

-PublicIpAddress $publicIP

$backendPool = New-AzureRmLoadBalancerBackendAddressPoolConfig -Name myBackEndPool

# Create the load balancer

$lb = New-AzureRmLoadBalancer `

-ResourceGroupName myResourceGroupScaleSet `

-Name myLoadBalancer `

-Location EastUS `

-FrontendIpConfiguration $frontendIP `

-BackendAddressPool $backendPool

# Create a load balancer health probe on port 80

Add-AzureRmLoadBalancerProbeConfig -Name myHealthProbe `

-LoadBalancer $lb `

-Protocol tcp `

-Port 80 `

-IntervalInSeconds 15 `

-ProbeCount 2

# Create a load balancer rule to distribute traffic on port 80

Add-AzureRmLoadBalancerRuleConfig `

-Name myLoadBalancerRule `

-LoadBalancer $lb `

-FrontendIpConfiguration $lb.FrontendIpConfigurations[0] `

-BackendAddressPool $lb.BackendAddressPools[0] `

-Protocol Tcp `

-FrontendPort 80 `

-BackendPort 80

# Update the load balancer configuration

Set-AzureRmLoadBalancer -LoadBalancer $lb

Create a scale set

Now create a virtual machine scale set with [New-AzureRmVmss](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/new-azurermvm). The following example creates a scale set named *myScaleSet*:

PowerShellCopy

# Reference a virtual machine image from the gallery

Set-AzureRmVmssStorageProfile $vmssConfig `

-ImageReferencePublisher MicrosoftWindowsServer `

-ImageReferenceOffer WindowsServer `

-ImageReferenceSku 2016-Datacenter `

-ImageReferenceVersion latest

# Set up information for authenticating with the virtual machine

Set-AzureRmVmssOsProfile $vmssConfig `

-AdminUsername azureuser `

-AdminPassword P@ssword! `

-ComputerNamePrefix myVM

# Create the virtual network resources

$subnet = New-AzureRmVirtualNetworkSubnetConfig `

-Name "mySubnet" `

-AddressPrefix 10.0.0.0/24

$vnet = New-AzureRmVirtualNetwork `

-ResourceGroupName "myResourceGroupScaleSet" `

-Name "myVnet" `

-Location "EastUS" `

-AddressPrefix 10.0.0.0/16 `

-Subnet $subnet

$ipConfig = New-AzureRmVmssIpConfig `

-Name "myIPConfig" `

-LoadBalancerBackendAddressPoolsId $lb.BackendAddressPools[0].Id `

-SubnetId $vnet.Subnets[0].Id

# Attach the virtual network to the config object

Add-AzureRmVmssNetworkInterfaceConfiguration `

-VirtualMachineScaleSet $vmssConfig `

-Name "network-config" `

-Primary $true `

-IPConfiguration $ipConfig

# Create the scale set with the config object (this step might take a few minutes)

New-AzureRmVmss `

-ResourceGroupName myResourceGroupScaleSet `

-Name myScaleSet `

-VirtualMachineScaleSet $vmssConfig

It takes a few minutes to create and configure all the scale set resources and VMs.

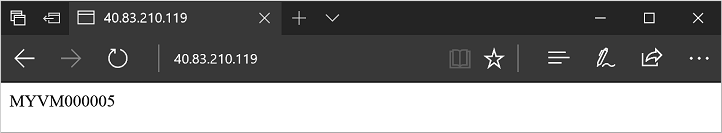
Test your app

To see your IIS website in action, 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 as part of the scale set:

PowerShellCopy

Get-AzureRmPublicIPAddress -ResourceGroupName myResourceGroupScaleSet -Name myPublicIP | select IpAddress

Enter the public IP address in to a web browser. The app is displayed, including the hostname of the VM that the load balancer distributed traffic to:



To see the scale set in action, you can force-refresh your web browser to see the load balancer distribute traffic across all the VMs running your app.

Management tasks

Throughout the lifecycle of the scale set, you may need to run one or more management tasks. Additionally, you may want to create scripts that automate various lifecycle-tasks. Azure PowerShell provides a quick way to do those tasks. Here are a few common tasks.

View VMs in a scale set

To view a list of VMs running in your scale set, use [Get-AzureRmVmssVM](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/get-azurermvmssvm) as follows:

PowerShellCopy

# Get current scale set

$scaleset = Get-AzureRmVmss `

-ResourceGroupName myResourceGroupScaleSet `

-VMScaleSetName myScaleSet

# Loop through the instanaces in your scale set

for ($i=1; $i -le ($scaleset.Sku.Capacity - 1); $i++) {

Get-AzureRmVmssVM -ResourceGroupName myResourceGroupScaleSet `

-VMScaleSetName myScaleSet `

-InstanceId $i

}

Increase or decrease VM instances

To see the number of instances you currently have in a scale set, use [Get-AzureRmVmss](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/get-azurermvmss) and query on *sku.capacity*:

PowerShellCopy

Get-AzureRmVmss -ResourceGroupName myResourceGroupScaleSet `

-VMScaleSetName myScaleSet | `

Select -ExpandProperty Sku

You can then manually increase or decrease the number of virtual machines in the scale set with [Update-AzureRmVmss](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/update-azurermvmss). The following example sets the number of VMs in your scale set to *5*:

PowerShellCopy

# Get current scale set

$scaleset = Get-AzureRmVmss `

-ResourceGroupName myResourceGroupScaleSet `

-VMScaleSetName myScaleSet

# Set and update the capacity of your scale set

$scaleset.sku.capacity = 5

Update-AzureRmVmss -ResourceGroupName myResourceGroupScaleSet `

-Name myScaleSet `

-VirtualMachineScaleSet $scaleset

If takes a few minutes to update the specified number of instances in your scale set.

Configure autoscale rules

Rather than manually scaling the number of instances in your scale set, you define autoscale rules. These rules monitor the instances in your scale set and respond accordingly based on metrics and thresholds you define. The following example scales out the number of instances by one when the average CPU load is greater than 60% over a 5 minute period. If the average CPU load then drops below 30% over a 5 minute period, the instances are scaled in by one instance:

PowerShellCopy

# Define your scale set information

$mySubscriptionId = (Get-AzureRmSubscription).Id

$myResourceGroup = "myResourceGroupScaleSet"

$myScaleSet = "myScaleSet"

$myLocation = "East US"

# Create a scale up rule to increase the number instances after 60% average CPU usage exceeded for a 5 minute period

$myRuleScaleUp = New-AzureRmAutoscaleRule `

-MetricName "Percentage CPU" `

-MetricResourceId /subscriptions/$mySubscriptionId/resourceGroups/$myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/$myScaleSet `

-Operator GreaterThan `

-MetricStatistic Average `

-Threshold 60 `

-TimeGrain 00:01:00 `

-TimeWindow 00:05:00 `

-ScaleActionCooldown 00:05:00 `

-ScaleActionDirection Increase `

-ScaleActionValue 1

# Create a scale down rule to decrease the number of instances after 30% average CPU usage over a 5 minute period

$myRuleScaleDown = New-AzureRmAutoscaleRule `

-MetricName "Percentage CPU" `

-MetricResourceId /subscriptions/$mySubscriptionId/resourceGroups/$myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/$myScaleSet `

-Operator LessThan `

-MetricStatistic Average `

-Threshold 30 `

-TimeGrain 00:01:00 `

-TimeWindow 00:05:00 `

-ScaleActionCooldown 00:05:00 `

-ScaleActionDirection Decrease `

-ScaleActionValue 1

# Create a scale profile with your scale up and scale down rules

$myScaleProfile = New-AzureRmAutoscaleProfile `

-DefaultCapacity 2 `

-MaximumCapacity 10 `

-MinimumCapacity 2 `

-Rules $myRuleScaleUp,$myRuleScaleDown `

-Name "autoprofile"

# Apply the autoscale rules

Add-AzureRmAutoscaleSetting `

-Location $myLocation `

-Name "autosetting" `

-ResourceGroup $myResourceGroup `

-TargetResourceId /subscriptions/$mySubscriptionId/resourceGroups/$myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/$myScaleSet `

-AutoscaleProfiles $myScaleProfile

For more design information on the use of autoscale,