**Windows Virtual Machine**

**Create and deploy highly available virtual machines with Azure PowerShell**

Increase the availability and reliability of your Virtual Machine solutions on Azure using a capability called Availability Sets. Availability sets ensure that the VMs you deploy on Azure are distributed across multiple isolated hardware nodes in a cluster. Doing this ensures that if a hardware or software failure within Azure happens, only a subset of your VMs are impacted and that your overall solution remains available and operational.

You can create an availability set using [New-AzureRmAvailabilitySet](https://docs.microsoft.com/en-us/powershell/module/azurerm.compute/new-azurermavailabilityset). In this example, set both the number of update and fault domains at 2 for the availability set named myAvailabilitySet in the myResourceGroupAvailability resource group.

New-AzureRmAvailabilitySet `

-Location "EastUS" `

-Name "myAvailabilitySet" `

-ResourceGroupName "myResourceGroupAvailability" `

-Sku aligned `

-PlatformFaultDomainCount 2 `

-PlatformUpdateDomainCount 2

**Create VMs inside an availability set**

VMs must be created within the availability set to make sure they are correctly distributed across the hardware. You can't add an existing VM to an availability set after it is created.

The hardware in a location is divided in to multiple update domains and fault domains. An **update domain** is a group of VMs and underlying physical hardware that can be rebooted at the same time. VMs in the same **fault domain** share common storage as well as a common power source and network switch.

for ($i=1; $i -le 2; $i++)

{

New-AzureRmVm `

-ResourceGroupName "myResourceGroupAvailability" `

-Name "myVM$i" `

-Location "East US" `

-VirtualNetworkName "myVnet" `

-SubnetName "mySubnet" `

-SecurityGroupName "myNetworkSecurityGroup" `

-PublicIpAddressName "myPublicIpAddress$i" `

-AvailabilitySetName "myAvailabilitySet" `

-Credential $cred

}

# Create a virtual machine scale set and deploy a highly available app on Windows with Azure PowerShell

A virtual machine scale set allows you to deploy and manage a set of identical, autoscaling virtual machines. VMs in a scale set are distributed across logic fault and update domains in one or more placement groups. Placement groups 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.

## **Create a scale set**

Use the Custom Script Extension to install a basic IIS web server. Apply the Custom Script Extension that installs IIS as follows:

## **Allow traffic to application**

To allow access to the basic web application, create a network security group with [New-AzureRmNetworkSecurityRuleConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermnetworksecurityruleconfig) and [New-AzureRmNetworkSecurityGroup](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermnetworksecuritygroup).

## **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*:

# Get current scale set

$scaleset = Get-AzureRmVmss `

-ResourceGroupName "myResourceGroupScaleSet" `

-VMScaleSetName "myScaleSet"

# Set and update the capacity of your scale set

$scaleset.sku.capacity = 3

Update-AzureRmVmss -ResourceGroupName "myResourceGroupScaleSet" `

-Name "myScaleSet" `

-VirtualMachineScaleSet $scaleset

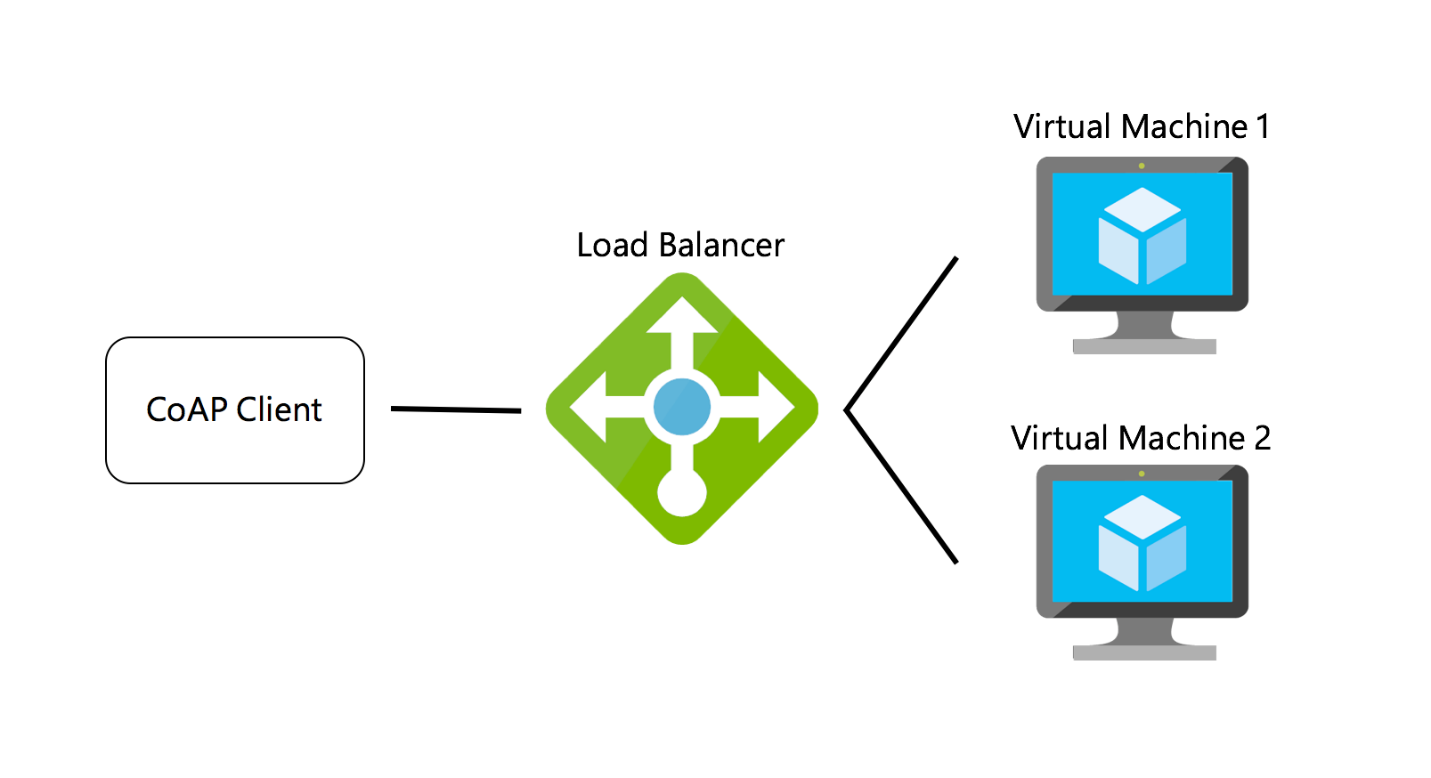
Finally configure auto scale rule to scale the virtual machine.

# Load balance Windows virtual machines in Azure to create a highly available application with Azure PowerShell

Load balancing provides a higher level of availability by spreading incoming requests across multiple virtual machines.

# 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 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).

# Create a load balancer

Create a frontend IP pool with [New-AzureRmLoadBalancerFrontendIpConfig](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermloadbalancerfrontendipconfig).

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

Now, create the load balancer with [New-AzureRmLoadBalancer](https://docs.microsoft.com/en-us/powershell/module/azurerm.network/new-azurermloadbalancer).

# 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.

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.