

Contents

[Virtual Machines Scale Sets Documentation](#)

[Overview](#)

[What are virtual machine scale sets?](#)

[Quickstarts](#)

[Create in the Azure portal](#)

[Create with the Azure CLI](#)

[Create with Azure PowerShell](#)

[Create with a template](#)

[Linux scale set](#)

[Windows scale set](#)

[Tutorials](#)

[1 - Create / manage scale set](#)

[Azure CLI](#)

[Azure PowerShell](#)

[2 - Use data disks](#)

[Azure CLI](#)

[Azure PowerShell](#)

[3 - Use a custom VM image](#)

[Azure CLI](#)

[Azure PowerShell](#)

[4 - Deploy apps to a scale set](#)

[Azure CLI](#)

[Azure PowerShell](#)

[Template](#)

[5 - Autoscale a scale set](#)

[Azure CLI](#)

[Azure PowerShell](#)

[Template](#)

[Samples](#)

[Azure CLI](#)

[PowerShell](#)

[Concepts](#)

[Azure Resource Manager](#)

[Regions](#)

[Availability and performance](#)

[Availability](#)

[Co-location](#)

[Network performance](#)

[Manage fault domains in scale sets](#)

[VM types and sizes](#)

[General purpose](#)

[B-series burstable](#)

[Compute optimized](#)

[Memory optimized](#)

[Constrained vCPUs](#)

[Storage optimized](#)

[GPU optimized](#)

[Setup GPU drivers](#)

[High performance compute](#)

[Azure compute units \(ACU\)](#)

[Benchmark scores](#)

[Endorsed Linux distros](#)

[Maintenance and updates](#)

[Orchestration modes](#)

[Disk storage](#)

[Managed Disks](#)

[Select a disk type](#)

[Premium storage performance](#)

[Ephemeral OS disks](#)

[Scalability targets for disks](#)

[Backup and disaster recovery for disks](#)

- [Networking](#)
- [Infrastructure automation](#)
- [Security](#)
 - [Security and policy](#)
 - [Azure Disk Encryption](#)
 - [Built-in security controls](#)
- [Monitoring](#)
- [Deployment considerations](#)
- [vCPU quotas](#)
- [Scale Set FAQ](#)
- [How To](#)
 - [Plan and design](#)
 - [Design considerations](#)
 - [Understand instance IDs](#)
 - [Create a template](#)
 - [Learn about scale set templates](#)
 - [Use an existing virtual network](#)
 - [Use a custom image](#)
 - [Use guest-based autoscaling with a Linux scale set template](#)
 - [Deploy](#)
 - [Create with Visual Studio](#)
 - [Use Availability Zones](#)
 - [Autoscale a scale set](#)
 - [Use the Azure portal](#)
 - [Advanced autoscale](#)
 - [Troubleshoot autoscale](#)
 - [Applications on scale sets](#)
 - [Extensions on scale sets](#)
 - [Extension sequencing on scale sets](#)
 - [Application Health extension](#)
 - [Use data disks with scale sets](#)
 - [Encrypt disks in scale sets](#)

- [Use PowerShell](#)
- [Use the Azure CLI](#)
- [Use Azure Resource Manager templates](#)
- [Extension sequencing](#)
- [Key vault for Azure Disk Encryption](#)
- [Work with large scale sets](#)
- [Convert a scale set template to use managed disk](#)
- [Use spot instance](#)
- [Manage](#)
 - [Common management tasks](#)
 - [Use the Azure CLI](#)
 - [Use Azure PowerShell](#)
 - [Modify a scale set](#)
 - [Networking for scale sets](#)
 - [Automatic OS upgrades](#)
 - [Instance protection](#)
 - [Scale-In Policy](#)
 - [Terminate notification on delete](#)
 - [Shared image galleries](#)
 - [Overview](#)
 - [CLI](#)
 - [PowerShell](#)
 - [Share images across tenants](#)
 - [Troubleshoot shared images](#)
 - [Create a proximity placement group](#)
 - [Vertical scaling in a scale set](#)
 - [Using DSC and scale sets](#)
 - [Convert a template to managed disks](#)
 - [Planned maintenance](#)
- [Use Ansible](#)
 - [Create and manage scale sets](#)
 - [Deploy apps to scale sets](#)

[Autoscale a scale set](#)

[Reference](#)

[Azure PowerShell](#)

[Azure CLI](#)

[REST](#)

[Azure templates](#)

[Resources](#)

[Azure Roadmap](#)

[Pricing](#)

[Linux](#)

[Windows](#)

[Pricing calculator](#)

[Stack Overflow](#)

What are virtual machine scale sets?

1/19/2020 • 4 minutes to read • [Edit Online](#)

Azure virtual machine scale sets let you create and manage a group of identical, load balanced VMs. The number of VM instances can automatically increase or decrease in response to demand or a defined schedule. Scale sets provide high availability to your applications, and allow you to centrally manage, configure, and update a large number of VMs. With virtual machine scale sets, you can build large-scale services for areas such as compute, big data, and container workloads.

Why use virtual machine scale sets?

To provide redundancy and improved performance, applications are typically distributed across multiple instances. Customers may access your application through a load balancer that distributes requests to one of the application instances. If you need to perform maintenance or update an application instance, your customers must be distributed to another available application instance. To keep up with additional customer demand, you may need to increase the number of application instances that run your application.

Azure virtual machine scale sets provide the management capabilities for applications that run across many VMs, [automatic scaling of resources](#), and load balancing of traffic. Scale sets provide the following key benefits:

- **Easy to create and manage multiple VMs**

- When you have many VMs that run your application, it's important to maintain a consistent configuration across your environment. For reliable performance of your application, the VM size, disk configuration, and application installs should match across all VMs.
- With scale sets, all VM instances are created from the same base OS image and configuration. This approach lets you easily manage hundreds of VMs without additional configuration tasks or network management.
- Scale sets support the use of the [Azure load balancer](#) for basic layer-4 traffic distribution, and [Azure Application Gateway](#) for more advanced layer-7 traffic distribution and SSL termination.

- **Provides high availability and application resiliency**

- Scale sets are used to run multiple instances of your application. If one of these VM instances has a problem, customers continue to access your application through one of the other VM instances with minimal interruption.
- For additional availability, you can use [Availability Zones](#) to automatically distribute VM instances in a scale set within a single datacenter or across multiple datacenters.

- **Allows your application to automatically scale as resource demand changes**

- Customer demand for your application may change throughout the day or week. To match customer demand, scale sets can automatically increase the number of VM instances as application demand increases, then reduce the number of VM instances as demand decreases.
- Autoscale also minimizes the number of unnecessary VM instances that run your application when demand is low, while customers continue to receive an acceptable level of performance as demand grows and additional VM instances are automatically added. This ability helps reduce costs and efficiently create Azure resources as required.

- **Works at large-scale**

- Scale sets support up to 1,000 VM instances. If you create and upload your own custom VM images, the limit is 600 VM instances.

- For the best performance with production workloads, use [Azure Managed Disks](#).

Differences between virtual machines and scale sets

Scale sets are built from virtual machines. With scale sets, the management and automation layers are provided to run and scale your applications. You could instead manually create and manage individual VMs, or integrate existing tools to build a similar level of automation. The following table outlines the benefits of scale sets compared to manually managing multiple VM instances.

SCENARIO	MANUAL GROUP OF VMs	VIRTUAL MACHINE SCALE SET
Add additional VM instances	Manual process to create, configure, and ensure compliance	Automatically create from central configuration
Traffic balancing and distribution	Manual process to create and configure Azure load balancer or Application Gateway	Can automatically create and integrate with Azure load balancer or Application Gateway
High availability and redundancy	Manually create Availability Set or distribute and track VMs across Availability Zones	Automatic distribution of VM instances across Availability Zones or Availability Sets
Scaling of VMs	Manual monitoring and Azure Automation	Autoscale based on host metrics, in-guest metrics, Application Insights, or schedule

There is no additional cost to scale sets. You only pay for the underlying compute resources such as the VM instances, load balancer, or Managed Disk storage. The management and automation features, such as autoscale and redundancy, incur no additional charges over the use of VMs.

How to monitor your scale sets

Use [Azure Monitor for VMs](#), which has a simple onboarding process and will automate the collection of important CPU, memory, disk, and network performance counters from the VMs in your scale set. It also includes additional monitoring capabilities and pre-defined visualizations that help you focus on the availability and performance of your scale sets.

Enable monitoring for your [virtual machine scale set application](#) with Application Insights to collect detailed information about your application including page views, application requests, and exceptions. Further verify the availability of your application by configuring an [availability test](#) to simulate user traffic.

Next steps

To get started, create your first virtual machine scale set in the Azure portal.

[Create a scale set in the Azure portal](#)

Quickstart: Create a virtual machine scale set in the Azure portal

1/23/2020 • 3 minutes to read • [Edit Online](#)

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 like CPU, memory demand, or network traffic. An Azure load balancer then distributes traffic to the VM instances in the scale set. In this quickstart, you create a virtual machine scale set in the Azure portal.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Log in to Azure

Log in to the Azure portal at <https://portal.azure.com>.

Create a load balancer

Azure [load balancer](#) distributes incoming traffic among healthy virtual machine instances.

First, create a public Standard Load Balancer by using the portal. The name and public IP address you create are automatically configured as the load balancer's front end.

1. In the search box, type **load balancer**. Under **Marketplace** in the search results, pick **Load balancer**.
2. In the **Basics** tab of the **Create load balancer** page, enter or select the following information:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select Create new and type <i>myVMSSResourceGroup</i> in the text box.
Name	<i>myLoadBalancer</i>
Region	Select East US .
Type	Select Public .
SKU	Select Standard .
Public IP address	Select Create new .
Public IP address name	<i>MyPip</i>
Assignment	Static

3. When you are done, select **Review + create**
4. After it passes validation, select **Create**.

Create load balancer

[Basics](#) [Tags](#) [Review + create](#)

Azure load balancer is a layer 4 load balancer that distributes incoming traffic among healthy virtual machine instances. Load balancers uses a hash-based distribution algorithm. By default, it uses a 5-tuple (source IP, source port, destination IP, destination port, protocol type) hash to map traffic to available servers. Load balancers can either be internet-facing where it is accessible via public IP addresses, or internal where it is only accessible from a virtual network. Azure load balancers also support Network Address Translation (NAT) to route traffic between public and private IP addresses. [Learn more.](#)

Project details

Subscription *	<input type="text" value="Pay-As-You-Go"/>	▼
Resource group *	<input type="text" value="(New) myVMSSResourceGroup"/>	▼
	Create new	

Instance details

Name *	<input type="text" value="myLoadBalancer"/>	✓
Region *	<input type="text" value="(US) East US"/>	▼
Type * ⓘ	<input type="radio"/> Internal <input checked="" type="radio"/> Public	
SKU * ⓘ	<input type="radio"/> Basic <input checked="" type="radio"/> Standard	

Public IP address

Public IP address * ⓘ	<input checked="" type="radio"/> Create new <input type="radio"/> Use existing	
Public IP address name *	<input type="text" value="myPip"/>	✓
Public IP address SKU	Standard	
Assignment *	<input type="radio"/> Dynamic <input checked="" type="radio"/> Static	
Availability zone *	<input type="text" value="Zone-redundant"/>	▼

[Review + create](#)

< Previous

Next : Tags >

[Download a template for automation](#)

Create virtual machine scale set

You can deploy a scale set with a Windows Server image or Linux image such as RHEL, CentOS, Ubuntu, or SLES.

1. Type **Scale set** in the search box. In the results, under **Marketplace**, select **Virtual machine scale sets**. The **Create a virtual machine scale set** page will open.
2. In the **Basics** tab, under **Project details**, make sure the correct subscription is selected and then choose to **Create new** resource group. Type *myVMSSResourceGroup* for the name and then select **OK**.
3. Type *myScaleSet* as the name for your scale set.

4. In **Region**, select a region that is close to you area.
5. Leave the default value of **ScaleSet VMs** for **Orchestrator**.
6. Select a marketplace image for **Image**. In this example, we have chosen *Ubuntu Server 18.04 LTS*.
7. Enter your desired username, and select which authentication type you prefer.
 - A **Password** must be at least 12 characters long and meet three out of the four following complexity requirements: one lower case character, one upper case character, one number, and one special character. For more information, see [username and password requirements](#).
 - If you select a Linux OS disk image, you can instead choose **SSH public key**. Only provide your public key, such as `~/.ssh/id_rsa.pub`. You can use the Azure Cloud Shell from the portal to [create and use SSH keys](#).

Home > Virtual machine scale sets > Create a virtual machine scale set

Create a virtual machine scale set

[Basics](#) [Instance](#) [Disks](#) [Networking](#) [Scaling](#) [Management](#) [Health](#) [Advanced](#) [Tags](#) [Review + create](#)

Azure virtual machine scale sets let you create and manage a group of load balanced VMs. The number of VM instances can automatically increase or decrease in response to demand or a defined schedule. Scale sets provide high availability to your applications, and allow you to centrally manage, configure, and update a large number of VMs.

[Learn more about virtual machine scale sets](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * [Create new](#)

Instance details

Virtual machine scale set name *

Region *

Availability zone [?](#)

Orchestrator * [?](#) **ScaleSet VMs**

Image * [?](#) [Browse all public and private images](#)

Size * [?](#) **Standard D2s v3**
2 vcpus, 8 GiB memory
[Change size](#)

Administrator account

Authentication type [?](#) Password SSH public key

Username *

SSH public key * [?](#)

[Review + create](#) [< Previous](#) [Next : Instance >](#)

8. Select **Next** to move to the other pages.
9. Leave the defaults for the **Instance** and **Disks** pages.

10. On the **Networking** page, under **Load balancing**, select **Yes** to put the scale set instances behind a load balancer.
11. In **Load balancing options**, select **Azure load balancer**.
12. In **Select a load balancer**, select *myLoadBalancer* that you created earlier.
13. For **Select a backend pool**, select **Create new**, type *myBackendPool*, then select **Create**.
14. When you are done, select **Review + create**.
15. After it passes validation, select **Create** to deploy the scale set.

Clean up resources

When no longer needed, delete the resource group, scale set, and all related resources. To do so, select the resource group for the scale set and then select **Delete**.

Next steps

In this quickstart, you created a basic scale set in the Azure portal. To learn more, continue to the tutorial for how to create and manage Azure virtual machine scale sets.

[Create and manage Azure virtual machine scale sets](#)

Quickstart: Create a virtual machine scale set with the Azure CLI

1/19/2020 • 4 minutes to read • [Edit Online](#)

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 like CPU, memory demand, or network traffic. An Azure load balancer then distributes traffic to the VM instances in the scale set. In this quickstart, you create a virtual machine scale set and deploy a sample application with the Azure CLI.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a scale set

Before you can create a scale set, create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

```
az group create --name myResourceGroup --location eastus
```

Now create a virtual machine scale set with [az vmss create](#). The following example creates a scale set named *myScaleSet* that is set to automatically update as changes are applied, and generates SSH keys if they do not exist in *~/ssh/id_rsa*. These SSH keys are used if you need to log in to the VM instances. To use an existing set of SSH keys, instead use the `--ssh-key-value` parameter and specify the location of your keys.

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--admin-username azureuser \
--generate-ssh-keys
```

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

Deploy sample application

To test your scale set, install a basic web application. The Azure Custom Script Extension is used to download and run a script that installs an application on the VM instances. This extension is useful for post deployment configuration, software installation, or any other configuration / management task. For more information, see the [Custom Script Extension overview](#).

Use the Custom Script Extension to install a basic NGINX web server. Apply the Custom Script Extension that installs NGINX with [az vmss extension set](#) as follows:

```
az vmss extension set \
--publisher Microsoft.Azure.Extensions \
--version 2.0 \
--name CustomScript \
--resource-group myResourceGroup \
--vmss-name myScaleSet \
--settings '{"fileUris": ["https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/automate_nginx.sh"], "commandToExecute": "./automate_nginx.sh"}'
```

Allow traffic to application

When the scale set was created, an Azure load balancer was automatically deployed. The load balancer distributes traffic to the VM instances in the scale set. To allow traffic to reach the sample web application, create a load balancer rule with [az network lb rule create](#). The following example creates a rule named *myLoadBalancerRuleWeb*:

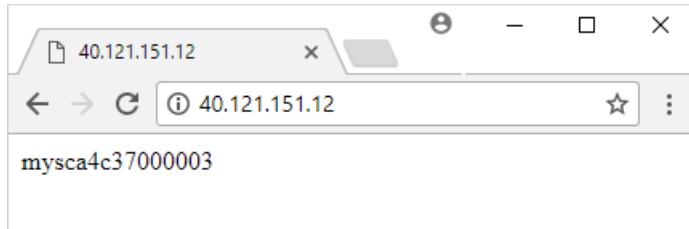
```
az network lb rule create \
--resource-group myResourceGroup \
--name myLoadBalancerRuleWeb \
--lb-name myScaleSetLB \
--backend-pool-name myScaleSetLBEPool \
--backend-port 80 \
--frontend-ip-name loadBalancerFrontEnd \
--frontend-port 80 \
--protocol tcp
```

Test your scale set

To see your scale set in action, access the sample web application in a web browser. Obtain the public IP address of your load balancer with [az network public-ip show](#). The following example obtains the IP address for `myScaleSetLBPublicIP` created as part of the scale set:

```
az network public-ip show \
--resource-group myResourceGroup \
--name myScaleSetLBPublicIP \
--query '[ipAddress]' \
--output tsv
```

Enter the public IP address of the load balancer in to a web browser. The load balancer distributes traffic to one of your VM instances, as shown in the following example:



Clean up resources

When no longer needed, you can use [az group delete](#) to remove the resource group, scale set, and all related resources as follows. The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --yes --no-wait
```

Next steps

In this quickstart, you created a basic scale set and used the Custom Script Extension to install a basic NGINX web server on the VM instances. To learn more, continue to the tutorial for how to create and manage Azure virtual machine scale sets.

[Create and manage Azure virtual machine scale sets](#)

Quickstart: Create a virtual machine scale set with Azure PowerShell

1/19/2020 • 4 minutes to read • [Edit Online](#)

A virtual machine scale set allows you to deploy and manage a set of identical, autoscaling virtual machines. You can scale the number of VMs in the scale set manually, or define rules to autoscale based on resource usage like CPU, memory demand, or network traffic. An Azure load balancer then distributes traffic to the VM instances in the scale set. In this quickstart, you create a virtual machine scale set and deploy a sample application with Azure PowerShell.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

Create a scale set

Before you can create a scale set, create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

```
New-AzResourceGroup -ResourceGroupName "myResourceGroup" -Location "EastUS"
```

Now create a virtual machine scale set with [New-AzVmss](#). The following example creates a scale set named

myScaleSet that uses the *Windows Server 2016 Datacenter* platform image. The Azure network resources for virtual network, public IP address, and load balancer are automatically created. When prompted, you can set your own administrative credentials for the VM instances in the scale set:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "EastUS" ` 
-VMSScaleSetName "myScaleSet" ` 
-VirtualNetworkName "myVnet" ` 
-SubnetName "mySubnet" ` 
-PublicIpAddressName "myPublicIPAddress" ` 
-LoadBalancerName "myLoadBalancer" ` 
-UpgradePolicyMode "Automatic"
```

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

Deploy sample application

To test your scale set, install a basic web application. The Azure Custom Script Extension is used to download and run a script that installs IIS on the VM instances. This extension is useful for post deployment configuration, software installation, or any other configuration / management task. For more information, see the [Custom Script Extension overview](#).

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

```
# 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"
}

# Get information about the scale set
$vmss = Get-AzVmss ` 
    -ResourceGroupName "myResourceGroup" ` 
    -VMSScaleSetName "myScaleSet"

# Use Custom Script Extension to install IIS and configure basic website
Add-AzVmssExtension -VirtualMachineScaleSet $vmss ` 
    -Name "customScript" ` 
    -Publisher "Microsoft.Compute" ` 
    -Type "CustomScriptExtension" ` 
    -TypeHandlerVersion 1.8 ` 
    -Setting $publicSettings

# Update the scale set and apply the Custom Script Extension to the VM instances
Update-AzVmss ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Name "myScaleSet" ` 
    -VirtualMachineScaleSet $vmss
```

Allow traffic to application

To allow access to the basic web application, create a network security group with [New-AzNetworkSecurityRuleConfig](#) and [New-AzNetworkSecurityGroup](#). For more information, see [Networking for Azure virtual machine scale sets](#).

```

# Get information about the scale set
$vmss = Get-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -VMScaleSetName "myScaleSet"

#Create a rule to allow traffic over port 80
$nsgFrontendRule = New-AzNetworkSecurityRuleConfig `

    -Name myFrontendNSGRule `

    -Protocol Tcp `

    -Direction Inbound `

    -Priority 200 `

    -SourceAddressPrefix * `

    -SourcePortRange * `

    -DestinationAddressPrefix * `

    -DestinationPortRange 80 `

    -Access Allow

#Create a network security group and associate it with the rule
$nsgFrontend = New-AzNetworkSecurityGroup `

    -ResourceGroupName "myResourceGroup" `

    -Location EastUS `

    -Name myFrontendNSG `

    -SecurityRules $nsgFrontendRule

$vnet = Get-AzVirtualNetwork `

    -ResourceGroupName "myResourceGroup" `

    -Name myVnet

$frontendSubnet = $vnet.Subnets[0]

$frontendSubnetConfig = Set-AzVirtualNetworkSubnetConfig `

    -VirtualNetwork $vnet `

    -Name mySubnet `

    -AddressPrefix $frontendSubnet.AddressPrefix `

    -NetworkSecurityGroup $nsgFrontend

Set-AzVirtualNetwork -VirtualNetwork $vnet

# Update the scale set and apply the Custom Script Extension to the VM instances
Update-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -Name "myScaleSet" `

    -VirtualMachineScaleSet $vmss

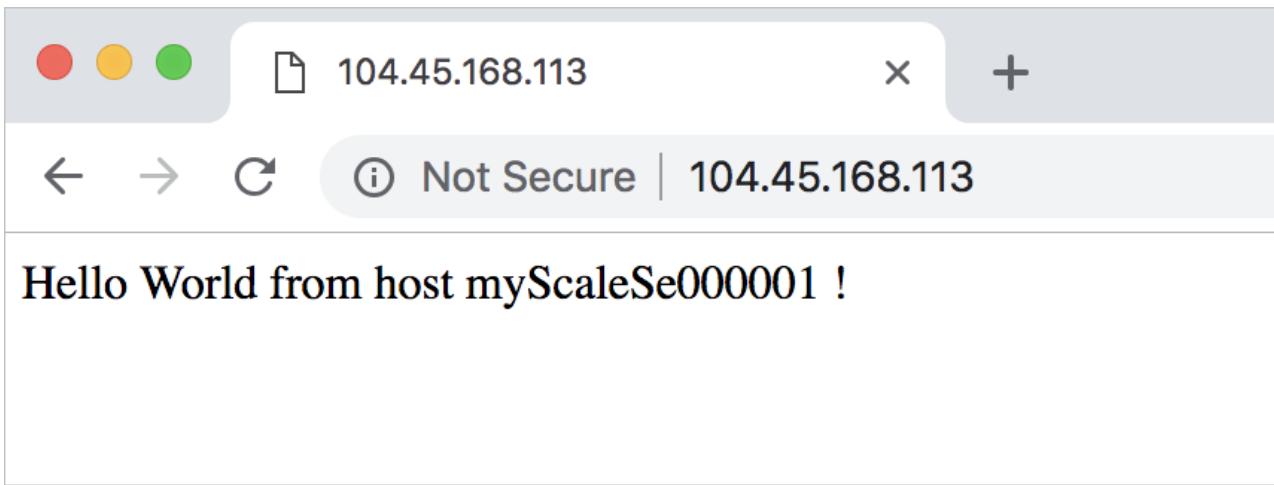
```

Test your scale set

To see your scale set in action, access the sample web application in a web browser. Get the public IP address of your load balancer with [Get-AzPublicIpAddress](#). The following example displays the IP address created in the *myResourceGroup* resource group:

```
Get-AzPublicIpAddress -ResourceGroupName "myResourceGroup" | Select IPAddress
```

Enter the public IP address of the load balancer in to a web browser. The load balancer distributes traffic to one of your VM instances, as shown in the following example:



Clean up resources

When no longer needed, you can use the `Remove-AzResourceGroup` to remove the resource group, scale set, and all related resources as follows. The `-Force` parameter confirms that you wish to delete the resources without an additional prompt to do so. The `-AsJob` parameter returns control to the prompt without waiting for the operation to complete.

```
Remove-AzResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this quickstart, you created a basic scale set and used the Custom Script Extension to install a basic IIS web server on the VM instances. To learn more, continue to the tutorial for how to create and manage Azure virtual machine scale sets.

[Create and manage Azure virtual machine scale sets](#)

Quickstart: Create a Linux virtual machine scale set with an Azure template

1/19/2020 • 6 minutes to read • [Edit Online](#)

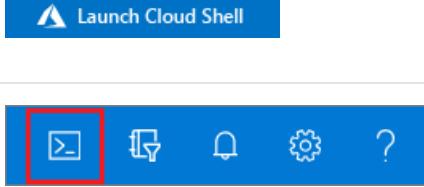
A virtual machine scale set allows you to deploy and manage a set of identical, auto-scaling virtual machines. You can scale the number of VMs in the scale set manually, or define rules to autoscale based on resource usage like CPU, memory demand, or network traffic. An Azure load balancer then distributes traffic to the VM instances in the scale set. In this quickstart, you create a virtual machine scale set and deploy a sample application with an Azure Resource Manager template.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Define a scale set in a template

Azure Resource Manager templates let you deploy groups of related resources. Templates are written in JavaScript Object Notation (JSON) and define the entire Azure infrastructure environment for your application. In a single template, you can create the virtual machine scale set, install applications, and configure autoscale rules. With the

use of variables and parameters, this template can be reused to update existing, or create additional, scale sets. You can deploy templates through the Azure portal, Azure CLI, or Azure PowerShell, or from continuous integration / continuous delivery (CI/CD) pipelines.

For more information on templates, see [Azure Resource Manager overview](#). For JSON syntax and properties, see [Microsoft.Compute/virtualMachineScaleSets](#) template reference.

To create a scale with a template, you define the appropriate resources. The core parts of the virtual machine scale set resource type are:

PROPERTY	DESCRIPTION OF PROPERTY	EXAMPLE TEMPLATE VALUE
type	Azure resource type to create	Microsoft.Compute/virtualMachineScale Sets
name	The scale set name	myScaleSet
location	The location to create the scale set	East US
sku.name	The VM size for each scale set instance	Standard_A1
sku.capacity	The number of VM instances to initially create	2
upgradePolicy.mode	VM instance upgrade mode when changes occur	Automatic
imageReference	The platform or custom image to use for the VM instances	Canonical Ubuntu Server 16.04-LTS
osProfile.computerNamePrefix	The name prefix for each VM instance	myvmss
osProfile.adminUsername	The username for each VM instance	azureuser
osProfile.adminPassword	The password for each VM instance	P@ssw0rd!

The following example shows the core scale set resource definition. To customize a scale set template, you can change the VM size or initial capacity, or use a different platform or a custom image.

```
{
  "type": "Microsoft.Compute/virtualMachineScaleSets",
  "name": "myScaleSet",
  "location": "East US",
  "apiVersion": "2017-12-01",
  "sku": {
    "name": "Standard_A1",
    "capacity": "2"
  },
  "properties": {
    "upgradePolicy": {
      "mode": "Automatic"
    },
    "virtualMachineProfile": {
      "storageProfile": {
        "osDisk": {
          "caching": "ReadWrite",
          "createOption": "FromImage"
        },
        "imageReference": {
          "publisher": "Canonical",
          "offer": "UbuntuServer",
          "sku": "16.04-LTS",
          "version": "latest"
        }
      },
      "osProfile": {
        "computerNamePrefix": "myvmss",
        "adminUsername": "azureuser",
        "adminPassword": "P@ssw0rd!"
      }
    }
  }
}
```

To keep the sample short, the virtual network interface card (NIC) configuration is not shown. Additional components, such as a load balancer, are also not shown. A complete scale set template is shown [at the end of this article](#).

Add a sample application

To test your scale set, install a basic web application. When you deploy a scale set, VM extensions can provide post-deployment configuration and automation tasks, such as installing an app. Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run-time. To apply an extension to your scale set, you add the *extensionProfile* section to the preceding resource example. The extension profile typically defines the following properties:

- Extension type
- Extension publisher
- Extension version
- Location of configuration or install scripts
- Commands to execute on the VM instances

The [Python HTTP server on Linux](#) template uses the Custom Script Extension to install [Bottle](#), a Python web framework, and a simple HTTP server.

Two scripts are defined in **fileUris** - *installserver.sh*, and *workserver.py*. These files are downloaded from GitHub, then *commandToExecute* runs `bash installserver.sh` to install and configure the app:

```
"extensionProfile": {
  "extensions": [
    {
      "name": "AppInstall",
      "properties": {
        "publisher": "Microsoft.Azure.Extensions",
        "type": "CustomScript",
        "typeHandlerVersion": "2.0",
        "autoUpgradeMinorVersion": true,
        "settings": {
          "fileUris": [
            "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vmss-bottle-autoscale/installserver.sh",
            "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vmss-bottle-autoscale/workserver.py"
          ],
          "commandToExecute": "bash installserver.sh"
        }
      }
    }
  ]
}
```

Deploy the template

You can deploy the [Python HTTP server on Linux](#) template with the following **Deploy to Azure** button. This button opens the Azure portal, loads the complete template, and prompts for a few parameters such as a scale set name, instance count, and admin credentials.



You can also use the Azure CLI to install the Python HTTP server on Linux with [az group deployment create](#) as follows:

```
# Create a resource group
az group create --name myResourceGroup --location EastUS

# Deploy template into resource group
az group deployment create \
--resource-group myResourceGroup \
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vmss-bottle-autoscale/azuredploy.json
```

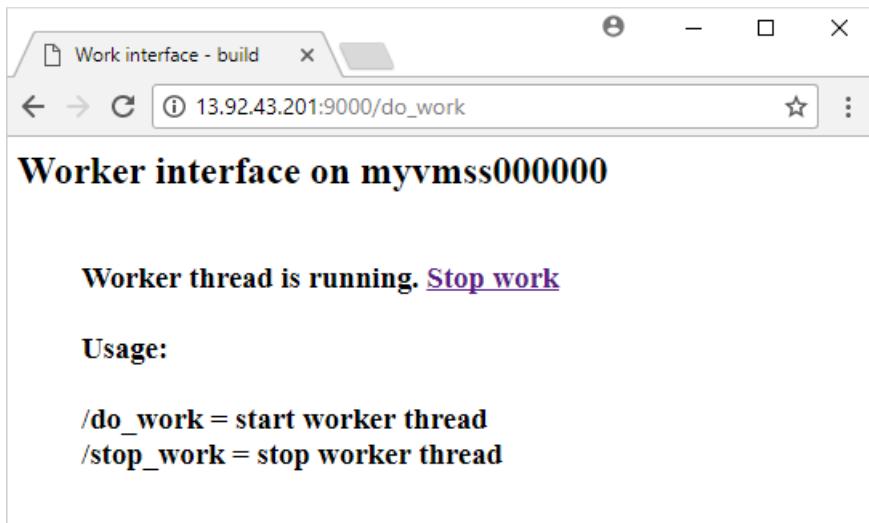
Answer the prompts to provide a scale set name, instance count, and admin credentials for the VM instances. It takes a few minutes for the scale set and supporting resources to be created.

Test your scale set

To see your scale set in action, access the sample web application in a web browser. Obtain the public IP address of the load balancer with [az network public-ip list](#) as follows:

```
az network public-ip list \
--resource-group myResourceGroup \
--query [*].ipAddress -o tsv
```

Enter the public IP address of the load balancer in to a web browser in the format http://publicIpAddress:9000/do_work. The load balancer distributes traffic to one of your VM instances, as shown in the following example:



Clean up resources

When no longer needed, you can use [az group delete](#) to remove the resource group, scale set, and all related resources as follows. The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --yes --no-wait
```

Next steps

In this quickstart, you created a Linux scale set with an Azure template and used the Custom Script Extension to install a basic Python web server on the VM instances. To learn more, continue to the tutorial for how to create and manage Azure virtual machine scale sets.

[Create and manage Azure virtual machine scale sets](#)

Quickstart: Create a Windows virtual machine scale set with an Azure template

1/19/2020 • 5 minutes to read • [Edit Online](#)

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 like CPU, memory demand, or network traffic. An Azure load balancer then distributes traffic to the VM instances in the scale set. In this quickstart, you create a virtual machine scale set and deploy a sample application with an Azure Resource Manager template.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

Define a scale set in a template

Azure Resource Manager templates let you deploy groups of related resources. Templates are written in JavaScript Object Notation (JSON) and define the entire Azure infrastructure environment for your application. In a single template, you can create the virtual machine scale set, install applications, and configure autoscale rules. With the use of variables and parameters, this template can be reused to update existing, or create additional, scale sets. You can deploy templates through the Azure portal, Azure CLI, or Azure PowerShell, or from continuous integration / continuous delivery (CI/CD) pipelines.

For more information on templates, see [Azure Resource Manager overview](#). For JSON syntax and properties, see [Microsoft.Compute/virtualMachineScaleSets](#) template reference.

A template defines the configuration for each resource type. A virtual machine scale set resource type is similar to an individual VM. The core parts of the virtual machine scale set resource type are:

PROPERTY	DESCRIPTION OF PROPERTY	EXAMPLE TEMPLATE VALUE
type	Azure resource type to create	Microsoft.Compute/virtualMachineScale Sets
name	The scale set name	myScaleSet
location	The location to create the scale set	East US
sku.name	The VM size for each scale set instance	Standard_A1
sku.capacity	The number of VM instances to initially create	2
upgradePolicy.mode	VM instance upgrade mode when changes occur	Automatic
imageReference	The platform or custom image to use for the VM instances	Microsoft Windows Server 2016 Datacenter
osProfile.computerNamePrefix	The name prefix for each VM instance	myvmss
osProfile.adminUsername	The username for each VM instance	azureuser
osProfile.adminPassword	The password for each VM instance	P@ssw0rd!

The following example shows the core scale set resource definition. To customize a scale set template, you can change the VM size or initial capacity, or use a different platform or a custom image.

```
{
  "type": "Microsoft.Compute/virtualMachineScaleSets",
  "name": "myScaleSet",
  "location": "East US",
  "apiVersion": "2017-12-01",
  "sku": {
    "name": "Standard_A1",
    "capacity": "2"
  },
  "properties": {
    "upgradePolicy": {
      "mode": "Automatic"
    },
    "virtualMachineProfile": {
      "storageProfile": {
        "osDisk": {
          "caching": "ReadWrite",
          "createOption": "FromImage"
        },
        "imageReference": {
          "publisher": "MicrosoftWindowsServer",
          "offer": "WindowsServer",
          "sku": "2016-Datacenter",
          "version": "latest"
        }
      },
      "osProfile": {
        "computerNamePrefix": "myvmss",
        "adminUsername": "azureuser",
        "adminPassword": "P@ssw0rd!"
      }
    }
  }
}
```

To keep the sample short, the virtual network interface card (NIC) configuration is not shown. Additional components, such as a load balancer, are also not shown. A complete scale set template is shown [at the end of this article](#).

Add a sample application

To test your scale set, install a basic web application. When you deploy a scale set, VM extensions can provide post-deployment configuration and automation tasks, such as installing an app. Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run-time. To apply an extension to your scale set, you add the *extensionProfile* section to the preceding resource example. The extension profile typically defines the following properties:

- Extension type
- Extension publisher
- Extension version
- Location of configuration or install scripts
- Commands to execute on the VM instances

The [ASP.NET application on Windows](#) sample template uses the PowerShell DSC extension to install an ASP.NET MVC app that runs in IIS.

An install script is downloaded from GitHub, as defined in *url*. The extension then runs *InstallIIS* from the *IISInstall.ps1* script, as defined in *function* and *Script*. The ASP.NET app itself is provided as a Web Deploy package, which is also downloaded from GitHub, as defined in *WebDeployPackagePath*:

```

"extensionProfile": {
  "extensions": [
    {
      "name": "Microsoft.PowerShell.DSC",
      "properties": {
        "publisher": "Microsoft.PowerShell",
        "type": "DSC",
        "typeHandlerVersion": "2.9",
        "autoUpgradeMinorVersion": true,
        "forceUpdateTag": "1.0",
        "settings": {
          "configuration": {
            "url": "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vmss-windows-webapp-dsc-autoscale/DSC/IISInstall.ps1.zip",
            "script": "IISInstall.ps1",
            "function": "InstallIIS"
          },
          "configurationArguments": {
            "nodeName": "localhost",
            "WebDeployPackagePath": "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vmss-windows-webapp-dsc-autoscale/WebDeploy/DefaultASPWebApp.v1.0.zip"
          }
        }
      }
    }
  ]
}

```

Deploy the template

You can deploy the [ASP.NET MVC application on Windows](#) template with the following **Deploy to Azure** button. This button opens the Azure portal, loads the complete template, and prompts for a few parameters such as a scale set name, instance count, and admin credentials.



You can also use Azure PowerShell to install the ASP.NET application on Windows with [New-AzResourceGroupDeployment](#) as follows:

```

# Create a resource group
New-AzResourceGroup -Name myResourceGroup -Location EastUS

# Deploy template into resource group
New-AzResourceGroupDeployment ` 
  -ResourceGroupName myResourceGroup ` 
  -TemplateURI https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vmss-windows-webapp-dsc-autoscale/azuredploy.json

# Update the scale set and apply the extension
Update-AzVmss ` 
  -ResourceGroupName myResourceGroup ` 
  -VmScaleSetName myVMS ` 
  -VirtualMachineScaleSet $vmssConfig

```

Answer the prompts to provide a scale set name and admin credentials for the VM instances. It can take 10-15 minutes for the scale set to be created and apply the extension to configure the app.

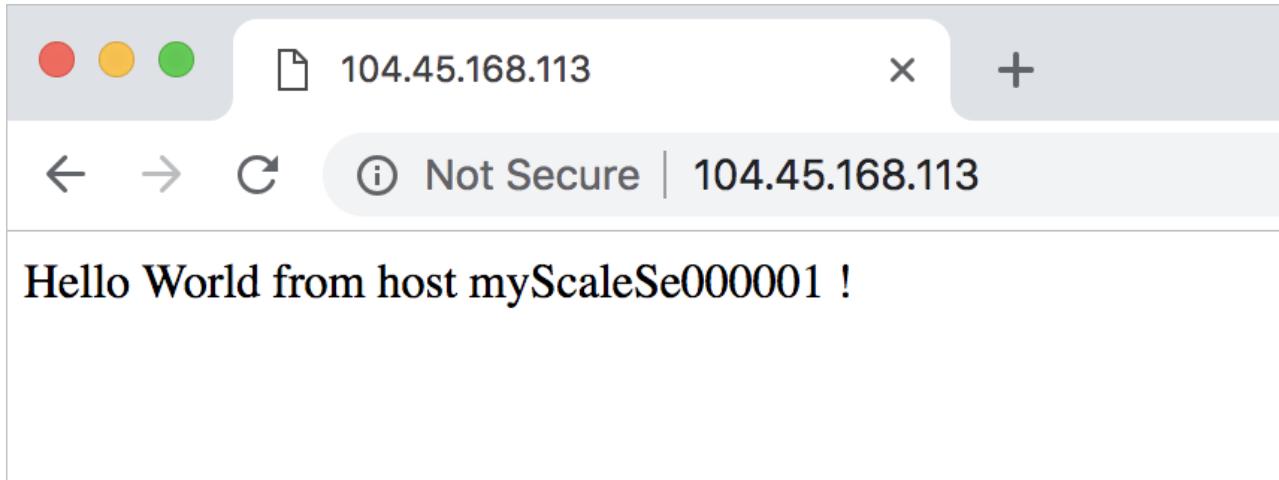
Test your scale set

To see your scale set in action, access the sample web application in a web browser. Obtain the public IP address of

your load balancer with [Get-AzPublicIpAddress](#) as follows:

```
Get-AzPublicIpAddress -ResourceGroupName myResourceGroup | Select IpAddress
```

Enter the public IP address of the load balancer in to a web browser in the format `http://publicIpAddress/MyApp`. The load balancer distributes traffic to one of your VM instances, as shown in the following example:



Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) to remove the resource group, scale set. The `-Force` parameter confirms that you wish to delete the resources without an additional prompt to do so. The `-AsJob` parameter returns control to the prompt without waiting for the operation to complete.

```
Remove-AzResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this quickstart, you created a Windows scale set with an Azure template and used the PowerShell DSC extension to install a basic ASP.NET app on the VM instances. To learn more, continue to the tutorial for how to create and manage Azure virtual machine scale sets.

[Create and manage Azure virtual machine scale sets](#)

Tutorial: Create and manage a virtual machine scale set with the Azure CLI

1/19/2020 • 10 minutes to read • [Edit Online](#)

A virtual machine scale set allows you to deploy and manage a set of identical, auto-scaling virtual machines. Throughout the lifecycle of a virtual machine scale set, you may need to run one or more management tasks. In this tutorial you learn how to:

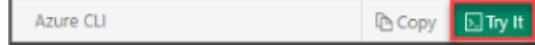
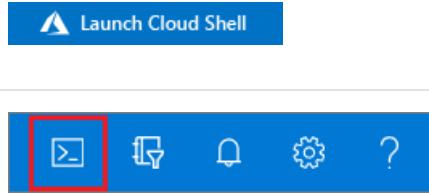
- Create and connect to a virtual machine scale set
- Select and use VM images
- View and use specific VM instance sizes
- Manually scale a scale set
- Perform common scale set management tasks

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a resource group

An Azure resource group is a logical container into which Azure resources are deployed and managed. A resource group must be created before a virtual machine scale set. Create a resource group with the [az group create](#) command. In this example, a resource group named *myResourceGroup* is created in the *eastus* region.

```
az group create --name myResourceGroup --location eastus
```

The resource group name is specified when you create or modify a scale set throughout this tutorial.

Create a scale set

You create a virtual machine scale set with the [az vmss create](#) command. The following example creates a scale set named *myScaleSet*, and generates SSH keys if they do not exist:

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--admin-username azureuser \
--generate-ssh-keys
```

It takes a few minutes to create and configure all the scale set resources and VM instances. To distribute traffic to the individual VM instances, a load balancer is also created.

View the VM instances in a scale set

To view a list of VM instances in a scale set, use [az vmss list-instances](#) as follows:

```
az vmss list-instances \
--resource-group myResourceGroup \
--name myScaleSet \
--output table
```

The following example output shows two VM instances in the scale set:

InstanceId	LatestModelApplied	Location	Name	ProvisioningState	ResourceGroup	VmId
1	True	eastus	myScaleSet_1	Succeeded	MYRESOURCEGROUP	c059be0c-37a2-497a-b111-41272641533c
3	True	eastus	myScaleSet_3	Succeeded	MYRESOURCEGROUP	ec19e7a7-a4cd-4b24-9670-438f4876c1f9

The first column in the output shows an *InstanceId*. To view additional information about a specific VM instance, add the `--instance-id` parameter to [az vmss get-instance-view](#). The following example views information about VM instance 1:

```
az vmss get-instance-view \
--resource-group myResourceGroup \
--name myScaleSet \
--instance-id 1
```

List connection information

A public IP address is assigned to the load balancer that routes traffic to the individual VM instances. By default,

Network Address Translation (NAT) rules are added to the Azure load balancer that forwards remote connection traffic to each VM on a given port. To connect to the VM instances in a scale set, you create a remote connection to an assigned public IP address and port number.

To list the address and ports to connect to VM instances in a scale set, use [az vmss list-instance-connection-info](#):

```
az vmss list-instance-connection-info \
--resource-group myResourceGroup \
--name myScaleSet
```

The following example output shows the instance name, public IP address of the load balancer, and port number that the NAT rules forward traffic to:

```
{
  "instance 1": "13.92.224.66:50001",
  "instance 3": "13.92.224.66:50003"
}
```

SSH to your first VM instance. Specify your public IP address and port number with the `-p` parameter, as shown from the preceding command:

```
ssh azureuser@13.92.224.66 -p 50001
```

Once logged in to the VM instance, you could perform some manual configuration changes as needed. For now, close the SSH session as normal:

```
exit
```

Understand VM instance images

When you created a scale set at the start of the tutorial, a `--image` of *Ubuntu LTS* was specified for the VM instances. The Azure marketplace includes many images that can be used to create VM instances. To see a list of the most commonly used images, use the [az vm image list](#) command.

```
az vm image list --output table
```

The following example output shows the most common VM images on Azure. The *UrnAlias* can be used to specify one of these common images when you create a scale set.

Offer UrnAlias	Publisher Version	Sku	Urn
Centos	OpenLogic	7.3	OpenLogic:CentOS:7.3:latest
Centos	latest		
Coreos	CoreOS	Stable	CoreOS:CoreOS:Stable:latest
Coreos	latest		
Debian	credativ	8	credativ:Debian:8:latest
Debian	latest		
openSUSE-Leap	SUSE	42.2	SUSE:openSUSE-Leap:42.2:latest
openSUSE-Leap	latest		
RHEL	RedHat	7.3	RedHat:RHEL:7.3:latest
RHEL	latest		
SLES	SUSE	12-SP2	SUSE:SLES:12-SP2:latest
SLES	latest		
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:latest
UbuntuLTS	latest		
WindowsServer	MicrosoftWindowsServer	2016-Datacenter	MicrosoftWindowsServer:WindowsServer:2016-
Datacenter:latest	Win2016Datacenter	latest	
WindowsServer	MicrosoftWindowsServer	2012-R2-Datacenter	MicrosoftWindowsServer:WindowsServer:2012-R2-
Datacenter:latest	Win2012R2Datacenter	latest	
WindowsServer	MicrosoftWindowsServer	2012-Datacenter	MicrosoftWindowsServer:WindowsServer:2012-
Datacenter:latest	Win2012Datacenter	latest	
WindowsServer	MicrosoftWindowsServer	2008-R2-SP1	MicrosoftWindowsServer:WindowsServer:2008-R2-
SP1:latest	Win2008R2SP1	latest	

To view a full list, add the `--all` argument. The image list can also be filtered by `--publisher` or `--offer`. In the following example, the list is filtered for all images with an offer that matches *CentOS*:

```
az vm image list --offer CentOS --all --output table
```

The following condensed output shows some of the *CentOS 7.3* images available:

Offer	Publisher	Sku	Urn	Version
Centos	OpenLogic	7.3	OpenLogic:CentOS:7.3:7.3.20161221	7.3.20161221
Centos	OpenLogic	7.3	OpenLogic:CentOS:7.3:7.3.20170421	7.3.20170421
Centos	OpenLogic	7.3	OpenLogic:CentOS:7.3:7.3.20170517	7.3.20170517
Centos	OpenLogic	7.3	OpenLogic:CentOS:7.3:7.3.20170612	7.3.20170612
Centos	OpenLogic	7.3	OpenLogic:CentOS:7.3:7.3.20170707	7.3.20170707
Centos	OpenLogic	7.3	OpenLogic:CentOS:7.3:7.3.20170925	7.3.20170925

To deploy a scale set that uses a specific image, use the value in the *Urn* column. When you specify the image, the image version number can be replaced with *latest*, which selects the latest version of the distribution. In the following example, the `--image` argument is used to specify the latest version of a *CentOS 7.3* image.

As it takes a few minutes to create and configure all the scale set resources and VM instances, you don't have to deploy the following scale set:

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSetCentOS \
--image OpenLogic:CentOS:7.3:latest \
--admin-user azureuser \
--generate-ssh-keys
```

Understand VM instance sizes

A VM instance size, or *SKU*, determines the amount of compute resources such as CPU, GPU, and memory that are made available to the VM instance. VM instances in a scale set need to be sized appropriately for the expected work load.

VM instance sizes

The following table categorizes common VM sizes into use cases.

Type	Common Sizes	Description
General purpose	Dsv3, Dv3, DSv2, Dv2, DS, D, Av2, A0-7	Balanced CPU-to-memory. Ideal for dev / test and small to medium applications and data solutions.
Compute optimized	Fs, F	High CPU-to-memory. Good for medium traffic applications, network appliances, and batch processes.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, DS, Dv2, D	High memory-to-core. Great for relational databases, medium to large caches, and in-memory analytics.
Storage optimized	Ls	High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases.
GPU	NV, NC	Specialized VMs targeted for heavy graphic rendering and video editing.
High performance	H, A8-11	Our most powerful CPU VMs with optional high-throughput network interfaces (RDMA).

Find available VM instance sizes

To see a list of VM instance sizes available in a particular region, use the [az vm list-sizes](#) command.

```
az vm list-sizes --location eastus --output table
```

The output is similar to the following condensed example, which shows the resources assigned to each VM size:

MaxDataDiskCount	MemoryInMb	Name	NumberOfCores	OsDiskSizeInMb
ResourceDiskSizeInMb				
7168	4	3584 Standard_DS1_v2	1	1047552
14336	8	7168 Standard_DS2_v2	2	1047552
[...]				
20480	1	768 Standard_A0	1	1047552
71680	2	1792 Standard_A1	1	1047552
[...]				
16384	4	2048 Standard_F1	1	1047552
32768	8	4096 Standard_F2	2	1047552
[...]				
38912	24	57344 Standard_NV6	6	1047552
696320	48	114688 Standard_NV12	12	1047552

Create a scale set with a specific VM instance size

When you created a scale set at the start of the tutorial, a default VM SKU of *Standard_D1_v2* was provided for the VM instances. You can specify a different VM instance size based on the output from [az vm list-sizes](#). The following example would create a scale set with the `--vm-sku` parameter to specify a VM instance size of *Standard_F1*. As it takes a few minutes to create and configure all the scale set resources and VM instances, you don't have to deploy the following scale set:

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSetF1Sku \
--image UbuntuLTS \
--vm-sku Standard_F1 \
--admin-user azureuser \
--generate-ssh-keys
```

Change the capacity of a scale set

When you created a scale set at the start of the tutorial, two VM instances were deployed by default. You can specify the `--instance-count` parameter with [az vmss create](#) to change the number of instances created with a scale set. To increase or decrease the number of VM instances in your existing scale set, you can manually change the capacity. The scale set creates or removes the required number of VM instances, then configures the load balancer to distribute traffic.

To manually increase or decrease the number of VM instances in the scale set, use [az vmss scale](#). The following example sets the number of VM instances in your scale set to 3:

```
az vmss scale \
--resource-group myResourceGroup \
--name myScaleSet \
--new-capacity 3
```

If takes a few minutes to update the capacity of your scale set. To see the number of instances you now have in the scale set, use [az vmss show](#) and query on *sku.capacity*:

```
az vmss show \
--resource-group myResourceGroup \
--name myScaleSet \
--query [sku.capacity] \
--output table
```

Common management tasks

You can now create a scale set, list connection information, and connect to VM instances. You learned how you could use a different OS image for your VM instances, select a different VM size, or manually scale the number of instances. As part of day to day management, you may need to stop, start, or restart the VM instances in your scale set.

Stop and deallocate VM instances in a scale set

To stop one or more VM instances in a scale set, use [az vmss stop](#). The `--instance-ids` parameter allows you to specify one or more VM instances to stop. If you do not specify an instance ID, all VM instances in the scale set are stopped. The following example stops instance 1:

```
az vmss stop --resource-group myResourceGroup --name myScaleSet --instance-ids 1
```

Stopped VM instances remain allocated and continue to incur compute charges. If you instead wish the VM instances to be deallocated and only incur storage charges, use [az vmss deallocate](#). The following example stops and deallocates instance 1:

```
az vmss deallocate --resource-group myResourceGroup --name myScaleSet --instance-ids 1
```

Start VM instances in a scale set

To start one or more VM instances in a scale set, use [az vmss start](#). The `--instance-ids` parameter allows you to specify one or more VM instances to start. If you do not specify an instance ID, all VM instances in the scale set are started. The following example starts instance 1:

```
az vmss start --resource-group myResourceGroup --name myScaleSet --instance-ids 1
```

Restart VM instances in a scale set

To restart one or more VM instances in a scale set, use [az vmss restart](#). The `--instance-ids` parameter allows you to specify one or more VM instances to restart. If you do not specify an instance ID, all VM instances in the scale set are restarted. The following example restarts instance 1:

```
az vmss restart --resource-group myResourceGroup --name myScaleSet --instance-ids 1
```

Clean up resources

When you delete a resource group, all resources contained within, such as the VM instances, virtual network, and disks, are also deleted. The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --no-wait --yes
```

Next steps

In this tutorial, you learned how to perform some basic scale set creation and management tasks with the Azure CLI:

- Create and connect to a virtual machine scale set
- Select and use VM images
- View and use specific VM sizes
- Manually scale a scale set
- Perform common scale set management tasks

Advance to the next tutorial to learn about scale set disks.

[Use data disks with scale sets](#)

Tutorial: Create and manage a virtual machine scale set with Azure PowerShell

1/19/2020 • 11 minutes to read • [Edit Online](#)

A virtual machine scale set allows you to deploy and manage a set of identical, auto-scaling virtual machines. Throughout the lifecycle of a virtual machine scale set, you may need to run one or more management tasks. In this tutorial you learn how to:

- Create and connect to a virtual machine scale set
- Select and use VM images
- View and use specific VM instance sizes
- Manually scale a scale set
- Perform common scale set management tasks

If you don't have an Azure subscription, create a [free account](#) before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by

selecting **Cmd+Shift+V** on macOS.

4. Select **Enter** to run the code.

Create a resource group

An Azure resource group is a logical container into which Azure resources are deployed and managed. A resource group must be created before a virtual machine scale set. Create a resource group with the [New-AzResourceGroup](#) command. In this example, a resource group named *myResourceGroup* is created in the *EastUS* region.

```
New-AzResourceGroup -ResourceGroupName "myResourceGroup" -Location "EastUS"
```

The resource group name is specified when you create or modify a scale set throughout this tutorial.

Create a scale set

First, set an administrator username and password for the VM instances with [Get-Credential](#):

```
$cred = Get-Credential
```

Now create a virtual machine scale set with [New-AzVmss](#). To distribute traffic to the individual VM instances, a load balancer is also created. The load balancer includes rules to distribute traffic on TCP port 80, as well as allow remote desktop traffic on TCP port 3389 and PowerShell remoting on TCP port 5985:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-VMSScaleSetName "myScaleSet" ` 
-Location "EastUS" ` 
-VirtualNetworkName "myVnet" ` 
-SubnetName "mySubnet" ` 
-PublicIpAddressName "myPublicIPAddress" ` 
-LoadBalancerName "myLoadBalancer" ` 
-Credential $cred
```

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

View the VM instances in a scale set

To view a list of VM instances in a scale set, use [Get-AzVmssVM](#) as follows:

```
Get-AzVmssVM -ResourceGroupName "myResourceGroup" -VMSScaleSetName "myScaleSet"
```

The following example output shows two VM instances in the scale set:

ResourceGroupName	Name	Location	Sku	InstanceID	ProvisioningState
MYRESOURCEGROUP	myScaleSet_0	eastus	Standard_DS1_v2	0	Succeeded
MYRESOURCEGROUP	myScaleSet_1	eastus	Standard_DS1_v2	1	Succeeded

To view additional information about a specific VM instance, add the `-InstanceId` parameter to [Get-AzVmssVM](#).

The following example views information about VM instance 1:

```
Get-AzVmssVM -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "1"
```

List connection information

A public IP address is assigned to the load balancer that routes traffic to the individual VM instances. By default, Network Address Translation (NAT) rules are added to the Azure load balancer that forwards remote connection traffic to each VM on a given port. To connect to the VM instances in a scale set, you create a remote connection to an assigned public IP address and port number.

To list the NAT ports to connect to VM instances in a scale set, first get the load balancer object with [Get-AzLoadBalancer](#). Then, view the inbound NAT rules with [Get-AzLoadBalancerInboundNatRuleConfig](#):

```
# Get the load balancer object
$lb = Get-AzLoadBalancer -ResourceGroupName "myResourceGroup" -Name "myLoadBalancer"

# View the list of inbound NAT rules
Get-AzLoadBalancerInboundNatRuleConfig -LoadBalancer $lb | Select-Object
Name,Protocol,FrontEndPort,BackEndPort
```

The following example output shows the instance name, public IP address of the load balancer, and port number that the NAT rules forward traffic to:

Name	Protocol	FrontendPort	BackendPort
myScaleSet3389.0	Tcp	50001	3389
myScaleSet5985.0	Tcp	51001	5985
myScaleSet3389.1	Tcp	50002	3389
myScaleSet5985.1	Tcp	51002	5985

The *Name* of the rule aligns with the name of the VM instance as shown in a previous [Get-AzVmssVM](#) command. For example, to connect to VM instance 0, you use *myScaleSet3389.0* and connect to port 50001. To connect to VM instance 1, use the value from *myScaleSet3389.1* and connect to port 50002. To use PowerShell remoting, you connect to the appropriate VM instance rule for TCP port 5985.

View the public IP address of the load balancer with [Get-AzPublicIpAddress](#):

```
Get-AzPublicIpAddress -ResourceGroupName "myResourceGroup" -Name "myPublicIPAddress" | Select IpAddress
```

Example output:

```
IpAddress
-----
52.168.121.216
```

Create a remote connection to your first VM instance. Specify your public IP address and port number of the required VM instance, as shown from the preceding commands. When prompted, enter the credentials used when you created the scale set (by default in the sample commands, *azureuser* and *P@ssw0rd!*). If you use the Azure Cloud Shell, perform this step from a local PowerShell prompt or Remote Desktop Client. The following example connects to VM instance 7:

```
mstsc /v 52.168.121.216:50001
```

Once logged in to the VM instance, you could perform some manual configuration changes as needed. For now,

close the remote connection.

Understand VM instance images

The Azure marketplace includes many images that can be used to create VM instances. To see a list of available publishers, use the [Get-AzVMImagePublisher](#) command.

```
Get-AzVMImagePublisher -Location "EastUS"
```

To view a list of images for a given publisher, use [Get-AzVMImageSku](#). The image list can also be filtered by `-PublisherName` or `-Offer`. In the following example, the list is filtered for all images with publisher name of *MicrosoftWindowsServer* and an offer that matches *WindowsServer*:

```
Get-AzVMImageSku -Location "EastUS" -PublisherName "MicrosoftWindowsServer" -Offer "WindowsServer"
```

The following example output shows all of the available Windows Server images:

Skus	Offer	PublisherName	Location
2008-R2-SP1	WindowsServer	MicrosoftWindowsServer	eastus
2008-R2-SP1-smalldisk	WindowsServer	MicrosoftWindowsServer	eastus
2012-Datacenter	WindowsServer	MicrosoftWindowsServer	eastus
2012-Datacenter-smalldisk	WindowsServer	MicrosoftWindowsServer	eastus
2012-R2-Datacenter	WindowsServer	MicrosoftWindowsServer	eastus
2012-R2-Datacenter-smalldisk	WindowsServer	MicrosoftWindowsServer	eastus
2016-Datacenter	WindowsServer	MicrosoftWindowsServer	eastus
2016-Datacenter-Server-Core	WindowsServer	MicrosoftWindowsServer	eastus
2016-Datacenter-Server-Core-smalldisk	WindowsServer	MicrosoftWindowsServer	eastus
2016-Datacenter-smalldisk	WindowsServer	MicrosoftWindowsServer	eastus
2016-Datacenter-with-Containers	WindowsServer	MicrosoftWindowsServer	eastus
2016-Datacenter-with-RDSH	WindowsServer	MicrosoftWindowsServer	eastus
2016-Nano-Server	WindowsServer	MicrosoftWindowsServer	eastus

When you created a scale set at the start of the tutorial, a default VM image of *Windows Server 2016 DataCenter* was provided for the VM instances. You can specify a different VM image based on the output from [Get-AzVMImageSku](#). The following example would create a scale set with the `-ImageName` parameter to specify a VM image of *MicrosoftWindowsServer:WindowsServer:2016-Datacenter-with-Containers:latest*. As it takes a few minutes to create and configure all the scale set resources and VM instances, you don't have to deploy the following scale set:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup2" ` 
-Location "EastUS" ` 
-VMSScaleSetName "myScaleSet2" ` 
-VirtualNetworkName "myVnet2" ` 
-SubnetName "mySubnet2" ` 
-PublicIpAddressName "myPublicIPAddress2" ` 
-LoadBalancerName "myLoadBalancer2" ` 
-UpgradePolicyMode "Automatic" ` 
-ImageName "MicrosoftWindowsServer:WindowsServer:2016-Datacenter-with-Containers:latest" ` 
-Credential $cred
```

Understand VM instance sizes

A VM instance size, or *SKU*, determines the amount of compute resources such as CPU, GPU, and memory that are made available to the VM instance. VM instances in a scale set need to be sized appropriately for the expected

work load.

VM instance sizes

The following table categorizes common VM sizes into use cases.

Type	Common Sizes	Description
General purpose	Dsv3, Dv3, DSv2, Dv2, DS, D, Av2, A0-7	Balanced CPU-to-memory. Ideal for dev / test and small to medium applications and data solutions.
Compute optimized	Fs, F	High CPU-to-memory. Good for medium traffic applications, network appliances, and batch processes.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, DS, Dv2, D	High memory-to-core. Great for relational databases, medium to large caches, and in-memory analytics.
Storage optimized	Ls	High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases.
GPU	NV, NC	Specialized VMs targeted for heavy graphic rendering and video editing.
High performance	H, A8-11	Our most powerful CPU VMs with optional high-throughput network interfaces (RDMA).

Find available VM instance sizes

To see a list of VM instance sizes available in a particular region, use the [Get-AzVmSize](#) command.

```
Get-AzVmSize -Location "EastUS"
```

The output is similar to the following condensed example, which shows the resources assigned to each VM size:

Name	NumberOfCores	MemoryInMB	MaxDataDiskCount	OSDiskSizeInMB	ResourceDiskSizeInMB
Standard_DS1_v2	1	3584	4	1047552	7168
Standard_DS2_v2	2	7168	8	1047552	14336
[...]					
Standard_A0	1	768	1	1047552	20480
Standard_A1	1	1792	2	1047552	71680
[...]					
Standard_F1	1	2048	4	1047552	16384
Standard_F2	2	4096	8	1047552	32768
[...]					
Standard_NV6	6	57344	24	1047552	389120
Standard_NV12	12	114688	48	1047552	696320

When you created a scale set at the start of the tutorial, a default VM SKU of *Standard_DS1_v2* was provided for the VM instances. You can specify a different VM instance size based on the output from [Get-AzVmSize](#). The following example would create a scale set with the `-VmSize` parameter to specify a VM instance size of *Standard_F1*. As it takes a few minutes to create and configure all the scale set resources and VM instances, you don't have to deploy the following scale set:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup3" ` 
-Location "EastUS" ` 
-VMSScaleSetName "myScaleSet3" ` 
-VirtualNetworkName "myVnet3" ` 
-SubnetName "mySubnet3" ` 
-PublicIpAddressName "myPublicIPAddress3" ` 
-LoadBalancerName "myLoadBalancer3" ` 
-UpgradePolicyMode "Automatic" ` 
-VmSize "Standard_F1" ` 
-Credential $cred
```

Change the capacity of a scale set

When you created a scale set, you requested two VM instances. To increase or decrease the number of VM instances in the scale set, you can manually change the capacity. The scale set creates or removes the required number of VM instances, then configures the load balancer to distribute traffic.

First, create a scale set object with [Get-AzVmss](#), then specify a new value for `sku.capacity`. To apply the capacity change, use [Update-AzVmss](#). The following example sets the number of VM instances in your scale set to 3:

```
# Get current scale set
$vmss = Get-AzVmss -ResourceGroupName "myResourceGroup" -VMSScaleSetName "myScaleSet"

# Set and update the capacity of your scale set
$vmss.sku.capacity = 3
Update-AzVmss -ResourceGroupName "myResourceGroup" -Name "myScaleSet" -VirtualMachineScaleSet $vmss
```

If takes a few minutes to update the capacity of your scale set. To see the number of instances you now have in the scale set, use [Get-AzVmss](#):

```
Get-AzVmss -ResourceGroupName "myResourceGroup" -VMSScaleSetName "myScaleSet"
```

The following example output shows that the capacity of the scale set is now 3:

```
Sku      : 
  Name    : Standard_DS2
  Tier    : Standard
  Capacity : 3
```

Common management tasks

You can now create a scale set, list connection information, and connect to VM instances. You learned how you could use a different OS image for your VM instances, select a different VM size, or manually scale the number of instances. As part of day to day management, you may need to stop, start, or restart the VM instances in your scale set.

Stop and deallocate VM instances in a scale set

To stop one or more VMs in a scale set, use [Stop-AzVmss](#). The `-InstanceId` parameter allows you to specify one or more VMs to stop. If you do not specify an instance ID, all VMs in the scale set are stopped. The following example stops instance 1:

```
Stop-AzVmss -ResourceGroupName "myResourceGroup" -VMSScaleSetName "myScaleSet" -InstanceId "1"
```

By default, stopped VMs are deallocated and do not incur compute charges. If you wish the VM to remain in a provisioned state when stopped, add the `-StayProvisioned` parameter to the preceding command. Stopped VMs that remain provisioned incur regular compute charges.

Start VM instances in a scale set

To start one or more VMs in a scale set, use [Start-AzVmss](#). The `-InstanceId` parameter allows you to specify one or more VMs to start. If you do not specify an instance ID, all VMs in the scale set are started. The following example starts instance 1:

```
Start-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "1"
```

Restart VM instances in a scale set

To restart one or more VMs in a scale set, use [Restart-AzVmss](#). The `-InstanceId` parameter allows you to specify one or more VMs to restart. If you do not specify an instance ID, all VMs in the scale set are restarted. The following example restarts instance 1:

```
Restart-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "1"
```

Clean up resources

When you delete a resource group, all resources contained within, such as the VM instances, virtual network, and disks, are also deleted. The `-Force` parameter confirms that you wish to delete the resources without an additional prompt to do so. The `-AsJob` parameter returns control to the prompt without waiting for the operation to complete.

```
Remove-AzResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this tutorial, you learned how to perform some basic scale set creation and management tasks with Azure PowerShell:

- Create and connect to a virtual machine scale set
- Select and use VM images
- View and use specific VM sizes
- Manually scale a scale set
- Perform common scale set management tasks

Advance to the next tutorial to learn about scale set disks.

[Use data disks with scale sets](#)

Tutorial: Create and use disks with virtual machine scale set with the Azure CLI

1/19/2020 • 9 minutes to read • [Edit Online](#)

Virtual machine scale sets use disks to store the VM instance's operating system, applications, and data. As you create and manage a scale set, it is important to choose a disk size and configuration appropriate to the expected workload. This tutorial covers how to create and manage VM disks. In this tutorial you learn how to:

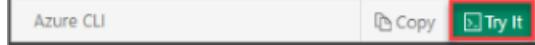
- OS disks and temporary disks
- Data disks
- Standard and Premium disks
- Disk performance
- Attach and prepare data disks

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Default Azure disks

When a scale set is created or scaled, two disks are automatically attached to each VM instance.

Operating system disk - Operating system disks can be sized up to 2 TB, and hosts the VM instance's operating system. The OS disk is labeled `/dev/sda` by default. The disk caching configuration of the OS disk is optimized for OS performance. Because of this configuration, the OS disk **should not** host applications or data. For applications and data, use data disks, which are detailed later in this article.

Temporary disk - Temporary disks use a solid-state drive that is located on the same Azure host as the VM instance. These are high-performance disks and may be used for operations such as temporary data processing. However, if the VM instance is moved to a new host, any data stored on a temporary disk is removed. The size of the temporary disk is determined by the VM instance size. Temporary disks are labeled `/dev/sdb` and have a mountpoint of `/mnt`.

Temporary disk sizes

Type	Common sizes	Max temp disk size (GiB)
General purpose	A, B, and D series	1600
Compute optimized	F series	576
Memory optimized	D, E, G, and M series	6144
Storage optimized	L series	5630
GPU	N series	1440
High performance	A and H series	2000

Azure data disks

Additional data disks can be added if you need to install applications and store data. Data disks should be used in any situation where durable and responsive data storage is desired. Each data disk has a maximum capacity of 4 TB. The size of the VM instance determines how many data disks can be attached. For each VM vCPU, two data disks can be attached.

Max data disks per VM

Type	Common sizes	Max data disks per VM
General purpose	A, B, and D series	64
Compute optimized	F series	64
Memory optimized	D, E, G, and M series	64
Storage optimized	L series	64
GPU	N series	64
High performance	A and H series	64

VM disk types

Azure provides two types of disk.

Standard disk

Standard Storage is backed by HDDs, and delivers cost-effective storage and performance. Standard disks are ideal for a cost effective dev and test workload.

Premium disk

Premium disks are backed by SSD-based high-performance, low-latency disk. These disks are recommended for VMs that run production workloads. Premium Storage supports DS-series, DSv2-series, GS-series, and FS-series VMs. When you select a disk size, the value is rounded up to the next type. For example, if the disk size is less than 128 GB, the disk type is P10. If the disk size is between 129 GB and 512 GB, the size is a P20. Over, 512 GB, the size is a P30.

Premium disk performance

PREMIUM STORAGE DISK TYPE	P4	P6	P10	P20	P30	P40	P50
Disk size (round up)	32 GB	64 GB	128 GB	512 GB	1,024 GB (1 TB)	2,048 GB (2 TB)	4,095 GB (4 TB)
Max IOPS per disk	120	240	500	2,300	5,000	7,500	7,500
Throughput per disk	25 MB/s	50 MB/s	100 MB/s	150 MB/s	200 MB/s	250 MB/s	250 MB/s

While the above table identifies max IOPS per disk, a higher level of performance can be achieved by striping multiple data disks. For instance, a Standard_GS5 VM can achieve a maximum of 80,000 IOPS. For detailed information on max IOPS per VM, see [Linux VM sizes](#).

Create and attach disks

You can create and attach disks when you create a scale set, or with an existing scale set.

Attach disks at scale set creation

First, create a resource group with the `az group create` command. In this example, a resource group named `myResourceGroup` is created in the `eastus` region.

```
az group create --name myResourceGroup --location eastus
```

Create a virtual machine scale set with the `az vmss create` command. The following example creates a scale set named `myScaleSet`, and generates SSH keys if they do not exist. Two disks are created with the

`--data-disk-sizes-gb` parameter. The first disk is 64 GB in size, and the second disk is 128 GB:

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy automatic \
--admin-username azureuser \
--generate-ssh-keys \
--data-disk-sizes-gb 64 128
```

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

Attach a disk to existing scale set

You can also attach disks to an existing scale set. Use the scale set created in the previous step to add another disk with [az vmss disk attach](#). The following example attaches an additional 128 GB disk:

```
az vmss disk attach \
--resource-group myResourceGroup \
--name myScaleSet \
--size-gb 128
```

Prepare the data disks

The disks that are created and attached to your scale set VM instances are raw disks. Before you can use them with your data and applications, the disks must be prepared. To prepare the disks, you create a partition, create a filesystem, and mount them.

To automate the process across multiple VM instances in a scale set, you can use the Azure Custom Script Extension. This extension can execute scripts locally on each VM instance, such as to prepare attached data disks. For more information, see the [Custom Script Extension overview](#).

The following example executes a script from a GitHub sample repo on each VM instance with [az vmss extension set](#) that prepares all the raw attached data disks:

```
az vmss extension set \
--publisher Microsoft.Azure.Extensions \
--version 2.0 \
--name CustomScript \
--resource-group myResourceGroup \
--vmss-name myScaleSet \
--settings '{"fileUris":["https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/prepare_vm_disks.sh"],"commandToExecute": "./prepare_vm_disks.sh"}'
```

To confirm that the disks have been prepared correctly, SSH to one of the VM instances. List the connection information for your scale set with [az vmss list-instance-connection-info](#):

```
az vmss list-instance-connection-info \
--resource-group myResourceGroup \
--name myScaleSet
```

Use your own public IP address and port number to connect to the first VM instance, as shown in the following example:

```
ssh azureuser@52.226.67.166 -p 50001
```

Examine the partitions on the VM instance as follows:

```
sudo fdisk -l
```

The following example output shows that three disks are attached to the VM instance - `/dev/sdc`, `/dev/sdd`, and `/dev/sde`. Each of those disks has a single partition that uses all the available space:

```

Disk /dev/sdc: 64 GiB, 68719476736 bytes, 134217728 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa47874cb

Device      Boot Start       End   Sectors Size Id Type
/dev/sdc1        2048 134217727 134215680  64G 83 Linux

Disk /dev/sdd: 128 GiB, 137438953472 bytes, 268435456 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xd5c95ca0

Device      Boot Start       End   Sectors Size Id Type
/dev/sdd1        2048 268435455 268433408 128G 83 Linux

Disk /dev/sde: 128 GiB, 137438953472 bytes, 268435456 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x9312fdf5

Device      Boot Start       End   Sectors Size Id Type
/dev/sde1        2048 268435455 268433408 128G 83 Linux

```

Examine the filesystems and mount points on the VM instance as follows:

```
sudo df -h
```

The following example output shows that the three disks have their filesystems correctly mounted under `/datadisks`:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda1	30G	1.3G	28G	5%	/
/dev/sdb1	50G	52M	47G	1%	/mnt
/dev/sdc1	63G	52M	60G	1%	/datadisks/disk1
/dev/sdd1	126G	60M	120G	1%	/datadisks/disk2
/dev/sde1	126G	60M	120G	1%	/datadisks/disk3

The disks on each VM instance in your scale are automatically prepared in the same way. As your scale set would scale up, the required data disks are attached to the new VM instances. The Custom Script Extension also runs to automatically prepare the disks.

Close the SSH connection to your VM instance:

```
exit
```

List attached disks

To view information about disks attached to a scale set, use [az vmss show](#) and query on `virtualMachineProfile.storageProfile.dataDisks`:

```
az vmss show \
--resource-group myResourceGroup \
--name myScaleSet \
--query virtualMachineProfile.storageProfile.dataDisks
```

Information on the disk size, storage tier, and LUN (Logical Unit Number) is shown. The following example output details the three data disks attached to the scale set:

```
[  
 {  
   "additionalProperties": {},  
   "caching": "None",  
   "createOption": "Empty",  
   "diskSizeGb": 64,  
   "lun": 0,  
   "managedDisk": {  
     "additionalProperties": {},  
     "storageAccountType": "Standard_LRS"  
   },  
   "name": null  
 },  
 {  
   "additionalProperties": {},  
   "caching": "None",  
   "createOption": "Empty",  
   "diskSizeGb": 128,  
   "lun": 1,  
   "managedDisk": {  
     "additionalProperties": {},  
     "storageAccountType": "Standard_LRS"  
   },  
   "name": null  
 },  
 {  
   "additionalProperties": {},  
   "caching": "None",  
   "createOption": "Empty",  
   "diskSizeGb": 128,  
   "lun": 2,  
   "managedDisk": {  
     "additionalProperties": {},  
     "storageAccountType": "Standard_LRS"  
   },  
   "name": null  
 }  
 ]
```

Detach a disk

When you no longer need a given disk, you can detach it from the scale set. The disk is removed from all VM instances in the scale set. To detach a disk from a scale set, use [az vmss disk detach](#) and specify the LUN of the disk. The LUNs are shown in the output from [az vmss show](#) in the previous section. The following example detaches LUN 2 from the scale set:

```
az vmss disk detach \
--resource-group myResourceGroup \
--name myScaleSet \
--lun 2
```

Clean up resources

To remove your scale set and disks, delete the resource group and all its resources with [az group delete](#). The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --no-wait --yes
```

Next steps

In this tutorial, you learned how to create and use disks with scale sets with the Azure CLI:

- OS disks and temporary disks
- Data disks
- Standard and Premium disks
- Disk performance
- Attach and prepare data disks

Advance to the next tutorial to learn how to use a custom image for your scale set VM instances.

[Use a custom image for scale set VM instances](#)

Tutorial: Create and use disks with virtual machine scale set with Azure PowerShell

1/19/2020 • 10 minutes to read • [Edit Online](#)

Virtual machine scale sets use disks to store the VM instance's operating system, applications, and data. As you create and manage a scale set, it is important to choose a disk size and configuration appropriate to the expected workload. This tutorial covers how to create and manage VM disks. In this tutorial you learn how to:

- OS disks and temporary disks
- Data disks
- Standard and Premium disks
- Disk performance
- Attach and prepare data disks

If you don't have an Azure subscription, create a [free account](#) before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by

selecting **Cmd+Shift+V** on macOS.

4. Select **Enter** to run the code.

Default Azure disks

When a scale set is created or scaled, two disks are automatically attached to each VM instance.

Operating system disk - Operating system disks can be sized up to 2 TB, and hosts the VM instance's operating system. The OS disk is labeled `/dev/sda` by default. The disk caching configuration of the OS disk is optimized for OS performance. Because of this configuration, the OS disk **should not** host applications or data. For applications and data, use data disks, which are detailed later in this article.

Temporary disk - Temporary disks use a solid-state drive that is located on the same Azure host as the VM instance. These are high-performance disks and may be used for operations such as temporary data processing. However, if the VM instance is moved to a new host, any data stored on a temporary disk is removed. The size of the temporary disk is determined by the VM instance size. Temporary disks are labeled `/dev/sdb` and have a mountpoint of `/mnt`.

Temporary disk sizes

Type	Common Sizes	Max Temp Disk Size (GiB)
General purpose	A, B, and D series	1600
Compute optimized	F series	576
Memory optimized	D, E, G, and M series	6144
Storage optimized	L series	5630
GPU	N series	1440
High performance	A and H series	2000

Azure data disks

Additional data disks can be added if you need to install applications and store data. Data disks should be used in any situation where durable and responsive data storage is desired. Each data disk has a maximum capacity of 4 TB. The size of the VM instance determines how many data disks can be attached. For each VM vCPU, two data disks can be attached.

Max data disks per VM

Type	Common Sizes	Max Data Disks per VM
General purpose	A, B, and D series	64
Compute optimized	F series	64
Memory optimized	D, E, G, and M series	64
Storage optimized	L series	64
GPU	N series	64

Type	Common sizes	Max data disks per VM
High performance	A and H series	64

VM disk types

Azure provides two types of disk.

Standard disk

Standard Storage is backed by HDDs, and delivers cost-effective storage and performance. Standard disks are ideal for a cost effective dev and test workload.

Premium disk

Premium disks are backed by SSD-based high-performance, low-latency disks. These disks are recommended for VMs that run production workloads. Premium Storage supports DS-series, DSv2-series, GS-series, and FS-series VMs. When you select a disk size, the value is rounded up to the next type. For example, if the disk size is less than 128 GB, the disk type is P10. If the disk size is between 129 GB and 512 GB, the size is a P20. Over 512 GB, the size is a P30.

Premium disk performance

Premium storage disk type	P4	P6	P10	P20	P30	P40	P50
Disk size (round up)	32 GB	64 GB	128 GB	512 GB	1,024 GB (1 TB)	2,048 GB (2 TB)	4,095 GB (4 TB)
Max IOPS per disk	120	240	500	2,300	5,000	7,500	7,500
Throughput per disk	25 MB/s	50 MB/s	100 MB/s	150 MB/s	200 MB/s	250 MB/s	250 MB/s

While the above table identifies max IOPS per disk, a higher level of performance can be achieved by striping multiple data disks. For instance, a Standard_GS5 VM can achieve a maximum of 80,000 IOPS. For detailed information on max IOPS per VM, see [Windows VM sizes](#).

Create and attach disks

You can create and attach disks when you create a scale set, or with an existing scale set.

Attach disks at scale set creation

Create a virtual machine scale set with [New-AzVms](#). When prompted, provide a username and password for the VM instances. To distribute traffic to the individual VM instances, a load balancer is also created. The load balancer includes rules to distribute traffic on TCP port 80, as well as allow remote desktop traffic on TCP port 3389 and PowerShell remoting on TCP port 5985.

Two disks are created with the `-DataDiskSizeGb` parameter. The first disk is 64 GB in size, and the second disk is 128 GB. When prompted, provide your own desired administrative credentials for the VM instances in the scale set:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "EastUS" ` 
-VMScaleSetName "myScaleSet" ` 
-VirtualNetworkName "myVnet" ` 
-SubnetName "mySubnet" ` 
-PublicIpAddressName "myPublicIPAddress" ` 
-LoadBalancerName "myLoadBalancer" ` 
-UpgradePolicyMode "Automatic" ` 
-DataDiskSizeInGb 64,128
```

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

Attach a disk to existing scale set

You can also attach disks to an existing scale set. Use the scale set created in the previous step to add another disk with [Add-AzVmssDataDisk](#). The following example attaches an additional 128 GB disk to an existing scale set:

```
# Get scale set object
$vmss = Get-AzVmss ` 
    -ResourceGroupName "myResourceGroup" ` 
    -VMScaleSetName "myScaleSet"

# Attach a 128 GB data disk to LUN 2
Add-AzVmssDataDisk ` 
    -VirtualMachineScaleSet $vmss ` 
    -CreateOption Empty ` 
    -Lun 2 ` 
    -DiskSizeGB 128

# Update the scale set to apply the change
Update-AzVmss ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Name "myScaleSet" ` 
    -VirtualMachineScaleSet $vmss
```

Prepare the data disks

The disks that are created and attached to your scale set VM instances are raw disks. Before you can use them with your data and applications, the disks must be prepared. To prepare the disks, you create a partition, create a filesystem, and mount them.

To automate the process across multiple VM instances in a scale set, you can use the Azure Custom Script Extension. This extension can execute scripts locally on each VM instance, such as to prepare attached data disks. For more information, see the [Custom Script Extension overview](#).

The following example executes a script from a GitHub sample repo on each VM instance with [Add-AzVmssExtension](#) that prepares all the raw attached data disks:

```

# Get scale set object
$vmss = Get-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -VMScaleSetName "myScaleSet"

# Define the script for your Custom Script Extension to run
$publicSettings = @{

    "fileUris" = (,"https://raw.githubusercontent.com/Azure-Samples/compute-automation-
configurations/master/prepare_vm_disks.ps1");
    "commandToExecute" = "powershell -ExecutionPolicy Unrestricted -File prepare_vm_disks.ps1"
}

# Use Custom Script Extension to prepare the attached data disks
Add-AzVmssExtension -VirtualMachineScaleSet $vmss `

    -Name "customScript" `

    -Publisher "Microsoft.Compute" `

    -Type "CustomScriptExtension" `

    -TypeHandlerVersion 1.8 `

    -Setting $publicSettings

# Update the scale set and apply the Custom Script Extension to the VM instances
Update-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -Name "myScaleSet" `

    -VirtualMachineScaleSet $vmss

```

To confirm that the disks have been prepared correctly, RDP to one of the VM instances.

First, get the load balancer object with [Get-AzLoadBalancer](#). Then, view the inbound NAT rules with [Get-AzLoadBalancerInboundNatRuleConfig](#). The NAT rules list the *FrontendPort* for each VM instance that RDP listens on. Finally, get the public IP address of the load balancer with [Get-AzPublicIpAddress](#):

```

# Get the load balancer object
$lb = Get-AzLoadBalancer -ResourceGroupName "myResourceGroup" -Name "myLoadBalancer"

# View the list of inbound NAT rules
Get-AzLoadBalancerInboundNatRuleConfig -LoadBalancer $lb | Select-Object
Name,Protocol,FrontEndPort,BackEndPort

# View the public IP address of the load balancer
Get-AzPublicIpAddress -ResourceGroupName "myResourceGroup" -Name myPublicIPAddress | Select IpAddress

```

To connect to your VM, specify your own public IP address and port number of the required VM instance, as shown from the preceding commands. When prompted, enter the credentials used when you created the scale set. If you use the Azure Cloud Shell, perform this step from a local PowerShell prompt or Remote Desktop Client. The following example connects to VM instance 1:

```
mstsc /v 52.168.121.216:50001
```

Open a local PowerShell session on the VM instance and look at the connected disks with [Get-Disk](#):

```
Get-Disk
```

The following example output shows that the three data disks are attached to the VM instance.

Number	Friendly Name	HealthStatus	OperationalStatus	Total Size	Partition Style
0	Virtual HD	Healthy	Online	127 GB	MBR
1	Virtual HD	Healthy	Online	14 GB	MBR
2	Msft Virtual Disk	Healthy	Online	64 GB	MBR
3	Msft Virtual Disk	Healthy	Online	128 GB	MBR
4	Msft Virtual Disk	Healthy	Online	128 GB	MBR

Examine the filesystems and mount points on the VM instance as follows:

```
Get-Partition
```

The following example output shows that the three data disks are assigned drive letters:

```
DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#000001

PartitionNumber DriveLetter Offset Size Type
----- ----- ----- ----- -----
1 F 1048576 64 GB IFS

DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#000002

PartitionNumber DriveLetter Offset Size Type
----- ----- ----- ----- -----
1 G 1048576 128 GB IFS

DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#000003

PartitionNumber DriveLetter Offset Size Type
----- ----- ----- ----- -----
1 H 1048576 128 GB IFS
```

The disks on each VM instance in your scale set are automatically prepared in the same way. As your scale set would scale up, the required data disks are attached to the new VM instances. The Custom Script Extension also runs to automatically prepare the disks.

Close the remote desktop connection session with the VM instance.

List attached disks

To view information about disks attached to a scale set, use [Get-AzVmss](#) as follows:

```
Get-AzVmss -ResourceGroupName "myResourceGroup" -Name "myScaleSet"
```

Under the `VirtualMachineProfile.StorageProfile` property, the list of `DataDisks` is shown. Information on the disk size, storage tier, and LUN (Logical Unit Number) is shown. The following example output details the three data disks attached to the scale set:

```

DataDisks[0]          :
  Lun                  : 0
  Caching              : None
  CreateOption         : Empty
  DiskSizeGB           : 64
  ManagedDisk          :
    StorageAccountType: PremiumLRS
DataDisks[1]          :
  Lun                  : 1
  Caching              : None
  CreateOption         : Empty
  DiskSizeGB           : 128
  ManagedDisk          :
    StorageAccountType: PremiumLRS
DataDisks[2]          :
  Lun                  : 2
  Caching              : None
  CreateOption         : Empty
  DiskSizeGB           : 128
  ManagedDisk          :
    StorageAccountType: PremiumLRS

```

Detach a disk

When you no longer need a given disk, you can detach it from the scale set. The disk is removed from all VM instances in the scale set. To detach a disk from a scale set, use [Remove-AzVmssDataDisk](#) and specify the LUN of the disk. The LUNs are shown in the output from [Get-AzVmss](#) in the previous section. The following example detaches LUN 3 from the scale set:

```

# Get scale set object
$vmss = Get-AzVmss `

  -ResourceGroupName "myResourceGroup" `

  -VMScaleSetName "myScaleSet"

# Detach a disk from the scale set
Remove-AzVmssDataDisk `

  -VirtualMachineScaleSet $vmss `

  -Lun 2

# Update the scale set and detach the disk from the VM instances
Update-AzVmss `

  -ResourceGroupName "myResourceGroup" `

  -Name "myScaleSet" `

  -VirtualMachineScaleSet $vmss

```

Clean up resources

To remove your scale set and disks, delete the resource group and all its resources with [Remove-AzResourceGroup](#). The `-Force` parameter confirms that you wish to delete the resources without an additional prompt to do so. The `-AsJob` parameter returns control to the prompt without waiting for the operation to complete.

```
Remove-AzResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this tutorial, you learned how to create and use disks with scale sets with Azure PowerShell:

- OS disks and temporary disks
- Data disks
- Standard and Premium disks
- Disk performance
- Attach and prepare data disks

Advance to the next tutorial to learn how to use a custom image for your scale set VM instances.

[Use a custom image for scale set VM instances](#)

Tutorial: Create and use a custom image for virtual machine scale sets with the Azure CLI

1/19/2020 • 5 minutes to read • [Edit Online](#)

When you create a scale set, you specify an image to be used when the VM instances are deployed. To reduce the number of tasks after VM instances are deployed, you can use a custom VM image. This custom VM image includes any required application installs or configurations. Any VM instances created in the scale set use the custom VM image and are ready to serve your application traffic. In this tutorial you learn how to:

- Create and customize a VM
- Deprovision and generalize the VM
- Create a custom VM image
- Deploy a scale set that uses the custom VM image

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create and configure a source VM

NOTE

This tutorial walks through the process of creating and using a generalized VM image. It is not supported to create a scale set from a specialized VM image.

First, create a resource group with `az group create`, then create a VM with `az vm create`. This VM is then used as the source for a custom VM image. The following example creates a VM named `myVM` in the resource group named `myResourceGroup`:

```
az group create --name myResourceGroup --location eastus

az vm create \
    --resource-group myResourceGroup \
    --name myVM \
    --image ubuntults \
    --admin-username azureuser \
    --generate-ssh-keys
```

The public IP address of your VM is shown in the output of the `az vm create` command. SSH to the public IP address of your VM as follows:

```
ssh azureuser@<publicIpAddress>
```

To customize your VM, let's install a basic web server. When the VM instance in the scale set would be deployed, it would then have all the required packages to run a web application. Use `apt-get` to install `NG/NX` as follows:

```
sudo apt-get install -y nginx
```

The final step to prepare your VM for use as a custom image is to deprovision your VM. This step removes machine-specific information from the VM and makes it possible to deploy many VMs from a single image. When the VM is deprovisioned, the host name is reset to `localhost.localdomain`. SSH host keys, nameserver configurations, root password, and cached DHCP leases are also deleted.

To deprovision the VM, use the Azure VM agent (`waagent`). The Azure VM agent is installed on every VM and is used to communicate with the Azure platform. The `-force` parameter tells the agent to accept prompts to reset the machine-specific information.

```
sudo waagent -deprovision+user -force
```

Close your SSH connection to the VM:

```
exit
```

Create a custom VM image from the source VM

The source VM is now customized with the Nginx web server installed. Let's create the custom VM image to use with a scale set.

To create an image, the VM needs to be deallocated. Deallocate the VM with `az vm deallocate`. Then, set the state of the VM as generalized with `az vm generalize` so that the Azure platform knows the VM is ready for use as a custom image. You can only create an image from a generalized VM:

```
az vm deallocate --resource-group myResourceGroup --name myVM  
az vm generalize --resource-group myResourceGroup --name myVM
```

It may take a few minutes to deallocate and generalize the VM.

Now, create an image of the VM with [az image create](#). The following example creates an image named *myImage* from your VM:

[NOTE] If the Resource Group and Virtual Machine location are different, you can add the `--location` parameter to the below commands to specify the location of source VM used to create the image.

```
az image create \  
  --resource-group myResourceGroup \  
  --name myImage \  
  --source myVM
```

Create a scale set from the custom VM image

Create a scale set with [az vmss create](#). Instead of a platform image, such as *Ubuntu LTS* or *CentOS*, specify the name of your custom VM image. The following example creates a scale set named *myScaleSet* that uses the custom image named *myImage* from the previous step:

```
az vmss create \  
  --resource-group myResourceGroup \  
  --name myScaleSet \  
  --image myImage \  
  --admin-username azureuser \  
  --generate-ssh-keys
```

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

Test your scale set

To allow traffic to reach your scale set and verify that the web server works correctly, create a load balancer rule with [az network lb rule create](#). The following example creates a rule named *myLoadBalancerRuleWeb* that allows traffic on *TCP port 80*:

```
az network lb rule create \  
  --resource-group myResourceGroup \  
  --name myLoadBalancerRuleWeb \  
  --lb-name myScaleSetLB \  
  --backend-pool-name myScaleSetLBEPool \  
  --backend-port 80 \  
  --frontend-ip-name loadBalancerFrontEnd \  
  --frontend-port 80 \  
  --protocol tcp
```

To see your scale set in action, get the public IP address of your load balancer with [az network public-ip show](#). The following example gets the IP address for *myScaleSetLBPublicIP* created as part of the scale set:

```
az network public-ip show \
--resource-group myResourceGroup \
--name myScaleSetLBPublicIP \
--query [ipAddress] \
--output tsv
```

Type the public IP address into your web browser. The default NGINX web page is displayed, as shown in the following example:



Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with `az group delete`. The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --no-wait --yes
```

Next steps

In this tutorial, you learned how to create and use a custom VM image for your scale sets with the Azure CLI:

- Create and customize a VM
- Deprovision and generalize the VM
- Create a custom VM image
- Deploy a scale set that uses the custom VM image

Advance to the next tutorial to learn how to deploy applications to your scale set.

[Deploy applications to your scale sets](#)

Tutorial: Create and use a custom image for virtual machine scale sets with Azure PowerShell

1/19/2020 • 6 minutes to read • [Edit Online](#)

When you create a scale set, you specify an image to be used when the VM instances are deployed. To reduce the number of tasks after VM instances are deployed, you can use a custom VM image. This custom VM image includes any required application installs or configurations. Any VM instances created in the scale set use the custom VM image and are ready to serve your application traffic. In this tutorial you learn how to:

- Create and customize a VM
- Deprovision and generalize the VM
- Create a custom VM image from the source VM
- Deploy a scale set that uses the custom VM image

If you don't have an Azure subscription, create a [free account](#) before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by

selecting **Cmd+Shift+V** on macOS.

4. Select **Enter** to run the code.

Create and configure a source VM

NOTE

This tutorial walks through the process of creating and using a generalized VM image. Creating a scale set from a specialized VHD is not supported.

First, create a resource group with [New-AzResourceGroup](#), then create a VM with [New-AzVM](#). This VM is then used as the source for a custom VM image. The following example creates a VM named *myCustomVM* in the resource group named *myResourceGroup*. When prompted, enter a username and password to be used as logon credentials for the VM:

```
# Create a resource a group
New-AzResourceGroup -Name "myResourceGroup" -Location "EastUS"

# Create a Windows Server 2016 Datacenter VM
New-AzVm ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Name "myCustomVM" ` 
    -ImageName "Win2016Datacenter" ` 
    -OpenPorts 3389
```

To connect to your VM, list the public IP address with [Get-AzPublicIpAddress](#) as follows:

```
Get-AzPublicIpAddress -ResourceGroupName myResourceGroup | SelectIpAddress
```

Create a remote connection with the VM. If you use the Azure Cloud Shell, perform this step from a local PowerShell prompt or Remote Desktop Client. Provide your own IP address from the previous command. When prompted, enter the credentials used when you created the VM in the first step:

```
mstsc /v:<IpAddress>
```

To customize your VM, let's install a basic web server. When the VM instance in the scale set would be deployed, it would then have all the required packages to run a web application. Open a local PowerShell prompt on the VM and install the IIS web server with [Install-WindowsFeature](#) as follows:

```
Install-WindowsFeature -name Web-Server -IncludeManagementTools
```

The final step to prepare your VM for use as a custom image is to generalize the VM. Sysprep removes all your personal account information and configurations, and resets the VM to a clean state for future deployments. For more information, see [How to Use Sysprep: An Introduction](#).

To generalize the VM, run Sysprep and set the VM for an out-of-the-box experience. When finished, instruct Sysprep to shut down the VM:

```
C:\Windows\system32\sysprep\sysprep.exe /oobe /generalize /shutdown
```

The remote connection to the VM is automatically closed when Sysprep completes the process and the VM is shut

down.

Create a custom VM image from the source VM

The source VM now customized with the IIS web server installed. Let's create the custom VM image to use with a scale set.

To create an image, the VM needs to be deallocated. Deallocate the VM with [Stop-AzVm](#). Then, set the state of the VM as generalized with [Set-AzVm](#) so that the Azure platform knows the VM is ready for use a custom image. You can only create an image from a generalized VM:

```
Stop-AzVm -ResourceGroupName "myResourceGroup" -Name "myCustomVM" -Force  
Set-AzVm -ResourceGroupName "myResourceGroup" -Name "myCustomVM" -Generalized
```

It may take a few minutes to deallocate and generalize the VM.

Now, create an image of the VM with [New-AzImageConfig](#) and [New-AzImage](#). The following example creates an image named *myImage* from your VM:

```
# Get VM object  
$vm = Get-AzVm -Name "myCustomVM" -ResourceGroupName "myResourceGroup"  
  
# Create the VM image configuration based on the source VM  
$image = New-AzImageConfig -Location "EastUS" -SourceVirtualMachineId $vm.ID  
  
# Create the custom VM image  
New-AzImage -Image $image -ImageName "myImage" -ResourceGroupName "myResourceGroup"
```

Configure the Network Security Group Rules

Before creating the Scale Set, we need to configure the associating Network Security Group rules to allow access to HTTP, RDP and Remoting

```
$rule1 = New-AzNetworkSecurityRuleConfig -Name web-rule -Description "Allow HTTP" -Access Allow -Protocol Tcp  
-Direction Inbound -Priority 100 -SourceAddressPrefix Internet -SourcePortRange * -DestinationAddressPrefix *  
-DestinationPortRange 80  
  
$rule2 = New-AzNetworkSecurityRuleConfig -Name rdp-rule -Description "Allow RDP" -Access Allow -Protocol Tcp  
-Direction Inbound -Priority 110 -SourceAddressPrefix Internet -SourcePortRange * -DestinationAddressPrefix *  
-DestinationPortRange 3389  
  
$rule3 = New-AzNetworkSecurityRuleConfig -Name remoting-rule -Description "Allow PS Remoting" -Access Allow -  
Protocol Tcp -Direction Inbound -Priority 120 -SourceAddressPrefix Internet -SourcePortRange * -  
DestinationAddressPrefix * -DestinationPortRange 5985  
  
New-AzNetworkSecurityGroup -Name "myNSG" -ResourceGroupName "myResourceGroup" -Location "EastUS" -  
SecurityRules $rule1,$rule2,$rule3
```

Create a scale set from the custom VM image

Now create a scale set with [New-AzVmss](#) that uses the `-ImageName` parameter to define the custom VM image created in the previous step. To distribute traffic to the individual VM instances, a load balancer is also created. The load balancer includes rules to distribute traffic on TCP port 80, as well as allow remote desktop traffic on TCP port 3389 and PowerShell remoting on TCP port 5985. When prompted, provide your own desired administrative credentials for the VM instances in the scale set:

```
New-AzVmss
  -ResourceGroupName "myResourceGroup"
  -Location "EastUS"
  -VMScaleSetName "myScaleSet"
  -VirtualNetworkName "myVnet"
  -SubnetName "mySubnet"
  -SecurityGroupName "myNSG"
  -PublicIpAddressName "myPublicIPAddress"
  -LoadBalancerName "myLoadBalancer"
  -UpgradePolicyMode "Automatic"
  -ImageName "myImage"
```

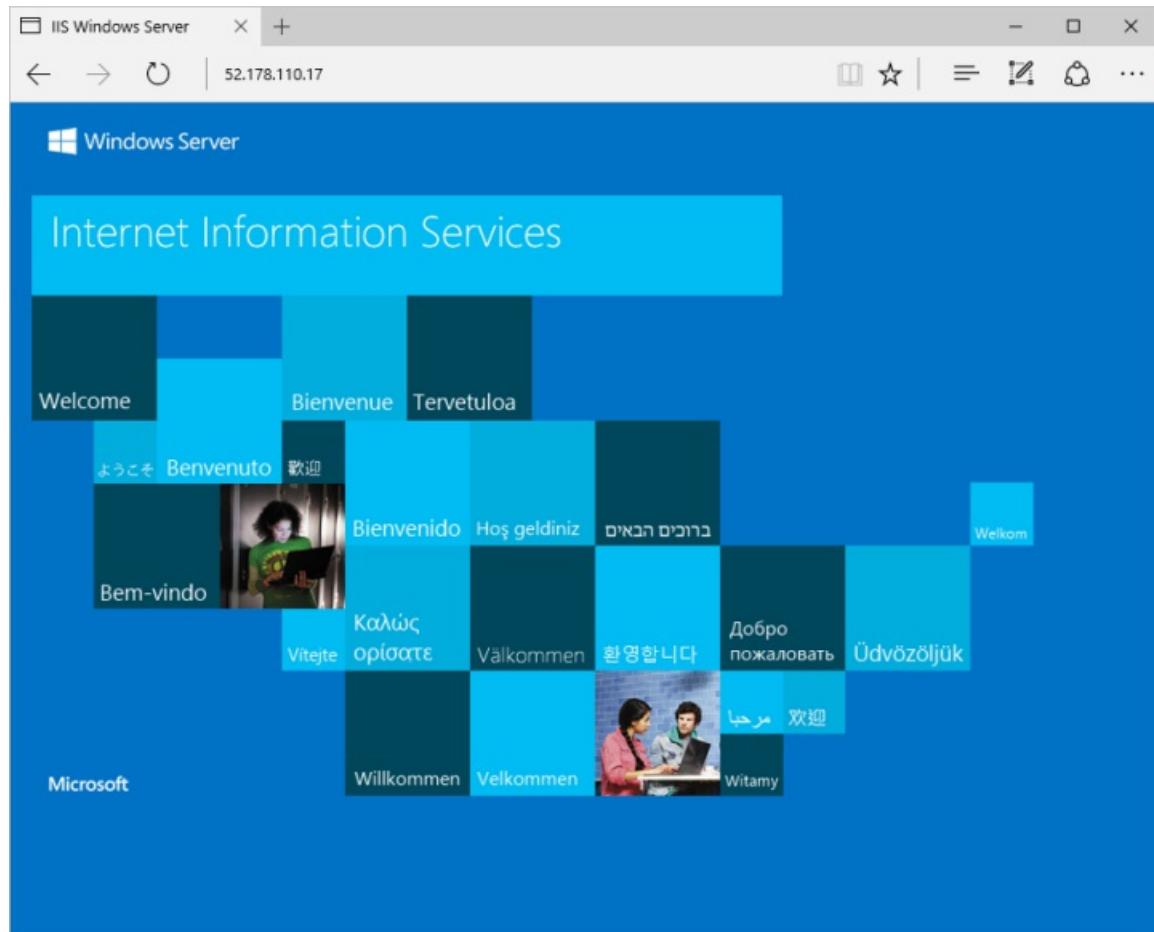
It takes a few minutes to create and configure all the scale set resources and VMs.

Test your scale set

To see your scale set in action, get the public IP address of your load balancer with [Get-AzPublicIpAddress](#) as follows:

```
Get-AzPublicIpAddress
  -ResourceGroupName "myResourceGroup"
  -Name "myPublicIPAddress" | Select IpAddress
```

Type the public IP address into your web browser. The default IIS web page is displayed, as shown in the following example:



Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with [Remove-AzResourceGroup](#). The `-Force` parameter confirms that you wish to delete the resources without an additional

prompt to do so. The `-AsJob` parameter returns control to the prompt without waiting for the operation to complete.

```
Remove-AzResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this tutorial, you learned how to create and use a custom VM image for your scale sets with Azure PowerShell:

- Create and customize a VM
- Deprovision and generalize the VM
- Create a custom VM image
- Deploy a scale set that uses the custom VM image

Advance to the next tutorial to learn how to deploy applications to your scale set.

[Deploy applications to your scale sets](#)

Tutorial: Install applications in virtual machine scale sets with the Azure CLI

1/19/2020 • 6 minutes to read • [Edit Online](#)

To run applications on virtual machine (VM) instances in a scale set, you first need to install the application components and required files. In a previous tutorial, you learned how to create and use a custom VM image to deploy your VM instances. This custom image included manual application installs and configurations. You can also automate the install of applications to a scale set after each VM instance is deployed, or update an application that already runs on a scale set. In this tutorial you learn how to:

- Automatically install applications to your scale set
- Use the Azure Custom Script Extension
- Update a running application on a scale set

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

What is the Azure Custom Script Extension?

The Custom Script Extension downloads and executes scripts on Azure VMs. This extension is useful for post deployment configuration, software installation, or any other configuration / management task. Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run-time.

The Custom Script extension integrates with Azure Resource Manager templates, and can also be used with the Azure CLI, Azure PowerShell, Azure portal, or the REST API. For more information, see the [Custom Script Extension overview](#).

To use the Custom Script Extension with the Azure CLI, you create a JSON file that defines what files to obtain and commands to execute. These JSON definitions can be reused across scale set deployments to apply consistent application installs.

Create Custom Script Extension definition

To see the Custom Script Extension in action, let's create a scale set that installs the NGINX web server and outputs the hostname of the scale set VM instance. The following Custom Script Extension definition downloads a sample script from GitHub, installs the required packages, then writes the VM instance hostname to a basic HTML page.

In your current shell, create a file named *customConfig.json* and paste the following configuration. For example, create the file in the Cloud Shell not on your local machine. You can use any editor you wish. Enter `sensible-editor customConfig.json` in the Cloud Shell to create the file and see a list of available editors.

```
{
  "fileUris": ["https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/automate_nginx.sh"],
  "commandToExecute": "./automate_nginx.sh"
}
```

Create a scale set

Create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

```
az group create --name myResourceGroup --location eastus
```

Now create a virtual machine scale set with [az vmss create](#). The following example creates a scale set named *myScaleSet*, and generates SSH keys if they do not exist:

```
az vmss create \
  --resource-group myResourceGroup \
  --name myScaleSet \
  --image UbuntuLTS \
  --upgrade-policy-mode automatic \
  --admin-username azureuser \
  --generate-ssh-keys
```

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

Apply the Custom Script Extension

Apply the Custom Script Extension configuration to the VM instances in your scale set with [az vmss extension set](#). The following example applies the *customConfig.json* configuration to the *myScaleSet* VM instances in the resource group named *myResourceGroup*:

```
az vmss extension set \
--publisher Microsoft.Azure.Extensions \
--version 2.0 \
--name CustomScript \
--resource-group myResourceGroup \
--vmss-name myScaleSet \
--settings @customConfig.json
```

Each VM instance in the scale set downloads and runs the script from GitHub. In a more complex example, multiple application components and files could be installed. If the scale set is scaled up, the new VM instances automatically apply the same Custom Script Extension definition and install the required application.

Test your scale set

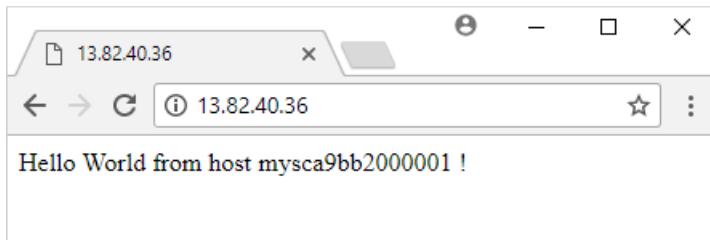
To allow traffic to reach the web server, create a load balancer rule with [az network lb rule create](#). The following example creates a rule named *myLoadBalancerRuleWeb*:

```
az network lb rule create \
--resource-group myResourceGroup \
--name myLoadBalancerRuleWeb \
--lb-name myScaleSetLB \
--backend-pool-name myScaleSetLBBEPool \
--backend-port 80 \
--frontend-ip-name loadBalancerFrontEnd \
--frontend-port 80 \
--protocol tcp
```

To see your web server in action, obtain the public IP address of your load balancer with [az network public-ip show](#). The following example obtains the IP address for *myScaleSetLBPublicIP* created as part of the scale set:

```
az network public-ip show \
--resource-group myResourceGroup \
--name myScaleSetLBPublicIP \
--query [ipAddress] \
--output tsv
```

Enter the public IP address of the load balancer in to a web browser. The load balancer distributes traffic to one of your VM instances, as shown in the following example:



Leave the web browser open so that you can see an updated version in the next step.

Update app deployment

Throughout the lifecycle of a scale set, you may need to deploy an updated version of your application. With the Custom Script Extension, you can reference an updated deploy script and then reapply the extension to your scale set. When the scale set was created in a previous step, the `--upgrade-policy-mode` was set to *automatic*. This setting allows the VM instances in the scale set to automatically update and apply the latest version of your application.

In your current shell, create a file named `customConfigv2.json` and paste the following configuration. This definition runs an updated v2 version of the application install script:

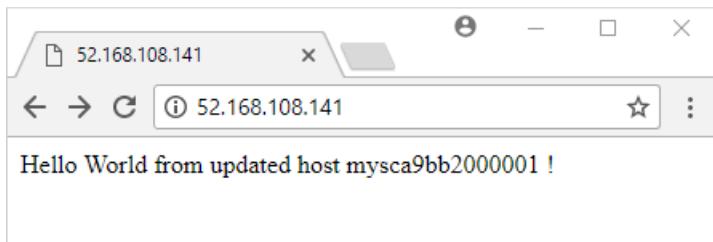
```
{  
  "fileUris": ["https://raw.githubusercontent.com/Azure-Samples/compute-automation-  
configurations/master/automate_nginx_v2.sh"],  
  "commandToExecute": "./automate_nginx_v2.sh"  
}
```

Apply the Custom Script Extension configuration to the VM instances in your scale set again with [az vmss extension set](#). The `customConfigv2.json` is used to apply the updated version of the application:

```
az vmss extension set \  
  --publisher Microsoft.Azure.Extensions \  
  --version 2.0 \  
  --name CustomScript \  
  --resource-group myResourceGroup \  

```

All VM instances in the scale set are automatically updated with the latest version of the sample web page. To see the updated version, refresh the web site in your browser:



Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with [az group delete](#). The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --no-wait --yes
```

Next steps

In this tutorial, you learned how to automatically install and update applications on your scale set with the Azure CLI:

- Automatically install applications to your scale set
- Use the Azure Custom Script Extension
- Update a running application on a scale set

Advance to the next tutorial to learn how to automatically scale your scale set.

[Automatically scale your scale sets](#)

Tutorial: Install applications in virtual machine scale sets with Azure PowerShell

1/19/2020 • 6 minutes to read • [Edit Online](#)

To run applications on virtual machine (VM) instances in a scale set, you first need to install the application components and required files. In a previous tutorial, you learned how to create and use a custom VM image to deploy your VM instances. This custom image included manual application installs and configurations. You can also automate the install of applications to a scale set after each VM instance is deployed, or update an application that already runs on a scale set. In this tutorial you learn how to:

- Automatically install applications to your scale set
- Use the Azure Custom Script Extension
- Update a running application on a scale set

If you don't have an Azure subscription, create a [free account](#) before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by

selecting **Cmd+Shift+V** on macOS.

4. Select **Enter** to run the code.

What is the Azure Custom Script Extension?

The Custom Script Extension downloads and executes scripts on Azure VMs. This extension is useful for post deployment configuration, software installation, or any other configuration / management task. Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run-time.

The Custom Script extension integrates with Azure Resource Manager templates. It can also be used with the Azure CLI, Azure PowerShell, Azure portal, or the REST API. For more information, see the [Custom Script Extension overview](#).

To see the Custom Script Extension in action, create a scale set that installs the IIS web server and outputs the hostname of the scale set VM instance. The Custom Script Extension definition downloads a sample script from GitHub, installs the required packages, then writes the VM instance hostname to a basic HTML page.

Create a scale set

Now create a virtual machine scale set with [New-AzVmss](#). To distribute traffic to the individual VM instances, a load balancer is also created. The load balancer includes rules to distribute traffic on TCP port 80. It also allows remote desktop traffic on TCP port 3389 and PowerShell remoting on TCP port 5985. When prompted, you can set your own administrative credentials for the VM instances in the scale set:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-VMSScaleSetName "myScaleSet" ` 
-Location "EastUS" ` 
-VirtualNetworkName "myVnet" ` 
-SubnetName "mySubnet" ` 
-PublicIpAddressName "myPublicIPAddress" ` 
-LoadBalancerName "myLoadBalancer" ` 
-UpgradePolicyMode "Automatic"
```

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

Create Custom Script Extension definition

Azure PowerShell uses a hashtable to store the file to download and the command to execute. In the following example, a sample script from GitHub is used. First, create this configuration object as follows:

```
$customConfig = @{
    "fileUris" = (,"https://raw.githubusercontent.com/Azure-Samples/compute-automation-
configurations/master/automate-iis.ps1");
    "commandToExecute" = "powershell -ExecutionPolicy Unrestricted -File automate-iis.ps1"
}
```

Now, apply the Custom Script Extension with [Add-AzVmssExtension](#). The configuration object previously defined is passed to the extension. Update and run the extension on the VM instances with [Update-AzVmss](#).

```
# Get information about the scale set
$vmss = Get-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -VMScaleSetName "myScaleSet"

# Add the Custom Script Extension to install IIS and configure basic website
$vmss = Add-AzVmssExtension `

    -VirtualMachineScaleSet $vmss `

    -Name "customScript" `

    -Publisher "Microsoft.Compute" `

    -Type "CustomScriptExtension" `

    -TypeHandlerVersion 1.9 `

    -Setting $customConfig

# Update the scale set and apply the Custom Script Extension to the VM instances
Update-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -Name "myScaleSet" `

    -VirtualMachineScaleSet $vmss
```

Each VM instance in the scale set downloads and runs the script from GitHub. In a more complex example, multiple application components and files could be installed. If the scale set is scaled up, the new VM instances automatically apply the same Custom Script Extension definition and install the required application.

Allow traffic to application

To allow access to the basic web application, create a network security group with [New-AzNetworkSecurityRuleConfig](#) and [New-AzNetworkSecurityGroup](#). For more information, see [Networking for Azure virtual machine scale sets](#).

```

#Create a rule to allow traffic over port 80
$nsgFrontendRule = New-AzNetworkSecurityRuleConfig ` 
    -Name myFrontendNSGRule ` 
    -Protocol Tcp ` 
    -Direction Inbound ` 
    -Priority 200 ` 
    -SourceAddressPrefix * ` 
    -SourcePortRange * ` 
    -DestinationAddressPrefix * ` 
    -DestinationPortRange 80 ` 
    -Access Allow

#Create a network security group and associate it with the rule
$nsgFrontend = New-AzNetworkSecurityGroup ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Location EastUS ` 
    -Name myFrontendNSG ` 
    -SecurityRules $nsgFrontendRule

$vnet = Get-AzVirtualNetwork ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Name myVnet

$frontendSubnet = $vnet.Subnets[0]

$frontendSubnetConfig = Set-AzVirtualNetworkSubnetConfig ` 
    -VirtualNetwork $vnet ` 
    -Name mySubnet ` 
    -AddressPrefix $frontendSubnet.AddressPrefix ` 
    -NetworkSecurityGroup $nsgFrontend

Set-AzVirtualNetwork -VirtualNetwork $vnet

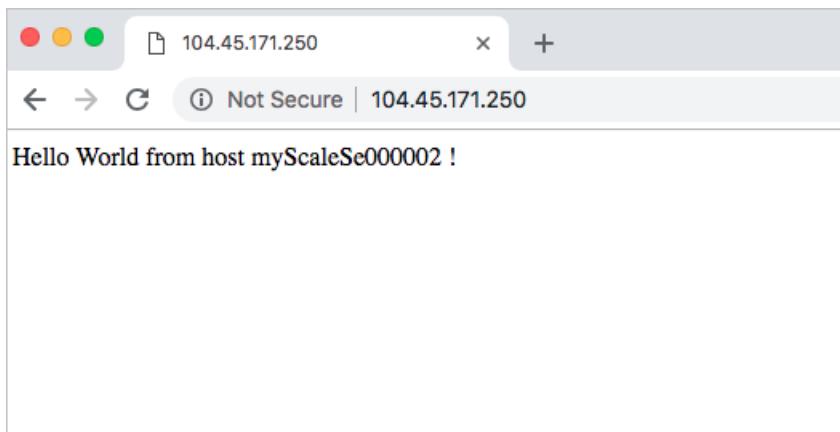
```

Test your scale set

To see your web server in action, get the public IP address of your load balancer with [Get-AzPublicIpAddress](#). The following example displays the IP address created in the *myResourceGroup* resource group:

```
Get-AzPublicIpAddress -ResourceGroupName "myResourceGroup" | Select IpAddress
```

Enter the public IP address of the load balancer in to a web browser. The load balancer distributes traffic to one of your VM instances, as shown in the following example:



Leave the web browser open so that you can see an updated version in the next step.

Update app deployment

Throughout the lifecycle of a scale set, you may need to deploy an updated version of your application. With the Custom Script Extension, you can reference an updated deploy script and then reapply the extension to your scale set. When the scale set was created in a previous step, the `-UpgradePolicyMode` was set to *Automatic*. This setting allows the VM instances in the scale set to automatically update and apply the latest version of your application.

Create a new config definition named *customConfigv2*. This definition runs an updated v2 version of the application install script:

```
$customConfigv2 = @{
    "fileUris" = (,"https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/automate-iis-v2.ps1");
    "commandToExecute" = "powershell -ExecutionPolicy Unrestricted -File automate-iis-v2.ps1"
}
```

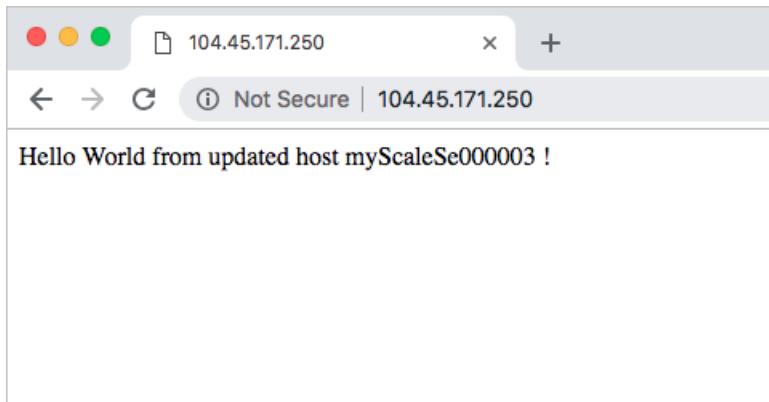
Update the Custom Script Extension configuration to the VM instances in your scale set. The *customConfigv2* definition is used to apply the updated version of the application:

```
$vmss = Get-AzVmss ` 
    -ResourceGroupName "myResourceGroup" ` 
    -VMScaleSetName "myScaleSet"

$vmss.VirtualMachineProfile.ExtensionProfile[0].Extensions[0].Settings = $customConfigv2

Update-AzVmss ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Name "myScaleSet" ` 
    -VirtualMachineScaleSet $vmss
```

All VM instances in the scale set are automatically updated with the latest version of the sample web page. To see the updated version, refresh the web site in your browser:



Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with `Remove-AzResourceGroup`. The `-Force` parameter confirms that you wish to delete the resources without an additional prompt to do so. The `-AsJob` parameter returns control to the prompt without waiting for the operation to complete.

```
Remove-AzResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this tutorial, you learned how to automatically install and update applications on your scale set with Azure PowerShell:

- Automatically install applications to your scale set
- Use the Azure Custom Script Extension
- Update a running application on a scale set

Advance to the next tutorial to learn how to automatically scale your scale set.

[Automatically scale your scale sets](#)

Tutorial: Install applications in virtual machine scale sets with an Azure template

1/19/2020 • 6 minutes to read • [Edit Online](#)

To run applications on virtual machine (VM) instances in a scale set, you first need to install the application components and required files. In a previous tutorial, you learned how to create and use a custom VM image to deploy your VM instances. This custom image included manual application installs and configurations. You can also automate the install of applications to a scale set after each VM instance is deployed, or update an application that already runs on a scale set. In this tutorial you learn how to:

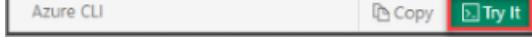
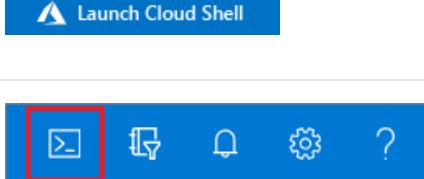
- Automatically install applications to your scale set
- Use the Azure Custom Script Extension
- Update a running application on a scale set

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

What is the Azure Custom Script Extension?

The Custom Script Extension downloads and executes scripts on Azure VMs. This extension is useful for post deployment configuration, software installation, or any other configuration / management task. Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run-time.

The Custom Script extension integrates with Azure Resource Manager templates, and can also be used with the Azure CLI, Azure PowerShell, Azure portal, or the REST API. For more information, see the [Custom Script Extension overview](#).

To see the Custom Script Extension in action, create a scale set that installs the NGINX web server and outputs the hostname of the scale set VM instance. The following Custom Script Extension definition downloads a sample script from GitHub, installs the required packages, then writes the VM instance hostname to a basic HTML page.

Create Custom Script Extension definition

When you define a virtual machine scale set with an Azure template, the `Microsoft.Compute/virtualMachineScaleSets` resource provider can include a section on extensions. The `extensionsProfile` details what is applied to the VM instances in a scale set. To use the Custom Script Extension, you specify a publisher of `Microsoft.Azure.Extensions` and a type of `CustomScript`.

The `fileUris` property is used to define the source install scripts or packages. To start the install process, the required scripts are defined in `commandToExecute`. The following example defines a sample script from GitHub that installs and configures the NGINX web server:

```
"extensionProfile": {
  "extensions": [
    {
      "name": "AppInstall",
      "properties": {
        "publisher": "Microsoft.Azure.Extensions",
        "type": "CustomScript",
        "typeHandlerVersion": "2.0",
        "autoUpgradeMinorVersion": true,
        "settings": {
          "fileUris": [
            "https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/automate_nginx.sh"
          ],
          "commandToExecute": "bash automate_nginx.sh"
        }
      }
    }
  ]
}
```

For a complete example of an Azure template that deploys a scale set and the Custom Script Extension, see https://github.com/Azure-Samples/compute-automation-configurations/blob/master/scale_sets/azuredeploy.json. This sample template is used in the next section.

Create a scale set

Let's use the sample template to create a scale set and apply the Custom Script Extension. First, create a resource group with [az group create](#). The following example creates a resource group named `myResourceGroup` in the `eastus` location:

```
az group create --name myResourceGroup --location eastus
```

Now create a virtual machine scale set with [az group deployment create](#). When prompted, provide your own

username and password that is used as the credentials for each VM instance:

```
az group deployment create \
--resource-group myResourceGroup \
--template-uri https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/scale_sets/azuredeploy.json
```

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

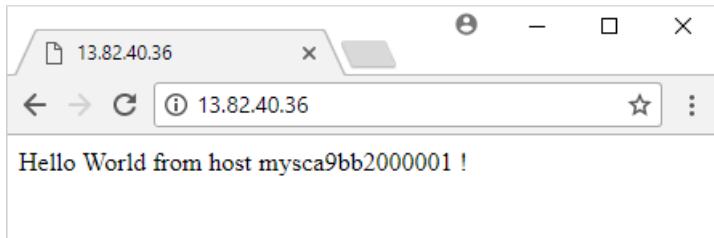
Each VM instance in the scale set downloads and runs the script from GitHub. In a more complex example, multiple application components and files could be installed. If the scale set is scaled up, the new VM instances automatically apply the same Custom Script Extension definition and install the required application.

Test your scale set

To see your web server in action, obtain the public IP address of your load balancer with [az network public-ip show](#). The following example obtains the IP address for *myScaleSetPublicIP* created as part of the scale set:

```
az network public-ip show \
--resource-group myResourceGroup \
--name myScaleSetPublicIP \
--query [ipAddress] \
--output tsv
```

Enter the public IP address of the load balancer in to a web browser. The load balancer distributes traffic to one of your VM instances, as shown in the following example:



Leave the web browser open so that you can see an updated version in the next step.

Update app deployment

Throughout the lifecycle of a scale set, you may need to deploy an updated version of your application. With the Custom Script Extension, you can reference an updated deploy script and then reapply the extension to your scale set. When the scale set was created in a previous step, the *upgradePolicy* was set to *Automatic*. This setting allows the VM instances in the scale set to automatically update and apply the latest version of your application.

To update the Custom Script Extension definition, edit your template to reference a new install script. A new filename must be used for the Custom Script Extension to recognize the change. The Custom Script Extension does not examine the contents of the script to determine any changes. This following definition uses an updated install script with *_v2* appended to its name:

```

"extensionProfile": {
  "extensions": [
    {
      "name": "AppInstall",
      "properties": {
        "publisher": "Microsoft.Azure.Extensions",
        "type": "CustomScript",
        "typeHandlerVersion": "2.0",
        "autoUpgradeMinorVersion": true,
        "settings": {
          "fileUris": [
            "https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/automate_nginx_v2.sh"
          ],
          "commandToExecute": "bash automate_nginx_v2.sh"
        }
      }
    }
  ]
}

```

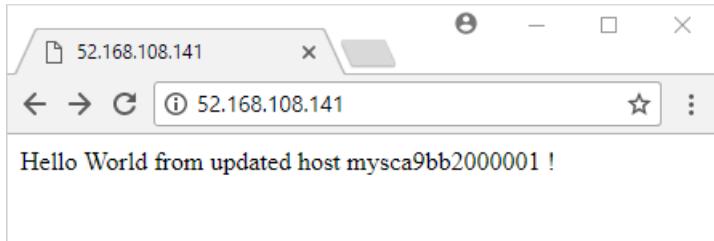
Apply the Custom Script Extension configuration to the VM instances in your scale set again with [az group deployment create](#). This `azuredeployv2.json` template is used to apply the updated version of the application. In practice, you edit the existing `azuredeploy.json` template to reference the updated install script, as shown in the previous section. When prompted, enter the same username and password credentials as used when you first created the scale set:

```

az group deployment create \
--resource-group myResourceGroup \
--template-uri https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/scale_sets/azuredeploy_v2.json

```

All VM instances in the scale set are automatically updated with the latest version of the sample web page. To see the updated version, refresh the web site in your browser:



Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with [az group delete](#). The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```

az group delete --name myResourceGroup --no-wait --yes

```

Next steps

In this tutorial, you learned how to automatically install and update applications on your scale set with Azure templates:

- Automatically install applications to your scale set

- Use the Azure Custom Script Extension
- Update a running application on a scale set

Advance to the next tutorial to learn how to automatically scale your scale set.

[Automatically scale your scale sets](#)

Tutorial: Automatically scale a virtual machine scale set with the Azure CLI

1/19/2020 • 7 minutes to read • [Edit Online](#)

When you create a scale set, you define the number of VM instances that you wish to run. As your application demand changes, you can automatically increase or decrease the number of VM instances. The ability to autoscale lets you keep up with customer demand or respond to application performance changes throughout the lifecycle of your app. In this tutorial you learn how to:

- Use autoscale with a scale set
- Create and use autoscale rules
- Stress-test VM instances and trigger autoscale rules
- Autoscale back in as demand is reduced

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.32 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a scale set

Create a resource group with [az group create](#) as follows:

```
az group create --name myResourceGroup --location eastus
```

Now create a virtual machine scale set with [az vmss create](#). The following example creates a scale set with an instance count of 2, and generates SSH keys if they do not exist:

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--instance-count 2 \
--admin-username azureuser \
--generate-ssh-keys
```

Define an autoscale profile

To enable autoscale on a scale set, you first define an autoscale profile. This profile defines the default, minimum, and maximum scale set capacity. These limits let you control cost by not continually creating VM instances, and balance acceptable performance with a minimum number of instances that remain in a scale-in event. Create an autoscale profile with [az monitor autoscale create](#). The following example sets the default, and minimum, capacity of 2 VM instances, and a maximum of 10:

```
az monitor autoscale create \
--resource-group myResourceGroup \
--resource myScaleSet \
--resource-type Microsoft.Compute/virtualMachineScaleSets \
--name autoscale \
--min-count 2 \
--max-count 10 \
--count 2
```

Create a rule to autoscale out

If your application demand increases, the load on the VM instances in your scale set increases. If this increased load is consistent, rather than just a brief demand, you can configure autoscale rules to increase the number of VM instances in the scale set. When these VM instances are created and your applications are deployed, the scale set starts to distribute traffic to them through the load balancer. You control what metrics to monitor, such as CPU or disk, how long the application load must meet a given threshold, and how many VM instances to add to the scale set.

Let's create a rule with [az monitor autoscale rule create](#) that increases the number of VM instances in a scale set when the average CPU load is greater than 70% over a 5-minute period. When the rule triggers, the number of VM instances is increased by three.

```
az monitor autoscale rule create \
--resource-group myResourceGroup \
--autoscale-name autoscale \
--condition "Percentage CPU > 70 avg 5m" \
--scale out 3
```

Create a rule to autoscale in

On an evening or weekend, your application demand may decrease. If this decreased load is consistent over a period of time, you can configure autoscale rules to decrease the number of VM instances in the scale set. This scale-in action reduces the cost to run your scale set as you only run the number of instances required to meet the current demand.

Create another rule with [az monitor autoscale rule create](#) that decreases the number of VM instances in a scale set when the average CPU load then drops below 30% over a 5-minute period. The following example defines the rule to scale in the number of VM instances by one:

```
az monitor autoscale rule create \
--resource-group myResourceGroup \
--autoscale-name autoscale \
--condition "Percentage CPU < 30 avg 5m" \
--scale in 1
```

Generate CPU load on scale set

To test the autoscale rules, generate some CPU load on the VM instances in the scale set. This simulated CPU load causes the autoscales to scale out and increase the number of VM instances. As the simulated CPU load is then decreased, the autoscale rules scale in and reduce the number of VM instances.

First, list the address and ports to connect to VM instances in a scale set with [az vmss list-instance-connection-info](#):

```
az vmss list-instance-connection-info \
--resource-group myResourceGroup \
--name myScaleSet
```

The following example output shows the instance name, public IP address of the load balancer, and port number that the Network Address Translation (NAT) rules forward traffic to:

```
{
  "instance 1": "13.92.224.66:50001",
  "instance 3": "13.92.224.66:50003"
}
```

SSH to your first VM instance. Specify your own public IP address and port number with the `-p` parameter, as shown from the preceding command:

```
ssh azureuser@13.92.224.66 -p 50001
```

Once logged in, install the **stress** utility. Start 10 **stress** workers that generate CPU load. These workers run for 420 seconds, which is enough to cause the autoscale rules to implement the desired action.

```
sudo apt-get -y install stress
sudo stress --cpu 10 --timeout 420 &
```

When **stress** shows output similar to *stress: info: [2688] dispatching hogs: 10 cpu, 0 io, 0 vm, 0 hdd*, press the *Enter* key to return to the prompt.

To confirm that **stress** generates CPU load, examine the active system load with the **top** utility:

```
top
```

Exit **top**, then close your connection to the VM instance. **stress** continues to run on the VM instance.

```
Ctrl-c  
exit
```

Connect to second VM instance with the port number listed from the previous [az vmss list-instance-connection-info](#):

```
ssh azureuser@13.92.224.66 -p 50003
```

Install and run **stress**, then start ten workers on this second VM instance.

```
sudo apt-get -y install stress  
sudo stress --cpu 10 --timeout 420 &
```

Again, when **stress** shows output similar to *stress: info: [2713] dispatching hogs: 10 cpu, 0 io, 0 vm, 0 hdd*, press the *Enter* key to return to the prompt.

Close your connection to the second VM instance. **stress** continues to run on the VM instance.

```
exit
```

Monitor the active autoscale rules

To monitor the number of VM instances in your scale set, use **watch**. It takes 5 minutes for the autoscale rules to begin the scale-out process in response to the CPU load generated by **stress** on each of the VM instances:

```
watch az vmss list-instances \  
--resource-group myResourceGroup \  
--name myScaleSet \  
--output table
```

Once the CPU threshold has been met, the autoscale rules increase the number of VM instances in the scale set. The following output shows three VMs created as the scale set autoscales out:

```
Every 2.0s: az vmss list-instances --resource-group myResourceGroup --name myScaleSet --output table

  InstanceId  LatestModelApplied  Location  Name  ProvisioningState  ResourceGroup  VmId
  -----  -----  -----  -----  -----  -----  -----
  1  True  eastus  myScaleSet_1  Succeeded  myResourceGroup  4f92f350-
2b68-464f-8a01-e5e590557955
  2  True  eastus  myScaleSet_2  Succeeded  myResourceGroup  d734cd3d-
fb38-4302-817c-cfe35655d48e
  4  True  eastus  myScaleSet_4  Creating  myResourceGroup  061b4c90-
0d73-49fc-a066-19eab0b3d95c
  5  True  eastus  myScaleSet_5  Creating  myResourceGroup  4beff8b9-
4e65-40cb-9652-43899309da27
  6  True  eastus  myScaleSet_6  Creating  myResourceGroup  9e4133dd-
2c57-490e-ae45-90513ce3b336
```

Once **stress** stops on the initial VM instances, the average CPU load returns to normal. After another 5 minutes, the autoscale rules then scale in the number of VM instances. Scale in actions remove VM instances with the highest IDs first. When a scale set uses Availability Sets or Availability Zones, scale in actions are evenly distributed across those VM instances. The following example output shows one VM instance deleted as the scale set autoscales in:

```
6 True          eastus      myScaleSet_6 Deleting      myResourceGroup 9e4133dd-  
2c57-490e-ae45-90513ce3b336
```

Exit *watch* with `ctrl-c`. The scale set continues to scale in every 5 minutes and remove one VM instance until the minimum instance count of two is reached.

Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with `az group delete`. The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --yes --no-wait
```

Next steps

In this tutorial, you learned how to automatically scale in or out a scale set with the Azure CLI:

- Use autoscale with a scale set
- Create and use autoscale rules
- Stress-test VM instances and trigger autoscale rules
- Autoscale back in as demand is reduced

For more examples of virtual machine scale sets in action, see the following sample Azure CLI sample scripts:

[Scale set script samples for Azure CLI](#)

Tutorial: Automatically scale a virtual machine scale set with Azure PowerShell

1/19/2020 • 10 minutes to read • [Edit Online](#)

IMPORTANT

Using this Azure feature from PowerShell requires the `AzureRM` module installed. This is an older module only available for Windows PowerShell 5.1 that no longer receives new features. The `Az` and `AzureRM` modules are **not** compatible when installed for the same versions of PowerShell. If you need both versions:

1. [Uninstall the Az module from a PowerShell 5.1 session.](#)
2. [Install the AzureRM module from a PowerShell 5.1 session.](#)
3. [Download and install PowerShell Core 6.x or later.](#)
4. [Install the Az module in a PowerShell Core session.](#)

When you create a scale set, you define the number of VM instances that you wish to run. As your application demand changes, you can automatically increase or decrease the number of VM instances. The ability to autoscale lets you keep up with customer demand or respond to application performance changes throughout the lifecycle of your app. In this tutorial you learn how to:

- Use autoscale with a scale set
- Create and use autoscale rules
- Stress-test VM instances and trigger autoscale rules
- Autoscale back in as demand is reduced

If you don't have an Azure subscription, create a [free account](#) before you begin.

There's a known issue that affects Azure PowerShell module version 6.8.1 or later, including the current version of the Azure Cloud Shell. This tutorial can only run using Azure PowerShell module version 6.0.0 to 6.8.0. Run `Get-Module -ListAvailable AzureRM` to find the version. If you are running PowerShell locally, you also need to run `Connect-AzureRmAccount` to create a connection with Azure.

Create a scale set

To make it easier to create the autoscale rules, define some variables for your scale set. The following example defines variables for the scale set named `myScaleSet` in the resource group named `myResourceGroup` and in the *East US* region. Your subscription ID is obtained with [Get-AzureRmSubscription](#). If you have multiple subscriptions associated with your account, only the first subscription is returned. Adjust the names and subscription ID as follows:

```
$mySubscriptionId = (Get-AzureRmSubscription)[0].Id
$myResourceGroup = "myResourceGroup"
$myScaleSet = "myScaleSet"
$myLocation = "East US"
```

Now create a virtual machine scale set with [New-AzureRmVmss](#). To distribute traffic to the individual VM instances, a load balancer is also created. The load balancer includes rules to distribute traffic on TCP port 80, as well as allow remote desktop traffic on TCP port 3389 and PowerShell remoting on TCP port 5985. When prompted, provide your own desired administrative credentials for the VM instances in the scale set:

```

New-AzureRmVmss ` 
-ResourceGroupName $myResourceGroup ` 
-VMSScaleSetName $myScaleSet ` 
-Location $myLocation ` 
-VirtualNetworkName "myVnet" ` 
-SubnetName "mySubnet" ` 
-PublicIpAddressName "myPublicIPAddress" ` 
-LoadBalancerName "myLoadBalancer"

```

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

Create a rule to autoscale out

If your application demand increases, the load on the VM instances in your scale set increases. If this increased load is consistent, rather than just a brief demand, you can configure autoscale rules to increase the number of VM instances in the scale set. When these VM instances are created and your applications are deployed, the scale set starts to distribute traffic to them through the load balancer. You control what metrics to monitor, such as CPU or disk, how long the application load must meet a given threshold, and how many VM instances to add to the scale set.

Let's create a rule with [New-AzureRmAutoscaleRule](#) that increases the number of VM instances in a scale set when the average CPU load is greater than 70% over a 5-minute period. When the rule triggers, the number of VM instances is increased by three.

The following parameters are used for this rule:

PARAMETER	EXPLANATION	VALUE
<code>-MetricName</code>	The performance metric to monitor and apply scale set actions on.	Percentage CPU
<code>-TimeGrain</code>	How often the metrics are collected for analysis.	1 minute
<code>-MetricStatistic</code>	Defines how the collected metrics should be aggregated for analysis.	Average
<code>-TimeWindow</code>	The amount of time monitored before the metric and threshold values are compared.	5 minutes
<code>-Operator</code>	Operator used to compare the metric data against the threshold.	Greater Than
<code>-Threshold</code>	The value that causes the autoscale rule to trigger an action.	70%
<code>-ScaleActionDirection</code>	Defines if the scale set should scale up or down when the rule applies.	Increase
<code>-ScaleActionScaleType</code>	Indicates that the number of VM instances should be changed by a specific value.	Change Count
<code>-ScaleActionValue</code>	The percentage of VM instances should be changed when the rule triggers.	3

PARAMETER	EXPLANATION	VALUE
<code>-ScaleActionCooldown</code>	The amount of time to wait before the rule is applied again so that the autoscale actions have time to take effect.	5 minutes

The following example creates an object named `myRuleScaleOut` that holds this scale up rule. The `-MetricResourceId` uses the variables previously defined for the subscription ID, resource group name, and scale set name:

```
$myRuleScaleOut = New-AzureRmAutoscaleRule ` 
    -MetricName "Percentage CPU" ` 
    -MetricResourceId 
    /subscriptions/$mySubscriptionId/resourceGroups/$myResourceGroup/providers/Microsoft.Compute/virtualMachineSca 
    leSets/$myScaleSet ` 
    -TimeGrain 00:01:00 ` 
    -MetricStatistic "Average" ` 
    -TimeWindow 00:05:00 ` 
    -Operator "GreaterThan" ` 
    -Threshold 70 ` 
    -ScaleActionDirection "Increase" ` 
    -ScaleActionScaleType "ChangeCount" ` 
    -ScaleActionValue 3 ` 
    -ScaleActionCooldown 00:05:00
```

Create a rule to autoscale in

On an evening or weekend, your application demand may decrease. If this decreased load is consistent over a period of time, you can configure autoscale rules to decrease the number of VM instances in the scale set. This scale-in action reduces the cost to run your scale set as you only run the number of instances required to meet the current demand.

Create another rule with [New-AzureRmAutoscaleRule](#) that decreases the number of VM instances in a scale set when the average CPU load then drops below 30% over a 5-minute period. When the rule triggers, the number of VM instances is decreased by one. The following example creates an object named `myRuleScaleDown` that holds this scale up rule. The `-MetricResourceId` uses the variables previously defined for the subscription ID, resource group name, and scale set name:

```
$myRuleScaleIn = New-AzureRmAutoscaleRule ` 
    -MetricName "Percentage CPU" ` 
    -MetricResourceId 
    /subscriptions/$mySubscriptionId/resourceGroups/$myResourceGroup/providers/Microsoft.Compute/virtualMachineSca 
    leSets/$myScaleSet ` 
    -Operator "LessThan" ` 
    -MetricStatistic "Average" ` 
    -Threshold 30 ` 
    -TimeGrain 00:01:00 ` 
    -TimeWindow 00:05:00 ` 
    -ScaleActionCooldown 00:05:00 ` 
    -ScaleActionDirection "Decrease" ` 
    -ScaleActionScaleType "ChangeCount" ` 
    -ScaleActionValue 1
```

Define an autoscale profile

To associate your autoscale rules with a scale set, create a profile. The autoscale profile defines the default, minimum, and maximum scale set capacity, and associates your autoscale rules. Create an autoscale profile with

[New-AzureRmAutoscaleProfile](#). The following example sets the default, and minimum, capacity of 2 VM instances, and a maximum of 10. The scale out and scale in rules created in the preceding steps are then attached:

```
$myScaleProfile = New-AzureRmAutoscaleProfile `  
    -DefaultCapacity 2 `  
    -MaximumCapacity 10 `  
    -MinimumCapacity 2 `  
    -Rule $myRuleScaleOut,$myRuleScaleIn `  
    -Name "autoprofile"
```

Apply autoscale profile to a scale set

The final step is to apply the autoscale profile to your scale set. Your scale set is then able to automatically scale in or out based on the application demand. Apply the autoscale profile with [Add-AzureRmAutoscaleSetting](#) as follows:

```
Add-AzureRmAutoscaleSetting `  
    -Location $myLocation `  
    -Name "autosetting" `  
    -ResourceGroup $myResourceGroup `  
    -TargetResourceId  
/subscriptions/$mySubscriptionId/resourceGroups/$myResourceGroup/providers/Microsoft.Compute/virtualMachineSca  
leSets/$myScaleSet `  
    -AutoscaleProfile $myScaleProfile
```

Generate CPU load on scale set

To test the autoscale rules, generate some CPU load on the VM instances in the scale set. This simulated CPU load causes the autoscale rules to scale out and increase the number of VM instances. As the simulated CPU load is then decreased, the autoscale rules scale in and reduce the number of VM instances.

To list the NAT ports to connect to VM instances in a scale set, first get the load balancer object with [Get-AzureRmLoadBalancer](#). Then, view the inbound NAT rules with [Get-AzureRmLoadBalancerInboundNatRuleConfig](#):

```
# Get the load balancer object  
$lb = Get-AzureRmLoadBalancer -ResourceGroupName "myResourceGroup" -Name "myLoadBalancer"  
  
# View the list of inbound NAT rules  
Get-AzureRmLoadBalancerInboundNatRuleConfig -LoadBalancer $lb | Select-Object  
Name,Protocol,FrontEndPort,BackEndPort
```

The following example output shows the instance name, public IP address of the load balancer, and port number that the NAT rules forward traffic to:

Name	Protocol	FrontendPort	BackendPort
myRDPRule.0	Tcp	50001	3389
myRDPRule.1	Tcp	50002	3389

The *Name* of the rule aligns with the name of the VM instance as shown in a previous [Get-AzureRmVmssVM](#) command. For example, to connect to VM instance 0, you use *myRDPRule.0* and connect to port 50001. To connect to VM instance 1, use the value from *myRDPRule.1* and connect to port 50002.

View the public IP address of the load balancer with [Get-AzureRmPublicIpAddress](#):

```
Get-AzureRmPublicIpAddress -ResourceGroupName "myResourceGroup" -Name myPublicIP | Select IPAddress
```

Example output:

```
IpAddress  
-----  
52.168.121.216
```

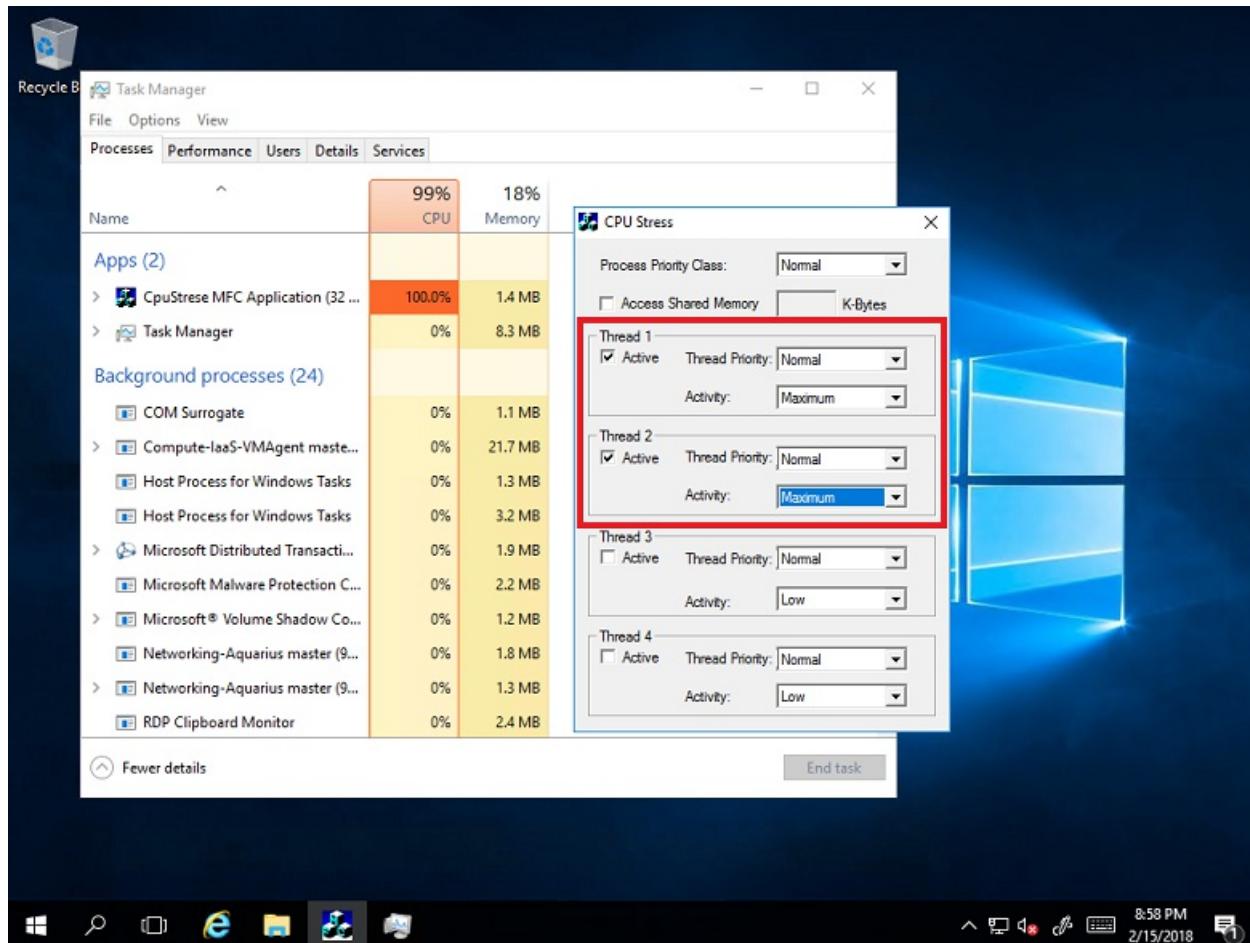
Create a remote connection to your first VM instance. Specify your own public IP address and port number of the required VM instance, as shown from the preceding commands. When prompted, enter the credentials used when you created the scale set (by default in the sample commands, they are *azureuser* and *P@ssw0rd!*). If you use the Azure Cloud Shell, perform this step from a local PowerShell prompt or Remote Desktop Client. The following example connects to VM instance 0:

```
mstsc /v 52.168.121.216:50001
```

Once logged in, open Internet Explorer from the taskbar.

- Select **OK** to accept the prompt to *Use recommended security, privacy, and compatibility settings*
- Type <http://download.sysinternals.com/files/CPUSTRES.zip> in the address bar.
- As Internet Explorer Enhanced Security Configuration is enabled, choose to **Add** the <http://download.sysinternals.com> domain to your list of trusted sites.
- When prompted for the file download, select **Open**, then select and **Run** the *CPUSTRES.EXE* tool.

To generate some CPU load, check two boxes for **Active** threads. From the **Activity** drop-down menu for both threads, select *Maximum*. You can open Task Manager to confirm that the CPU load on the VM is at 100%.



Leave the remote desktop connection session open, and open another remote desktop connection session.

Connect to the second VM instance with the port number listed from the previous [Get-AzureRmLoadBalancerInboundNatRuleConfig](#) cmdlet:

```
mstsc /v 52.168.121.216:50002
```

Once logged in to the second VM instance, repeat the previous steps to download and run *CPUSTRES.EXE*. Again, start two **Active** threads, and set the activity to *Maximum*.

To allow the **CPU Stress** tool to continue running, leave both remote desktop connection sessions open.

Monitor the active autoscale rules

To monitor the number of VM instances in your scale set, use **while**. It takes 5 minutes for the autoscale scales to begin the scale out process in response to the CPU load generated by **CPUSStress* on each of the VM instances:

```
while (1) {Get-AzureRmVmssVM ` 
    -ResourceGroupName $myResourceGroup ` 
    -VMScaleSetName $myScaleSet; sleep 10}
```

Once the CPU threshold has been met, the autoscale rules increase the number of VM instances in the scale set.

The following output shows three VMs created as the scale set autoscales out:

ResourceGroupName	Name	Location	Sku	Capacity	InstanceID	ProvisioningState
MYRESOURCEGROUP	myScaleSet_2	eastus	Standard_DS2	2		Succeeded
MYRESOURCEGROUP	myScaleSet_3	eastus	Standard_DS2	3		Succeeded
MYRESOURCEGROUP	myScaleSet_4	eastus	Standard_DS2	4		Creating
MYRESOURCEGROUP	myScaleSet_5	eastus	Standard_DS2	5		Creating
MYRESOURCEGROUP	myScaleSet_6	eastus	Standard_DS2	6		Creating

In your remote desktop connection session to each of your VM instances, close the **CPU Stress** tool. The average CPU load across the scale set returns to normal. After another 5 minutes, the autoscale rules then scale in the number of VM instances. Scale in actions remove VM instances with the highest IDs first. When a scale set uses Availability Sets or Availability Zones, scale in actions are evenly distributed across those VM instances. The following example output shows one VM instance deleted as the scale set autoscales in:

```
MYRESOURCEGROUP  myScaleSet_6  eastus Standard_DS2          6      Deleting
```

Exit **while** with **ctrl-c**. The scale set continues to scale in every 5 minutes and remove one VM instance until the minimum instance count of two is reached.

Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with [Remove-AzureRmResourceGroup](#). The **-Force** parameter confirms that you wish to delete the resources without an additional prompt to do so. The **-AsJob** parameter returns control to the prompt without waiting for the operation to complete.

```
Remove-AzureRmResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this tutorial, you learned how to automatically scale in or out a scale set with Azure PowerShell:

- Use autoscale with a scale set
- Create and use autoscale rules
- Stress-test VM instances and trigger autoscale rules
- Autoscale back in as demand is reduced

For more examples of virtual machine scale sets in action, see the following sample Azure PowerShell sample scripts:

[Scale set script samples for Azure PowerShell](#)

Tutorial: Automatically scale a virtual machine scale set with an Azure template

1/19/2020 • 8 minutes to read • [Edit Online](#)

When you create a scale set, you define the number of VM instances that you wish to run. As your application demand changes, you can automatically increase or decrease the number of VM instances. The ability to autoscale lets you keep up with customer demand or respond to application performance changes throughout the lifecycle of your app. In this tutorial you learn how to:

- Use autoscale with a scale set
- Create and use autoscale rules
- Stress-test VM instances and trigger autoscale rules
- Autoscale back in as demand is reduced

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.29 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Define an autoscale profile

You define an autoscale profile in an Azure template with the `Microsoft.insights/autoscalesettings` resource provider. A *profile* details the capacity of the scale set, and any associated rules. The following example defines a profile named *Autoscale by percentage based on CPU usage* and sets the default, and minimum, capacity of 2 VM instances, and a maximum of 10:

```
{  
  "type": "Microsoft.insights/autoscalesettings",  
  "name": "Autoscale",  
  "apiVersion": "2015-04-01",  
  "location": "[variables('location')]",  
  "scale": null,  
  "properties": {  
    "profiles": [  
      {  
        "name": "Autoscale by percentage based on CPU usage",  
        "capacity": {  
          "minimum": "2",  
          "maximum": "10",  
          "default": "2"  
        }  
      }  
    ]  
  }  
}
```

Define a rule to autoscale out

If your application demand increases, the load on the VM instances in your scale set increases. If this increased load is consistent, rather than just a brief demand, you can configure autoscale rules to increase the number of VM instances in the scale set. When these VM instances are created and your applications are deployed, the scale set starts to distribute traffic to them through the load balancer. You control what metrics to monitor, such as CPU or disk, how long the application load must meet a given threshold, and how many VM instances to add to the scale set.

In the following example, a rule is defined that increases the number of VM instances in a scale set when the average CPU load is greater than 70% over a 5-minute period. When the rule triggers, the number of VM instances is increased by three.

The following parameters are used for this rule:

PARAMETER	EXPLANATION	VALUE
<code>metricName</code>	The performance metric to monitor and apply scale set actions on.	Percentage CPU
<code>timeGrain</code>	How often the metrics are collected for analysis.	1 minute
<code>timeAggregation</code>	Defines how the collected metrics should be aggregated for analysis.	Average
<code>timeWindow</code>	The amount of time monitored before the metric and threshold values are compared.	5 minutes
<code>operator</code>	Operator used to compare the metric data against the threshold.	Greater Than

PARAMETER	EXPLANATION	VALUE
<i>threshold</i>	The value that causes the autoscale rule to trigger an action.	70%
<i>direction</i>	Defines if the scale set should scale in or out when the rule applies.	Increase
<i>type</i>	Indicates that the number of VM instances should be changed by a specific value.	Change Count
<i>value</i>	How many VM instances should be scaled in or out when the rule applies.	3
<i>cooldown</i>	The amount of time to wait before the rule is applied again so that the autoscale actions have time to take effect.	5 minutes

The following rule would be added into the profile section of the *Microsoft.insights/autoscalesettings* resource provider from the previous section:

```

"rules": [
  {
    "metricTrigger": {
      "metricName": "Percentage CPU",
      "metricNamespace": "",
      "metricResourceUri": "[concat('/subscriptions/',subscription().subscriptionId, '/resourceGroups/', resourceGroup().name, '/providers/Microsoft.Compute/virtualMachineScaleSets/', variables('vmssName'))]",
      "metricResourceLocation": "[variables('location')]",
      "timeGrain": "PT1M",
      "statistic": "Average",
      "timeWindow": "PT5M",
      "timeAggregation": "Average",
      "operator": "GreaterThan",
      "threshold": 70
    },
    "scaleAction": {
      "direction": "Increase",
      "type": "ChangeCount",
      "value": "3",
      "cooldown": "PT5M"
    }
  }
]
  
```

Define a rule to autoscale in

On an evening or weekend, your application demand may decrease. If this decreased load is consistent over a period of time, you can configure autoscale rules to decrease the number of VM instances in the scale set. This scale-in action reduces the cost to run your scale set as you only run the number of instances required to meet the current demand.

The following example defines a rule to scale in the number of VM instances by one when the average CPU load then drops below 30% over a 5-minute period. This rule would be added to the autoscale profile after the previous rule to scale out:

```
{
  "metricTrigger": {
    "metricName": "Percentage CPU",
    "metricNamespace": "",
    "metricResourceUri": "[concat('/subscriptions/',subscription().subscriptionId, '/resourceGroups/', resourceGroup().name, '/providers/Microsoft.Compute/virtualMachineScaleSets/', variables('vmssName'))]",
    "metricResourceLocation": "[variables('location')]",
    "timeGrain": "PT1M",
    "statistic": "Average",
    "timeWindow": "PT5M",
    "timeAggregation": "Average",
    "operator": "LessThan",
    "threshold": 30
  },
  "scaleAction": {
    "direction": "Decrease",
    "type": "ChangeCount",
    "value": "1",
    "cooldown": "PT5M"
  }
}
```

Create an autoscaling scale set

Let's use a sample template to create a scale set and apply autoscale rules. You can [review the complete template](#), or [see the *Microsoft.insights/autoscalesettings* resource provider section](#) of the template.

First, create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

```
az group create --name myResourceGroup --location eastus
```

Now create a virtual machine scale set with [az group deployment create](#). When prompted, provide your own username, such as *azureuser*, and password that is used as the credentials for each VM instance:

```
az group deployment create \
  --resource-group myResourceGroup \
  --template-uri https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/scale_sets/autoscale.json
```

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

Generate CPU load on scale set

To test the autoscale rules, generate some CPU load on the VM instances in the scale set. This simulated CPU load causes the autoscale rules to scale out and increase the number of VM instances. As the simulated CPU load is then decreased, the autoscale rules scale in and reduce the number of VM instances.

First, list the address and ports to connect to VM instances in a scale set with [az vmss list-instance-connection-info](#):

```
az vmss list-instance-connection-info \
  --resource-group myResourceGroup \
  --name myScaleSet
```

The following example output shows the instance name, public IP address of the load balancer, and port number that the Network Address Translation (NAT) rules forward traffic to:

```
{  
    "instance 1": "13.92.224.66:50001",  
    "instance 3": "13.92.224.66:50003"  
}
```

SSH to your first VM instance. Specify your own public IP address and port number with the `-p` parameter, as shown from the preceding command:

```
ssh azureuser@13.92.224.66 -p 50001
```

Once logged in, install the **stress** utility. Start 10 **stress** workers that generate CPU load. These workers run for 420 seconds, which is enough to cause the autoscale rules to implement the desired action.

```
sudo apt-get -y install stress  
sudo stress --cpu 10 --timeout 420 &
```

When **stress** shows output similar to *stress: info: [2688] dispatching hogs: 10 cpu, 0 io, 0 vm, 0 hdd*, press the *Enter* key to return to the prompt.

To confirm that **stress** generates CPU load, examine the active system load with the **top** utility:

```
top
```

Exit **top**, then close your connection to the VM instance. **stress** continues to run on the VM instance.

```
Ctrl-c  
exit
```

Connect to second VM instance with the port number listed from the previous [az vmss list-instance-connection-info](#):

```
ssh azureuser@13.92.224.66 -p 50003
```

Install and run **stress**, then start ten workers on this second VM instance.

```
sudo apt-get -y install stress  
sudo stress --cpu 10 --timeout 420 &
```

Again, when **stress** shows output similar to *stress: info: [2713] dispatching hogs: 10 cpu, 0 io, 0 vm, 0 hdd*, press the *Enter* key to return to the prompt.

Close your connection to the second VM instance. **stress** continues to run on the VM instance.

```
exit
```

Monitor the active autoscale rules

To monitor the number of VM instances in your scale set, use **watch**. It takes 5 minutes for the autoscale rules to begin the scale out process in response to the CPU load generated by **stress** on each of the VM instances:

```
watch az vmss list-instances \
--resource-group myResourceGroup \
--name myScaleSet \
--output table
```

Once the CPU threshold has been met, the autoscale rules increase the number of VM instances in the scale set. The following output shows three VMs created as the scale set autoscales out:

```
Every 2.0s: az vmss list-instances --resource-group myResourceGroup --name myScaleSet --output table
```

InstanceId	LatestModelApplied	Location	Name	ProvisioningState	ResourceGroup	VmId
1	True	eastus	myScaleSet_1	Succeeded	MYRESOURCEGROUP	4f92f350-2b68-464f-8a01-e5e590557955
2	True	eastus	myScaleSet_2	Succeeded	MYRESOURCEGROUP	d734cd3d-fb38-4302-817c-cfe35655d48e
4	True	eastus	myScaleSet_4	Creating	MYRESOURCEGROUP	061b4c90-0d73-49fc-a066-19eab0b3d95c
5	True	eastus	myScaleSet_5	Creating	MYRESOURCEGROUP	4beff8b9-4e65-40cb-9652-43899309da27
6	True	eastus	myScaleSet_6	Creating	MYRESOURCEGROUP	9e4133dd-2c57-490e-ae45-90513ce3b336

Once **stress** stops on the initial VM instances, the average CPU load returns to normal. After another 5 minutes, the autoscale rules then scale in the number of VM instances. Scale in actions remove VM instances with the highest IDs first. When a scale set uses Availability Sets or Availability Zones, scale in actions are evenly distributed across those VM instances. The following example output shows one VM instance deleted as the scale set autoscales in:

```
6 True eastus myScaleSet_6 Deleting MYRESOURCEGROUP 9e4133dd-2c57-490e-ae45-90513ce3b336
```

Exit *watch* with `ctrl-c`. The scale set continues to scale in every 5 minutes and remove one VM instance until the minimum instance count of two is reached.

Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with [az group delete](#):

```
az group delete --name myResourceGroup --yes --no-wait
```

Next steps

In this tutorial, you learned how to automatically scale in or out a scale set with the Azure CLI:

- Use autoscale with a scale set
- Create and use autoscale rules
- Stress-test VM instances and trigger autoscale rules
- Autoscale back in as demand is reduced

For more examples of virtual machine scale sets in action, see the following sample Azure CLI sample scripts:

[Scale set script samples for Azure CLI](#)

Azure CLI samples for virtual machine scale sets

1/19/2020 • 2 minutes to read • [Edit Online](#)

The following table includes links to bash scripts built using the Azure CLI.

Create and manage a scale set	
Create a virtual machine scale set	Creates a virtual machine scale set with minimal configuration.
Create a scale set from a custom VM image	Creates a virtual machine scale set that uses a custom VM image.
Install applications to a scale set	Use the Azure Custom Script Extension to install a basic web application into a scale set.
Manage storage	
Create and attach disks to a scale set	Creates a virtual machine scale set with attached data disks.
Manage scale and redundancy	
Enable host-based autoscale	Creates a virtual machine scale that is configured to automatically scale based on CPU usage.
Create a single-zone scale set	Creates a virtual machine scale that uses a single Availability Zone.
Create a zone-redundant scale set	Creates a virtual machine scale across multiple Availability Zones.

Azure PowerShell samples for virtual machine scale sets.

1/19/2020 • 2 minutes to read • [Edit Online](#)

The following table includes links to bash scripts built using Azure PowerShell.

Create and manage a scale set	
Create a simple virtual machine scale set	Creates a virtual machine scale set with minimal configuration.
Create a complete virtual machine scale set	Creates a virtual machine scale set and associated resources with a configuration file.
Create a scale set from a custom VM image	Creates a virtual machine scale set that uses a custom VM image.
Install applications to a scale set	Use the Azure Custom Script Extension to install a basic web application into a scale set.
Manage storage	
Create and attach disks to a scale set	Creates a virtual machine scale set with attached data disks.
Manage scale and redundancy	
Enable host-based autoscale	Creates a virtual machine scale that is configured to automatically scale based on CPU usage.
Create a single-zone scale set	Creates a virtual machine scale that uses a single Availability Zone.
Create a zone-redundant scale set	Creates a virtual machine scale across multiple Availability Zones.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Azure Resource Manager overview

12/23/2019 • 5 minutes to read • [Edit Online](#)

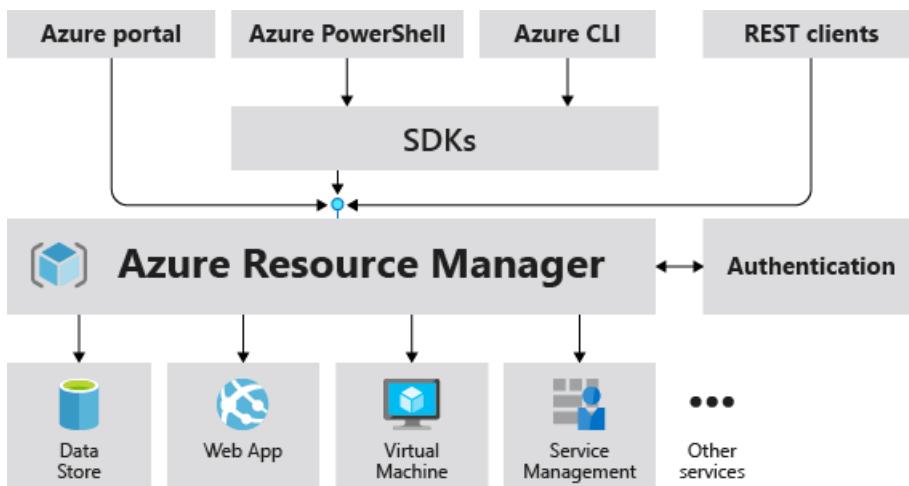
Azure Resource Manager is the deployment and management service for Azure. It provides a management layer that enables you to create, update, and delete resources in your Azure subscription. You use management features, like access control, locks, and tags, to secure and organize your resources after deployment.

To learn about Azure Resource Manager templates, see [Template deployment overview](#).

Consistent management layer

When a user sends a request from any of the Azure tools, APIs, or SDKs, Resource Manager receives the request. It authenticates and authorizes the request. Resource Manager sends the request to the Azure service, which takes the requested action. Because all requests are handled through the same API, you see consistent results and capabilities in all the different tools.

The following image shows the role Azure Resource Manager plays in handling Azure requests.



All capabilities that are available in the portal are also available through PowerShell, Azure CLI, REST APIs, and client SDKs. Functionality initially released through APIs will be represented in the portal within 180 days of initial release.

Terminology

If you're new to Azure Resource Manager, there are some terms you might not be familiar with.

- **resource** - A manageable item that is available through Azure. Virtual machines, storage accounts, web apps, databases, and virtual networks are examples of resources.
- **resource group** - A container that holds related resources for an Azure solution. The resource group includes those resources that you want to manage as a group. You decide which resources belong in a resource group based on what makes the most sense for your organization. See [Resource groups](#).
- **resource provider** - A service that supplies Azure resources. For example, a common resource provider is Microsoft.Compute, which supplies the virtual machine resource. Microsoft.Storage is another common resource provider. See [Resource providers and types](#).
- **Resource Manager template** - A JavaScript Object Notation (JSON) file that defines one or more resources to deploy to a resource group or subscription. The template can be used to deploy the resources consistently and repeatedly. See [Template deployment overview](#).

- **declarative syntax** - Syntax that lets you state "Here is what I intend to create" without having to write the sequence of programming commands to create it. The Resource Manager template is an example of declarative syntax. In the file, you define the properties for the infrastructure to deploy to Azure. See [Template deployment overview](#).

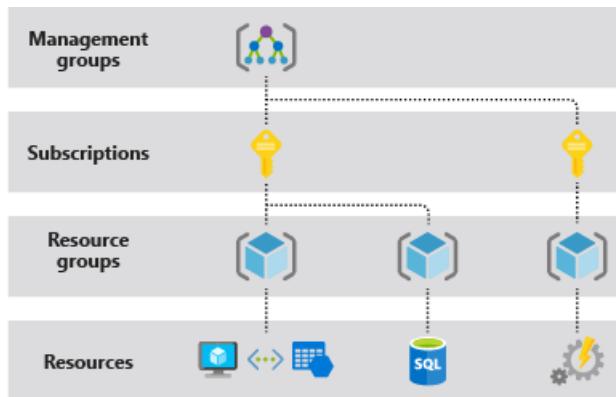
The benefits of using Resource Manager

With Resource Manager, you can:

- Manage your infrastructure through declarative templates rather than scripts.
- Deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.
- Redeploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.
- Define the dependencies between resources so they're deployed in the correct order.
- Apply access control to all services in your resource group because Role-Based Access Control (RBAC) is natively integrated into the management platform.
- Apply tags to resources to logically organize all the resources in your subscription.
- Clarify your organization's billing by viewing costs for a group of resources sharing the same tag.

Understand scope

Azure provides four levels of scope: [management groups](#), subscriptions, [resource groups](#), and resources. The following image shows an example of these layers.



You apply management settings at any of these levels of scope. The level you select determines how widely the setting is applied. Lower levels inherit settings from higher levels. For example, when you apply a [policy](#) to the subscription, the policy is applied to all resource groups and resources in your subscription. When you apply a policy on the resource group, that policy is applied to the resource group and all its resources. However, another resource group doesn't have that policy assignment.

You can deploy templates to management groups, subscriptions, or resource groups.

Resource groups

There are some important factors to consider when defining your resource group:

- All the resources in your group should share the same lifecycle. You deploy, update, and delete them together. If one resource, such as a database server, needs to exist on a different deployment cycle it should be in another resource group.

- Each resource can only exist in one resource group.
- You can add or remove a resource to a resource group at any time.
- You can move a resource from one resource group to another group. For more information, see [Move resources to new resource group or subscription](#).
- A resource group can contain resources that are located in different regions.
- A resource group can be used to scope access control for administrative actions.
- A resource can interact with resources in other resource groups. This interaction is common when the two resources are related but don't share the same lifecycle (for example, web apps connecting to a database).

When creating a resource group, you need to provide a location for that resource group. You may be wondering, "Why does a resource group need a location? And, if the resources can have different locations than the resource group, why does the resource group location matter at all?" The resource group stores metadata about the resources. When you specify a location for the resource group, you're specifying where that metadata is stored. For compliance reasons, you may need to ensure that your data is stored in a particular region.

If the resource group's region is temporarily unavailable, you can't update resources in the resource group because the metadata is unavailable. The resources in other regions will still function as expected, but you can't update them. For more information about building reliable applications, see [Designing reliable Azure applications](#).

Resiliency of Azure Resource Manager

The Azure Resource Manager service is designed for resiliency and continuous availability. Resource Manager and control plane operations (requests sent to management.azure.com) in the REST API are:

- Distributed across regions. Some services are regional.
- Distributed across Availability Zones (as well regions) in locations that have multiple Availability Zones.
- Not dependent on a single logical data center.
- Never taken down for maintenance activities.

This resiliency applies to services that receive requests through Resource Manager. For example, Key Vault benefits from this resiliency.

Next steps

- For all the operations offered by resource providers, see the [Azure REST APIs](#).
- To learn about moving resources, see [Move resources to new resource group or subscription](#).
- To learn about tagging resources, see [Use tags to organize your Azure resources](#).
- To learn about locking resources, see [Lock resources to prevent unexpected changes](#).
- For information about creating templates for deployments, see [Template deployment overview](#).

Regions for virtual machines in Azure

1/19/2020 • 4 minutes to read • [Edit Online](#)

It is important to understand how and where your virtual machines (VMs) operate in Azure, along with your options to maximize performance, availability, and redundancy. This article provides you with an overview of the availability and redundancy features of Azure.

What are Azure regions?

Azure operates in multiple datacenters around the world. These datacenters are grouped in to geographic regions, giving you flexibility in choosing where to build your applications.

You create Azure resources in defined geographic regions like 'West US', 'North Europe', or 'Southeast Asia'. You can review the [list of regions and their locations](#). Within each region, multiple datacenters exist to provide for redundancy and availability. This approach gives you flexibility as you design applications to create VMs closest to your users and to meet any legal, compliance, or tax purposes.

Special Azure regions

Azure has some special regions that you may wish to use when building out your applications for compliance or legal purposes. These special regions include:

- **US Gov Virginia and US Gov Iowa**

- A physical and logical network-isolated instance of Azure for US government agencies and partners, operated by screened US persons. Includes additional compliance certifications such as [FedRAMP](#) and [DISA](#). Read more about [Azure Government](#).

- **China East and China North**

- These regions are available through a unique partnership between Microsoft and 21Vianet, whereby Microsoft does not directly maintain the datacenters. See more about [Azure China 21Vianet](#).

- **Germany Central and Germany Northeast**

- These regions are available via a data trustee model whereby customer data remains in Germany under control of T-Systems, a Deutsche Telekom company, acting as the German data trustee.

Region pairs

Each Azure region is paired with another region within the same geography (such as US, Europe, or Asia). This approach allows for the replication of resources, such as VM storage, across a geography that should reduce the likelihood of natural disasters, civil unrest, power outages, or physical network outages affecting both regions at once. Additional advantages of region pairs include:

- In the event of a wider Azure outage, one region is prioritized out of every pair to help reduce the time to restore for applications.
- Planned Azure updates are rolled out to paired regions one at a time to minimize downtime and risk of application outage.
- Data continues to reside within the same geography as its pair (except for Brazil South) for tax and law enforcement jurisdiction purposes.

Examples of region pairs include:

PRIMARY	SECONDARY
West US	East US
North Europe	West Europe
Southeast Asia	East Asia

You can see the full [list of regional pairs here](#).

Feature availability

Some services or VM features are only available in certain regions, such as specific VM sizes or storage types. There are also some global Azure services that do not require you to select a particular region, such as [Azure Active Directory](#), [Traffic Manager](#), or [Azure DNS](#). To assist you in designing your application environment, you can check the [availability of Azure services across each region](#). You can also [programmatically query the supported VM sizes and restrictions in each region](#).

Storage availability

Understanding Azure regions and geographies becomes important when you consider the available storage replication options. Depending on the storage type, you have different replication options.

Azure Managed Disks

- Locally redundant storage (LRS)
 - Replicates your data three times within the region in which you created your storage account.

Storage account-based disks

- Locally redundant storage (LRS)
 - Replicates your data three times within the region in which you created your storage account.
- Zone redundant storage (ZRS)
 - Replicates your data three times across two to three facilities, either within a single region or across two regions.
- Geo-redundant storage (GRS)
 - Replicates your data to a secondary region that is hundreds of miles away from the primary region.
- Read-access geo-redundant storage (RA-GRS)
 - Replicates your data to a secondary region, as with GRS, but also then provides read-only access to the data in the secondary location.

The following table provides a quick overview of the differences between the storage replication types:

REPLICATION STRATEGY	LRS	ZRS	GRS	RA-GRS
Data is replicated across multiple facilities.	No	Yes	Yes	Yes
Data can be read from the secondary location and from the primary location.	No	No	No	Yes

REPLICATION STRATEGY	LRS	ZRS	GRS	RA-GRS
Number of copies of data maintained on separate nodes.	3	3	6	6

You can read more about [Azure Storage replication options here](#). For more information about managed disks, see [Azure Managed Disks overview](#).

Storage costs

Prices vary depending on the storage type and availability that you select.

Azure Managed Disks

- Premium Managed Disks are backed by Solid-State Drives (SSDs) and Standard Managed Disks are backed by regular spinning disks. Both Premium and Standard Managed Disks are charged based on the provisioned capacity for the disk.

Unmanaged disks

- Premium storage is backed by Solid-State Drives (SSDs) and is charged based on the capacity of the disk.
- Standard storage is backed by regular spinning disks and is charged based on the in-use capacity and desired storage availability.
 - For RA-GRS, there is an additional Geo-Replication Data Transfer charge for the bandwidth of replicating that data to another Azure region.

See [Azure Storage Pricing](#) for pricing information on the different storage types and availability options.

Availability options for virtual machines in Azure

1/19/2020 • 5 minutes to read • [Edit Online](#)

This article provides you with an overview of the availability features of Azure virtual machines (VMs).

High availability

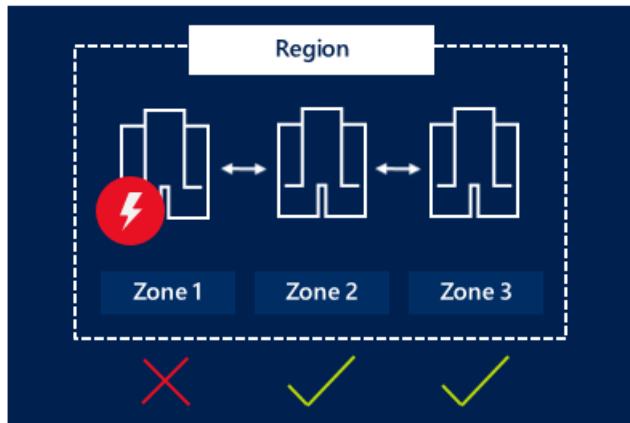
Workloads are typically spread across different virtual machines to gain high throughput, performance, and to create redundancy in case a VM is impacted due to an update or other event.

There are few options that Azure provides to achieve High Availability. First let's talk about basic constructs.

Availability zones

[Availability zones](#) expand the level of control you have to maintain the availability of the applications and data on your VMs. An Availability Zone is a physically separate zone, within an Azure region. There are three Availability Zones per supported Azure region.

Each Availability Zone has a distinct power source, network, and cooling. By architecting your solutions to use replicated VMs in zones, you can protect your apps and data from the loss of a datacenter. If one zone is compromised, then replicated apps and data are instantly available in another zone.



Learn more about deploying a [Windows](#) or [Linux](#) VM in an Availability Zone.

Fault domains

A fault domain is a logical group of underlying hardware that share a common power source and network switch, similar to a rack within an on-premises datacenter.

Update domains

An update domain is a logical group of underlying hardware that can undergo maintenance or be rebooted at the same time.

This approach ensures that at least one instance of your application always remains running as the Azure platform undergoes periodic maintenance. The order of update domains being rebooted may not proceed sequentially during maintenance, but only one update domain is rebooted at a time.

Virtual Machines Scale Sets

Azure virtual machine scale sets let you create and manage a group of load balanced VMs. The number of VM instances can automatically increase or decrease in response to demand or a defined schedule. Scale sets provide high availability to your applications, and allow you to centrally manage, configure, and update many VMs. We

recommended that two or more VMs are created within a scale set to provide for a highly available application and to meet the [99.95% Azure SLA](#). There is no cost for the scale set itself, you only pay for each VM instance that you create. When a single VM is using [Azure premium SSDs](#), the Azure SLA applies for unplanned maintenance events. Virtual machines in a scale set can be deployed across multiple update domains and fault domains to maximize availability and resilience to outages due to data center outages, and planned or unplanned maintenance events. Virtual machines in a scale set can also be deployed into a single Availability zone, or regionally. Availability zone deployment options may differ based on the orchestration mode.

Preview: Orchestration mode Preview

Virtual machines scale sets allow you to specify orchestration mode. With the virtual machine scale set orchestration mode (preview), you can now choose whether the scale set should orchestrate virtual machines which are created explicitly outside of a scale set configuration model, or virtual machine instances created implicitly based on the configuration model. Choose the orchestration mode that VM orchestration model allows you group explicitly defined Virtual Machines together in a region or in an availability zone. Virtual machines deployed in an Availability Zone provides zonal isolation to VMs as they are bound to the availability zone boundary and are not subjected to any failures that may occur in other availability zone in the region.

	"ORCHESTRATIONMODE": "VM" (VIRTUALMACHINE)	"ORCHESTRATIONMODE": "SCALESETVM" (VIRTUALMACHINESCALESET VM)
VM configuration model	None. VirtualMachineProfile is undefined in the scale set model.	Required. VirtualMachineProfile is populated in the scale set model.
Adding new VM to Scale Set	VMs are explicitly added to the scale set when the VM is created.	VMs are implicitly created and added to the scale set based on the VM configuration model, instance count, and AutoScaling rules.
Availability Zones	Supports regional deployment or VMs in one Availability Zone	Supports regional deployment or multiple Availability Zones; Can define the zone balancing strategy
Fault domains	Can define fault domains count. 2 or 3 based on regional support and 5 for Availability zone. The assigned VM fault domain will persist with VM lifecycle, including deallocate and restart.	Can define 1, 2, or 3 fault domains for non-zonal deployments, and 5 for Availability zone deployments. The assigned VM fault domain does not persist with VM lifecycle, virtual machines are assigned a fault domain at time of allocation.
Update domains	N/A. Update domains are automatically mapped to fault domains	N/A. Update domains are automatically mapped to fault domains

Fault domains and update domains

Virtual machine scale sets simplify designing for high availability by aligning fault domains and update domains. You will only have to define fault domains count for the scale set. The number of fault domains available to the scale sets may vary by region. See [Manage the availability of virtual machines in Azure](#).

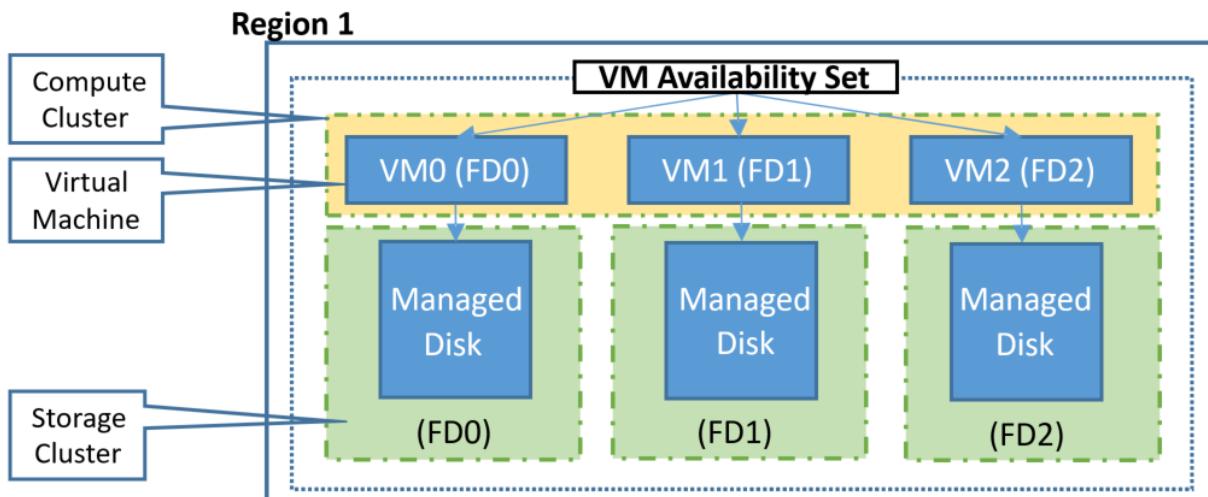
Availability sets

An availability set is a logical grouping of VMs within a datacenter that allows Azure to understand how your application is built to provide for redundancy and availability. We recommended that two or more VMs are created within an availability set to provide for a highly available application and to meet the [99.95% Azure SLA](#). There is no cost for the Availability Set itself, you only pay for each VM instance that you create. When a single VM is using [Azure premium SSDs](#), the Azure SLA applies for unplanned maintenance events.

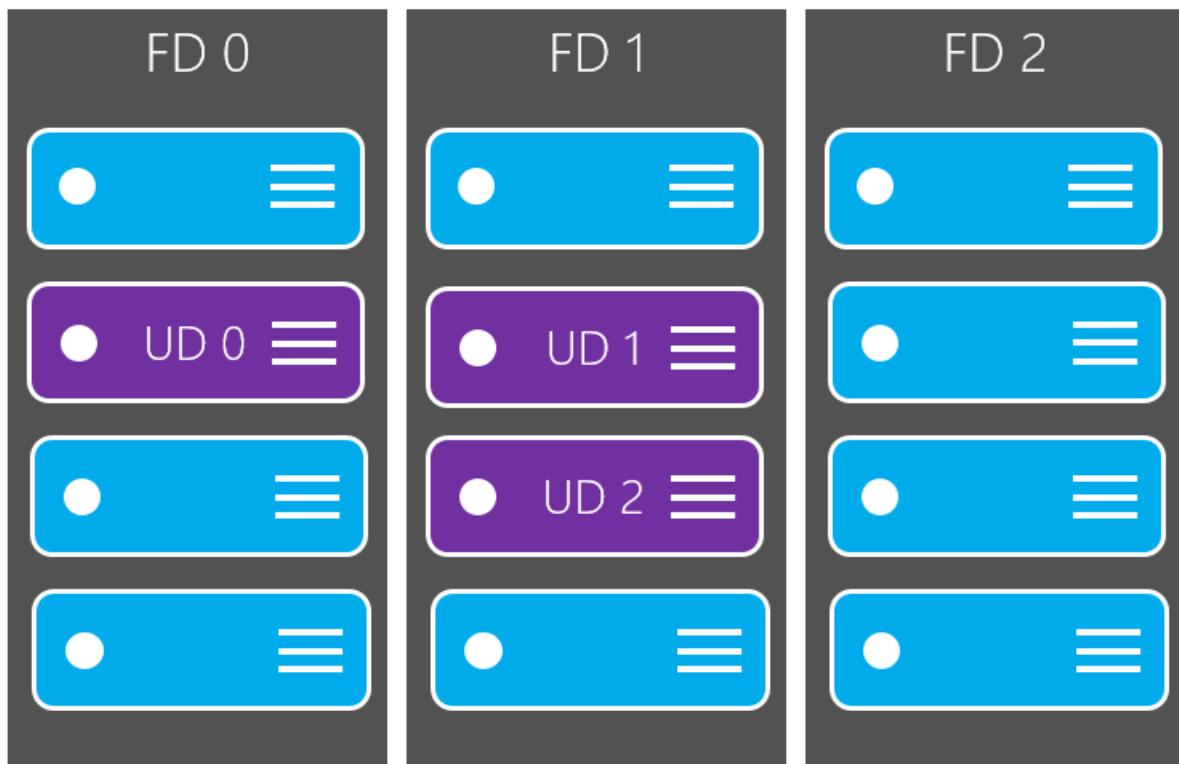
In an availability set, VMs are automatically distributed across these fault domains. This approach limits the impact of potential physical hardware failures, network outages, or power interruptions.

For VMs using [Azure Managed Disks](#), VMs are aligned with managed disk fault domains when using a managed availability set. This alignment ensures that all the managed disks attached to a VM are within the same managed disk fault domain.

Only VMs with managed disks can be created in a managed availability set. The number of managed disk fault domains varies by region - either two or three managed disk fault domains per region. You can read more about these managed disk fault domains for [Linux VMs](#) or [Windows VMs](#).



VMs within an availability set are also automatically distributed across update domains.



Next steps

You can now start to use these availability and redundancy features to build your Azure environment. For best practices information, see [Azure availability best practices](#).

Co-location

1/19/2020 • 3 minutes to read • [Edit Online](#)

One of the largest contributors to latency between VMs is simply distance.

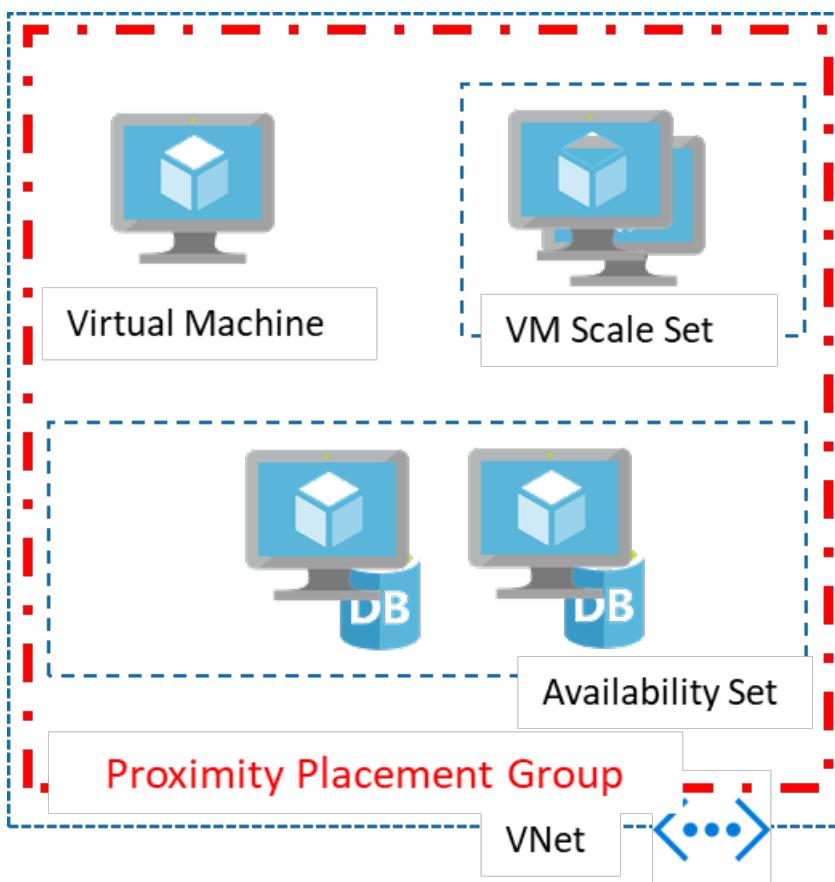
Preview: Proximity placement groups

Placing VMs in a single region reduces the physical distance between the instances. Placing them within a single availability zone will also bring them physically closer together. However, as the Azure footprint grows, a single availability zone may span multiple physical data centers, which may result in a network latency impacting your application.

To get VMs as close as possible, achieving the lowest possible latency, you should deploy them within a proximity placement group.

A proximity placement group is a logical grouping used to make sure that Azure compute resources are physically located close to each other. Proximity placement groups are useful for workloads where low latency is a requirement.

- Low latency between stand-alone VMs.
- Low Latency between VMs in a single availability set or a virtual machine scale set.
- Low latency between stand-alone VMs, VMs in multiple Availability Sets, or multiple scale sets. You can have multiple compute resources in a single placement group to bring together a multi-tiered application.
- Low latency between multiple application tiers using different hardware types. For example, running the backend using M-series in an availability set and the front end on a D-series instance, in a scale set, in a single proximity placement group.



Using Proximity Placement Groups

A proximity placement group is a new resource type in Azure. You need to create one before using it with other resources. Once created, it could be used with virtual machines, availability sets, or virtual machine scale sets. You specify a proximity placement group when creating compute resources providing the proximity placement group ID.

You can also move an existing resource into a proximity placement group. When moving a resource into a proximity placement group, you should stop (deallocate) the asset first since it will be redeployed potentially into a different data center in the region so satisfy the colocation constraint.

In the case of availability sets and virtual machine scale sets, you should set the proximity placement group at the resource level rather than the individual virtual machines.

A proximity placement group is a colocation constraint rather than a pinning mechanism. It is pinned to a specific data center with the deployment of the first resource to use it. Once all resources using the proximity placement group have been stopped (deallocated) or deleted, it is no longer pinned. Therefore, when using a proximity placement group with multiple VM series, it is important to specify all the required types upfront in a template when possible or follow a deployment sequence which will improve your chances for a successful deployment. If your deployment fails, restart the deployment with the VM size which has failed as the first size to be deployed.

Best practices

- For the lowest latency, use proximity placement groups together with accelerated networking. For more information, see [Create a Linux virtual machine with Accelerated Networking](#) or [Create a Windows virtual machine with Accelerated Networking](#).
- Deploy all VM sizes in a single template. In order to avoid landing on hardware that doesn't support all the VM SKUs and sizes you require, include all of the application tiers in a single template so that they will all be deployed at the same time.
- If you are scripting your deployment using PowerShell, CLI or the SDK, you may get an allocation error `OverconstrainedAllocationRequest`. In this case, you should stop/deallocate all the existing VMs, and change the sequence in the deployment script to begin with the VM SKU/sizes that failed.
- When reusing an existing placement group from which VMs were deleted, wait for the deletion to fully complete before adding VMs to it.
- If latency is your first priority, put VMs in a proximity placement group and the entire solution in an availability zone. But, if resiliency is your top priority, spread your instances across multiple availability zones (a single proximity placement group cannot span zones).

Next steps

Create a [proximity placement group](#) for your scale-set.

Optimize network throughput for Azure virtual machines

1/6/2020 • 3 minutes to read • [Edit Online](#)

Azure virtual machines (VM) have default network settings that can be further optimized for network throughput. This article describes how to optimize network throughput for Microsoft Azure Windows and Linux VMs, including major distributions such as Ubuntu, CentOS, and Red Hat.

Windows VM

If your Windows VM supports [Accelerated Networking](#), enabling that feature would be the optimal configuration for throughput. For all other Windows VMs, using Receive Side Scaling (RSS) can reach higher maximal throughput than a VM without RSS. RSS may be disabled by default in a Windows VM. To determine whether RSS is enabled, and enable it if it's currently disabled, complete the following steps:

1. See if RSS is enabled for a network adapter with the `Get-NetAdapterRss` PowerShell command. In the following example output returned from the `Get-NetAdapterRss`, RSS is not enabled.

```
Name      : Ethernet
InterfaceDescription : Microsoft Hyper-V Network Adapter
Enabled    : False
```

2. To enable RSS, enter the following command:

```
Get-NetAdapter | % {Enable-NetAdapterRss -Name $_.Name}
```

The previous command does not have an output. The command changed NIC settings, causing temporary connectivity loss for about one minute. A Reconnecting dialog box appears during the connectivity loss. Connectivity is typically restored after the third attempt.

3. Confirm that RSS is enabled in the VM by entering the `Get-NetAdapterRss` command again. If successful, the following example output is returned:

```
Name      : Ethernet
InterfaceDescription : Microsoft Hyper-V Network Adapter
Enabled    : True
```

Linux VM

RSS is always enabled by default in an Azure Linux VM. Linux kernels released since October 2017 include new network optimizations options that enable a Linux VM to achieve higher network throughput.

Ubuntu for new deployments

The Ubuntu Azure kernel provides the best network performance on Azure and has been the default kernel since September 21, 2017. In order to get this kernel, first install the latest supported version of 16.04-LTS, as follows:

```
"Publisher": "Canonical",
"Offer": "UbuntuServer",
"Sku": "16.04-LTS",
"Version": "latest"
```

After the creation is complete, enter the following commands to get the latest updates. These steps also work for VMs currently running the Ubuntu Azure kernel.

```
#run as root or preface with sudo
apt-get -y update
apt-get -y upgrade
apt-get -y dist-upgrade
```

The following optional command set may be helpful for existing Ubuntu deployments that already have the Azure kernel but that have failed to further updates with errors.

```
#optional steps may be helpful in existing deployments with the Azure kernel
#run as root or preface with sudo
apt-get -f install
apt-get --fix-missing install
apt-get clean
apt-get -y update
apt-get -y upgrade
apt-get -y dist-upgrade
```

Ubuntu Azure kernel upgrade for existing VMs

Significant throughput performance can be achieved by upgrading to the Azure Linux kernel. To verify whether you have this kernel, check your kernel version.

```
#Azure kernel name ends with "-azure"
uname -r

#sample output on Azure kernel:
#4.13.0-1007-azure
```

If your VM does not have the Azure kernel, the version number usually begins with "4.4." If the VM does not have the Azure kernel, run the following commands as root:

```
#run as root or preface with sudo
apt-get update
apt-get upgrade -y
apt-get dist-upgrade -y
apt-get install "linux-azure"
reboot
```

CentOS

In order to get the latest optimizations, it is best to create a VM with the latest supported version by specifying the following parameters:

```
"Publisher": "OpenLogic",
"Offer": "CentOS",
"Sku": "7.4",
"Version": "latest"
```

New and existing VMs can benefit from installing the latest Linux Integration Services (LIS). The throughput

optimization is in LIS, starting from 4.2.2-2, although later versions contain further improvements. Enter the following commands to install the latest LIS:

```
sudo yum update  
sudo reboot  
sudo yum install microsoft-hyper-v
```

Red Hat

In order to get the optimizations, it is best to create a VM with the latest supported version by specifying the following parameters:

```
"Publisher": "RedHat"  
"Offer": "RHEL"  
"Sku": "7-RAW"  
"Version": "latest"
```

New and existing VMs can benefit from installing the latest Linux Integration Services (LIS). The throughput optimization is in LIS, starting from 4.2. Enter the following commands to download and install LIS:

```
wget https://aka.ms/lis  
tar xvf lis  
cd LISISO  
sudo ./install.sh #or upgrade.sh if prior LIS was previously installed
```

Learn more about Linux Integration Services Version 4.2 for Hyper-V by viewing the [download page](#).

Next steps

- See the optimized result with [Bandwidth/Throughput testing Azure VM](#) for your scenario.
- Read about how [bandwidth is allocated to virtual machines](#)
- Learn more with [Azure Virtual Network frequently asked questions \(FAQ\)](#)

Choosing the right number of fault domains for virtual machine scale set

1/19/2020 • 2 minutes to read • [Edit Online](#)

Virtual machine scale sets are created with five fault domains by default in Azure regions with no zones. For the regions that support zonal deployment of virtual machine scale sets and this option is selected, the default value of the fault domain count is 1 for each of the zones. FD=1 in this case implies that the VM instances belonging to the scale set will be spread across many racks on a best effort basis.

You can also consider aligning the number of scale set fault domains with the number of Managed Disks fault domains. This alignment can help prevent loss of quorum if an entire Managed Disks fault domain goes down. The FD count can be set to less than or equal to the number of Managed Disks fault domains available in each of the regions. Refer to this [document](#) to learn about the number of Managed Disks fault domains by region.

REST API

You can set the property `properties.platformFaultDomainCount` to 1, 2, or 3 (default of 5 if not specified). Refer to the documentation for REST API [here](#).

Azure CLI

You can set the parameter `--platform-fault-domain-count` to 1, 2, or 3 (default of 5 if not specified). Refer to the documentation for Azure CLI [here](#).

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--admin-username azureuser \
--platform-fault-domain-count 3 \
--generate-ssh-keys
```

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

Next steps

- Learn more about [availability and redundancy features](#) for Azure environments.

Sizes for Windows virtual machines in Azure

2/25/2020 • 2 minutes to read • [Edit Online](#)

This article describes the available sizes and options for the Azure virtual machines you can use to run your Windows apps and workloads. It also provides deployment considerations to be aware of when you're planning to use these resources. This article is also available for [Linux virtual machines](#).

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, Dasv4, Dav4, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, Easv4, Eav4, Mv2, M, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Lsv2	High disk throughput and IO ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases.
GPU	NC, NCv2, NCv3, ND, NDv2 (Preview), NV, NVv3, NVv4	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	HB, HC, H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

- For information about pricing of the various sizes, see [Virtual Machines Pricing](#).
- To see general limits on Azure VMs, see [Azure subscription and service limits, quotas, and constraints](#).
- Storage costs are calculated separately based on used pages in the storage account. For details, [Azure Storage Pricing](#).
- Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

REST API

For information on using the REST API to query for VM sizes, see the following:

- [List available virtual machine sizes for resizing](#)
- [List available virtual machine sizes for a subscription](#)
- [List available virtual machine sizes in an availability set](#)

ACU

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

Benchmark scores

Learn more about compute performance for Windows VMs using the [CoreMark benchmark scores](#).

Next steps

Learn more about the different VM sizes that are available:

- [General purpose](#)
- [Compute optimized](#)
- [Memory optimized](#)
- [Storage optimized](#)
- [GPU optimized](#)
- [High performance compute](#)
- Check the [Previous generation](#) page for A Standard, Dv1 (D1-4 and D11-14 v1), and A8-A11 series

2 minutes to read

2 minutes to read

2 minutes to read

2 minutes to read

Constrained vCPU capable VM sizes

2/28/2020 • 2 minutes to read • [Edit Online](#)

Some database workloads like SQL Server or Oracle require high memory, storage, and I/O bandwidth, but not a high core count. Many database workloads are not CPU-intensive. Azure offers certain VM sizes where you can constrain the VM vCPU count to reduce the cost of software licensing, while maintaining the same memory, storage, and I/O bandwidth.

The vCPU count can be constrained to one half or one quarter of the original VM size. These new VM sizes have a suffix that specifies the number of active vCPUs to make them easier for you to identify.

For example, the current VM size Standard_GS5 comes with 32 vCPUs, 448 GB RAM, 64 disks (up to 256 TB), and 80,000 IOPs or 2 GB/s of I/O bandwidth. The new VM sizes Standard_GS5-16 and Standard_GS5-8 comes with 16 and 8 active vCPUs respectively, while maintaining the rest of the specs of the Standard_GS5 for memory, storage, and I/O bandwidth.

The licensing fees charged for SQL Server or Oracle are constrained to the new vCPU count, and other products should be charged based on the new vCPU count. This results in a 50% to 75% increase in the ratio of the VM specs to active (billable) vCPUs. These new VM sizes allow customer workloads to use the same memory, storage, and I/O bandwidth while optimizing their software licensing cost. At this time, the compute cost, which includes OS licensing, remains the same one as the original size. For more information, see [Azure VM sizes for more cost-effective database workloads](#).

NAME	VCPU	SPECS
Standard_M8-2ms	2	Same as M8ms
Standard_M8-4ms	4	Same as M8ms
Standard_M16-4ms	4	Same as M16ms
Standard_M16-8ms	8	Same as M16ms
Standard_M32-8ms	8	Same as M32ms
Standard_M32-16ms	16	Same as M32ms
Standard_M64-32ms	32	Same as M64ms
Standard_M64-16ms	16	Same as M64ms
Standard_M128-64ms	64	Same as M128ms
Standard_M128-32ms	32	Same as M128ms
Standard_E4-2s_v3	2	Same as E4s_v3
Standard_E8-4s_v3	4	Same as E8s_v3
Standard_E8-2s_v3	2	Same as E8s_v3

NAME	VCPUs	SPECS
Standard_E16-8s_v3	8	Same as E16s_v3
Standard_E16-4s_v3	4	Same as E16s_v3
Standard_E32-16s_v3	16	Same as E32s_v3
Standard_E32-8s_v3	8	Same as E32s_v3
Standard_E64-32s_v3	32	Same as E64s_v3
Standard_E64-16s_v3	16	Same as E64s_v3
Standard_GS4-8	8	Same as GS4
Standard_GS4-4	4	Same as GS4
Standard_GS5-16	16	Same as GS5
Standard_GS5-8	8	Same as GS5
Standard_DS11-1_v2	1	Same as DS11_v2
Standard_DS12-2_v2	2	Same as DS12_v2
Standard_DS12-1_v2	1	Same as DS12_v2
Standard_DS13-4_v2	4	Same as DS13_v2
Standard_DS13-2_v2	2	Same as DS13_v2
Standard_DS14-8_v2	8	Same as DS14_v2
Standard_DS14-4_v2	4	Same as DS14_v2

Other sizes

- [Compute optimized](#)
- [Memory optimized](#)
- [Storage optimized](#)
- [GPU](#)
- [High performance compute](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

2 minutes to read

2 minutes to read

Install NVIDIA GPU drivers on N-series VMs running Windows

2/28/2020 • 3 minutes to read • [Edit Online](#)

To take advantage of the GPU capabilities of Azure N-series VMs running Windows, NVIDIA GPU drivers must be installed. The [NVIDIA GPU Driver Extension](#) installs appropriate NVIDIA CUDA or GRID drivers on an N-series VM. Install or manage the extension using the Azure portal or tools such as Azure PowerShell or Azure Resource Manager templates. See the [NVIDIA GPU Driver Extension documentation](#) for supported operating systems and deployment steps.

If you choose to install GPU drivers manually, this article provides supported operating systems, drivers, and installation and verification steps. Manual driver setup information is also available for [Linux VMs](#).

For basic specs, storage capacities, and disk details, see [GPU Windows VM sizes](#).

Supported operating systems and drivers

NVIDIA Tesla (CUDA) drivers

NVIDIA Tesla (CUDA) drivers for NC, NCv2, NCv3, ND, and NDv2-series VMs (optional for NV-series) are supported only on the operating systems listed in the following table. Driver download links are current at time of publication. For the latest drivers, visit the [NVIDIA](#) website.

TIP

As an alternative to manual CUDA driver installation on a Windows Server VM, you can deploy an Azure [Data Science Virtual Machine](#) image. The DSVM editions for Windows Server 2016 pre-install NVIDIA CUDA drivers, the CUDA Deep Neural Network Library, and other tools.

OS	Driver
Windows Server 2016	398.75 (.exe)
Windows Server 2012 R2	398.75 (.exe)

NVIDIA GRID drivers

Microsoft redistributes NVIDIA GRID driver installers for NV and NVv3-series VMs used as virtual workstations or for virtual applications. Install only these GRID drivers on Azure NV-series VMs, only on the operating systems listed in the following table. These drivers include licensing for GRID Virtual GPU Software in Azure. You do not need to set up a NVIDIA vGPU software license server.

Please note that the Nvidia extension will always install the latest driver. We provide links to the previous version here for customers, who have dependency on an older version.

For Windows Server 2019, Windows Server 2016, and Windows 10(up to build 1909):

- [GRID 10.1 \(442.06\) \(.exe\)](#)
- [GRID 10.0 \(441.66\) \(.exe\)](#)

For Windows Server 2012 R2, Windows Server 2008 R2, Windows 8, and Windows 7:

- [GRID 10.1 \(442.06\) \(.exe\)](#)
- [GRID 10.0 \(441.66\) \(.exe\)](#)

Driver installation

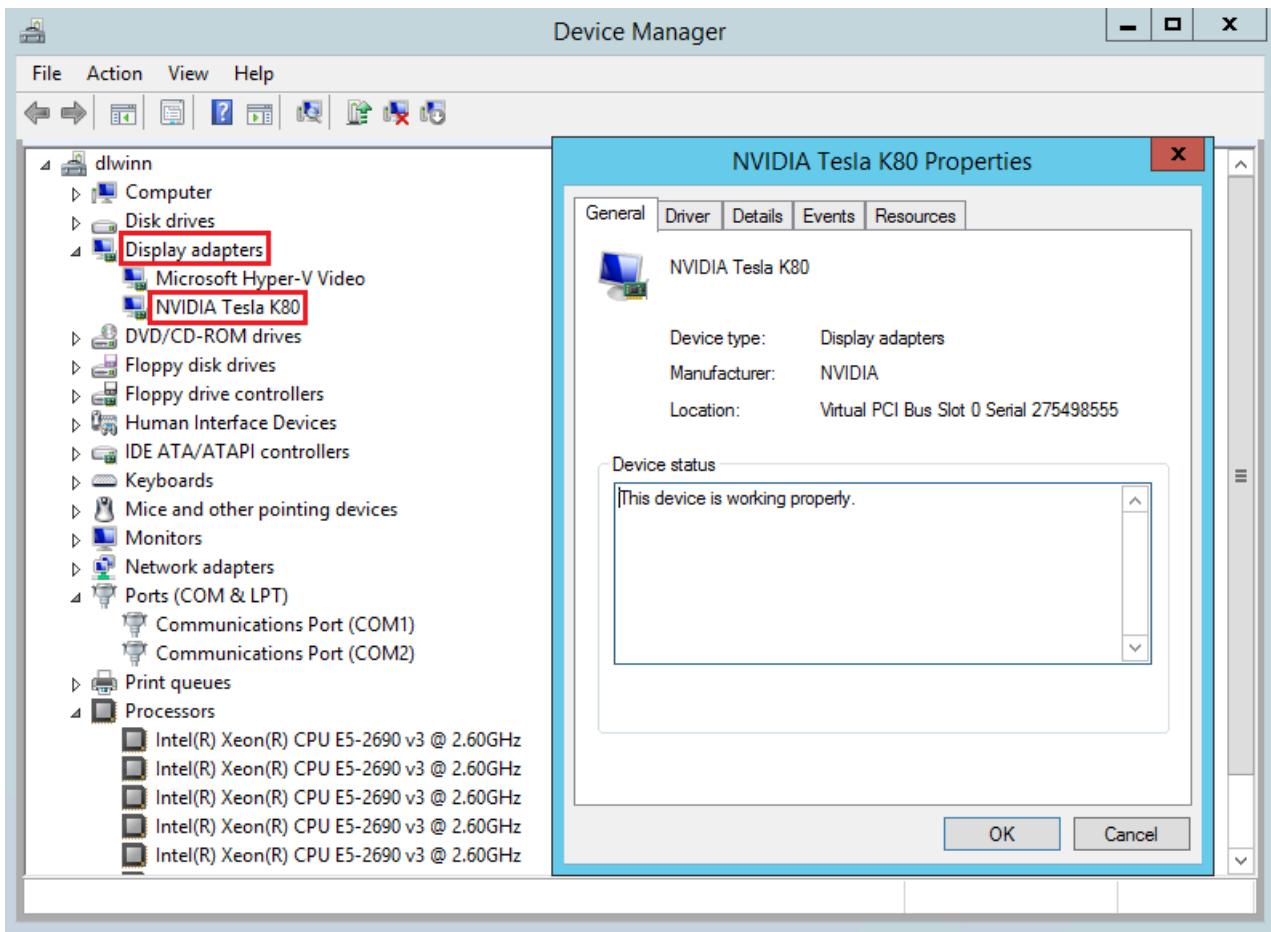
1. Connect by Remote Desktop to each N-series VM.
2. Download, extract, and install the supported driver for your Windows operating system.

After GRID driver installation on a VM, a restart is required. After CUDA driver installation, a restart is not required.

Verify driver installation

Please note that the Nvidia Control panel is only accessible with the GRID driver installation. If you have installed CUDA drivers then the Nvidia control panel will not be visible.

You can verify driver installation in Device Manager. The following example shows successful configuration of the Tesla K80 card on an Azure NC VM.



To query the GPU device state, run the `nvidia-smi` command-line utility installed with the driver.

1. Open a command prompt and change to the **C:\Program Files\NVIDIA Corporation\NVSMI** directory.
2. Run `nvidia-smi`. If the driver is installed, you will see output similar to the following. The **GPU-Util** shows **0%** unless you are currently running a GPU workload on the VM. Your driver version and GPU details may be different from the ones shown.

```
C:\Program Files\NVIDIA Corporation\NVSMI>nvidia-smi
Wed Nov 23 20:49:33 2016
+-----+-----+
| NVIDIA-SMI 369.73 | Driver Version: 369.73 |
+-----+-----+
| GPU  Name    TCC/WDDM | Bus-Id     Disp.A  Volatile Uncorr. ECC |
| Fan  Temp   Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
|-----+-----+
| 0  Tesla K80    TCC | B794:00:00.0 Off    0MiB / 11423MiB |      0% Default |
| N/A   56C   P8    28W / 149W |                |
+-----+-----+
+-----+-----+
| Processes:                               GPU Memory |
| GPU  PID  Type  Process name        Usage          |
|-----+-----+
| No running processes found            |
+-----+-----+
```

RDMA network connectivity

RDMA network connectivity can be enabled on RDMA-capable N-series VMs such as NC24r deployed in the same availability set or in a single placement group in a virtual machine scale set. The HpcVmDrivers extension must be added to install Windows network device drivers that enable RDMA connectivity. To add the VM extension to an RDMA-enabled N-series VM, use [Azure PowerShell](#) cmdlets for Azure Resource Manager.

To install the latest version 1.1 HpcVMDrivers extension on an existing RDMA-capable VM named myVM in the West US region:

```
Set-AzVMExtension -ResourceGroupName "myResourceGroup" -Location "westus" -VMName "myVM" -ExtensionName "HpcVmDrivers" -Publisher "Microsoft.HpcCompute" -Type "HpcVmDrivers" -TypeHandlerVersion "1.1"
```

For more information, see [Virtual machine extensions and features for Windows](#).

The RDMA network supports Message Passing Interface (MPI) traffic for applications running with [Microsoft MPI](#) or Intel MPI 5.x.

Next steps

- Developers building GPU-accelerated applications for the NVIDIA Tesla GPUs can also download and install the latest [CUDA Toolkit](#). For more information, see the [CUDA Installation Guide](#).

2 minutes to read

2 minutes to read

Compute benchmark scores for Windows VMs

2/26/2020 • 17 minutes to read • [Edit Online](#)

The following SPECInt benchmark scores show compute performance for select Azure VMs running Windows Server. Compute benchmark scores are also available for [Linux VMs](#).

Av2 - General Compute

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_A1_v2	1	1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	12	14.2	0.3
Standard_A1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	9	13.2	0.6
Standard_A1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	10	14.1	0.7
Standard_A2_v2	2	1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	14	28.9	0.6
Standard_A2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	10	27.4	1.6
Standard_A2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	17	28.9	1.8
Standard_A2_m_v2	2	1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	14	29.0	0.5
Standard_A2_m_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	11	26.3	0.8
Standard_A2_m_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	21	28.4	1.0

SIZE	VCPUS	NUMA NODES	CPU	RUNS	Avg Base Rate	STDDEV
Standard_A4_v2	4	1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	27	56.6	1.0
Standard_A4_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	13	52.8	2.0
Standard_A4_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	15	52.1	4.5
Standard_A4_m_v2	4	1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	17	56.4	1.8
Standard_A4_m_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	6	53.4	1.9
Standard_A4_m_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	23	57.1	3.6
Standard_A8_v2	8	1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	14	109.1	1.6
Standard_A8_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	6	101.5	2.8
Standard_A8_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	11	101.9	2.7
Standard_A8_m_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	11	101.4	1.2

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_A8m_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	10	104.5	5.1
Standard_A8m_v2	8	2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	13	111.6	2.3

Note: Av2-series VMs can be deployed on a variety of hardware types and processors (as seen above). Av2-series VMs have CPU performance and memory configurations best suited for entry level workloads like development and test. The size is throttled to offer relatively consistent processor performance for the running instance, regardless of the hardware it is deployed on; however, software that takes advantage of specific newer processor optimizations may see more significant variation across processor types.

B - Burstable

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_B1ms	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	9	6.3	0.2
Standard_B1ms	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	47	6.4	0.2
Standard_B2ms	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	36	19.8	0.8
Standard_B2s	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	13.0	0.0
Standard_B2s	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	29	13.0	0.5
Standard_B4ms	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	6	27.1	1.0
Standard_B4ms	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	43	28.3	0.7

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_B8ms	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	3	42.0	0.0
Standard_B8ms	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	25	41.4	0.9

Note: B-Series VMs are for workloads with burstable performance requirements. VM instances accumulate credits when using less than its baseline. When the VM has accumulated credit, the VM can burst above the baseline using up to 100% to meet short CPU burst requirements. Burst time depends on available credits which is a function of VM size and time.

SPEC Int is a fairly long running test that typically exhausts available burst credits. Therefore the numbers above are closer to the baseline performance of the VM (although they may reflect some burst time accumulated between runs). For short, bursty, workloads (typical on B-Series) performance will typically be closer to that of the Ds v3 Series..

DSv3 - General Compute + Premium Storage

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_D2s_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	10	40.8	2.3
Standard_D2s_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	52	43.3	2.1
Standard_D4s_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	21	77.9	2.6
Standard_D4s_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	29	82.3	2.5
Standard_D8s_v3	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	7	148.3	1.9
Standard_D8s_v3	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	28	155.4	5.6

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_D16_s_v3	16	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	3	275.7	5.1
Standard_D16_s_v3	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	38	298.2	4.4
Standard_D32_s_v3	32	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	24	545.8	10.5
Standard_D32_s_v3	32	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	9	535.6	12.6
Standard_D64_s_v3	64	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	35	1070.6	2.4

Dv3 - General Compute

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_D2_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	10	38.6	1.8
Standard_D2_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	24	41.8	3.3
Standard_D4_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	17	77.8	1.3
Standard_D4_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	45	82.7	4.5
Standard_D8_v3	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	9	146.7	10.4

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_D8_v3	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	27	159.9	8.3
Standard_D16_v3	16	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	10	274.1	3.8
Standard_D16_v3	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	300.7	8.8
Standard_D32_v3	32	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	24	549.3	11.1
Standard_D32_v3	32	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	7	538.6	9.4
Standard_D64_v3	64	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	1070.6	12.4

DSv2 - Storage Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_DS1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	12	33.0	1.1
Standard_DS1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	37	33.8	2.5
Standard_DS2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	33	63.9	1.7
Standard_DS2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	66.6	4.8

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_DS3_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	15	125.5	3.2
Standard_DS3_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	47	130.1	4.3
Standard_DS4_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	23	235.7	6.6
Standard_DS4_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	34	249.4	2.8
Standard_DS5_v2	16	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	11	414.9	5.1
Standard_DS5_v2	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	31	470.6	5.7
Standard_DS1_1_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	22	66.3	2.8
Standard_DS1_1_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	34	64.8	2.8
Standard_DS1_1-1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	17	33.6	1.8
Standard_DS1_1-1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	41	36.0	1.7
Standard_DS1_2_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	10	126.8	2.7

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_DS1_2_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	30	127.5	3.3
Standard_DS1_2-1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	20	33.5	1.4
Standard_DS1_2-1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	30	34.8	2.4
Standard_DS1_2-2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	17	65.5	2.3
Standard_DS1_2-2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	33	67.7	5.1
Standard_DS1_3_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	20	234.1	7.1
Standard_DS1_3_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	23	248.0	2.2
Standard_DS1_3-2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	17	65.2	3.1
Standard_DS1_3-2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	15	72.8	3.8
Standard_DS1_3-4_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	24	126.1	4.3
Standard_DS1_3-4_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	27	133.3	2.8

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_DS1_4_v2	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	22	469.5	6.9
Standard_DS1_4_v2	16	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	456.6	7.3
Standard_DS1_4-4_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	28	132.8	6.6
Standard_DS1_4-4_v2	4	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	125.1	4.8
Standard_DS1_4-8_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	27	251.3	2.4
Standard_DS1_4-8_v2	8	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	14	247.4	10.2
Standard_DS1_5_v2	20	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	45	546.1	10.5

Dv2 - General Compute

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_D1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	30	33.5	1.7
Standard_D1_v2	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	31	34.7	2.5
Standard_D2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	18	66.0	1.8

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_D2_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	31	69.9	5.0
Standard_D3_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	27	127.7	3.0
Standard_D3_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	27	133.4	9.1
Standard_D4_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	15	238.7	4.4
Standard_D4_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	36	248.9	4.8
Standard_D5_v2	16	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	9	413.9	14.1
Standard_D5_v2	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	27	470.2	8.1
Standard_D5_v2	16	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	5	466.0	0.0
Standard_D11_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	22	66.4	2.9
Standard_D11_v2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	27	69.0	6.7
Standard_D12_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	24	127.7	4.6

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_D12_v2	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	20	135.9	9.3
Standard_D13_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	237.4	6.6
Standard_D13_v2	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	28	250.2	3.8
Standard_D14_v2	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	23	473.0	9.4
Standard_D14_v2	16	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	17	443.9	18.8
Standard_D15_v2	20	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	37	558.8	8.4

Esv3 - Memory Optimized + Premium Storage

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_E2s_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	39	42.5	2.2
Standard_E4s_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	28	81.4	3.3
Standard_E8s_v3	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	29	156.3	5.1
Standard_E8-2s_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	57	41.8	2.6

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_E8-4s_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	45	82.9	3.0
Standard_E16s_v3	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	31	295.7	4.5
Standard_E16-4s_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	45	82.7	3.8
Standard_E16-8s_v3	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	39	158.3	4.5
Standard_E20s_v3	20	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	27	369.7	3.2
Standard_E32s_v3	32	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	31	577.9	9.4
Standard_E32-8s_v3	8	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	31	163.4	6.8
Standard_E32-16s_v3	16	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	41	307.1	8.7
Standard_E4-2s_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	65	41.9	2.4
Standard_E64s_v3	64	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1080.0	0.0
Standard_E64-16s_v3	16	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	3	334.3	1.5

SIZE	VCPUS	NUMA NODES	CPU	RUNS	Avg Base Rate	StdDev
Standard_E64-32s_v3	32	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	592.5	4.4

Eisv3 - Memory Opt + Premium Storage (isolated)

SIZE	VCPUS	NUMA NODES	CPU	RUNS	Avg Base Rate	StdDev
Standard_E64i_s_v3	64	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	28	1073.9	5.7

Ev3 - Memory Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	Avg Base Rate	StdDev
Standard_E2_v3	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	41	41.2	2.4
Standard_E4_v3	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	43	81.4	5.3
Standard_E8_v3	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	39	157.4	8.1
Standard_E16_v3	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	49	301.6	8.9
Standard_E20_v3	20	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	35	371.0	6.9
Standard_E32_v3	32	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	35	579.9	16.1
Standard_E64_v3	64	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	31	1080.0	11.3

Eiv3 - Memory Optimized (isolated)

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_E64i_v3	64	2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	28	1081.4	11.1

Fsv2 - Compute + Storage Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_F2s_v2	2	1	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	46	56.5	2.4
Standard_F4s_v2	4	1	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	60	110.2	4.7
Standard_F8s_v2	8	1	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	36	215.2	5.3
Standard_F16s_v2	16	1	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	36	409.3	15.5
Standard_F32s_v2	32	1	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	31	760.9	16.9
Standard_F64s_v2	64	2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	33	1440.9	26.0
Standard_F72s_v2	72	2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	29	1372.1	8.2

Fs - Compute and Storage Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_F1s	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	31	33.2	1.0
Standard_F1s	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	41	35.1	2.0
Standard_F2s	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	18	63.7	1.8
Standard_F2s	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	21	66.6	3.8
Standard_F4s	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	14	128.4	2.9
Standard_F4s	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	25	127.7	4.5
Standard_F8s	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	11	234.9	3.7
Standard_F8s	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	19	251.2	4.5
Standard_F16s	16	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	9	413.9	3.6
Standard_F16s	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	36	471.8	7.5

F - Compute Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
------	-------	------------	-----	------	---------------	--------

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_F1	1	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	15	32.8	1.8
Standard_F1	1	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	13	33.3	2.0
Standard_F2	2	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	27	64.9	6.0
Standard_F2	2	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	21	67.8	4.9
Standard_F4	4	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	18	128.4	3.3
Standard_F4	4	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	132.1	7.8
Standard_F8	8	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	17	239.4	2.3
Standard_F8	8	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	25	251.2	7.0
Standard_F16	16	1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	19	424.1	8.2
Standard_F16	16	1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	467.8	11.1
Standard_F16	16	2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	6	472.3	13.2

GS - Storage Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_GS1	2	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	29	63.6	4.7
Standard_GS2	4	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	29	122.3	6.9
Standard_GS3	8	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	222.4	8.1
Standard_GS4	16	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	391.4	28.6
Standard_GS4-4	4	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	28	127.5	5.3
Standard_GS4-8	8	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	226.7	5.8
Standard_GS5	32	2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	760.9	6.2
Standard_GS5-8	8	2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	259.5	2.7
Standard_GS5-16	16	2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	447.9	4.0

G - Compute Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_G1	2	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	29	64.7	9.2

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_G2	4	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	30	127.9	12.2
Standard_G3	8	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	30	231.7	12.6
Standard_G4	16	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	400.2	3.9
Standard_G5	32	2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	774.1	4.1

H - High Performance Compute (HPC)

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_H8	8	1	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	31	296.1	1.4
Standard_H8m	8	1	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	34	295.1	1.5
Standard_H16	16	2	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	19	563.5	4.3
Standard_H16m	16	2	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	19	562.9	3.3
Standard_H16mr	16	2	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	18	563.6	3.7
Standard_H16r	16	2	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	17	562.2	4.2

Ls - Storage Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_L4s	4	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	29	122.7	6.6
Standard_L8s	8	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	30	223.3	7.5
Standard_L16s	16	1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	397.3	2.5
Standard_L32s	32	2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	31	766.1	3.5

M - Memory Optimized

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_M8-2ms	2	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	15	42.1	2.1
Standard_M8-4ms	4	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	13	81.6	2.9
Standard_M16-4ms	4	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	14	82.5	2.5
Standard_M16-8ms	8	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	20	157.2	6.0
Standard_M32-8ms	8	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	18	162.5	2.1
Standard_M32-16ms	16	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	306.5	0.5

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_M6_4	64	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	11	1010.9	5.4
Standard_M6_4-16ms	16	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	13	316.0	2.4
Standard_M6_4-32ms	32	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	586.8	5.4
Standard_M6_4m	64	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	1005.5	12.3
Standard_M6_4ms	64	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	1012.9	12.5
Standard_M6_4s	64	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	1012.5	4.5
Standard_M1_28	128	4	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	11	1777.3	15.6
Standard_M1_28-32ms	32	4	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	13	620.5	2.5
Standard_M1_28-64ms	64	4	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	1140.8	2.9
Standard_M1_28m	128	4	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	1778.3	10.3
Standard_M1_28ms	128	4	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	15	1780.7	18.3

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_M1 28s	128	4	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	1775.8	11.6
Standard_M1 6ms	16	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	20	293.1	11.8
Standard_M3 2ls	32	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	13	535.2	4.8
Standard_M3 2ms	32	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	11	534.1	4.6
Standard_M3 2ms	32	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	1	589.0	0.0
Standard_M3 2ts	32	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	12	538.6	3.2
Standard_M6 4ls	64	2	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	13	1015.2	10.0
Standard_M8 ms	8	1	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	13	158.2	5.5

NCSv3 - GPU Enabled

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_NC6 s_v3	6	1	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	6	230.2	1.6
Standard_NC1 2s_v3	12	1	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	7	425.0	3.6

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_NC2_4rs_v3	24	2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	2	811.0	4.2
Standard_NC2_4s_v3	24	2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	3	809.3	2.3

NCSv2 - GPU Enabled

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_NC6_s_v2	6	1	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	11	227.0	6.2
Standard_NC1_2s_v2	12	1	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	9	427.3	1.3
Standard_NC2_4rs_v2	24	2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	12	811.0	5.4
Standard_NC2_4s_v2	24	2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	11	811.5	4.4

NC - GPU Enabled

SIZE	VCPUS	NUMA NODES	CPU	RUNS	AVG BASE RATE	STDDEV
Standard_NC6	6	1	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	27	209.6	4.4
Standard_NC1_2	12	1	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	28	394.4	3.8
Standard_NC2_4	24	2	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	28	751.7	3.5

SIZE	VCPUS	NUMA NODES	CPU	RUNS	Avg Base Rate	StdDev
Standard_NC2_4r	24	2	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	27	752.9	3.4

NDs- GPU Enabled

SIZE	VCPUS	NUMA NODES	CPU	RUNS	Avg Base Rate	StdDev
Standard_ND6_s	6	1	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	8	230.1	1.2
Standard_ND1_2s	12	1	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	11	426.5	1.4
Standard_ND2_4rs	24	2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	10	811.4	3.5
Standard_ND2_4s	24	2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	11	812.6	4.4

NV - GPU Enabled

SIZE	VCPUS	NUMA NODES	CPU	RUNS	Avg Base Rate	StdDev
Standard_NV6	6	1	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	28	210.5	6.1
Standard_NV1_2	12	1	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	28	394.5	2.3
Standard_NV2_4	24	2	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	26	752.2	4.4

About SPECint

Windows numbers were computed by running [SPECint 2006](#) on Windows Server. SPECint was run using the base rate option (SPECint_rate2006), with one copy per vCPU. SPECint consists of 12 separate tests, each run three times, taking the median value from each test and weighting them to form a composite score. Those tests

were then run across multiple VMs to provide the average scores shown.

Next steps

- For storage capacities, disk details, and additional considerations for choosing among VM sizes, see [Sizes for virtual machines](#).

Endorsed Linux distributions on Azure

1/8/2020 • 5 minutes to read • [Edit Online](#)

Partners provide Linux images in the Azure Marketplace. We are working with various Linux communities to add even more flavors to the Endorsed Distribution list. In the meantime, for distributions that are not available from the Marketplace, you can always bring your own Linux by following the guidelines at [Create and upload a virtual hard disk that contains the Linux operating system](#).

Supported distributions and versions

The following table lists the Linux distributions and versions that are supported on Azure. Refer to [Support for Linux images in Microsoft Azure](#) for more detailed information about support for Linux and open-source technology in Azure.

The Linux Integration Services (LIS) drivers for Hyper-V and Azure are kernel modules that Microsoft contributes directly to the upstream Linux kernel. Some LIS drivers are built into the distribution's kernel by default. Older distributions that are based on Red Hat Enterprise (RHEL)/CentOS are available as a separate download at [Linux Integration Services Version 4.2 for Hyper-V and Azure](#). See [Linux kernel requirements](#) for more information about the LIS drivers.

The Azure Linux Agent is already pre-installed on the Azure Marketplace images and is typically available from the distribution's package repository. Source code can be found on [GitHub](#).

DISTRIBUTION	VERSION	DRIVERS	AGENT
CentOS	CentOS 6.3+, 7.0+, 8.0+	CentOS 6.3: LIS download CentOS 6.4+: In kernel	Package: In repo under "WALinuxAgent" Source code: GitHub
CoreOS	494.4.0+	In kernel	Source code: GitHub
Debian	Debian 7.9+, 8.2+, 9, 10	In kernel	Package: In repo under "waagent" Source code: GitHub
Oracle Linux	6.4+, 7.0+	In kernel	Package: In repo under "WALinuxAgent" Source code: GitHub
Red Hat Enterprise Linux	RHEL 6.7+, 7.1+, 8.0+	In kernel	Package: In repo under "WALinuxAgent" Source code: GitHub
SUSE Linux Enterprise	SLES/SLES for SAP 11 SP4 12 SP1+ 15	In kernel	Package: for 11 in Cloud:Tools repo for 12 included in "Public Cloud" Module under "python-azure-agent" Source code: GitHub

DISTRIBUTION	VERSION	DRIVERS	AGENT
openSUSE	openSUSE Leap 42.2+	In kernel	Package: In Cloud:Tools repo under "python-azure-agent" Source code: GitHub
Ubuntu	Ubuntu 12.04+ ¹	In kernel	Package: In repo under "walinuxagent" Source code: GitHub

- ¹ Information about extended support for Ubuntu 12.04 and 14.04 can be found here: [Ubuntu Extended Security Maintenance](#).

Image update cadence

Azure requires that the publishers of the endorsed Linux distributions regularly update their images in the Azure Marketplace with the latest patches and security fixes, at a quarterly or faster cadence. Updated images in the Azure Marketplace are available automatically to customers as new versions of an image SKU. More information about how to find Linux images: [Find Linux VM images in the Azure Marketplace](#).

Additional links

- [SUSE Public Cloud Image Lifecycle](#)

Azure-tuned kernels

Azure works closely with various endorsed Linux distributions to optimize the images that they published to the Azure Marketplace. One aspect of this collaboration is the development of "tuned" Linux kernels that are optimized for the Azure platform and delivered as fully supported components of the Linux distribution. The Azure-Tuned kernels incorporate new features and performance improvements, and at a faster (typically quarterly) cadence compared to the default or generic kernels that are available from the distribution.

In most cases you will find these kernels pre-installed on the default images in the Azure Marketplace, and so Azure customers will immediately get the benefit of these optimized kernels. More information about these Azure-Tuned kernels can be found in the following links:

- CentOS Azure-Tuned Kernel - Available via the CentOS Virtualization SIG - [More Info](#)
- Debian Cloud Kernel - Available with the Debian 10 and Debian 9 "backports" image on Azure - [More Info](#)
- SLES Azure-Tuned Kernel - [More Info](#)
- Ubuntu Azure-Tuned Kernel - [More Info](#)

Partners

CoreOS

<https://coreos.com/docs/running-coreos/cloud-providers/azure/>

From the CoreOS website:

CoreOS is designed for security, consistency, and reliability. Instead of installing packages via yum or apt, CoreOS uses Linux containers to manage your services at a higher level of abstraction. A single service's code and all dependencies are packaged within a container that can be run on one or many CoreOS machines.

Creativ

<https://www.creativ.co.uk/creativ-blog/debian-images-microsoft-azure>

Creativ is an independent consulting and services company that specializes in the development and

implementation of professional solutions by using free software. As leading open-source specialists, Credativ has international recognition with many IT departments that use their support. In conjunction with Microsoft, Credativ is currently preparing corresponding Debian images for Debian 8 (Jessie) and Debian before 7 (Wheezy). Both images are specially designed to run on Azure and can be easily managed via the platform. Credativ will also support the long-term maintenance and updating of the Debian images for Azure through its Open Source Support Centers.

Oracle

<https://www.oracle.com/technetwork/topics/cloud/faq-1963009.html>

Oracle's strategy is to offer a broad portfolio of solutions for public and private clouds. The strategy gives customers choice and flexibility in how they deploy Oracle software in Oracle clouds and other clouds. Oracle's partnership with Microsoft enables customers to deploy Oracle software in Microsoft public and private clouds with the confidence of certification and support from Oracle. Oracle's commitment and investment in Oracle public and private cloud solutions is unchanged.

Red Hat

<https://www.redhat.com/en/partners/strategic-alliance/microsoft>

The world's leading provider of open source solutions, Red Hat helps more than 90% of Fortune 500 companies solve business challenges, align their IT and business strategies, and prepare for the future of technology. Red Hat does this by providing secure solutions through an open business model and an affordable, predictable subscription model.

SUSE

<https://www.suse.com/suse-linux-enterprise-server-on-azure>

SUSE Linux Enterprise Server on Azure is a proven platform that provides superior reliability and security for cloud computing. SUSE's versatile Linux platform seamlessly integrates with Azure cloud services to deliver an easily manageable cloud environment. With more than 9,200 certified applications from more than 1,800 independent software vendors for SUSE Linux Enterprise Server, SUSE ensures that workloads running supported in the data center can be confidently deployed on Azure.

Canonical

<https://www.ubuntu.com/cloud/azure>

Canonical engineering and open community governance drive Ubuntu's success in client, server, and cloud computing, which includes personal cloud services for consumers. Canonical's vision of a unified, free platform in Ubuntu, from phone to cloud, provides a family of coherent interfaces for the phone, tablet, TV, and desktop. This vision makes Ubuntu the first choice for diverse institutions from public cloud providers to the makers of consumer electronics and a favorite among individual technologists.

With developers and engineering centers around the world, Canonical is uniquely positioned to partner with hardware makers, content providers, and software developers to bring Ubuntu solutions to market for PCs, servers, and handheld devices.

2 minutes to read

Orchestration mode (preview)

1/19/2020 • 3 minutes to read • [Edit Online](#)

Virtual machines scale sets provide a logical grouping of platform-managed virtual machines. With scale sets, you create a virtual machine configuration model, automatically add or remove additional instances based on CPU or memory load, and automatically upgrade to the latest OS version. Traditionally, scale sets allow you to create virtual machines using a VM configuration model provided at time of scale set creation, and the scale set can only manage virtual machines that are implicitly created based on the configuration model.

With the scale set orchestration mode (preview), you can now choose whether the scale set should orchestrate virtual machines which are created explicitly outside of a scale set configuration model, or virtual machine instances created implicitly based on the configuration model. Scale set orchestration mode also helps you design your VM infrastructure for high availability by deploying these VMs in fault domains and Availability Zones.

Virtual machine scale sets will support 2 distinct orchestration modes:

- **ScaleSetVM** – Virtual machine instances added to the scale set are based on the scale set configuration model. The virtual machine instance lifecycle - creation, update, deletion - is managed by the scale set.
- **VM (virtual machines)** – Virtual machines created outside of the scale set can be explicitly added to the scaleset.

IMPORTANT

The orchestration mode is defined when you create the scale set and cannot be changed or updated later.

This feature of virtual machine scale sets is currently in public preview. This preview version is provided without a service level agreement, and it's not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

Orchestration modes

	"ORCHESTRATIONMODE": "VM" (VIRTUALMACHINE)	"ORCHESTRATIONMODE": "SCALESETVM" (VIRTUALMACHINESCALESETV M)	
VM configuration model	None	Required	
Adding new VM to Scale Set	VMs are explicitly added to the scale set when the VM is created.	VMs are implicitly created and added to the scale set based on the VM configuration model, instance count, and AutoScaling rules	
Delete VM	VMs have to be deleted individually, the scale set will not be deleted if it has any VMs in it.	VMs can be deleted individually, deleting the scale set will delete all of the VM instances.	
Attach/Detach VMs	Not supported	Not supported	

	"ORCHESTRATIONMODE": "VM" (VIRTUALMACHINE)	"ORCHESTRATIONMODE": "SCALESETVM" (VIRTUALMACHINESCALESETV M)
Instance Lifecycle (Creation through Deletion)	VMs and their artifacts (like disks and NICs) can be managed independently.	Instances and their artifacts (like disks and NICs) are implicit to the scale set instances that create them. They cannot be detached or managed separately outside the scale set
Fault domains	Can define fault domains. 2 or 3 based on regional support and 5 for Availability zone.	Can define fault domains going from 1 through 5
Update domains	Update domains are automatically mapped to fault domains	Update domains are automatically mapped to fault domains
Availability Zones	Supports regional deployment or VMs in one Availability Zone	Supports regional deployment or multiple Availability Zones; Can define the zone balancing strategy
AutoScale	Not supported	Supported
OS upgrade	Not supported	Supported
Model updates	Not supported	Supported
Instance control	Full VM Control. VMs have fully qualified URI that support the full range of Azure VM management capabilities (like Azure Policy, Azure Backup, and Azure Site Recovery)	VMs are dependent resources of the scale set. Instances can be accessed for management only through the scale set.
Instance Model	Microsoft.Compute/VirtualMachines model definition.	Microsoft.Compute/VirtualMachineScaleSets/VirtualMachines model definition.
Capacity	An empty scale set can be created; up to 200 VMs can be added to the scale set	Scale sets can be defined with an instance count 0 - 1000
Move	Supported	Supported
Single placement group == false	Not supported	Supported

Next steps

For more information, see the [Overview of availability options](#).

Introduction to Azure managed disks

12/16/2019 • 9 minutes to read • [Edit Online](#)

Azure managed disks are block-level storage volumes that are managed by Azure and used with Azure Virtual Machines. Managed disks are like a physical disk in an on-premises server but virtualized. With managed disks, all you have to do is specify the disk size, the disk type, and provision the disk. Once you provision the disk, Azure handles the rest.

The available types of disks are ultra disks, premium solid-state drives (SSD), standard SSDs, and standard hard disk drives (HDD). For information about each individual disk type, see [Select a disk type for IaaS VMs](#).

Benefits of managed disks

Let's go over some of the benefits you gain by using managed disks.

Highly durable and available

Managed disks are designed for 99.999% availability. Managed disks achieve this by providing you with three replicas of your data, allowing for high durability. If one or even two replicas experience issues, the remaining replicas help ensure persistence of your data and high tolerance against failures. This architecture has helped Azure consistently deliver enterprise-grade durability for infrastructure as a service (IaaS) disks, with an industry-leading ZERO% annualized failure rate.

Simple and scalable VM deployment

Using managed disks, you can create up to 50,000 VM **disks** of a type in a subscription per region, allowing you to create thousands of **VMs** in a single subscription. This feature also further increases the scalability of [virtual machine scale sets](#) by allowing you to create up to 1,000 VMs in a virtual machine scale set using a Marketplace image.

Integration with availability sets

Managed disks are integrated with availability sets to ensure that the disks of [VMs in an availability set](#) are sufficiently isolated from each other to avoid a single point of failure. Disks are automatically placed in different storage scale units (stamps). If a stamp fails due to hardware or software failure, only the VM instances with disks on those stamps fail. For example, let's say you have an application running on five VMs, and the VMs are in an Availability Set. The disks for those VMs won't all be stored in the same stamp, so if one stamp goes down, the other instances of the application continue to run.

Integration with Availability Zones

Managed disks support [Availability Zones](#), which is a high-availability offering that protects your applications from datacenter failures. Availability Zones are unique physical locations within an Azure region. Each zone is made up of one or more datacenters equipped with independent power, cooling, and networking. To ensure resiliency, there's a minimum of three separate zones in all enabled regions. With Availability Zones, Azure offers industry best 99.99% VM uptime SLA.

Azure Backup support

To protect against regional disasters, [Azure Backup](#) can be used to create a backup job with time-based backups and backup retention policies. This allows you to perform easy VM restorations at will. Currently Azure Backup supports disk sizes up to four tebibyte (TiB) disks. Azure Backup supports backup and restore of managed disks. [Learn more](#) about Azure VM backup support.

Granular access control

You can use [Azure role-based access control \(RBAC\)](#) to assign specific permissions for a managed disk to one or more users. Managed disks expose a variety of operations, including read, write (create/update), delete, and retrieving a [shared access signature \(SAS\) URI](#) for the disk. You can grant access to only the operations a person needs to perform their job. For example, if you don't want a person to copy a managed disk to a storage account, you can choose not to grant access to the export action for that managed disk. Similarly, if you don't want a person to use an SAS URI to copy a managed disk, you can choose not to grant that permission to the managed disk.

Upload your vhd

Direct upload makes it easy to transfer your vhd to an Azure managed disk. Previously, you had to follow a more involved process that included staging your data in a storage account. Now, there are fewer steps. It is easier to upload on premises VMs to Azure, upload to large managed disks, and the backup and restore process is simplified. It also reduces cost by allowing you to upload data to managed disks directly without attaching them to VMs. You can use direct upload to upload vhds up to 32 TiB in size.

To learn how to transfer your vhd to Azure, see the [CLI](#) or [PowerShell](#) articles.

Encryption

Managed disks offer two different kinds of encryption. The first is Server Side Encryption (SSE), which is performed by the storage service. The second one is Azure Disk Encryption (ADE), which you can enable on the OS and data disks for your VMs.

Server-side encryption

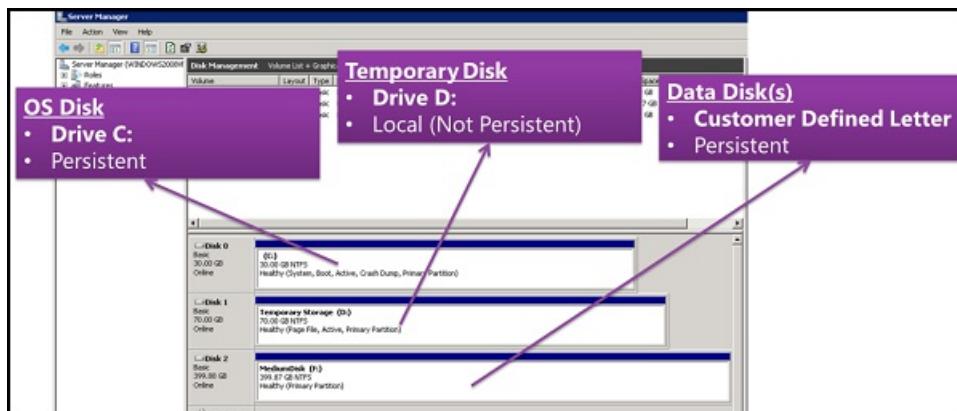
[Azure Server-side Encryption](#) provides encryption-at-rest and safeguards your data to meet your organizational security and compliance commitments. Server-side encryption is enabled by default for all managed disks, snapshots, and images in all the regions where managed disks are available. You can either allow Azure to manage your keys for you, these are platform-managed keys, or you can manage the keys yourself, these are customer-managed keys. Visit the [Managed Disks FAQ page](#) for more details.

Azure Disk Encryption

Azure Disk Encryption allows you to encrypt the OS and Data disks used by an IaaS Virtual Machine. This encryption includes managed disks. For Windows, the drives are encrypted using industry-standard BitLocker encryption technology. For Linux, the disks are encrypted using the DM-Crypt technology. The encryption process is integrated with Azure Key Vault to allow you to control and manage the disk encryption keys. For more information, see [Azure Disk Encryption for IaaS VMs](#).

Disk roles

There are three main disk roles in Azure: the data disk, the OS disk, and the temporary disk. These roles map to disks that are attached to your virtual machine.



Data disk

A data disk is a managed disk that's attached to a virtual machine to store application data, or other data you need to keep. Data disks are registered as SCSI drives and are labeled with a letter that you choose. Each data disk has a maximum capacity of 32,767 gibibytes (GiB). The size of the virtual machine determines how many data disks you can attach to it and the type of storage you can use to host the disks.

OS disk

Every virtual machine has one attached operating system disk. That OS disk has a pre-installed OS, which was selected when the VM was created. This disk contains the boot volume.

This disk has a maximum capacity of 2,048 GiB.

Temporary disk

Every VM contains a temporary disk, which is not a managed disk. The temporary disk provides short-term storage for applications and processes and is intended to only store data such as page or swap files. Data on the temporary disk may be lost during a [maintenance event](#) event or when you [redeploy a VM](#). On Azure Linux VMs, the temporary disk is /dev/sdb by default and on Windows VMs the temporary disk is D: by default. During a successful standard reboot of the VM, the data on the temporary disk will persist.

Managed disk snapshots

A managed disk snapshot is a read-only crash-consistent full copy of a managed disk that is stored as a standard managed disk by default. With snapshots, you can back up your managed disks at any point in time. These snapshots exist independent of the source disk and can be used to create new managed disks.

Snapshots are billed based on the used size. For example, if you create a snapshot of a managed disk with provisioned capacity of 64 GiB and actual used data size of 10 GiB, that snapshot is billed only for the used data size of 10 GiB. You can see the used size of your snapshots by looking at the [Azure usage report](#). For example, if the used data size of a snapshot is 10 GiB, the **daily** usage report will show $10 \text{ GiB} / (31 \text{ days}) = 0.3226$ as the consumed quantity.

To learn more about how to create snapshots for managed disks, see the following resources:

- [Create a snapshot of a managed disk in Windows](#)
- [Create a snapshot of a managed disk in Linux](#)

Images

Managed disks also support creating a managed custom image. You can create an image from your custom VHD in a storage account or directly from a generalized (sysprepped) VM. This process captures a single image. This image contains all managed disks associated with a VM, including both the OS and data disks. This managed custom image enables creating hundreds of VMs using your custom image without the need to copy or manage any storage accounts.

For information on creating images, see the following articles:

- [How to capture a managed image of a generalized VM in Azure](#)
- [How to generalize and capture a Linux virtual machine using the Azure CLI](#)

Images versus snapshots

It's important to understand the difference between images and snapshots. With managed disks, you can take an image of a generalized VM that has been deallocated. This image includes all of the disks attached to the VM. You can use this image to create a VM, and it includes all of the disks.

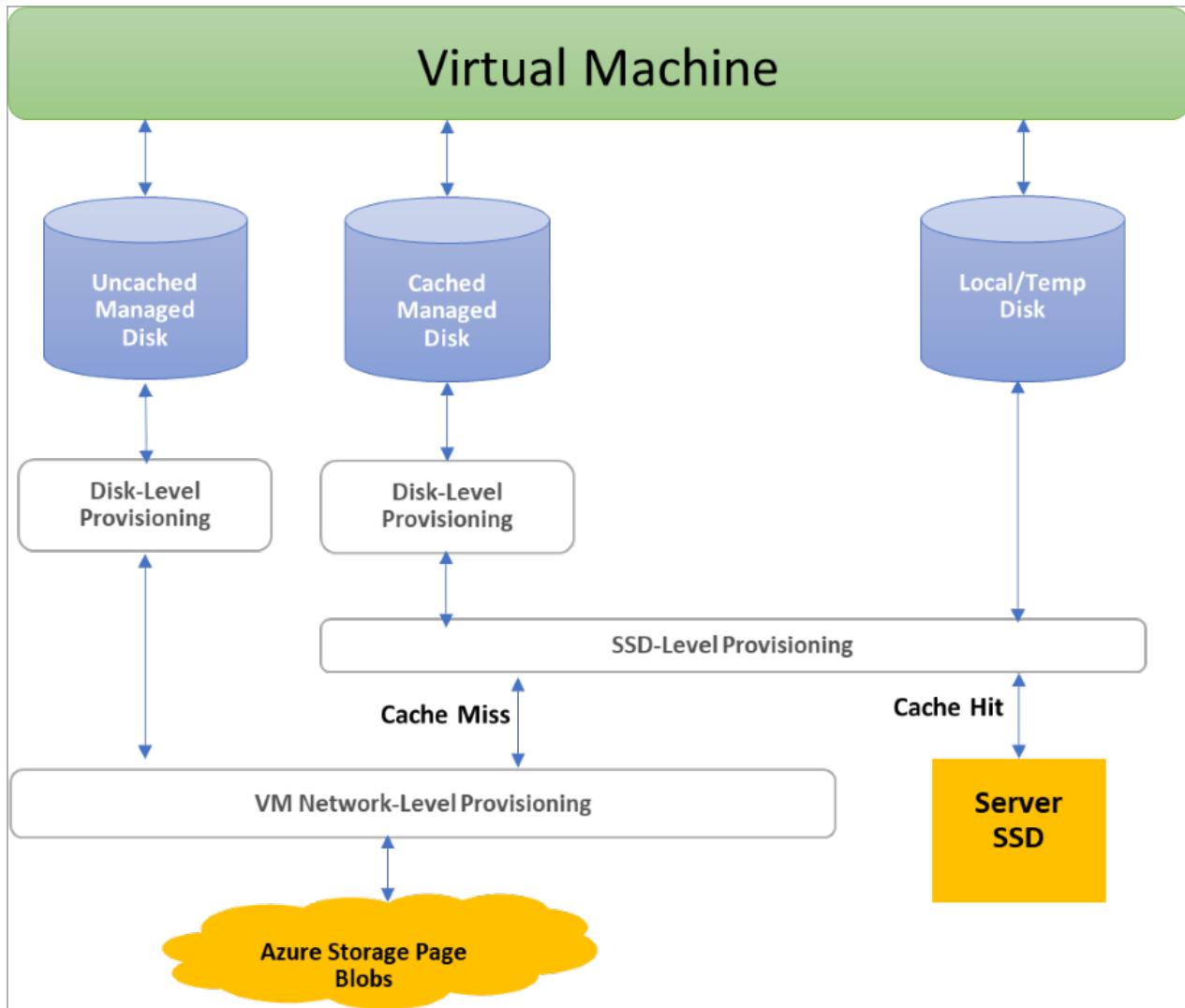
A snapshot is a copy of a disk at the point in time the snapshot is taken. It applies only to one disk. If you have a VM that has one disk (the OS disk), you can take a snapshot or an image of it and create a VM from either the snapshot or the image.

A snapshot doesn't have awareness of any disk except the one it contains. This makes it problematic to use in

scenarios that require the coordination of multiple disks, such as striping. Snapshots would need to be able to coordinate with each other and this is currently not supported.

Disk allocation and performance

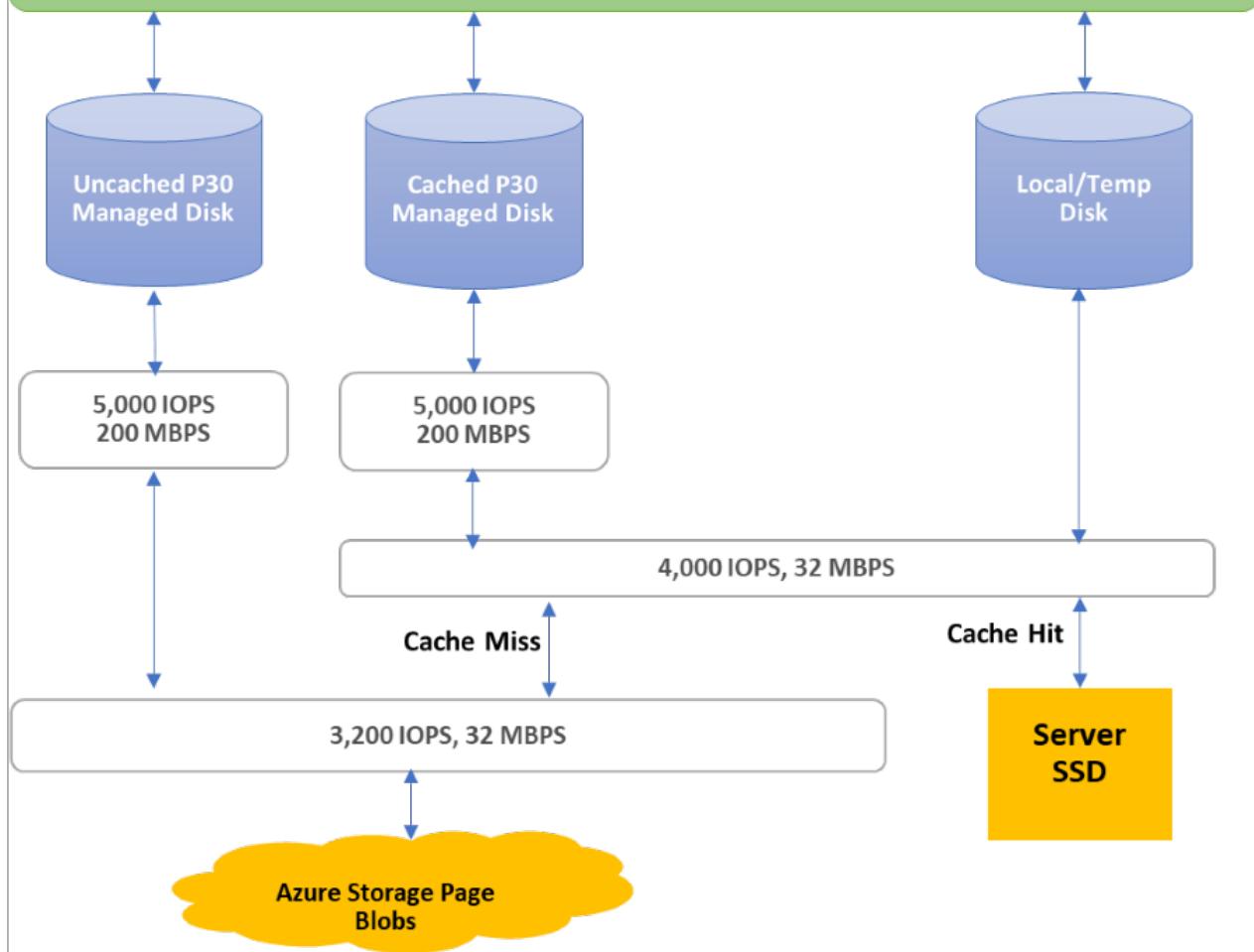
The following diagram depicts real-time allocation of bandwidth and IOPS for disks, using a three-level provisioning system:



The first level provisioning sets the per-disk IOPS and bandwidth assignment. At the second level, compute server host implements SSD provisioning, applying it only to data that is stored on the server's SSD, which includes disks with caching (ReadWrite and ReadOnly) as well as local and temp disks. Finally, VM network provisioning takes place at the third level for any I/O that the compute host sends to Azure Storage's backend. With this scheme, the performance of a VM depends on a variety of factors, from how the VM uses the local SSD, to the number of disks attached, as well as the performance and caching type of the disks it has attached.

As an example of these limitations, a Standard_DS1v1 VM is prevented from achieving the 5,000 IOPS potential of a P30 disk, whether it is cached or not, because of limits at the SSD and network levels:

Virtual Machine: Standard_DS1v1



Azure uses prioritized network channel for disk traffic, which gets the precedence over other low priority of network traffic. This helps disks maintain their expected performance in case of network contentions. Similarly, Azure Storage handles resource contentions and other issues in the background with automatic load balancing. Azure Storage allocates required resources when you create a disk, and applies proactive and reactive balancing of resources to handle the traffic level. This further ensures disks can sustain their expected IOPS and throughput targets. You can use the VM-level and Disk-level metrics to track the performance and setup alerts as needed.

Refer to our [design for high performance](#) article, to learn the best practices for optimizing VM + Disk configurations so that you can achieve your desired performance

Next steps

If you'd like a video going into more detail on managed disks, check out: [Better Azure VM Resiliency with Managed Disks](#).

Learn more about the individual disk types Azure offers, which type is a good fit for your needs, and learn about their performance targets in our article on disk types.

[Select a disk type for IaaS VMs](#)

What disk types are available in Azure?

11/12/2019 • 13 minutes to read • [Edit Online](#)

Azure managed disks currently offers four disk types, each type is aimed towards specific customer scenarios.

Disk comparison

The following table provides a comparison of ultra disks, premium solid-state drives (SSD), standard SSD, and standard hard disk drives (HDD) for managed disks to help you decide what to use.

	ULTRA DISK	PREMIUM SSD	STANDARD SSD	STANDARD HDD
Disk type	SSD	SSD	SSD	HDD
Scenario	IO-intensive workloads such as SAP HANA, top tier databases (for example, SQL, Oracle), and other transaction-heavy workloads.	Production and performance sensitive workloads	Web servers, lightly used enterprise applications and dev/test	Backup, non-critical, infrequent access
Max disk size	65,536 gibibyte (GiB)	32,767 GiB	32,767 GiB	32,767 GiB
Max throughput	2,000 MiB/s	900 MiB/s	750 MiB/s	500 MiB/s
Max IOPS	160,000	20,000	6,000	2,000

Ultra disk

Azure ultra disks deliver high throughput, high IOPS, and consistent low latency disk storage for Azure IaaS VMs. Some additional benefits of ultra disks include the ability to dynamically change the performance of the disk, along with your workloads, without the need to restart your virtual machines (VM). Ultra disks are suited for data-intensive workloads such as SAP HANA, top tier databases, and transaction-heavy workloads. Ultra disks can only be used as data disks. We recommend using premium SSDs as OS disks.

Performance

When you provision an ultra disk, you can independently configure the capacity and the performance of the disk. Ultra disks come in several fixed sizes, ranging from 4 GiB up to 64 TiB, and feature a flexible performance configuration model that allows you to independently configure IOPS and throughput.

Some key capabilities of ultra disks are:

- Disk capacity: Ultra disks capacity ranges from 4 GiB up to 64 TiB.
- Disk IOPS: Ultra disks support IOPS limits of 300 IOPS/GiB, up to a maximum of 160 K IOPS per disk. To achieve the IOPS that you provisioned, ensure that the selected Disk IOPS are less than the VM IOPS limit. The minimum IOPS per disk is 2 IOPS/GiB, with an overall baseline minimum of 100 IOPS. For example, if you had a 4 GiB ultra disk, you will have a minimum of 100 IOPS, instead of eight IOPS.
- Disk throughput: With ultra disks, the throughput limit of a single disk is 256 KiB/s for each provisioned IOPS, up to a maximum of 2000 MBps per disk (where MBps = 10^6 Bytes per second). The minimum throughput

per disk is 4KiB/s for each provisioned IOPS, with an overall baseline minimum of 1 MBps.

- Ultra disks support adjusting the disk performance attributes (IOPS and throughput) at runtime without detaching the disk from the virtual machine. Once a disk performance resize operation has been issued on a disk, it can take up to an hour for the change to actually take effect. There is a limit of four performance resize operations during a 24 hour window. It is possible for a performance resize operation to fail due to a lack of performance bandwidth capacity.

Disk size

DISK SIZE (GiB)	IOPS CAP	THROUGHPUT CAP (MBPS)
4	1,200	300
8	2,400	600
16	4,800	1,200
32	9,600	2,000
64	19,200	2,000
128	38,400	2,000
256	76,800	2,000
512	80,000	2,000
1,024-65,536 (sizes in this range increasing in increments of 1 TiB)	160,000	2,000

GA scope and limitations

For now, ultra disks have additional limitations, they are as follows:

- Are supported in the following regions, with a varying number of availability zones per region:
 - East US 2
 - East US
 - West US 2
 - SouthEast Asia
 - North Europe
 - West Europe
 - UK South
- Can only be used with availability zones (availability sets and single VM deployments outside of zones will not have the ability to attach an ultra disk)
- Are only supported on the following VM series:
 - [ESv3](#)
 - [DSv3](#)
 - [FSv2](#)
 - [M](#)
 - [Mv2](#)
- Not every VM size is available in every supported region with ultra disks
- Are only available as data disks and only support 4k physical sector size. Due to the 4K native sector size of Ultra Disk, there are some applications that won't be compatible with ultra disks. One example would be Oracle

Database, which requires release 12.2 or later in order to support ultra disks.

- Can only be created as empty disks
- Do not yet support disk snapshots, VM images, availability sets, and Azure disk encryption
- Do not yet support integration with Azure Backup or Azure Site Recovery
- The current maximum limit for IOPS on GA VMs is 80,000.
- If you would like to participate in a limited preview of a VM that can accomplish 160,000 IOPS with ultra disks, please email UltraDiskFeedback@microsoft.com

If you would like to start using ultra disks, see our article on the subject: [Using Azure ultra disks](#).

Premium SSD

Azure premium SSDs deliver high-performance and low-latency disk support for virtual machines (VMs) with input/output (IO)-intensive workloads. To take advantage of the speed and performance of premium storage disks, you can migrate existing VM disks to Premium SSDs. Premium SSDs are suitable for mission-critical production applications. Premium SSDs can only be used with VM series that are premium storage-compatible.

To learn more about individual VM types and sizes in Azure for Windows, including which sizes are premium storage-compatible, see [Windows VM sizes](#). To learn more about individual VM types and sizes in Azure for Linux, including which sizes are premium storage-compatible, see [Linux VM sizes](#).

Disk size

PRE MIU M SSD SIZE S	P1*	P2*	P3*	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
Disk size in GiB	4	8	16	32	64	128	256	512	1,02 4	2,04 8	4,09 6	8,19 2	16,3 84	32,7 67
IOP S per disk	120	120	120	120	240	500	1,10 0	2,30 0	5,00 0	7,50 0	7,50 0	16,0 00	18,0 00	20,0 00
Thr oug hpu t per disk	25 MiB /sec	25 MiB /sec	25 MiB /sec	25 MiB /sec	50 MiB /sec	100 MiB /sec	125 MiB /sec	150 MiB /sec	200 MiB /sec	250 MiB /sec	250 MiB /sec	500 MiB /sec	750 MiB /sec	900 MiB /sec
Max bur st IOP S per disk **	3,5 00	3,5 00	3,5 00	3,5 00	3,5 00	3,50 0	3,50 0	3,50 0	3,50 0					

PRE MIU M SSD SIZE S	P1*	P2*	P3*	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
Max burst throughput per disk **	170 MiB /sec													
Max burst duration**	30 min													
Eligible for reservation	No	Yes, up to one year												

*Denotes a disk size that is currently in preview, for regional availability information see [New disk sizes: Managed and unmanaged](#).

**Denotes a feature that is currently in preview, see [Disk bursting](#) for more information.

When you provision a premium storage disk, unlike standard storage, you are guaranteed the capacity, IOPS, and throughput of that disk. For example, if you create a P50 disk, Azure provisions 4,095-GB storage capacity, 7,500 IOPS, and 250-MB/s throughput for that disk. Your application can use all or part of the capacity and performance. Premium SSD disks are designed to provide low single-digit millisecond latencies and target IOPS and throughput described in the preceding table 99.9% of the time.

Bursting (preview)

Premium SSD sizes smaller than P30 now offer disk bursting (preview) and can burst their IOPS per disk up to 3,500 and their bandwidth up to 170 Mbps. Bursting is automated and operates based on a credit system. Credits are automatically accumulated in a burst bucket when disk traffic is below the provisioned performance target and credits are automatically consumed when traffic bursts beyond the target, up to the max burst limit. The max burst limit defines the ceiling of disk IOPS & Bandwidth even if you have burst credits to consume from. Disk bursting provides better tolerance on unpredictable changes of IO patterns. You can best leverage it for OS disk boot and applications with spiky traffic.

Disk bursting support will be enabled on new deployments of applicable disk sizes in the [preview regions](#) by default, with no user action required. For existing disks of the applicable sizes, you can enable bursting with either of two options: detach and reattach the disk or stop and restart the attached VM. All burst applicable disk sizes will start with a full burst credit bucket when the disk is attached to a Virtual Machine that supports a max duration at peak burst limit of 30 mins. To learn more about how bursting work on Azure Disks, see [Premium SSD bursting](#).

Transactions

For premium SSDs, each I/O operation less than or equal to 256 KiB of throughput is considered a single I/O

operation. I/O operations larger than 256 KiB of throughput are considered multiple I/Os of size 256 KiB.

Standard SSD

Azure standard SSDs are a cost-effective storage option optimized for workloads that need consistent performance at lower IOPS levels. Standard SSD offers a good entry level experience for those who wish to move to the cloud, especially if you experience issues with the variance of workloads running on your HDD solutions on premises. Compared to standard HDDs, standard SSDs deliver better availability, consistency, reliability, and latency. Standard SSDs are suitable for Web servers, low IOPS application servers, lightly used enterprise applications, and Dev/Test workloads. Like standard HDDs, standard SSDs are available on all Azure VMs.

Disk size

STANDBY RDSSD SIZE S	E1*	E2*	E3*	E4	E6	E10	E15	E20	E30	E40	E50	E60	E70	E80
Disk size in GiB	4	8	16	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384	32,767
IOPS per disk	Up to 120	Up to 120	Up to 120	Up to 120	Up to 240	Up to 500	Up to 2,000	Up to 4,000	Up to 6,000	Up to 6,000				
Throughput per disk	Up to 25 MiB /sec	Up to 50 MiB /sec	Up to 60 MiB /sec	Up to 400 MiB /sec	Up to 600 MiB /sec	Up to 750 MiB /sec	Up to 750 MiB /sec							

*Denotes a disk size that is currently in preview, for regional availability information see [New disk sizes: Managed and unmanaged](#).

Standard SSDs are designed to provide single-digit millisecond latencies and the IOPS and throughput up to the limits described in the preceding table 99% of the time. Actual IOPS and throughput may vary sometimes depending on the traffic patterns. Standard SSDs will provide more consistent performance than the HDD disks with the lower latency.

Transactions

For standard SSDs, each I/O operation less than or equal to 256 KiB of throughput is considered a single I/O operation. I/O operations larger than 256 KiB of throughput are considered multiple I/Os of size 256 KiB. These transactions have a billing impact.

Standard HDD

Azure standard HDDs deliver reliable, low-cost disk support for VMs running latency-insensitive workloads. With standard storage, the data is stored on hard disk drives (HDDs). Latency, IOPS, and Throughput of Standard HDD disks may vary more widely as compared to SSD-based disks. Standard HDD Disks are designed to deliver write latencies under 10ms and read latencies under 20ms for most IO operations, however the actual performance may vary depending on the IO size and workload pattern. When working with VMs, you can use standard HDD disks for dev/test scenarios and less critical workloads. Standard HDDs are available in all Azure regions and can be

used with all Azure VMs.

Disk size

STANDARD DISK TYPE	S4	S6	S10	S15	S20	S30	S40	S50	S60	S70	S80
Disk size in GiB	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384	32,767
IOPS per disk	Up to 500	Up to 1,300	Up to 2,000	Up to 2,000							
Throughput per disk	Up to 60 MiB/s ec	Up to 300 MiB/s ec	Up to 500 MiB/s ec	Up to 500 MiB/s ec							

Transactions

For Standard HDDs, each IO operation is considered as a single transaction, regardless of the I/O size. These transactions have a billing impact.

Billing

When using managed disks, the following billing considerations apply:

- Disk type
- managed disk Size
- Snapshots
- Outbound data transfers
- Number of transactions

Managed disk size: managed disks are billed on the provisioned size. Azure maps the provisioned size (rounded up) to the nearest offered disk size. For details of the disk sizes offered, see the previous tables. Each disk maps to a supported provisioned disk size offering and is billed accordingly. For example, if you provisioned a 200 GiB Standard SSD, it maps to the disk size offer of E15 (256 GiB). Billing for any provisioned disk is prorated hourly by using the monthly price for the Premium Storage offer. For example, if you provisioned an E10 disk and deleted it after 20 hours, you're billed for the E10 offering prorated to 20 hours. This is regardless of the amount of actual data written to the disk.

Snapshots: Snapshots are billed based on the size used. For example, if you create a snapshot of a managed disk with provisioned capacity of 64 GiB and actual used data size of 10 GiB, the snapshot is billed only for the used data size of 10 GiB.

For more information on snapshots, see the section on snapshots in the [managed disk overview](#).

Outbound data transfers: [Outbound data transfers](#) (data going out of Azure data centers) incur billing for bandwidth usage.

Transactions: You're billed for the number of transactions that you perform on a standard managed disk. For standard SSDs, each I/O operation less than or equal to 256 KiB of throughput is considered a single I/O operation. I/O operations larger than 256 KiB of throughput are considered multiple I/Os of size 256 KiB. For Standard HDDs, each IO operation is considered as a single transaction, regardless of the I/O size.

For detailed information on pricing for Managed Disks, including transaction costs, see [Managed Disks Pricing](#).

Ultra disk VM reservation fee

Azure VMs have the capability to indicate if they are compatible with ultra disks. An ultra disk compatible VM allocates dedicated bandwidth capacity between the compute VM instance and the block storage scale unit to optimize the performance and reduce latency. Adding this capability on the VM results in a reservation charge that is only imposed if you enabled ultra disk capability on the VM without attaching an ultra disk to it. When an ultra disk is attached to the ultra disk compatible VM, this charge would not be applied. This charge is per vCPU provisioned on the VM.

NOTE

For [constrained core VM sizes](#), the reservation fee is based on the actual number of vCPUs and not the constrained cores. For Standard_E32-8s_v3, the reservation fee will be based on 32 cores.

Refer to the [Azure Disks pricing page](#) for ultra disk pricing details.

Azure disk reservation

Disk reservation is the option to purchase one year of disk storage in advance at a discount, reducing your total cost. When purchasing a disk reservation, you select a specific Disk SKU in a target region, for example, 10 P30 (1TiB) premium SSDs in East US 2 region for a one year term. The reservation experience is similar to reserved virtual machine (VM) instances. You can bundle VM and Disk reservations to maximize your savings. For now, Azure Disks Reservation offers one year commitment plan for premium SSD SKUs from P30 (1TiB) to P80 (32 TiB) in all production regions. For more details on the Reserved Disks pricing, see [Azure Disks pricing page](#).

Azure premium storage: design for high performance

12/23/2019 • 37 minutes to read • [Edit Online](#)

This article provides guidelines for building high performance applications using Azure Premium Storage. You can use the instructions provided in this document combined with performance best practices applicable to technologies used by your application. To illustrate the guidelines, we have used SQL Server running on Premium Storage as an example throughout this document.

While we address performance scenarios for the Storage layer in this article, you will need to optimize the application layer. For example, if you are hosting a SharePoint Farm on Azure Premium Storage, you can use the SQL Server examples from this article to optimize the database server. Additionally, optimize the SharePoint Farm's Web server and Application server to get the most performance.

This article will help answer following common questions about optimizing application performance on Azure Premium Storage,

- How to measure your application performance?
- Why are you not seeing expected high performance?
- Which factors influence your application performance on Premium Storage?
- How do these factors influence performance of your application on Premium Storage?
- How can you optimize for IOPS, Bandwidth and Latency?

We have provided these guidelines specifically for Premium Storage because workloads running on Premium Storage are highly performance sensitive. We have provided examples where appropriate. You can also apply some of these guidelines to applications running on IaaS VMs with Standard Storage disks.

NOTE

Sometimes, what appears to be a disk performance issue is actually a network bottleneck. In these situations, you should optimize your [network performance](#).

If you are looking to benchmark your disk, see our article on [Benchmarking a disk](#).

If your VM supports accelerated networking, you should make sure it is enabled. If it is not enabled, you can enable it on already deployed VMs on both [Windows](#) and [Linux](#).

Before you begin, if you are new to Premium Storage, first read the [Select an Azure disk type for IaaS VMs](#) and [Scalability targets for premium page blob storage accounts](#).

Application performance indicators

We assess whether an application is performing well or not using performance indicators like, how fast an application is processing a user request, how much data an application is processing per request, how many requests an application processing in a specific period of time, how long a user has to wait to get a response after submitting their request. The technical terms for these performance indicators are, IOPS, Throughput or Bandwidth, and Latency.

In this section, we will discuss the common performance indicators in the context of Premium Storage. In the following section, Gathering Application Requirements, you will learn how to measure these performance indicators for your application. Later in Optimizing Application Performance, you will learn about the factors affecting these performance indicators and recommendations to optimize them.

IOPS

IOPS, or Input/output Operations Per Second, is the number of requests that your application is sending to the storage disks in one second. An input/output operation could be read or write, sequential, or random. Online Transaction Processing (OLTP) applications like an online retail website need to process many concurrent user requests immediately. The user requests are insert and update intensive database transactions, which the application must process quickly. Therefore, OLTP applications require very high IOPS. Such applications handle millions of small and random IO requests. If you have such an application, you must design the application infrastructure to optimize for IOPS. In the later section, *Optimizing Application Performance*, we discuss in detail all the factors that you must consider to get high IOPS.

When you attach a premium storage disk to your high scale VM, Azure provisions for you a guaranteed number of IOPS as per the disk specification. For example, a P50 disk provisions 7500 IOPS. Each high scale VM size also has a specific IOPS limit that it can sustain. For example, a Standard GS5 VM has 80,000 IOPS limit.

Throughput

Throughput, or bandwidth is the amount of data that your application is sending to the storage disks in a specified interval. If your application is performing input/output operations with large IO unit sizes, it requires high throughput. Data warehouse applications tend to issue scan intensive operations that access large portions of data at a time and commonly perform bulk operations. In other words, such applications require higher throughput. If you have such an application, you must design its infrastructure to optimize for throughput. In the next section, we discuss in detail the factors you must tune to achieve this.

When you attach a premium storage disk to a high scale VM, Azure provisions throughput as per that disk specification. For example, a P50 disk provisions 250 MB per second disk throughput. Each high scale VM size also has a specific throughput limit that it can sustain. For example, Standard GS5 VM has a maximum throughput of 2,000 MB per second.

There is a relation between throughput and IOPS as shown in the formula below.

$$\text{IOPS} \times \text{IO Size} = \text{Throughput}$$

Therefore, it is important to determine the optimal throughput and IOPS values that your application requires. As you try to optimize one, the other also gets affected. In a later section, *Optimizing Application Performance*, we will discuss in more details about optimizing IOPS and Throughput.

Latency

Latency is the time it takes an application to receive a single request, send it to the storage disks and send the response to the client. This is a critical measure of an application's performance in addition to IOPS and Throughput. The Latency of a premium storage disk is the time it takes to retrieve the information for a request and communicate it back to your application. Premium Storage provides consistent low latencies. Premium Disks are designed to provide single-digit millisecond latencies for most IO operations. If you enable ReadOnly host caching on premium storage disks, you can get much lower read latency. We will discuss Disk Caching in more detail in later section on *Optimizing Application Performance*.

When you are optimizing your application to get higher IOPS and Throughput, it will affect the latency of your application. After tuning the application performance, always evaluate the latency of the application to avoid unexpected high latency behavior.

The following control plane operations on Managed Disks may involve movement of the Disk from one Storage location to another. This is orchestrated via background copy of data that can take several hours to complete,

typically less than 24 hours depending on the amount of data in the disks. During that time your application can experience higher than usual read latency as some reads can get redirected to the original location and can take longer to complete. There is no impact on write latency during this period.

- Update the storage type.
- Detach and attach a disk from one VM to another.
- Create a managed disk from a VHD.
- Create a managed disk from a snapshot.
- Convert unmanaged disks to managed disks.

Performance Application Checklist for disks

The first step in designing high-performance applications running on Azure Premium Storage is understanding the performance requirements of your application. After you have gathered performance requirements, you can optimize your application to achieve the most optimal performance.

In the previous section, we explained the common performance indicators, IOPS, Throughput, and Latency. You must identify which of these performance indicators are critical to your application to deliver the desired user experience. For example, high IOPS matters most to OLTP applications processing millions of transactions in a second. Whereas, high Throughput is critical for Data Warehouse applications processing large amounts of data in a second. Extremely low Latency is crucial for real-time applications like live video streaming websites.

Next, measure the maximum performance requirements of your application throughout its lifetime. Use the sample checklist below as a start. Record the maximum performance requirements during normal, peak, and off-hours workload periods. By identifying requirements for all workloads levels, you will be able to determine the overall performance requirement of your application. For example, the normal workload of an e-commerce website will be the transactions it serves during most days in a year. The peak workload of the website will be the transactions it serves during holiday season or special sale events. The peak workload is typically experienced for a limited period, but can require your application to scale two or more times its normal operation. Find out the 50 percentile, 90 percentile, and 99 percentile requirements. This helps filter out any outliers in the performance requirements and you can focus your efforts on optimizing for the right values.

Application performance requirements checklist

PERFORMANCE REQUIREMENTS	50 PERCENTILE	90 PERCENTILE	99 PERCENTILE
Max. Transactions per second			
% Read operations			
% Write operations			
% Random operations			
% Sequential operations			
IO request size			
Average Throughput			
Max. Throughput			

PERFORMANCE REQUIREMENTS	50 PERCENTILE	90 PERCENTILE	99 PERCENTILE
Min. Latency			
Average Latency			
Max. CPU			
Average CPU			
Max. Memory			
Average Memory			
Queue Depth			

NOTE

You should consider scaling these numbers based on expected future growth of your application. It is a good idea to plan for growth ahead of time, because it could be harder to change the infrastructure for improving performance later.

If you have an existing application and want to move to Premium Storage, first build the checklist above for the existing application. Then, build a prototype of your application on Premium Storage and design the application based on guidelines described in *Optimizing Application Performance* in a later section of this document. The next article describes the tools you can use to gather the performance measurements.

Counters to measure application performance requirements

The best way to measure performance requirements of your application, is to use performance-monitoring tools provided by the operating system of the server. You can use PerfMon for Windows and iostat for Linux. These tools capture counters corresponding to each measure explained in the above section. You must capture the values of these counters when your application is running its normal, peak, and off-hours workloads.

The PerfMon counters are available for processor, memory and, each logical disk and physical disk of your server. When you use premium storage disks with a VM, the physical disk counters are for each premium storage disk, and logical disk counters are for each volume created on the premium storage disks. You must capture the values for the disks that host your application workload. If there is a one to one mapping between logical and physical disks, you can refer to physical disk counters; otherwise refer to the logical disk counters. On Linux, the iostat command generates a CPU and disk utilization report. The disk utilization report provides statistics per physical device or partition. If you have a database server with its data and logs on separate disks, collect this data for both disks. Below table describes counters for disks, processors, and memory:

COUNTER	DESCRIPTION	PERFMON	IOSTAT
IOPS or Transactions per second	Number of I/O requests issued to the storage disk per second.	Disk Reads/sec Disk Writes/sec	tps r/s w/s
Disk Reads and Writes	% of Reads and Write operations performed on the disk.	% Disk Read Time % Disk Write Time	r/s w/s

COUNTER	DESCRIPTION	PERFMON	IOSTAT
Throughput	Amount of data read from or written to the disk per second.	Disk Read Bytes/sec Disk Write Bytes/sec	kB_read/s kB_wrtn/s
Latency	Total time to complete a disk IO request.	Average Disk sec/Read Average disk sec/Write	await svctm
IO size	The size of I/O requests issued to the storage disks.	Average Disk Bytes/Read Average Disk Bytes/Write	avgrq-sz
Queue Depth	Number of outstanding I/O requests waiting to be read from or written to the storage disk.	Current Disk Queue Length	avgqu-sz
Max. Memory	Amount of memory required to run application smoothly	% Committed Bytes in Use	Use vmstat
Max. CPU	Amount CPU required to run application smoothly	% Processor time	%util

Learn more about [iostat](#) and [PerfMon](#).

Optimize application performance

The main factors that influence performance of an application running on Premium Storage are Nature of IO requests, VM size, Disk size, Number of disks, disk caching, multithreading, and queue depth. You can control some of these factors with knobs provided by the system. Most applications may not give you an option to alter the IO size and Queue Depth directly. For example, if you are using SQL Server, you cannot choose the IO size and queue depth. SQL Server chooses the optimal IO size and queue depth values to get the most performance. It is important to understand the effects of both types of factors on your application performance, so that you can provision appropriate resources to meet performance needs.

Throughout this section, refer to the application requirements checklist that you created, to identify how much you need to optimize your application performance. Based on that, you will be able to determine which factors from this section you will need to tune. To witness the effects of each factor on your application performance, run benchmarking tools on your application setup. Refer to the Benchmarking article, linked at the end, for steps to run common benchmarking tools on Windows and Linux VMs.

Optimize IOPS, throughput, and latency at a glance

The table below summarizes performance factors and the steps necessary to optimize IOPS, throughput, and latency. The sections following this summary will describe each factor in much more depth.

For more information on VM sizes and on the IOPS, throughput, and latency available for each type of VM, see [Linux VM sizes](#) or [Windows VM sizes](#).

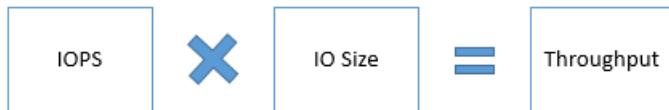
	IOPS	THROUGHPUT	LATENCY
Example Scenario	Enterprise OLTP application requiring very high transactions per second rate.	Enterprise Data warehousing application processing large amounts of data.	Near real-time applications requiring instant responses to user requests, like online gaming.

	IOPS	THROUGHPUT	LATENCY
Performance factors			
IO size	Smaller IO size yields higher IOPS.	Larger IO size to yields higher Throughput.	
VM size	Use a VM size that offers IOPS greater than your application requirement.	Use a VM size with throughput limit greater than your application requirement.	Use a VM size that offers scale limits greater than your application requirement.
Disk size	Use a disk size that offers IOPS greater than your application requirement.	Use a disk size with Throughput limit greater than your application requirement.	Use a disk size that offers scale limits greater than your application requirement.
VM and Disk Scale Limits	IOPS limit of the VM size chosen should be greater than total IOPS driven by storage disks attached to it.	Throughput limit of the VM size chosen should be greater than total Throughput driven by premium storage disks attached to it.	Scale limits of the VM size chosen must be greater than total scale limits of attached premium storage disks.
Disk Caching	Enable ReadOnly Cache on premium storage disks with Read heavy operations to get higher Read IOPS.		Enable ReadOnly Cache on premium storage disks with Ready heavy operations to get very low Read latencies.
Disk Striping	Use multiple disks and stripe them together to get a combined higher IOPS and Throughput limit. The combined limit per VM should be higher than the combined limits of attached premium disks.		
Stripe Size	Smaller stripe size for random small IO pattern seen in OLTP applications. For example, use stripe size of 64 KB for SQL Server OLTP application.	Larger stripe size for sequential large IO pattern seen in Data Warehouse applications. For example, use 256 KB stripe size for SQL Server Data warehouse application.	
Multithreading	Use multithreading to push higher number of requests to Premium Storage that will lead to higher IOPS and Throughput. For example, on SQL Server set a high MAXDOP value to allocate more CPUs to SQL Server.		
Queue Depth	Larger Queue Depth yields higher IOPS.	Larger Queue Depth yields higher Throughput.	Smaller Queue Depth yields lower latencies.

Nature of IO requests

An IO request is a unit of input/output operation that your application will be performing. Identifying the nature of IO requests, random or sequential, read or write, small or large, will help you determine the performance requirements of your application. It is important to understand the nature of IO requests, to make the right decisions when designing your application infrastructure. IOs must be distributed evenly to achieve the best performance possible.

IO size is one of the more important factors. The IO size is the size of the input/output operation request generated by your application. The IO size has a significant impact on performance especially on the IOPS and Bandwidth that the application is able to achieve. The following formula shows the relationship between IOPS, IO size, and Bandwidth/Throughput.



Some applications allow you to alter their IO size, while some applications do not. For example, SQL Server determines the optimal IO size itself, and does not provide users with any knobs to change it. On the other hand, Oracle provides a parameter called [DB_BLOCK_SIZE](#) using which you can configure the I/O request size of the database.

If you are using an application, which does not allow you to change the IO size, use the guidelines in this article to optimize the performance KPI that is most relevant to your application. For example,

- An OLTP application generates millions of small and random IO requests. To handle these types of IO requests, you must design your application infrastructure to get higher IOPS.
- A data warehousing application generates large and sequential IO requests. To handle these types of IO requests, you must design your application infrastructure to get higher Bandwidth or Throughput.

If you are using an application, which allows you to change the IO size, use this rule of thumb for the IO size in addition to other performance guidelines,

- Smaller IO size to get higher IOPS. For example, 8 KB for an OLTP application.
- Larger IO size to get higher Bandwidth/Throughput. For example, 1024 KB for a data warehouse application.

Here is an example on how you can calculate the IOPS and Throughput/Bandwidth for your application. Consider an application using a P30 disk. The maximum IOPS and Throughput/Bandwidth a P30 disk can achieve is 5000 IOPS and 200 MB per second respectively. Now, if your application requires the maximum IOPS from the P30 disk and you use a smaller IO size like 8 KB, the resulting Bandwidth you will be able to get is 40 MB per second. However, if your application requires the maximum Throughput/Bandwidth from P30 disk, and you use a larger IO size like 1024 KB, the resulting IOPS will be less, 200 IOPS. Therefore, tune the IO size such that it meets both your application's IOPS and Throughput/Bandwidth requirement. The following table summarizes the different IO sizes and their corresponding IOPS and Throughput for a P30 disk.

APPLICATION REQUIREMENT	I/O SIZE	IOPS	THROUGHPUT/BANDWIDTH
Max IOPS	8 KB	5,000	40 MB per second
Max Throughput	1024 KB	200	200 MB per second
Max Throughput + high IOPS	64 KB	3,200	200 MB per second
Max IOPS + high Throughput	32 KB	5,000	160 MB per second

To get IOPS and Bandwidth higher than the maximum value of a single premium storage disk, use multiple premium disks striped together. For example, stripe two P30 disks to get a combined IOPS of 10,000 IOPS or a combined Throughput of 400 MB per second. As explained in the next section, you must use a VM size that supports the combined disk IOPS and Throughput.

NOTE

As you increase either IOPS or Throughput the other also increases, make sure you do not hit throughput or IOPS limits of the disk or VM when increasing either one.

To witness the effects of IO size on application performance, you can run benchmarking tools on your VM and disks. Create multiple test runs and use different IO size for each run to see the impact. Refer to the Benchmarking article, linked at the end, for more details.

High scale VM sizes

When you start designing an application, one of the first things to do is, choose a VM to host your application. Premium Storage comes with High Scale VM sizes that can run applications requiring higher compute power and a high local disk I/O performance. These VMs provide faster processors, a higher memory-to-core ratio, and a Solid-State Drive (SSD) for the local disk. Examples of High Scale VMs supporting Premium Storage are the DS and GS series VMs.

High Scale VMs are available in different sizes with a different number of CPU cores, memory, OS, and temporary disk size. Each VM size also has maximum number of data disks that you can attach to the VM. Therefore, the chosen VM size will affect how much processing, memory, and storage capacity is available for your application. It also affects the Compute and Storage cost. For example, below are the specifications of the largest VM size in a DS series and a GS series:

VM SIZE	CPU CORES	MEMORY	VM DISK SIZES	MAX. DATA DISKS	CACHE SIZE	IOPS	BANDWIDTH CACHE IO LIMITS
Standard_D S14	16	112 GB	OS = 1023 GB Local SSD = 224 GB	32	576 GB	50,000 IOPS 512 MB per second	4,000 IOPS and 33 MB per second
Standard_G S5	32	448 GB	OS = 1023 GB Local SSD = 896 GB	64	4224 GB	80,000 IOPS 2,000 MB per second	5,000 IOPS and 50 MB per second

To view a complete list of all available Azure VM sizes, refer to [Windows VM sizes](#) or [Linux VM sizes](#). Choose a VM size that can meet and scale to your desired application performance requirements. In addition to this, take into account following important considerations when choosing VM sizes.

Scale Limits

The maximum IOPS limits per VM and per disk are different and independent of each other. Make sure that the application is driving IOPS within the limits of the VM as well as the premium disks attached to it. Otherwise, application performance will experience throttling.

As an example, suppose an application requirement is a maximum of 4,000 IOPS. To achieve this, you provision a P30 disk on a DS1 VM. The P30 disk can deliver up to 5,000 IOPS. However, the DS1 VM is limited to 3,200 IOPS. Consequently, the application performance will be constrained by the VM limit at 3,200 IOPS and there will be degraded performance. To prevent this situation, choose a VM and disk size that will both meet application requirements.

Cost of Operation

In many cases, it is possible that your overall cost of operation using Premium Storage is lower than using Standard Storage.

For example, consider an application requiring 16,000 IOPS. To achieve this performance, you will need a Standard_D14 Azure IaaS VM, which can give a maximum IOPS of 16,000 using 32 standard storage 1 TB disks. Each 1-TB standard storage disk can achieve a maximum of 500 IOPS. The estimated cost of this VM per month will be \$1,570. The monthly cost of 32 standard storage disks will be \$1,638. The estimated total monthly cost will be \$3,208.

However, if you hosted the same application on Premium Storage, you will need a smaller VM size and fewer premium storage disks, thus reducing the overall cost. A Standard_DS13 VM can meet the 16,000 IOPS requirement using four P30 disks. The DS13 VM has a maximum IOPS of 25,600 and each P30 disk has a maximum IOPS of 5,000. Overall, this configuration can achieve $5,000 \times 4 = 20,000$ IOPS. The estimated cost of this VM per month will be \$1,003. The monthly cost of four P30 premium storage disks will be \$544.34. The estimated total monthly cost will be \$1,544.

Table below summarizes the cost breakdown of this scenario for Standard and Premium Storage.

	STANDARD	PREMIUM
Cost of VM per month	\$1,570.58 (Standard_D14)	\$1,003.66 (Standard_DS13)
Cost of Disks per month	\$1,638.40 (32 x 1-TB disks)	\$544.34 (4 x P30 disks)
Overall Cost per month	\$3,208.98	\$1,544.34

Linux Distros

With Azure Premium Storage, you get the same level of Performance for VMs running Windows and Linux. We support many flavors of Linux distros, and you can see the complete list [here](#). It is important to note that different distros are better suited for different types of workloads. You will see different levels of performance depending on the distro your workload is running on. Test the Linux distros with your application and choose the one that works best.

When running Linux with Premium Storage, check the latest updates about required drivers to ensure high performance.

Premium storage disk sizes

Azure Premium Storage offers a variety of sizes so you can choose one that best suits your needs. Each disk size has a different scale limit for IOPS, bandwidth, and storage. Choose the right Premium Storage Disk size depending on the application requirements and the high scale VM size. The table below shows the disks sizes and their capabilities. P4, P6, P15, P60, P70, and P80 sizes are currently only supported for Managed Disks.

PRE MIU M SSD SIZE S	P1*	P2*	P3*	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
Disk size in GiB	4	8	16	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384	32,767

PRE MIU M SSD SIZE S	P1*	P2*	P3*	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
IOP S per disk	120	120	120	120	240	500	1,10 0	2,30 0	5,00 0	7,50 0	7,50 0	16,0 00	18,0 00	20,0 00
Thr oug hpu t per disk	25 MiB /sec	25 MiB /sec	25 MiB /sec	25 MiB /sec	50 MiB /sec	100 MiB /sec	125 MiB /sec	150 MiB /sec	200 MiB /sec	250 MiB /sec	250 MiB /sec	500 MiB /sec	750 MiB /sec	900 MiB /sec
Max bur st IOP S per disk **	3,5 00	3,5 00	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0						
Max bur st thro ugh put per disk **	170 MiB /sec	170 MiB /sec	170 MiB /sec	170 MiB /sec	170 MiB /sec	170 MiB /sec								
Max bur st dur at io n**	30 min	30 min	30 min	30 min	30 min	30 min								
Elig ible for rese rvat ion	No	Yes, up to one year	Yes, up to one year	Yes, up to one year	Yes, up to one year	Yes, up to one year	Yes, up to one year							

*Denotes a disk size that is currently in preview, for regional availability information see [New disk sizes: Managed and unmanaged](#).

**Denotes a feature that is currently in preview, see [Disk bursting](#) for more information.

How many disks you choose depends on the disk size chosen. You could use a single P50 disk or multiple P10 disks to meet your application requirement. Take into account considerations listed below when making the choice.

Scale Limits (IOPS and Throughput)

The IOPS and Throughput limits of each Premium disk size is different and independent from the VM scale limits.

Make sure that the total IOPS and Throughput from the disks are within scale limits of the chosen VM size.

For example, if an application requirement is a maximum of 250 MB/sec Throughput and you are using a DS4 VM with a single P30 disk. The DS4 VM can give up to 256 MB/sec Throughput. However, a single P30 disk has Throughput limit of 200 MB/sec. Consequently, the application will be constrained at 200 MB/sec due to the disk limit. To overcome this limit, provision more than one data disks to the VM or resize your disks to P40 or P50.

NOTE

Reads served by the cache are not included in the disk IOPS and Throughput, hence not subject to disk limits. Cache has its separate IOPS and Throughput limit per VM.

For example, initially your reads and writes are 60MB/sec and 40MB/sec respectively. Over time, the cache warms up and serves more and more of the reads from the cache. Then, you can get higher write Throughput from the disk.

Number of Disks

Determine the number of disks you will need by assessing application requirements. Each VM size also has a limit on the number of disks that you can attach to the VM. Typically, this is twice the number of cores. Ensure that the VM size you choose can support the number of disks needed.

Remember, the Premium Storage disks have higher performance capabilities compared to Standard Storage disks. Therefore, if you are migrating your application from Azure IaaS VM using Standard Storage to Premium Storage, you will likely need fewer premium disks to achieve the same or higher performance for your application.

Disk caching

High Scale VMs that leverage Azure Premium Storage have a multi-tier caching technology called BlobCache. BlobCache uses a combination of the Virtual Machine RAM and local SSD for caching. This cache is available for the Premium Storage persistent disks and the VM local disks. By default, this cache setting is set to Read/Write for OS disks and ReadOnly for data disks hosted on Premium Storage. With disk caching enabled on the Premium Storage disks, the high scale VMs can achieve extremely high levels of performance that exceed the underlying disk performance.

WARNING

Disk Caching is not supported for disks 4 TiB and larger. If multiple disks are attached to your VM, each disk that is smaller than 4 TiB will support caching.

Changing the cache setting of an Azure disk detaches and re-attaches the target disk. If it is the operating system disk, the VM is restarted. Stop all applications/services that might be affected by this disruption before changing the disk cache setting.

To learn more about how BlobCache works, refer to the Inside [Azure Premium Storage](#) blog post.

It is important to enable cache on the right set of disks. Whether you should enable disk caching on a premium disk or not will depend on the workload pattern that disk will be handling. Table below shows the default cache settings for OS and Data disks.

DISK TYPE	DEFAULT CACHE SETTING
OS disk	ReadWrite
Data disk	ReadOnly

Following are the recommended disk cache settings for data disks,

DISK CACHING SETTING	RECOMMENDATION ON WHEN TO USE THIS SETTING
None	Configure host-cache as None for write-only and write-heavy disks.
ReadOnly	Configure host-cache as ReadOnly for read-only and read-write disks.
ReadWrite	Configure host-cache as ReadWrite only if your application properly handles writing cached data to persistent disks when needed.

ReadOnly

By configuring ReadOnly caching on Premium Storage data disks, you can achieve low Read latency and get very high Read IOPS and Throughput for your application. This is due two reasons,

1. Reads performed from cache, which is on the VM memory and local SSD, are much faster than reads from the data disk, which is on the Azure blob storage.
2. Premium Storage does not count the Reads served from cache, towards the disk IOPS and Throughput.
Therefore, your application is able to achieve higher total IOPS and Throughput.

ReadWrite

By default, the OS disks have ReadWrite caching enabled. We have recently added support for ReadWrite caching on data disks as well. If you are using ReadWrite caching, you must have a proper way to write the data from cache to persistent disks. For example, SQL Server handles writing cached data to the persistent storage disks on its own. Using ReadWrite cache with an application that does not handle persisting the required data can lead to data loss, if the VM crashes.

None

Currently, **None** is only supported on data disks. It is not supported on OS disks. If you set **None** on an OS disk it will override this internally and set it to **ReadOnly**.

As an example, you can apply these guidelines to SQL Server running on Premium Storage by doing the following,

1. Configure "ReadOnly" cache on premium storage disks hosting data files.
 - a. The fast reads from cache lower the SQL Server query time since data pages are retrieved much faster from the cache compared to directly from the data disks.
 - b. Serving reads from cache, means there is additional Throughput available from premium data disks. SQL Server can use this additional Throughput towards retrieving more data pages and other operations like backup/restore, batch loads, and index rebuilds.
2. Configure "None" cache on premium storage disks hosting the log files.
 - a. Log files have primarily write-heavy operations. Therefore, they do not benefit from the ReadOnly cache.

Optimize performance on Linux VMs

For all premium SSDs or ultra disks with cache set to **ReadOnly** or **None**, you must disable "barriers" when you mount the file system. You don't need barriers in this scenario because the writes to premium storage disks are durable for these cache settings. When the write request successfully finishes, data has been written to the persistent store. To disable "barriers," use one of the following methods. Choose the one for your file system:

- For **reiserFS**, to disable barriers, use the `barrier=none` mount option. (To enable barriers, use `barrier=flush`.)
- For **ext3/ext4**, to disable barriers, use the `barrier=0` mount option. (To enable barriers, use `barrier=1`.)
- For **XFS**, to disable barriers, use the `nobarrier` mount option. (To enable barriers, use `barrier`.)
- For premium storage disks with cache set to **ReadWrite**, enable barriers for write durability.
- For volume labels to persist after you restart the VM, you must update `/etc/fstab` with the universally unique

identifier (UUID) references to the disks. For more information, see [Add a managed disk to a Linux VM](#).

The following Linux distributions have been validated for premium SSDs. For better performance and stability with premium SSDs, we recommend that you upgrade your VMs to one of these versions or newer.

Some of the versions require the latest Linux Integration Services (LIS), v4.0, for Azure. To download and install a distribution, follow the link listed in the following table. We add images to the list as we complete validation. Our validations show that performance varies for each image. Performance depends on workload characteristics and your image settings. Different images are tuned for different kinds of workloads.

DISTRIBUTION	VERSION	SUPPORTED KERNEL	DETAILS
Ubuntu	12.04 or newer	3.2.0-75.110+	
Ubuntu	14.04 or newer	3.13.0-44.73+	
Debian	7.x, 8.x or newer	3.16.7-ckt4-1+	
SUSE	SLES 12 or newer	3.12.36-38.1+	
SUSE	SLES 11 SP4 or newer	3.0.101-0.63.1+	
CoreOS	584.0.0+ or newer	3.18.4+	
CentOS	6.5, 6.6, 6.7, 7.0, or newer		LIS4 required <i>See note in the next section</i>
CentOS	7.1+ or newer	3.10.0-229.1.2.el7+	LIS4 recommended <i>See note in the next section</i>
Red Hat Enterprise Linux (RHEL)	6.8+, 7.2+, or newer		
Oracle	6.0+, 7.2+, or newer		UEK4 or RHCK
Oracle	7.0-7.1 or newer		UEK4 or RHCK w/ LIS4
Oracle	6.4-6.7 or newer		UEK4 or RHCK w/ LIS4

LIS drivers for OpenLogic CentOS

If you're running OpenLogic CentOS VMs, run the following command to install the latest drivers:

```
sudo yum remove hypervkvpd ## (Might return an error if not installed. That's OK.)  
sudo yum install microsoft-hyper-v  
sudo reboot
```

In some cases the command above will upgrade the kernel as well. If a kernel update is required then you may need to run the above commands again after rebooting to fully install the microsoft-hyper-v package.

Disk striping

When a high scale VM is attached with several premium storage persistent disks, the disks can be striped together to aggregate their IOPs, bandwidth, and storage capacity.

On Windows, you can use Storage Spaces to stripe disks together. You must configure one column for each disk in a pool. Otherwise, the overall performance of striped volume can be lower than expected, due to uneven distribution of traffic across the disks.

Important: Using Server Manager UI, you can set the total number of columns up to 8 for a striped volume. When attaching more than eight disks, use PowerShell to create the volume. Using PowerShell, you can set the number of columns equal to the number of disks. For example, if there are 16 disks in a single stripe set; specify 16 columns in the *NumberOfColumns* parameter of the *New-VirtualDisk* PowerShell cmdlet.

On Linux, use the MDADM utility to stripe disks together. For detailed steps on striping disks on Linux refer to [Configure Software RAID on Linux](#).

Stripe Size

An important configuration in disk striping is the stripe size. The stripe size or block size is the smallest chunk of data that application can address on a striped volume. The stripe size you configure depends on the type of application and its request pattern. If you choose the wrong stripe size, it could lead to IO misalignment, which leads to degraded performance of your application.

For example, if an IO request generated by your application is bigger than the disk stripe size, the storage system writes it across stripe unit boundaries on more than one disk. When it is time to access that data, it will have to seek across more than one stripe units to complete the request. The cumulative effect of such behavior can lead to substantial performance degradation. On the other hand, if the IO request size is smaller than stripe size, and if it is random in nature, the IO requests may add up on the same disk causing a bottleneck and ultimately degrading the IO performance.

Depending on the type of workload your application is running, choose an appropriate stripe size. For random small IO requests, use a smaller stripe size. Whereas for large sequential IO requests use a larger stripe size. Find out the stripe size recommendations for the application you will be running on Premium Storage. For SQL Server, configure stripe size of 64 KB for OLTP workloads and 256 KB for data warehousing workloads. See [Performance best practices for SQL Server on Azure VMs](#) to learn more.

NOTE

You can stripe together a maximum of 32 premium storage disks on a DS series VM and 64 premium storage disks on a GS series VM.

Multi-threading

Azure has designed Premium Storage platform to be massively parallel. Therefore, a multi-threaded application achieves much higher performance than a single-threaded application. A multi-threaded application splits up its tasks across multiple threads and increases efficiency of its execution by utilizing the VM and disk resources to the maximum.

For example, if your application is running on a single core VM using two threads, the CPU can switch between the two threads to achieve efficiency. While one thread is waiting on a disk IO to complete, the CPU can switch to the other thread. In this way, two threads can accomplish more than a single thread would. If the VM has more than one core, it further decreases running time since each core can execute tasks in parallel.

You may not be able to change the way an off-the-shelf application implements single threading or multi-threading. For example, SQL Server is capable of handling multi-CPU and multi-core. However, SQL Server decides under what conditions it will leverage one or more threads to process a query. It can run queries and build indexes using multi-threading. For a query that involves joining large tables and sorting data before returning to the user, SQL Server will likely use multiple threads. However, a user cannot control whether SQL Server executes a query using a single thread or multiple threads.

There are configuration settings that you can alter to influence this multi-threading or parallel processing of an application. For example, in case of SQL Server it is the maximum Degree of Parallelism configuration. This setting called MAXDOP, allows you to configure the maximum number of processors SQL Server can use when parallel processing. You can configure MAXDOP for individual queries or index operations. This is beneficial when you want to balance resources of your system for a performance critical application.

For example, say your application using SQL Server is executing a large query and an index operation at the same time. Let us assume that you wanted the index operation to be more performant compared to the large query. In such a case, you can set MAXDOP value of the index operation to be higher than the MAXDOP value for the query. This way, SQL Server has more number of processors that it can leverage for the index operation compared to the number of processors it can dedicate to the large query. Remember, you do not control the number of threads SQL Server will use for each operation. You can control the maximum number of processors being dedicated for multi-threading.

Learn more about [Degrees of Parallelism](#) in SQL Server. Find out such settings that influence multi-threading in your application and their configurations to optimize performance.

Queue depth

The queue depth or queue length or queue size is the number of pending IO requests in the system. The value of queue depth determines how many IO operations your application can line up, which the storage disks will be processing. It affects all the three application performance indicators that we discussed in this article viz., IOPS, throughput, and latency.

Queue Depth and multi-threading are closely related. The Queue Depth value indicates how much multi-threading can be achieved by the application. If the Queue Depth is large, application can execute more operations concurrently, in other words, more multi-threading. If the Queue Depth is small, even though application is multi-threaded, it will not have enough requests lined up for concurrent execution.

Typically, off the shelf applications do not allow you to change the queue depth, because if set incorrectly it will do more harm than good. Applications will set the right value of queue depth to get the optimal performance. However, it is important to understand this concept so that you can troubleshoot performance issues with your application. You can also observe the effects of queue depth by running benchmarking tools on your system.

Some applications provide settings to influence the Queue Depth. For example, the MAXDOP (maximum degree of parallelism) setting in SQL Server explained in previous section. MAXDOP is a way to influence Queue Depth and multi-threading, although it does not directly change the Queue Depth value of SQL Server.

High queue depth

A high queue depth lines up more operations on the disk. The disk knows the next request in its queue ahead of time. Consequently, the disk can schedule operations ahead of time and process them in an optimal sequence. Since the application is sending more requests to the disk, the disk can process more parallel IOs. Ultimately, the application will be able to achieve higher IOPS. Since application is processing more requests, the total Throughput of the application also increases.

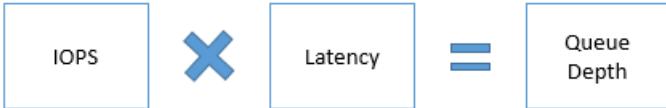
Typically, an application can achieve maximum Throughput with 8-16+ outstanding IOs per attached disk. If a queue depth is one, application is not pushing enough IOs to the system, and it will process less amount of in a given period. In other words, less Throughput.

For example, in SQL Server, setting the MAXDOP value for a query to "4" informs SQL Server that it can use up to four cores to execute the query. SQL Server will determine what is best queue depth value and the number of cores for the query execution.

Optimal queue depth

Very high queue depth value also has its drawbacks. If queue depth value is too high, the application will try to drive very high IOPS. Unless application has persistent disks with sufficient provisioned IOPS, this can negatively

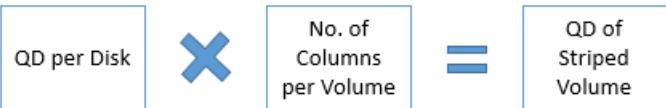
affect application latencies. Following formula shows the relationship between IOPS, latency, and queue depth.



You should not configure Queue Depth to any high value, but to an optimal value, which can deliver enough IOPS for the application without affecting latencies. For example, if the application latency needs to be 1 millisecond, the Queue Depth required to achieve 5,000 IOPS is, $QD = 5000 \times 0.001 = 5$.

Queue Depth for Striped Volume

For a striped volume, maintain a high enough queue depth such that, every disk has a peak queue depth individually. For example, consider an application that pushes a queue depth of 2 and there are four disks in the stripe. The two IO requests will go to two disks and remaining two disks will be idle. Therefore, configure the queue depth such that all the disks can be busy. Formula below shows how to determine the queue depth of striped volumes.



Throttling

Azure Premium Storage provisions specified number of IOPS and Throughput depending on the VM sizes and disk sizes you choose. Anytime your application tries to drive IOPS or Throughput above these limits of what the VM or disk can handle, Premium Storage will throttle it. This manifests in the form of degraded performance in your application. This can mean higher latency, lower Throughput, or lower IOPS. If Premium Storage does not throttle, your application could completely fail by exceeding what its resources are capable of achieving. So, to avoid performance issues due to throttling, always provision sufficient resources for your application. Take into consideration what we discussed in the VM sizes and Disk sizes sections above. Benchmarking is the best way to figure out what resources you will need to host your application.

Next steps

If you are looking to benchmark your disk, see our article on [Benchmarking a disk](#).

Learn more about the available disk types: [Select a disk type](#)

For SQL Server users, read articles on Performance Best Practices for SQL Server:

- [Performance Best Practices for SQL Server in Azure Virtual Machines](#)
- [Azure Premium Storage provides highest performance for SQL Server in Azure VM](#)

Ephemeral OS disks for VM instances

1/19/2020 • 6 minutes to read • [Edit Online](#)

Ephemeral OS disks are created on the local virtual machine (VM) storage and not saved to the remote Azure Storage. Ephemeral OS disks work well for stateless workloads, where applications are tolerant of individual VM failures, but are more affected by VM deployment time or reimaging the individual VM instances. With Ephemeral OS disk, you get lower read/write latency to the OS disk and faster VM reimage.

The key features of ephemeral disks are:

- Ideal for stateless applications.
- They can be used with both Marketplace and custom images.
- Ability to fast reset or reimage VMs and scale set instances to the original boot state.
- Lower latency, similar to a temporary disk.
- Ephemeral OS disks are free, you incur no storage cost for OS disk.
- They are available in all Azure regions.
- Ephemeral OS Disk is supported by [Shared Image Gallery](#).

Key differences between persistent and ephemeral OS disks:

	PERSISTENT OS DISK	EPHEMERAL OS DISK
Size limit for OS disk	2 TiB	Cache size for the VM size or 2TiB, whichever is smaller. For the cache size in GiB , see DS , ES , M , FS , and GS
VM sizes supported	All	DSv1, DSv2, DSv3, Esv3, Fs, FsV2, GS, M
Disk type support	Managed and unmanaged OS disk	Managed OS disk only
Region support	All regions	All regions
Data persistence	OS disk data written to OS disk are stored in Azure Storage	Data written to OS disk is stored to the local VM storage and is not persisted to Azure Storage.
Stop-deallocated state	VMs and scale set instances can be stop-deallocated and restarted from the stop-deallocated state	VMs and scale set instances cannot be stop-deallocated
Specialized OS disk support	Yes	No
OS disk resize	Supported during VM creation and after VM is stop-deallocated	Supported during VM creation only

	PERSISTENT OS DISK	EPHEMERAL OS DISK
Resizing to a new VM size	OS disk data is preserved	Data on the OS disk is deleted, OS is re-provisioned

Size requirements

You can deploy VM and instance images up to the size of the VM cache. For example, Standard Windows Server images from the marketplace are about 127 GiB, which means that you need a VM size that has a cache larger than 127 GiB. In this case, the [Standard_DS2_v2](#) has a cache size of 86 GiB, which is not large enough. The Standard_DS3_v2 has a cache size of 172 GiB, which is large enough. In this case, the Standard_DS3_v2 is the smallest size in the DSv2 series that you can use with this image. Basic Linux images in the Marketplace and Windows Server images that are denoted by `[smallsize]` tend to be around 30 GiB and can use most of the available VM sizes.

Ephemeral disks also require that the VM size supports Premium storage. The sizes usually (but not always) have an `s` in the name, like DSv2 and EsV3. For more information, see [Azure VM sizes](#) for details around which sizes support Premium storage.

PowerShell

To use an ephemeral disk for a PowerShell VM deployment, use [Set-AzVMOSDisk](#) in your VM configuration. Set the `-DiffDiskSetting` to `Local` and `-Caching` to `ReadOnly`.

```
Set-AzVMOSDisk -DiffDiskSetting Local -Caching ReadOnly
```

For scale set deployments, use the [Set-AzVmssStorageProfile](#) cmdlet in your configuration. Set the `-DiffDiskSetting` to `Local` and `-Caching` to `ReadOnly`.

```
Set-AzVmssStorageProfile -DiffDiskSetting Local -OsDiskCaching ReadOnly
```

CLI

To use an ephemeral disk for a CLI VM deployment, set the `--ephemeral-os-disk` parameter in [az vm create](#) to `true` and the `--os-disk-caching` parameter to `ReadOnly`.

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--image UbuntuLTS \
--ephemeral-os-disk true \
--os-disk-caching ReadOnly \
--admin-username azureuser \
--generate-ssh-keys
```

For scale sets, you use the same `--ephemeral-os-disk true` parameter for [az-vmss-create](#) and set the `--os-disk-caching` parameter to `ReadOnly`.

Portal

In the Azure portal, you can choose to use ephemeral disks when deploying a VM by opening the **Advanced** section of the **Disks** tab. For **Use ephemeral OS disk** select **Yes**.

Home > New > Create a virtual machine

Create a virtual machine

Basics **Disks** **Networking** **Management** **Advanced** **Tags** **Review + create**

Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. [Learn more](#)

Disk options

* OS disk type Ephemeral OS Disks only support the Standard HDD disk type.

Enable Ultra SSD compatibility (Preview) Yes No Ultra SSD disks are not available when using ephemeral disks.

Data disks

You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.

LUN	NAME	SIZE (GiB)	DISK TYPE	HOST CACHING

[Create and attach a new disk](#) [Attach an existing disk](#)

Advanced

Use managed disks No Yes

Use ephemeral OS disk No Yes

If the option for using an ephemeral disk is greyed out, you might have selected a VM size that does not have a cache size larger than the OS image or that doesn't support Premium storage. Go back to the **Basics** page and try choosing another VM size.

You can also create scale-sets with ephemeral OS disks using the portal. Just make sure you select a VM size with a large enough cache size and then in **Use ephemeral OS disk** select **Yes**.

INSTANCES

* Instance count

* Instance size **Standard DS3 v2**
4 vcpus, 14 GiB memory [Change size](#)

Deploy as low priority (preview) No Yes

Use managed disks No Yes

Use ephemeral OS disk No Yes

Scale set template deployment

The process to create a scale set that uses an ephemeral OS disk is to add the `diffDiskSettings` property to the `Microsoft.Compute/virtualMachineScaleSets/virtualMachineProfile` resource type in the template. Also, the caching policy must be set to `ReadOnly` for the ephemeral OS disk.

```
{
  "type": "Microsoft.Compute/virtualMachineScaleSets",
  "name": "myScaleSet",
  "location": "East US 2",
  "apiVersion": "2018-06-01",
  "sku": {
    "name": "Standard_DS2_v2",
    "capacity": "2"
  },
  "properties": {
    "upgradePolicy": {
      "mode": "Automatic"
    },
    "virtualMachineProfile": {
      "storageProfile": {
        "osDisk": {
          "diffDiskSettings": {
            "option": "Local"
          },
          "caching": "ReadOnly",
          "createOption": "FromImage"
        },
        "imageReference": {
          "publisher": "Canonical",
          "offer": "UbuntuServer",
          "sku": "16.04-LTS",
          "version": "latest"
        }
      },
      "osProfile": {
        "computerNamePrefix": "myvmss",
        "adminUsername": "azureuser",
        "adminPassword": "P@ssw0rd!"
      }
    }
  }
}
```

VM template deployment

You can deploy a VM with an ephemeral OS disk using a template. The process to create a VM that uses ephemeral OS disks is to add the `diffDiskSettings` property to the `Microsoft.Compute/virtualMachines` resource type in the template. Also, the caching policy must be set to `ReadOnly` for the ephemeral OS disk.

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "name": "myVirtualMachine",
  "location": "East US 2",
  "apiVersion": "2018-06-01",
  "properties": {
    "storageProfile": {
      "osDisk": {
        "diffDiskSettings": {
          "option": "Local"
        },
        "caching": "ReadOnly",
        "createOption": "FromImage"
      },
      "imageReference": {
        "publisher": "MicrosoftWindowsServer",
        "offer": "WindowsServer",
        "sku": "2016-Datacenter-smalldisk",
        "version": "latest"
      },
      "hardwareProfile": {
        "vmSize": "Standard_DS2_v2"
      }
    },
    "osProfile": {
      "computerNamePrefix": "myvirtualmachine",
      "adminUsername": "azureuser",
      "adminPassword": "P@ssw0rd!"
    }
  }
}
```

Reimage a VM using REST

You can reimage a Virtual Machine instance with ephemeral OS disk using REST API as described below and via Azure Portal by going to Overview pane of the VM. For scale sets, reimaging is already available through Powershell, CLI, and the portal.

```
POST https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{rgName}/providers/Microsoft.Compute/VirtualMachines/{vmName}/reimage?api-version=2018-06-01"
```

Frequently asked questions

Q: What is the size of the local OS Disks?

A: We support platform and custom images, up to the VM cache size, where all read/writes to the OS disk will be local on the same node as the Virtual Machine.

Q: Can the ephemeral OS disk be resized?

A: No, once the ephemeral OS disk is provisioned, the OS disk cannot be resized.

Q: Can I attach a Managed Disks to an Ephemeral VM?

A: Yes, you can attach a managed data disk to a VM that uses an ephemeral OS disk.

Q: Will all VM sizes be supported for ephemeral OS disks?

A: No, all Premium Storage VM sizes are supported (DS, ES, FS, GS and M) except the B-series, N-series, and H-series sizes.

Q: Can the ephemeral OS disk be applied to existing VMs and scale sets?

A: No, ephemeral OS disk can only be used during VM and scale set creation.

Q: Can you mix ephemeral and normal OS disks in a scale set?

A: No, you can't have a mix of ephemeral and persistent OS disk instances within the same scale set.

Q: Can the ephemeral OS disk be created using Powershell or CLI?

A: Yes, you can create VMs with Ephemeral OS Disk using REST, Templates, PowerShell and CLI.

Q: What features are not supported with ephemeral OS disk?

A: Ephemeral disks do not support:

- Capturing VM images
- Disk snapshots
- Azure Disk Encryption
- Azure Backup
- Azure Site Recovery
- OS Disk Swap

Next steps

For more information about the different sizes available for virtual machines, see [Azure Virtual Machine sizes](#).

Scalability and performance targets for VM disks on Windows

1/3/2020 • 5 minutes to read • [Edit Online](#)

You can attach a number of data disks to an Azure virtual machine. Based on the scalability and performance targets for a VM's data disks, you can determine the number and type of disk that you need to meet your performance and capacity requirements.

IMPORTANT

For optimal performance, limit the number of highly utilized disks attached to the virtual machine to avoid possible throttling. If all attached disks aren't highly utilized at the same time, the virtual machine can support a larger number of disks.

For Azure managed disks:

The following table illustrates the default and maximum limits of the number of resources per region per subscription. There is no limit for the number of Managed Disks, snapshots and images per resource group.

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Standard managed disks	50,000	50,000
Standard SSD managed disks	50,000	50,000
Premium managed disks	50,000	50,000
Standard_LRS snapshots	50,000	50,000
Standard_ZRS snapshots	50,000	50,000
Managed image	50,000	50,000

- **For Standard storage accounts:** A Standard storage account has a maximum total request rate of 20,000 IOPS. The total IOPS across all of your virtual machine disks in a Standard storage account should not exceed this limit.

You can roughly calculate the number of highly utilized disks supported by a single Standard storage account based on the request rate limit. For example, for a Basic tier VM, the maximum number of highly utilized disks is about 66, which is $20,000/300$ IOPS per disk. The maximum number of highly utilized disks for a Standard tier VM is about 40, which is $20,000/500$ IOPS per disk.

- **For Premium storage accounts:** A Premium storage account has a maximum total throughput rate of 50 Gbps. The total throughput across all of your VM disks should not exceed this limit.

See [Windows VM sizes](#) for additional details.

Managed virtual machine disks

Sizes denoted with an asterisk are currently in preview. See our [FAQ](#) to learn what regions they are available in.

Standard HDD managed disks

STANDARD DISK TYPE	S4	S6	S10	S15	S20	S30	S40	S50	S60	S70	S80
Disk size in GiB	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384	32,767
IOPS per disk	Up to 500	Up to 1,300	Up to 2,000	Up to 2,000							
Throughput per disk	Up to 60 MiB/sec	Up to 300 MiB/sec	Up to 500 MiB/sec	Up to 500 MiB/sec							

Standard SSD managed disks

STANDARD SSD SIZE S	E1*	E2*	E3*	E4	E6	E10	E15	E20	E30	E40	E50	E60	E70	E80
Disk size in GiB	4	8	16	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384	32,767
IOPS per disk	Up to 120	Up to 120	Up to 120	Up to 120	Up to 240	Up to 500	Up to 2,000	Up to 4,000	Up to 6,000					
Throughput per disk	Up to 25 MiB/sec	Up to 50 MiB/sec	Up to 60 MiB/sec	Up to 400 MiB/sec	Up to 600 MiB/sec	Up to 750 MiB/sec								

*Denotes a disk size that is currently in preview, for regional availability information see [New disk sizes: Managed and unmanaged](#).

Premium SSD managed disks: Per-disk limits

PREMIUM SSD SIZE S	P1*	P2*	P3*	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
Disk size in GiB	4	8	16	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384	32,767

PRE MIU M SSD SIZE S	P1*	P2*	P3*	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
-------------------------------------	-----	-----	-----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----

IOP S per disk	120	120	120	120	240	500	1,10 0	2,30 0	5,00 0	7,50 0	7,50 0	16,0 00	18,0 00	20,0 00
Throughput per disk	25 MiB /sec	25 MiB /sec	25 MiB /sec	25 MiB /sec	50 MiB /sec	100 MiB /sec	125 MiB /sec	150 MiB /sec	200 MiB /sec	250 MiB /sec	250 MiB /sec	500 MiB /sec	750 MiB /sec	900 MiB /sec
Max burst IOP S per disk **	3,5 00	3,5 00	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0	3,50 0
Max burst throughput per disk **	170 MiB /sec	170 MiB /sec	170 MiB /sec	170 MiB /sec	170 MiB /sec	170 MiB /sec								
Max burst duration**	30 min	30 min	30 min	30 min	30 min	30 min								
Eligible for reservation	No	Yes, up to one year												

*Denotes a disk size that is currently in preview, for regional availability information see [New disk sizes: Managed and unmanaged](#).

**Denotes a feature that is currently in preview, see [Disk bursting](#) for more information.

Premium SSD managed disks: Per-VM limits

RESOURCE	DEFAULT LIMIT
Maximum IOPS Per VM	80,000 IOPS with GS5 VM
Maximum throughput per VM	2,000 MB/s with GS5 VM

Unmanaged virtual machine disks

Standard unmanaged virtual machine disks: Per-disk limits

VM TIER	BASIC TIER VM	STANDARD TIER VM
Disk size	4,095 GB	4,095 GB
Maximum 8-KB IOPS per persistent disk	300	500
Maximum number of disks that perform the maximum IOPS	66	40

Premium unmanaged virtual machine disks: Per-account limits

RESOURCE	DEFAULT LIMIT
Total disk capacity per account	35 TB
Total snapshot capacity per account	10 TB
Maximum bandwidth per account (ingress + egress) ¹	<=50 Gbps

¹*Ingress* refers to all data from requests that are sent to a storage account. *Egress* refers to all data from responses that are received from a storage account.

Premium unmanaged virtual machine disks: Per-disk limits

PREMIUM STORAGE DISK TYPE	P10	P20	P30	P40	P50
Disk size	128 GiB	512 GiB	1,024 GiB (1 TB)	2,048 GiB (2 TB)	4,095 GiB (4 TB)
Maximum IOPS per disk	500	2,300	5,000	7,500	7,500
Maximum throughput per disk	100 MB/sec	150 MB/sec	200 MB/sec	250 MB/sec	250 MB/sec
Maximum number of disks per storage account	280	70	35	17	8

Premium unmanaged virtual machine disks: Per-VM limits

RESOURCE	DEFAULT LIMIT
Maximum IOPS per VM	80,000 IOPS with GS5 VM
Maximum throughput per VM	2,000 MB/sec with GS5 VM

See also

[Azure subscription and service limits, quotas, and constraints](#)

Backup and disaster recovery for Azure IaaS disks

12/10/2019 • 21 minutes to read • [Edit Online](#)

This article explains how to plan for backup and disaster recovery (DR) of IaaS virtual machines (VMs) and disks in Azure. This document covers both managed and unmanaged disks.

First, we cover the built-in fault tolerance capabilities in the Azure platform that helps guard against local failures. We then discuss the disaster scenarios not fully covered by the built-in capabilities. We also show several examples of workload scenarios where different backup and DR considerations can apply. We then review possible solutions for the DR of IaaS disks.

Introduction

The Azure platform uses various methods for redundancy and fault tolerance to help protect customers from localized hardware failures. Local failures can include problems with an Azure Storage server machine that stores part of the data for a virtual disk or failures of an SSD or HDD on that server. Such isolated hardware component failures can happen during normal operations.

The Azure platform is designed to be resilient to these failures. Major disasters can result in failures or the inaccessibility of many storage servers or even a whole datacenter. Although your VMs and disks are normally protected from localized failures, additional steps are necessary to protect your workload from region-wide catastrophic failures, such as a major disaster, that can affect your VM and disks.

In addition to the possibility of platform failures, problems with a customer application or data can occur. For example, a new version of your application might inadvertently make a change to the data that causes it to break. In that case, you might want to revert the application and the data to a prior version that contains the last known good state. This requires maintaining regular backups.

For regional disaster recovery, you must back up your IaaS VM disks to a different region.

Before we look at backup and DR options, let's recap a few methods available for handling localized failures.

Azure IaaS resiliency

Resiliency refers to the tolerance for normal failures that occur in hardware components. Resiliency is the ability to recover from failures and continue to function. It's not about avoiding failures, but responding to failures in a way that avoids downtime or data loss. The goal of resiliency is to return the application to a fully functioning state following a failure. Azure virtual machines and disks are designed to be resilient to common hardware faults. Let's look at how the Azure IaaS platform provides this resiliency.

A virtual machine consists mainly of two parts: a compute server and the persistent disks. Both affect the fault tolerance of a virtual machine.

If the Azure compute host server that houses your VM experiences a hardware failure, which is rare, Azure is designed to automatically restore the VM on another server. If this scenario, your computer reboots, and the VM comes back up after some time. Azure automatically detects such hardware failures and executes recoveries to help ensure the customer VM is available as soon as possible.

Regarding IaaS disks, the durability of data is critical for a persistent storage platform. Azure customers have important business applications running on IaaS, and they depend on the persistence of the data. Azure designs protection for these IaaS disks, with three redundant copies of the data that is stored locally. These copies provide for high durability against local failures. If one of the hardware components that holds your disk fails, your VM is not affected, because there are two additional copies to support disk requests. It works fine, even if two different

hardware components that support a disk fail at the same time (which is rare).

To ensure that you always maintain three replicas, Azure Storage automatically spawns a new copy of the data in the background if one of the three copies becomes unavailable. Therefore, it should not be necessary to use RAID with Azure disks for fault tolerance. A simple RAID 0 configuration should be sufficient for striping the disks, if necessary, to create larger volumes.

Because of this architecture, Azure has consistently delivered enterprise-grade durability for IaaS disks, with an industry-leading zero percent [annualized failure rate](#).

Localized hardware faults on the compute host or in the Storage platform can sometimes result in the temporary unavailability of the VM that is covered by the [Azure SLA](#) for VM availability. Azure also provides an industry-leading SLA for single VM instances that use Azure premium SSDs.

To safeguard application workloads from downtime due to the temporary unavailability of a disk or VM, customers can use [availability sets](#). Two or more virtual machines in an availability set provide redundancy for the application. Azure then creates these VMs and disks in separate fault domains with different power, network, and server components.

Because of these separate fault domains, localized hardware failures typically do not affect multiple VMs in the set at the same time. Having separate fault domains provides high availability for your application. It's considered a good practice to use availability sets when high availability is required. The next section covers the disaster recovery aspect.

Backup and disaster recovery

Disaster recovery is the ability to recover from rare, but major, incidents. These incidents include non-transient, wide-scale failures, such as service disruption that affects an entire region. Disaster recovery includes data backup and archiving, and might include manual intervention, such as restoring a database from a backup.

The Azure platform's built-in protection against localized failures might not fully protect the VMs/disks if a major disaster causes large-scale outages. These large-scale outages include catastrophic events, such as if a datacenter is hit by a hurricane, earthquake, fire, or if there is a large-scale hardware unit failure. In addition, you might encounter failures due to application or data issues.

To help protect your IaaS workloads from outages, you should plan for redundancy and have backups to enable recovery. For disaster recovery, you should back up in a different geographic location away from the primary site. This approach helps ensure your backup is not affected by the same event that originally affected the VM or disks. For more information, see [Disaster recovery for Azure applications](#).

Your DR considerations might include the following aspects:

- High availability: The ability of the application to continue running in a healthy state, without significant downtime. By *healthy state*, this state means that the application is responsive, and users can connect to the application and interact with it. Certain mission-critical applications and databases might be required to always be available, even when there are failures in the platform. For these workloads, you might need to plan redundancy for the application, as well as the data.
- Data durability: In some cases, the main consideration is ensuring that the data is preserved if a disaster happens. Therefore, you might need a backup of your data in a different site. For such workloads, you might not need full redundancy for the application, but only a regular backup of the disks.

Backup and DR scenarios

Let's look at a few typical examples of application workload scenarios and the considerations for planning for disaster recovery.

Scenario 1: Major database solutions

Consider a production database server, like SQL Server or Oracle, that can support high availability. Critical production applications and users depend on this database. The disaster recovery plan for this system might need to support the following requirements:

- The data must be protected and recoverable.
- The server must be available for use.

The disaster recovery plan might require maintaining a replica of the database in a different region as a backup. Depending on the requirements for server availability and data recovery, the solution might range from an active-active or active-passive replica site to periodic offline backups of the data. Relational databases, such as SQL Server and Oracle, provide various options for replication. For SQL Server, use [SQL Server AlwaysOn Availability Groups](#) for high availability.

NoSQL databases, like MongoDB, also support [replicas](#) for redundancy. The replicas for high availability are used.

Scenario 2: A cluster of redundant VMs

Consider a workload handled by a cluster of VMs that provide redundancy and load balancing. One example is a Cassandra cluster deployed in a region. This type of architecture already provides a high level of redundancy within that region. However, to protect the workload from a regional-level failure, you should consider spreading the cluster across two regions or making periodic backups to another region.

Scenario 3: IaaS application workload

Let's look at the IaaS application workload. For example, this application might be a typical production workload running on an Azure VM. It might be a web server or file server holding the content and other resources of a site. It might also be a custom-built business application running on a VM that stored its data, resources, and application state on the VM disks. In this case, it's important to make sure you take backups on a regular basis. Backup frequency should be based on the nature of the VM workload. For example, if the application runs every day and modifies data, then the backup should be taken every hour.

Another example is a reporting server that pulls data from other sources and generates aggregated reports. The loss of this VM or disks might lead to the loss of the reports. However, it might be possible to rerun the reporting process and regenerate the output. In that case, you don't really have a loss of data, even if the reporting server is hit with a disaster. As a result, you might have a higher level of tolerance for losing part of the data on the reporting server. In that case, less frequent backups are an option to reduce costs.

Scenario 4: IaaS application data issues

IaaS application data issues are another possibility. Consider an application that computes, maintains, and serves critical commercial data, such as pricing information. A new version of your application had a software bug that incorrectly computed the pricing and corrupted the existing commerce data served by the platform. Here, the best course of action is to revert to the earlier version of the application and the data. To enable this, take periodic backups of your system.

Disaster recovery solution: Azure Backup

[Azure Backup](#) is used for backups and DR, and it works with [managed disks](#) as well as unmanaged disks. You can create a backup job with time-based backups, easy VM restoration, and backup retention policies.

If you use [premium SSDs](#), [managed disks](#), or other disk types with the [locally redundant storage](#) option, it's especially important to make periodic DR backups. Azure Backup stores the data in your recovery services vault for long-term retention. Choose the [geo-redundant storage](#) option for the backup recovery services vault. That option ensures that backups are replicated to a different Azure region for safeguarding from regional disasters.

For unmanaged disks, you can use the locally redundant storage type for IaaS disks, but ensure that Azure Backup is enabled with the geo-redundant storage option for the recovery services vault.

NOTE

If you use the [geo-redundant storage](#) or [read-access geo-redundant storage](#) option for your unmanaged disks, you still need consistent snapshots for backup and DR. Use either [Azure Backup](#) or [consistent snapshots](#).

The following table is a summary of the solutions available for DR.

SCENARIO	AUTOMATIC REPLICATION	DR SOLUTION
Premium SSD disks	Local (locally redundant storage)	Azure Backup
Managed disks	Local (locally redundant storage)	Azure Backup
Unmanaged locally redundant storage disks	Local (locally redundant storage)	Azure Backup
Unmanaged geo-redundant storage disks	Cross region (geo-redundant storage)	Azure Backup Consistent snapshots
Unmanaged read-access geo-redundant storage disks	Cross region (read-access geo-redundant storage)	Azure Backup Consistent snapshots

High availability is best met by using managed disks in an availability set along with Azure Backup. If you use unmanaged disks, you can still use Azure Backup for DR. If you are unable to use Azure Backup, then taking [consistent snapshots](#), as described in a later section, is an alternative solution for backup and DR.

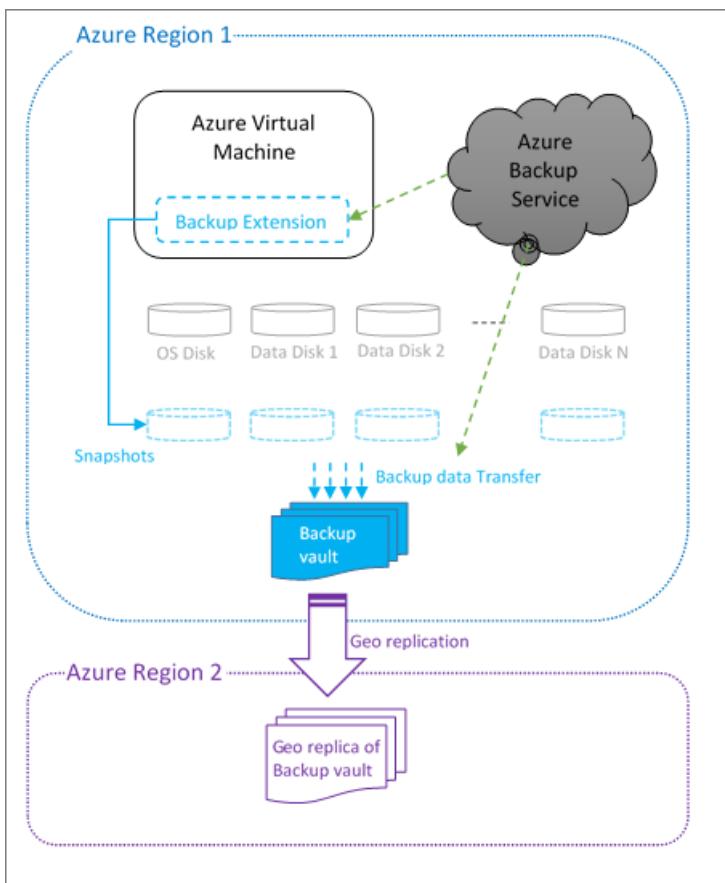
Your choices for high availability, backup, and DR at application or infrastructure levels can be represented as follows:

LEVEL	HIGH AVAILABILITY	BACKUP OR DR
Application	SQL Server AlwaysOn	Azure Backup
Infrastructure	Availability set	Geo-redundant storage with consistent snapshots

Using Azure Backup

[Azure Backup](#) can back up your VMs running Windows or Linux to the Azure recovery services vault. Backing up and restoring business-critical data is complicated by the fact that business-critical data must be backed up while the applications that produce the data are running.

To address this issue, Azure Backup provides application-consistent backups for Microsoft workloads. It uses the volume shadow service to ensure that data is written correctly to storage. For Linux VMs, the default backup consistency mode is file-consistent backups, because Linux does not have functionality equivalent to the volume shadow service as in the case of Windows. For Linux machines, see [Application-consistent backup of Azure Linux VMs](#).



When Azure Backup initiates a backup job at the scheduled time, it triggers the backup extension installed in the VM to take a point-in-time snapshot. A snapshot is taken in coordination with the volume shadow service to get a consistent snapshot of the disks in the virtual machine without having to shut it down. The backup extension in the VM flushes all writes before taking a consistent snapshot of all of the disks. After taking the snapshot, the data is transferred by Azure Backup to the backup vault. To make the backup process more efficient, the service identifies and transfers only the blocks of data that have changed after the last backup.

To restore, you can view the available backups through Azure Backup and then initiate a restore. You can create and restore Azure backups through the [Azure portal](#), by [using PowerShell](#), or by using the [Azure CLI](#).

Steps to enable a backup

Use the following steps to enable backups of your VMs by using the [Azure portal](#). There is some variation depending on your exact scenario. Refer to the [Azure Backup](#) documentation for full details. Azure Backup also [supports VMs with managed disks](#).

1. Create a recovery services vault for a VM:
 - a. In the [Azure portal](#), browse **All resources** and find **Recovery Services vaults**.
 - b. On the **Recovery Services vaults** menu, click **Add** and follow the steps to create a new vault in the same region as the VM. For example, if your VM is in the West US region, pick West US for the vault.
2. Verify the storage replication for the newly created vault. Access the vault under **Recovery Services vaults** and go to **Properties > Backup Configuration > Update**. Ensure the **geo-redundant storage** option is selected by default. This option ensures that your vault is automatically replicated to a secondary datacenter. For example, your vault in West US is automatically replicated to East US.
3. Configure the backup policy and select the VM from the same UI.
4. Make sure the Backup Agent is installed on the VM. If your VM is created by using an Azure gallery image, then the Backup Agent is already installed. Otherwise (that is, if you use a custom image), use the instructions to [install the VM agent on a virtual machine](#).

- After the previous steps are completed, the backup runs at regular intervals as specified in the backup policy.
If necessary, you can trigger the first backup manually from the vault dashboard on the Azure portal.

For automating Azure Backup by using scripts, refer to [PowerShell cmdlets for VM backup](#).

Steps for recovery

If you need to repair or rebuild a VM, you can restore the VM from any of the backup recovery points in the vault. There are a couple of different options for performing the recovery:

- You can create a new VM as a point-in-time representation of your backed-up VM.
- You can restore the disks, and then use the template for the VM to customize and rebuild the restored VM.

For more information, see the instructions to [use the Azure portal to restore virtual machines](#). This document also explains the specific steps for restoring backed-up VMs to a paired datacenter by using your geo-redundant backup vault if there is a disaster at the primary datacenter. In that case, Azure Backup uses the Compute service from the secondary region to create the restored virtual machine.

You can also use PowerShell for [creating a new VM from restored disks](#).

Alternative solution: Consistent snapshots

If you are unable to use Azure Backup, you can implement your own backup mechanism by using snapshots. Creating consistent snapshots for all the disks used by a VM and then replicating those snapshots to another region is complicated. For this reason, Azure considers using the Backup service as a better option than building a custom solution.

If you use read-access geo-redundant storage/geo-redundant storage for disks, snapshots are automatically replicated to a secondary datacenter. If you use locally redundant storage for disks, you need to replicate the data yourself. For more information, see [Back up Azure-unmanaged VM disks with incremental snapshots](#).

A snapshot is a representation of an object at a specific point in time. A snapshot incurs billing for the incremental size of the data it holds. For more information, see [Create a blob snapshot](#).

Create snapshots while the VM is running

Although you can take a snapshot at any time, if the VM is running, there is still data being streamed to the disks. The snapshots might contain partial operations that were in flight. Also, if there are several disks involved, the snapshots of different disks might have occurred at different times. These scenarios may cause to the snapshots to be uncoordinated. This lack of co-ordination is especially problematic for striped volumes whose files might be corrupted if changes were being made during backup.

To avoid this situation, the backup process must implement the following steps:

- Freeze all the disks.
- Flush all the pending writes.
- [Create a blob snapshot](#) for all the disks.

Some Windows applications, like SQL Server, provide a coordinated backup mechanism via a volume shadow service to create application-consistent backups. On Linux, you can use a tool like `fsfreeze` for coordinating the disks. This tool provides file-consistent backups, but not application-consistent snapshots. This process is complex, so you should consider using [Azure Backup](#) or a third-party backup solution that already implements this procedure.

The previous process results in a collection of coordinated snapshots for all of the VM disks, representing a specific point-in-time view of the VM. This is a backup restore point for the VM. You can repeat the process at scheduled intervals to create periodic backups. See [Copy the backups to another region](#) for steps to copy the snapshots to

another region for DR.

Create snapshots while the VM is offline

Another option to create consistent backups is to shut down the VM and take blob snapshots of each disk. Taking blob snapshots is easier than coordinating snapshots of a running VM, but it requires a few minutes of downtime.

1. Shut down the VM.
2. Create a snapshot of each virtual hard drive blob, which only takes a few seconds.

To create a snapshot, you can use [PowerShell](#), the [Azure Storage REST API](#), [Azure CLI](#), or one of the Azure Storage client libraries, such as [the Storage client library for .NET](#).

3. Start the VM, which ends the downtime. Typically, the entire process finishes within a few minutes.

This process yields a collection of consistent snapshots for all the disks, providing a backup restore point for the VM.

Copy the snapshots to another region

Creation of the snapshots alone might not be sufficient for DR. You must also replicate the snapshot backups to another region.

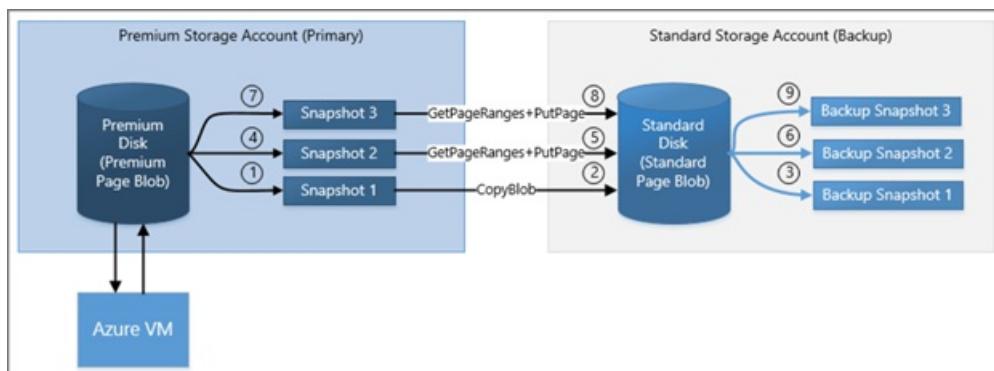
If you use geo-redundant storage or read-access geo-redundant storage for your disks, then the snapshots are replicated to the secondary region automatically. There can be a few minutes of lag before the replication. If the primary datacenter goes down before the snapshots finish replicating, you cannot access the snapshots from the secondary datacenter. The likelihood of this is small.

NOTE

Only having the disks in a geo-redundant storage or read-access geo-redundant storage account does not protect the VM from disasters. You must also create coordinated snapshots or use Azure Backup. This is required to recover a VM to a consistent state.

If you use locally redundant storage, you must copy the snapshots to a different storage account immediately after creating the snapshot. The copy target might be a locally redundant storage account in a different region, resulting in the copy being in a remote region. You can also copy the snapshot to a read-access geo-redundant storage account in the same region. In this case, the snapshot is lazily replicated to the remote secondary region. Your backup is protected from disasters at the primary site after the copying and replication is complete.

To copy your incremental snapshots for DR efficiently, review the instructions in [Back up Azure unmanaged VM disks with incremental snapshots](#).



Recovery from snapshots

To retrieve a snapshot, copy it to make a new blob. If you are copying the snapshot from the primary account, you can copy the snapshot over to the base blob of the snapshot. This process reverts the disk to the snapshot. This process is known as promoting the snapshot. If you are copying the snapshot backup from a secondary account, in

the case of a read-access geo-redundant storage account, you must copy it to a primary account. You can copy a snapshot by [using PowerShell](#) or by using the AzCopy utility. For more information, see [Transfer data with the AzCopy command-line utility](#).

For VMs with multiple disks, you must copy all the snapshots that are part of the same coordinated restore point. After you copy the snapshots to writable VHD blobs, you can use the blobs to recreate your VM by using the template for the VM.

Other options

SQL Server

SQL Server running in a VM has its own built-in capabilities to back up your SQL Server database to Azure Blob storage or a file share. If the storage account is geo-redundant storage or read-access geo-redundant storage, you can access those backups in the storage account's secondary datacenter in the event of a disaster, with the same restrictions as previously discussed. For more information, see [Back up and restore for SQL Server in Azure virtual machines](#). In addition to back up and restore, [SQL Server AlwaysOn availability groups](#) can maintain secondary replicas of databases. This ability greatly reduces the disaster recovery time.

Other considerations

This article has discussed how to back up or take snapshots of your VMs and their disks to support disaster recovery and how to use those backups or snapshots to recover your data. With the Azure Resource Manager model, many people use templates to create their VMs and other infrastructures in Azure. You can use a template to create a VM that has the same configuration every time. If you use custom images for creating your VMs, you must also make sure that your images are protected by using a read-access geo-redundant storage account to store them.

Consequently, your backup process can be a combination of two things:

- Back up the data (disks).
- Back up the configuration (templates and custom images).

Depending on the backup option you choose, you might have to handle the backup of both the data and the configuration, or the backup service might handle all of that for you.

Appendix: Understanding the impact of data redundancy

For storage accounts in Azure, there are three types of data redundancy that you should consider regarding disaster recovery: locally redundant, geo-redundant, or geo-redundant with read access.

Locally redundant storage retains three copies of the data in the same datacenter. When the VM writes the data, all three copies are updated before success is returned to the caller, so you know they are identical. Your disk is protected from local failures, because it's unlikely that all three copies are affected at the same time. In the case of locally redundant storage, there is no geo-redundancy, so the disk is not protected from catastrophic failures that can affect an entire datacenter or storage unit.

With geo-redundant storage and read-access geo-redundant storage, three copies of your data are retained in the primary region that is selected by you. Three more copies of your data are retained in a corresponding secondary region that is set by Azure. For example, if you store data in West US, the data is replicated to East US. Copy retention is done asynchronously, and there is a small delay between updates to the primary and secondary sites. Replicas of the disks on the secondary site are consistent on a per-disk basis (with the delay), but replicas of multiple active disks might not be in sync with each other. To have consistent replicas across multiple disks, consistent snapshots are needed.

The main difference between geo-redundant storage and read-access geo-redundant storage is that with read-

access geo-redundant storage, you can read the secondary copy at any time. If there is a problem that renders the data in the primary region inaccessible, the Azure team makes every effort to restore access. While the primary is down, if you have read-access geo-redundant storage enabled, you can access the data in the secondary datacenter. Therefore, if you plan to read from the replica while the primary is inaccessible, then read-access geo-redundant storage should be considered.

If it turns out to be a significant outage, the Azure team might trigger a geo-failover and change the primary DNS entries to point to secondary storage. At this point, if you have either geo-redundant storage or read-access geo-redundant storage enabled, you can access the data in the region that used to be the secondary. In other words, if your storage account is geo-redundant storage and there is a problem, you can access the secondary storage only if there is a geo-failover.

For more information, see [What to do if an Azure Storage outage occurs](#).

NOTE

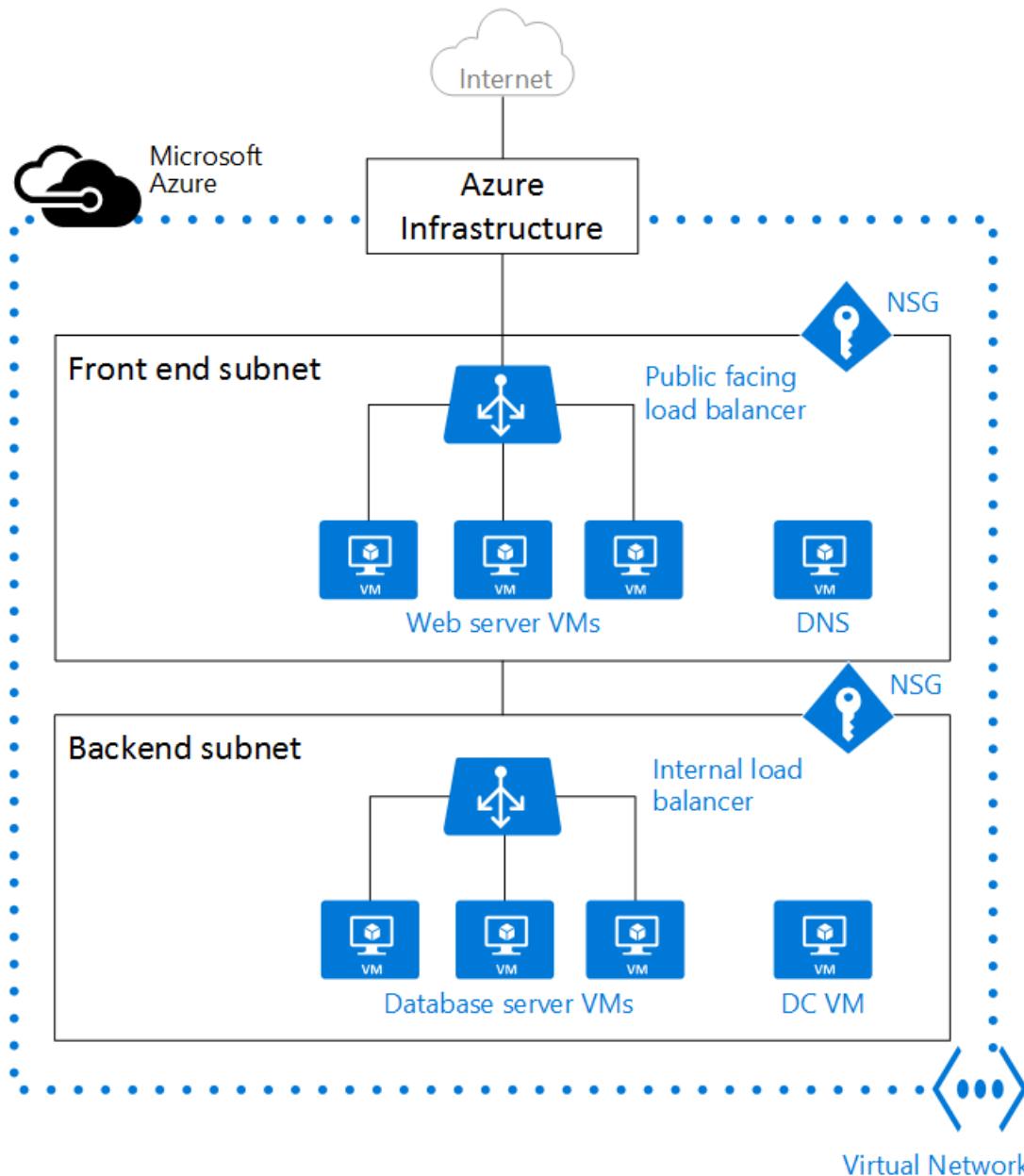
Microsoft controls whether a failover occurs. Failover is not controlled per storage account, so it's not decided by individual customers. To implement disaster recovery for specific storage accounts or virtual machine disks, you must use the techniques described previously in this article.

Virtual networks and virtual machines in Azure

11/13/2019 • 13 minutes to read • [Edit Online](#)

When you create an Azure virtual machine (VM), you must create a [virtual network](#) (VNet) or use an existing VNet. You also need to decide how your VMs are intended to be accessed on the VNet. It is important to [plan before creating resources](#) and make sure that you understand the [limits of networking resources](#).

In the following figure, VMs are represented as web servers and database servers. Each set of VMs are assigned to separate subnets in the VNet.



You can create a VNet before you create a VM or you can do so as you create a VM. You create these resources to support communication with a VM:

- Network interfaces
- IP addresses
- Virtual network and subnets

In addition to those basic resources, you should also consider these optional resources:

- Network security groups
- Load balancers

Network interfaces

A [network interface \(NIC\)](#) is the interconnection between a VM and a virtual network (VNet). A VM must have at least one NIC, but can have more than one, depending on the size of the VM you create. Learn about how many NICs each VM size supports for [Windows](#) or [Linux](#).

You can create a VM with multiple NICs, and add or remove NICs through the lifecycle of a VM. Multiple NICs allow a VM to connect to different subnets and send or receive traffic over the most appropriate interface. VMs with any number of network interfaces can exist in the same availability set, up to the number supported by the VM size.

Each NIC attached to a VM must exist in the same location and subscription as the VM. Each NIC must be connected to a VNet that exists in the same Azure location and subscription as the NIC. You can change the subnet a VM is connected to after it's created, but you cannot change the VNet. Each NIC attached to a VM is assigned a MAC address that doesn't change until the VM is deleted.

This table lists the methods that you can use to create a network interface.

METHOD	DESCRIPTION
Azure portal	When you create a VM in the Azure portal, a network interface is automatically created for you (you cannot use a NIC you create separately). The portal creates a VM with only one NIC. If you want to create a VM with more than one NIC, you must create it with a different method.
Azure PowerShell	Use New-AzNetworkInterface with the -PublicIpAddressId parameter to provide the identifier of the public IP address that you previously created.
Azure CLI	To provide the identifier of the public IP address that you previously created, use az network nic create with the --public-ip-address parameter.
Template	Use Network Interface in a Virtual Network with Public IP Address as a guide for deploying a network interface using a template.

IP addresses

You can assign these types of [IP addresses](#) to a NIC in Azure:

- **Public IP addresses** - Used to communicate inbound and outbound (without network address translation (NAT)) with the Internet and other Azure resources not connected to a VNet. Assigning a public IP address to a NIC is optional. Public IP addresses have a nominal charge, and there's a maximum number that can be used per subscription.
- **Private IP addresses** - Used for communication within a VNet, your on-premises network, and the Internet (with NAT). You must assign at least one private IP address to a VM. To learn more about NAT in Azure, read [Understanding outbound connections in Azure](#).

You can assign public IP addresses to VMs or internet-facing load balancers. You can assign private IP addresses to VMs and internal load balancers. You assign IP addresses to a VM using a network interface.

There are two methods in which an IP address is allocated to a resource - dynamic or static. The default allocation method is dynamic, where an IP address is not allocated when it's created. Instead, the IP address is allocated when you create a VM or start a stopped VM. The IP address is released when you stop or delete the VM.

To ensure the IP address for the VM remains the same, you can set the allocation method explicitly to static. In this case, an IP address is assigned immediately. It is released only when you delete the VM or change its allocation method to dynamic.

This table lists the methods that you can use to create an IP address.

METHOD	DESCRIPTION
Azure portal	By default, public IP addresses are dynamic and the address associated to them may change when the VM is stopped or deleted. To guarantee that the VM always uses the same public IP address, create a static public IP address. By default, the portal assigns a dynamic private IP address to a NIC when creating a VM. You can change this IP address to static after the VM is created.
Azure PowerShell	You use New-AzPublicIpAddress with the -AllocationMethod parameter as Dynamic or Static.
Azure CLI	You use az network public-ip create with the --allocation-method parameter as Dynamic or Static.
Template	Use Network Interface in a Virtual Network with Public IP Address as a guide for deploying a public IP address using a template.

After you create a public IP address, you can associate it with a VM by assigning it to a NIC.

Virtual network and subnets

A subnet is a range of IP addresses in the VNet. You can divide a VNet into multiple subnets for organization and security. Each NIC in a VM is connected to one subnet in one VNet. NICs connected to subnets (same or different) within a VNet can communicate with each other without any extra configuration.

When you set up a VNet, you specify the topology, including the available address spaces and subnets. If the VNet is to be connected to other VNets or on-premises networks, you must select address ranges that don't overlap. The IP addresses are private and can't be accessed from the Internet, which was true only for the non-routable IP addresses such as 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16. Now, Azure treats any address range as part of the private VNet IP address space that is only reachable within the VNet, within interconnected VNets, and from your on-premises location.

If you work within an organization in which someone else is responsible for the internal networks, you should talk to that person before selecting your address space. Make sure there is no overlap and let them know the space you want to use so they don't try to use the same range of IP addresses.

By default, there is no security boundary between subnets, so VMs in each of these subnets can talk to one another. However, you can set up Network Security Groups (NSGs), which allow you to control the traffic flow to and from subnets and to and from VMs.

This table lists the methods that you can use to create a VNet and subnets.

METHOD	DESCRIPTION
Azure portal	If you let Azure create a VNet when you create a VM, the name is a combination of the resource group name that contains the VNet and -vnet . The address space is 10.0.0.0/24, the required subnet name is default , and the subnet address range is 10.0.0.0/24.
Azure PowerShell	You use New-AzVirtualNetworkSubnetConfig and New-AzVirtualNetwork to create a subnet and a VNet. You can also use Add-AzVirtualNetworkSubnetConfig to add a subnet to an existing VNet.
Azure CLI	The subnet and the VNet are created at the same time. Provide a --subnet-name parameter to az network vnet create with the subnet name.
Template	The easiest way to create a VNet and subnets is to download an existing template, such as Virtual Network with two subnets , and modify it for your needs.

Network security groups

A [network security group \(NSG\)](#) contains a list of Access Control List (ACL) rules that allow or deny network traffic to subnets, NICs, or both. NSGs can be associated with either subnets or individual NICs connected to a subnet. When an NSG is associated with a subnet, the ACL rules apply to all the VMs in that subnet. In addition, traffic to an individual NIC can be restricted by associating an NSG directly to a NIC.

NSGs contain two sets of rules: inbound and outbound. The priority for a rule must be unique within each set. Each rule has properties of protocol, source and destination port ranges, address prefixes, direction of traffic, priority, and access type.

All NSGs contain a set of default rules. The default rules cannot be deleted, but because they are assigned the lowest priority, they can be overridden by the rules that you create.

When you associate an NSG to a NIC, the network access rules in the NSG are applied only to that NIC. If an NSG is applied to a single NIC on a multi-NIC VM, it does not affect traffic to the other NICs. You can associate different NSGs to a NIC (or VM, depending on the deployment model) and the subnet that a NIC or VM is bound to. Priority is given based on the direction of traffic.

Be sure to [plan](#) your NSGs when you plan your VMs and VNet.

This table lists the methods that you can use to create a network security group.

METHOD	DESCRIPTION
Azure portal	When you create a VM in the Azure portal, an NSG is automatically created and associated to the NIC the portal creates. The name of the NSG is a combination of the name of the VM and -nsg . This NSG contains one inbound rule with a priority of 1000, service set to RDP, the protocol set to TCP, port set to 3389, and action set to Allow. If you want to allow any other inbound traffic to the VM, you must add additional rules to the NSG.

METHOD	DESCRIPTION
Azure PowerShell	Use New-AzNetworkSecurityRuleConfig and provide the required rule information. Use New-AzNetworkSecurityGroup to create the NSG. Use Set-AzVirtualNetworkSubnetConfig to configure the NSG for the subnet. Use Set-AzVirtualNetwork to add the NSG to the VNet.
Azure CLI	Use az network nsg create to initially create the NSG. Use az network nsg rule create to add rules to the NSG. Use az network vnet subnet update to add the NSG to the subnet.
Template	Use Create a Network Security Group as a guide for deploying a network security group using a template.

Load balancers

[Azure Load Balancer](#) delivers high availability and network performance to your applications. A load balancer can be configured to [balance incoming Internet traffic](#) to VMs or [balance traffic between VMs in a VNet](#). A load balancer can also balance traffic between on-premises computers and VMs in a cross-premises network, or forward external traffic to a specific VM.

The load balancer maps incoming and outgoing traffic between the public IP address and port on the load balancer and the private IP address and port of the VM.

When you create a load balancer, you must also consider these configuration elements:

- **Front-end IP configuration** – A load balancer can include one or more front-end IP addresses, otherwise known as virtual IPs (VIPs). These IP addresses serve as ingress for the traffic.
- **Back-end address pool** – IP addresses that are associated with the NIC to which load is distributed.
- **NAT rules** - Defines how inbound traffic flows through the front-end IP and distributed to the back-end IP.
- **Load balancer rules** - Maps a given front-end IP and port combination to a set of back-end IP addresses and port combination. A single load balancer can have multiple load balancing rules. Each rule is a combination of a front-end IP and port and back-end IP and port associated with VMs.
- **Probes** - Monitors the health of VMs. When a probe fails to respond, the load balancer stops sending new connections to the unhealthy VM. The existing connections are not affected, and new connections are sent to healthy VMs.

This table lists the methods that you can use to create an internet-facing load balancer.

METHOD	DESCRIPTION
Azure portal	You can load balance internet traffic to VMs using the Azure portal .
Azure PowerShell	To provide the identifier of the public IP address that you previously created, use New-AzLoadBalancerFrontendIpConfig with the -PublicIpAddress parameter. Use New-AzLoadBalancerBackendAddressPoolConfig to create the configuration of the back-end address pool. Use New-AzLoadBalancerInboundNatRuleConfig to create inbound NAT rules associated with the front-end IP configuration that you created. Use New-AzLoadBalancerProbeConfig to create the probes that you need. Use New-AzLoadBalancerRuleConfig to create the load balancer configuration. Use New-AzLoadBalancer to create the load balancer.

METHOD	DESCRIPTION
Azure CLI	Use az network lb create to create the initial load balancer configuration. Use az network lb frontend-ip create to add the public IP address that you previously created. Use az network lb address-pool create to add the configuration of the back-end address pool. Use az network lb inbound-nat-rule create to add NAT rules. Use az network lb rule create to add the load balancer rules. Use az network lb probe create to add the probes.
Template	Use 2 VMs in a Load Balancer and configure NAT rules on the LB as a guide for deploying a load balancer using a template.

This table lists the methods that you can use to create an internal load balancer.

METHOD	DESCRIPTION
Azure portal	You can balance internal traffic load with a Basic load balancer in the Azure portal .
Azure PowerShell	To provide a private IP address in the network subnet, use New-AzLoadBalancerFrontendIpConfig with the -PrivateIpAddress parameter. Use New-AzLoadBalancerBackendAddressPoolConfig to create the configuration of the back-end address pool. Use New-AzLoadBalancerInboundNatRuleConfig to create inbound NAT rules associated with the front-end IP configuration that you created. Use New-AzLoadBalancerProbeConfig to create the probes that you need. Use New-AzLoadBalancerRuleConfig to create the load balancer configuration. Use New-AzLoadBalancer to create the load balancer.
Azure CLI	Use the az network lb create command to create the initial load balancer configuration. To define the private IP address, use az network lb frontend-ip create with the --private-ip-address parameter. Use az network lb address-pool create to add the configuration of the back-end address pool. Use az network lb inbound-nat-rule create to add NAT rules. Use az network lb rule create to add the load balancer rules. Use az network lb probe create to add the probes.
Template	Use 2 VMs in a Load Balancer and configure NAT rules on the LB as a guide for deploying a load balancer using a template.

VMs

VMs can be created in the same VNet and they can connect to each other using private IP addresses. They can connect even if they are in different subnets without the need to configure a gateway or use public IP addresses. To put VMs into a VNet, you create the VNet and then as you create each VM, you assign it to the VNet and subnet. VMs acquire their network settings during deployment or startup.

VMs are assigned an IP address when they are deployed. If you deploy multiple VMs into a VNet or subnet, they are assigned IP addresses as they boot up. You can also allocate a static IP to a VM. If you allocate a static IP, you should consider using a specific subnet to avoid accidentally reusing a static IP for another VM.

If you create a VM and later want to migrate it into a VNet, it is not a simple configuration change. You must redeploy the VM into the VNet. The easiest way to redeploy is to delete the VM, but not any disks attached to it,

and then re-create the VM using the original disks in the VNet.

This table lists the methods that you can use to create a VM in a VNet.

METHOD	DESCRIPTION
Azure portal	Uses the default network settings that were previously mentioned to create a VM with a single NIC. To create a VM with multiple NICs, you must use a different method.
Azure PowerShell	Includes the use of <code>Add-AzVMNetworkInterface</code> to add the NIC that you previously created to the VM configuration.
Azure CLI	Create and connect a VM to a Vnet, subnet, and NIC that build as individual steps.
Template	Use Very simple deployment of a Windows VM as a guide for deploying a VM using a template.

Next steps

For VM-specific steps on how to manage Azure virtual networks for VMs, see the [Windows](#) or [Linux](#) tutorials.

There are also tutorials on how to load balance VMs and create highly available applications for [Windows](#) or [Linux](#).

- Learn how to configure [user-defined routes and IP forwarding](#).
- Learn how to configure [VNet to VNet connections](#).
- Learn how to [Troubleshoot routes](#).
- Learn more about [Virtual machine network bandwidth](#).

Use infrastructure automation tools with virtual machines in Azure

11/13/2019 • 6 minutes to read • [Edit Online](#)

To create and manage Azure virtual machines (VMs) in a consistent manner at scale, some form of automation is typically desired. There are many tools and solutions that allow you to automate the complete Azure infrastructure deployment and management lifecycle. This article introduces some of the infrastructure automation tools that you can use in Azure. These tools commonly fit in to one of the following approaches:

- Automate the configuration of VMs
 - Tools include [Ansible](#), [Chef](#), and [Puppet](#).
 - Tools specific to VM customization include [cloud-init](#) for Linux VMs, [PowerShell Desired State Configuration \(DSC\)](#), and the [Azure Custom Script Extension](#) for all Azure VMs.
- Automate infrastructure management
 - Tools include [Packer](#) to automate custom VM image builds, and [Terraform](#) to automate the infrastructure build process.
 - [Azure Automation](#) can perform actions across your Azure and on-premises infrastructure.
- Automate application deployment and delivery
 - Examples include [Azure DevOps Services](#) and [Jenkins](#).

Ansible

[Ansible](#) is an automation engine for configuration management, VM creation, or application deployment. Ansible uses an agent-less model, typically with SSH keys, to authenticate and manage target machines. Configuration tasks are defined in playbooks, with a number of Ansible modules available to carry out specific tasks. For more information, see [How Ansible works](#).

Learn how to:

- [Install and configure Ansible on Linux for use with Azure](#).
- [Create a Linux virtual machine](#).
- [Manage a Linux virtual machine](#).

Chef

[Chef](#) is an automation platform that helps define how your infrastructure is configured, deployed, and managed. Additional components included Chef Habitat for application lifecycle automation rather than the infrastructure, and Chef InSpec that helps automate compliance with security and policy requirements. Chef Clients are installed on target machines, with one or more central Chef Servers that store and manage the configurations. For more information, see [An Overview of Chef](#).

Learn how to:

- [Deploy Chef Automate from the Azure Marketplace](#).
- [Install Chef on Windows and create Azure VMs](#).

Puppet

Puppet is an enterprise-ready automation platform that handles the application delivery and deployment process. Agents are installed on target machines to allow Puppet Master to run manifests that define the desired configuration of the Azure infrastructure and VMs. Puppet can integrate with other solutions such as Jenkins and GitHub for an improved devops workflow. For more information, see [How Puppet works](#).

Learn how to:

- [Deploy Puppet from the Azure Marketplace](#).

Cloud-init

[Cloud-init](#) is a widely used approach to customize a Linux VM as it boots for the first time. You can use cloud-init to install packages and write files, or to configure users and security. Because cloud-init is called during the initial boot process, there are no additional steps or required agents to apply your configuration. For more information on how to properly format your `#cloud-config` files, see the [cloud-init documentation site](#). `#cloud-config` files are text files encoded in base64.

Cloud-init also works across distributions. For example, you don't use **apt-get install** or **yum install** to install a package. Instead you can define a list of packages to install. Cloud-init automatically uses the native package management tool for the distro you select.

We are actively working with our endorsed Linux distro partners in order to have cloud-init enabled images available in the Azure marketplace. These images make your cloud-init deployments and configurations work seamlessly with VMs and virtual machine scale sets. Learn more details about cloud-init on Azure:

- [Cloud-init support for Linux virtual machines in Azure](#)
- [Try a tutorial on automated VM configuration using cloud-init](#).

PowerShell DSC

[PowerShell Desired State Configuration \(DSC\)](#) is a management platform to define the configuration of target machines. DSC can also be used on Linux through the [Open Management Infrastructure \(OMI\) server](#).

DSC configurations define what to install on a machine and how to configure the host. A Local Configuration Manager (LCM) engine runs on each target node that processes requested actions based on pushed configurations. A pull server is a web service that runs on a central host to store the DSC configurations and associated resources. The pull server communicates with the LCM engine on each target host to provide the required configurations and report on compliance.

Learn how to:

- [Create a basic DSC configuration](#).
- [Configure a DSC pull server](#).
- [Use DSC for Linux](#).

Azure Custom Script Extension

The Azure Custom Script Extension for [Linux](#) or [Windows](#) downloads and executes scripts on Azure VMs. You can use the extension when you create a VM, or any time after the VM is in use.

Scripts can be downloaded from Azure storage or any public location such as a GitHub repository. With the Custom Script Extension, you can write scripts in any language that runs on the source VM. These scripts can be used to install applications or configure the VM as desired. To secure credentials, sensitive information such as passwords can be stored in a protected configuration. These credentials are only decrypted inside the VM.

Learn how to:

- [Create a Linux VM with the Azure CLI and use the Custom Script Extension.](#)
- [Create a Windows VM with Azure PowerShell and use the Custom Script Extension.](#)

Packer

[Packer](#) automates the build process when you create a custom VM image in Azure. You use Packer to define the OS and run post-configuration scripts that customize the VM for your specific needs. Once configured, the VM is then captured as a Managed Disk image. Packer automates the process to create the source VM, network and storage resources, run configuration scripts, and then create the VM image.

Learn how to:

- [Use Packer to create a Linux VM image in Azure.](#)
- [Use Packer to create a Windows VM image in Azure.](#)

Terraform

[Terraform](#) is an automation tool that allows you to define and create an entire Azure infrastructure with a single template format language - the HashiCorp Configuration Language (HCL). With Terraform, you define templates that automate the process to create network, storage, and VM resources for a given application solution. You can use your existing Terraform templates for other platforms with Azure to ensure consistency and simplify the infrastructure deployment without needing to convert to an Azure Resource Manager template.

Learn how to:

- [Install and configure Terraform with Azure.](#)
- [Create an Azure infrastructure with Terraform.](#)

Azure Automation

[Azure Automation](#) uses runbooks to process a set of tasks on the VMs you target. Azure Automation is used to manage existing VMs rather than to create an infrastructure. Azure Automation can run across both Linux and Windows VMs, as well as on-premises virtual or physical machines with a hybrid runbook worker. Runbooks can be stored in a source control repository, such as GitHub. These runbooks can then run manually or on a defined schedule.

Azure Automation also provides a Desired State Configuration (DSC) service that allows you to create definitions for how a given set of VMs should be configured. DSC then ensures that the required configuration is applied and the VM stays consistent. Azure Automation DSC runs on both Windows and Linux machines.

Learn how to:

- [Create a PowerShell runbook.](#)
- [Use Hybrid Runbook Worker to manage on-premises resources.](#)
- [Use Azure Automation DSC.](#)

Azure DevOps Services

[Azure DevOps Services](#) is a suite of tools that help you share and track code, use automated builds, and create a complete continuous integration and development (CI/CD) pipeline. Azure DevOps Services integrates with Visual Studio and other editors to simplify usage. Azure DevOps Services can also create and configure Azure VMs and then deploy code to them.

Learn more about:

- [Azure DevOps Services.](#)

Jenkins

Jenkins is a continuous integration server that helps deploy and test applications, and create automated pipelines for code delivery. There are hundreds of plugins to extend the core Jenkins platform, and you can also integrate with many other products and solutions through webhooks. You can manually install Jenkins on an Azure VM, run Jenkins from within a Docker container, or use a pre-built Azure Marketplace image.

Learn how to:

- [Create a development infrastructure on a Linux VM in Azure with Jenkins, GitHub, and Docker.](#)

Next steps

There are many different options to use infrastructure automation tools in Azure. You have the freedom to use the solution that best fits your needs and environment. To get started and try some of the tools built-in to Azure, see how to automate the customization of a [Linux](#) or [Windows](#) VM.

Secure and use policies on virtual machines in Azure

11/13/2019 • 4 minutes to read • [Edit Online](#)

It's important to keep your virtual machine (VM) secure for the applications that you run. Securing your VMs can include one or more Azure services and features that cover secure access to your VMs and secure storage of your data. This article provides information that enables you to keep your VM and applications secure.

Antimalware

The modern threat landscape for cloud environments is dynamic, increasing the pressure to maintain effective protection in order to meet compliance and security requirements. [Microsoft Antimalware for Azure](#) is a free real-time protection capability that helps identify and remove viruses, spyware, and other malicious software. Alerts can be configured to notify you when known malicious or unwanted software attempts to install itself or run on your VM. It is not supported on VMs running Linux or Windows Server 2008.

Azure Security Center

[Azure Security Center](#) helps you prevent, detect, and respond to threats to your VMs. Security Center provides integrated security monitoring and policy management across your Azure subscriptions, helps detect threats that might otherwise go unnoticed, and works with a broad ecosystem of security solutions.

Security Center's just-in-time access can be applied across your VM deployment to lock down inbound traffic to your Azure VMs, reducing exposure to attacks while providing easy access to connect to VMs when needed. When just-in-time is enabled and a user requests access to a VM, Security Center checks what permissions the user has for the VM. If they have the correct permissions, the request is approved and Security Center automatically configures the Network Security Groups (NSGs) to allow inbound traffic to the selected ports for a limited amount of time. After the time has expired, Security Center restores the NSGs to their previous states.

Encryption

For enhanced [Windows VM](#) and [Linux VM](#) security and compliance, virtual disks in Azure can be encrypted. Virtual disks on Windows VMs are encrypted at rest using BitLocker. Virtual disks on Linux VMs are encrypted at rest using dm-crypt.

There is no charge for encrypting virtual disks in Azure. Cryptographic keys are stored in Azure Key Vault using software-protection, or you can import or generate your keys in Hardware Security Modules (HSMs) certified to FIPS 140-2 level 2 standards. These cryptographic keys are used to encrypt and decrypt virtual disks attached to your VM. You retain control of these cryptographic keys and can audit their use. An Azure Active Directory service principal provides a secure mechanism for issuing these cryptographic keys as VMs are powered on and off.

Key Vault and SSH Keys

Secrets and certificates can be modeled as resources and provided by [Key Vault](#). You can use Azure PowerShell to create key vaults for [Windows VMs](#) and the Azure CLI for [Linux VMs](#). You can also create keys for encryption.

Key vault access policies grant permissions to keys, secrets, and certificates separately. For example, you can give a user access to only keys, but no permissions for secrets. However, permissions to access keys or secrets or certificates are at the vault level. In other words, [key vault access policy](#) does not support object level permissions.

When you connect to VMs, you should use public-key cryptography to provide a more secure way to sign in to them. This process involves a public and private key exchange using the secure shell (SSH) command to

authenticate yourself rather than a username and password. Passwords are vulnerable to brute-force attacks, especially on Internet-facing VMs such as web servers. With a secure shell (SSH) key pair, you can create a [Linux VM](#) that uses SSH keys for authentication, eliminating the need for passwords to sign-in. You can also use SSH keys to connect from a [Windows VM](#) to a Linux VM.

Managed identities for Azure resources

A common challenge when building cloud applications is how to manage the credentials in your code for authenticating to cloud services. Keeping the credentials secure is an important task. Ideally, the credentials never appear on developer workstations and aren't checked into source control. Azure Key Vault provides a way to securely store credentials, secrets, and other keys, but your code has to authenticate to Key Vault to retrieve them.

The managed identities for Azure resources feature in Azure Active Directory (Azure AD) solves this problem. The feature provides Azure services with an automatically managed identity in Azure AD. You can use the identity to authenticate to any service that supports Azure AD authentication, including Key Vault, without any credentials in your code. Your code that's running on a VM can request a token from two endpoints that are accessible only from within the VM. For more detailed information about this service, review the [managed identities for Azure resources](#) overview page.

Policies

[Azure policies](#) can be used to define the desired behavior for your organization's [Windows VMs](#) and [Linux VMs](#). By using policies, an organization can enforce various conventions and rules throughout the enterprise. Enforcement of the desired behavior can help mitigate risk while contributing to the success of the organization.

Role-based access control

Using [role-based access control \(RBAC\)](#), you can segregate duties within your team and grant only the amount of access to users on your VM that they need to perform their jobs. Instead of giving everybody unrestricted permissions on the VM, you can allow only certain actions. You can configure access control for the VM in the [Azure portal](#), using the [Azure CLI](#), or [Azure PowerShell](#).

Next steps

- Walk through the steps to monitor virtual machine security by using Azure Security Center for [Linux](#) or [Windows](#).

Azure Disk Encryption for Virtual Machine Scale Sets

1/19/2020 • 2 minutes to read • [Edit Online](#)

Azure Disk Encryption provides volume encryption for the OS and data disks of your virtual machines, helping protect and safeguard your data to meet organizational security and compliance commitments. To learn more, see [Azure Disk Encryption: Linux VMs](#) and [Azure Disk Encryption: Windows VMs](#)

Azure Disk Encryption can also be applied to Windows and Linux virtual machine scale sets, in these instances:

- Scale sets created with managed disks. Azure Disk encryption is not supported for native (or unmanaged) disk scale sets.
- OS and data volumes in Windows scale sets.
- Data volumes in Linux scale sets. OS disk encryption is NOT supported at present for Linux scale sets.

You can learn the fundamentals of Azure Disk Encryption for virtual machine scale sets in just a few minutes with the [Encrypt a virtual machine scale sets using the Azure CLI](#) or the [Encrypt a virtual machine scale sets using the Azure PowerShell](#) tutorials.

Next steps

- [Encrypt a virtual machine scale sets using the Azure Resource Manager](#)
- [Create and configure a key vault for Azure Disk Encryption](#)
- [Use Azure Disk Encryption with virtual machine scale set extension sequencing](#)

Security controls for Azure Virtual Machine Scale Sets

2/12/2020 • 2 minutes to read • [Edit Online](#)

This article documents the security controls built into Azure Virtual Machine Scale Sets.

A security control is a quality or feature of an Azure service that contributes to the service's ability to prevent, detect, and respond to security vulnerabilities.

For each control, we use "Yes" or "No" to indicate whether it is currently in place for the service, "N/A" for a control that is not applicable to the service. We might also provide a note or links to more information about an attribute.

Network

SECURITY CONTROL	YES/NO	NOTES
Service endpoint support	Yes	
VNet injection support	Yes	
Network Isolation and Firewalling support	Yes	
Forced tunneling support	Yes	See Configure forced tunneling using the Azure Resource Manager deployment model .

Monitoring & logging

SECURITY CONTROL	YES/NO	NOTES
Azure monitoring support (Log analytics, App insights, etc.)	Yes	See Monitor and update a Linux virtual machine in Azure and Monitor and update a Windows virtual machine in Azure .
Control and management plane logging and audit	Yes	
Data plane logging and audit	No	

Identity

SECURITY CONTROL	YES/NO	NOTES
Authentication	Yes	
Authorization	Yes	

Data protection

SECURITY CONTROL	YES/NO	NOTES
Server-side encryption at rest: Microsoft-managed keys	Yes	See Azure Disk Encryption for Virtual Machine Scale Sets .
Encryption in transit (such as ExpressRoute encryption, in VNet encryption, and VNet-VNet encryption)	Yes	Azure Virtual Machines supports ExpressRoute and VNet encryption. See In-transit encryption in VMs .
Server-side encryption at rest: customer-managed keys (BYOK)	Yes	Customer-managed keys is a supported Azure encryption scenario; see See Azure Disk Encryption for Virtual Machine Scale Sets
Column level encryption (Azure Data Services)	N/A	
API calls encrypted	Yes	Via HTTPS and TLS.

Configuration management

SECURITY CONTROL	YES/NO	NOTES
Configuration management support (versioning of configuration, etc.)	Yes	

Next steps

- Learn more about the [built-in security controls across Azure services](#).

How to monitor virtual machines in Azure

11/13/2019 • 5 minutes to read • [Edit Online](#)

With the significant growth of VMs hosted in Azure, it's important to identify performance and health issues that impact applications and infrastructure services they support. Basic monitoring is delivered by default with Azure by the metric types CPU usage, disk utilization, memory utilization, and network traffic collected by the host hypervisor. Additional metric and log data can be collected using [extensions](#) to configure diagnostics on your VMs from the guest operating system.

To detect and help diagnose performance and health issues with the guest operating system, .NET based or Java web application components running inside the VM, Azure Monitor delivers centralized monitoring with comprehensive features such as Azure Monitor for VMs and Application Insights.

Diagnostics and metrics

You can set up and monitor the collection of [diagnostics data](#) using [metrics](#) in the Azure portal, the Azure CLI, Azure PowerShell, and programming Applications Programming Interfaces (APIs). For example, you can:

- **Observe basic metrics for the VM.** On the Overview screen of the Azure portal, the basic metrics shown include CPU usage, network usage, total of disk bytes, and disk operations per second.
- **Enable the collection of boot diagnostics and view it using the Azure portal.** When bringing your own image to Azure or even booting one of the platform images, there can be many reasons why a VM gets into a non-bootable state. You can easily enable boot diagnostics when you create a VM by clicking **Enabled** for Boot Diagnostics under the Monitoring section of the Settings screen.

As VMs boot, the boot diagnostic agent captures boot output and stores it in Azure storage. This data can be used to troubleshoot VM boot issues. Boot diagnostics are not automatically enabled when you create a VM from command-line tools. Before enabling boot diagnostics, a storage account needs to be created for storing boot logs. If you enable boot diagnostics in the Azure portal, a storage account is automatically created for you.

If you didn't enable boot diagnostics when the VM was created, you can always enable it later by using [Azure CLI](#), [Azure PowerShell](#), or an [Azure Resource Manager template](#).

- **Enable the collection of guest OS diagnostics data.** When you create a VM, you have the opportunity on the settings screen to enable guest OS diagnostics. When you do enable the collection of diagnostics data, the [IaaS Diagnostics extension for Linux](#) or the [IaaS Diagnostics extension for Windows](#) is added to the VM, which enables you to collect additional disk, CPU, and memory data.

Using the collected diagnostics data, you can configure autoscaling for your VMs. You can also configure [Azure Monitor Logs](#) to store the data and set up alerts to let you know when performance isn't right.

Alerts

You can create [alerts](#) based on specific performance metrics. Examples of the issues you can be alerted about include when average CPU usage exceeds a certain threshold, or available free disk space drops below a certain amount. Alerts can be configured in the [Azure portal](#), using [Azure Resource Manager templates](#), or [Azure CLI](#).

Azure Service Health

[Azure Service Health](#) provides personalized guidance and support when issues in Azure services affect you, and

helps you prepare for upcoming planned maintenance. Azure Service Health alerts you and your teams using targeted and flexible notifications.

Azure Resource Health

[Azure Resource health](#) helps you diagnose and get support when an Azure issue impacts your resources. It informs you about the current and past health of your resources and helps you mitigate issues. Resource health provides technical support when you need help with Azure service issues.

Azure Activity Log

The [Azure Activity Log](#) is a subscription log that provides insight into subscription-level events that have occurred in Azure. The log includes a range of data, from Azure Resource Manager operational data to updates on Service Health events. You can click Activity Log in the Azure portal to view the log for your VM.

Some of the things you can do with the activity log include:

- Create an [alert on an Activity Log event](#).
- [Stream it to an Event Hub](#) for ingestion by a third-party service or custom analytics solution such as Power BI.
- Analyze it in Power BI using the [Power BI content pack](#).
- [Save it to a storage account](#) for archival or manual inspection. You can specify the retention time (in days) using the Log Profile.

You can also access activity log data by using [Azure PowerShell](#), the [Azure CLI](#), or [Monitor REST APIs](#).

[Azure Resource Logs](#) are logs emitted by your VM that provide rich, frequent data about its operation. Resource logs differ from the activity log by providing insight about operations that were performed within the VM.

Some of the things you can do with diagnostics logs include:

- [Save them to a storage account](#) for auditing or manual inspection. You can specify the retention time (in days) using Resource Diagnostic Settings.
- [Stream them to Event Hubs](#) for ingestion by a third-party service or custom analytics solution such as Power BI.
- Analyze them with [Log Analytics](#).

Advanced monitoring

For visibility of the application or service supported by the Azure VM and virtual machine scale sets, identification of issues with the guest OS or workload running in the VM to understand if it is impacting availability or performance of the application, or is an issue with the application, enable both [Azure Monitor for VMs](#) and [Application Insights](#).

Azure Monitor for VMs monitors your Azure virtual machines (VM) at scale by analyzing the performance and health of your Windows and Linux VMs, including the different processes and interconnected dependencies on other resources and external processes it discovers. It includes several trend performance charts to help during investigation of problems and assess capacity of your VMs. The dependency map shows monitored and unmonitored machines, failed and active network connections between processes and these machines, and shows trend charts with standard network connection metrics. Combined with Application Insights, you monitor your application and capture telemetry such as HTTP requests, exceptions, etc. so you can correlate issues between the VMs and your application. Configure [Azure Monitor alerts](#) to alert you on important conditions detected from monitoring data collected by Azure Monitor for VMs.

Next steps

- Walk through the steps in [Monitor a Windows Virtual Machine with Azure PowerShell](#) or [Monitor a Linux](#)

[Virtual Machine with the Azure CLI.](#)

- Learn more about the best practices around [Monitoring and diagnostics](#).

Virtual machine vCPU quotas

1/10/2020 • 2 minutes to read • [Edit Online](#)

The vCPU quotas for virtual machines and virtual machine scale sets are arranged in two tiers for each subscription, in each region. The first tier is the Total Regional vCPUs, and the second tier is the various VM size family cores such as the D-series vCPUs. Any time a new VM is deployed the vCPUs for the VM must not exceed the vCPU quota for the VM size family or the total regional vCPU quota. If either of those quotas are exceeded, the VM deployment will not be allowed. There is also a quota for the overall number of virtual machines in the region. The details on each of these quotas can be seen in the **Usage + quotas** section of the **Subscription** page in the [Azure portal](#), or you can query for the values using PowerShell.

Check usage

You can use the [Get-AzVMUsage](#) cmdlet to check on your quota usage.

```
Get-AzVMUsage -Location "East US"
```

The output will look similar to this:

Name	Current	Value	Limit	Unit
Availability Sets	0	2000	Count	
Total Regional vCPUs	4	260	Count	
Virtual Machines	4	10000	Count	
Virtual Machine Scale Sets	1	2000	Count	
Standard B Family vCPUs	1	10	Count	
Standard DSV2 Family vCPUs	1	100	Count	
Standard Dv2 Family vCPUs	2	100	Count	
Basic A Family vCPUs	0	100	Count	
Standard A0-A7 Family vCPUs	0	250	Count	
Standard A8-A11 Family vCPUs	0	100	Count	
Standard D Family vCPUs	0	100	Count	
Standard G Family vCPUs	0	100	Count	
Standard DS Family vCPUs	0	100	Count	
Standard GS Family vCPUs	0	100	Count	
Standard F Family vCPUs	0	100	Count	
Standard FS Family vCPUs	0	100	Count	
Standard NV Family vCPUs	0	24	Count	
Standard NC Family vCPUs	0	48	Count	
Standard H Family vCPUs	0	8	Count	
Standard Av2 Family vCPUs	0	100	Count	
Standard LS Family vCPUs	0	100	Count	
Standard Dv2 Promo Family vCPUs	0	100	Count	
Standard DSV2 Promo Family vCPUs	0	100	Count	
Standard MS Family vCPUs	0	0	Count	
Standard Dv3 Family vCPUs	0	100	Count	
Standard DSV3 Family vCPUs	0	100	Count	
Standard Ev3 Family vCPUs	0	100	Count	
Standard ESv3 Family vCPUs	0	100	Count	
Standard FSv2 Family vCPUs	0	100	Count	
Standard ND Family vCPUs	0	0	Count	
Standard NCv2 Family vCPUs	0	0	Count	
Standard NCv3 Family vCPUs	0	0	Count	
Standard LSv2 Family vCPUs	0	0	Count	
Standard Storage Managed Disks	2	10000	Count	
Premium Storage Managed Disks	1	10000	Count	

Reserved VM Instances

Reserved VM Instances, which are scoped to a single subscription without VM size flexibility, will add a new aspect to the vCPU quotas. These values describe the number of instances of the stated size that must be deployable in the subscription. They work as a placeholder in the quota system to ensure that quota is reserved to ensure reserved VM instances are deployable in the subscription. For example, if a specific subscription has 10 Standard_D1 reserved VM instances the usages limit for Standard_D1 reserved VM instances will be 10. This will cause Azure to ensure that there are always at least 10 vCPUs available in the Total Regional vCPUs quota to be used for Standard_D1 instances and there are at least 10 vCPUs available in the Standard D Family vCPU quota to be used for Standard_D1 instances.

If a quota increase is required to purchase a Single Subscription RI, you can [request a quota increase](#) on your subscription.

Next steps

For more information about billing and quotas, see [Azure subscription and service limits, quotas, and constraints](#).

Azure virtual machine scale sets FAQs

1/19/2020 • 25 minutes to read • [Edit Online](#)

Get answers to frequently asked questions about virtual machine scale sets in Azure.

Top frequently asked questions for scale sets

How many VMs can I have in a scale set?

A scale set can have 0 to 1,000 VMs based on platform images, or 0 to 600 VMs based on custom images.

Are data disks supported within scale sets?

Yes. A scale set can define an attached data disks configuration that applies to all VMs in the set. For more information, see [Azure scale sets and attached data disks](#). Other options for storing data include:

- Azure files (SMB shared drives)
- OS drive
- Temp drive (local, not backed by Azure Storage)
- Azure data service (for example, Azure tables, Azure blobs)
- External data service (for example, remote database)

Which Azure regions support scale sets?

All regions support scale sets.

How do I create a scale set by using a custom image?

Create and capture a VM image, then use that as the source for your scale set. For a tutorial on how to create and use a custom VM image, you can use the [Azure CLI](#) or [Azure PowerShell](#)

If I reduce my scale set capacity from 20 to 15, which VMs are removed?

Virtual machines are removed from the scale set evenly across update domains and fault domains to maximize availability. VMs with the highest IDs are removed first.

What if I then increase the capacity from 15 to 18?

If you increase capacity to 18, then 3 new VMs are created. Each time, the VM instance ID is incremented from the previous highest value (for example, 20, 21, 22). VMs are balanced across fault domains and update domains.

When I'm using multiple extensions in a scale set, can I enforce an execution sequence?

Yes, you can use scale set [extension sequencing](#).

Do scale sets work with Azure availability sets?

A regional (non-zonal) scale set uses *placement groups*, which act as an implicit availability set with five fault domains and five update domains. Scale sets of more than 100 VMs span multiple placement groups. For more information about placement groups, see [Working with large virtual machine scale sets](#). An availability set of VMs can exist in the same virtual network as a scale set of VMs. A common configuration is to put control node VMs (which often require unique configuration) in an availability set and put data nodes in the scale set.

Do scale sets work with Azure availability zones?

Yes! For more information, see the [scale set zone doc](#).

Autoscale

What are best practices for Azure Autoscale?

For best practices for Autoscale, see [Best practices for autoscaling virtual machines](#).

Where do I find metric names for autoscaling that uses host-based metrics?

For metric names for autoscaling that uses host-based metrics, see [Supported metrics with Azure Monitor](#).

Are there any examples of autoscaling based on an Azure Service Bus topic and queue length?

Yes. For examples of autoscaling based on an Azure Service Bus topic and queue length, see [Azure Monitor autoscaling common metrics](#).

For a Service Bus queue, use the following JSON:

```
"metricName": "MessageCount",
"metricNamespace": "",
"metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ServiceBus/namespaces/mySB/queues/myqueue"
```

For a storage queue, use the following JSON:

```
"metricName": "ApproximateMessageCount",
"metricNamespace": "",
"metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ClassicStorage/storageAccounts/mystorage/services/que
ue/queues/mystoragequeue"
```

Replace example values with your resource Uniform Resource Identifiers (URIs).

Should I autoscale by using host-based metrics or a diagnostics extension?

You can create an autoscale setting on a VM to use host-level metrics or guest OS-based metrics.

For a list of supported metrics, see [Azure Monitor autoscaling common metrics](#).

For a full sample for virtual machine scale sets, see [Advanced autoscale configuration by using Resource Manager templates for virtual machine scale sets](#).

The sample uses the host-level CPU metric and a message count metric.

How do I set alert rules on a virtual machine scale set?

You can create alerts on metrics for virtual machine scale sets via PowerShell or Azure CLI. For more information, see [Azure Monitor PowerShell quickstart samples](#) and [Azure Monitor cross-platform CLI quickstart samples](#).

The TargetResourceId of the virtual machine scale set looks like this:

```
/subscriptions/yoursubscriptionid/resourceGroups/yourresourcegroup/providers/Microsoft.Compute/virtualMach
ineScaleSets/yourvmssname
```

You can choose any VM performance counter as the metric to set an alert for. For more information, see [Guest OS metrics for Resource Manager-based Windows VMs](#) and [Guest OS metrics for Linux VMs](#) in the [Azure Monitor autoscaling common metrics](#) article.

How do I set up autoscale on a virtual machine scale set by using PowerShell?

To set up autoscale on a virtual machine scale set by using PowerShell, see [automatically scale a virtual machine scale set](#). You can also configure autoscale with the [Azure CLI](#) and [Azure templates](#).

If I have stopped (deallocated) a VM, is that VM started as part of an autoscale operation?

No. If autoscale rules require additional VM instances as part of a scale set, a new VM instance is created. VM instances that are stopped (deallocated) are not started as part of an autoscale event. However, those stopped

(deallocated) VMs may be deleted as part of an autoscale event that scales in the number of instances, the same way that any VM instance may be deleted based on the order of VM instance ID.

Certificates

How do I securely ship a certificate to the VM?

To securely ship a certificate to the VM, you can install a customer certificate directly into a Windows certificate store from the customer's key vault.

Use the following JSON:

```
"secrets": [
  {
    "sourceVault": {
      "id": "/subscriptions/{subscriptionid}/resourceGroups/myrg1/providers/Microsoft.KeyVault/vaults/mykeyvault1"
    },
    "vaultCertificates": [
      {
        "certificateUrl": "https://mykeyvault1.vault.azure.net/secrets/{secretname}/{secret-version}",
        "certificateStore": "certificateStoreName"
      }
    ]
  }
]
```

The code supports Windows and Linux.

For more information, see [Create or update a virtual machine scale set](#).

How do I use self-signed certificates provisioned for Azure Service Fabric Clusters?

For the latest example use the following azure CLI statement within azure shell, read Service Fabrics CLI module Example documentation, which will be printed to stdout:

```
az sf cluster create -h
```

Self-signed certificates can not be used for distributed trust provided by a Certificate Authority, and should not be used for any Service Fabric Cluster intended to host enterprise production solutions; for additional Service Fabric Security guidance, review [Azure Service Fabric Security Best Practices](#) and [Service Fabric cluster security scenarios](#).

Can I specify an SSH key pair to use for SSH authentication with a Linux virtual machine scale set from a Resource Manager template?

Yes. The REST API for **osProfile** is similar to the standard VM REST API.

Include **osProfile** in your template:

```

"osProfile": {
    "computerName": "[variables('vmName')]",
    "adminUsername": "[parameters('adminUserName')]",
    "linuxConfiguration": {
        "disablePasswordAuthentication": "true",
        "ssh": {
            "publicKeys": [
                {
                    "path": "[variables('sshKeyPath')]",
                    "keyData": "[parameters('sshKeyData')]"
                }
            ]
        }
    }
}

```

This JSON block is used in [this Azure quickstart template](#).

For more information, see [Create or update a virtual machine scale set](#).

How do I remove deprecated certificates?

To remove deprecated certificates, remove the old certificate from the vault certificates list. Leave all the certificates that you want to remain on your computer in the list. This does not remove the certificate from all your VMs. It also does not add the certificate to new VMs that are created in the virtual machine scale set.

To remove the certificate from existing VMs, use a custom script extension to manually remove the certificates from your certificate store.

How do I inject an existing SSH public key into the virtual machine scale set SSH layer during provisioning?

If you are providing the VMs only with a public SSH key, you don't need to put the public keys in Key Vault. Public keys are not secret.

You can provide SSH public keys in plain text when you create a Linux VM:

```

"linuxConfiguration": {
    "ssh": {
        "publicKeys": [
            {
                "path": "path",
                "keyData": "publickey"
            }
        ]
    }
}

```

LINUXCONFIGURATION ELEMENT NAME	REQUIRED	TYPE	DESCRIPTION
ssh	No	Collection	Specifies the SSH key configuration for a Linux OS
path	Yes	String	Specifies the Linux file path where the SSH keys or certificate should be located
keyData	Yes	String	Specifies a base64-encoded SSH public key

For an example, see [the 101-vm-sshkey GitHub quickstart template](#).

When I run `Update-AzVmss` after adding more than one certificate from the same key vault, I see the following message:

```
Update-AzVmss: List secret contains repeated instances of /subscriptions/<my-subscription-id>/resourceGroups/internal-rg-dev/providers/Microsoft.KeyVault/vaults/internal-keyvault-dev, which is disallowed.
```

This can happen if you try to re-add the same vault instead of using a new vault certificate for the existing source vault. The `Add-AzVmssSecret` command does not work correctly if you are adding additional secrets.

To add more secrets from the same key vault, update the `$vmss.properties.osProfile.secrets[0].vaultCertificates` list.

For the expected input structure, see [Create or update a virtual machine set](#).

Find the secret in the virtual machine scale set object that is in the key vault. Then, add your certificate reference (the URL and the secret store name) to the list associated with the vault.

NOTE

Currently, you cannot remove certificates from VMs by using the virtual machine scale set API.

New VMs will not have the old certificate. However, VMs that have the certificate and which are already deployed will have the old certificate.

Can I push certificates to the virtual machine scale set without providing the password, when the certificate is in the secret store?

You do not need to hard-code passwords in scripts. You can dynamically retrieve passwords with the permissions you use to run the deployment script. If you have a script that moves a certificate from the secret store key vault, the secret store `get certificate` command also outputs the password of the .pfx file.

How does the Secrets property of virtualMachineProfile.osProfile for a virtual machine scale set work? Why do I need the sourceVault value when I have to specify the absolute URI for a certificate by using the certificateUrl property?

A Windows Remote Management (WinRM) certificate reference must be present in the Secrets property of the OS profile.

The purpose of indicating the source vault is to enforce access control list (ACL) policies that exist in a user's Azure Cloud Service model. If the source vault isn't specified, users who do not have permissions to deploy or access secrets to a key vault would be able to through a Compute Resource Provider (CRP). ACLs exist even for resources that do not exist.

If you provide an incorrect source vault ID but a valid key vault URL, an error is reported when you poll the operation.

If I add secrets to an existing virtual machine scale set, are the secrets injected into existing VMs, or only into new ones?

Certificates are added to all your VMs, even pre-existing ones. If your virtual machine scale set upgradePolicy property is set to **manual**, the certificate is added to the VM when you perform a manual update on the VM.

Where do I put certificates for Linux VMs?

To learn how to deploy certificates for Linux VMs, see [Deploy certificates to VMs from a customer-managed key vault](#).

How do I add a new vault certificate to a new certificate object?

To add a vault certificate to an existing secret, see the following PowerShell example. Use only one secret object.

```
$newVaultCertificate = New-AzVmssVaultCertificateConfig -CertificateStore MY -CertificateUrl
https://sansunallapps1.vault.azure.net:443/secrets/dg-private-enc/55fa0332edc44a84ad655298905f1809

$vmss.VirtualMachineProfile.OsProfile.Secrets[0].VaultCertificates.Add($newVaultCertificate)

Update-AzVmss -VirtualMachineScaleSet $vmss -ResourceGroup $rg -Name $vmssName
```

What happens to certificates if you reimagine a VM?

If you reimagine a VM, certificates are deleted. Reimaging deletes the entire OS disk.

What happens if you delete a certificate from the key vault?

If the secret is deleted from the key vault, and then you run `stop deallocate` for all your VMs and then start them again, you encounter a failure. The failure occurs because the CRP needs to retrieve the secrets from the key vault, but it cannot. In this scenario, you can delete the certificates from the virtual machine scale set model.

The CRP component does not persist customer secrets. If you run `stop deallocate` for all VMs in the virtual machine scale set, the cache is deleted. In this scenario, secrets are retrieved from the key vault.

You don't encounter this problem when scaling out because there is a cached copy of the secret in Azure Service Fabric (in the single-fabric tenant model).

Why do I have to specify the certificate version when I use Key Vault?

The purpose of the Key Vault requirement to specify the certificate version is to make it clear to the user what certificate is deployed on their VMs.

If you create a VM and then update your secret in the key vault, the new certificate is not downloaded to your VMs. But your VMs appear to reference it, and new VMs get the new secret. To avoid this, you are required to reference a secret version.

My team works with several certificates that are distributed to us as .cer public keys. What is the recommended approach for deploying these certificates to a virtual machine scale set?

To deploy .cer public keys to a virtual machine scale set, you can generate a .pfx file that contains only .cer files. To do this, use `x509ContentType = Pfx`. For example, load the .cer file as an `x509Certificate2` object in C# or PowerShell, and then call the method.

For more information, see [X509Certificate.Export Method \(X509ContentType, String\)](#).

How do I pass in certificates as base64 strings?

To emulate passing in a certificate as a base64 string, you can extract the latest versioned URL in a Resource Manager template. Include the following JSON property in your Resource Manager template:

```
"certificateUrl": "[reference(resourceId(parameters('vaultResourceGroup'), 'Microsoft.KeyVault/vaults/secrets', parameters('vaultName'), parameters('secretName')), '2015-06-01').secretUriWithVersion]"
```

Do I have to wrap certificates in JSON objects in key vaults?

In virtual machine scale sets and VMs, certificates must be wrapped in JSON objects.

We also support the content type application/x-pkcs12.

We currently do not support .cer files. To use .cer files, export them into .pfx containers.

Compliance and Security

Are virtual machine scale sets PCI-compliant?

Virtual machine scale sets are a thin API layer on top of the CRP. Both components are part of the compute

platform in the Azure service tree.

From a compliance perspective, virtual machine scale sets are a fundamental part of the Azure compute platform. They share a team, tools, processes, deployment methodology, security controls, just-in-time (JIT) compilation, monitoring, alerting, and so on, with the CRP itself. Virtual machine scale sets are Payment Card Industry (PCI)-compliant because the CRP is part of the current PCI Data Security Standard (DSS) attestation.

For more information, see [the Microsoft Trust Center](#).

Does managed identities for Azure resources work with virtual machine scale sets?

Yes. You can see some example MSI templates in Azure Quickstart templates for [Linux](#) and [Windows](#).

Deleting

Will the locks I set in place on virtual machine scale set instances be respected when deleting instances?

In the Azure Portal, you have the ability to delete an individual instance or bulk delete by selecting multiple instances. If you attempt to delete a single instance that has a lock in place, the lock is respected and you will not be able to delete the instance. However, if you bulk select multiple instances and any of those instances have a lock in place, the lock(s) will not be respected and all of the selected instances will be deleted.

In Azure CLI, you only have the ability to delete an individual instance. If you attempt to delete a single instance that has a lock in place, the lock is respected and you will not be able to delete that instance.

Extensions

How do I delete a virtual machine scale set extension?

To delete a virtual machine scale set extension, use the following PowerShell example:

```
$vmss = Get-AzVmss -ResourceGroupName "resource_group_name" -VMScaleSetName "vmssName"

$vmss=Remove-AzVmssExtension -VirtualMachineScaleSet $vmss -Name "extensionName"

Update-AzVmss -ResourceGroupName "resource_group_name" -VMScaleSetName "vmssName" -VirtualMachineScaleSet $vmss
```

You can find the extensionName value in `$vmss`.

Is there a virtual machine scale set template example that integrates with Azure Monitor logs?

For a virtual machine scale set template example that integrates with Azure Monitor logs, see the second example in [Deploy an Azure Service Fabric cluster and enable monitoring by using Azure Monitor logs](#).

How do I add an extension to all VMs in my virtual machine scale set?

If update policy is set to **automatic**, redeploying the template with the new extension properties updates all VMs.

If update policy is set to **manual**, first update the extension, and then manually update all instances in your VMs.

If the extensions associated with an existing virtual machine scale set are updated, are existing VMs affected?

If the extension definition in the virtual machine scale set model is updated and the upgradePolicy property is set to **automatic**, it updates the VMs. If the upgradePolicy property is set to **manual**, extensions are flagged as not matching the model.

Are extensions run again when an existing machine is service-healed or reimaged?

If an existing VM is service-healed, it appears as a reboot, and the extensions are not run again. If a VM is reimaged, the process is similar replacing the OS drive with the source image. Any specialization from the latest model, such as extensions, are run again.

How do I join a virtual machine scale set to an Active Directory domain?

To join a virtual machine scale set to an Active Directory (AD) domain, you can define an extension.

To define an extension, use the `JsonADDomainExtension` property:

```
"extensionProfile": {
    "extensions": [
        {
            "name": "joindomain",
            "properties": {
                "publisher": "Microsoft.Compute",
                "type": "JsonADDomainExtension",
                "typeHandlerVersion": "1.3",
                "settings": {
                    "Name": "[parameters('domainName')]",
                    "OUPath": "[variables('ouPath')]",
                    "User": "[variables('domainAndUsername')]",
                    "Restart": "true",
                    "Options": "[variables('domainJoinOptions')]"
                },
                "protectedSettings": {
                    "Password": "[parameters('domainJoinPassword')]"
                }
            }
        }
    ]
}
```

My virtual machine scale set extension is trying to install something that requires a reboot.

If your virtual machine scale set extension is trying to install something that requires a reboot, you can use the Azure Automation Desired State Configuration (Automation DSC) extension. If the operating system is Windows Server 2012 R2, Azure pulls in the Windows Management Framework (WMF) 5.0 setup, reboots, and then continues with the configuration.

How do I turn on antimalware in my virtual machine scale set?

To turn on antimalware on your virtual machine scale set, use the following PowerShell example:

```
$rgname = 'autolap'
$vmssname = 'autolapbr'
$location = 'eastus'

# Retrieve the most recent version number of the extension.
$allVersions= (Get-AzVMEExtensionImage -Location $location -PublisherName "Microsoft.Azure.Security" -Type "IaaSAntimalware").Version
$versionString = $allVersions[($allVersions.count)-1].Split(".")[0] + "." + $allVersions[($allVersions.count)-1].Split(".")[1]

$VMSS = Get-AzVmss -ResourceGroupName $rgname -VMScaleSetName $vmssname
echo $VMSS
Add-AzVmssExtension -VirtualMachineScaleSet $VMSS -Name "IaaSAntimalware" -Publisher "Microsoft.Azure.Security" -Type "IaaSAntimalware" -TypeHandlerVersion $versionString
Update-AzVmss -ResourceGroupName $rgname -Name $vmssname -VirtualMachineScaleSet $VMSS
```

How do I execute a custom script that's hosted in a private storage account?

To execute a custom script that's hosted in a private storage account, set up protected settings with the storage account key and name. For more information, see [Custom Script Extension](#).

Passwords

How do I reset the password for VMs in my virtual machine scale set?

There are two main ways to change the password for VMs in scale sets.

- Change the virtual machine scale set model directly. Available with API 2017-12-01 and later.

Update the admin credentials directly in the scale set model (for example using the Azure Resource Explorer, PowerShell or CLI). Once the scale set is updated, all new VMs have the new credentials. Existing VMs only have the new credentials if they are reimaged.

- Reset the password using the VM access extensions.

Use the following PowerShell example:

```
$vmssName = "myvmss"
$vmssResourceGroup = "myvmssrg"
$publicConfig = @{"UserName" = "newuser"}
$privateConfig = @{"Password" = "*****"}

$extName = "VMAccessAgent"
$publisher = "Microsoft.Compute"
$vmss = Get-AzVmss -ResourceGroupName $vmssResourceGroup -VMScaleSetName $vmssName
$vmss = Add-AzVmssExtension -VirtualMachineScaleSet $vmss -Name $extName -Publisher $publisher -Setting
$publicConfig -ProtectedSetting $privateConfig -Type $extName -TypeHandlerVersion "2.0" -
AutoUpgradeMinorVersion $true
Update-AzVmss -ResourceGroupName $vmssResourceGroup -Name $vmssName -VirtualMachineScaleSet $vmss
```

Networking

Is it possible to assign a Network Security Group (NSG) to a scale set, so that it applies to all the VM NICs in the set?

Yes. A Network Security Group can be applied directly to a scale set by referencing it in the networkInterfaceConfigurations section of the network profile. Example:

```

"networkProfile": {
    "networkInterfaceConfigurations": [
        {
            "name": "nic1",
            "properties": {
                "primary": "true",
                "ipConfigurations": [
                    {
                        "name": "ip1",
                        "properties": {
                            "subnet": {
                                "id": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/virtualNetworks/', variables('vnetName'), '/subnets/subnet1')]"
                            },
                            "loadBalancerInboundNatPools": [
                                {
                                    "id": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/loadBalancers/', variables('lbName'), '/inboundNatPools/natPool1')]"
                                }
                            ],
                            "loadBalancerBackendAddressPools": [
                                {
                                    "id": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/loadBalancers/', variables('lbName'), '/backendAddressPools/addressPool1')]"
                                }
                            ]
                        }
                    }
                ],
                "networkSecurityGroup": {
                    "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/networkSecurityGroups/', variables('nsgName'))]"
                }
            }
        }
    ]
}

```

How do I do a VIP swap for virtual machine scale sets in the same subscription and same region?

If you have two virtual machine scale sets with Azure Load Balancer front-ends, and they are in the same subscription and region, you could deallocate the public IP addresses from each one, and assign to the other. See [VIP Swap: Blue-green deployment in Azure Resource Manager](#) for example. This does imply a delay though as the resources are deallocated/allocated at the network level. A faster option is to use Azure Application Gateway with two backend pools, and a routing rule. Alternatively, you could host your application with [Azure App service](#) which provides support for fast switching between staging and production slots.

How do I specify a range of private IP addresses to use for static private IP address allocation?

IP addresses are selected from a subnet that you specify.

The allocation method of virtual machine scale set IP addresses is always "dynamic," but that doesn't mean that these IP addresses can change. In this case, "dynamic" only means that you do not specify the IP address in a PUT request. Specify the static set by using the subnet.

How do I deploy a virtual machine scale set to an existing Azure virtual network?

To deploy a virtual machine scale set to an existing Azure virtual network, see [Deploy a virtual machine scale set to an existing virtual network](#).

Can I use scale sets with Accelerated Networking?

Yes. To use accelerated networking, set enableAcceleratedNetworking to true in your scale set's networkInterfaceConfigurations settings. For example

```
"networkProfile": {  
    "networkInterfaceConfigurations": [  
        {  
            "name": "niconfig1",  
            "properties": {  
                "primary": true,  
                "enableAcceleratedNetworking" : true,  
                "ipConfigurations": [  
                ]  
            }  
        }  
    ]  
}
```

How can I configure the DNS servers used by a scale set?

To create a virtual machine scale set with a custom DNS configuration, add a dnsSettings JSON packet to the scale set networkInterfaceConfigurations section. Example:

```
"dnsSettings":{  
    "dnsServers": ["10.0.0.6", "10.0.0.5"]  
}
```

How can I configure a scale set to assign a public IP address to each VM?

To create a virtual machine scale set that assigns a public IP address to each VM, make sure the API version of the Microsoft.Compute/virtualMachineScaleSets resource is 2017-03-30, and add a *publicipaddressconfiguration* JSON packet to the scale set ipConfigurations section. Example:

```
"publicipaddressconfiguration": {  
    "name": "pub1",  
    "properties": {  
        "idleTimeoutInMinutes": 15  
    }  
}
```

Can I configure a scale set to work with multiple Application Gateways?

Yes. You can add the resource IDs for multiple Application Gateway backend address pools to the *applicationGatewayBackendAddressPools* list in the *ipConfigurations* section of your scale set network profile.

Scale

In what case would I create a virtual machine scale set with fewer than two VMs?

One reason to create a virtual machine scale set with fewer than two VMs would be to use the elastic properties of a virtual machine scale set. For example, you could deploy a virtual machine scale set with zero VMs to define your infrastructure without paying VM running costs. Then, when you are ready to deploy VMs, increase the "capacity" of the virtual machine scale set to the production instance count.

Another reason you might create a virtual machine scale set with fewer than two VMs is if you're concerned less with availability than in using an availability set with discrete VMs. Virtual machine scale sets give you a way to work with undifferentiated compute units that are fungible. This uniformity is a key differentiator for virtual machine scale sets versus availability sets. Many stateless workloads do not track individual units. If the workload drops, you can scale down to one compute unit, and then scale up to many when the workload increases.

How do I change the number of VMs in a virtual machine scale set?

To change the number of VMs in a virtual machine scale set in the Azure portal, from the virtual machine scale set properties section, click on the "Scaling" blade and use the slider bar.

How do I define custom alerts for when certain thresholds are reached?

You have some flexibility in how you handle alerts for specified thresholds. For example, you can define customized webhooks. The following webhook example is from a Resource Manager template:

```
{
  "type": "Microsoft.Insights/autoscaleSettings",
  "apiVersion": "[variables('insightsApi')]",
  "name": "autoscale",
  "location": "[parameters('resourceLocation')]",
  "dependsOn": [
    "[concat('Microsoft.Compute/virtualMachineScaleSets/', parameters('vmSSName'))]"
  ],
  "properties": {
    "name": "autoscale",
    "targetResourceUri": "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/', resourceGroup().name, '/providers/Microsoft.Compute/virtualMachineScaleSets/', parameters('vmSSName'))]",
    "enabled": true,
    "notifications": [
      {
        "operation": "Scale",
        "email": {
          "sendToSubscriptionAdministrator": true,
          "sendToSubscriptionCoAdministrators": true,
          "customEmails": [
            "youremail@address.com"
          ]
        },
        "webhooks": [
          {
            "serviceUri": "<service uri>",
            "properties": {
              "key1": "custommetric",
              "key2": "scalevmss"
            }
          }
        ]
      }
    ]
  }
}
```

Patching and operations

Can I create a scale set in an existing resource group?

Yes, you can create a scale set in an existing resource group.

Can I move a scale set to another resource group?

Yes, you can move scale set resources to a new subscription or resource group.

How to I update my virtual machine scale set to a new image? How do I manage patching?

To update your virtual machine scale set to a new image, and to manage patching, see [Upgrade a virtual machine scale set](#).

Can I use the reimagine operation to reset a VM without changing the image? (That is, I want reset a VM to factory settings rather than to a new image.)

Yes, you can use the reimagine operation to reset a VM without changing the image. However, if your virtual

machine scale set references a platform image with `version = latest`, your VM can update to a later OS image when you call `reimage`.

Is it possible to integrate scale sets with Azure Monitor logs?

Yes, you can by installing the Azure Monitor extension on the scale set VMs. Here is an Azure CLI example:

```
az vmss extension set --name MicrosoftMonitoringAgent --publisher Microsoft.EnterpriseCloud.Monitoring --resource-group Team-03 --vmss-name nt01 --settings "{'workspaceId': '<your workspace ID here>'}" --protected-settings "{'workspaceKey': '<your workspace key here>'}"
```

You can find the required workspaceId and workspaceKey in the Log Analytics workspace of Azure portal. On the Overview page, click the Settings tile. Click the Connected Sources tab at the top.

NOTE

If your scale set `upgradePolicy` is set to Manual, you need to apply the extension to all VMs in the set by calling `upgrade` on them. In CLI this would be `az vmss update-instances`.

NOTE

This article was recently updated to use the term Azure Monitor logs instead of Log Analytics. Log data is still stored in a Log Analytics workspace and is still collected and analyzed by the same Log Analytics service. We are updating the terminology to better reflect the role of [logs in Azure Monitor](#). See [Azure Monitor terminology changes](#) for details.

Troubleshooting

How do I turn on boot diagnostics?

To turn on boot diagnostics, first, create a storage account. Then, put this JSON block in your virtual machine scale set **virtualMachineProfile**, and update the virtual machine scale set:

```
"diagnosticsProfile": {  
    "bootDiagnostics": {  
        "enabled": true,  
        "storageUri": "http://yourstorageaccount.blob.core.windows.net"  
    }  
}
```

When a new VM is created, the `InstanceView` property of the VM shows the details for the screenshot, and so on. Here's an example:

```
"bootDiagnostics": {  
    "consoleScreenshotBlobUri": "https://o0sz3nhtbmkg6geswarm5.blob.core.windows.net/bootdiagnostics-swarmagen-4157d838-8335-4f78-bf0e-b616a99bc8bd/swarm-agent-9574AE92vmss-0_2.4157d838-8335-4f78-bf0e-b616a99bc8bd.screenshot.bmp",  
    "serialConsoleLogBlobUri": "https://o0sz3nhtbmkg6geswarm5.blob.core.windows.net/bootdiagnostics-swarmagen-4157d838-8335-4f78-bf0e-b616a99bc8bd/swarm-agent-9574AE92vmss-0_2.4157d838-8335-4f78-bf0e-b616a99bc8bd.serialconsole.log"  
}
```

Virtual machine properties

How do I get property information for each VM without making multiple calls? For example, how would I get the fault domain for each of the 100 VMs in my virtual machine scale set?

To get property information for each VM without making multiple calls, you can call `ListVMInstanceViews` by doing a REST API `GET` on the following resource URI:

```
/subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.Compute/virtualMachineScaleSets/<scaleset_name>/virtualMachines?$expand=instanceView&$select=instanceView
```

Can I pass different extension arguments to different VMs in a virtual machine scale set?

No, you cannot pass different extension arguments to different VMs in a virtual machine scale set. However, extensions can act based on the unique properties of the VM they are running on, such as on the machine name. Extensions also can query instance metadata on <http://169.254.169.254> to get more information about the VM.

Why are there gaps between my virtual machine scale set VM machine names and VM IDs? For example: 0, 1, 3...

There are gaps between your virtual machine scale set VM machine names and VM IDs because your virtual machine scale set **overprovision** property is set to the default value of **true**. If overprovisioning is set to **true**, more VMs than requested are created. Extra VMs are then deleted. In this case, you gain increased deployment reliability, but at the expense of contiguous naming and contiguous Network Address Translation (NAT) rules.

You can set this property to **false**. For small virtual machine scale sets, this doesn't significantly affect deployment reliability.

What is the difference between deleting a VM in a virtual machine scale set and deallocating the VM? When should I choose one over the other?

The main difference between deleting a VM in a virtual machine scale set and deallocating the VM is that `deallocate` doesn't delete the virtual hard disks (VHDs). There are storage costs associated with running `stop deallocate`. You might use one or the other for one of the following reasons:

- You want to stop paying compute costs, but you want to keep the disk state of the VMs.
- You want to start a set of VMs more quickly than you could scale out a virtual machine scale set.
 - Related to this scenario, you might have created your own autoscale engine and want a faster end-to-end scale.
- You have a virtual machine scale set that is unevenly distributed across fault domains or update domains. This might be because you selectively deleted VMs, or because VMs were deleted after overprovisioning. Running `stop deallocate` followed by `start` on the virtual machine scale set evenly distributes the VMs across fault domains or update domains.

How do I take a snapshot of a virtual machine scale set instance?

Create a snapshot from an instance of a virtual machine scale set.

```
$rgname = "myResourceGroup"
$vmssname = "myVMSScaleSet"
$Id = 0
$location = "East US"

$vmss1 = Get-AzVmssVM -ResourceGroupName $rgname -VMScaleSetName $vmssname -InstanceId $Id
$snapshotconfig = New-AzSnapshotConfig -Location $location -AccountType Standard_LRS -OsType Windows -
CreateOption Copy -SourceUri $vmss1.StorageProfile.OsDisk.ManagedDisk.id
New-AzSnapshot -ResourceGroupName $rgname -SnapshotName 'mySnapshot' -Snapshot $snapshotconfig
```

Create a managed disk from the snapshot.

```
$snapshotName = "mySnapshot"
$snapshot = Get-AzSnapshot -ResourceGroupName $rgname -SnapshotName $snapshotName
$diskConfig = New-AzDiskConfig -AccountType Premium_LRS -Location $location -CreateOption Copy -
SourceResourceId $snapshot.Id
$osDisk = New-AzDisk -Disk $diskConfig -ResourceGroupName $rgname -DiskName ($snapshotName + '_Disk')
```

Design Considerations For Scale Sets

1/19/2020 • 4 minutes to read • [Edit Online](#)

This article discusses design considerations for Virtual Machine Scale Sets. For information about what Virtual Machine Scale Sets are, refer to [Virtual Machine Scale Sets Overview](#).

When to use scale sets instead of virtual machines?

Generally, scale sets are useful for deploying highly available infrastructure where a set of machines has similar configuration. However, some features are only available in scale sets while other features are only available in VMs. In order to make an informed decision about when to use each technology, you should first take a look at some of the commonly used features that are available in scale sets but not VMs:

Scale set-specific features

- Once you specify the scale set configuration, you can update the *capacity* property to deploy more VMs in parallel. This process is better than writing a script to orchestrate deploying many individual VMs in parallel.
- You can [use Azure Autoscale to automatically scale a scale set](#) but not individual VMs.
- You can [reimage scale set VMs](#) but not individual VMs.
- You can [overprovision](#) scale set VMs for increased reliability and quicker deployment times. You cannot overprovision individual VMs unless you write custom code to perform this action.
- You can specify an [upgrade policy](#) to make it easy to roll out upgrades across VMs in your scale set. With individual VMs, you must orchestrate updates yourself.

VM-specific features

Some features are currently only available in VMs:

- You can capture an image from an individual VM, but not from a VM in a scale set.
- You can migrate an individual VM from native disks to managed disks, but you cannot migrate VM instances in a scale set.
- You can assign IPv6 public IP addresses to individual VM virtual network interface cards (NICs), but cannot do so for VM instances in a scale set. You can assign IPv6 public IP addresses to load balancers in front of either individual VMs or scale set VMs.

Storage

Scale sets with Azure Managed Disks

Scale sets can be created with [Azure Managed Disks](#) instead of traditional Azure storage accounts. Managed Disks provide the following benefits:

- You do not have to pre-create a set of Azure storage accounts for the scale set VMs.
- You can define [attached data disks](#) for the VMs in your scale set.
- Scale sets can be configured to [support up to 1,000 VMs in a set](#).

If you have an existing template, you can also [update the template to use Managed Disks](#).

User-managed Storage

A scale set that is not defined with Azure Managed Disks relies on user-created storage accounts to store the OS disks of the VMs in the set. A ratio of 20 VMs per storage account or less is recommended to achieve maximum IO and also take advantage of *overprovisioning* (see below). It is also recommended that you spread the beginning characters of the storage account names across the alphabet. Doing so helps spread load across different internal

systems.

Overprovisioning

Scale sets currently default to "overprovisioning" VMs. With overprovisioning turned on, the scale set actually spins up more VMs than you asked for, then deletes the extra VMs once the requested number of VMs are successfully provisioned. Overprovisioning improves provisioning success rates and reduces deployment time. You are not billed for the extra VMs, and they do not count toward your quota limits.

While overprovisioning does improve provisioning success rates, it can cause confusing behavior for an application that is not designed to handle extra VMs appearing and then disappearing. To turn overprovisioning off, ensure you have the following string in your template: `"overprovision": "false"`. More details can be found in the [Scale Set REST API documentation](#).

If your scale set uses user-managed storage, and you turn off overprovisioning, you can have more than 20 VMs per storage account, but it is not recommended to go above 40 for IO performance reasons.

Limits

A scale set built on a Marketplace image (also known as a platform image) and configured to use Azure Managed Disks supports a capacity of up to 1,000 VMs. If you configure your scale set to support more than 100 VMs, not all scenarios work the same (for example load balancing). For more information, see [Working with large virtual machine scale sets](#).

A scale set configured with user-managed storage accounts is currently limited to 100 VMs (and 5 storage accounts are recommended for this scale).

A scale set built on a custom image (one built by you) can have a capacity of up to 600 VMs when configured with Azure Managed disks. If the scale set is configured with user-managed storage accounts, it must create all OS disk VHDs within one storage account. As a result, the maximum recommended number of VMs in a scale set built on a custom image and user-managed storage is 20. If you turn off overprovisioning, you can go up to 40.

For more VMs than these limits allow, you need to deploy multiple scale sets as shown in [this template](#).

Understand instance IDs for Azure VM scale set VMs

1/19/2020 • 2 minutes to read • [Edit Online](#)

This article describes instance IDs for scale sets and the various ways they surface.

Scale set instance IDs

Each VM in a scale set gets an instance ID that uniquely identifies it. This instance ID is used in the scale set APIs to do operations on a specific VM in the scale set. For instance, you can specify a specific instance ID to reimagine when using the reimagine API:

REST API:

```
POST  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachineScaleSets/{vmScaleSetName}/reimage?  
api-version={apiVersion}
```

(for more information, see the [REST API documentation](#))

Powershell: `Set-AzVmssVM -ResourceGroupName {resourceGroupName} -VMSScaleSetName {vmScaleSetName} -InstanceId {instanceId} -Reimage` (for more information, see the [Powershell documentation](#))

CLI: `az vmss reimagine -g {resourceGroupName} -n {vmScaleSetName} --instance-id {instanceId}` (for more information, see the [CLI documentation](#)).

You can get the list of instance IDs by listing all instances in a scale set:

REST API:

```
GET  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachineScaleSets/{vmScaleSetName}/virtualMachines?  
api-version={apiVersion}
```

(for more information, see the [REST API documentation](#))

Powershell: `Get-AzVmssVM -ResourceGroupName {resourceGroupName} -VMSScaleSetName {vmScaleSetName}` (for more information, see the [Powershell documentation](#))

CLI: `az vmss list-instances -g {resourceGroupName} -n {vmScaleSetName}` (for more information, see the [CLI documentation](#)).

You can also use [resources.azure.com](#) or the [Azure SDKs](#) to list the VMs in a scale set.

The exact presentation of the output depends on the options you provide to the command, but here is some sample output from the CLI:

```
$ az vmss show -g {resourceGroupName} -n {vmScaleSetName}  
[  
 {  
 "instanceId": "85",  
 "latestModelApplied": true,  
 "location": "westus",  
 "name": "nsgvms_85",  
 .  
 .  
 .
```

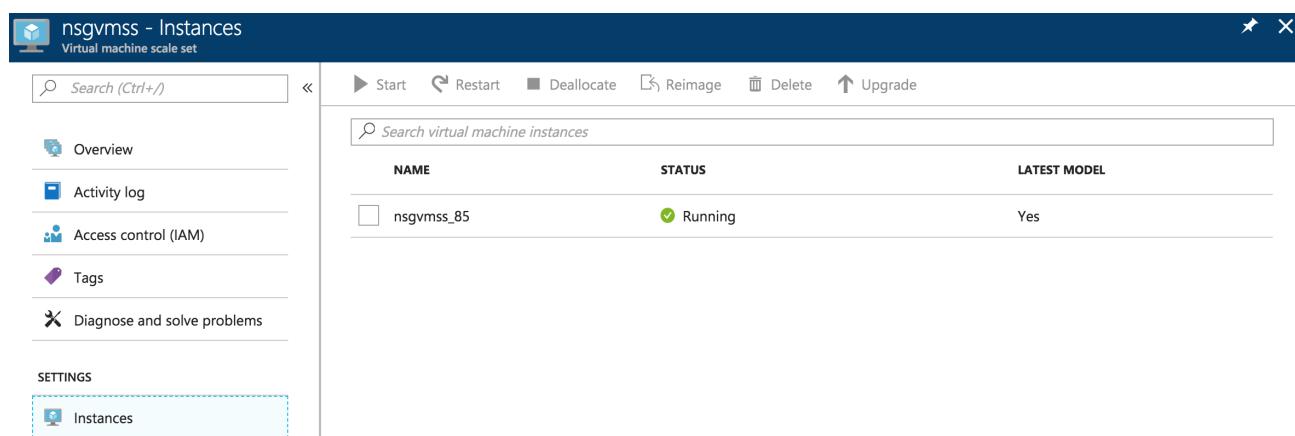
As you can see, the "instancetype" property is just a decimal number. The instance IDs may be reused for new instances once old instances are deleted.

NOTE

There is **no guarantee** on the way instance IDs are assigned to the VMs in the scale set. They might seem sequentially increasing at times, but this is not always the case. Do not take a dependency on the specific way in which instance IDs are assigned to the VMs.

Scale set VM names

In the sample output above, there is also a "name" for the VM. This name takes the form "{scale-set-name}_{instance-id}". This name is the one that you see in the Azure portal when you list instances in a scale set:



NAME	STATUS	LATEST MODEL
nsgvms_85	Running	Yes

The {instance-id} part of the name is the same decimal number as the "instancetype" property discussed previously.

Instance Metadata VM name

If you query the [instance metadata](#) from within a scale set VM, you see a "name" in the output:

```
{  
  "compute": {  
    "location": "westus",  
    "name": "nsgvms_85",  
    .  
    .  
    .
```

This name is the same as the name discussed previously.

Scale set VM computer name

Each VM in a scale set also gets a computer name assigned to it. This computer name is the hostname of the VM in the [Azure-provided DNS name resolution within the virtual network](#). This computer name is of the form "{computer-name-prefix}{base-36-instance-id}".

The {base-36-instance-id} is in [base 36](#) and is always six digits in length. If the base 36 representation of the number takes fewer than six digits, the {base-36-instance-id} is padded with zeros to make it six digits in length. For example, an instance with {computer-name-prefix} "nsgvms" and instance ID 85 will have computer name "nsgvms00002D".

NOTE

The computer name prefix is a property of the scale set model that you can set, so it can be different from the scale set name itself.

Learn about virtual machine scale set templates

1/19/2020 • 5 minutes to read • [Edit Online](#)

Azure Resource Manager templates are a great way to deploy groups of related resources. This tutorial series shows how to create a basic scale set template and how to modify this template to suit various scenarios. All examples come from this [GitHub repository](#).

This template is intended to be simple. For more complete examples of scale set templates, see the [Azure Quickstart Templates GitHub repository](#) and search for folders that contain the string `vmss`.

If you are already familiar with creating templates, you can skip to the "Next steps" section to see how to modify this template.

Define `$schema` and `contentVersion`

First, define `$schema` and `contentVersion` in the template. The `$schema` element defines the version of the template language and is used for Visual Studio syntax highlighting and similar validation features. The `contentVersion` element is not used by Azure. Instead, it helps you keep track of the template version.

```
{  
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json",  
  "contentVersion": "1.0.0.0",
```

Define parameters

Next, define two parameters, `adminUsername` and `adminPassword`. Parameters are values you specify at the time of deployment. The `adminUsername` parameter is simply a `string` type, but because `adminPassword` is a secret, give it type `securestring`. Later, these parameters are passed into the scale set configuration.

```
"parameters": {  
  "adminUsername": {  
    "type": "string"  
  },  
  "adminPassword": {  
    "type": "securestring"  
  }  
},
```

Define variables

Resource Manager templates also let you define variables to be used later in the template. The example doesn't use any variables, so the JSON object is empty.

```
"variables": {},
```

Define resources

Next is the resources section of the template. Here, you define what you actually want to deploy. Unlike `parameters` and `variables` (which are JSON objects), `resources` is a JSON list of JSON objects.

```
"resources": [
```

All resources require `type`, `name`, `apiVersion`, and `location` properties. This example's first resource has type [Microsoft.Network/virtualNetwork](#), name `myVnet`, and `apiVersion` `2018-11-01`. (To find the latest API version for a resource type, see the [Azure Resource Manager template reference](#).)

```
{  
    "type": "Microsoft.Network/virtualNetworks",  
    "name": "myVnet",  
    "apiVersion": "2018-11-01",
```

Specify location

To specify the location for the virtual network, use a [Resource Manager template function](#). This function must be enclosed in quotes and square brackets like this: `"[<template-function>]"`. In this case, use the `resourceGroup` function. It takes in no arguments and returns a JSON object with metadata about the resource group this deployment is being deployed to. The resource group is set by the user at the time of deployment. This value is then indexed into this JSON object with `.location` to get the location from the JSON object.

```
"location": "[resourceGroup().location]",
```

Specify virtual network properties

Each Resource Manager resource has its own `properties` section for configurations specific to the resource. In this case, specify that the virtual network should have one subnet using the private IP address range `10.0.0.0/16`. A scale set is always contained within one subnet. It cannot span subnets.

```
"properties": {  
    "addressSpace": {  
        "addressPrefixes": [  
            "10.0.0.0/16"  
        ]  
    },  
    "subnets": [  
        {  
            "name": "mySubnet",  
            "properties": {  
                "addressPrefix": "10.0.0.0/16"  
            }  
        }  
    ]  
},
```

Add dependsOn list

In addition to the required `type`, `name`, `apiVersion`, and `location` properties, each resource can have an optional `dependsOn` list of strings. This list specifies which other resources from this deployment must finish before deploying this resource.

In this case, there is only one element in the list, the virtual network from the previous example. You specify this dependency because the scale set needs the network to exist before creating any VMs. This way, the scale set can give these VMs private IP addresses from the IP address range previously specified in the network properties. The format of each string in the `dependsOn` list is `<type>/<name>`. Use the same `type` and `name` used previously in the virtual network resource definition.

```
{  
    "type": "Microsoft.Compute/virtualMachineScaleSets",  
    "name": "myScaleSet",  
    "apiVersion": "2019-03-01",  
    "location": "[resourceGroup().location]",  
    "dependsOn": [  
        "Microsoft.Network/virtualNetworks/myVnet"  
    ],
```

Specify scale set properties

Scale sets have many properties for customizing the VMs in the scale set. For a full list of these properties, see the [template reference](#). For this tutorial, only a few commonly used properties are set.

Supply VM size and capacity

The scale set needs to know what size of VM to create ("sku name") and how many such VMs to create ("sku capacity"). To see which VM sizes are available, see the [VM Sizes documentation](#).

```
"sku": {  
    "name": "Standard_A1",  
    "capacity": 2  
},
```

Choose type of updates

The scale set also needs to know how to handle updates on the scale set. Currently, there are three options, [Manual](#), [Rolling](#) and [Automatic](#). For more information on the differences between the two, see the documentation on [how to upgrade a scale set](#).

```
"properties": {  
    "upgradePolicy": {  
        "mode": "Manual"  
    },
```

Choose VM operating system

The scale set needs to know what operating system to put on the VMs. Here, create the VMs with a fully patched Ubuntu 16.04-LTS image.

```
"virtualMachineProfile": {  
    "storageProfile": {  
        "imageReference": {  
            "publisher": "Canonical",  
            "offer": "UbuntuServer",  
            "sku": "16.04-LTS",  
            "version": "latest"  
        }  
    },
```

Specify computerNamePrefix

The scale set deploys multiple VMs. Instead of specifying each VM name, specify [computerNamePrefix](#). The scale set appends an index to the prefix for each VM, so VM names have the form [`<computerNamePrefix>_<auto-generated-index>`](#).

In the following snippet, use the parameters from before to set the administrator username and password for all VMs in the scale set. This process uses the [parameters](#) template function. This function takes in a string that specifies which parameter to refer to and outputs the value for that parameter.

```
"osProfile": {  
    "computerNamePrefix": "vm",  
    "adminUsername": "[parameters('adminUsername')]",  
    "adminPassword": "[parameters('adminPassword')]"  
},
```

Specify VM network configuration

Finally, specify the network configuration for the VMs in the scale set. In this case, you only need to specify the ID of the subnet created earlier. This tells the scale set to put the network interfaces in this subnet.

You can get the ID of the virtual network containing the subnet by using the `resourceId` template function. This function takes in the type and name of a resource and returns the fully qualified identifier of that resource. This ID has the form:

```
/subscriptions/<subscriptionId>/resourceGroups/<resourceGroupName>/<resourceProviderNamespace>/<resourceType>/<resourceName>
```

However, the identifier of the virtual network is not enough. Provide the specific subnet that the scale set VMs should be in. To do this, concatenate `/subnets/mySubnet` to the ID of the virtual network. The result is the fully qualified ID of the subnet. Do this concatenation with the `concat` function, which takes in a series of strings and returns their concatenation.

```
"networkProfile": {  
    "networkInterfaceConfigurations": [  
        {  
            "name": "myNic",  
            "properties": {  
                "primary": "true",  
                "ipConfigurations": [  
                    {  
                        "name": "myIpConfig",  
                        "properties": {  
                            "subnet": {  
                                "id": "[concat(resourceId('Microsoft.Network/virtualNetworks', 'myVnet'),  
'/subnets/mySubnet')]"  
                            }  
                        }  
                    }  
                ]  
            }  
        }  
    ]  
}
```

Next steps

You can deploy the preceding template by following the [Azure Resource Manager documentation](#).

You can start this tutorial series from the [basic scale set template article](#).

You can see how to modify the [basic scale set template](#) to deploy the scale set into an existing virtual network.

You can see how to modify the [basic scale set template](#) to deploy the scale set with a custom image.

You can see how to modify the [basic scale set template](#) to deploy a Linux scale set with guest-based autoscale.

For more information about scale sets, refer to the [scale set overview page](#).

Add reference to an existing virtual network in an Azure scale set template

1/19/2020 • 2 minutes to read • [Edit Online](#)

This article shows how to modify the [basic scale set template](#) to deploy into an existing virtual network instead of creating a new one.

Change the template definition

In a [previous article](#) we had created a basic scale set template. We will now use that earlier template and modify it to create a template that deploys a scale set into an existing virtual network.

First, add a `subnetId` parameter. This string is passed into the scale set configuration, allowing the scale set to identify the pre-created subnet to deploy virtual machines into. This string must be of the form:

```
/subscriptions/<subscription-id>resourceGroups/<resource-group-name>/providers/Microsoft.Network/virtualNetworks/<virtual-network-name>/subnets/<subnet-name>
```

For instance, to deploy the scale set into an existing virtual network with name `myvnet`, subnet `mysubnet`, resource group `myrg`, and subscription `00000000-0000-0000-0000-000000000000`, the subnetId would be:

```
/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/myrg/providers/Microsoft.Network/virtualNetworks/myvnet/subnets/mysubnet
```

```
},
"adminPassword": {
    "type": "securestring"
},
+
"subnetId": {
    "type": "string"
},
}
```

Next, delete the virtual network resource from the `resources` array, as you use an existing virtual network and don't need to deploy a new one.

```

"variables": {},
"resources": [
-   {
-     "type": "Microsoft.Network/virtualNetworks",
-     "name": "myVnet",
-     "location": "[resourceGroup().location]",
-     "apiVersion": "2018-11-01",
-     "properties": {
-       "addressSpace": {
-         "addressPrefixes": [
-           "10.0.0.0/16"
-         ]
-       },
-       "subnets": [
-         {
-           "name": "mySubnet",
-           "properties": {
-             "addressPrefix": "10.0.0.0/16"
-           }
-         }
-       ]
-     }
-   },

```

The virtual network already exists before the template is deployed, so there is no need to specify a dependsOn clause from the scale set to the virtual network. Delete the following lines:

```

{
  "type": "Microsoft.Compute/virtualMachineScaleSets",
  "name": "myScaleSet",
  "location": "[resourceGroup().location]",
  "apiVersion": "2019-03-01",
-  "dependsOn": [
-    "Microsoft.Network/virtualNetworks/myVnet"
-  ],
  "sku": {
    "name": "Standard_A1",
    "capacity": 2
}

```

Finally, pass in the `subnetId` parameter set by the user (instead of using `resourceId` to get the ID of a vnet in the same deployment, which is what the basic viable scale set template does).

```

        "name": "myIpConfig",
        "properties": {
          "subnet": {
-            "id": "[concat(resourceId('Microsoft.Network/virtualNetworks', 'myVnet'),
+            "id": "[parameters('subnetId')]"
          }
        }
      }

```

Next steps

You can deploy the preceding template by following the [Azure Resource Manager documentation](#).

You can start this tutorial series from the [basic scale set template article](#).

You can see how to modify the [basic scale set template](#) to deploy the scale set into an existing virtual network.

You can see how to modify the [basic scale set template](#) to deploy the scale set with a custom image.

You can see how to modify the [basic scale set template](#) to deploy a Linux scale set with guest-based autoscale.

For more information about scale sets, refer to the [scale set overview page](#).

Add a custom image to an Azure scale set template

1/19/2020 • 2 minutes to read • [Edit Online](#)

This article shows how to modify the [basic scale set template](#) to deploy from custom image.

Change the template definition

In a [previous article](#) we had created a basic scale set template. We will now use that earlier template and modify it to create a template that deploys a scale set from a custom image.

Creating a managed disk image

If you already have a custom managed disk image (a resource of type `Microsoft.Compute/images`), then you can skip this section.

First, add a `sourceImageVhdUri` parameter, which is the URI to the generalized blob in Azure Storage that contains the custom image to deploy from.

```
    },
    "adminPassword": {
        "type": "securestring"
    },
    "sourceImageVhdUri": {
        "type": "string",
        "metadata": {
            "description": "The source of the generalized blob containing the custom image"
        }
    },
    "variables": {}
},
```

Next, add a resource of type `Microsoft.Compute/images`, which is the managed disk image based on the generalized blob located at URI `sourceImageVhdUri`. This image must be in the same region as the scale set that uses it. In the properties of the image, specify the OS type, the location of the blob (from the `sourceImageVhdUri` parameter), and the storage account type:

```

"resources": [
  {
    "type": "Microsoft.Compute/images",
    "apiVersion": "2019-03-01",
    "name": "myCustomImage",
    "location": "[resourceGroup().location]",
    "properties": {
      "storageProfile": {
        "osDisk": {
          "osType": "Linux",
          "osState": "Generalized",
          "blobUri": "[parameters('sourceImageVhdUri')]",
          "storageAccountType": "Standard_LRS"
        }
      }
    }
  },
  {
    "type": "Microsoft.Network/virtualNetworks",
    "name": "myVnet",
    "location": "[resourceGroup().location]",

```

In the scale set resource, add a `dependsOn` clause referring to the custom image to make sure the image gets created before the scale set tries to deploy from that image:

```

"location": "[resourceGroup().location]",
"apiVersion": "2019-03-01-preview",
"dependsOn": [
  "Microsoft.Network/virtualNetworks/myVnet",
  "Microsoft.Network/virtualNetworks/myVnet",
  "Microsoft.Compute/images/myCustomImage"
],
"sku": {
  "name": "Standard_A1",

```

Changing scale set properties to use the managed disk image

In the `imageReference` of the scale set `storageProfile`, instead of specifying the publisher, offer, sku, and version of a platform image, specify the `id` of the `Microsoft.Compute/images` resource:

```

"virtualMachineProfile": {
  "storageProfile": {
    "imageReference": {
      "id": "[resourceId('Microsoft.Compute/images', 'myCustomImage')]"
    }
  },
  "osProfile": {

```

In this example, use the `resourceId` function to get the resource ID of the image created in the same template. If you have created the managed disk image beforehand, you should provide the ID of that image instead. This ID must be of the form:

```
/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Compute/images/<image-name>
```

Next Steps

You can deploy the preceding template by following the [Azure Resource Manager documentation](#).

You can start this tutorial series from the [basic scale set template article](#).

You can see how to modify the [basic scale set template](#) to deploy the scale set into an existing virtual network.

You can see how to modify the [basic scale set template](#) to deploy the scale set with a custom image.

You can see how to modify the [basic scale set template](#) to deploy a Linux scale set with guest-based autoscale.

For more information about scale sets, refer to the [scale set overview page](#).

Autoscale using guest metrics in a Linux scale set template

1/19/2020 • 4 minutes to read • [Edit Online](#)

There are two broad types of metrics in Azure that are gathered from VMs and scale sets: Host metrics and Guest metrics. At a high level, if you would like to use standard CPU, disk, and network metrics, then host metrics are a good fit. If, however, you need a larger selection of metrics, then guest metrics should be looked into.

Host metrics do not require additional setup because they are collected by the host VM, whereas guest metrics require you to install the [Windows Azure Diagnostics extension](#) or the [Linux Azure Diagnostics extension](#) in the guest VM. One common reason to use guest metrics instead of host metrics is that guest metrics provide a larger selection of metrics than host metrics. One such example is memory-consumption metrics, which are only available via guest metrics. The supported host metrics are listed [here](#), and commonly used guest metrics are listed [here](#). This article shows how to modify the [basic viable scale set template](#) to use autoscale rules based on guest metrics for Linux scale sets.

Change the template definition

In a [previous article](#) we had created a basic scale set template. We will now use that earlier template and modify it to create a template that deploys a Linux scale set with guest metric based autoscale.

First, add parameters for `storageAccountName` and `storageAccountSasToken`. The diagnostics agent stores metric data in a [table](#) in this storage account. As of the Linux Diagnostics Agent version 3.0, using a storage access key is no longer supported. Instead, use a [SAS Token](#).

```
    },
    "adminPassword": {
        "type": "securestring"
    },
    "storageAccountName": {
        "type": "string"
    },
    "storageAccountSasToken": {
        "type": "securestring"
    }
},
```

Next, modify the scale set `extensionProfile` to include the diagnostics extension. In this configuration, specify the resource ID of the scale set to collect metrics from, as well as the storage account and SAS token to use to store the metrics. Specify how frequently the metrics are aggregated (in this case, every minute) and which metrics to track (in this case, percent used memory). For more detailed information on this configuration and metrics other than percent used memory, see [this documentation](#).

```

        }
    ]
},
"extensionProfile": {
    "extensions": [
    {
        "name": "LinuxDiagnosticExtension",
        "properties": {
            "publisher": "Microsoft.Azure.Diagnostics",
            "type": "LinuxDiagnostic",
            "typeHandlerVersion": "3.0",
            "settings": {
                "StorageAccount": "[parameters('storageAccountName')]",
                "ladCfg": {
                    "diagnosticMonitorConfiguration": {
                        "performanceCounters": {
                            "sinks": "WADMetricJsonBlob",
                            "performanceCounterConfiguration": [
                                {
                                    "unit": "percent",
                                    "type": "builtin",
                                    "class": "memory",
                                    "counter": "percentUsedMemory",
                                    "counterSpecifier": "/builtin/memory/percentUsedMemory",
                                    "condition": "IsAggregate=TRUE"
                                }
                            ]
                        },
                        "metrics": {
                            "metricAggregation": [
                                {
                                    "scheduledTransferPeriod": "PT1M"
                                }
                            ],
                            "resourceId": "[resourceId('Microsoft.Compute/virtualMachineScaleSets',
'myScaleSet')]"
                        }
                    }
                }
            }
        },
        "protectedSettings": {
            "storageAccountName": "[parameters('storageAccountName')]",
            "storageAccountSasToken": "[parameters('storageAccountSasToken')]",
            "sinksConfig": {
                "sink": [
                    {
                        "name": "WADMetricJsonBlob",
                        "type": "JsonBlob"
                    }
                ]
            }
        }
    ]
}
}

```

Finally, add an `autoscaleSettings` resource to configure autoscale based on these metrics. This resource has a `dependsOn` clause that references the scale set to ensure that the scale set exists before attempting to autoscale it. If you choose a different metric to autoscale on, you would use the `counterSpecifier` from the diagnostics extension configuration as the `metricName` in the autoscale configuration. For more information on autoscale configuration, see the [autoscale best practices](#) and the [Azure Monitor REST API reference documentation](#).

```
+     },
+     {
+         "type": "Microsoft.Insights/autoscaleSettings",
+         "apiVersion": "2015-04-01",
+         "name": "guestMetricsAutoscale",
+         "location": "[resourceGroup().location]",
+         "dependsOn": [
+             "Microsoft.Compute/virtualMachineScaleSets/myScaleSet"
+         ],
+         "properties": {
+             "name": "guestMetricsAutoscale",
+             "targetResourceUri": "[resourceId('Microsoft.Compute/virtualMachineScaleSets', 'myScaleSet')]",
+             "enabled": true,
+             "profiles": [
+                 {
+                     "name": "Profile1",
+                     "capacity": {
+                         "minimum": "1",
+                         "maximum": "10",
+                         "default": "3"
+                     },
+                     "rules": [
+                         {
+                             "metricTrigger": {
+                                 "metricName": "/builtin/memory/percentUsedMemory",
+                                 "metricNamespace": "",
+                                 "metricResourceUri": "[resourceId('Microsoft.Compute/virtualMachineScaleSets', 'myScaleSet')]",
+                                 "timeGrain": "PT1M",
+                                 "statistic": "Average",
+                                 "timeWindow": "PT5M",
+                                 "timeAggregation": "Average",
+                                 "operator": "GreaterThan",
+                                 "threshold": 60
+                             },
+                             "scaleAction": {
+                                 "direction": "Increase",
+                                 "type": "ChangeCount",
+                                 "value": "1",
+                                 "cooldown": "PT1M"
+                             }
+                         },
+                         {
+                             "metricTrigger": {
+                                 "metricName": "/builtin/memory/percentUsedMemory",
+                                 "metricNamespace": "",
+                                 "metricResourceUri": "[resourceId('Microsoft.Compute/virtualMachineScaleSets', 'myScaleSet')]",
+                                 "timeGrain": "PT1M",
+                                 "statistic": "Average",
+                                 "timeWindow": "PT5M",
+                                 "timeAggregation": "Average",
+                                 "operator": "LessThan",
+                                 "threshold": 30
+                             },
+                             "scaleAction": {
+                                 "direction": "Decrease",
+                                 "type": "ChangeCount",
+                                 "value": "1",
+                                 "cooldown": "PT1M"
+                             }
+                         }
+                     ]
+                 }
+             ]
+         }
+     }
]
```

```
}
```

Next steps

You can deploy the preceding template by following the [Azure Resource Manager documentation](#).

You can start this tutorial series from the [basic scale set template article](#).

You can see how to modify the [basic scale set template](#) to deploy the scale set into an existing virtual network.

You can see how to modify the [basic scale set template](#) to deploy the scale set with a custom image.

You can see how to modify the [basic scale set template](#) to deploy a Linux scale set with guest-based autoscale.

For more information about scale sets, refer to the [scale set overview page](#).

How to create a Virtual Machine Scale Set with Visual Studio

1/19/2020 • 3 minutes to read • [Edit Online](#)

This article shows you how to deploy an Azure Virtual Machine Scale Set using a Visual Studio Resource Group deployment.

Azure Virtual Machine Scale Sets is an Azure Compute resource to deploy and manage a collection of similar virtual machines with autoscale and load balancing. You can provision and deploy Virtual Machine Scale Sets using [Azure Resource Manager Templates](#). Azure Resource Manager templates can be deployed using Azure CLI, PowerShell, REST and also directly from Visual Studio. Visual Studio provides a set of example templates, which you can deploy as part of an Azure Resource Group deployment project.

Azure Resource Group deployments are a way to group and publish a set of related Azure resources in a single deployment operation. For more information, see [Creating and deploying Azure resource groups through Visual Studio](#).

Prerequisites

To get started deploying Virtual Machine Scale Sets in Visual Studio, you need the following prerequisites:

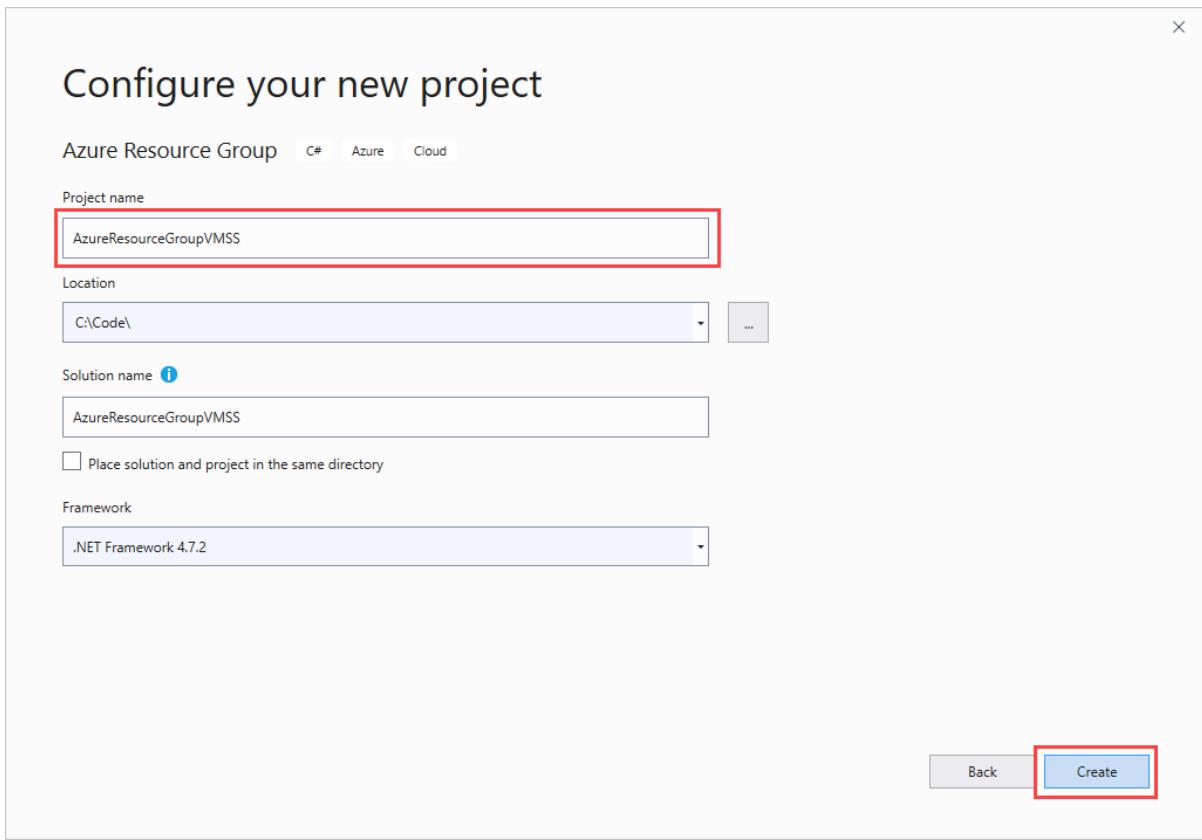
- Visual Studio 2013 or later
- Azure SDK 2.7, 2.8 or 2.9

NOTE

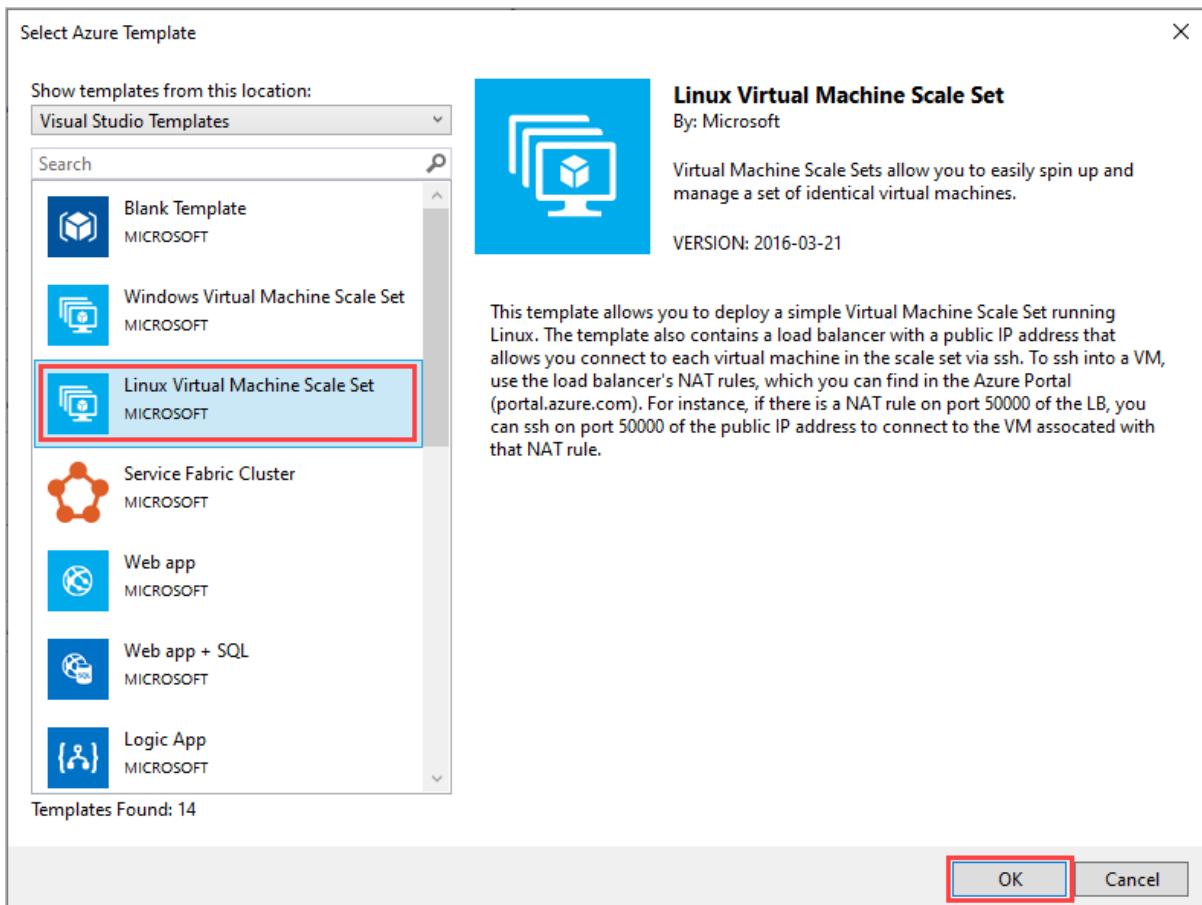
This article uses Visual Studio 2019 with [Azure SDK 2.8](#).

Create a Project

1. Open Visual Studio and select **Create a new project**.
2. In **Create a new project**, choose **Azure Resource Group** for C# and then select **Next**.
3. In **Configure your new project**, enter a name and select **Create**.



4. From the list of templates, choose either the **Windows Virtual Machine Scale Set** or **Linux Virtual Machine Scale Set** template. Select **OK**.



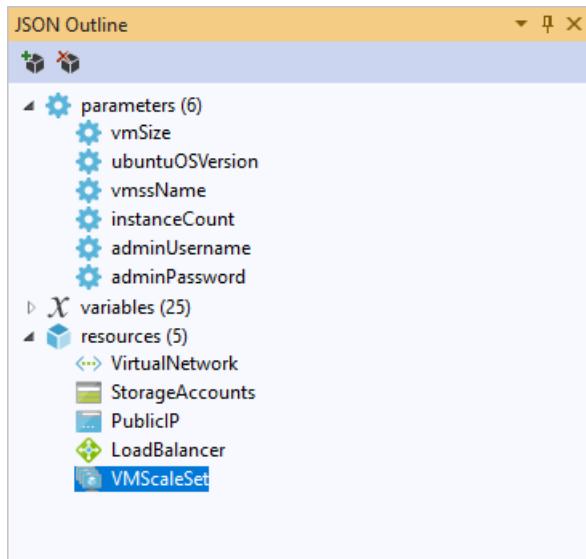
After you create your project, **Solution Explorer** contains a PowerShell deployment script, an Azure Resource Manager template, and a parameter file for the Virtual Machine Scale Set.

Customize your project

Now you can edit the template to customize it for your application's needs. You could add virtual machine extension properties or edit load-balancing rules. By default, the Virtual Machine Scale Set templates are configured to deploy the **AzureDiagnostics** extension, which makes it easy to add autoscale rules. The templates also deploy a load balancer with a public IP address, configured with inbound NAT rules.

The load balancer lets you connect to the virtual machine instances with SSH (Linux) or RDP (Windows). The front-end port range starts at 50000. For Linux, if you SSH to port 50000, load balancing routes you to port 22 of the first virtual machine in the Scale Set. Connecting to port 50001 is routed to port 22 of the second virtual machine, and so on.

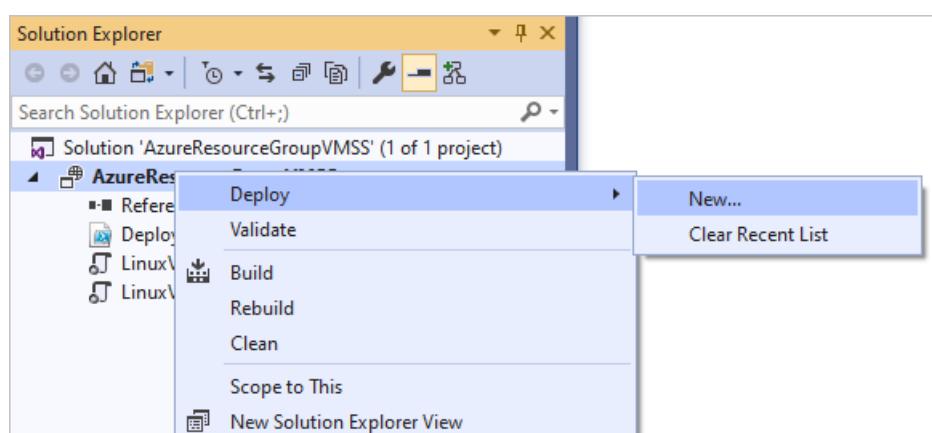
A good way to edit your templates with Visual Studio is to use the **JSON Outline**. You can organize the parameters, variables, and resources. With an understanding of the schema, Visual Studio can point out errors in your template before you deploy it.



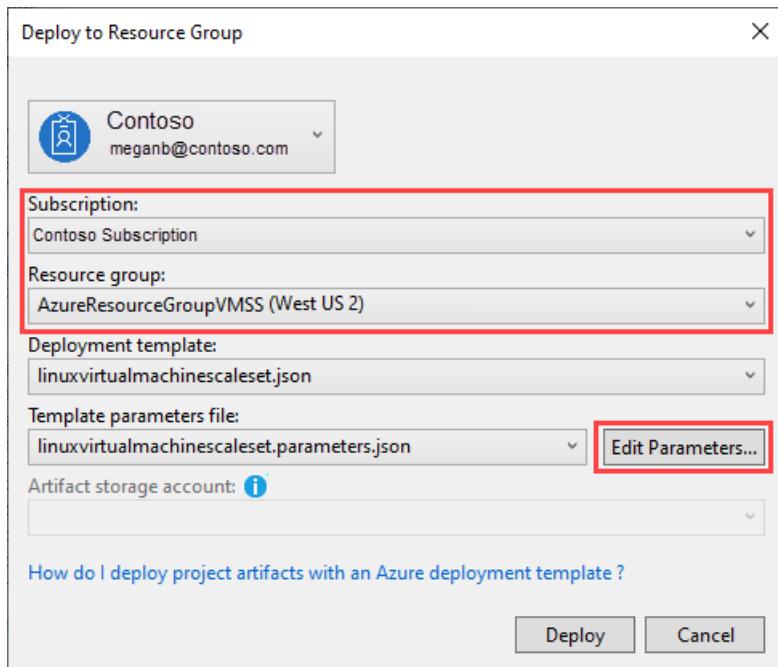
Deploy the project

Deploy the Azure Resource Manager template to create the Virtual Machine Scale Set resource:

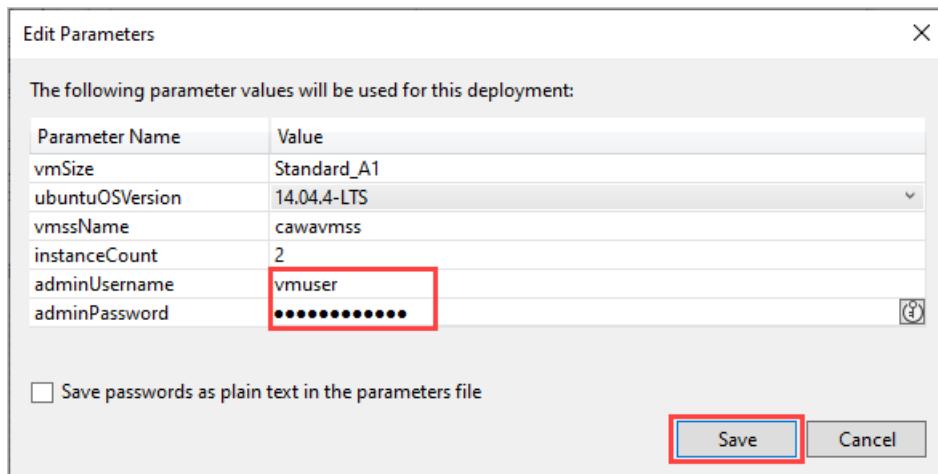
1. In **Solution Explorer**, right-click the project and choose **Deploy > New**.



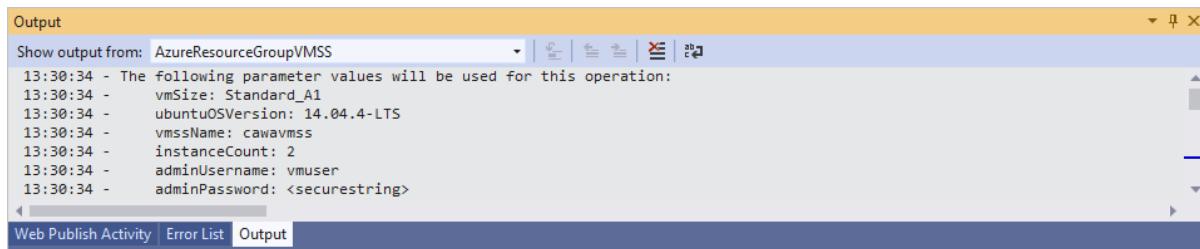
2. In **Deploy to Resource Group**, choose which subscription to use and select a resource group. You can create a resource group, if necessary.
3. Next, select **Edit Parameters** to enter parameters that are passed to your template.



4. Provide the username and password for the operating system. These values are required to create the deployment. If you don't have PowerShell Tools for Visual Studio installed, select **Save passwords** to avoid a hidden PowerShell command prompt, or use [Key Vault support](#). Select **Save** to continue.

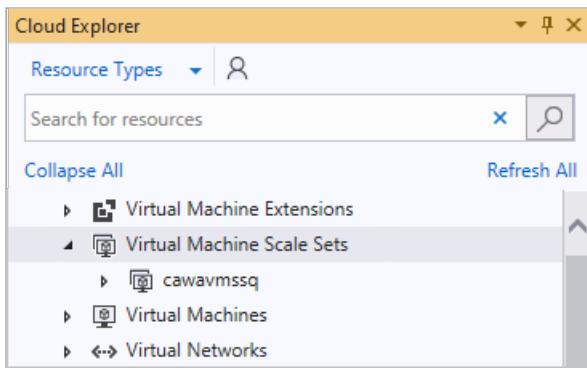


5. In **Deploy to Resource Group**, select **Deploy**. The action runs the **Deploy-AzureResourceGroup.ps1** script. The **Output** window shows the deployment progress.



Explore your Virtual Machine Scale Set

Select **View > Cloud Explorer** to view the new Virtual Machine Scale Set. Use **Refresh All**, if necessary.



Cloud Explorer lets you manage Azure resources in Visual Studio while developing applications. You can also view your Virtual Machine Scale Set in the [Azure portal](#) and [Azure Resource Explorer](#).

The portal provides the best way to manage your Azure infrastructure with a web browser. Azure Resource Explorer provides an easy way to explore and debug Azure resources. Azure Resource Explorer offers the instance view and also shows PowerShell commands for the resources you're looking at.

Next steps

Once you've successfully deployed Virtual Machine Scale Sets through Visual Studio, you can further customize your project to suit your application requirements. For example, configure autoscale by adding an **Insights** resource. You could add infrastructure to your template, such as standalone virtual machines, or deploy applications using the custom script extension. Good example templates can be found in the [Azure Quickstart Templates](#) GitHub repository. Search for `vmss`.

Create a virtual machine scale set that uses Availability Zones

1/19/2020 • 9 minutes to read • [Edit Online](#)

To protect your virtual machine scale sets from datacenter-level failures, you can create a scale set across Availability Zones. Azure regions that support Availability Zones have a minimum of three separate zones, each with their own independent power source, network, and cooling. For more information, see [Overview of Availability Zones](#).

Availability considerations

When you deploy a scale set into one or more zones as of API version `2017-12-01`, you have the option to deploy with "max spreading" or "static 5 fault domain spreading". With max spreading, the scale set spreads your VMs across as many fault domains as possible within each zone. This spreading could be across greater or fewer than five fault domains per zone. With "static 5 fault domain spreading", the scale set spreads your VMs across exactly five fault domains per zone. If the scale set cannot find five distinct fault domains per zone to satisfy the allocation request, the request fails.

We recommend deploying with max spreading for most workloads, as this approach provides the best spreading in most cases. If you need replicas to be spread across distinct hardware isolation units, we recommend spreading across Availability Zones and utilize max spreading within each zone.

With max spreading, you only see one fault domain in the scale set VM instance view and in the instance metadata regardless of how many fault domains the VMs are spread across. The spreading within each zone is implicit.

To use max spreading, set `platformFaultDomainCount` to 1. To use static five fault domain spreading, set `platformFaultDomainCount` to 5. In API version `2017-12-01`, `platformFaultDomainCount` defaults to 1 for single-zone and cross-zone scale sets. Currently, only static five fault domain spreading is supported for regional (non-zonal) scale sets.

Placement groups

When you deploy a scale set, you also have the option to deploy with a single [placement group](#) per Availability Zone, or with multiple per zone. For regional (non-zonal) scale sets, the choice is to have a single placement group in the region or to have multiple in the region. For most workloads, we recommend multiple placement groups, which allows for greater scale. In API version `2017-12-01`, scale sets default to multiple placement groups for single-zone and cross-zone scale sets, but they default to single placement group for regional (non-zonal) scale sets.

NOTE

If you use max spreading, you must use multiple placement groups.

Zone balancing

Finally, for scale sets deployed across multiple zones, you also have the option of choosing "best effort zone balance" or "strict zone balance". A scale set is considered "balanced" if each zone has the same number of VMs or +/- 1 VM in all other zones for the scale set. For example:

- A scale set with 2 VMs in zone 1, 3 VMs in zone 2, and 3 VMs in zone 3 is considered balanced. There is only one zone with a different VM count and it is only 1 less than the other zones.

- A scale set with 1 VM in zone 1, 3 VMs in zone 2, and 3 VMs in zone 3 is considered unbalanced. Zone 1 has 2 fewer VMs than zones 2 and 3.

It's possible that VMs in the scale set are successfully created, but extensions on those VMs fail to deploy. These VMs with extension failures are still counted when determining if a scale set is balanced. For instance, a scale set with 3 VMs in zone 1, 3 VMs in zone 2, and 3 VMs in zone 3 is considered balanced even if all extensions failed in zone 1 and all extensions succeeded in zones 2 and 3.

With best-effort zone balance, the scale set attempts to scale in and out while maintaining balance. However, if for some reason this is not possible (for example, if one zone goes down, the scale set cannot create a new VM in that zone), the scale set allows temporary imbalance to successfully scale in or out. On subsequent scale-out attempts, the scale set adds VMs to zones that need more VMs for the scale set to be balanced. Similarly, on subsequent scale in attempts, the scale set removes VMs from zones that need fewer VMs for the scale set to be balanced. With "strict zone balance", the scale set fails any attempts to scale in or out if doing so would cause unbalance.

To use best-effort zone balance, set `zoneBalance` to `false`. This setting is the default in API version `2017-12-01`. To use strict zone balance, set `zoneBalance` to `true`.

Single-zone and zone-redundant scale sets

When you deploy a virtual machine scale set, you can choose to use a single Availability Zone in a region, or multiple zones.

When you create a scale set in a single zone, you control which zone all those VM instances run in, and the scale set is managed and autoscales only within that zone. A zone-redundant scale set lets you create a single scale set that spans multiple zones. As VM instances are created, by default they are evenly balanced across zones. Should an interruption occur in one of the zones, a scale set does not automatically scale out to increase capacity. A best practice would be to configure autoscale rules based on CPU or memory usage. The autoscale rules would allow the scale set to respond to a loss of the VM instances in that one zone by scaling out new instances in the remaining operational zones.

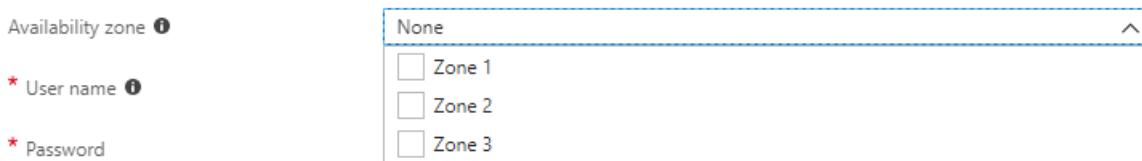
To use Availability Zones, your scale set must be created in a [supported Azure region](#). You can create a scale set that uses Availability Zones with one of the following methods:

- [Azure portal](#)
- [Azure CLI](#)
- [Azure PowerShell](#)
- [Azure Resource Manager templates](#)

Use the Azure portal

The process to create a scale set that uses an Availability Zone is the same as detailed in the [getting started article](#).

When you select a supported Azure region, you can create a scale set in one or more available zones, as shown in the following example:



The scale set and supporting resources, such as the Azure load balancer and public IP address, are created in the single zone that you specify.

Use the Azure CLI

The process to create a scale set that uses an Availability Zone is the same as detailed in the [getting started article](#). To use Availability Zones, you must create your scale set in a supported Azure region.

Add the `--zones` parameter to the `az vmss create` command and specify which zone to use (such as zone 1, 2, or 3). The following example creates a single-zone scale set named `myScaleSet` in zone 1:

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--admin-username azureuser \
--generate-ssh-keys \
--zones 1
```

For a complete example of a single-zone scale set and network resources, see [this sample CLI script](#)

Zone-redundant scale set

To create a zone-redundant scale set, you use a *Standard* SKU public IP address and load balancer. For enhanced redundancy, the *Standard* SKU creates zone-redundant network resources. For more information, see [Azure Load Balancer Standard overview](#) and [Standard Load Balancer and Availability Zones](#).

To create a zone-redundant scale set, specify multiple zones with the `--zones` parameter. The following example creates a zone-redundant scale set named `myScaleSet` across zones 1,2,3:

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--admin-username azureuser \
--generate-ssh-keys \
--zones 1 2 3
```

It takes a few minutes to create and configure all the scale set resources and VMs in the zone(s) that you specify. For a complete example of a zone-redundant scale set and network resources, see [this sample CLI script](#)

Use Azure PowerShell

To use Availability Zones, you must create your scale set in a supported Azure region. Add the `-Zone` parameter to the `New-AzVmssConfig` command and specify which zone to use (such as zone 1, 2, or 3).

The following example creates a single-zone scale set named `myScaleSet` in *East US 2* zone 1. The Azure network resources for virtual network, public IP address, and load balancer are automatically created. When prompted, provide your own desired administrative credentials for the VM instances in the scale set:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "EastUS2" ` 
-VMSScaleSetName "myScaleSet" ` 
-VirtualNetworkName "myVnet" ` 
-SubnetName "mySubnet" ` 
-PublicIpAddressName "myPublicIPAddress" ` 
-LoadBalancerName "myLoadBalancer" ` 
-UpgradePolicy "Automatic" ` 
-Zone "1"
```

Zone-redundant scale set

To create a zone-redundant scale set, specify multiple zones with the `-Zone` parameter. The following example creates a zone-redundant scale set named *myScaleSet* across *East US 2* zones 1, 2, 3. The zone-redundant Azure network resources for virtual network, public IP address, and load balancer are automatically created. When prompted, provide your own desired administrative credentials for the VM instances in the scale set:

```
New-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "EastUS2" ` 
-VMSScaleSetName "myScaleSet" ` 
-VirtualNetworkName "myVnet" ` 
-SubnetName "mySubnet" ` 
-PublicIpAddressName "myPublicIPAddress" ` 
-LoadBalancerName "myLoadBalancer" ` 
-UpgradePolicy "Automatic" ` 
-Zone "1", "2", "3"
```

Use Azure Resource Manager templates

The process to create a scale set that uses an Availability Zone is the same as detailed in the getting started article for [Linux](#) or [Windows](#). To use Availability Zones, you must create your scale set in a supported Azure region. Add the `zones` property to the *Microsoft.Compute/virtualMachineScaleSets* resource type in your template and specify which zone to use (such as zone 1, 2, or 3).

The following example creates a Linux single-zone scale set named *myScaleSet* in *East US 2* zone 1:

```
{
  "type": "Microsoft.Compute/virtualMachineScaleSets",
  "name": "myScaleSet",
  "location": "East US 2",
  "apiVersion": "2017-12-01",
  "zones": ["1"],
  "sku": {
    "name": "Standard_A1",
    "capacity": "2"
  },
  "properties": {
    "upgradePolicy": {
      "mode": "Automatic"
    },
    "virtualMachineProfile": {
      "storageProfile": {
        "osDisk": {
          "caching": "ReadWrite",
          "createOption": "FromImage"
        },
        "imageReference": {
          "publisher": "Canonical",
          "offer": "UbuntuServer",
          "sku": "16.04-LTS",
          "version": "latest"
        }
      },
      "osProfile": {
        "computerNamePrefix": "myvmss",
        "adminUsername": "azureuser",
        "adminPassword": "P@ssw0rd!"
      }
    }
  }
}
```

For a complete example of a single-zone scale set and network resources, see [this sample Resource Manager template](#)

Zone-redundant scale set

To create a zone-redundant scale set, specify multiple values in the `zones` property for the `Microsoft.Compute/virtualMachineScaleSets` resource type. The following example creates a zone-redundant scale set named `myScaleSet` across *East US 2* zones 1,2,3:

```
{  
  "type": "Microsoft.Compute/virtualMachineScaleSets",  
  "name": "myScaleSet",  
  "location": "East US 2",  
  "apiVersion": "2017-12-01",  
  "zones": [  
    "1",  
    "2",  
    "3"  
  ]  
}
```

If you create a public IP address or a load balancer, specify the `"sku": { "name": "Standard" }` property to create zone-redundant network resources. You also need to create a Network Security Group and rules to permit any traffic. For more information, see [Azure Load Balancer Standard overview](#) and [Standard Load Balancer and Availability Zones](#).

For a complete example of a zone-redundant scale set and network resources, see [this sample Resource Manager template](#)

Next steps

Now that you have created a scale set in an Availability Zone, you can learn how to [Deploy applications on virtual machine scale sets](#) or [Use autoscale with virtual machine scale sets](#).

Overview of autoscale with Azure virtual machine scale sets

1/19/2020 • 6 minutes to read • [Edit Online](#)

An Azure virtual machine scale set can automatically increase or decrease the number of VM instances that run your application. This automated and elastic behavior reduces the management overhead to monitor and optimize the performance of your application. You create rules that define the acceptable performance for a positive customer experience. When those defined thresholds are met, autoscale rules take action to adjust the capacity of your scale set. You can also schedule events to automatically increase or decrease the capacity of your scale set at fixed times. This article provides an overview of which performance metrics are available and what actions autoscale can perform.

Benefits of autoscale

If your application demand increases, the load on the VM instances in your scale set increases. If this increased load is consistent, rather than just a brief demand, you can configure autoscale rules to increase the number of VM instances in the scale set.

When these VM instances are created and your applications are deployed, the scale set starts to distribute traffic to them through the load balancer. You control what metrics to monitor, such as CPU or memory, how long the application load must meet a given threshold, and how many VM instances to add to the scale set.

On an evening or weekend, your application demand may decrease. If this decreased load is consistent over a period of time, you can configure autoscale rules to decrease the number of VM instances in the scale set. This scale-in action reduces the cost to run your scale set as you only run the number of instances required to meet the current demand.

Use host-based metrics

You can create autoscale rules that built-in host metrics available from your VM instances. Host metrics give you visibility into the performance of the VM instances in a scale set without the need to install or configure additional agents and data collections. Autoscale rules that use these metrics can scale out or in the number of VM instances in response to CPU usage, memory demand, or disk access.

Autoscale rules that use host-based metrics can be created with one of the following tools:

- [Azure portal](#)
- [Azure PowerShell](#)
- [Azure CLI](#)
- [Azure template](#)

To create autoscale rules that use more detailed performance metrics, you can [install and configure the Azure diagnostics extension](#) on VM instances, or [configure your application use App Insights](#).

Autoscale rules that use host-based metrics, in-guest VM metrics with the Azure diagnostic extension, and App Insights can use the following configuration settings.

Metric sources

Autoscale rules can use metrics from one of the following sources:

METRIC SOURCE	USE CASE
Current scale set	For host-based metrics that do not require additional agents to be installed or configured.
Storage account	The Azure diagnostic extension writes performance metrics to Azure storage that is then consumed to trigger autoscale rules.
Service Bus Queue	Your application or other components can transmit messages on an Azure Service Bus queue to trigger rules.
Application Insights	An instrumentation package installed in your application that streams metrics directly from the app.

Autoscale rule criteria

The following host-based metrics are available for use when you create autoscale rules. If you use the Azure diagnostic extension or App Insights, you define which metrics to monitor and use with autoscale rules.

METRIC NAME
Percentage CPU
Network In
Network Out
Disk Read Bytes
Disk Write Bytes
Disk Read Operations/Sec
Disk Write Operations/Sec
CPU Credits Remaining
CPU Credits Consumed

When you create autoscale rules to monitor a given metric, the rules look at one of the following metrics aggregation actions:

AGGREGATION TYPE
Average
Minimum
Maximum
Total
Last

AGGREGATION TYPE
Count

The autoscale rules are then triggered when the metrics are compared against your defined threshold with one of the following operators:

OPERATOR
Greater than
Greater than or equal to
Less than
Less than or equal to
Equal to
Not equal to

Actions when rules trigger

When an autoscale rule triggers, your scale set can automatically scale in one of the following ways:

SCALE OPERATION	USE CASE
Increase count by	A fixed number of VM instances to create. Useful in scale sets with a smaller number of VMs.
Increase percent by	A percentage-based increase of VM instances. Good for larger scale sets where a fixed increase may not noticeably improve performance.
Increase count to	Create as many VM instances are required to reach a desired maximum amount.
Decrease count by	A fixed number of VM instances to remove. Useful in scale sets with a smaller number of VMs.
Decrease percent by	A percentage-based decrease of VM instances. Good for larger scale sets where a fixed increase may not noticeably reduce resource consumption and costs.
Decrease count to	Remove as many VM instances are required to reach a desired minimum amount.

In-guest VM metrics with the Azure diagnostics extension

The Azure diagnostics extension is an agent that runs inside a VM instance. The agent monitors and saves performance metrics to Azure storage. These performance metrics contain more detailed information about the status of the VM, such as *AverageReadTime* for disks or *PercentIdleTime* for CPU. You can create autoscale rules based on a more detailed awareness of the VM performance, not just the percentage of CPU usage or memory consumption.

To use the Azure diagnostics extension, you must create Azure storage accounts for your VM instances, install the Azure diagnostics agent, then configure the VMs to stream specific performance counters to the storage account.

For more information, see the articles for how to enable the Azure diagnostics extension on a [Linux VM](#) or [Windows VM](#).

Application-level metrics with App Insights

To gain more visibility into the performance of your applications, you can use Application Insights. You install a small instrumentation package in your application that monitors the app and sends telemetry to Azure. You can monitor metrics such as the response times of your application, the page load performance, and the session counts. These application metrics can be used to create autoscale rules at a granular and embedded level as you trigger rules based on actionable insights that may impact the customer experience.

For more information about App Insights, see [What is Application Insights](#).

Scheduled autoscale

You can also create autoscale rules based on schedules. These schedule-based rules allow you to automatically scale the number of VM instances at fixed times. With performance-based rules, there may be a performance impact on the application before the autoscale rules trigger and the new VM instances are provisioned. If you can anticipate such demand, the additional VM instances are provisioned and ready for the additional customer use and application demand.

The following examples are scenarios that may benefit the use of schedule-based autoscale rules:

- Automatically scale out the number of VM instances at the start of the work day when customer demand increases. At the end of the work day, automatically scale in the number of VM instances to minimize resource costs overnight when application use is low.
- If a department uses an application heavily at certain parts of the month or fiscal cycle, automatically scale the number of VM instances to accommodate their additional demands.
- When there is a marketing event, promotion, or holiday sale, you can automatically scale the number of VM instances ahead of anticipated customer demand.

Next steps

You can create autoscale rules that use host-based metrics with one of the following tools:

- [Azure PowerShell](#)
- [Azure CLI](#)
- [Azure template](#)

This overview detailed how to use autoscale rules to scale horizontally and increase or decrease the *number* of VM instances in your scale set. You can also scale vertically to increase or decrease the VM instance size. For more information, see [Vertical autoscale with Virtual Machine Scale sets](#).

For information on how to manage your VM instances, see [Manage virtual machine scale sets with Azure PowerShell](#).

To learn how to generate alerts when your autoscale rules trigger, see [Use autoscale actions to send email and webhook alert notifications in Azure Monitor](#). You can also [Use audit logs to send email and webhook alert notifications in Azure Monitor](#).

Automatically scale a virtual machine scale set in the Azure portal

1/19/2020 • 6 minutes to read • [Edit Online](#)

When you create a scale set, you define the number of VM instances that you wish to run. As your application demand changes, you can automatically increase or decrease the number of VM instances. The ability to autoscale lets you keep up with customer demand or respond to application performance changes throughout the lifecycle of your app.

This article shows you how to create autoscale rules in the Azure portal that monitor the performance of the VM instances in your scale set. These autoscale rules increase or decrease the number of VM instances in response to these performance metrics. You can also complete these steps with [Azure PowerShell](#) or the [Azure CLI](#).

Prerequisites

To create autoscale rules, you need an existing virtual machine scale set. You can create a scale set with the [Azure portal](#), [Azure PowerShell](#), or [Azure CLI](#).

Create a rule to automatically scale out

If your application demand increases, the load on the VM instances in your scale set increases. If this increased load is consistent, rather than just a brief demand, you can configure autoscale rules to increase the number of VM instances in the scale set. When these VM instances are created and your applications are deployed, the scale set starts to distribute traffic to them through the load balancer. You control what metrics to monitor, such as CPU or disk, how long the application load must meet a given threshold, and how many VM instances to add to the scale set.

1. Open the Azure portal and select **Resource groups** from the menu on the left-hand side of the dashboard.
2. Select the resource group that contains your scale set, then choose your scale set from the list of resources.
3. Choose **Scaling** from the menu on the left-hand side of the scale set window. Select the button to **Enable autoscale**:

4. Enter a name for your settings, such as *autoscale*, then select the option to **Add a rule**.
5. Let's create a rule that increases the number of VM instances in a scale set when the average CPU load is greater than 70% over a 10-minute period. When the rule triggers, the number of VM instances is increased by 20%. In scale sets with a small number of VM instances, you could set the **Operation** to *Increase count by* and then specify 1 or 2 for the *Instance count*. In scale sets with a large number of VM instances, an increase of 10% or 20% VM instances may be more appropriate.

Specify the following settings for your rule:

PARAMETER	EXPLANATION	VALUE
<i>Time Aggregation</i>	Defines how the collected metrics should be aggregated for analysis.	Average
<i>Metric Name</i>	The performance metric to monitor and apply scale set actions on.	Percentage CPU
<i>Time grain statistic</i>	Defines how the collected metrics in each time grain should be aggregated for analysis.	Average
<i>Operator</i>	Operator used to compare the metric data against the threshold.	Greater than
<i>Threshold</i>	The percentage that causes the autoscale rule to trigger an action.	70

PARAMETER	EXPLANATION	VALUE
-----------	-------------	-------

<i>Duration</i>	The amount of time monitored before the metric and threshold values are compared.	10 minutes
<i>Operation</i>	Defines if the scale set should scale up or down when the rule applies and by what increment	Increase percent by
<i>Instance count</i>	The percentage of VM instances should be changed when the rule triggers.	20
<i>Cool down (minutes)</i>	The amount of time to wait before the rule is applied again so that the autoscale actions have time to take effect.	5 minutes

The following examples show a rule created in the Azure portal that matches these settings:

Scale rule

Metric source
Current resource (myScaleSet)

Resource type
Virtual machine scale sets

Resource
myScaleSet

Criteria

* Time aggregation ⓘ
Average

* Metric name
Percentage CPU
1 minute time grain

* Time grain statistic ⓘ
Average

* Operator
Greater than

* Threshold
70

* Duration (in minutes) ⓘ
10

Action

* Operation
Increase percent by

* Instance count
20 ✓

* Cool down (minutes) ⓘ
5

Add

The dialog is titled 'Scale rule'. Under 'Metric source', 'Current resource (myScaleSet)' is selected. 'Resource type' is set to 'Virtual machine scale sets' and 'Resource' is 'myScaleSet'. In the 'Criteria' section, 'Time aggregation' is 'Average', 'Metric name' is 'Percentage CPU' with a '1 minute time grain', 'Time grain statistic' is 'Average', 'Operator' is 'Greater than', 'Threshold' is '70', and 'Duration (in minutes)' is '10'. In the 'Action' section, 'Operation' is 'Increase percent by', 'Instance count' is '20' (with a checkmark), and 'Cool down (minutes)' is '5'. A blue 'Add' button is at the bottom.

6. To create the rule, select **Add**

Create a rule to automatically scale in

On an evening or weekend, your application demand may decrease. If this decreased load is consistent over a period of time, you can configure autoscale rules to decrease the number of VM instances in the scale set. This scale-in action reduces the cost to run your scale set as you only run the number of instances required to meet the current demand.

1. Choose to **Add a rule** again.
2. Create a rule that decreases the number of VM instances in a scale set when the average CPU load then drops below 30% over a 10-minute period. When the rule triggers, the number of VM instances is decreased by 20%.

Use the same approach as with the previous rule. Adjust the following settings for your rule:

PARAMETER	EXPLANATION	VALUE
<i>Operator</i>	Operator used to compare the metric data against the threshold.	Less than
<i>Threshold</i>	The percentage that causes the autoscale rule to trigger an action.	30
<i>Operation</i>	Defines if the scale set should scale up or down when the rule applies and by what increment	Decrease percent by
<i>Instance count</i>	The percentage of VM instances should be changed when the rule triggers.	20

3. To create the rule, select **Add**

Define autoscale instance limits

Your autoscale profile must define a minimum, maximum, and default number of VM instances. When your autoscale rules are applied, these instance limits make sure that you do not scale out beyond the maximum number of instances, or scale in beyond the minimum of instances.

1. Set the following instance limits:

MINIMUM	MAXIMUM	DEFAULT
2	10	2

2. To apply your autoscale rules and instance limits, select **Save**.

Monitor number of instances in a scale set

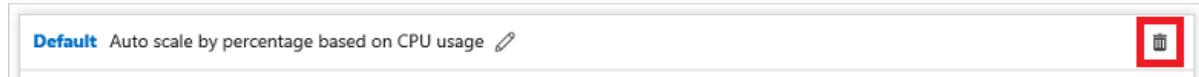
To see the number and status of VM instances, select **Instances** from the menu on the left-hand side of the scale set window. The status indicates if the VM instance is *Creating* as the scale set automatically scales out, or is *Deleting* as the scale automatically scales in.

NAME	STATUS	LATEST MODEL
myScaleSet_1	Running	Yes
myScaleSet_2	Running	Yes
myScaleSet_4	Creating	Yes
myScaleSet_5	Creating	Yes
myScaleSet_6	Creating	Yes
myScaleSet_7	Creating	Yes

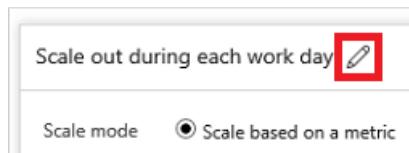
Autoscale based on a schedule

The previous examples automatically scaled a scale set in or out with basic host metrics such as CPU usage. You can also create autoscale rules based on schedules. These schedule-based rules allow you to automatically scale out the number of VM instances ahead of an anticipated increase in application demand, such as core work hours, and then automatically scale in the number of instances at a time that you anticipate less demand, such as the weekend.

1. Choose **Scaling** from the menu on the left-hand side of the scale set window. To delete the existing autoscale rules created in the previous examples, choose the trash can icon.



2. Choose to **Add a scale condition**. Select the pencil icon next to rule name, and provide a name such as *Scale out during each work day*.



3. Select the radio button to **Scale to a specific instance count**.
4. To scale up the number of instances, enter *10* as the instance count.
5. Choose **Repeat specific days** for the **Schedule** type.
6. Select all the work days, Monday through Friday.
7. Choose the appropriate timezone, then specify a **Start time** of *09:00*.
8. Choose to **Add a scale condition** again. Repeat the process to create a schedule named *Scale in during the evening* that scales to 3 instances, repeats every weekday, and starts at 18:00.
9. To apply your schedule-based autoscale rules, select **Save**.

The screenshot shows the Azure portal's Scaling blade for a Virtual Machine Scale Set named "myScaleSet". The left sidebar lists various management tabs like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Instances, Storage, Operating system, Size, Continuous delivery (Preview), Properties, Locks, Automation script, Metrics, Alert rules, and New support request. The "Scaling" tab is currently selected and highlighted with a blue background. The main content area displays two scale conditions:

- Scale out during each work day**: Set to scale to a specific instance count of 10. The schedule repeats specific days (Monday through Friday) from 09:00 to 17:00. The timezone is set to (UTC-08:00) Pacific Time (US & Canada).
- Scale in during the evening**: Set to scale to a specific instance count of 3. The schedule repeats specific days (Monday through Friday) from 18:00 to 22:00. The timezone is set to (UTC-08:00) Pacific Time (US & Canada).

A red box highlights the "Save" button at the top of the blade.

To see how your autoscale rules are applied, select **Run history** across the top of the **Scaling** window. The graph and events list shows when the autoscale rules trigger and the number of VM instances in your scale set increases or decreases.

Next steps

In this article, you learned how to use autoscale rules to scale horizontally and increase or decrease the *number* of VM instances in your scale set. You can also scale vertically to increase or decrease the VM instance *size*. For more information, see [Vertical autoscale with Virtual Machine Scale sets](#).

For information on how to manage your VM instances, see [Manage virtual machine scale sets with Azure PowerShell](#).

To learn how to generate alerts when your autoscale rules trigger, see [Use autoscale actions to send email and webhook alert notifications in Azure Monitor](#). You can also [Use audit logs to send email and webhook alert notifications in Azure Monitor](#).

Advanced autoscale configuration using Resource Manager templates for VM Scale Sets

12/23/2019 • 5 minutes to read • [Edit Online](#)

You can scale-in and scale-out in Virtual Machine Scale Sets based on performance metric thresholds, by a recurring schedule, or by a particular date. You can also configure email and webhook notifications for scale actions. This walkthrough shows an example of configuring all these objects using a Resource Manager template on a VM Scale Set.

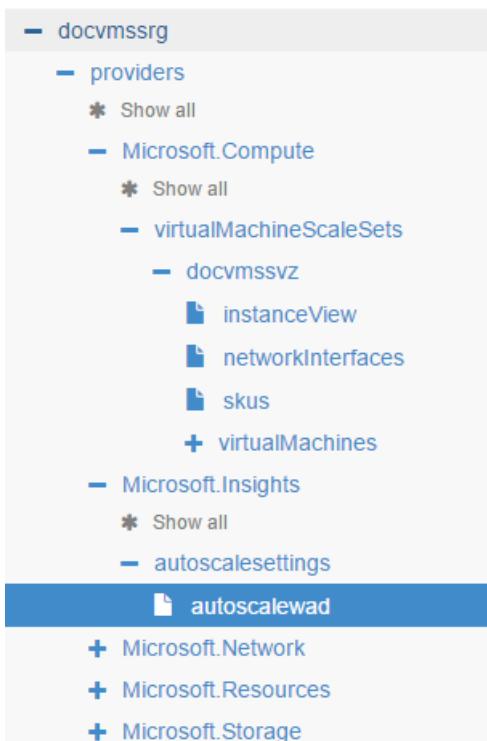
NOTE

While this walkthrough explains the steps for VM Scale Sets, the same information applies to autoscaling [Cloud Services](#), [App Service - Web Apps](#), and [API Management services](#). For a simple scale in/out setting on a VM Scale Set based on a simple performance metric such as CPU, refer to the [Linux](#) and [Windows](#) documents.

Walkthrough

In this walkthrough, we use [Azure Resource Explorer](#) to configure and update the autoscale setting for a scale set. Azure Resource Explorer is an easy way to manage Azure resources via Resource Manager templates. If you are new to Azure Resource Explorer tool, read [this introduction](#).

1. Deploy a new scale set with a basic autoscale setting. This article uses the one from the Azure QuickStart Gallery, which has a Windows scale set with a basic autoscale template. Linux scale sets work the same way.
2. After the scale set is created, navigate to the scale set resource from Azure Resource Explorer. You see the following under Microsoft.Insights node.



The template execution has created a default autoscale setting with the name '**autoscalewad**'. On the right-hand side, you can view the full definition of this autoscale setting. In this case, the default autoscale setting

comes with a CPU% based scale-out and scale-in rule.

3. You can now add more profiles and rules based on the schedule or specific requirements. We create an autoscale setting with three profiles. To understand profiles and rules in autoscale, review [Autoscale Best Practices](#).

PROFILES & RULES	DESCRIPTION
Profile	Performance/metric based
Rule	Service Bus Queue Message Count > x
Rule	Service Bus Queue Message Count < y
Rule	CPU% > n
Rule	CPU% < p
Profile	Weekday morning hours (no rules)
Profile	Product Launch day (no rules)

4. Here is a hypothetical scaling scenario that we use for this walk-through.

- **Load based** - I'd like to scale out or in based on the load on my application hosted on my scale set.*
- **Message Queue size** - I use a Service Bus Queue for the incoming messages to my application. I use the queue's message count and CPU% and configure a default profile to trigger a scale action if either of message count or CPU hits the threshold.*
- **Time of week and day** - I want a weekly recurring 'time of the day' based profile called 'Weekday Morning Hours'. Based on historical data, I know it is better to have certain number of VM instances to handle my application's load during this time.*
- **Special Dates** - I added a 'Product Launch Day' profile. I plan ahead for specific dates so my application is ready to handle the load due marketing announcements and when we put a new product in the application.*
- *The last two profiles can also have other performance metric based rules within them. In this case, I decided not to have one and instead to rely on the default performance metric based rules. Rules are optional for the recurring and date-based profiles.*

Autoscale engine's prioritization of the profiles and rules is also captured in the [autoscaling best practices](#) article. For a list of common metrics for autoscale, refer [Common metrics for Autoscale](#)

5. Make sure you are on the **Read/Write** mode in Resource Explorer

Microsoft (microsoft.onmicrosoft.com) ▾ Read Only Read/Write

autoscalewad

Data (GET, PUT) Actions (POST, DELETE) Create Documentation

GET **Edit** <https://management.azure.com/subscriptions/df602c9c-7aa0-407d-a>

6. Click Edit. Replace the 'profiles' element in autoscale setting with the following configuration:

```

"profiles": [
    {
        "name": "Perf_Based_Scale",
        "capacity": {
            "minimum": "2",
            "maximum": "12",
            "default": "2"
        },
        "rules": [
            {
                "metricTrigger": {
                    "metricName": "MessageCount",
                    "metricNamespace": "",
                    "metricResourceUri": "/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ServiceBus/namespaces/mySB/queues/myqueue",
                    "timeGrain": "PT5M",
                    "statistic": "Average",
                    "timeWindow": "PT5M",
                    "timeAggregation": "Average",
                    "operator": "GreaterThan",
                    "threshold": 10
                },
                "scaleAction": {
                    "direction": "Increase",
                    "type": "ChangeCount",
                    "value": "1",
                    "cooldown": "PT5M"
                }
            },
            {
                "metricTrigger": {
                    "metricName": "MessageCount",
                    "metricNamespace": "",
                    "metricResourceUri": "/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ServiceBus/namespaces/mySB/queues/myqueue",
                    "timeGrain": "PT5M",
                    "statistic": "Average",
                    "timeWindow": "PT5M",
                    "timeAggregation": "Average",
                    "operator": "LessThan",
                    "threshold": 3
                },
                "scaleAction": {
                    "direction": "Decrease",
                    "type": "ChangeCount",
                    "value": "1",
                    "cooldown": "PT5M"
                }
            }
        ]
    }
]

```

```

        "metricTrigger": {
            "metricName": "Percentage CPU",
            "metricNamespace": "",
            "metricResourceUri": "/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.Compute/virtualMachineScaleSets/<this_vmss_name>",
            "timeGrain": "PT5M",
            "statistic": "Average",
            "timeWindow": "PT30M",
            "timeAggregation": "Average",
            "operator": "GreaterThan",
            "threshold": 85
        },
        "scaleAction": {
            "direction": "Increase",
            "type": "ChangeCount",
            "value": "1",
            "cooldown": "PT5M"
        }
    },
    {
        "metricTrigger": {
            "metricName": "Percentage CPU",
            "metricNamespace": "",
            "metricResourceUri": "/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.Compute/virtualMachineScaleSets/<this_vmss_name>",
            "timeGrain": "PT5M",
            "statistic": "Average",
            "timeWindow": "PT30M",
            "timeAggregation": "Average",
            "operator": "LessThan",
            "threshold": 60
        },
        "scaleAction": {
            "direction": "Decrease",
            "type": "ChangeCount",
            "value": "1",
            "cooldown": "PT5M"
        }
    }
]
},
{
    "name": "Weekday_Morning_Hours_Scale",
    "capacity": {
        "minimum": "4",
        "maximum": "12",
        "default": "4"
    },
    "rules": [],
    "recurrence": {
        "frequency": "Week",
        "schedule": {
            "timeZone": "Pacific Standard Time",
            "days": [
                "Monday",
                "Tuesday",
                "Wednesday",
                "Thursday",
                "Friday"
            ],
            "hours": [
                6
            ],
            "minutes": [
                0
            ]
        }
    }
}

```

```

        }
    },
    {
        "name": "Product_Launch_Day",
        "capacity": {
            "minimum": "6",
            "maximum": "20",
            "default": "6"
        },
        "rules": [],
        "fixedDate": {
            "timeZone": "Pacific Standard Time",
            "start": "2016-06-20T00:06:00Z",
            "end": "2016-06-21T23:59:00Z"
        }
    }
}

```

For supported fields and their values, see [Autoscale REST API documentation](#). Now your autoscale setting contains the three profiles explained previously.

- Finally, look at the Autoscale **notification** section. Autoscale notifications allow you to do three things when a scale-out or in action is successfully triggered.

- Notify the admin and co-admins of your subscription
- Email a set of users
- Trigger a webhook call. When fired, this webhook sends metadata about the autoscaling condition and the scale set resource. To learn more about the payload of autoscale webhook, see [Configure Webhook & Email Notifications for Autoscale](#).

Add the following to the Autoscale setting replacing your **notification** element whose value is null

```

"notifications": [
    {
        "operation": "Scale",
        "email": {
            "sendToSubscriptionAdministrator": true,
            "sendToSubscriptionCoAdministrators": false,
            "customEmails": [
                "user1@mycompany.com",
                "user2@mycompany.com"
            ]
        },
        "webhooks": [
            {
                "serviceUri": "https://foo.webhook.example.com?token=abcd1234",
                "properties": {
                    "optional_key1": "optional_value1",
                    "optional_key2": "optional_value2"
                }
            }
        ]
    }
]

```

Hit **Put** button in Resource Explorer to update the autoscale setting.

You have updated an autoscale setting on a VM Scale set to include multiple scale profiles and scale notifications.

Next Steps

Use these links to learn more about autoscaling.

[TroubleShoot Autoscale with Virtual Machine Scale Sets](#)

[Common Metrics for Autoscale](#)

[Best Practices for Azure Autoscale](#)

[Manage Autoscale using PowerShell](#)

[Manage Autoscale using CLI](#)

[Configure Webhook & Email Notifications for Autoscale](#)

[Microsoft.Insights/autoscalesettings template reference](#)

Troubleshooting autoscale with Virtual Machine Scale Sets

1/19/2020 • 4 minutes to read • [Edit Online](#)

Problem – you've created an autoscaling infrastructure in Azure Resource Manager using virtual machine scale sets – for example, by deploying a template like this one: <https://github.com/Azure/azure-quickstart-templates/tree/master/201-vmss-bottle-autoscale> – you have your scale rules defined and it works great, except no matter how much load you put on the VMs, it doesn't autoscale.

Troubleshooting steps

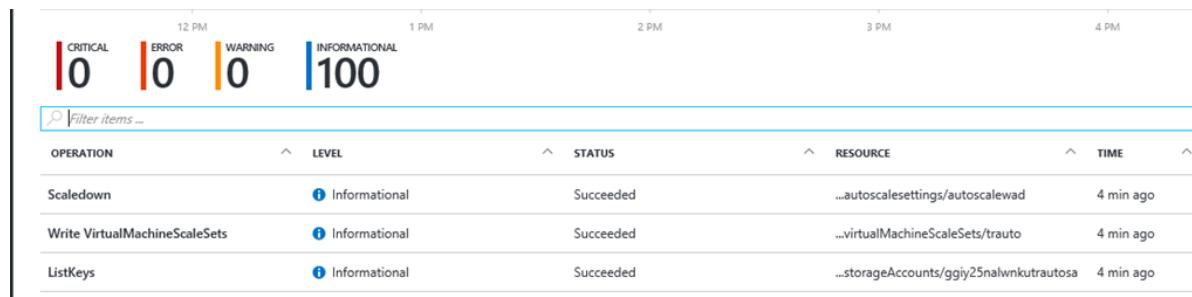
Some things to consider include:

- How many vCPUs does each VM have, and are you loading each vCPU? The preceding sample Azure Quickstart template has a `do_work.php` script, which loads a single vCPU. If you're using a VM bigger than a single-vCPU VM size like Standard_A1 or D1, you'd need to run this load multiple times. Check how many vCPUs for your VMs by reviewing [Sizes for Windows virtual machines in Azure](#)
- How many VMs in the virtual machine scale set, are you doing work on each one?

A scale-out event only takes place when the average CPU across **all** the VMs in a scale set exceeds the threshold value, over the time interval defined in the autoscale rules.

- Did you miss any scale events?

Check the audit logs in the Azure portal for scale events. Maybe there was a scale up and a scale down that was missed. You can filter by "Scale".



- Are your scale-in and scale-out thresholds sufficiently different?

Suppose you set a rule to scale out when average CPU is greater than 50% over five minutes, and to scale in when average CPU is less than 50%. This setting would cause a "flapping" problem when CPU usage is close to the threshold, with scale actions constantly increasing and decreasing the size of the set. Because of this setting, the autoscale service tries to prevent "flapping", which can manifest as not scaling. Therefore, be sure your scale-out and scale-in thresholds are sufficiently different to allow some space in between scaling.

- Did you write your own JSON template?

It is easy to make mistakes, so start with a template like the one above which is proven to work, and make small incremental changes.

- Can you manually scale in or out?

Try redeploying the virtual machine scale set resource with a different "capacity" setting to change the

number of VMs manually. An example template is here: <https://github.com/Azure/azure-quickstart-templates/tree/master/201-vmss-scale-existing> – you might need to edit the template to make sure it has the same machine size as your Scale Set uses. If you can successfully change the number of VMs manually, you then know the problem is isolated to autoscale.

- Check your Microsoft.Compute/virtualMachineScaleSet, and Microsoft.Insights resources in the [Azure Resource Explorer](#)

The Azure Resource Explorer is an indispensable troubleshooting tool that shows you the state of your Azure Resource Manager resources. Click on your subscription and look at the Resource Group you are troubleshooting. Under the Compute resource provider, look at the virtual machine scale set you created and check the Instance View, which shows you the state of a deployment. Also, check the instance view of VMs in the virtual machine scale set. Then, go into the Microsoft.Insights resource provider and check that the autoscale rules look right.

- Is the Diagnostic extension working and emitting performance data?

Update: Azure autoscale has been enhanced to use a host-based metrics pipeline, which no longer requires a diagnostics extension to be installed. The next few paragraphs no longer apply if you create an autoscaling application using the new pipeline. An example of Azure templates that have been converted to use the host pipeline is available here: <https://github.com/Azure/azure-quickstart-templates/tree/master/201-vmss-bottle-autoscale>.

Using host-based metrics for autoscale is better for the following reasons:

- Fewer moving parts as no diagnostics extensions need to be installed.
- Simpler templates. Just add insights autoscale rules to an existing scale set template.
- More reliable reporting and faster launching of new VMs.

The only reasons you might want to keep using a diagnostic extension is if you need memory diagnostics reporting/scaling. Host-based metrics don't report memory.

With that in mind, only follow the rest of this article if you're using diagnostic extensions for your autoscaling.

Autoscale in Azure Resource Manager can work (but no longer has to) by means of a VM extension called the Diagnostics Extension. It emits performance data to a storage account you define in the template. This data is then aggregated by the Azure Monitor service.

If the Insights service can't read data from the VMs, it is supposed to send you an email. For example, you get an email if the VMs are down. Be sure to check your email, at the email address you specified when you created your Azure account.

You can also look at the data yourself. Look at the Azure storage account using a cloud explorer. For example, using the [Visual Studio Cloud Explorer](#), log in and pick the Azure subscription you're using. Then, look at the Diagnostics storage account name referenced in the Diagnostics extension definition in your deployment template.

The screenshot shows the Microsoft Azure Cloud Explorer interface. In the top navigation bar, there's a 'Resource Types' dropdown, a refresh icon, and a settings gear icon. Below the navigation is a search bar with the placeholder 'Search for resources'. The main content area displays a hierarchical tree view of resources. At the top level, there's a folder named 'gbstorage'. Underneath it is a folder named 'ggiy25nalwnkutrautosa', which contains 'Blob Containers' and 'Queues'. Below 'ggiy25nalwnkutrautosa' is another folder named 'Tables', which contains several table entries: 'LinuxCpuVer2v0' (selected), 'LinuxDiskVer2v0', 'LinuxMemoryVer2v0', 'LinuxsyslogVer2v0', and 'SchemasTable'. A status bar at the bottom indicates 'WADML - DETAILS - DETAILS - DETAILS - DETAILS - DETAILS - DETAILS'.

You see a bunch of tables where the data from each VM is being stored. Taking Linux and the CPU metric as an example, look at the most recent rows. The Visual Studio cloud explorer supports a query language so you can run a query. For example, you can run a query for "Timestamp gt datetime'2016-02-02T21:20:00Z'" to make sure you get the most recent events. The timezone corresponds to UTC. Does the data you see in there correspond to the scale rules you set up? In the following example, the CPU for machine 20 started increasing to 100% over the last five minutes.

PartitionKey	RowKey	Timestamp	Host	N	PercentIOWaitTime	PercentIdleTime	PercentProcessor
0000000000000000...	trauto-20_007...	2/2/2016 9:20:2...	trauto-20	0000000000000000...	0	4	96
0000000000000000...	trauto-20_034...	2/2/2016 9:21:2...	trauto-20	0000000000000000...	0	26	74
0000000000000000...	trauto-20_058...	2/2/2016 9:22:2...	trauto-20	0000000000000000...	0	47	53
0000000000000000...	trauto-20_034...	2/2/2016 9:23:2...	trauto-20	0000000000000000...	0	67	33
0000000000000000...	trauto-20_007...	2/2/2016 9:24:2...	trauto-20	0000000000000000...	0	85	15
0000000000000000...	trauto-20_007...	2/2/2016 9:25:2...	trauto-20	0000000000000000...	0	91	9
0000000000000000...	trauto-20_002...	2/2/2016 9:26:2...	trauto-20	0000000000000000...	0	73	27
0000000000000000...	trauto-20_011...	2/2/2016 9:27:2...	trauto-20	0000000000000000...	0	53	47
0000000000000000...	trauto-20_067...	2/2/2016 9:28:2...	trauto-20	0000000000000000...	0	33	67
0000000000000000...	trauto-20_027...	2/2/2016 9:29:2...	trauto-20	0000000000000000...	0	11	89
0000000000000000...	trauto-20_034...	2/2/2016 9:30:2...	trauto-20	0000000000000000...	0	0	100
0000000000000000...	trauto-20_029...	2/2/2016 9:31:2...	trauto-20	0000000000000000...	0	0	100
0000000000000000...	trauto-20_070...	2/2/2016 9:32:2...	trauto-20	0000000000000000...	0	0	100

If the data is not there, it implies the problem is with the diagnostic extension running in the VMs. If the data is there, it implies there is either a problem with your scale rules, or with the Insights service. Check [Azure Status](#).

Once you've been through these steps, if you're still having autoscale problems, you can try the following resources:

- Read the forums on [MSDN](#), or [Stack overflow](#)
- Log a support call. Be prepared to share the template and a view of your performance data.

Deploy your application on virtual machine scale sets

1/19/2020 • 4 minutes to read • [Edit Online](#)

To run applications on virtual machine (VM) instances in a scale set, you first need to install the application components and required files. This article introduces ways to build a custom VM image for instances in a scale set, or automatically run install scripts on existing VM instances. You also learn how to manage application or OS updates across a scale set.

Build a custom VM image

When you use one of the Azure platform images to create the instances in your scale set, no additional software is installed or configured. You can automate the install of these components, however that adds to the time it takes to provision VM instances to your scale sets. If you apply many configuration changes to the VM instances, there is management overhead with those configuration scripts and tasks.

To reduce the configuration management and time to provision a VM, you can create a custom VM image that is ready to run your application as soon as an instance is provisioned in the scale set. For more information on how to create and use a custom VM image with a scale set, see the following tutorials:

- [Azure CLI](#)
- [Azure PowerShell](#)

Install an app with the Custom Script Extension

The Custom Script Extension downloads and executes scripts on Azure VMs. This extension is useful for post deployment configuration, software installation, or any other configuration / management task. Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run-time. For more information on how to install an app with a Custom Script Extension, see the following tutorials:

- [Azure CLI](#)
- [Azure PowerShell](#)
- [Azure Resource Manager template](#)

Install an app to a Windows VM with PowerShell DSC

[PowerShell Desired State Configuration \(DSC\)](#) is a management platform to define the configuration of target machines. DSC configurations define what to install on a machine and how to configure the host. A Local Configuration Manager (LCM) engine runs on each target node that processes requested actions based on pushed configurations.

The PowerShell DSC extension lets you customize VM instances in a scale set with PowerShell. The following example:

- Instructs the VM instances to download a DSC package from GitHub - <https://github.com/Azure-Samples/compute-automation-configurations/raw/master/dsc.zip>
- Sets the extension to run an install script - `configure-http.ps1`
- Gets information about a scale set with [Get-AzVmss](#)
- Applies the extension to the VM instances with [Update-AzVmss](#)

The DSC extension is applied to the `myScaleSet` VM instances in the resource group named `myResourceGroup`. Enter your own names as follows:

```

# Define the script for your Desired Configuration to download and run
$dscConfig = @{
    "wmfVersion" = "latest";
    "configuration" = @{
        "url" = "https://github.com/Azure-Samples/compute-automation-configurations/raw/master/dsc.zip";
        "script" = "configure-http.ps1";
        "function" = "WebsiteTest";
    };
}

# Get information about the scale set
$vmss = Get-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -VMScaleSetName "myScaleSet"

# Add the Desired State Configuration extension to install IIS and configure basic website
$vmss = Add-AzVmssExtension `

    -VirtualMachineScaleSet $vmss `

    -Publisher Microsoft.PowerShell `

    -Type DSC `

    -TypeHandlerVersion 2.24 `

    -Name "DSC" `

    -Setting $dscConfig

# Update the scale set and apply the Desired State Configuration extension to the VM instances
Update-AzVmss `

    -ResourceGroupName "myResourceGroup" `

    -Name "myScaleSet" `

    -VirtualMachineScaleSet $vmss

```

If the upgrade policy on your scale set is *manual*, update your VM instances with [Update-AzVmssInstance](#). This cmdlet applies the updated scale set configuration to the VM instances and installs your application.

Install an app to a Linux VM with cloud-init

[Cloud-init](#) is a widely used approach to customize a Linux VM as it boots for the first time. You can use cloud-init to install packages and write files, or to configure users and security. As cloud-init runs during the initial boot process, there are no additional steps or required agents to apply your configuration.

Cloud-init also works across distributions. For example, you don't use **apt-get install** or **yum install** to install a package. Instead you can define a list of packages to install. Cloud-init automatically uses the native package management tool for the distro you select.

For more information, including an example *cloud-init.txt* file, see [Use cloud-init to customize Azure VMs](#).

To create a scale set and use a cloud-init file, add the `--custom-data` parameter to the `az vmss create` command and specify the name of a cloud-init file. The following example creates a scale set named *myScaleSet* in *myResourceGroup* and configures VM instances with a file named *cloud-init.txt*. Enter your own names as follows:

```

az vmss create \
    --resource-group myResourceGroup \
    --name myScaleSet \
    --image UbuntuLTS \
    --upgrade-policy-mode automatic \
    --custom-data cloud-init.txt \
    --admin-username azureuser \
    --generate-ssh-keys

```

Install applications with OS updates

When new OS releases are available, you can use or build a new custom image and [deploy OS upgrades](#) to a

scale set. Each VM instance is upgraded to the latest image that you specify. You can use a custom image with the application pre-installed, the Custom Script Extension, or PowerShell DSC to have your application automatically available as you perform the upgrade. You may need to plan for application maintenance as you perform this process to ensure that there are no version compatibility issues.

If you use a custom VM image with the application pre-installed, you could integrate the application updates with a deployment pipeline to build the new images and deploy OS upgrades across the scale set. This approach allows the pipeline to pick up the latest application builds, create and validate a VM image, then upgrade the VM instances in the scale set. To run a deployment pipeline that builds and deploys application updates across custom VM images, you could [create a Packer image and deploy with Azure DevOps Services](#), or use another platform such as [Spinnaker](#) or [Jenkins](#).

Next steps

As you build and deploy applications to your scale sets, you can review the [Scale Set Design Overview](#). For more information on how to manage your scale set, see [Use PowerShell to manage your scale set](#).

Sequence extension provisioning in virtual machine scale sets

1/19/2020 • 5 minutes to read • [Edit Online](#)

Azure virtual machine extensions provide capabilities such as post-deployment configuration and management, monitoring, security, and more. Production deployments typically use a combination of multiple extensions configured for the VM instances to achieve desired results.

When using multiple extensions on a virtual machine, it's important to ensure that extensions requiring the same OS resources aren't trying to acquire these resources at the same time. Some extensions also depend on other extensions to provide required configurations such as environment settings and secrets. Without the correct ordering and sequencing in place, dependent extension deployments can fail.

This article details how you can sequence extensions to be configured for the VM instances in virtual machine scale sets.

Prerequisites

This article assumes that you're familiar with:

- Azure virtual machine [extensions](#)
- [Modifying](#) virtual machine scale sets

When to use extension sequencing

Sequencing extensions is not mandatory for scale sets, and unless specified, extensions can be provisioned on a scale set instance in any order.

For example, if your scale set model has two extensions – ExtensionA and ExtensionB – specified in the model, then either of the following provisioning sequences may occur:

- ExtensionA -> ExtensionB
- ExtensionB -> ExtensionA

If your application requires Extension A to always be provisioned before Extension B, then you should use extension sequencing as described in this article. With extension sequencing, only one sequence will now occur:

- ExtensionA -> ExtensionB

Any extensions not specified in a defined provisioning sequence can be provisioned at any time, including before, after, or during a defined sequence. Extension sequencing only specifies that a specific extension will be provisioned after another specific extension. It does not impact the provisioning of any other extension defined in the model.

For example, if your scale set model has three extensions – Extension A, Extension B and Extension C – specified in the model, and Extension C is set to be provisioned after Extension A, then either of the following provisioning sequences may occur:

- ExtensionA -> ExtensionC -> ExtensionB
- ExtensionB -> ExtensionA -> ExtensionC
- ExtensionA -> ExtensionB -> ExtensionC

If you need to ensure that no other extension is provisioned while the defined extension sequence is executing, we recommend sequencing all extensions in your scale set model. In the above example, Extension B can be set to be provisioned after Extension C such that only one sequence can occur:

- ExtensionA -> ExtensionC -> ExtensionB

How to use extension sequencing

To sequence extension provisioning, you must update the extension definition in the scale set model to include the property "provisionAfterExtensions", which accepts an array of extension names. The extensions mentioned in the property array value must be fully defined in the scale set model.

Template Deployment

The following example defines a template where the scale set has three extensions – ExtensionA, ExtensionB, and ExtensionC – such that extensions are provisioned in the order:

- ExtensionA -> ExtensionB -> ExtensionC

```
"virtualMachineProfile": {
  "extensionProfile": {
    "extensions": [
      {
        "name": "ExtensionA",
        "properties": {
          "publisher": "ExtensionA.Publisher",
          "settings": {},
          "typeHandlerVersion": "1.0",
          "autoUpgradeMinorVersion": true,
          "type": "ExtensionA"
        }
      },
      {
        "name": "ExtensionB",
        "properties": {
          "provisionAfterExtensions": [
            "ExtensionA"
          ],
          "publisher": "ExtensionB.Publisher",
          "settings": {},
          "typeHandlerVersion": "2.0",
          "autoUpgradeMinorVersion": true,
          "type": "ExtensionB"
        }
      },
      {
        "name": "ExtensionC",
        "properties": {
          "provisionAfterExtensions": [
            "ExtensionB"
          ],
          "publisher": "ExtensionC.Publisher",
          "settings": {},
          "typeHandlerVersion": "3.0",
          "autoUpgradeMinorVersion": true,
          "type": "ExtensionC"
        }
      }
    ]
  }
}
```

Since the property "provisionAfterExtensions" accepts an array of extension names, the above example can be modified such that ExtensionC is provisioned after ExtensionA and ExtensionB, but no ordering is required

between ExtensionA and ExtensionB. The following template can be used to achieve this scenario:

```
"virtualMachineProfile": {
  "extensionProfile": [
    "extensions": [
      {
        "name": "ExtensionA",
        "properties": {
          "publisher": "ExtensionA.Publisher",
          "settings": {},
          "typeHandlerVersion": "1.0",
          "autoUpgradeMinorVersion": true,
          "type": "ExtensionA"
        }
      },
      {
        "name": "ExtensionB",
        "properties": {
          "publisher": "ExtensionB.Publisher",
          "settings": {},
          "typeHandlerVersion": "2.0",
          "autoUpgradeMinorVersion": true,
          "type": "ExtensionB"
        }
      },
      {
        "name": "ExtensionC",
        "properties": {
          "provisionAfterExtensions": [
            "ExtensionA", "ExtensionB"
          ],
          "publisher": "ExtensionC.Publisher",
          "settings": {},
          "typeHandlerVersion": "3.0",
          "autoUpgradeMinorVersion": true,
          "type": "ExtensionC"
        }
      }
    ]
  }
}
```

REST API

The following example adds a new extension named ExtensionC to a scale set model. ExtensionC has dependencies on ExtensionA and ExtensionB, which have already been defined in the scale set model.

```
PUT on
`/subscriptions/subscription_id/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/extensions/ExtensionC?api-version=2018-10-01`
```

```
{  
  "name": "ExtensionC",  
  "properties": {  
    "provisionAfterExtensions": [  
      "ExtensionA", "ExtensionB"  
    ],  
    "publisher": "ExtensionC.Publisher",  
    "settings": {},  
    "typeHandlerVersion": "3.0",  
    "autoUpgradeMinorVersion": true,  
    "type": "ExtensionC"  
  }  
}
```

If ExtensionC was defined earlier in the scale set model and you now want to add its dependencies, you can execute a `PATCH` to edit the already deployed extension's properties.

```
PATCH on  
`/subscriptions/subscription_id/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScale  
Sets/myScaleSet/extensions/ExtensionC?api-version=2018-10-01`
```

```
{  
  "properties": {  
    "provisionAfterExtensions": [  
      "ExtensionA", "ExtensionB"  
    ]  
  }  
}
```

Changes to existing scale set instances are applied on the next [upgrade](#).

Azure PowerShell

Use the [Add-AzVmssExtension](#) cmdlet to add the Application Health extension to the scale set model definition. Extension sequencing requires the use of Az PowerShell 1.2.0 or above.

The following example adds the [Application Health extension](#) to the `extensionProfile` in a scale set model of a Windows-based scale set. The Application Health extension will be provisioned after provisioning the [Custom Script Extension](#), already defined in the scale set.

```

# Define the scale set variables
$vmScaleSetName = "myVMScaleSet"
$vmScaleSetResourceGroup = "myVMScaleSetResourceGroup"

# Define the Application Health extension properties
$publicConfig = @{"protocol" = "http"; "port" = 80; "requestPath" = "/healthEndpoint"};
$extensionName = "myHealthExtension"
$extensionType = "ApplicationHealthWindows"
$publisher = "Microsoft.ManagedServices"

# Get the scale set object
$vmScaleSet = Get-AzVmss `

    -ResourceGroupName $vmScaleSetResourceGroup `

    -VMScaleSetName $vmScaleSetName

# Add the Application Health extension to the scale set model
Add-AzVmssExtension -VirtualMachineScaleSet $vmScaleSet `

    -Name $extensionName `

    -Publisher $publisher `

    -Setting $publicConfig `

    -Type $extensionType `

    -TypeHandlerVersion "1.0" `

    -ProvisionAfterExtension "CustomScriptExtension" `

    -AutoUpgradeMinorVersion $True

# Update the scale set
Update-AzVmss -ResourceGroupName $vmScaleSetResourceGroup `

    -Name $vmScaleSetName `

    -VirtualMachineScaleSet $vmScaleSet

```

Azure CLI 2.0

Use [az vmss extension set](#) to add the Application Health extension to the scale set model definition. Extension sequencing requires the use of Azure CLI 2.0.55 or above.

The following example adds the [Application Health extension](#) to the scale set model of a Windows-based scale set. The Application Health extension will be provisioned after provisioning the [Custom Script Extension](#), already defined in the scale set.

```

az vmss extension set \
    --name ApplicationHealthWindows \
    --publisher Microsoft.ManagedServices \
    --version 1.0 \
    --resource-group <myVMScaleSetResourceGroup> \
    --vmss-name <myVMScaleSet> \
    --provision-after-extensions CustomScriptExtension \
    --settings ./extension.json

```

Troubleshoot

Not able to add extension with dependencies?

1. Ensure that the extensions specified in provisionAfterExtensions are defined in the scale set model.
2. Ensure there are no circular dependencies being introduced. For example, the following sequence isn't allowed:
ExtensionA -> ExtensionB -> ExtensionC -> ExtensionA
3. Ensure that any extensions that you take dependencies on, have a "settings" property under extension "properties". For example, if ExtensionB needs to be provisioned after ExtensionA, then ExtensionA must have the "settings" field under ExtensionA "properties". You can specify an empty "settings" property if the extension does not mandate any required settings.

Not able to remove extensions?

Ensure that the extensions being removed are not listed under provisionAfterExtensions for any other extensions.

Next steps

Learn how to [deploy your application](#) on virtual machine scale sets.

Using Application Health extension with virtual machine scale sets

1/19/2020 • 3 minutes to read • [Edit Online](#)

Monitoring your application health is an important signal for managing and upgrading your deployment. Azure virtual machine scale sets provide support for [rolling upgrades](#) including [automatic OS-image upgrades](#), which rely on health monitoring of the individual instances to upgrade your deployment.

This article describes how you can use the Application Health extension to monitor the health of your applications deployed on virtual machine scale sets.

Prerequisites

This article assumes that you are familiar with:

- Azure virtual machine [extensions](#)
- [Modifying](#) virtual machine scale sets

When to use the Application Health extension

The Application Health extension is deployed inside a virtual machine scale set instance and reports on VM health from inside the scale set instance. You can configure the extension to probe on an application endpoint and update the status of the application on that instance. This instance status is checked by Azure to determine whether an instance is eligible for upgrade operations.

As the extension reports health from within a VM, the extension can be used in situations where external probes such as Application Health Probes (that utilize custom Azure Load Balancer [probes](#)) can't be used.

Extension schema

The following JSON shows the schema for the Application Health extension. The extension requires at a minimum either a "tcp" or "http" request with an associated port or request path respectively.

```
{  
  "type": "extensions",  
  "name": "HealthExtension",  
  "apiVersion": "2018-10-01",  
  "location": "<location>",  
  "properties": {  
    "publisher": "Microsoft.ManagedServices",  
    "type": "< ApplicationHealthLinux or ApplicationHealthWindows >",  
    "autoUpgradeMinorVersion": true,  
    "typeHandlerVersion": "1.0",  
    "settings": {  
      "protocol": "<protocol>",  
      "port": "<port>",  
      "requestPath": "</requestPath>"  
    }  
  }  
}
```

Property values

NAME	VALUE / EXAMPLE	DATA TYPE
apiVersion	2018-10-01	date
publisher	Microsoft.ManagedServices	string
type	ApplicationHealthLinux (Linux), ApplicationHealthWindows (Windows)	string
typeHandlerVersion	1.0	int

Settings

NAME	VALUE / EXAMPLE	DATA TYPE
protocol	http or tcp	string
port	Optional when protocol is http , mandatory when protocol is tcp	int
requestPath	Mandatory when protocol is http , not allowed when protocol is tcp	string

Deploy the Application Health extension

There are multiple ways of deploying the Application Health extension to your scale sets as detailed in the examples below.

REST API

The following example adds the Application Health extension (with name myHealthExtension) to the extensionProfile in the scale set model of a Windows-based scale set.

```
PUT on
`/subscriptions/subscription_id/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/extensions/myHealthExtension?api-version=2018-10-01`
```

```
{
  "name": "myHealthExtension",
  "properties": {
    "publisher": " Microsoft.ManagedServices",
    "type": "< ApplicationHealthWindows>",
    "autoUpgradeMinorVersion": true,
    "typeHandlerVersion": "1.0",
    "settings": {
      "protocol": "<protocol>",
      "port": "<port>",
      "requestPath": "</requestPath>"
    }
  }
}
```

Use PATCH to edit an already deployed extension.

Azure PowerShell

Use the [Add-AzVmssExtension](#) cmdlet to add the Application Health extension to the scale set model definition.

The following example adds the Application Health extension to the `extensionProfile` in the scale set model of a Windows-based scale set. The example uses the new Az PowerShell module.

```
# Define the scale set variables
$vmScaleSetName = "myVMScaleSet"
$vmScaleSetResourceGroup = "myVMScaleSetResourceGroup"

# Define the Application Health extension properties
$publicConfig = @{"protocol" = "http"; "port" = 80; "requestPath" = "/healthEndpoint"};
$extensionName = "myHealthExtension"
$extensionType = "ApplicationHealthWindows"
$publisher = "Microsoft.ManagedServices"

# Get the scale set object
$vmScaleSet = Get-AzVmss `-
    -ResourceGroupName $vmScaleSetResourceGroup `-
    -VMScaleSetName $vmScaleSetName

# Add the Application Health extension to the scale set model
Add-AzVmssExtension -VirtualMachineScaleSet $vmScaleSet `-
    -Name $extensionName `-
    -Publisher $publisher `-
    -Setting $publicConfig `-
    -Type $extensionType `-
    -TypeHandlerVersion "1.0" `-
    -AutoUpgradeMinorVersion $True

# Update the scale set
Update-AzVmss -ResourceGroupName $vmScaleSetResourceGroup `-
    -Name $vmScaleSetName `-
    -VirtualMachineScaleSet $vmScaleSet
```

Azure CLI 2.0

Use [az vmss extension set](#) to add the Application Health extension to the scale set model definition.

The following example adds the Application Health extension to the scale set model of a Linux-based scale set.

```
az vmss extension set \
    --name ApplicationHealthLinux \
    --publisher Microsoft.ManagedServices \
    --version 1.0 \
    --resource-group <myVMScaleSetResourceGroup> \
    --vmss-name <myVMScaleSet> \
    --settings ./extension.json
```

The extension.json file content.

```
{
  "protocol": "<protocol>",
  "port": "<port>",
  "requestPath": "</requestPath>"
}
```

Troubleshoot

Extension execution output is logged to files found in the following directories:

```
C:\WindowsAzure\Logs\Plugins\Microsoft.ManagedServices.ApplicationHealthWindows\<version>\
```

```
/var/lib/waagent/apphealth
```

The logs also periodically capture the application health status.

Next steps

Learn how to [deploy your application](#) on virtual machine scale sets.

Azure virtual machine scale sets and attached data disks

1/19/2020 • 2 minutes to read • [Edit Online](#)

To expand your available storage, Azure [virtual machine scale sets](#) support VM instances with attached data disks. You can attach data disks when the scale set is created, or to an existing scale set.

NOTE

When you create a scale set with attached data disks, you need to mount and format the disks from within a VM to use them (just like for standalone Azure VMs). A convenient way to complete this process is to use a Custom Script Extension that calls a script to partition and format all the data disks on a VM. For examples of this, see [Azure CLI](#) [Azure PowerShell](#).

Create and manage disks in a scale set

For detailed information on how to create a scale set with attached data disks, prepare and format, or add and remove data disks, see one of the following tutorials:

- [Azure CLI](#)
- [Azure PowerShell](#)

The rest of this article outlines specific use cases such as Service Fabric clusters that require data disks, or attaching existing data disks with content to a scale set.

Create a Service Fabric cluster with attached data disks

Each [node type](#) in a [Service Fabric](#) cluster running in Azure is backed by a virtual machine scale set. Using an Azure Resource Manager template, you can attach data disks to the scale set(s) that make up the Service Fabric cluster. You can use an [existing template](#) as a starting point. In the template, include a *dataDisks* section in the *storageProfile* of the *Microsoft.Compute/virtualMachineScaleSets* resource(s) and deploy the template. The following example attaches a 128 GB data disk:

```
"dataDisks": [  
    {  
        "diskSizeGB": 128,  
        "lun": 0,  
        "createOption": "Empty"  
    }  
]
```

You can automatically partition, format, and mount the data disks when the cluster is deployed. Add a custom script extension to the *extensionProfile* of the *virtualMachineProfile* of the scale set(s).

To automatically prepare the data disk(s) in a Windows cluster, add the following:

```
{
  "name": "customScript",
  "properties": {
    "publisher": "Microsoft.Compute",
    "type": "CustomScriptExtension",
    "typeHandlerVersion": "1.8",
    "autoUpgradeMinorVersion": true,
    "settings": {
      "fileUris": [
        "https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/prepare_vm_disks.ps1"
      ],
      "commandToExecute": "powershell -ExecutionPolicy Unrestricted -File prepare_vm_disks.ps1"
    }
  }
}
```

To automatically prepare the data disk(s) in a Linux cluster, add the following:

```
{
  "name": "lapextension",
  "properties": {
    "publisher": "Microsoft.Azure.Extensions",
    "type": "CustomScript",
    "typeHandlerVersion": "2.0",
    "autoUpgradeMinorVersion": true,
    "settings": {
      "fileUris": [
        "https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/prepare_vm_disks.sh"
      ],
      "commandToExecute": "bash prepare_vm_disks.sh"
    }
  }
}
```

Adding pre-populated data disks to an existing scale set

Data disks specified in the scale set model are always empty. However, you may attach an existing data disk to a specific VM in a scale set. This feature is in preview, with examples on [GitHub](#). If you wish to propagate data across all VMs in the scale set, you may duplicate your data disk and attach it to each VM in the scale set, you may create a custom image that contains the data and provision the scale set from this custom image, or you may use Azure Files or a similar data storage offering.

Additional notes

Support for Azure Managed disks and scale set attached data disks is available in API version [2016-04-30-preview](#) or later of the Microsoft.Compute API.

Azure portal support for attached data disks in scale sets is initially limited. Depending on your requirements you can use Azure templates, CLI, PowerShell, SDKs, and REST API to manage attached disks.

Encrypt OS and attached data disks in a virtual machine scale set with Azure PowerShell

1/19/2020 • 4 minutes to read • [Edit Online](#)

The Azure PowerShell module is used to create and manage Azure resources from the PowerShell command line or in scripts. This article shows you how to use Azure PowerShell to create and encrypt a virtual machine scale set. For more information on applying Azure Disk Encryption to a virtual machine scale set, see [Azure Disk Encryption for Virtual Machine Scale Sets](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

Create an Azure Key Vault enabled for disk encryption

Azure Key Vault can store keys, secrets, or passwords that allow you to securely implement them in your applications and services. Cryptographic keys are stored in Azure Key Vault using software-protection, or you can import or generate your keys in Hardware Security Modules (HSMs) certified to FIPS 140-2 level 2 standards. These cryptographic keys are used to encrypt and decrypt virtual disks attached to your VM. You retain control of these cryptographic keys and can audit their use.

Create a Key Vault with [New-AzKeyVault](#). To allow the Key Vault to be used for disk encryption, set the *EnabledForDiskEncryption* parameter. The following example also defines variables for resource group name, Key Vault Name, and location. Provide your own unique Key Vault name:

```
$rgName="myResourceGroup"
$vaultName="myuniquekeyvault"
$location = "EastUS"

New-AzResourceGroup -Name $rgName -Location $location
New-AzKeyVault -VaultName $vaultName -ResourceGroupName $rgName -Location $location -EnabledForDiskEncryption
```

Use an existing Key Vault

This step is only required if you have an existing Key Vault that you wish to use with disk encryption. Skip this step if you created a Key Vault in the previous section.

You can enable an existing Key Vault in the same subscription and region as the scale set for disk encryption with [Set-AzKeyVaultAccessPolicy](#). Define the name of your existing Key Vault in the `$vaultName` variable as follows:

```
$vaultName="myexistingkeyvault"
Set-AzKeyVaultAccessPolicy -VaultName $vaultName -EnabledForDiskEncryption
```

Create a scale set

First, set an administrator username and password for the VM instances with [Get-Credential](#):

```
$cred = Get-Credential
```

Now create a virtual machine scale set with [New-AzVmss](#). To distribute traffic to the individual VM instances, a load balancer is also created. The load balancer includes rules to distribute traffic on TCP port 80, as well as allow remote desktop traffic on TCP port 3389 and PowerShell remoting on TCP port 5985:

```
$vmssName="myScaleSet"

New-AzVmss ` 
    -ResourceGroupName $rgName ` 
    -VMScaleSetName $vmssName ` 
    -Location $location ` 
    -VirtualNetworkName "myVnet" ` 
    -SubnetName "mySubnet" ` 
    -PublicIpAddressName "myPublicIPAddress" ` 
    -LoadBalancerName "myLoadBalancer" ` 
    -UpgradePolicy "Automatic" ` 
    -Credential $cred
```

Enable encryption

To encrypt VM instances in a scale set, first get some information on the Key Vault URI and resource ID with [Get-AzKeyVault](#). These variables are used to then start the encryption process with [Set-AzVmssDiskEncryptionExtension](#):

```
$diskEncryptionKeyVaultUrl=(Get-AzKeyVault -ResourceGroupName $rgName -Name $vaultName).VaultUri
$keyVaultResourceId=(Get-AzKeyVault -ResourceGroupName $rgName -Name $vaultName).ResourceId

Set-AzVmssDiskEncryptionExtension -ResourceGroupName $rgName -VMScaleSetName $vmssName ` 
    -DiskEncryptionKeyVaultUrl $diskEncryptionKeyVaultUrl -DiskEncryptionKeyVaultId $keyVaultResourceId - 
    VolumeType "All"
```

When prompted, type `y` to continue the disk encryption process on the scale set VM instances.

Enable encryption using KEK to wrap the key

You can also use a Key Encryption Key for added security when encrypting the virtual machine scale set.

```
$diskEncryptionKeyVaultUrl=(Get-AzKeyVault -ResourceGroupName $rgName -Name $vaultName).VaultUri  
$keyVaultResourceId=(Get-AzKeyVault -ResourceGroupName $rgName -Name $vaultName).ResourceId  
$keyEncryptionKeyUrl = (Get-AzKeyVaultKey -VaultName $vaultName -Name $keyEncryptionKeyName).Key.kid;  
  
Set-AzVmssDiskEncryptionExtension -ResourceGroupName $rgName -VMScaleSetName $vmssName `  
-DiskEncryptionKeyVaultUrl $diskEncryptionKeyVaultUrl -DiskEncryptionKeyVaultId $keyVaultResourceId `  
-KeyEncryptionKeyUrl $keyEncryptionKeyUrl -KeyEncryptionKeyId $keyVaultResourceId -VolumeType "All"
```

NOTE

The syntax for the value of disk-encryption-keyvault parameter is the full identifier string:

/subscriptions/[subscription-id-guid]/resourceGroups/[resource-group-name]/providers/Microsoft.KeyVault/vaults/[keyvault-name]

The syntax for the value of the key-encryption-key parameter is the full URI to the KEK as in:

https://[keyvault-name].vault.azure.net/keys/[kekname]/[kek-unique-id]

Check encryption progress

To check on the status of disk encryption, use [Get-AzVmssDiskEncryption](#):

```
Get-AzVmssDiskEncryption -ResourceGroupName $rgName -VMScaleSetName $vmssName
```

When VM instances are encrypted, the *EncryptionSummary* code reports *ProvisioningState/succeeded* as shown in the following example output:

```
ResourceGroupName      : myResourceGroup  
VmScaleSetName        : myScaleSet  
EncryptionSettings    :  
  KeyVaultURL          : https://myuniquekeyvault.vault.azure.net/  
  KeyEncryptionKeyURL  :  
  KeyVaultResourceId   :  
/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.KeyVault/vaults/myuniquekeyvault  
  KekVaultResourceId   :  
  KeyEncryptionAlgorithm :  
  VolumeType           : All  
  EncryptionOperation  : EnableEncryption  
EncryptionSummary[0]    :  
  Code                 : ProvisioningState/succeeded  
  Count                : 2  
  EncryptionEnabled    : True  
  EncryptionExtensionInstalled : True
```

Disable encryption

If you no longer wish to use encrypted VM instances disks, you can disable encryption with [Disable-AzVmssDiskEncryption](#) as follows:

```
Disable-AzVmssDiskEncryption -ResourceGroupName $rgName -VMScaleSetName $vmssName
```

Next steps

- In this article, you used Azure PowerShell to encrypt a virtual machine scale set. You can also use the [Azure CLI](#) or [Azure Resource Manager templates](#).
- If you wish to have Azure Disk Encryption applied after another extension is provisioned, you can use [extension sequencing](#).

Encrypt OS and attached data disks in a virtual machine scale set with the Azure CLI

1/19/2020 • 5 minutes to read • [Edit Online](#)

The Azure CLI is used to create and manage Azure resources from the command line or in scripts. This quickstart shows you how to use the Azure CLI to create and encrypt a virtual machine scale set. For more information on applying Azure Disk encryption to a virtual machine scale set, see [Azure Disk Encryption for Virtual Machine Scale Sets](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.31 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a scale set

Before you can create a scale set, create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

```
az group create --name myResourceGroup --location eastus
```

Now create a virtual machine scale set with [az vmss create](#). The following example creates a scale set named

myScaleSet that is set to automatically update as changes are applied, and generates SSH keys if they do not exist in *~/.ssh/id_rsa*. A 32Gb data disk is attached to each VM instance, and the Azure [Custom Script Extension](#) is used to prepare the data disks with [az vmss extension set](#):

```
# Create a scale set with attached data disk
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--admin-username azureuser \
--generate-ssh-keys \
--data-disk-sizes-gb 32

# Prepare the data disk for use with the Custom Script Extension
az vmss extension set \
--publisher Microsoft.Azure.Extensions \
--version 2.0 \
--name CustomScript \
--resource-group myResourceGroup \
--vmss-name myScaleSet \
--settings '{"fileUris":["https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/prepare_vm_disks.sh"],"commandToExecute":"./prepare_vm_disks.sh"}'
```

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

Create an Azure key vault enabled for disk encryption

Azure Key Vault can store keys, secrets, or passwords that allow you to securely implement them in your applications and services. Cryptographic keys are stored in Azure Key Vault using software-protection, or you can import or generate your keys in Hardware Security Modules (HSMs) certified to FIPS 140-2 level 2 standards. These cryptographic keys are used to encrypt and decrypt virtual disks attached to your VM. You retain control of these cryptographic keys and can audit their use.

Define your own unique *keyvault_name*. Then, create a KeyVault with [az keyvault create](#) in the same subscription and region as the scale set, and set the *--enabled-for-disk-encryption* access policy.

```
# Provide your own unique Key Vault name
keyvault_name=myuniquekeyvaultname

# Create Key Vault
az keyvault create --resource-group myResourceGroup --name $keyvault_name --enabled-for-disk-encryption
```

Use an existing Key Vault

This step is only required if you have an existing Key Vault that you wish to use with disk encryption. Skip this step if you created a Key Vault in the previous section.

Define your own unique *keyvault_name*. Then, updated your KeyVault with [az keyvault update](#) and set the *--enabled-for-disk-encryption* access policy.

```
# Provide your own unique Key Vault name
keyvault_name=myuniquekeyvaultname

# Create Key Vault
az keyvault update --name $keyvault_name --enabled-for-disk-encryption
```

Enable encryption

To encrypt VM instances in a scale set, first get some information on the Key Vault resource ID with [az keyvault show](#). These variables are used to then start the encryption process with [az vmss encryption enable](#):

```
# Get the resource ID of the Key Vault
vaultResourceId=$(az keyvault show --resource-group myResourceGroup --name $keyvault_name --query id -o tsv)

# Enable encryption of the data disks in a scale set
az vmss encryption enable \
--resource-group myResourceGroup \
--name myScaleSet \
--disk-encryption-keyvault $vaultResourceId \
--volume-type DATA
```

It may take a minute or two for the encryption process to start.

As the scale set's upgrade policy on the scale set created in an earlier step is set to *automatic*, the VM instances automatically start the encryption process. On scale sets where the upgrade policy is to manual, start the encryption policy on the VM instances with [az vmss update-instances](#).

Enable encryption using KEK to wrap the key

You can also use a Key Encryption Key for added security when encrypting the virtual machine scale set.

```
# Get the resource ID of the Key Vault
vaultResourceId=$(az keyvault show --resource-group myResourceGroup --name $keyvault_name --query id -o tsv)

# Enable encryption of the data disks in a scale set
az vmss encryption enable \
--resource-group myResourceGroup \
--name myScaleSet \
--disk-encryption-keyvault $vaultResourceId \
--key-encryption-key myKEK \
--key-encryption-keyvault $vaultResourceId \
--volume-type DATA
```

NOTE

The syntax for the value of disk-encryption-keyvault parameter is the full identifier string:
/subscriptions/[subscription-id-guid]/resourceGroups/[resource-group-name]/providers/Microsoft.KeyVault/vaults/[keyvault-name]

The syntax for the value of the key-encryption-key parameter is the full URI to the KEK as in:
[https://\[keyvault-name\].vault.azure.net/keys/\[kekname\]/\[kek-unique-id\]](https://[keyvault-name].vault.azure.net/keys/[kekname]/[kek-unique-id])

Check encryption progress

To check on the status of disk encryption, use [az vmss encryption show](#):

```
az vmss encryption show --resource-group myResourceGroup --name myScaleSet
```

When VM instances are encrypted, the status code reports *EncryptionState/encrypted*, as shown in the following example output:

```
[  
  {  
    "disks": [  
      {  
        "encryptionSettings": null,  
        "name": "myScaleSet_myScaleSet_0_disk2_3f39c2019b174218b98b3dfa3424e69",  
        "statuses": [  
          {  
            "additionalProperties": {},  
            "code": "EncryptionState/encrypted",  
            "displayStatus": "Encryption is enabled on disk",  
            "level": "Info",  
            "message": null,  
            "time": null  
          }  
        ]  
      }  
    ],  
    "id":  
      "/subscriptions/guid/resourceGroups/MYRESOURCEGROUP/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/virtualMachines/0",  
      "resourceGroup": "MYRESOURCEGROUP"  
    }  
  ]
```

Disable encryption

If you no longer wish to use encrypted VM instances disks, you can disable encryption with [az vmss encryption disable](#) as follows:

```
az vmss encryption disable --resource-group myResourceGroup --name myScaleSet
```

Next steps

- In this article, you used the Azure CLI to encrypt a virtual machine scale set. You can also use [Azure PowerShell](#) or [Azure Resource Manager templates](#).
- If you wish to have Azure Disk Encryption applied after another extension is provisioned, you can use [extension sequencing](#).
- An end-to-end batch file example for Linux scale set data disk encryption can be found [here](#). This example creates a resource group, Linux scale set, mounts a 5-GB data disk, and encrypts the virtual machine scale set.

Encrypt virtual machine scale sets with Azure Resource Manager

10/17/2019 • 2 minutes to read • [Edit Online](#)

You can encrypt or decrypt Linux virtual machine scale sets using Azure Resource Manager templates.

Deploying templates

First, select the template that fits your scenario.

- [Enable disk encryption on a running Linux virtual machine scale set](#)
- [Enable disk encryption on a running Windows virtual machine scale set](#)
 - [Deploy a virtual machine scale set of Linux VMs with a jumpbox and enables encryption on Linux virtual machine scale sets](#)
 - [Deploy a virtual machine scale set of Windows VMs with a jumpbox and enables encryption on Windows virtual machine scale sets](#)
- [Disable disk encryption on a running Linux virtual machine scale set](#)
- [Disable disk encryption on a running Windows virtual machine scale set](#)

Then follow these steps:

1. Click **Deploy to Azure**.
2. Fill in the required fields then agree to the terms and conditions.
3. Click **Purchase** to deploy the template.

Next steps

- [Azure Disk Encryption for virtual machine scale sets](#)
- [Encrypt a virtual machine scale sets using the Azure CLI](#)
- [Encrypt a virtual machine scale sets using the Azure PowerShell](#)
- [Create and configure a key vault for Azure Disk Encryption](#)
- [Use Azure Disk Encryption with virtual machine scale set extension sequencing](#)

Use Azure Disk Encryption with virtual machine scale set extension sequencing

1/19/2020 • 2 minutes to read • [Edit Online](#)

Extensions such as Azure disk encryption can be added to an Azure virtual machines scale set in a specified order. To do so, use [extension sequencing](#).

In general, encryption should be applied to a disk:

- After extensions or custom scripts that prepare the disks or volumes.
- Before extensions or custom scripts that access or consume the data on the encrypted disks or volumes.

In either case, the `provisionAfterExtensions` property designates which extension should be added later in the sequence.

Sample Azure templates

If you wish to have Azure Disk Encryption applied after another extension, put the `provisionAfterExtensions` property in the `AzureDiskEncryption` extension block.

Here is an example using "CustomScriptExtension", a Powershell script that initializes and formats a Windows disk, followed by "AzureDiskEncryption":

```

"virtualMachineProfile": {
  "extensionProfile": {
    "extensions": [
      {
        "type": "Microsoft.Compute/virtualMachineScaleSets/extensions",
        "name": "CustomScriptExtension",
        "location": "[resourceGroup().location]",
        "properties": {
          "publisher": "Microsoft.Compute",
          "type": "CustomScriptExtension",
          "typeHandlerVersion": "1.9",
          "autoUpgradeMinorVersion": true,
          "forceUpdateTag": "[parameters('forceUpdateTag')]",
          "settings": {
            "fileUris": [
              "https://raw.githubusercontent.com/Azure-Samples/compute-automation-configurations/master/ade-vmss/FormatMBRDisk.ps1"
            ]
          },
          "protectedSettings": {
            "commandToExecute": "powershell -ExecutionPolicy Unrestricted -File FormatMBRDisk.ps1"
          }
        }
      },
      {
        "type": "Microsoft.Compute/virtualMachineScaleSets/extensions",
        "name": "AzureDiskEncryption",
        "location": "[resourceGroup().location]",
        "properties": {
          "provisionAfterExtensions": [
            "CustomScriptExtension"
          ],
          "publisher": "Microsoft.Azure.Security",
          "type": "AzureDiskEncryption",
          "typeHandlerVersion": "2.2",
          "autoUpgradeMinorVersion": true,
          "forceUpdateTag": "[parameters('forceUpdateTag')]",
          "settings": {
            "EncryptionOperation": "EnableEncryption",
            "KeyVaultURL": "[reference(variables('keyVaultResourceId'), '2018-02-14-preview').vaultUri]",
            "KeyVaultResourceId": "[variables('keyVaultResourceID')]",
            "KeyEncryptionKeyURL": "[parameters('keyEncryptionKeyURL')]",
            "KekVaultResourceId": "[variables('keyVaultResourceID')]",
            "KeyEncryptionAlgorithm": "[parameters('keyEncryptionAlgorithm')]",
            "volumeType": "[parameters('volumeType')]",
            "SequenceVersion": "[parameters('sequenceVersion')]"
          }
        }
      }
    ]
  }
}

```

If you wish to have Azure Disk Encryption applied before another extension, put the `provisionAfterExtensions` property in the block of the extension to follow.

Here is an example using "AzureDiskEncryption" followed by "VMDiagnosticsSettings", an extension that provides monitoring and diagnostics capabilities on a Windows-based Azure VM:

```

"virtualMachineProfile": {
  "extensionProfile": {
    "extensions": [
      {
        "name": "AzureDiskEncryption",
        "type": "Microsoft.Compute/virtualMachineScaleSets/extensions",
        "location": "[resourceGroup().location]",
        "properties": {
          "publisher": "Microsoft.Azure.Security",
          "type": "AzureDiskEncryption",
          "typeHandlerVersion": "2.2",
          "autoUpgradeMinorVersion": true,
          "forceUpdateTag": "[parameters('forceUpdateTag')]",
          "settings": {
            "EncryptionOperation": "EnableEncryption",
            "KeyVaultURL": "[reference(variables('keyVaultResourceId'), '2018-02-14-preview').vaultUri]",
            "KeyVaultResourceId": "[variables('keyVaultResourceID')]",
            "KeyEncryptionKeyURL": "[parameters('keyEncryptionKeyURL')]",
            "KekVaultResourceId": "[variables('keyVaultResourceID')]",
            "KeyEncryptionAlgorithm": "[parameters('keyEncryptionAlgorithm')]",
            "VolumeType": "[parameters('volumeType')]",
            "SequenceVersion": "[parameters('sequenceVersion')]"
          }
        }
      },
      {
        "name": "Microsoft.Insights.VMDiagnosticsSettings",
        "type": "extensions",
        "location": "[resourceGroup().location]",
        "apiVersion": "2016-03-30",
        "dependsOn": [
          "[concat('Microsoft.Compute/virtualMachines/myVM', copyindex())]"
        ],
        "properties": {
          "provisionAfterExtensions": [
            "AzureDiskEncryption"
          ],
          "publisher": "Microsoft.Azure.Diagnostics",
          "type": "IaaSDiagnostics",
          "typeHandlerVersion": "1.5",
          "autoUpgradeMinorVersion": true,
          "settings": {
            "xmlCfg": "[base64(concat(variables('wadcfgxstart'),
              variables('wadmetricsresourceid'),
              concat('myVM', copyindex()),
              variables('wadcfgxend')))]",
            "storageAccount": "[variables('storageName')]"
          },
          "protectedSettings": {
            "storageAccountName": "[variables('storageName')]",
            "storageAccountKey": "[listkeys(variables('accountid'),
              '2015-06-15').key1]",
            "storageAccountEndPoint": "https://core.windows.net"
          }
        }
      }
    ],
    "dependsOn": []
  }
}

```

For a more in-depth template, see:

- Apply the Azure Disk Encryption extension after a custom shell script that formats the disk (Linux): [deploy-extseq-linux-ADE-after-customscript.json](#)

Next steps

- Learn more about extension sequencing: [Sequence extension provisioning in virtual machine scale sets](#).
- Learn more about the `provisionAfterExtensions` property: [Microsoft.Compute virtualMachineScaleSets/extensions template reference](#).
- [Azure Disk Encryption for virtual machine scale sets](#)
- [Encrypt a virtual machine scale sets using the Azure CLI](#)
- [Encrypt a virtual machine scale sets using the Azure PowerShell](#)
- [Create and configure a key vault for Azure Disk Encryption](#)

Creating and configuring a key vault for Azure Disk Encryption

1/19/2020 • 5 minutes to read • [Edit Online](#)

Azure Disk Encryption uses Azure Key Vault to control and manage disk encryption keys and secrets. For more information about key vaults, see [Get started with Azure Key Vault](#) and [Secure your key vault](#).

Creating and configuring a key vault for use with Azure Disk Encryption involves three steps:

1. Creating a resource group, if needed.
2. Creating a key vault.
3. Setting key vault advanced access policies.

These steps are illustrated in the following quickstarts:

You may also, if you wish, generate or import a key encryption key (KEK).

Install tools and connect to Azure

The steps in this article can be completed with the [Azure CLI](#), the [Azure PowerShell Az module](#), or the [Azure portal](#).

Connect to your Azure account

Before using the Azure CLI or Azure PowerShell, you must first connect to your Azure subscription. You do so by [Signing in with Azure CLI](#), [Signing in with Azure Powershell](#), or supplying your credentials to the Azure portal when prompted.

```
az login
```

```
Connect-AzAccount
```

Create a resource group

If you already have a resource group, you can skip to [Create a key vault](#).

A resource group is a logical container into which Azure resources are deployed and managed.

Create a resource group using the `az group create` Azure CLI command, the `New-AzResourceGroup` Azure PowerShell command, or from the [Azure portal](#).

Azure CLI

```
az group create --name "myResourceGroup" --location eastus
```

Azure PowerShell

```
New-AzResourceGroup -Name "myResourceGroup" -Location "EastUS"
```

Create a key vault

If you already have a key vault, you can skip to [Set key vault advanced access policies](#).

Create a key vault using the [az keyvault create](#) Azure CLI command, the [New-AzKeyvault](#) Azure Powershell command, the [Azure portal](#), or a [Resource Manager template](#).

WARNING

To ensure that encryption secrets don't cross regional boundaries, Azure Disk Encryption requires the Key Vault and the VMs to be co-located in the same region. Create and use a Key Vault that is in the same region as the VMs to be encrypted.

Each Key Vault must have a unique name. Replace with the name of your key vault in the following examples.

Azure CLI

When creating a key vault using Azure CLI, add the "--enabled-for-disk-encryption" flag.

```
az keyvault create --name "<your-unique-keyvault-name>" --resource-group "myResourceGroup" --location "eastus" --enabled-for-disk-encryption
```

Azure PowerShell

When creating a key vault using Azure PowerShell, add the "-EnabledForDiskEncryption" flag.

```
New-AzKeyvault -name "<your-unique-keyvault-name>" -ResourceGroupName "myResourceGroup" -Location "eastus" -EnabledForDiskEncryption
```

Resource Manager template

You can also create a key vault by using the [Resource Manager template](#).

1. On the Azure quickstart template, click **Deploy to Azure**.
2. Select the subscription, resource group, resource group location, Key Vault name, Object ID, legal terms, and agreement, and then click **Purchase**.

Set key vault advanced access policies

The Azure platform needs access to the encryption keys or secrets in your key vault to make them available to the VM for booting and decrypting the volumes.

If you did not enable your key vault for disk encryption, deployment, or template deployment at the time of creation (as demonstrated in the previous step), you must update its advanced access policies.

Azure CLI

Use [az keyvault update](#) to enable disk encryption for the key vault.

- **Enable Key Vault for disk encryption:** Enabled-for-disk-encryption is required.

```
az keyvault update --name "<your-unique-keyvault-name>" --resource-group "MyResourceGroup" --enabled-for-disk-encryption "true"
```

- **Enable Key Vault for deployment, if needed:** Enables the Microsoft.Compute resource provider to retrieve secrets from this key vault when this key vault is referenced in resource creation, for example when creating a virtual machine.

```
az keyvault update --name "<your-unique-keyvault-name>" --resource-group "MyResourceGroup" --enabled-for-deployment "true"
```

- **Enable Key Vault for template deployment, if needed:** Allow Resource Manager to retrieve secrets from the vault.

```
az keyvault update --name "<your-unique-keyvault-name>" --resource-group "MyResourceGroup" --enabled-for-template-deployment "true"
```

Azure PowerShell

Use the key vault PowerShell cmdlet [Set-AzKeyVaultAccessPolicy](#) to enable disk encryption for the key vault.

- **Enable Key Vault for disk encryption:** EnabledForDiskEncryption is required for Azure Disk encryption.

```
Set-AzKeyVaultAccessPolicy -VaultName "<your-unique-keyvault-name>" -ResourceGroupName "MyResourceGroup" -EnabledForDiskEncryption
```

- **Enable Key Vault for deployment, if needed:** Enables the Microsoft.Compute resource provider to retrieve secrets from this key vault when this key vault is referenced in resource creation, for example when creating a virtual machine.

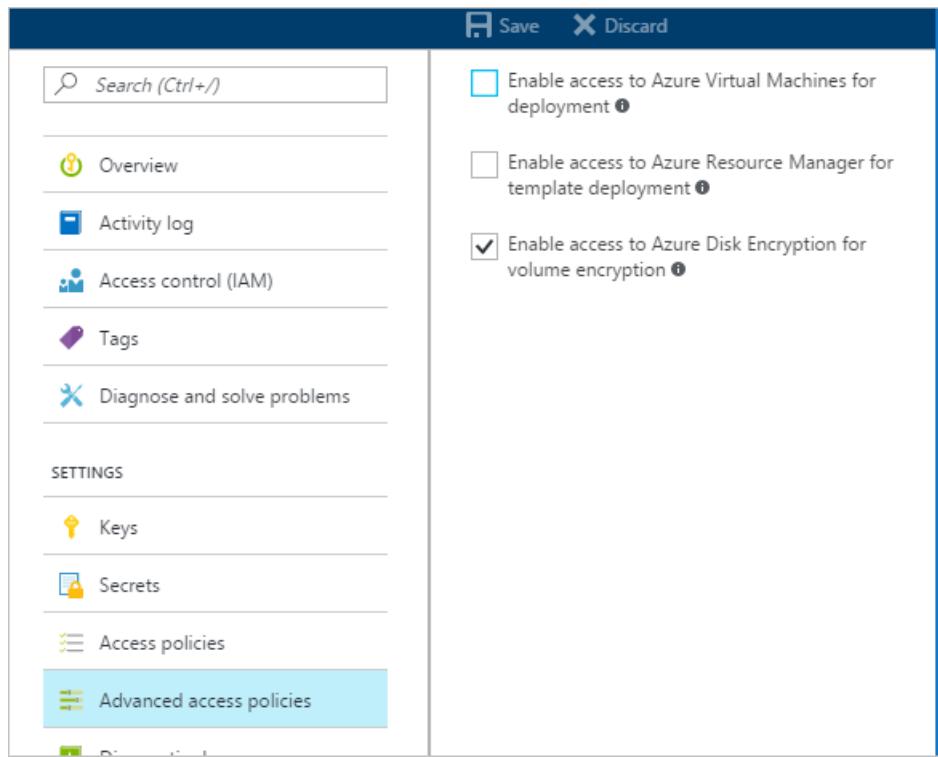
```
Set-AzKeyVaultAccessPolicy -VaultName "<your-unique-keyvault-name>" -ResourceGroupName "MyResourceGroup" -EnabledForDeployment
```

- **Enable Key Vault for template deployment, if needed:** Enables Azure Resource Manager to get secrets from this key vault when this key vault is referenced in a template deployment.

```
Set-AzKeyVaultAccessPolicy -VaultName "<your-unique-keyvault-name>" -ResourceGroupName "MyResourceGroup" -EnabledForTemplateDeployment
```

Azure portal

1. Select your key vault, go to **Access Policies**, and [Click to show advanced access policies](#).
2. Select the box labeled **Enable access to Azure Disk Encryption for volume encryption**.
3. Select **Enable access to Azure Virtual Machines for deployment** and/or **Enable Access to Azure Resource Manager for template deployment**, if needed.
4. Click **Save**.



Set up a key encryption key (KEK)

If you want to use a key encryption key (KEK) for an additional layer of security for encryption keys, add a KEK to your key vault. When a key encryption key is specified, Azure Disk Encryption uses that key to wrap the encryption secrets before writing to Key Vault.

You can generate a new KEK using the Azure CLI `az keyvault key create` command, the Azure PowerShell `Add-AzKeyVaultKey` cmdlet, or the [Azure portal](#). You must generate an RSA key type; Azure Disk Encryption does not yet support using Elliptic Curve keys.

You can instead import a KEK from your on-premises key management HSM. For more information, see [Key Vault Documentation](#).

Your key vault KEK URLs must be versioned. Azure enforces this restriction of versioning. For valid secret and KEK URLs, see the following examples:

- Example of a valid secret URL:
<https://contosovault.vault.azure.net/secrets/EncryptionSecretWithKek/xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
- Example of a valid KEK URL:
<https://contosovault.vault.azure.net/keys/diskencryptionkek/xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

Azure Disk Encryption doesn't support specifying port numbers as part of key vault secrets and KEK URLs. For examples of non-supported and supported key vault URLs, see the following examples:

- Acceptable key vault URL:
<https://contosovault.vault.azure.net/secrets/contososecret/xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
- Unacceptable key vault URL:
<https://contosovault.vault.azure.net:443/secrets/contososecret/xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

Azure CLI

Use the Azure CLI `az keyvault key create` command to generate a new KEK and store it in your key vault.

```
az keyvault key create --name "myKEK" --vault-name "<your-unique-keyvault-name>" --kty RSA-HSM
```

You may instead import a private key using the Azure CLI [az keyvault key import](#) command:

In either case, you will supply the name of your KEK to the Azure CLI [az vm encryption enable](#) --key-encryption-key parameter.

```
az vm encryption enable -g "MyResourceGroup" --name "myVM" --disk-encryption-keyvault "<your-unique-keyvault-name>" --key-encryption-key "myKEK"
```

Azure PowerShell

Use the Azure PowerShell [Add-AzKeyVaultKey](#) cmdlet to generate a new KEK and store it in your key vault.

```
Add-AzKeyVaultKey -Name "myKEK" -VaultName "<your-unique-keyvault-name>" -Destination "HSM"
```

You may instead import a private key using the Azure PowerShell [az keyvault key import](#) command.

In either case, you will supply the ID of your KEK key Vault and the URL of your KEK to the Azure PowerShell [Set-AzVMDiskEncryptionExtension](#) -KeyEncryptionKeyVaultId and -KeyEncryptionKeyUrl parameters. Note that this example assumes that you are using the same key vault for both the disk encryption key and the KEK.

```
$KeyVault = Get-AzKeyVault -VaultName "<your-unique-keyvault-name>" -ResourceGroupName "myResourceGroup"  
$KEK = Get-AzKeyVaultKey -VaultName "<your-unique-keyvault-name>" -Name "myKEK"  
  
Set-AzVMDiskEncryptionExtension -ResourceGroupName MyResourceGroup -VMName "MyVM" -DiskEncryptionKeyVaultUrl  
$KeyVault.VaultUri -DiskEncryptionKeyVaultId $KeyVault.ResourceId -KeyEncryptionKeyVaultId  
$KeyVault.ResourceId -KeyEncryptionKeyUrl $KEK.Id -SkipVmBackup -VolumeType All
```

Next steps

- [Azure Disk Encryption overview](#)
- [Encrypt a virtual machine scale sets using the Azure CLI](#)
- [Encrypt a virtual machine scale sets using the Azure PowerShell](#)

Working with large virtual machine scale sets

1/19/2020 • 5 minutes to read • [Edit Online](#)

You can now create Azure [virtual machine scale sets](#) with a capacity of up to 1,000 VMs. In this document, a *large virtual machine scale set* is defined as a scale set capable of scaling to greater than 100 VMs. This capability is set by a scale set property (*singlePlacementGroup=False*).

Certain aspects of large scale sets, such as load balancing and fault domains behave differently to a standard scale set. This document explains the characteristics of large scale sets, and describes what you need to know to successfully use them in your applications.

A common approach for deploying cloud infrastructure at large scale is to create a set of *scale units*, for example by creating multiple VMs scale sets across multiple VNets and storage accounts. This approach provides easier management compared to single VMs, and multiple scale units are useful for many applications, particularly those that require other stackable components like multiple virtual networks and endpoints. If your application requires a single large cluster however, it can be more straightforward to deploy a single scale set of up to 1,000 VMs.

Example scenarios include centralized big data deployments, or compute grids requiring simple management of a large pool of worker nodes. Combined with virtual machine scale set [attached data disks](#), large scale sets enable you to deploy a scalable infrastructure consisting of thousands of vCPUs and petabytes of storage, as a single operation.

Placement groups

What makes a *large scale set* special is not the number of VMs, but the number of *placement groups* it contains. A placement group is a construct similar to an Azure availability set, with its own fault domains and upgrade domains. By default, a scale set consists of a single placement group with a maximum size of 100 VMs. If a scale set property called *singlePlacementGroup* is set to *false*, the scale set can be composed of multiple placement groups and has a range of 0-1,000 VMs. When set to the default value of *true*, a scale set is composed of a single placement group, and has a range of 0-100 VMs.

Checklist for using large scale sets

To decide whether your application can make effective use of large scale sets, consider the following requirements:

- If you are planning to deploy large number of VMs, your Compute vCPU quota limits may need to be increased.
- Scale sets created from Azure Marketplace images can scale up to 1,000 VMs.
- Scale sets created from custom images (VM images you create and upload yourself) can currently scale up to 600 VMs.
- Large scale sets require Azure Managed Disks. Scale sets that are not created with Managed Disks require multiple storage accounts (one for every 20 VMs). Large scale sets are designed to work exclusively with Managed Disks to reduce your storage management overhead, and to avoid the risk of running into subscription limits for storage accounts.
- Layer-4 load balancing with scale sets composed of multiple placement groups requires [Azure Load Balancer Standard SKU](#). The Load Balancer Standard SKU provides additional benefits, such as the ability to load balance between multiple scale sets. Standard SKU also requires that the scale set has a Network Security Group associated with it, otherwise NAT pools don't work correctly. If you need to use the Azure Load Balancer Basic SKU, make sure the scale set is configured to use a single placement group, which is the default setting.
- Layer-7 load balancing with the Azure Application Gateway is supported for all scale sets.

- A scale set is defined with a single subnet - make sure your subnet has an address space large enough for all the VMs you need. By default a scale set overprovisions (creates extra VMs at deployment time or when scaling out, which you are not charged for) to improve deployment reliability and performance. Allow for an address space 20% greater than the number of VMs you plan to scale to.
- Fault domains and upgrade domains are only consistent within a placement group. This architecture does not change the overall availability of a scale set, as VMs are evenly distributed across distinct physical hardware, but it does mean that if you need to guarantee two VMs are on different hardware, make sure they are in different fault domains in the same placement group. Please refer to this link [Availability options](#).
- Fault domain and placement group ID are shown in the *instance view* of a scale set VM. You can view the instance view of a scale set VM in the [Azure Resource Explorer](#).

Creating a large scale set

When you create a scale set in the Azure portal, just specify the *Instance count* value of up to 1,000. If it is more than 100 instances, *Enable scaling beyond 100 instances* will be set to Yes, which will allow it to scale to multiple placement groups.

INSTANCES

i You have specified an instance count greater than 100. This requires disk type to be Managed and scaling beyond 100 instances to be enabled.

* Instance count i	<input style="width: 150px; border: 1px solid #ccc; border-radius: 4px; padding: 2px 5px;" type="text" value="500"/> ✓
* Instance size (View full pricing details) i	
<input style="border: 1px solid #0078d4; border-radius: 4px; padding: 2px 10px; color: inherit; background-color: inherit; font-size: inherit; font-weight: inherit; font-family: inherit; width: 150px;" type="button" value="D1_v2 (1 vCPU, 3.5 GB)"/> ▼	
Deploy as low priority i	
<input checked="" style="border: 1px solid #ccc; border-radius: 4px; padding: 2px 10px; width: 40px;" type="button" value="No"/> <input style="border: 1px solid #ccc; border-radius: 4px; padding: 2px 10px; width: 40px;" type="button" value="Yes"/>	
Use managed disks i	
<input style="border: 1px solid #ccc; border-radius: 4px; padding: 2px 10px; width: 40px;" type="button" value="No"/> <input style="border: 1px solid #ccc; border-radius: 4px; padding: 2px 10px; width: 40px;" type="button" value="Yes"/>	
- Hide advanced settings	
Enable scaling beyond 100 instances i	
<input style="border: 1px solid #ccc; border-radius: 4px; padding: 2px 10px; width: 40px;" type="button" value="No"/> <input style="border: 1px solid #ccc; border-radius: 4px; padding: 2px 10px; width: 40px;" type="button" value="Yes"/>	

You can create a large virtual machine scale set using the [Azure CLI](#) `az vmss create` command. This command sets intelligent defaults such as subnet size based on the *instance-count* argument:

```
az group create -l southcentralus -n biginfra
az vmss create -g biginfra -n bigvmss --image ubuntults --instance-count 1000
```

The `vmss create` command defaults certain configuration values if you do not specify them. To see the available options that you can override, try:

```
az vmss create --help
```

If you are creating a large scale set by composing an Azure Resource Manager template, make sure the template creates a scale set based on Azure Managed Disks. You can set the *singlePlacementGroup* property to *false* in the *properties* section of the *Microsoft.Compute/virtualMachineScaleSets* resource. The following JSON fragment shows the beginning of a scale set template, including the 1,000 VM capacity and the "*singlePlacementGroup*" : *false* setting:

```
{  
  "type": "Microsoft.Compute/virtualMachineScaleSets",  
  "location": "australiaeast",  
  "name": "bigvmss",  
  "sku": {  
    "name": "Standard_DS1_v2",  
    "tier": "Standard",  
    "capacity": 1000  
  },  
  "properties": {  
    "singlePlacementGroup": false,  
    "upgradePolicy": {  
      "mode": "Automatic"  
    }  
  }  
}
```

For a complete example of a large scale set template, refer to <https://github.com/gbowerman/azure-myriad/blob/master/bigtest/bigbottle.json>.

Converting an existing scale set to span multiple placement groups

To make an existing virtual machine scale set capable of scaling to more than 100 VMs, you need to change the *singlePlacementGroup* property to *false* in the scale set model. You can test changing this property with the [Azure Resource Explorer](#). Find an existing scale set, select *Edit* and change the *singlePlacementGroup* property. If you do not see this property, you may be viewing the scale set with an older version of the Microsoft.Compute API.

NOTE

You can change a scale set from supporting a single placement group only (the default behavior) to a supporting multiple placement groups, but you cannot convert the other way around. Therefore make sure you understand the properties of large scale sets before converting.

Convert a scale set template to a managed disk scale set template

1/19/2020 • 4 minutes to read • [Edit Online](#)

Customers with a Resource Manager template for creating a scale set not using managed disk may wish to modify it to use managed disk. This article shows how to use managed disks, using as an example a pull request from the [Azure Quickstart Templates](#), a community-driven repo for sample Resource Manager templates. The full pull request can be seen here: <https://github.com/Azure/azure-quickstart-templates/pull/2998>, and the relevant parts of the diff are below, along with explanations:

Making the OS disks managed

In the following diff, several variables related to storage account and disk properties are removed. Storage account type is no longer necessary (Standard_LRS is the default), but you could specify it if desired. Only Standard_LRS and Premium_LRS are supported with managed disk. New storage account suffix, unique string array, and sa count were used in the old template to generate storage account names. These variables are no longer necessary in the new template because managed disk automatically creates storage accounts on the customer's behalf. Similarly, vhd container name and OS disk name are no longer necessary because managed disk automatically names the underlying storage blob containers and disks.

```
"variables": {
-   "storageAccountType": "Standard_LRS",
-   "namingInfix": "[toLowerCase(concat(parameters('vmssName'), uniqueString(resourceGroup().id), 0, 9))]",
-   "longNamingInfix": "[toLowerCase(parameters('vmssName'))]",
-   "newStorageAccountSuffix": "[concat(variables('namingInfix'), 'sa')]",
-   "uniqueStringArray": [
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '0')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '1')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '2')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '3')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '4')))]"
-   ],
-   "saCount": "[length(variables('uniqueStringArray'))]",
-   "vhdContainerName": "[concat(variables('namingInfix'), 'vhds')]",
-   "osDiskName": "[concat(variables('namingInfix'), 'osdisk')]",
-   "addressPrefix": "10.0.0.0/16",
-   "subnetPrefix": "10.0.0.0/24",
-   "virtualNetworkName": "[concat(variables('namingInfix'), 'vnet')]"}
```

In the following diff, you compute API is updated to version 2016-04-30-preview, which is the earliest required version for managed disk support with scale sets. You could use unmanaged disks in the new API version with the old syntax if desired. If you only update the compute API version and don't change anything else, the template should continue to work as before.

```

@@ -86,7 +74,7 @@
        "version": "latest"
    },
    "imageReference": "[variables('osType')]",
-   "computeApiVersion": "2016-03-30",
+   "computeApiVersion": "2016-04-30-preview",
    "networkApiVersion": "2016-03-30",
    "storageApiVersion": "2015-06-15"
},

```

In the following diff, the storage account resource is removed from the resources array completely. The resource is no longer needed as managed disk creates them automatically.

```

@@ -113,19 +101,6 @@
        }
    },
- {
-     "type": "Microsoft.Storage/storageAccounts",
-     "name": "[concat(variables('uniqueStringArray')[copyIndex()], variables('newStorageAccountSuffix'))]",
-     "location": "[resourceGroup().location]",
-     "apiVersion": "[variables('storageApiVersion')]",
-     "copy": {
-         "name": "storageLoop",
-         "count": "[variables('saCount')]"
-     },
-     "properties": {
-         "accountType": "[variables('storageAccountType')]"
-     }
- },
{
    "type": "Microsoft.Network/publicIPAddresses",
    "name": "[variables('publicIPAddressName')]",
    "location": "[resourceGroup().location]",

```

In the following diff, we can see that we are removing the depends on clause referring from the scale set to the loop that was creating storage accounts. In the old template, this was ensuring that the storage accounts were created before the scale set began creation, but this clause is no longer necessary with managed disk. The vhd containers property is also removed, along with the OS disk name property as these properties are automatically handled under the hood by managed disk. You could add `"managedDisk": { "storageAccountType": "Premium_LRS" }` in the "osDisk" configuration if you wanted premium OS disks. Only VMs with an uppercase or lowercase 's' in the VM sku can use premium disks.

```

@@ -183,7 +158,6 @@
    "location": "[resourceGroup().location]",
    "apiVersion": "[variables('computeApiVersion')]",
    "dependsOn": [
-     "storageLoop",
      "[concat('Microsoft.Network/loadBalancers/', variables('loadBalancerName'))]",
      "[concat('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]"
    ],
@@ -200,16 +174,8 @@
    "virtualMachineProfile": {
      "storageProfile": {
        "osDisk": {
-         "vhdContainers": [
-           "[concat('https://', variables('uniqueStringArray')[0], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[1], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[2], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[3], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[4], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]"
-         ],
-         "name": "[variables('osDiskName')]",
-       },
-       "imageReference": "[variables('imageReference')]"
      },
    }

```

There is no explicit property in the scale set configuration for whether to use managed or unmanaged disk. The scale set knows which to use based on the properties that are present in the storage profile. Thus, it is important when modifying the template to ensure that the right properties are in the storage profile of the scale set.

Data disks

With the changes above, the scale set uses managed disks for the OS disk, but what about data disks? To add data disks, add the "dataDisks" property under "storageProfile" at the same level as "osDisk". The value of the property is a JSON list of objects, each of which has properties "lun" (which must be unique per data disk on a VM), "createOption" ("empty" is currently the only supported option), and "diskSizeGB" (the size of the disk in gigabytes; must be greater than 0 and less than 1024) as in the following example:

```

"dataDisks": [
  {
    "lun": "1",
    "createOption": "empty",
    "diskSizeGB": "1023"
  }
]

```

If you specify `n` disks in this array, each VM in the scale set gets `n` data disks. Do note, however, that these data disks are raw devices. They are not formatted. It is up to the customer to attach, partition, and format the disks before using them. Optionally, you could also specify `"managedDisk": { "storageAccountType": "Premium_LRS" }` in each data disk object to specify that it should be a premium data disk. Only VMs with an uppercase or lowercase 's' in the VM sku can use premium disks.

To learn more about using data disks with scale sets, see [this article](#).

Next steps

For example Resource Manager templates using scale sets, search for "vmss" in the [Azure Quickstart Templates GitHub repo](#).

For general information, check out the [main landing page for scale sets](#).

Preview: Azure Spot VMs for virtual machine scale sets

2/12/2020 • 6 minutes to read • [Edit Online](#)

Using Azure Spot on scale sets allows you to take advantage of our unused capacity at a significant cost savings. At any point in time when Azure needs the capacity back, the Azure infrastructure will evict Spot instances. Therefore, Spot instances are great for workloads that can handle interruptions like batch processing jobs, dev/test environments, large compute workloads, and more.

The amount of available capacity can vary based on size, region, time of day, and more. When deploying Spot instances on scale sets, Azure will allocate the instance only if there is capacity available, but there is no SLA for these instances. A Spot scale set is deployed in a single fault domain and offers no high availability guarantees.

IMPORTANT

Spot instances are currently in public preview. This preview version is not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

Pricing

Pricing for Spot instances is variable, based on region and SKU. For more information, see pricing for [Linux](#) and [Windows](#).

With variable pricing, you have option to set a max price, in US dollars (USD), using up to 5 decimal places. For example, the value `0.98765` would be a max price of \$0.98765 USD per hour. If you set the max price to be `-1`, the instance won't be evicted based on price. The price for the instance will be the current price for Spot or the price for a standard instance, whichever is less, as long as there is capacity and quota available.

Eviction policy

When creating Spot scale sets, you can set the eviction policy to *Deallocate* (default) or *Delete*.

The *Deallocate* policy moves your evicted instances to the stopped-deallocated state allowing you to redeploy evicted instances. However, there is no guarantee that the allocation will succeed. The deallocated VMs will count against your scale set instance quota and you will be charged for your underlying disks.

If you would like your instances in your Spot scale set to be deleted when they are evicted, you can set the eviction policy to *delete*. With the eviction policy set to delete, you can create new VMs by increasing the scale set instance count property. The evicted VMs are deleted together with their underlying disks, and therefore you will not be charged for the storage. You can also use the auto-scaling feature of scale sets to automatically try and compensate for evicted VMs, however, there is no guarantee that the allocation will succeed. It is recommended you only use the auto-scale feature on Spot scale sets when you set the eviction policy to delete to avoid the cost of your disks and hitting quota limits.

Users can opt-in to receive in-VM notifications through [Azure Scheduled Events](#). This will notify you if your VMs are being evicted and you will have 30 seconds to finish any jobs and perform shutdown tasks prior to the eviction.

Deploying Spot VMs in scale sets

To deploy Spot VMs on scale sets, you can set the new *Priority* flag to *Spot*. All VMs in your scale set will be set to Spot. To create a scale set with Spot VMs, use one of the following methods:

- [Azure portal](#)
- [Azure CLI](#)
- [Azure PowerShell](#)
- [Azure Resource Manager templates](#)

Portal

The process to create a scale set that uses Spot VMs is the same as detailed in the [getting started article](#). When you are deploying a scale set, you can choose to set the Spot flag, and the eviction policy:

Azure Spot instance Yes No

Eviction type Capacity only: evict virtual machine when Azure needs the capacity for pay as you go workloads. Your max price is set to the pay as you go rate. Price or capacity: choose a max price and Azure will evict your virtual machine when the cost of the instance is greater than your max price or when Azure needs the capacity for pay as you go workloads.

Eviction policy Stop / Deallocate Delete

Size * Standard DS1 v2
1 vcpu, 3.5 GiB memory (\$0.05040/hour)
[Change size](#)

Maximum price you want to pay per hour (USD) ✓

[Compare prices in nearby regions](#)

Azure CLI

The process to create a scale set with Spot VMs is the same as detailed in the [getting started article](#). Just add the '--Priority Spot', and add `--max-price`. In this example, we use `-1` for `--max-price` so the instance won't be evicted based on price.

```
az vmss create \
--resource-group myResourceGroup \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--admin-username azureuser \
--generate-ssh-keys \
--priority Spot \
--max-price -1
```

PowerShell

The process to create a scale set with Spot VMs is the same as detailed in the [getting started article](#). Just add '-Priority Spot', and supply a `-max-price` to the [New-AzVmssConfig](#).

```
$vmssConfig = New-AzVmssConfig `  
    -Location "East US 2" `  
    -SkuCapacity 2 `  
    -SkuName "Standard_DS2" `  
    -UpgradePolicyMode Automatic `  
    -Priority "Spot" `  
    --max-price -1
```

Resource Manager templates

The process to create a scale set that uses Spot VMs is the same as detailed in the getting started article for [Linux](#) or [Windows](#).

For Spot template deployments, use `"apiVersion": "2019-03-01"` or later. Add the `priority`, `evictionPolicy` and `billingProfile` properties to the `"virtualMachineProfile":` section in your template:

```
    "priority": "Spot",  
    "evictionPolicy": "Deallocate",  
    "billingProfile": {  
        "maxPrice": -1  
    }
```

To delete the instance after it has been evicted, change the `evictionPolicy` parameter to `Delete`.

FAQ

Q: Once created, is a Spot instance the same as standard instance?

A: Yes, except there is no SLA for Spot VMs and they can be evicted at any time.

Q: What to do when you get evicted, but still need capacity?

A: We recommend you use standard VMs instead of Spot VMs if you need capacity right away.

Q: How is quota managed for Spot?

A: Spot instances and standard instances will have separate quota pools. Spot quota will be shared between VMs and scale-set instances. For more information, see [Azure subscription and service limits, quotas, and constraints](#).

Q: Can I request for additional quota for Spot?

A: Yes, you will be able to submit the request to increase your quota for Spot VMs through the [standard quota request process](#).

Q: Can I convert existing scale sets to Spot scale sets?

A: No, setting the `Spot` flag is only supported at creation time.

Q: If I was using `low` for low-priority scale sets, do I need to start using `Spot` instead?

A: For now, both `low` and `Spot` will work, but you should start transitioning to using `spot`.

Q: Can I create a scale set with both regular VMs and Spot VMs?

A: No, a scale set cannot support more than one priority type.

Q: Can I use autoscale with Spot scale sets?

A: Yes, you can set autoscaling rules on your Spot scale set. If your VMs are evicted, autoscale can try to create new Spot VMs. Remember, you are not guaranteed this capacity though.

Q: Does autoscale work with both eviction policies (deallocate and delete)?

A: It is recommended that you set your eviction policy to delete when using autoscale. This is because deallocated instances are counted against your capacity count on the scale set. When using autoscale, you will likely hit your target instance count quickly due to the deallocated, evicted instances.

Q: What channels support Spot VMs?

A: See the table below for Spot VM availability.

AZURE CHANNELS	AZURE SPOT VMs AVAILABILITY
Enterprise Agreement	Yes
Pay As You Go	Yes
Cloud Service Provider (CSP)	Contact your partner
Benefits	Not available
Sponsored	Not available
Free Trial	Not available

Q: Where can I post questions?

A: You can post and tag your question with `azure-spot` at [Q&A](#).

Next steps

Now that you have created a scale set with Spot VMs, try deploying our [auto scale template using Spot](#).

Check out the [virtual machine scale set pricing page](#) for pricing details.

Manage a virtual machine scale set with the Azure CLI

1/19/2020 • 4 minutes to read • [Edit Online](#)

Throughout the lifecycle of a virtual machine scale set, you may need to run one or more management tasks. Additionally, you may want to create scripts that automate various lifecycle-tasks. This article details some of the common Azure CLI commands that let you perform these tasks.

To complete these management tasks, you need the latest Azure CLI. For information, see [Install the Azure CLI](#). If you need to create a virtual machine scale set, you can [create a scale set with the Azure CLI](#).

View information about a scale set

To view the overall information about a scale set, use [az vmss show](#). The following example gets information about the scale set named *myScaleSet* in the *myResourceGroup* resource group. Enter your own names as follows:

```
az vmss show --resource-group myResourceGroup --name myScaleSet
```

View VMs in a scale set

To view a list of VM instance in a scale set, use [az vmss list-instances](#). The following example lists all VM instances in the scale set named *myScaleSet* in the *myResourceGroup* resource group. Provide your own values for these names:

```
az vmss list-instances \
--resource-group myResourceGroup \
--name myScaleSet \
--output table
```

To view additional information about a specific VM instance, add the `--instance-id` parameter to [az vmss get-instance-view](#) and specify an instance to view. The following example views information about VM instance 0 in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Enter your own names as follows:

```
az vmss get-instance-view \
--resource-group myResourceGroup \
--name myScaleSet \
--instance-id 0
```

List connection information for VMs

To connect to the VMs in a scale set, you SSH or RDP to an assigned public IP address and port number. By default, network address translation (NAT) rules are added to the Azure load balancer that forwards remote connection traffic to each VM. To list the address and ports to connect to VM instances in a scale set, use [az vmss list-instance-connection-info](#). The following example lists connection information for VM instances in the scale set named *myScaleSet* and in the *myResourceGroup* resource group. Provide your own values for these names:

```
az vmss list-instance-connection-info \
--resource-group myResourceGroup \
--name myScaleSet
```

Change the capacity of a scale set

The preceding commands showed information about your scale set and the VM instances. To increase or decrease the number of instances in the scale set, you can change the capacity. The scale set creates or removes the required number of VMs, then configures the VMs to receive application traffic.

To see the number of instances you currently have in a scale set, use [az vmss show](#) and query on `sku.capacity`:

```
az vmss show \
--resource-group myResourceGroup \
--name myScaleSet \
--query [sku.capacity] \
--output table
```

You can then manually increase or decrease the number of virtual machines in the scale set with [az vmss scale](#). The following example sets the number of VMs in your scale set to 5:

```
az vmss scale \
--resource-group myResourceGroup \
--name myScaleSet \
--new-capacity 5
```

It takes a few minutes to update the capacity of your scale set. If you decrease the capacity of a scale set, the VMs with the highest instance IDs are removed first.

Stop and start VMs in a scale set

To stop one or more VMs in a scale set, use [az vmss stop](#). The `--instance-ids` parameter allows you to specify one or more VMs to stop. If you do not specify an instance ID, all VMs in the scale set are stopped. To stop multiple VMs, separate each instance ID with a space.

The following example stops instance 0 in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
az vmss stop --resource-group myResourceGroup --name myScaleSet --instance-ids 0
```

Stopped VMs remain allocated and continue to incur compute charges. If you instead wish the VMs to be deallocated and only incur storage charges, use [az vmss deallocate](#). To deallocate multiple VMs, separate each instance ID with a space. The following example stops and deallocates instance 0 in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
az vmss deallocate --resource-group myResourceGroup --name myScaleSet --instance-ids 0
```

Start VMs in a scale set

To start one or more VMs in a scale set, use [az vmss start](#). The `--instance-ids` parameter allows you to specify one or more VMs to start. If you do not specify an instance ID, all VMs in the scale set are started. To start multiple VMs, separate each instance ID with a space.

The following example starts instance *0* in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
az vmss start --resource-group myResourceGroup --name myScaleSet --instance-ids 0
```

Restart VMs in a scale set

To restart one or more VMs in a scale set, use [az vmss restart](#). The `--instance-ids` parameter allows you to specify one or more VMs to restart. If you do not specify an instance ID, all VMs in the scale set are restarted. To restart multiple VMs, separate each instance ID with a space.

The following example restarts instance *0* in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
az vmss restart --resource-group myResourceGroup --name myScaleSet --instance-ids 0
```

Remove VMs from a scale set

To remove one or more VMs in a scale set, use [az vmss delete-instances](#). The `--instance-ids` parameter allows you to specify one or more VMs to remove. If you specify `*` for the instance ID, all VMs in the scale set are removed. To remove multiple VMs, separate each instance ID with a space.

The following example removes instance *0* in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
az vmss delete-instances --resource-group myResourceGroup --name myScaleSet --instance-ids 0
```

Next steps

Other common tasks for scale sets include how to [deploy an application](#), and [upgrade VM instances](#). You can also use Azure CLI to [configure auto-scale rules](#).

Manage a virtual machine scale set with Azure PowerShell

1/19/2020 • 4 minutes to read • [Edit Online](#)

Throughout the lifecycle of a virtual machine scale set, you may need to run one or more management tasks. Additionally, you may want to create scripts that automate various lifecycle-tasks. This article details some of the common Azure PowerShell cmdlets that let you perform these tasks.

If you need to create a virtual machine scale set, you can [create a scale set with Azure PowerShell](#).

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

View information about a scale set

To view the overall information about a scale set, use [Get-AzVmss](#). The following example gets information about the scale set named *myScaleSet* in the *myResourceGroup* resource group. Enter your own names as follows:

```
Get-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet"
```

View VMs in a scale set

To view a list of VM instance in a scale set, use [Get-AzVmssVM](#). The following example lists all VM instances in the scale set named *myScaleSet* and in the *myResourceGroup* resource group. Provide your own values for these names:

```
Get-AzVmssVM -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet"
```

To view additional information about a specific VM instance, add the `-InstanceId` parameter to [Get-AzVmssVM](#) and specify an instance to view. The following example views information about VM instance 0 in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Enter your own names as follows:

```
Get-AzVmssVM -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "0"
```

Change the capacity of a scale set

The preceding commands showed information about your scale set and the VM instances. To increase or decrease the number of instances in the scale set, you can change the capacity. The scale set automatically creates or removes the required number of VMs, then configures the VMs to receive application traffic.

First, create a scale set object with [Get-AzVmss](#), then specify a new value for `sku.capacity`. To apply the capacity change, use [Update-AzVmss](#). The following example updates *myScaleSet* in the *myResourceGroup* resource group to a capacity of 5 instances. Provide your own values as follows:

```
# Get current scale set
$vmss = Get-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet"

# Set and update the capacity of your scale set
$vmss.sku.capacity = 5
Update-AzVmss -ResourceGroupName "myResourceGroup" -Name "myScaleSet" -VirtualMachineScaleSet $vmss
```

If takes a few minutes to update the capacity of your scale set. If you decrease the capacity of a scale set, the VMs with the highest instance IDs are removed first.

Stop and start VMs in a scale set

To stop one or more VMs in a scale set, use [Stop-AzVmss](#). The `-InstanceId` parameter allows you to specify one or more VMs to stop. If you do not specify an instance ID, all VMs in the scale set are stopped. To stop multiple VMs, separate each instance ID with a comma.

The following example stops instance 0 in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
Stop-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "0"
```

By default, stopped VMs are deallocated and do not incur compute charges. If you wish the VM to remain in a provisioned state when stopped, add the `-StayProvisioned` parameter to the preceding command. Stopped VMs that remain provisioned incur regular compute charges.

Start VMs in a scale set

To start one or more VMs in a scale set, use [Start-AzVmss](#). The `-InstanceId` parameter allows you to specify one or more VMs to start. If you do not specify an instance ID, all VMs in the scale set are started. To start multiple VMs, separate each instance ID with a comma.

The following example starts instance 0 in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
Start-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "0"
```

Restart VMs in a scale set

To restart one or more VMs in a scale set, use [Restart-AzVmss](#). The `-InstanceId` parameter allows you to specify one or more VMs to restart. If you do not specify an instance ID, all VMs in the scale set are restarted. To restart multiple VMs, separate each instance ID with a comma.

The following example restarts instance 0 in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
Restart-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "0"
```

Remove VMs from a scale set

To remove one or more VMs in a scale set, use [Remove-AzVmss](#). The `-InstanceId` parameter allows you to specify one or more VMs to remove. If you do not specify an instance ID, all VMs in the scale set are removed. To remove multiple VMs, separate each instance ID with a comma.

The following example removes instance *0* in the scale set named *myScaleSet* and the *myResourceGroup* resource group. Provide your own values as follows:

```
Remove-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId "0"
```

Next steps

Other common tasks for scale sets include how to [deploy an application](#), and [upgrade VM instances](#). You can also use Azure PowerShell to [configure auto-scale rules](#).

Modify a virtual machine scale set

1/19/2020 • 12 minutes to read • [Edit Online](#)

Throughout the lifecycle of your applications, you may need to modify or update your virtual machine scale set. These updates may include how to update the configuration of the scale set, or change the application configuration. This article describes how to modify an existing scale set with the REST APIs, Azure PowerShell, or Azure CLI.

Fundamental concepts

The scale set model

A scale set has a "scale set model" that captures the *desired* state of the scale set as a whole. To query the model for a scale set, you can use the

- REST API with [compute/virtualmachinescalesets/get](#) as follows:

```
GET  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet?api-version={apiVersion}
```

- Azure PowerShell with [Get-AzVmss](#):

```
Get-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet"
```

- Azure CLI with [az vmss show](#):

```
az vmss show --resource-group myResourceGroup --name myScaleSet
```

- You can also use [resources.azure.com](#) or the language-specific [Azure SDKs](#).

The exact presentation of the output depends on the options you provide to the command. The following example shows condensed sample output from the Azure CLI:

```
az vmss show --resource-group myResourceGroup --name myScaleSet  
{  
  "location": "westus",  
  "overprovision": true,  
  "plan": null,  
  "singlePlacementGroup": true,  
  "sku": {  
    "additionalProperties": {},  
    "capacity": 1,  
    "name": "Standard_D2_v2",  
    "tier": "Standard"  
  },  
}
```

These properties apply to the scale set as a whole.

The scale set instance view

A scale set also has a "scale set instance view" that captures the current *runtime* state of the scale set as a whole.

To query the instance view for a scale set, you can use:

- REST API with [compute/virtualmachinescalesets/getinstanceview](#) as follows:

```
GET  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/instanceView?api-version={apiVersion}
```

- Azure PowerShell with [Get-AzVmss](#):

```
Get-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceView
```

- Azure CLI with [az vmss get-instance-view](#):

```
az vmss get-instance-view --resource-group myResourceGroup --name myScaleSet
```

- You can also use [resources.azure.com](#) or the language-specific [Azure SDKs](#)

The exact presentation of the output depends on the options you provide to the command. The following example shows condensed sample output from the Azure CLI:

```
$ az vmss get-instance-view --resource-group myResourceGroup --name myScaleSet  
{  
    "statuses": [  
        {  
            "additionalProperties": {},  
            "code": "ProvisioningState/succeeded",  
            "displayStatus": "Provisioning succeeded",  
            "level": "Info",  
            "message": null,  
            "time": "{time}"  
        }  
    ],  
    "virtualMachine": {  
        "additionalProperties": {},  
        "statusesSummary": [  
            {  
                "additionalProperties": {},  
                "code": "ProvisioningState/succeeded",  
                "count": 1  
            }  
        ]  
    }  
}
```

These properties provide a summary of the current runtime state of the VMs in the scale set, such as the status of extensions applied to the scale set.

The scale set VM model view

Similar to how a scale set has a model view, each VM instance in the scale set has its own model view. To query the model view for a particular VM instance in a scale set, you can use:

- REST API with [compute/virtualmachinescalesetvms/get](#) as follows:

```
GET  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/virtualmachines/instanceId?api-version={apiVersion}
```

- Azure PowerShell with [Get-AzVmssVm](#):

```
Get-AzVmssVm -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId instanceId
```

- Azure CLI with [az vmss show](#):

```
az vmss show --resource-group myResourceGroup --name myScaleSet --instance-id instanceId
```

- You can also use [resources.azure.com](#) or the [Azure SDKs](#).

The exact presentation of the output depends on the options you provide to the command. The following example shows condensed sample output from the Azure CLI:

```
$ az vmss show --resource-group myResourceGroup --name myScaleSet  
{  
  "location": "westus",  
  "name": "{name}",  
  "sku": {  
    "name": "Standard_D2_V2",  
    "tier": "Standard"  
  },  
}
```

These properties describe the configuration of a VM instance within a scale set, not the configuration of the scale set as a whole. For example, the scale set model has [overprovision](#) as a property, while the model for a VM instance within a scale set does not. This difference is because overprovisioning is a property for the scale set as a whole, not individual VM instances in the scale set (for more information about overprovisioning, see [Design considerations for scale sets](#)).

The scale set VM instance view

Similar to how a scale set has an instance view, each VM instance in the scale set has its own instance view. To query the instance view for a particular VM instance within a scale set, you can use:

- REST API with [compute/virtualmachinescalesetvms/getinstanceview](#) as follows:

```
GET  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/virtualmachines/instanceId/instanceView?api-version={apiVersion}
```

- Azure PowerShell with [Get-AzVmssVm](#):

```
Get-AzVmssVm -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -InstanceId instanceId  
-InstanceView
```

- Azure CLI with [az vmss get-instance-view](#)

```
az vmss get-instance-view --resource-group myResourceGroup --name myScaleSet --instance-id instanceId
```

- You can also use resources.azure.com or the Azure SDKs

The exact presentation of the output depends on the options you provide to the command. The following example shows condensed sample output from the Azure CLI:

```
$ az vmss get-instance-view --resource-group myResourceGroup --name myScaleSet --instance-id instanceId
{
  "additionalProperties": {
    "osName": "ubuntu",
    "osVersion": "16.04"
  },
  "disks": [
    {
      "name": "{name}",
      "statuses": [
        {
          "additionalProperties": {},
          "code": "ProvisioningState/succeeded",
          "displayStatus": "Provisioning succeeded",
          "time": "{time}"
        }
      ]
    }
  ],
  "statuses": [
    {
      "additionalProperties": {},
      "code": "ProvisioningState/succeeded",
      "displayStatus": "Provisioning succeeded",
      "time": "{time}"
    },
    {
      "additionalProperties": {},
      "code": "PowerState/running",
      "displayStatus": "VM running"
    }
  ],
  "vmAgent": {
    "statuses": [
      {
        "additionalProperties": {},
        "code": "ProvisioningState/succeeded",
        "displayStatus": "Ready",
        "level": "Info",
        "message": "Guest Agent is running",
        "time": "{time}"
      }
    ],
    "vmAgentVersion": "{version}"
  }
}
```

These properties describe the current runtime state of a VM instance within a scale set, which includes any extensions applied to the scale set.

How to update global scale set properties

To update a global scale set property, you must update the property in the scale set model. You can do this update via:

- REST API with [compute/virtualmachinescalesets/createorupdate](#) as follows:

```
PUT  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet?api-version={apiVersion}
```

- You can deploy a Resource Manager template with the properties from the REST API to update global scale set properties.
- Azure PowerShell with [Update-AzVmss](#):

```
Update-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -  
VirtualMachineScaleSet {scaleSetConfigPowershellObject}
```

- Azure CLI with [az vmss update](#):

- To modify a property:

```
az vmss update --set {propertyPath}={value}
```

- To add an object to a list property in a scale set:

```
az vmss update --add {propertyPath} {JSONObjectToAdd}
```

- To remove an object from a list property in a scale set:

```
az vmss update --remove {propertyPath} {indexToRemove}
```

- If you previously deployed the scale set with the `az vmss create` command, you can run the `az vmss create` command again to update the scale set. Make sure that all properties in the `az vmss create` command are the same as before, except for the properties that you wish to modify.

- You can also use [resources.azure.com](#) or the [Azure SDKs](#).

Once the scale set model is updated, the new configuration applies to any new VMs created in the scale set. However, the models for the existing VMs in the scale set must still be brought up-to-date with the latest overall scale set model. In the model for each VM is a boolean property called `latestModelApplied` that indicates whether or not the VM is up-to-date with the latest overall scale set model (`true` means the VM is up-to-date with the latest model).

How to bring VMs up-to-date with the latest scale set model

Scale sets have an "upgrade policy" that determine how VMs are brought up-to-date with the latest scale set model. The three modes for the upgrade policy are:

- **Automatic** - In this mode, the scale set makes no guarantees about the order of VMs being brought down. The scale set may take down all VMs at the same time.
- **Rolling** - In this mode, the scale set rolls out the update in batches with an optional pause time between batches.
- **Manual** - In this mode, when you update the scale set model, nothing happens to existing VMs.

To update existing VMs, you must do a "manual upgrade" of each existing VM. You can do this manual upgrade with:

- REST API with [compute/virtualmachinescalesets/updateinstances](#) as follows:

```
POST  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/manualupgrade?api-version={apiVersion}
```

- Azure PowerShell with [Update-AzVmssInstance](#):

```
Update-AzVmssInstance -ResourceGroupName "myResourceGroup" -VMSScaleSetName "myScaleSet" -InstanceId
```

- Azure CLI with [az vmss update-instances](#)

```
az vmss update-instances --resource-group myResourceGroup --name myScaleSet --instance-ids {instanceIds}
```

- You can also use the language-specific [Azure SDKs](#).

NOTE

Service Fabric clusters can only use *Automatic* mode, but the update is handled differently. For more information, see [Service Fabric application upgrades](#).

There is one type of modification to global scale set properties that does not follow the upgrade policy. Changes to the scale set OS Profile (such as admin username and password) can only be changed in API version 2017-12-01 or later. These changes only apply to VMs created after the change in the scale set model. To bring existing VMs up-to-date, you must do a "reimage" of each existing VM. You can do this reimage via:

- REST API with [compute/virtualmachinescalesets/reimage](#) as follows:

```
POST  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/reimage?api-version={apiVersion}
```

- Azure PowerShell with [Set-AzVmssVm](#):

```
Set-AzVmssVM -ResourceGroupName "myResourceGroup" -VMSScaleSetName "myScaleSet" -InstanceId instanceId  
-Reimage
```

- Azure CLI with [az vmss reimage](#):

```
az vmss reimage --resource-group myResourceGroup --name myScaleSet --instance-id instanceId
```

- You can also use the language-specific [Azure SDKs](#).

Properties with restrictions on modification

Create-time properties

Some properties can only be set when you create the scale set. These properties include:

- Availability Zones
- Image reference publisher
- Image reference offer

- Managed OS disk storage account type

Properties that can only be changed based on the current value

Some properties may be changed, with exceptions depending on the current value. These properties include:

- **singlePlacementGroup** - If singlePlacementGroup is true, it may be modified to false. However, if singlePlacementGroup is false, it **may not** be modified to true.
- **subnet** - The subnet of a scale set may be modified as long as the original subnet and the new subnet are in the same virtual network.

Properties that require deallocation to change

Some properties may only be changed to certain values if the VMs in the scale set are deallocated. These properties include:

- **SKU name** - If the new VM SKU is not supported on the hardware the scale set is currently on, you need to deallocate the VMs in the scale set before you modify the SKU name. For more information, see [how to resize an Azure VM](#).

VM-specific updates

Certain modifications may be applied to specific VMs instead of the global scale set properties. Currently, the only VM-specific update that is supported is to attach/detach data disks to/from VMs in the scale set. This feature is in preview. For more information, see the [preview documentation](#).

Scenarios

Application updates

If an application is deployed to a scale set through extensions, an update to the extension configuration causes the application to update in accordance with the upgrade policy. For instance, if you have a new version of a script to run in a Custom Script Extension, you could update the *fileUris* property to point to the new script. In some cases, you may wish to force an update even though the extension configuration is unchanged (for example, you updated the script without a change to the URL of the script). In these cases, you can modify the *forceUpdateTag* to force an update. The Azure platform does not interpret this property. If you change the value, there is no effect on how the extension runs. A change simply forces the extension to rerun. For more information on the *forceUpdateTag*, see the [REST API documentation for extensions](#). Note that the *forceUpdateTag* can be used with all extensions, not just the custom script extension.

It's also common for applications to be deployed through a custom image. This scenario is covered in the following section.

OS Updates

If you use Azure platform images, you can update the image by modifying the *imageReference* (more information, see the [REST API documentation](#)).

NOTE

With platform images, it is common to specify "latest" for the image reference version. When you create, scale out, and reimagine, VMs are created with the latest available version. However, it **does not** mean that the OS image is automatically updated over time as new image versions are released. A separate feature is currently in preview that provides automatic OS upgrades. For more information, see the [Automatic OS Upgrades documentation](#).

If you use custom images, you can update the image by updating the *imageReference* ID (more information, see the [REST API documentation](#)).

Examples

Update the OS image for your scale set

You may have a scale set that runs an old version of Ubuntu LTS 16.04. You want to update to a newer version of Ubuntu LTS 16.04, such as version `16.04.201801090`. The image reference version property is not part of a list, so you can directly modify these properties with one of the following commands:

- Azure PowerShell with [Update-AzVmss](#) as follows:

```
Update-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -ImageReferenceVersion  
16.04.201801090
```

- Azure CLI with [az vmss update](#):

```
az vmss update --resource-group myResourceGroup --name myScaleSet --set  
virtualMachineProfile.storageProfile.imageReference.version=16.04.201801090
```

Alternatively, you may want to change the image your scale set uses. For example, you may want to update or change a custom image used by your scale set. You can change the image your scale set uses by updating the image reference ID property. The image reference ID property is not part of a list, so you can directly modify this property with one of the following commands:

- Azure PowerShell with [Update-AzVmss](#) as follows:

```
Update-AzVmss `  
-ResourceGroupName "myResourceGroup" `  
-VMScaleSetName "myScaleSet" `  
-ImageReferenceId  
/subscriptions/{subscriptionID}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/images/myNewImage
```

- Azure CLI with [az vmss update](#):

```
az vmss update \  
--resource-group myResourceGroup \  
--name myScaleSet \  
--set  
virtualMachineProfile.storageProfile.imageReference.id=/subscriptions/{subscriptionID}/resourceGroups/  
myResourceGroup/providers/Microsoft.Compute/images/myNewImage
```

Update the load balancer for your scale set

Let's say you have a scale set with an Azure Load Balancer, and you want to replace the Azure Load Balancer with an Azure Application Gateway. The load balancer and Application Gateway properties for a scale set are part of a list, so you can use the commands to remove or add list elements instead of modifying the properties directly:

- Azure Powershell:

```

# Get the current model of the scale set and store it in a local PowerShell object named $vmss
$vmss=Get-AzVmss -ResourceGroupName "myResourceGroup" -Name "myScaleSet"

# Create a local PowerShell object for the new desired IP configuration, which includes the reference
# to the application gateway
$ipconf = New-AzVmssIPConfig "myNic" -ApplicationGatewayBackendAddressPoolsId
/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Network/application
Gateways/{applicationGatewayName}/backendAddressPools/{applicationGatewayBackendAddressPoolName} -
SubnetId
$vmss.VirtualMachineProfile.NetworkProfile.NetworkInterfaceConfigurations[0].IpConfigurations[0].Subne
t.Id -Name
$vmss.VirtualMachineProfile.NetworkProfile.NetworkInterfaceConfigurations[0].IpConfigurations[0].Name

# Replace the existing IP configuration in the local PowerShell object (which contains the references
# to the current Azure Load Balancer) with the new IP configuration
$vmss.VirtualMachineProfile.NetworkProfile.NetworkInterfaceConfigurations[0].IpConfigurations[0] =
$ipconf

# Update the model of the scale set with the new configuration in the local PowerShell object
Update-AzVmss -ResourceGroupName "myResourceGroup" -Name "myScaleSet" -virtualMachineScaleSet $vmss

```

- Azure CLI:

```

# Remove the load balancer backend pool from the scale set model
az vmss update --resource-group myResourceGroup --name myScaleSet --remove
virtualMachineProfile.networkProfile.networkInterfaceConfigurations[0].ipConfigurations[0].loadBalance
rBackendAddressPools 0

# Remove the load balancer backend pool from the scale set model; only necessary if you have NAT pools
# configured on the scale set
az vmss update --resource-group myResourceGroup --name myScaleSet --remove
virtualMachineProfile.networkProfile.networkInterfaceConfigurations[0].ipConfigurations[0].loadBalance
rInboundNatPools 0

# Add the application gateway backend pool to the scale set model
az vmss update --resource-group myResourceGroup --name myScaleSet --add
virtualMachineProfile.networkProfile.networkInterfaceConfigurations[0].ipConfigurations[0].Application
GatewayBackendAddressPools '{"id":'
"/subscriptions/{subscriptionId}/resourceGroups/myResourceGroup/providers/Microsoft.Network/applicatio
nGateways/{applicationGatewayName}/backendAddressPools/{applicationGatewayBackendPoolName}"}'

```

NOTE

These commands assume there is only one IP configuration and load balancer on the scale set. If there are multiple, you may need to use a list index other than 0.

Next steps

You can also perform common management tasks on scale sets with the [Azure CLI](#) or [Azure PowerShell](#).

Networking for Azure virtual machine scale sets

2/21/2020 • 8 minutes to read • [Edit Online](#)

When you deploy an Azure virtual machine scale set through the portal, certain network properties are defaulted, for example an Azure Load Balancer with inbound NAT rules. This article describes how to use some of the more advanced networking features that you can configure with scale sets.

You can configure all of the features covered in this article using Azure Resource Manager templates. Azure CLI and PowerShell examples are also included for selected features.

Accelerated Networking

Azure Accelerated Networking improves network performance by enabling single root I/O virtualization (SR-IOV) to a virtual machine. To learn more about using Accelerated networking, see Accelerated networking for [Windows](#) or [Linux](#) virtual machines. To use accelerated networking with scale sets, set `enableAcceleratedNetworking` to **true** in your scale set's `networkInterfaceConfigurations` settings. For example:

```
"networkProfile": {  
    "networkInterfaceConfigurations": [  
        {  
            "name": "niconfig1",  
            "properties": {  
                "primary": true,  
                "enableAcceleratedNetworking" : true,  
                "ipConfigurations": [  
                    ...  
                ]  
            }  
        }  
    ]  
}
```

Create a scale set that references an existing Azure Load Balancer

When a scale set is created using the Azure portal, a new load balancer is created for most configuration options. If you create a scale set, which needs to reference an existing load balancer, you can do this using the CLI. The following example script creates a load balancer and then creates a scale set, which references it:

```

az network lb create \
    -g lbtest \
    -n mylb \
    --vnet-name myvnet \
    --subnet mysubnet \
    --public-ip-address-allocation Static \
    --backend-pool-name mybackendpool

az vmss create \
    -g lbtest \
    -n myvmss \
    --image Canonical:UbuntuServer:16.04-LTS:latest \
    --admin-username negat \
    --ssh-key-value /home/myuser/.ssh/id_rsa.pub \
    --upgrade-policy-mode Automatic \
    --instance-count 3 \
    --vnet-name myvnet \
    --subnet mysubnet \
    --lb mylb \
    --backend-pool-name mybackendpool

```

NOTE

After the scale set has been created, the backend port cannot be modified for a load balancing rule used by a health probe of the load balancer. To change the port, you can remove the health probe by updating the Azure virtual machine scale set, update the port and then configure the health probe again.

Create a scale set that references an Application Gateway

To create a scale set that uses an application gateway, reference the backend address pool of the application gateway in the ipConfigurations section of your scale set as in this ARM template config:

```

"ipConfigurations": [
    {
        "name": "{config-name}",
        "properties": {
            "subnet": {
                "id": "{subnet-id}"
            },
            "ApplicationGatewayBackendAddressPools": [
                {
                    "id": "/subscriptions/{subscription-id}/resourceGroups/{resource-group-name}/providers/Microsoft.Network/applicationGateways/{gateway-name}/backendAddressPools/{pool-name}"
                }
            ]
        }
    }
]

```

NOTE

Note that the application gateway must be in the same virtual network as the scale set but must be in a different subnet from the scale set.

Configurable DNS Settings

By default, scale sets take on the specific DNS settings of the VNET and subnet they were created in. You can however, configure the DNS settings for a scale set directly.

Creating a scale set with configurable DNS servers

To create a scale set with a custom DNS configuration using the Azure CLI, add the **--dns-servers** argument to the **vmss create** command, followed by space separated server ip addresses. For example:

```
--dns-servers 10.0.0.6 10.0.0.5
```

To configure custom DNS servers in an Azure template, add a `dnsSettings` property to the scale set `networkInterfaceConfigurations` section. For example:

```
"dnsSettings":{  
    "dnsServers":["10.0.0.6", "10.0.0.5"]  
}
```

Creating a scale set with configurable virtual machine domain names

To create a scale set with a custom DNS name for virtual machines using the CLI, add the `--vm-domain-name` argument to the **virtual machine scale set create** command, followed by a string representing the domain name.

To set the domain name in an Azure template, add a `dnsSettings` property to the scale set `networkInterfaceConfigurations` section. For example:

```
"networkProfile": {  
    "networkInterfaceConfigurations": [  
        {  
            "name": "nic1",  
            "properties": {  
                "primary": true,  
                "ipConfigurations": [  
                    {  
                        "name": "ip1",  
                        "properties": {  
                            "subnet": {  
                                "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',  
resourceGroup().name, '/providers/Microsoft.Network/virtualNetworks/', variables('vnetName'),  
'/subnets/subnet1')]"  
                            },  
                            "publicIPAddressConfiguration": {  
                                "name": "publicip",  
                                "properties": {  
                                    "idleTimeoutInMinutes": 10,  
                                    "dnsSettings": {  
                                        "domainNameLabel": "[parameters('vmssDnsName')]"  
                                    }  
                                }  
                            }  
                        }  
                    }  
                ]  
            }  
        }  
    ]  
}
```

The output, for an individual virtual machine dns name would be in the following form:

```
<vm><vmindex>.<specifiedVmssDomainNameLabel>
```

Public IPv4 per virtual machine

In general, Azure scale set virtual machines do not require their own public IP addresses. For most scenarios, it is more economical and secure to associate a public IP address to a load balancer or to an individual virtual machine (aka a jumpbox), which then routes incoming connections to scale set virtual machines as needed (for example, through inbound NAT rules).

However, some scenarios do require scale set virtual machines to have their own public IP addresses. An example is gaming, where a console needs to make a direct connection to a cloud virtual machine, which is doing game physics processing. Another example is where virtual machines need to make external connections to one another across regions in a distributed database.

Creating a scale set with public IP per virtual machine

To create a scale set that assigns a public IP address to each virtual machine with the CLI, add the **--public-ip-per-vm** parameter to the **vmss create** command.

To create a scale set using an Azure template, make sure the API version of the Microsoft.Compute/virtualMachineScaleSets resource is at least **2017-03-30**, and add a **publicIpAddressConfiguration** JSON property to the scale set ipConfigurations section. For example:

```
"publicIpAddressConfiguration": {  
    "name": "pub1",  
    "properties": {  
        "idleTimeoutInMinutes": 15  
    }  
}
```

Example template: [201-vmss-public-ip-linux](#)

Querying the public IP addresses of the virtual machines in a scale set

To list the public IP addresses assigned to scale set virtual machines using the CLI, use the **az vmss list-instance-public-ips** command.

To list scale set public IP addresses using PowerShell, use the **Get-AzPublicIpAddress** command. For example:

```
Get-AzPublicIpAddress -ResourceGroupName myrg -VirtualMachineScaleSetName myvmss
```

You can also query the public IP addresses by referencing the resource ID of the public IP address configuration directly. For example:

```
Get-AzPublicIpAddress -ResourceGroupName myrg -Name myvmsspip
```

You can also display the public IP addresses assigned to the scale set virtual machines by querying the [Azure Resource Explorer](#) or the Azure REST API with version **2017-03-30** or higher.

To query the [Azure Resource Explorer](#):

1. Open [Azure Resource Explorer](#) in a web browser.
2. Expand *subscriptions* on the left side by clicking the + next to it. If you only have one item under *subscriptions*, it may already be expanded.
3. Expand your subscription.
4. Expand your resource group.
5. Expand *providers*.
6. Expand *Microsoft.Compute*.
7. Expand *virtualMachineScaleSets*.
8. Expand your scale set.
9. Click on *publicIPaddresses*.

To query the Azure REST API:

```
GET https://management.azure.com/subscriptions/{your sub ID}/resourceGroups/{RG name}/providers/Microsoft.Compute/virtualMachineScaleSets/{scale set name}/publicIPAddresses?api-version=2017-03-30
```

Example output from the [Azure Resource Explorer](#) and Azure REST API:

```
{
  "value": [
    {
      "name": "pub1",
      "id": "/subscriptions/your-subscription-id/resourceGroups/your-rg/providers/Microsoft.Compute/virtualMachineScaleSets/pipvmss/virtualMachines/0/networkInterfaces/pipvmssnic/ipConfigurations/yourvmssipconfig/publicIPAddresses/pub1",
      "etag": "W/\"a64060d5-4dea-4379-a11d-b23cd49a3c8d\"",
      "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "ee8cb20f-af8e-4cd6-892f-441ae2bf701f",
        "ipAddress": "13.84.190.11",
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Dynamic",
        "idleTimeoutInMinutes": 15,
        "ipConfiguration": {
          "id": "/subscriptions/your-subscription-id/resourceGroups/your-rg/providers/Microsoft.Compute/virtualMachineScaleSets/yourvmss/virtualMachines/0/networkInterfaces/yourvmssnic/ipConfigurations/yourvmssipconfig"
        }
      }
    },
    {
      "name": "pub1",
      "id": "/subscriptions/your-subscription-id/resourceGroups/your-rg/providers/Microsoft.Compute/virtualMachineScaleSets/yourvmss/virtualMachines/3/networkInterfaces/yourvmssnic/ipConfigurations/yourvmssipconfig/publicIPAddresses/pub1",
      "etag": "W/\"5f6ff30c-a24c-4818-883c-61ebd5f9eee8\"",
      "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "036ce266-403f-41bd-8578-d446d7397c2f",
        "ipAddress": "13.84.159.176",
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Dynamic",
        "idleTimeoutInMinutes": 15,
        "ipConfiguration": {
          "id": "/subscriptions/your-subscription-id/resourceGroups/your-rg/providers/Microsoft.Compute/virtualMachineScaleSets/yourvmss/virtualMachines/3/networkInterfaces/yourvmssnic/ipConfigurations/yourvmssipconfig"
        }
      }
    }
  ]
}
```

Multiple IP addresses per NIC

Every NIC attached to a VM in a scale set can have one or more IP configurations associated with it. Each configuration is assigned one private IP address. Each configuration may also have one public IP address resource associated with it. To understand how many IP addresses can be assigned to a NIC, and how many public IP addresses you can use in an Azure subscription, refer to [Azure limits](#).

Multiple NICs per virtual machine

You can have up to 8 NICs per virtual machine, depending on machine size. The maximum number of NICs per machine is available in the [VM size article](#). All NICs connected to a VM instance must connect to the same virtual network. The NICs can connect to different subnets, but all subnets must be part of the same virtual network.

The following example is a scale set network profile showing multiple NIC entries, and multiple public IPs per virtual machine:

```
"networkProfile": {
    "networkInterfaceConfigurations": [
        {
            "name": "nic1",
            "properties": {
                "primary": true,
                "ipConfigurations": [
                    {
                        "name": "ip1",
                        "properties": {
                            "subnet": {
                                "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/virtualNetworks/', variables('vnetName'),
'/subnets/subnet1')]"
                            },
                            "publicIpAddressConfiguration": {
                                "name": "pub1",
                                "properties": {
                                    "idleTimeoutInMinutes": 15
                                }
                            },
                            "loadBalancerInboundNatPools": [
                                {
                                    "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/', variables('lbName'),
'/inboundNatPools/natPool1')]"
                                }
                            ],
                            "loadBalancerBackendAddressPools": [
                                {
                                    "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/', variables('lbName'),
'/backendAddressPools/addressPool1')]"
                                }
                            ]
                        }
                    }
                ]
            }
        },
        {
            "name": "nic2",
            "properties": {
                "primary": false,
                "ipConfigurations": [
                    {
                        "name": "ip1",
                        "properties": {
                            "subnet": {
                                "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/virtualNetworks/', variables('vnetName'),
'/subnets/subnet1')]"
                            },
                            "publicIpAddressConfiguration": {
                                "name": "pub1",
                                "properties": {
                                    "idleTimeoutInMinutes": 15
                                }
                            },
                            "loadBalancerInboundNatPools": [
                                {
                                    "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/', variables('lbName'),
'/inboundNatPools/natPool1')]"
                                }
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

```
        }
    ],
    "loadBalancerBackendAddressPools": [
        {
            "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/', variables('lbName'),
'/backendAddressPools/addressPool1')]"
        }
    ]
}
]
}
```

NSG & ASGs per scale set

[Network Security Groups](#) allow you to filter traffic to and from Azure resources in an Azure virtual network using security rules. [Application Security Groups](#) enable you to handle network security of Azure resources and group them as an extension of your application's structure.

Network Security Groups can be applied directly to a scale set, by adding a reference to the network interface configuration section of the scale set virtual machine properties.

Application Security Groups can also be specified directly to a scale set, by adding a reference to the network interface ip configurations section of the scale set virtual machine properties.

For example:

```

"networkProfile": {
    "networkInterfaceConfigurations": [
        {
            "name": "nic1",
            "properties": {
                "primary": true,
                "ipConfigurations": [
                    {
                        "name": "ip1",
                        "properties": {
                            "subnet": {
                                "id": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/virtualNetworks/', variables('vnetName'), '/subnets/subnet1')]"
                            },
                            "applicationSecurityGroups": [
                                {
                                    "id": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/applicationSecurityGroups/', variables('asgName'))]"
                                }
                            ],
                            "loadBalancerInboundNatPools": [
                                {
                                    "id": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/loadBalancers/', variables('lbName'), '/inboundNatPools/natPool1')]"
                                }
                            ],
                            "loadBalancerBackendAddressPools": [
                                {
                                    "id": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/loadBalancers/', variables('lbName'), '/backendAddressPools/addressPool1')]"
                                }
                            ]
                        }
                    ],
                    "networkSecurityGroup": {
                        "id": "[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/networkSecurityGroups/', variables('nsgName'))]"
                    }
                }
            }
        ]
    ]
}

```

To verify your Network Security Group is associated with your scale set, use the `az vmss show` command. The below example uses `--query` to filter the results and only show the relevant section of the output.

```

az vmss show \
-g myResourceGroup \
-n myScaleSet \
--query virtualMachineProfile.networkProfile.networkInterfaceConfigurations[].networkSecurityGroup

[
{
    "id": "/subscriptions/.../resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/nsgName",
    "resourceGroup": "myResourceGroup"
}
]

```

To verify your Application Security Group is associated with your scale set, use the `az vmss show` command. The below example uses `--query` to filter the results and only show the relevant section of the output.

```
az vmss show \
    -g myResourceGroup \
    -n myScaleSet \
    --query
virtualMachineProfile.networkProfile.networkInterfaceConfigurations[].ipConfigurations[].applicationSecurityGroups

[
  [
    {
      "id": "/subscriptions/.../resourceGroups/myResourceGroup/providers/Microsoft.Network/applicationSecurityGroups/asgName",
      "resourceGroup": "myResourceGroup"
    }
  ]
]
```

Next steps

For more information about Azure virtual networks, see [Azure virtual networks overview](#).

Azure virtual machine scale set automatic OS image upgrades

1/19/2020 • 11 minutes to read • [Edit Online](#)

Enabling automatic OS image upgrades on your scale set helps ease update management by safely and automatically upgrading the OS disk for all instances in the scale set.

Automatic OS upgrade has the following characteristics:

- Once configured, the latest OS image published by image publishers is automatically applied to the scale set without user intervention.
- Upgrades batches of instances in a rolling manner each time a new platform image is published by the publisher.
- Integrates with application health probes and [Application Health extension](#).
- Works for all VM sizes, and for both Windows and Linux platform images.
- You can opt out of automatic upgrades at any time (OS Upgrades can be initiated manually as well).
- The OS Disk of a VM is replaced with the new OS Disk created with latest image version. Configured extensions and custom data scripts are run, while persisted data disks are retained.
- [Extension sequencing](#) is supported.
- Automatic OS image upgrade can be enabled on a scale set of any size.

How does automatic OS image upgrade work?

An upgrade works by replacing the OS disk of a VM with a new disk created using the latest image version. Any configured extensions and custom data scripts are run on the OS disk, while persisted data disks are retained. To minimize the application downtime, upgrades take place in batches, with no more than 20% of the scale set upgrading at any time. You can also integrate an Azure Load Balancer application health probe or [Application Health extension](#). We recommended incorporating an application heartbeat and validate upgrade success for each batch in the upgrade process.

The upgrade process works as follows:

- Before beginning the upgrade process, the orchestrator will ensure that no more than 20% of instances in the entire scale set are unhealthy (for any reason).
- The upgrade orchestrator identifies the batch of VM instances to upgrade, with any one batch having a maximum of 20% of the total instance count. For smaller scale sets with 5 or fewer instances, the batch size for an upgrade is one virtual machine instance.
- The OS disk of the selected batch of VM instances is replaced with a new OS disk created from the latest image. All specified extensions and configurations in the scale set model are applied to the upgraded instance.
- For scale sets with configured application health probes or Application Health extension, the upgrade waits up to 5 minutes for the instance to become healthy, before moving on to upgrade the next batch. If an instance does not recover its health in 5 minutes after an upgrade, then by default the previous OS disk for the instance is restored.
- The upgrade orchestrator also tracks the percentage of instances that become unhealthy post an upgrade. The upgrade will stop if more than 20% of upgraded instances become unhealthy during the upgrade process.
- The above process continues until all instances in the scale set have been upgraded.

The scale set OS upgrade orchestrator checks for the overall scale set health before upgrading every batch. While

upgrading a batch, there could be other concurrent planned or unplanned maintenance activities that could impact the health of your scale set instances. In such cases if more than 20% of the scale set's instances become unhealthy, then the scale set upgrade stops at the end of current batch.

Supported OS images

Only certain OS platform images are currently supported. Custom images aren't currently supported.

The following SKUs are currently supported (and more are added periodically):

PUBLISHER	OS OFFER	SKU
Canonical	UbuntuServer	16.04-LTS
Canonical	UbuntuServer	18.04-LTS
Rogue Wave (OpenLogic)	CentOS	7.5
CoreOS	CoreOS	Stable
Microsoft Corporation	WindowsServer	2012-R2-Datacenter
Microsoft Corporation	WindowsServer	2016-Datacenter
Microsoft Corporation	WindowsServer	2016-Datacenter-Smalldisk
Microsoft Corporation	WindowsServer	2016-Datacenter-with-Containers
Microsoft Corporation	WindowsServer	2019-Datacenter
Microsoft Corporation	WindowsServer	2019-Datacenter-Smalldisk
Microsoft Corporation	WindowsServer	2019-Datacenter-with-Containers
Microsoft Corporation	WindowsServer	Datacenter-Core-1903-with-Containers-smalldisk

Requirements for configuring automatic OS image upgrade

- The *version* property of the platform image must be set to *latest*.
- Use application health probes or [Application Health extension](#) for non-Service Fabric scale sets.
- Use Compute API version 2018-10-01 or higher.
- Ensure that external resources specified in the scale set model are available and updated. Examples include SAS URI for bootstrapping payload in VM extension properties, payload in storage account, reference to secrets in the model, and more.
- For scale sets using Windows virtual machines, starting with Compute API version 2019-03-01, the property *virtualMachineProfile.osProfile.windowsConfiguration.enableAutomaticUpdates* property must be set to *false* in the scale set model definition. The above property enables in-VM upgrades where "Windows Update" applies operating system patches without replacing the OS disk. With automatic OS image upgrades enabled on your scale set, an additional update through "Windows Update" is not required.

Service Fabric requirements

If you are using Service Fabric, ensure the following conditions are met:

- Service Fabric [durability level](#) is Silver or Gold, and not Bronze.
- The Service Fabric extension on the scale set model definition must have TypeHandlerVersion 1.1 or above.
- Durability level should be the same at the Service Fabric cluster and Service Fabric extension on the scale set model definition.

Ensure that durability settings are not mismatched on the Service Fabric cluster and Service Fabric extension, as a mismatch will result in upgrade errors. Durability levels can be modified per the guidelines outlined on [this page](#).

Configure automatic OS image upgrade

To configure automatic OS image upgrade, ensure that the `automaticOSUpgradePolicy.enableAutomaticOSUpgrade` property is set to `true` in the scale set model definition.

REST API

The following example describes how to set automatic OS upgrades on a scale set model:

```
PUT or PATCH on
`/subscriptions/subscription_id/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet?api-version=2018-10-01`
```

```
{
  "properties": {
    "upgradePolicy": {
      "automaticOSUpgradePolicy": {
        "enableAutomaticOSUpgrade": true
      }
    }
  }
}
```

Azure PowerShell

Use the [Update-AzVmss](#) cmdlet to configure automatic OS image upgrades for your scale set. The following example configures automatic upgrades for the scale set named `myScaleSet` in the resource group named `myResourceGroup`:

```
Update-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -AutomaticOSUpgrade $true
```

Azure CLI 2.0

Use [az vmss update](#) to configure automatic OS image upgrades for your scale set. Use Azure CLI 2.0.47 or above. The following example configures automatic upgrades for the scale set named `myScaleSet` in the resource group named `myResourceGroup`:

```
az vmss update --name myScaleSet --resource-group myResourceGroup --set
UpgradePolicy.AutomaticOSUpgradePolicy.EnableAutomaticOSUpgrade=true
```

Using Application Health Probes

During an OS Upgrade, VM instances in a scale set are upgraded one batch at a time. The upgrade should continue only if the customer application is healthy on the upgraded VM instances. We recommend that the application provides health signals to the scale set OS Upgrade engine. By default, during OS Upgrades the platform considers VM power state and extension provisioning state to determine if a VM instance is healthy after an upgrade. During the OS Upgrade of a VM instance, the OS disk on a VM instance is replaced with a new disk

based on latest image version. After the OS Upgrade has completed, the configured extensions are run on these VMs. The application is considered healthy only when all the extensions on the instance are successfully provisioned.

A scale set can optionally be configured with Application Health Probes to provide the platform with accurate information on the ongoing state of the application. Application Health Probes are Custom Load Balancer Probes that are used as a health signal. The application running on a scale set VM instance can respond to external HTTP or TCP requests indicating whether it's healthy. For more information on how Custom Load Balancer Probes work, see to [Understand load balancer probes](#). Application Health Probes are not supported for Service Fabric scale sets. Non-Service Fabric scale sets require either Load Balancer application health probes or [Application Health extension](#).

If the scale set is configured to use multiple placement groups, probes using a [Standard Load Balancer](#) need to be used.

Configuring a Custom Load Balancer Probe as Application Health Probe on a scale set

As a best practice, create a load balancer probe explicitly for scale set health. The same endpoint for an existing HTTP probe or TCP probe can be used, but a health probe could require different behavior from a traditional load-balancer probe. For example, a traditional load balancer probe could return unhealthy if the load on the instance is too high, but that would not be appropriate for determining the instance health during an automatic OS upgrade. Configure the probe to have a high probing rate of less than two minutes.

The load-balancer probe can be referenced in the *networkProfile* of the scale set and can be associated with either an internal or public facing load-balancer as follows:

```
"networkProfile": {  
    "healthProbe" : {  
        "id": "[concat(variables('lbId'), '/probes/', variables('sshProbeName'))]"  
    },  
    "networkInterfaceConfigurations":  
    ...  
}
```

NOTE

When using Automatic OS Upgrades with Service Fabric, the new OS image is rolled out Update Domain by Update Domain to maintain high availability of the services running in Service Fabric. To utilize Automatic OS Upgrades in Service Fabric your cluster must be configured to use the Silver Durability Tier or higher. For more information on the durability characteristics of Service Fabric clusters, please see [this documentation](#).

Keep credentials up-to-date

If your scale set uses any credentials to access external resources, such as a VM extension configured to use a SAS token for storage account, then ensure that the credentials are updated. If any credentials, including certificates and tokens, have expired, the upgrade will fail and the first batch of VMs will be left in a failed state.

The recommended steps to recover VMs and re-enable automatic OS upgrade if there's a resource authentication failure are:

- Regenerate the token (or any other credentials) passed into your extension(s).
- Ensure that any credential used from inside the VM to talk to external entities is up-to-date.
- Update extension(s) in the scale set model with any new tokens.
- Deploy the updated scale set, which will update all VM instances including the failed ones.

Using Application Health extension

The Application Health extension is deployed inside a virtual machine scale set instance and reports on VM health from inside the scale set instance. You can configure the extension to probe on an application endpoint and update the status of the application on that instance. This instance status is checked by Azure to determine whether an instance is eligible for upgrade operations.

As the extension reports health from within a VM, the extension can be used in situations where external probes such as Application Health Probes (that utilize custom Azure Load Balancer [probes](#)) can't be used.

There are multiple ways of deploying the Application Health extension to your scale sets as detailed in the examples in [this article](#).

Get the history of automatic OS image upgrades

You can check the history of the most recent OS upgrade performed on your scale set with Azure PowerShell, Azure CLI 2.0, or the REST APIs. You can get history for the last five OS upgrade attempts within the past two months.

REST API

The following example uses [REST API](#) to check the status for the scale set named *myScaleSet* in the resource group named *myResourceGroup*:

```
GET on  
`/subscriptions/subscription_id/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet/osUpgradeHistory?api-version=2018-10-01`
```

The GET call returns properties similar to the following example output:

```
{  
  "value": [  
    {  
      "properties": {  
        "runningStatus": {  
          "code": "RollingForward",  
          "startTime": "2018-07-24T17:46:06.1248429+00:00",  
          "completedTime": "2018-04-21T12:29:25.0511245+00:00"  
        },  
        "progress": {  
          "successfulInstanceCount": 16,  
          "failedInstanceCount": 0,  
          "inProgressInstanceCount": 4,  
          "pendingInstanceCount": 0  
        },  
        "startedBy": "Platform",  
        "targetImageReference": {  
          "publisher": "MicrosoftWindowsServer",  
          "offer": "WindowsServer",  
          "sku": "2016-Datacenter",  
          "version": "2016.127.20180613"  
        },  
        "rollbackInfo": {  
          "successfullyRolledbackInstanceCount": 0,  
          "failedRolledbackInstanceCount": 0  
        }  
      },  
      "type": "Microsoft.Compute/virtualMachineScaleSets/rollingUpgrades",  
      "location": "westeurope"  
    }  
  ]  
}
```

Azure PowerShell

Use the [Get-AzVmss](#) cmdlet to check OS upgrade history for your scale set. The following example details how you review the OS upgrade status for a scale set named *myScaleSet* in the resource group named *myResourceGroup*:

```
Get-AzVmss -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet" -OSUpgradeHistory
```

Azure CLI 2.0

Use [az vmss get-os-upgrade-history](#) to check the OS upgrade history for your scale set. Use Azure CLI 2.0.47 or above. The following example details how you review the OS upgrade status for a scale set named *myScaleSet* in the resource group named *myResourceGroup*:

```
az vmss get-os-upgrade-history --resource-group myResourceGroup --name myScaleSet
```

How to get the latest version of a platform OS image?

You can get the available image versions for automatic OS upgrade supported SKUs using the below examples:

REST API

```
GET on  
`/subscriptions/subscription_id/providers/Microsoft.Compute/locations/{location}/publishers/{publisherName}/artifacts/vmimage/offers/{offer}/skus/{skus}/versions?api-version=2018-10-01`
```

Azure PowerShell

```
Get-AzVmImage -Location "westus" -PublisherName "Canonical" -Offer "UbuntuServer" -Skus "16.04-LTS"
```

Azure CLI 2.0

```
az vm image list --location "westus" --publisher "Canonical" --offer "UbuntuServer" --sku "16.04-LTS" --all
```

Manually trigger OS image upgrades

With automatic OS image upgrade enabled on your scale set, you do not need to manually trigger image updates on your scale set. The OS upgrade orchestrator will automatically apply the latest available image version to your scale set instances without any manual intervention.

For specific cases where you do not want to wait for the orchestrator to apply the latest image, you can trigger an OS image upgrade manually using the below examples.

NOTE

Manual trigger of OS image upgrades does not provide automatic rollback capabilities. If an instance does not recover its health after an upgrade operation, its previous OS disk can't be restored.

REST API

Use the [Start OS Upgrade](#) API call to start a rolling upgrade to move all virtual machine scale set instances to the latest available platform image OS version. Instances that are already running the latest available OS version are not affected. The following example details how you can start a rolling OS upgrade on a scale set named

myScaleSet in the resource group named *myResourceGroup*:

```
POST on  
`/subscriptions/subscription_id/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachineScale  
Sets/myScaleSet/osRollingUpgrade?api-version=2018-10-01`
```

Azure PowerShell

Use the [Start-AzVmssRollingOSUpgrade](#) cmdlet to check OS upgrade history for your scale set. The following example details how you can start a rolling OS upgrade on a scale set named *myScaleSet* in the resource group named *myResourceGroup*:

```
Start-AzVmssRollingOSUpgrade -ResourceGroupName "myResourceGroup" -VMScaleSetName "myScaleSet"
```

Azure CLI 2.0

Use [az vmss rolling-upgrade start](#) to check the OS upgrade history for your scale set. Use Azure CLI 2.0.47 or above. The following example details how you can start a rolling OS upgrade on a scale set named *myScaleSet* in the resource group named *myResourceGroup*:

```
az vmss rolling-upgrade start --resource-group "myResourceGroup" --name "myScaleSet" --subscription  
"subscriptionId"
```

Deploy with a template

You can use templates to deploy a scale set with automatic OS upgrades for supported images such as [Ubuntu 16.04-LTS](#).



Next steps

For more examples on how to use automatic OS upgrades with scale sets, review the [GitHub repo](#).

Instance Protection for Azure virtual machine scale set instances

2/28/2020 • 5 minutes to read • [Edit Online](#)

Azure virtual machine scale sets enable better elasticity for your workloads through [Autoscale](#), so you can configure when your infrastructure scales-out and when it scales-in. Scale sets also enable you to centrally manage, configure, and update a large number of VMs through different [upgrade policy](#) settings. You can configure an update on the scale set model and the new configuration is applied automatically to every scale set instance if you've set the upgrade policy to Automatic or Rolling.

As your application processes traffic, there can be situations where you want specific instances to be treated differently from the rest of the scale set instance. For example, certain instances in the scale set could be performing long-running operations, and you don't want these instances to be scaled-in until the operations complete. You might also have specialized a few instances in the scale set to perform additional or different tasks than the other members of the scale set. You require these 'special' VMs not to be modified with the other instances in the scale set. Instance protection provides the additional controls to enable these and other scenarios for your application.

This article describes how you can apply and use the different instance protection capabilities with scale set instances.

Types of instance protection

Scale sets provide two types of instance protection capabilities:

- **Protect from scale-in**

- Enabled through **protectFromScaleIn** property on the scale set instance
- Protects instance from Autoscale initiated scale-in
- User-initiated instance operations (including instance delete) are **not blocked**
- Operations initiated on the scale set (upgrade, reimage, deallocate, etc.) are **not blocked**

- **Protect from scale set actions**

- Enabled through **protectFromScaleSetActions** property on the scale set instance
- Protects instance from Autoscale initiated scale-in
- Protects instance from operations initiated on the scale set (such as upgrade, reimage, deallocate, etc.)
- User-initiated instance operations (including instance delete) are **not blocked**
- Delete of the full scale set is **not blocked**

Protect from scale-in

Instance protection can be applied to scale set instances after the instances are created. Protection is applied and modified only on the [instance model](#) and not on the [scale set model](#).

There are multiple ways of applying scale-in protection on your scale set instances as detailed in the examples below.

Azure portal

You can apply scale-in protection through the Azure portal to an instance in the scale set. You cannot adjust more than one instance at a time. Repeat the steps for each instance you want to protect.

1. Go to an existing virtual machine scale set.
2. Select **Instances** from the menu on the left, under **Settings**.
3. Select the name of the instance you want to protect.
4. Select the **Protection Policy** tab.
5. In the **Protection Policy** blade, select the **Protect from scale-in** option.
6. Select **Save**.

REST API

The following example applies scale-in protection to an instance in the scale set.

```
PUT on  
`/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachineS  
caleSets/{vmScaleSetName}/virtualMachines/{instance-id}?api-version=2019-03-01`
```

```
{  
  "properties": {  
    "protectionPolicy": {  
      "protectFromScaleIn": true  
    }  
  }  
}
```

NOTE

Instance protection is only supported with API version 2019-03-01 and above

Azure PowerShell

Use the [Update-AzVmssVM](#) cmdlet to apply scale-in protection to your scale set instance.

The following example applies scale-in protection to an instance in the scale set having instance ID 0.

```
Update-AzVmssVM `  
-ResourceGroupName "myResourceGroup" `  
-VMScaleSetName "myVMScaleSet" `  
-InstanceId 0 `  
-ProtectFromScaleIn $true
```

Azure CLI 2.0

Use [az vmss update](#) to apply scale-in protection to your scale set instance.

The following example applies scale-in protection to an instance in the scale set having instance ID 0.

```
az vmss update \  
--resource-group <myResourceGroup> \  
--name <myVMScaleSet> \  
--instance-id 0 \  
--protect-from-scale-in true
```

Protect from scale set actions

Instance protection can be applied to scale set instances after the instances are created. Protection is applied and modified only on the [instance model](#) and not on the [scale set model](#).

Protecting an instance from scale set actions also protects the instance from Autoscale initiated scale-in.

There are multiple ways of applying scale set actions protection on your scale set instances as detailed in the examples below.

Azure portal

You can apply protection from scale set actions through the Azure portal to an instance in the scale set. You cannot adjust more than one instance at a time. Repeat the steps for each instance you want to protect.

1. Go to an existing virtual machine scale set.
2. Select **Instances** from the menu on the left, under **Settings**.
3. Select the name of the instance you want to protect.
4. Select the **Protection Policy** tab.
5. In the **Protection Policy** blade, select the **Protect from scale set actions** option.
6. Select **Save**.

REST API

The following example applies protection from scale set actions to an instance in the scale set.

```
PUT on  
`/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachineScaleSets/{vMScaleSetName}/virtualMachines/{instance-id}?api-version=2019-03-01`
```

```
{  
  "properties": {  
    "protectionPolicy": {  
      "protectFromScaleIn": true,  
      "protectFromScaleSetActions": true  
    }  
  }  
}
```

NOTE

Instance protection is only supported with API version 2019-03-01 and above.

Protecting an instance from scale set actions also protects the instance from Autoscale initiated scale-in. You can't specify "protectFromScaleIn": false when setting "protectFromScaleSetActions": true

Azure PowerShell

Use the [Update-AzVmssVM](#) cmdlet to apply protection from scale set actions to your scale set instance.

The following example applies protection from scale set actions to an instance in the scale set having instance ID 0.

```
Update-AzVmssVM `  
-ResourceGroupName "myResourceGroup" `  
-VMScaleSetName "myVMScaleSet" `  
-InstanceId 0 `  
-ProtectFromScaleIn $true `  
-ProtectFromScaleSetAction $true
```

Azure CLI 2.0

Use [az vmss update](#) to apply protection from scale set actions to your scale set instance.

The following example applies protection from scale set actions to an instance in the scale set having instance ID 0.

```
az vmss update \
--resource-group <myResourceGroup> \
--name <myVMScaleSet> \
--instance-id 0 \
--protect-from-scale-in true \
--protect-from-scale-set-actions true
```

Troubleshoot

No protectionPolicy on scale set model

Instance protection is only applicable on scale set instances and not on the scale set model.

No protectionPolicy on scale set instance model

By default, protection policy is not applied to an instance when it is created.

You can apply instance protection to scale set instances after the instances are created.

Not able to apply instance protection

Instance protection is only supported with API version 2019-03-01 and above. Check the API version being used and update as required. You might also need to update your PowerShell or CLI to the latest version.

Next steps

Learn how to [deploy your application](#) on virtual machine scale sets.

Use custom scale-in policies with Azure virtual machine scale sets

2/28/2020 • 8 minutes to read • [Edit Online](#)

A virtual machine scale set deployment can be scaled-out or scaled-in based on an array of metrics, including platform and user-defined custom metrics. While a scale-out creates new virtual machines based on the scale set model, a scale-in affects running virtual machines that may have different configurations and/or functions as the scale set workload evolves.

The scale-in policy feature provides users a way to configure the order in which virtual machines are scaled-in, by way of three scale-in configurations:

1. Default
2. NewestVM
3. OldestVM

Default scale-in policy

By default, virtual machine scale set applies this policy to determine which instance(s) will be scaled in. With the *Default* policy, VMs are selected for scale-in in the following order:

1. Balance virtual machines across availability zones (if the scale set is deployed in zonal configuration)
2. Balance virtual machines across fault domains (best effort)
3. Delete virtual machine with the highest instance ID

Users do not need to specify a scale-in policy if they just want the default ordering to be followed.

Note that balancing across availability zones or fault domains does not move instances across availability zones or fault domains. The balancing is achieved through deletion of virtual machines from the unbalanced availability zones or fault domains until the distribution of virtual machines becomes balanced.

NewestVM scale-in policy

This policy will delete the newest created virtual machine in the scale set, after balancing VMs across availability zones (for zonal deployments). Enabling this policy requires a configuration change on the virtual machine scale set model.

OldestVM scale-in policy

This policy will delete the oldest created virtual machine in the scale set, after balancing VMs across availability zones (for zonal deployments). Enabling this policy requires a configuration change on the virtual machine scale set model.

Enabling scale-in policy

A scale-in policy is defined in the virtual machine scale set model. As noted in the sections above, a scale-in policy definition is needed when using the 'NewestVM' and 'OldestVM' policies. Virtual machine scale set will automatically use the 'Default' scale-in policy if there is no scale-in policy definition found on the scale set model.

A scale-in policy can be defined on the virtual machine scale set model in the following ways:

Azure portal

The following steps define the scale-in policy when creating a new scale set.

1. Go to **Virtual machine scale sets**.
2. Select **+ Add** to create a new scale set.
3. Go to the **Scaling** tab.
4. Locate the **Scale-in policy** section.
5. Select a scale-in policy from the drop-down.
6. When you are done creating the new scale set, select **Review + create** button.

Using API

Execute a PUT on the virtual machine scale set using API 2019-03-01:

```
PUT
https://management.azure.com/subscriptions/<sub-
id>/resourceGroups/<myRG>/providers/Microsoft.Compute/virtualMachineScaleSets/<myVMSS>?api-version=2019-03-01

{
  "location": "<VMSS location>",
  "properties": {
    "scaleInPolicy": {
      "rules": ["OldestVM"]
    }
  }
}
```

Azure PowerShell

Create a resource group, then create a new scale set with scale-in policy set as *OldestVM*.

```
New-AzResourceGroup -ResourceGroupName "myResourceGroup" -Location "<VMSS location>"
New-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "<VMSS location>" ` 
-VMScaleSetName "myScaleSet" ` 
-ScaleInPolicy "OldestVM"
```

Azure CLI 2.0

The following example adds a scale-in policy while creating a new scale set. First create a resource group, then create a new scale set with scale-in policy as *OldestVM*.

```
az group create --name <myResourceGroup> --location <VMSSLocation>
az vmss create \
--resource-group <myResourceGroup> \
--name <myVMScaleSet> \
--image UbuntuLTS \
--admin-username <azureuser> \
--generate-ssh-keys \
--scale-in-policy OldestVM
```

Using Template

In your template, under “properties”, add the following:

```
"scaleInPolicy": {
  "rules": ["OldestVM"]
}
```

The above blocks specify that the virtual machine scale set will delete the Oldest VM in a zone-balanced scale set, when a scale-in is triggered (through Autoscale or manual delete).

When a virtual machine scale set is not zone balanced, the scale set will first delete VMs across the imbalanced zone(s). Within the imbalanced zones, the scale set will use the scale-in policy specified above to determine which VM to scale in. In this case, within an imbalanced zone, the scale set will select the Oldest VM in that zone to be deleted.

For non-zonal virtual machine scale set, the policy selects the oldest VM across the scale set for deletion.

The same process applies when using 'NewestVM' in the above scale-in policy.

Modifying scale-in policies

Modifying the scale-in policy follows the same process as applying the scale-in policy. For example, if in the above example, you want to change the policy from 'OldestVM' to 'NewestVM', you can do so by:

Azure portal

You can modify the scale-in policy of an existing scale set through the Azure portal.

1. In an existing virtual machine scale set, select **Scaling** from the menu on the left.
2. Select the **Scale-In Policy** tab.
3. Select a scale-in policy from the drop-down.
4. When you are done, select **Save**.

Using API

Execute a PUT on the virtual machine scale set using API 2019-03-01:

```
PUT  
https://management.azure.com/subscriptions/<sub-  
id>/resourceGroups/<myRG>/providers/Microsoft.Compute/virtualMachineScaleSets/<myVMSS>?api-version=2019-03-01  
  
{  
    "location": "<VMSS location>",  
    "properties": {  
        "scaleInPolicy": {  
            "rules": ["NewestVM"]  
        }  
    }  
}
```

Azure PowerShell

Update the scale-in policy of an existing scale set:

```
Update-AzVmss `  
-ResourceGroupName "myResourceGroup" `  
-VMScaleSetName "myScaleSet" `  
-ScaleInPolicy "OldestVM"
```

Azure CLI 2.0

The following is an example for updating the scale-in policy of an existing scale set:

```
az vmss update \  
--resource-group <myResourceGroup> \  
--name <myVMScaleSet> \  
--scale-in-policy OldestVM
```

Using Template

In your template, under "properties", modify the template as below and redeploy:

```

"scaleInPolicy": {
    "rules": [ "NewestVM" ]
}

```

The same process will apply if you decide to change 'NewestVM' to 'Default' or 'OldestVM'

Instance protection and scale-in policy

Virtual machine scale sets provide two types of [instance protection](#):

1. Protect from scale-in
2. Protect from scale-set actions

A protected virtual machine is not deleted through a scale-in action, regardless of the scale-in policy applied. For example, if VM_0 (oldest VM in the scale set) is protected from scale-in, and the scale set has 'OldestVM' scale-in policy enabled, VM_0 will not be considered for being scaled in, even though it is the oldest VM in the scale set.

A protected virtual machine can be manually deleted by the user at any time, regardless of the scale-in policy enabled on the scale set.

Usage examples

The below examples demonstrate how a virtual machine scale set will select VMs to be deleted when a scale-in event is triggered. Virtual machines with the highest instance IDs are assumed to be the newest VMs in the scale set and the VMs with the smallest instance IDs are assumed to be the oldest VMs in the scale set.

OldestVM scale-in policy

EVENT	INSTANCE IDS IN ZONE1	INSTANCE IDS IN ZONE2	INSTANCE IDS IN ZONE3	SCALE-IN SELECTION
Initial	3, 4, 5, 10	2, 6, 9, 11	1, 7, 8	
Scale-in	3, 4, 5, 10	2, 6, 9, 11	1, 7, 8	Choose between Zone 1 and 2, even though Zone 3 has the oldest VM. Delete VM2 from Zone 2 as it is the oldest VM in that zone.
Scale-in	3, 4, 5, 10	6, 9, 11	1, 7, 8	Choose Zone 1 even though Zone 3 has the oldest VM. Delete VM3 from Zone 1 as it is the oldest VM in that zone.
Scale-in	4, 5, 10	6, 9, 11	1, 7, 8	Zones are balanced. Delete VM1 in Zone 3 as it is the oldest VM in the scale set.

EVENT	INSTANCE IDS IN ZONE1	INSTANCE IDS IN ZONE2	INSTANCE IDS IN ZONE3	SCALE-IN SELECTION
Scale-in	4 , 5, 10	6, 9, 11	7, 8	Choose between Zone 1 and Zone 2. Delete VM4 in Zone 1 as it is the oldest VM across the two Zones.
Scale-in	5, 10	6 , 9, 11	7, 8	Choose Zone 2 even though Zone 1 has the oldest VM. Delete VM6 in Zone 1 as it is the oldest VM in that zone.
Scale-in	5 , 10	9, 11	7, 8	Zones are balanced. Delete VM5 in Zone 1 as it is the oldest VM in the scale set.

For non-zonal virtual machine scale sets, the policy selects the oldest VM across the scale set for deletion. Any "protected" VM will be skipped for deletion.

NewestVM scale-in policy

EVENT	INSTANCE IDS IN ZONE1	INSTANCE IDS IN ZONE2	INSTANCE IDS IN ZONE3	SCALE-IN SELECTION
Initial	3, 4, 5, 10	2, 6, 9, 11	1, 7, 8	
Scale-in	3, 4, 5, 10	2, 6, 9, 11	1, 7, 8	Choose between Zone 1 and 2. Delete VM11 from Zone 2 as it is the newest VM across the two zones.
Scale-in	3, 4, 5, 10	2, 6, 9	1, 7, 8	Choose Zone 1 as it has more VMs than the other two zones. Delete VM10 from Zone 1 as it is the newest VM in that Zone.
Scale-in	3, 4, 5	2, 6, 9	1, 7, 8	Zones are balanced. Delete VM9 in Zone 2 as it is the newest VM in the scale set.
Scale-in	3, 4, 5	2, 6	1, 7, 8	Choose between Zone 1 and Zone 3. Delete VM8 in Zone 3 as it is the newest VM in that Zone.

Event	Instance IDs in Zone1	Instance IDs in Zone2	Instance IDs in Zone3	Scale-in Selection
Scale-in	3, 4, 5	2, 6	1, 7	Choose Zone 1 even though Zone 3 has the newest VM. Delete VM5 in Zone 1 as it is the newest VM in that Zone.
Scale-in	3, 4	2, 6	1, 7	Zones are balanced. Delete VM7 in Zone 3 as it is the newest VM in the scale set.

For non-zonal virtual machine scale sets, the policy selects the newest VM across the scale set for deletion. Any "protected" VM will be skipped for deletion.

Troubleshoot

1. Failure to enable scaleInPolicy If you get a 'BadRequest' error with an error message stating "Could not find member 'scaleInPolicy' on object of type 'properties'", then check the API version used for virtual machine scale set. API version 2019-03-01 or higher is required for this feature.
2. Wrong selection of VMs for scale-in Refer to the examples above. If your virtual machine scale set is a Zonal deployment, scale-in policy is applied first to the imbalanced Zones and then across the scale set once it is zone balanced. If the order of scale-in is not consistent with the examples above, raise a query with the virtual machine scale set team for troubleshooting.

Next steps

Learn how to [deploy your application](#) on virtual machine scale sets.

Terminate notification for Azure virtual machine scale set instances

2/28/2020 • 7 minutes to read • [Edit Online](#)

Scale set instances can opt in to receive instance termination notifications and set a pre-defined delay timeout to the terminate operation. The termination notification is sent through Azure Metadata Service – [Scheduled Events](#), which provides notifications for and delaying of impactful operations such as reboots and redeploy. The solution adds another event – Terminate – to the list of Scheduled Events, and the associated delay of the terminate event will depend on the delay limit as specified by users in their scale set model configurations.

Once enrolled into the feature, scale set instances don't need to wait for specified timeout to expire before the instance is deleted. After receiving a Terminate notification, the instance can choose to be deleted at any time before the terminate timeout expires.

Enable terminate notifications

There are multiple ways of enabling termination notifications on your scale set instances, as detailed in the examples below.

Azure portal

The following steps enable terminate notification when creating a new scale set.

1. Go to **Virtual machine scale sets**.
2. Select **+ Add** to create a new scale set.
3. Go to the **Management** tab.
4. Locate the **Instance termination** section.
5. For **Instance termination notification**, select **On**.
6. For **Termination delay (minutes)**, set the desired default timeout.
7. When you are done creating the new scale set, select **Review + create** button.

NOTE

You are unable to set terminate notifications on existing scale sets in Azure portal

REST API

The following example enables terminate notification on the scale set model.

```
PUT on  
`/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachineS  
caleSets/{vmScaleSetName}?api-version=2019-03-01`
```

```
{
  "properties": {
    "virtualMachineProfile": {
      "scheduledEventsProfile": {
        "terminateNotificationProfile": {
          "notBeforeTimeout": "PT5M",
          "enable": true
        }
      }
    }
  }
}
```

The above block specifies a timeout delay of 5 minutes (as indicated by *PT5M*) for any terminate operation on all instances in your scale set. The field *notBeforeTimeout* can take any value between 5 and 15 minutes in ISO 8601 format. You can change the default timeout for the terminate operation by modifying the *notBeforeTimeout* property under *terminateNotificationProfile* described above.

After enabling *scheduledEventsProfile* on the scale set model and setting the *notBeforeTimeout*, update the individual instances to the [latest model](#) to reflect the changes.

NOTE

Terminate notifications on scale set instances can only be enabled with API version 2019-03-01 and above

Azure PowerShell

When creating a new scale set, you can enable termination notifications on the scale set by using the [New-AzVmssConfig](#) cmdlet.

This sample script walks through the creation of a scale set and associated resources using the configuration file: [Create a complete virtual machine scale set](#). You can provide configure terminate notification by adding the parameters *TerminateScheduledEvents* and *TerminateScheduledEventNotBeforeTimeoutInMinutes* to the configuration object for creating scale set. The following example enables the feature with a delay timeout of 10 minutes.

```
New-AzVmssConfig ` 
-Location "VMSSLocation" ` 
-SkuCapacity 2 ` 
-SkuName "Standard_DS2" ` 
-UpgradePolicyMode "Automatic" ` 
-TerminateScheduledEvents $true ` 
-TerminateScheduledEventNotBeforeTimeoutInMinutes 10
```

Use the [Update-AzVmss](#) cmdlet to enable termination notifications on an existing scale set.

```
Update-AzVmss ` 
-ResourceGroupName "myResourceGroup" ` 
-VMSScaleSetName "myScaleSet" ` 
-TerminateScheduledEvents $true ` 
-TerminateScheduledEventNotBeforeTimeoutInMinutes 15
```

The above example enables terminate notifications on an existing scale set and sets a 15-minute timeout for the terminate event.

After enabling scheduled events on the scale set model and setting the timeout, update the individual instances to the [latest model](#) to reflect the changes.

Azure CLI 2.0

The following example is for enabling termination notification while creating a new scale set.

```
az group create --name <myResourceGroup> --location <VMSSLocation>
az vmss create \
  --resource-group <myResourceGroup> \
  --name <myVMScaleSet> \
  --image UbuntuLTS \
  --admin-username <azureuser> \
  --generate-ssh-keys \
  --terminate-notification-time 10
```

The example above first creates a resource group, then creates a new scale set with terminate notifications enabled for a 10-minute default timeout.

The following example is for enabling termination notification in an existing scale set.

```
az vmss update \
  --resource-group <myResourceGroup> \
  --name <myVMScaleSet> \
  --enable-terminate-notification true \
  --terminate-notification-time 10
```

Get Terminate notifications

Terminate notifications are delivered through [Scheduled Events](#), which is an Azure Metadata Service. Azure Metadata service exposes information about running Virtual Machines using a REST Endpoint accessible from within the VM. The information is available via a non-routable IP so that it isn't exposed outside the VM.

Scheduled Events is enabled for your scale set the first time you make a request for events. You can expect a delayed response in your first call of up to two minutes. Query the endpoint periodically to detect upcoming maintenance events and the status of ongoing maintenance activities.

Scheduled Events is disabled for your scale set if the scale set instances don't make a request for 24 hours.

Endpoint discovery

For VNET enabled VMs, the Metadata Service is available from a static non-routable IP, 169.254.169.254.

The full endpoint for the latest version of Scheduled Events is:

```
'http://169.254.169.254/metadata/scheduledevents?api-version=2019-01-01'
```

Query response

A response contains an array of scheduled events. An empty array means that there are currently no events scheduled.

In the case where there are scheduled events, the response contains an array of events. For a "Terminate" event, the response will look as follows:

```
{
  "DocumentIncarnation": {IncarnationID},
  "Events": [
    {
      "EventId": {eventID},
      "EventType": "Terminate",
      "ResourceType": "VirtualMachine",
      "Resources": [{resourceName}],
      "EventStatus": "Scheduled",
      "NotBefore": {timeInUTC},
    }
  ]
}
```

The *DocumentIncarnation* is an ETag and provides an easy way to inspect if the Events payload has changed since the last query.

For more information on each of the fields above, see the Scheduled Events documentation for [Windows](#) and [Linux](#).

Respond to events

Once you've learnt of an upcoming event and completed your logic for graceful shutdown, you may approve the outstanding event by making a POST call to the metadata service with the EventId. The POST call indicates to Azure that it can continue with the VM delete.

Below is the json expected in the POST request body. The request should contain a list of StartRequests. Each StartRequest contains the EventId for the event you want to expedite:

```
{
  "StartRequests" : [
    {
      "EventId": {EventId}
    }
  ]
}
```

Ensure that every VM in the scale set is only approving the EventID relevant for that VM only. A VM can get its own VM name [through instance metadata](#). This name takes the form "{scale-set-name}_{instance-id}", and will be displayed in the 'Resources' section of the query response described above.

You can also refer to samples scripts for querying and responding to events using [PowerShell](#) and [Python](#).

Tips and best practices

- Terminate notifications only on 'delete' operations – All delete operations (manual delete or Autoscale-initiated scale-in) will generate Terminate events if your scale set has *scheduledEventsProfile* enabled. Other operations such as reboot, reimagine, redeploy, and stop/deallocate do not generate Terminate events. Terminate notifications can't be enabled for low-priority VMs.
- No mandatory wait for timeout – You can start the terminate operation at any time after the event has been received and before the event's *NotBefore* time expires.
- Mandatory delete at timeout – There isn't any capability of extending the timeout value after an event has been generated. Once the timeout expires, the pending terminate event will be processed and the VM will be deleted.
- Modifiable timeout value – You can modify the timeout value at any time before an instance is deleted, by modifying the *notBeforeTimeout* property on the scale set model and updating the VM instances to the latest model.
- Approve all pending deletes – If there's a pending delete on VM_1 that isn't approved, and you've approved

another terminate event on VM_2, then VM_2 isn't deleted until the terminate event for VM_1 is approved, or its timeout has elapsed. Once you approve the terminate event for VM_1, then both VM_1 and VM_2 are deleted.

- Approve all simultaneous deletes – Extending the above example, if VM_1 and VM_2 have the same *NotBefore* time, then both terminate events must be approved or neither VM is deleted before the timeout expires.

Troubleshoot

Failure to enable scheduledEventsProfile

If you get a 'BadRequest' error with an error message stating "Could not find member 'scheduledEventsProfile' on object of type 'VirtualMachineProfile'", check the API version used for the scale set operations. Compute API version **2019-03-01** or higher is required.

Failure to get Terminate events

If you are not getting any **Terminate** events through Scheduled Events, then check the API version used for getting the events. Metadata Service API version **2019-01-01** or higher is required for Terminate events.

```
'http://169.254.169.254/metadata/scheduledevents?api-version=2019-01-01'
```

Getting Terminate event with incorrect NotBefore time

After enabling *scheduledEventsProfile* on the scale set model and setting the *notBeforeTimeout*, update the individual instances to the [latest model](#) to reflect the changes.

Next steps

Learn how to [deploy your application](#) on virtual machine scale sets.

Shared Image Galleries overview

1/19/2020 • 17 minutes to read • [Edit Online](#)

Shared Image Gallery is a service that helps you build structure and organization around your managed images.

Shared Image Galleries provide:

- Managed global replication of images.
- Versioning and grouping of images for easier management.
- Highly available images with Zone Redundant Storage (ZRS) accounts in regions that support Availability Zones. ZRS offers better resilience against zonal failures.
- Sharing across subscriptions, and even between Active Directory (AD) tenants, using RBAC.
- Scaling your deployments with image replicas in each region.

Using a Shared Image Gallery you can share your images to different users, service principals, or AD groups within your organization. Shared images can be replicated to multiple regions, for quicker scaling of your deployments.

A managed image is a copy of either a full VM (including any attached data disks) or just the OS disk, depending on how you create the image. When you create a VM from the image, a copy of the VHDs in the image are used to create the disks for the new VM. The managed image remains in storage and can be used over and over again to create new VMs.

If you have a large number of managed images that you need to maintain and would like to make them available throughout your company, you can use a Shared Image Gallery as a repository that makes it easy to share your images.

The Shared Image Gallery feature has multiple resource types:

RESOURCE	DESCRIPTION
Managed image	A basic image that can be used alone or used to create an image version in an image gallery. Managed images are created from generalized VMs. A managed image is a special type of VHD that can be used to make multiple VMs and can now be used to create shared image versions.
Snapshot	A copy of a VHD that can be used to make an image version . Snapshots can be taken from a specialized VM (one that hasn't been generalized) then used alone or with snapshots of data disks, to create a specialized image version.
Image gallery	Like the Azure Marketplace, an image gallery is a repository for managing and sharing images, but you control who has access.
Image definition	Images are defined within a gallery and carry information about the image and requirements for using it within your organization. You can include information like whether the image is generalized or specialized, the operating system, minimum and maximum memory requirements, and release notes. It is a definition of a type of image.

RESOURCE	DESCRIPTION
Image version	An image version is what you use to create a VM when using a gallery. You can have multiple versions of an image as needed for your environment. Like a managed image, when you use an image version to create a VM, the image version is used to create new disks for the VM. Image versions can be used multiple times.

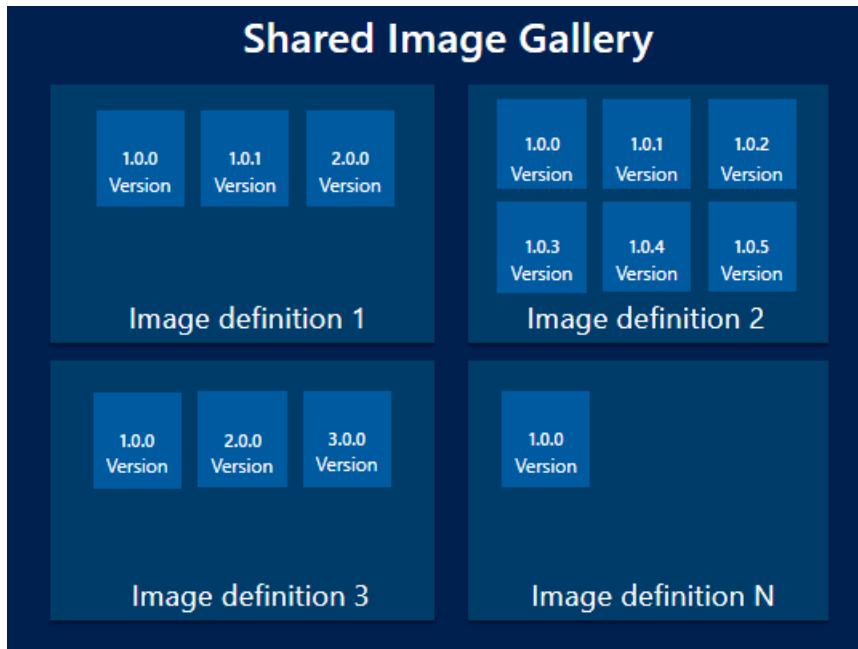


Image definitions

Image definitions are a logical grouping for versions of an image. The image definition holds information about why the image was created, what OS it is for, and information about using the image. An image definition is like a plan for all of the details around creating a specific image. You don't deploy a VM from an image definition, but from the image version created from the definition.

There are three parameters for each image definition that are used in combination - **Publisher**, **Offer** and **SKU**. These are used to find a specific image definition. You can have image versions that share one or two, but not all three values. For example, here are three image definitions and their values:

IMAGE DEFINITION	PUBLISHER	OFFER	SKU
myImage1	Contoso	Finance	Backend
myImage2	Contoso	Finance	Frontend
myImage3	Testing	Finance	Frontend

All three of these have unique sets of values. The format is similar to how you can currently specify publisher, offer, and SKU for [Azure Marketplace images](#) in Azure PowerShell to get the latest version of a Marketplace image. Each image definition needs to have a unique set of these values.

The following are other parameters that can be set on your image definition so that you can more easily track your resources:

- Operating system state - You can set the OS state to [generalized or specialized](#).

- Operating system - can be either Windows or Linux.
- Description - use description to give more detailed information on why the image definition exists. For example, you might have an image definition for your front-end server that has the application pre-installed.
- Eula - can be used to point to an end-user license agreement specific to the image definition.
- Privacy Statement and Release notes - store release notes and privacy statements in Azure storage and provide a URI for accessing them as part of the image definition.
- End-of-life date - attach an end-of-life date to your image definition to be able to use automation to delete old image definitions.
- Tag - you can add tags when you create your image definition. For more information about tags, see [Using tags to organize your resources](#)
- Minimum and maximum vCPU and memory recommendations - if your image has vCPU and memory recommendations, you can attach that information to your image definition.
- Disallowed disk types - you can provide information about the storage needs for your VM. For example, if the image isn't suited for standard HDD disks, you add them to the disallow list.

Generalized and specialized images

There are two operating system states supported by Shared Image Gallery. Typically images require that the VM used to create the image has been generalized before taking the image. Generalizing is a process that removes machine and user specific information from the VM. For Windows, the Sysprep tool is used. For Linux, you can use `waagent -deprovision` or `-deprovision+user` parameters.

Specialized VMs have not been through a process to remove machine specific information and accounts. Also, VMs created from specialized images do not have an `osProfile` associated with them. This means that specialized images will have some limitations.

- Accounts that could be used to log into the VM can also be used on any VM created using the specialized image that is created from that VM.
- VMs will have the **Computer name** of the VM the image was taken from. You should change the computer name to avoid collisions.
- The `osProfile` is how some sensitive information is passed to the VM, using `secrets`. This may cause issues using KeyVault, WinRM and other functionality that uses `secrets` in the `osProfile`. In some cases, you can use managed service identities (MSI) to work around these limitations.

IMPORTANT

Specialized images are currently in public preview. This preview version is provided without a service level agreement, and it's not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

Known preview limitations VMs can only be created from specialized images using the portal or API. There is no CLI or PowerShell support for the preview.

Regional Support

Source regions are listed in the table below. All public regions can be target regions, but to replicate to Australia Central and Australia Central 2 you need to have your subscription whitelisted. To request whitelisting, go to: <https://azure.microsoft.com/global-infrastructure/australia/contact/>

SOURCE REGIONS			
Australia Central	China East	South India	West Europe

SOURCE REGIONS			
Australia Central 2	China East 2	Southeast Asia	UK South
Australia East	China North	Japan East	UK West
Australia Southeast	China North 2	Japan West	US DoD Central
Brazil South	East Asia	Korea Central	US DoD East
Canada Central	East US	Korea South	US Gov Arizona
Canada East	East US 2	North Central US	US Gov Texas
Central India	East US 2 EUAP	North Europe	US Gov Virginia
Central US	France Central	South Central US	West India
Central US EUAP	France South	West Central US	West US
			West US 2

Limits

There are limits, per subscription, for deploying resources using Shared Image Galleries:

- 100 shared image galleries, per subscription, per region
- 1,000 image definitions, per subscription, per region
- 10,000 image versions, per subscription, per region
- Any disk attached to the image must be less than or equal to 1TB in size

For more information, see [Check resource usage against limits](#) for examples on how to check your current usage.

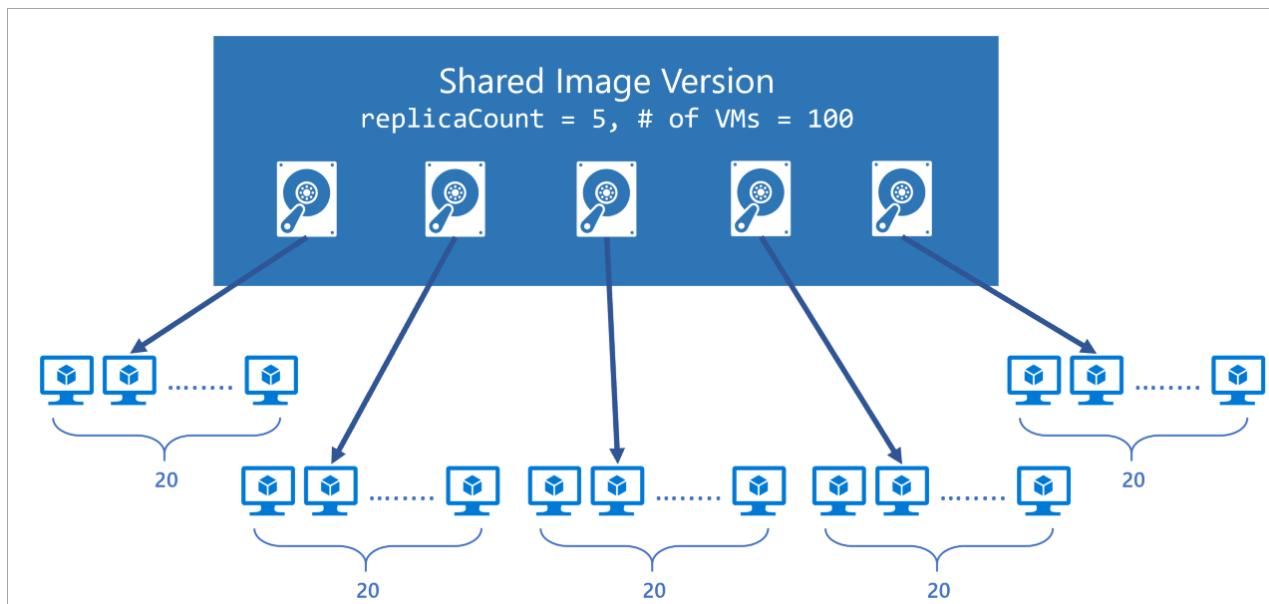
Scaling

Shared Image Gallery allows you to specify the number of replicas you want Azure to keep of the images. This helps in multi-VM deployment scenarios as the VM deployments can be spread to different replicas reducing the chance of instance creation processing being throttled due to overloading of a single replica.

With Shared Image Gallery, you can now deploy up to a 1,000 VM instances in a virtual machine scale set (up from 600 with managed images). Image replicas provide for better deployment performance, reliability and consistency. You can set a different replica count in each target region, based on the scale needs for the region. Since each replica is a deep copy of your image, this helps scale your deployments linearly with each extra replica. While we understand no two images or regions are the same, here's our general guideline on how to use replicas in a region:

- For non-Virtual Machine Scale Set (VMSS) Deployments - For every 20 VMs that you create concurrently, we recommend you keep one replica. For example, if you are creating 120 VMs concurrently using the same image in a region, we suggest you keep at least 6 replicas of your image.
- For Virtual Machine Scale Set (VMSS) deployments - For every scale set deployment with up to 600 instances, we recommend you keep at least one replica. For example, if you are creating 5 scale sets concurrently, each with 600 VM instances using the same image in a single region, we suggest you keep at least 5 replicas of your image.

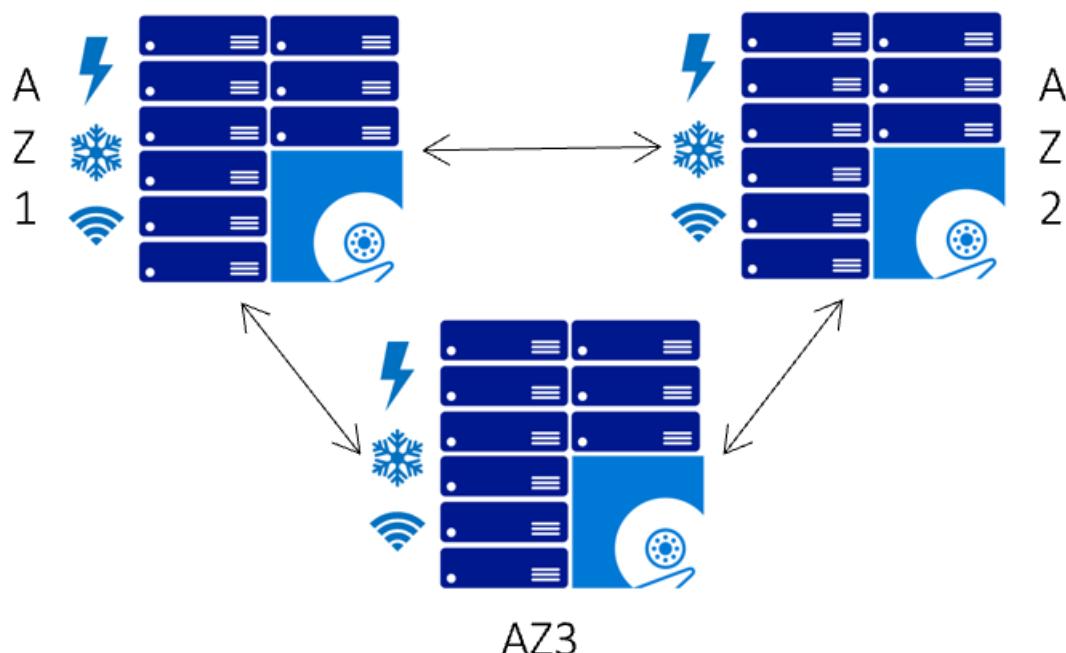
We always recommend you to overprovision the number of replicas due to factors like image size, content and OS type.



Make your images highly available

Azure Zone Redundant Storage (ZRS) provides resilience against an Availability Zone failure in the region. With the general availability of Shared Image Gallery, you can choose to store your images in ZRS accounts in regions with Availability Zones.

You can also choose the account type for each of the target regions. The default storage account type is Standard_LRS, but you can choose Standard_ZRS for regions with Availability Zones. Check the regional availability of ZRS [here](#).

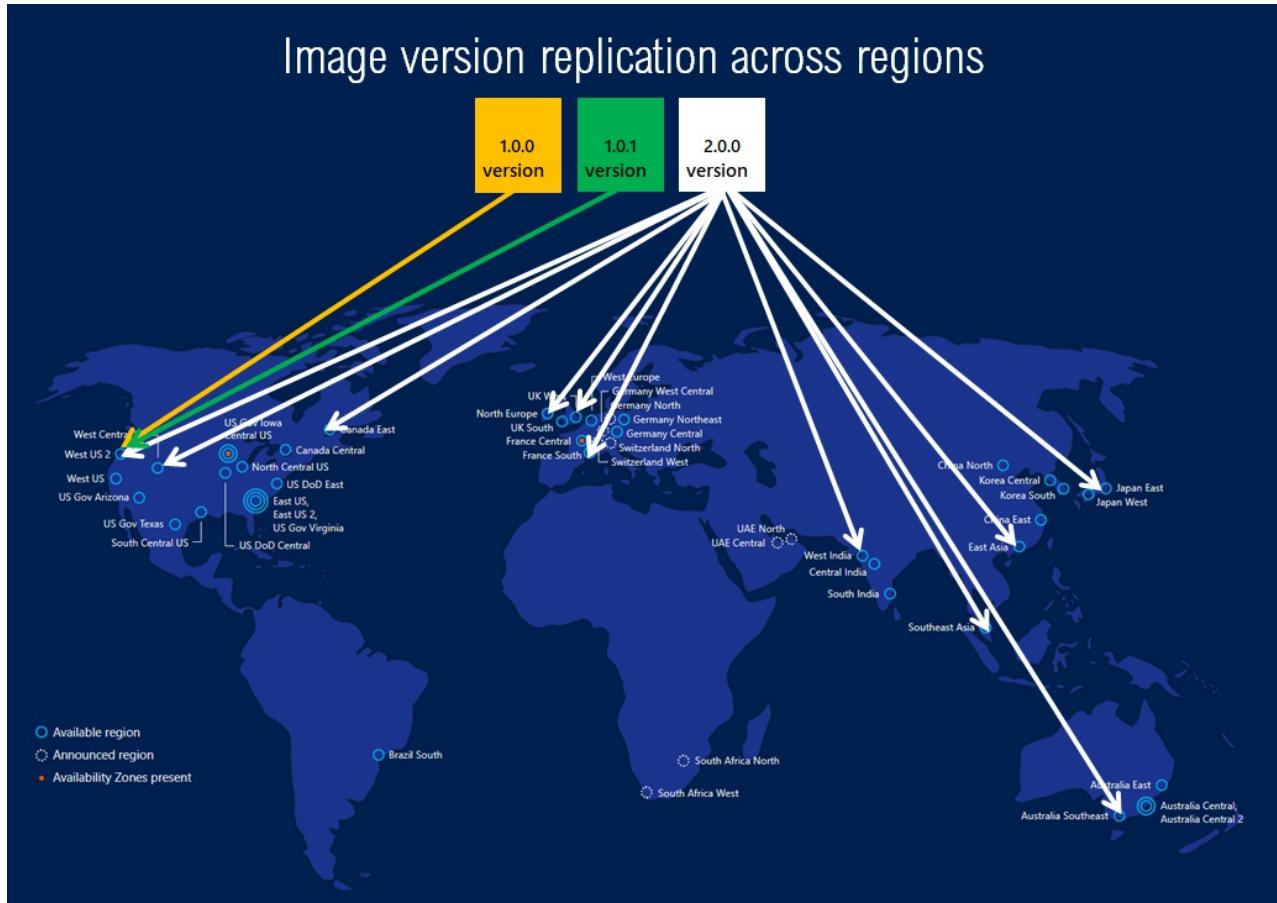


Replication

Shared Image Gallery also allows you to replicate your images to other Azure regions automatically. Each Shared

Image version can be replicated to different regions depending on what makes sense for your organization. One example is to always replicate the latest image in multi-regions while all older versions are only available in 1 region. This can help save on storage costs for Shared Image versions.

The regions a Shared Image version is replicated to can be updated after creation time. The time it takes to replicate to different regions depends on the amount of data being copied and the number of regions the version is replicated to. This can take a few hours in some cases. While the replication is happening, you can view the status of replication per region. Once the image replication is complete in a region, you can then deploy a VM or scale-set using that image version in the region.



Access

As the Shared Image Gallery, Image Definition, and Image version are all resources, they can be shared using the built-in native Azure RBAC controls. Using RBAC you can share these resources to other users, service principals, and groups. You can even share access to individuals outside of the tenant they were created within. Once a user has access to the Shared Image version, they can deploy a VM or a Virtual Machine Scale Set. Here is the sharing matrix that helps understand what the user gets access to:

SHARED WITH USER	SHARED IMAGE GALLERY	IMAGE DEFINITION	IMAGE VERSION
Shared Image Gallery	Yes	Yes	Yes
Image Definition	No	Yes	Yes

We recommend sharing at the Gallery level for the best experience. We do not recommend sharing individual image versions. For more information about RBAC, see [Manage access to Azure resources using RBAC](#).

Images can also be shared, at scale, even across tenants using a multi-tenant app registration. For more information about sharing images across tenants, see [Share gallery VM images across Azure tenants](#).

Billing

There is no extra charge for using the Shared Image Gallery service. You will be charged for the following resources:

- Storage costs of storing the Shared Image versions. Cost depends on the number of replicas of the image version and the number of regions the version is replicated to. For example, if you have 2 images and both are replicated to 3 regions, then you will be charged for 6 managed disks based on their size. For more information, see [Managed Disks pricing](#).
- Network egress charges for replication of the first image version from the source region to the replicated regions. Subsequent replicas are handled within the region, so there are no additional charges.

Updating resources

Once created, you can make some changes to the image gallery resources. These are limited to:

Shared image gallery:

- Description

Image definition:

- Recommended vCPUs
- Recommended memory
- Description
- End of life date

Image version:

- Regional replica count
- Target regions
- Exclude from latest
- End of life date

SDK support

The following SDKs support creating Shared Image Galleries:

- [.NET](#)
- [Java](#)
- [Node.js](#)
- [Python](#)
- [Go](#)

Templates

You can create Shared Image Gallery resource using templates. There are several Azure Quickstart Templates available:

- [Create a Shared Image Gallery](#)
- [Create an Image Definition in a Shared Image Gallery](#)
- [Create an Image Version in a Shared Image Gallery](#)
- [Create a VM from Image Version](#)

Frequently asked questions

- [How can I list all the Shared Image Gallery resources across subscriptions?](#)
- [Can I move my existing image to the shared image gallery?](#)
- [Can I create an image version from a specialized disk?](#)
- [Can I move the Shared Image Gallery resource to a different subscription after it has been created?](#)
- [Can I replicate my image versions across clouds such as Azure China 21Vianet or Azure Germany or Azure Government Cloud?](#)
- [Can I replicate my image versions across subscriptions?](#)
- [Can I share image versions across Azure AD tenants?](#)
- [How long does it take to replicate image versions across the target regions?](#)
- [What is the difference between source region and target region?](#)
- [How do I specify the source region while creating the image version?](#)
- [How do I specify the number of image version replicas to be created in each region?](#)
- [Can I create the shared image gallery in a different location than the one for the image definition and image version?](#)
- [What are the charges for using the Shared Image Gallery?](#)
- [What API version should I use to create Shared Image Gallery and Image Definition and Image Version?](#)
- [What API version should I use to create Shared VM or Virtual Machine Scale Set out of the Image Version?](#)

How can I list all the Shared Image Gallery resources across subscriptions?

To list all the Shared Image Gallery resources across subscriptions that you have access to on the Azure portal, follow the steps below:

1. Open the [Azure portal](#).
2. Go to **All Resources**.
3. Select all the subscriptions under which you'd like to list all the resources.
4. Look for resources of type **Private gallery**.

To see the image definitions and image versions, you should also select **Show hidden types**.

To list all the Shared Image Gallery resources across subscriptions that you have permissions to, use the following command in the Azure CLI:

```
az account list -otsv --query "[].id" | xargs -n 1 az sig list --subscription
```

Can I move my existing image to the shared image gallery?

Yes. There are 3 scenarios based on the types of images you may have.

Scenario 1: If you have a managed image in the same subscription as your SIG, then you can create an image definition and image version from it.

Scenario 2: If you have an unmanaged image in the same subscription as your SIG, you can create a managed image from it, and then create an image definition and image version from it.

Scenario 3: If you have a VHD in your local file system, then you need to upload the VHD to a managed image, then you can create an image definition and image version from it.

- If the VHD is of a Windows VM, see [Upload a VHD](#).
- If the VHD is for a Linux VM, see [Upload a VHD](#)

Can I create an image version from a specialized disk?

Yes, support for specialized disks as images is in preview. You can only create a VM from a specialized image using the portal ([Windows](#) or [Linux](#)) and API. There is no PowerShell support for the preview.

Can I move the Shared Image Gallery resource to a different subscription after it has been created?

No, you cannot move the shared image gallery resource to a different subscription. However, you will be able to replicate the image versions in the gallery to other regions as required.

Can I replicate my image versions across clouds such as Azure China 21Vianet or Azure Germany or Azure Government Cloud?

No, you cannot replicate image versions across clouds.

Can I replicate my image versions across subscriptions?

No, you may replicate the image versions across regions in a subscription and use it in other subscriptions through RBAC.

Can I share image versions across Azure AD tenants?

Yes, you can use RBAC to share to individuals across tenants. But, to share at scale, see "Share gallery images across Azure tenants" using [PowerShell](#) or [CLI](#).

How long does it take to replicate image versions across the target regions?

The image version replication time is entirely dependent on the size of the image and the number of regions it is being replicated to. However, as a best practice, it is recommended that you keep the image small, and the source and target regions close for best results. You can check the status of the replication using the -ReplicationStatus flag.

What is the difference between source region and target region?

Source region is the region in which your image version will be created, and target regions are the regions in which a copy of your image version will be stored. For each image version, you can only have one source region. Also, make sure that you pass the source region location as one of the target regions when you create an image version.

How do I specify the source region while creating the image version?

While creating an image version, you can use the **--location** tag in CLI and the **-Location** tag in PowerShell to specify the source region. Please ensure the managed image that you are using as the base image to create the image version is in the same location as the location in which you intend to create the image version. Also, make sure that you pass the source region location as one of the target regions when you create an image version.

How do I specify the number of image version replicas to be created in each region?

There are two ways you can specify the number of image version replicas to be created in each region:

1. The regional replica count which specifies the number of replicas you want to create per region.
2. The common replica count which is the default per region count in case regional replica count is not specified.

To specify the regional replica count, pass the location along with the number of replicas you want to create in that region: "South Central US=2".

If regional replica count is not specified with each location, then the default number of replicas will be the common replica count that you specified.

To specify the common replica count in CLI, use the **--replica-count** argument in the `az sig image-version create` command.

Can I create the shared image gallery in a different location than the one for the image definition and image version?

Yes, it is possible. But, as a best practice, we encourage you to keep the resource group, shared image gallery, image definition, and image version in the same location.

What are the charges for using the Shared Image Gallery?

There are no charges for using the Shared Image Gallery service, except the storage charges for storing the image versions and network egress charges for replicating the image versions from source region to target regions.

What API version should I use to create Shared Image Gallery and Image Definition and Image Version?

To work with shared image galleries, image definitions, and image versions, we recommend you use API version 2018-06-01. Zone Redundant Storage (ZRS) requires version 2019-03-01 or later.

What API version should I use to create Shared VM or Virtual Machine Scale Set out of the Image Version?

For VM and Virtual Machine Scale Set deployments using an image version, we recommend you use API version 2018-04-01 or higher.

Next steps

Learn how to deploy shared images using the [Azure CLI](#) and [Azure PowerShell](#)

Create and use shared images for virtual machine scale sets with the Azure CLI 2.0

1/19/2020 • 5 minutes to read • [Edit Online](#)

When you create a scale set, you specify an image to be used when the VM instances are deployed. [Shared Image Galleries](#) greatly simplifies custom image sharing across your organization. Custom images are like marketplace images, but you create them yourself. Custom images can be used to bootstrap configurations such as preloading applications, application configurations, and other OS configurations. The Shared Image Gallery lets you share your custom VM images with others in your organization, within or across regions, within an AAD tenant. Choose which images you want to share, which regions you want to make them available in, and who you want to share them with. You can create multiple galleries so that you can logically group shared images. The gallery is a top-level resource that provides full role-based access control (RBAC). Images can be versioned, and you can choose to replicate each image version to a different set of Azure regions. The gallery only works with Managed Images.

NOTE

This article walks through the process of using a generalized managed image. It is not supported to create a scale set from a specialized VM image.

Before you begin

To complete the example in this article, you must have an existing managed image of a generalized VM. For more information, see [Tutorial: Create a custom image of an Azure VM with the Azure CLI](#). If the managed image contains a data disk, the data disk size cannot be more than 1 TB.

Launch Azure Cloud Shell

The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account.

To open the Cloud Shell, just select **Try it** from the upper right corner of a code block. You can also launch Cloud Shell in a separate browser tab by going to <https://shell.azure.com/bash>. Select **Copy** to copy the blocks of code, paste it into the Cloud Shell, and press enter to run it.

If you prefer to install and use the CLI locally, see [Install Azure CLI](#).

Create an image gallery

An image gallery is the primary resource used for enabling image sharing. Allowed characters for Gallery name are uppercase or lowercase letters, digits, dots, and periods. The gallery name cannot contain dashes. Gallery names must be unique within your subscription.

Create an image gallery using `az sig create`. The following example creates a gallery named *myGallery* in *myGalleryRG*.

```
az group create --name myGalleryRG --location WestCentralUS
az sig create --resource-group myGalleryRG --gallery-name myGallery
```

Create an image definition

Image definitions create a logical grouping for images. They are used to manage information about the image versions that are created within them. Image definition names can be made up of uppercase or lowercase letters, digits, dots, dashes, and periods. For more information about the values you can specify for an image definition, see [Image definitions](#).

Create an initial image definition in the gallery using [az sig image-definition create](#).

```
az sig image-definition create \
--resource-group myGalleryRG \
--gallery-name myGallery \
--gallery-image-definition myImageDefinition \
--publisher myPublisher \
--offer myOffer \
--sku 16.04-LTS \
--os-type Linux
```

Create an image version

Create versions of the image as needed using [az image gallery create-image-version](#). You will need to pass in the ID of the managed image to use as a baseline for creating the image version. You can use [az image list](#) to get information about images that are in a resource group.

Allowed characters for image version are numbers and periods. Numbers must be within the range of a 32-bit integer. Format: *MajorVersion.MinorVersion.Patch*.

In this example, the version of our image is *1.0.0* and we are going to create 2 replicas in the *West Central US* region, 1 replica in the *South Central US* region and 1 replica in the *East US 2* region using zone-redundant storage.

```
az sig image-version create \
--resource-group myGalleryRG \
--gallery-name myGallery \
--gallery-image-definition myImageDefinition \
--gallery-image-version 1.0.0 \
--target-regions "WestCentralUS" "SouthCentralUS=1" "EastUS2=1=Standard_ZRS" \
--replica-count 2 \
--managed-image "/subscriptions/<subscription
ID>/resourceGroups/myResourceGroup/providers/Microsoft.Compute/images/myImage"
```

NOTE

You need to wait for the image version to completely finish being built and replicated before you can use the same managed image to create another image version.

You can also store all of your image version replicas in [Zone Redundant Storage](#) by adding

```
--storage-account-type standard_zrs
```

Share the gallery

We recommend that you share with other users at the gallery level. To get the object ID of your gallery, use [az sig show](#).

```
az sig show \
--resource-group myGalleryRG \
--gallery-name myGallery \
--query id
```

Use the object ID as a scope, along with an email address and [az role assignment create](#) to give a user access to the shared image gallery.

```
az role assignment create --role "Reader" --assignee <email address> --scope <gallery ID>
```

Create a scale set from the custom VM image

Create a scale set with [az vmss create](#). Instead of a platform image, such as *UbuntuLTS* or *CentOS*, specify the name of your custom VM image. The following example creates a scale set named *myScaleSet* that uses the custom image named *myImage* from the previous step:

```
az vmss create \
-g myGalleryRG \
-n myScaleSet \
--image "/subscriptions/<subscription
ID>/resourceGroups/myGalleryRG/providers/Microsoft.Compute/galleries/myGallery/images/myImageDefinition/version/1.0.0" \
--admin-username azureuser \
--generate-ssh-keys
```

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

Using RBAC to share images

You can share images across subscriptions using Role-Based Access Control (RBAC). Any user that has read permissions to an image version, even across subscriptions, will be able to deploy a Virtual Machine using the image version.

For more information about how to share resources using RBAC, see [Manage access using RBAC and Azure CLI](#).

List information

Get the location, status and other information about the available image galleries using [az sig list](#).

```
az sig list -o table
```

List the image definitions in a gallery, including information about OS type and status, using [az sig image-definition list](#).

```
az sig image-definition list --resource-group myGalleryRG --gallery-name myGallery -o table
```

List the shared image versions in a gallery, using [az sig image-version list](#).

```
az sig image-version list --resource-group myGalleryRG --gallery-name myGallery --gallery-image-definition
myImageDefinition -o table
```

Get the ID of an image version using [az sig image-version show](#).

```
az sig image-version show \
--resource-group myGalleryRG \
--gallery-name myGallery \
--gallery-image-definition myImageDefinition \
--gallery-image-version 1.0.0 \
--query "id"
```

Clean up resources

To remove your scale set and additional resources, delete the resource group and all its resources with [az group delete](#). The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

```
az group delete --name myResourceGroup --no-wait --yes
```

Next steps

You can also create Shared Image Gallery resource using templates. There are several Azure Quickstart Templates available:

- [Create a Shared Image Gallery](#)
- [Create an Image Definition in a Shared Image Gallery](#)
- [Create an Image Version in a Shared Image Gallery](#)
- [Create a VM from Image Version](#)

If you run into any issues, you can [troubleshoot shared image galleries](#).

Create and use shared images for virtual machine scale sets with the Azure PowerShell

2/14/2020 • 9 minutes to read • [Edit Online](#)

When you create a scale set, you specify an image to be used when the VM instances are deployed. The Shared Image Gallery service greatly simplifies custom image sharing across your organization. Custom images are like marketplace images, but you create them yourself. Custom images can be used to bootstrap configurations such as preloading applications, application configurations, and other OS configurations.

The Shared Image Gallery lets you share your custom VM images with others in your organization, within or across regions, within an AAD tenant. Choose which images you want to share, which regions you want to make them available in, and who you want to share them with. You can create multiple galleries so that you can logically group shared images.

The gallery is a top-level resource that provides full role-based access control (RBAC). Images can be versioned, and you can choose to replicate each image version to a different set of Azure regions. The gallery only works with Managed Images.

The Shared Image Gallery feature has multiple resource types. We will be using or building these in this article:

RESOURCE	DESCRIPTION
Managed image	This is a basic image that can be used alone or used to create an image version in an image gallery. Managed images are created from generalized VMs. A managed image is a special type of VHD that can be used to make multiple VMs and can now be used to create shared image versions.
Image gallery	Like the Azure Marketplace, an image gallery is a repository for managing and sharing images, but you control who has access.
Image definition	Images are defined within a gallery and carry information about the image and requirements for using it internally. This includes whether the image is Windows or Linux, release notes, and minimum and maximum memory requirements. It is a definition of a type of image.
Image version	An image version is what you use to create a VM when using a gallery. You can have multiple versions of an image as needed for your environment. Like a managed image, when you use an image version to create a VM, the image version is used to create new disks for the VM. Image versions can be used multiple times.

If you don't have an Azure subscription, create a [free account](#) before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

The steps below detail how to take an existing VM and turn it into a reusable custom image that you can use to create new VM instances.

To complete the example in this article, you must have an existing managed image. You can follow [Tutorial: Create and use a custom image for virtual machine scale sets with Azure PowerShell](#) to create one if needed. If the managed image contains a data disk, the data disk size cannot be more than 1 TB.

When working through the article, replace the resource group and VM names where needed.

Launch Azure Cloud Shell

The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account.

To open the Cloud Shell, just select **Try it** from the upper right corner of a code block. You can also launch Cloud Shell in a separate browser tab by going to <https://shell.azure.com/powershell>. Select **Copy** to copy the blocks of code, paste it into the Cloud Shell, and press enter to run it.

Get the managed image

You can see a list of images that are available in a resource group using [Get-AzImage](#). Once you know the image name and what resource group it is in, you can use `Get-AzImage` again to get the image object and store it in a variable to use later. This example gets an image named *myImage* from the "myResourceGroup" resource group and assigns it to the variable `$managedImage`.

```
$managedImage = Get-AzImage `  
-ImageName myImage `  
-ResourceGroupName myResourceGroup
```

Create an image gallery

An image gallery is the primary resource used for enabling image sharing. Allowed characters for Gallery name are uppercase or lowercase letters, digits, dots, and periods. The gallery name cannot contain dashes. Gallery names must be unique within your subscription.

Create an image gallery using [New-AzGallery](#). The following example creates a gallery named *myGallery* in the *myGalleryRG* resource group.

```
$resourceGroup = New-AzResourceGroup ` 
    -Name 'myGalleryRG' ` 
    -Location 'West Central US' 
$gallery = New-AzGallery ` 
    -GalleryName 'myGallery' ` 
    -ResourceGroupName $resourceGroup.ResourceGroupName ` 
    -Location $resourceGroup.Location ` 
    -Description 'Shared Image Gallery for my organization'
```

Create an image definition

Image definitions create a logical grouping for images. They are used to manage information about the image versions that are created within them. Image definition names can be made up of uppercase or lowercase letters, digits, dots, dashes and periods. For more information about the values you can specify for an image definition, see [Image definitions](#).

Create the image definition using [New-AzGalleryImageDefinition](#). In this example, the gallery image is named *myGalleryImage*.

```
$galleryImage = New-AzGalleryImageDefinition ` 
    -GalleryName $gallery.Name ` 
    -ResourceGroupName $resourceGroup.ResourceGroupName ` 
    -Location $gallery.Location ` 
    -Name 'myImageDefinition' ` 
    -OsState generalized ` 
    -OsType Windows ` 
    -Publisher 'myPublisher' ` 
    -Offer 'myOffer' ` 
    -Sku 'mySKU'
```

Create an image version

Create an image version from a managed image using [New-AzGalleryImageVersion](#).

Allowed characters for image version are numbers and periods. Numbers must be within the range of a 32-bit integer. Format: *MajorVersion.MinorVersion.Patch*.

In this example, the image version is *1.0.0* and it's replicated to both *West Central US* and *South Central US* datacenters. When choosing target regions for replication, remember that you also have to include the *source* region as a target for replication.

```
$region1 = @{Name='South Central US';ReplicaCount=1} 
$region2 = @{Name='West Central US';ReplicaCount=2} 
$targetRegions = @($region1,$region2) 
$job = $imageVersion = New-AzGalleryImageVersion ` 
    -GalleryImageDefinitionName $galleryImage.Name ` 
    -GalleryImageVersionName '1.0.0' ` 
    -GalleryName $gallery.Name ` 
    -ResourceGroupName $resourceGroup.ResourceGroupName ` 
    -Location $resourceGroup.Location ` 
    -TargetRegion $targetRegions ` 
    -Source $managedImage.Id.ToString() ` 
    -PublishingProfileEndOfLifeDate '2020-01-01' ` 
    -asJob
```

It can take a while to replicate the image to all of the target regions, so we have created a job so we can track the progress. To see the progress of the job, type `$job.State`.

```
$job.State
```

NOTE

You need to wait for the image version to completely finish being built and replicated before you can use the same managed image to create another image version.

You can also store your image version in [Zone Redundant Storage](#) by adding `-StorageAccountType Standard_ZRS` when you create the image version.

Share the gallery

We recommend that you share access at the image gallery level. Use an email address and the [Get-AzADUser](#) cmdlet to get the object ID for the user, then use [New-AzRoleAssignment](#) to give them access to the gallery. Replace the example email, alinne_montes@contoso.com in this example, with your own information.

```
# Get the object ID for the user
$user = Get-AzADUser -StartsWith alinne_montes@contoso.com
# Grant access to the user for our gallery
New-AzRoleAssignment ` 
    -ObjectId $user.Id ` 
    -RoleDefinitionName Reader ` 
    -ResourceName $gallery.Name ` 
    -ResourceType Microsoft.Compute/galleries ` 
    -ResourceGroupName $resourceGroup.ResourceGroupName
```

Create a scale set from the shared image version

Create a virtual machine scale set with [New-AzVmss](#). The following example creates a scale set from the new image version in the *South Central US* datacenter. When prompted, set your own administrative credentials for the VM instances in the scale set:

```
# Define variables
$resourceGroupName = "myVMSSRG"
$scaleSetName = "myScaleSet"
$location = "South Central US"
$cred = Get-Credential

# Create a resource group
New-AzResourceGroup -ResourceGroupName $resourceGroupName -Location $location

# Create a netowrking pieces
$subnet = New-AzVirtualNetworkSubnetConfig ` 
    -Name "mySubnet" ` 
    -AddressPrefix 10.0.0.0/24
$vnet = New-AzVirtualNetwork ` 
    -ResourceGroupName $resourceGroupName ` 
    -Name "myVnet" ` 
    -Location $location ` 
    -AddressPrefix 10.0.0.0/16 ` 
    -Subnet $subnet
$publicIP = New-AzPublicIpAddress ` 
    -ResourceGroupName $resourceGroupName ` 
    -Location $location ` 
    -AllocationMethod Static ` 
    -Name "myPublicIP"
$frontendIP = New-AzLoadBalancerFrontendIpConfig ` 
    -Name "myFrontEndPool" ` 
    -PublicIpAddress $publicIP
```

```

$backendPool = New-AzLoadBalancerBackendAddressPoolConfig -Name "myBackEndPool"
$inboundNATPool = New-AzLoadBalancerInboundNatPoolConfig ` 
    -Name "myRDPRule" ` 
    -FrontendIpConfigurationId $frontendIP.Id ` 
    -Protocol TCP ` 
    -FrontendPortRangeStart 50001 ` 
    -FrontendPortRangeEnd 50010 ` 
    -BackendPort 3389
# Create the load balancer and health probe
$lb = New-AzLoadBalancer ` 
    -ResourceGroupName $resourceGroupName ` 
    -Name "myLoadBalancer" ` 
    -Location $location ` 
    -FrontendIpConfiguration $frontendIP ` 
    -BackendAddressPool $backendPool ` 
    -InboundNatPool $inboundNATPool
Add-AzLoadBalancerProbeConfig -Name "myHealthProbe" ` 
    -LoadBalancer $lb ` 
    -Protocol TCP ` 
    -Port 80 ` 
    -IntervalInSeconds 15 ` 
    -ProbeCount 2
Add-AzLoadBalancerRuleConfig ` 
    -Name "myLoadBalancerRule" ` 
    -LoadBalancer $lb ` 
    -FrontendIpConfiguration $lb.FrontendIpConfigurations[0] ` 
    -BackendAddressPool $lb.BackendAddressPools[0] ` 
    -Protocol TCP ` 
    -FrontendPort 80 ` 
    -BackendPort 80 ` 
    -Probe (Get-AzLoadBalancerProbeConfig -Name "myHealthProbe" -LoadBalancer $lb)
Set-AzLoadBalancer -LoadBalancer $lb

# Create IP address configurations
$ipConfig = New-AzVmssIpConfig ` 
    -Name "myIPConfig" ` 
    -LoadBalancerBackendAddressPoolsId $lb.BackendAddressPools[0].Id ` 
    -LoadBalancerInboundNatPoolsId $inboundNATPool.Id ` 
    -SubnetId $vnet.Subnets[0].Id

# Create a configuration
$vmssConfig = New-AzVmssConfig ` 
    -Location $location ` 
    -SkuCapacity 2 ` 
    -SkuName "Standard_DS2" ` 
    -UpgradePolicyMode "Automatic"

# Reference the image version
Set-AzVmssStorageProfile $vmssConfig ` 
    -OsDiskCreateOption "FromImage" ` 
    -ImageReferenceId $imageVersion.Id

# Complete the configuration
Set-AzVmssOsProfile $vmssConfig ` 
    -AdminUsername $cred.UserName ` 
    -AdminPassword $cred.Password ` 
    -ComputerNamePrefix "myVM"
Add-AzVmssNetworkInterfaceConfiguration ` 
    -VirtualMachineScaleSet $vmssConfig ` 
    -Name "network-config" ` 
    -Primary $true ` 
    -IPConfiguration $ipConfig

# Create the scale set
New-AzVmss ` 
    -ResourceGroupName $resourceGroupName ` 
    -Name $scaleSetName ` 
    -VirtualMachineScaleSet $vmssConfig

```

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

Shared image management

Here are some examples of common management tasks and how to complete them using PowerShell.

List all galleries by name.

```
$galleries = Get-AzResource -ResourceType Microsoft.Compute/galleries  
$galleries.Name
```

List all image definitions by name.

```
$imageDefinitions = Get-AzResource -ResourceType Microsoft.Compute/galleries/images  
$imageDefinitions.Name
```

List all image versions by name.

```
$imageVersions = Get-AzResource -ResourceType Microsoft.Compute/galleries/images/versions  
$imageVersions.Name
```

Delete an image version. This example deletes the image version named *1.0.0*.

```
Remove-AzGalleryImageVersion `  
-GalleryImageDefinitionName myImageDefinition `  
-GalleryName myGallery `  
-Name 1.0.0 `  
-ResourceGroupName myGalleryRG
```

Update resources

There are some limitations on what can be updated. The following items can be updated:

Shared image gallery:

- Description

Image definition:

- Recommended vCPUs
- Recommended memory
- Description
- End of life date

Image version:

- Regional replica count
- Target regions
- Exclusion from latest
- End of life date

If you plan on adding replica regions, do not delete the source managed image. The source managed image is needed for replicating the image version to additional regions.

To update the description of a gallery, use [Update-AzGallery](#).

```
Update-AzGallery ` 
-Name $gallery.Name ` 
-ResourceGroupName $resourceGroup.Name
```

This example shows how to use [Update-AzGalleryImageDefinition](#) to update the end-of-life date for our image definition.

```
Update-AzGalleryImageDefinition ` 
-GalleryName $gallery.Name ` 
-Name $galleryImage.Name ` 
-ResourceGroupName $resourceGroup.Name ` 
-EndOfLifeDate 01/01/2030
```

This example shows how to use [Update-AzGalleryImageVersion](#) to exclude this image version from being used as the *latest* image.

```
Update-AzGalleryImageVersion ` 
-GalleryImageDefinitionName $galleryImage.Name ` 
-GalleryName $gallery.Name ` 
-Name $galleryVersion.Name ` 
-ResourceGroupName $resourceGroup.Name ` 
-PublishingProfileExcludeFromLatest
```

Clean up resources

When deleting resources, you need to start with last item in the nested resources - the image version. Once versions are deleted, you can delete the image definition. You can't delete the gallery until all resources beneath it have been deleted.

```
$resourceGroup = "myResourceGroup"
$gallery = "myGallery"
$imageDefinition = "myImageDefinition"
$imageVersion = "myImageVersion"

Remove-AzGalleryImageVersion ` 
-GalleryImageDefinitionName $imageDefinition ` 
-GalleryName $gallery ` 
-Name $imageVersion ` 
-ResourceGroupName $resourceGroup

Remove-AzGalleryImageDefinition ` 
-ResourceGroupName $resourceGroup ` 
-GalleryName $gallery ` 
-GalleryImageDefinitionName $imageDefinition

Remove-AzGallery ` 
-Name $gallery ` 
-ResourceGroupName $resourceGroup

Remove-AzResourceGroup -Name $resourceGroup
```

Next steps

You can also create Shared Image Gallery resource using templates. There are several Azure Quickstart Templates available:

- [Create a Shared Image Gallery](#)
- [Create an Image Definition in a Shared Image Gallery](#)
- [Create an Image Version in a Shared Image Gallery](#)
- [Create a VM from Image Version](#)

For more information about Shared Image Galleries, see the [Overview](#). If you run into issues, see [Troubleshooting shared image galleries](#).

Share gallery VM images across Azure tenants

1/19/2020 • 3 minutes to read • [Edit Online](#)

But, if you want to share images outside of your Azure tenant, at scale, you should create an app registration to facilitate sharing. Using an app registration can enable more complex sharing scenarios, like:

- Managing shared images when one company acquires another, and the Azure infrastructure is spread across separate tenants.
- Azure Partners manage Azure infrastructure on behalf of their customers. Customization of images is done within the partners tenant, but the infrastructure deployments will happen in the customer's tenant.

Create the app registration

Create an application registration that will be used by both tenants to share the image gallery resources.

1. Open the [App registrations \(preview\) in the Azure portal](#).
2. Select **New registration** from the menu at the top of the page.
3. In **Name**, type *myGalleryApp*.
4. In **Supported account types**, select **Accounts in any organizational directory and personal Microsoft accounts**.
5. In **Redirect URI**, type <https://www.microsoft.com> and then select **Register**. After the app registration has been created, the overview page will open.
6. On the overview page, copy the **Application (client) ID** and save for use later.
7. Select **Certificates & secrets**, and then select **New client secret**.
8. In **Description**, type *Shared image gallery cross-tenant app secret*.
9. In **Expires**, leave the default of **In 1 year** and then select **Add**.
10. Copy the value of the secret and save it to a safe place. You cannot retrieve it after you leave the page.

Give the app registration permission to use the shared image gallery.

1. In the Azure portal, select the Shared Image Gallery that you want to share with another tenant.
2. Select **select Access control (IAM)**, and under **Add role assignment** select **Add**.
3. Under **Role**, select **Reader**.
4. Under **Assign access to**, leave this as **Azure AD user, group, or service principal**.
5. Under **Select**, type *myGalleryApp* and select it when it shows up in the list. When you are done, select **Save**.

Give Tenant 2 access

Give Tenant 2 access to the application by requesting a sign-in using a browser. Replace <*Tenant2 ID*> with the tenant ID for the tenant that you would like to share your image gallery with. Replace <*Application (client) ID*> with the application ID of the app registration you created. When done making the replacements, paste the URL into a browser and follow the sign-in prompts to sign into Tenant 2.

```
https://login.microsoftonline.com/<Tenant 2 ID>/oauth2/authorize?client_id=<Application (client) ID>&response_type=code&redirect_uri=https%3A%2F%2Fwww.microsoft.com%2F
```

In the [Azure portal](#) sign in as Tenant 2 and give the app registration access to the resource group where you want to create the VM.

1. Select the resource group and then select **Access control (IAM)**. Under **Add role assignment** select **Add**.
2. Under **Role**, type **Contributor**.
3. Under **Assign access to**, leave this as **Azure AD user, group, or service principal**.
4. Under **Select** type *myGalleryApp* then select it when it shows up in the list. When you are done, select **Save**.

NOTE

You need to wait for the image version to completely finish being built and replicated before you can use the same managed image to create another image version.

Create a scale set using Azure CLI

Sign in the service principal for tenant 1 using the appID, the app key, and the ID of tenant 1. You can use

```
az account show --query "tenantId"
```

 to get the tenant IDs if needed.

```
az account clear  
az login --service-principal -u '<app ID>' -p '<Secret>' --tenant '<tenant 1 ID>'  
az account get-access-token
```

Sign in the service principal for tenant 2 using the appID, the app key, and the ID of tenant 2:

```
az login --service-principal -u '<app ID>' -p '<Secret>' --tenant '<tenant 2 ID>'  
az account get-access-token
```

Create the scale set. Replace the information in the example with your own.

```
az vmss create \  
-g myResourceGroup \  
-n myScaleSet \  
--image "/subscriptions/<Tenant 1 subscription>/resourceGroups/<Resource  
group>/providers/Microsoft.Compute/galleries/<Gallery>/images/<Image definition>/versions/<version>" \  
--admin-username azureuser \  
--generate-ssh-keys
```

Next steps

If you run into any issues, you can [troubleshoot shared image galleries](#).

Troubleshoot shared image galleries

1/19/2020 • 3 minutes to read • [Edit Online](#)

If you run into issues while performing any operations on shared image galleries, image definitions, and image versions, run the failing command again in debug mode. Debug mode is activated by passing the **-debug** switch with CLI and the **-Debug** switch with PowerShell. Once you've located the error, follow this document to troubleshoot the errors.

Unable to create a shared image gallery

Possible causes:

The gallery name is invalid.

Allowed characters for Gallery name are uppercase or lowercase letters, digits, dots, and periods. The gallery name cannot contain dashes. Change the gallery name and try again.

The gallery name is not unique within your subscription.

Pick another gallery name and try again.

Unable to create an image definition

Possible causes:

image definition name is invalid.

Allowed characters for image definition are uppercase or lowercase letters, digits, dots, dashes, and periods. Change the image definition name and try again.

The mandatory properties for creating an image definition are not populated.

The properties such as name, publisher, offer, sku, and OS type are mandatory. Verify if all the properties are being passed.

Make sure that the **OSType**, either Linux or Windows, of the image definition is the same as the source managed image that you are using to create the image version.

Unable to create an image version

Possible causes:

Image version name is invalid.

Allowed characters for image version are numbers and periods. Numbers must be within the range of a 32-bit integer. Format: *MajorVersion.MinorVersion.Patch*. Change the image version name and try again.

Source managed image from which the image version is being created is not found.

Check if the source image exists and is in the same region as the image version.

The managed image isn't done being provisioned.

Make sure the provisioning state of the source managed image is **Succeeded**.

The target region list does not include the source region.

The target region list must include the source region of the image version. Make sure you have included the source region in the list of target regions where you want Azure to replicate your image version to.

Replication to all the target regions not completed.

Use the **--expand ReplicationStatus** flag to check if the replication to all the specified target regions has been completed. If not, wait up to 6 hours for the job to complete. If it fails, run the command again to create and replicate the image version. If there are a lot of target regions the image version is being replicated to, consider doing the replication in phases.

Unable to create a VM or a scale set

Possible causes:

The user trying to create a VM or virtual machine scale set doesn't have the read access to the image version.

Contact the subscription owner and ask them to give read access to the image version or the parent resources (like the shared image gallery or image definition) through [Role Based Access Control](#) (RBAC).

The image version is not found.

Verify that the region you are trying to create a VM or virtual machine scale in is included in the list of target regions of the image version. If the region is already in the list of target regions, then verify if the replication job has been completed. You can use the **-ReplicationStatus** flag to check if the replication to all the specified target regions has been completed.

The VM or virtual machine scale set creation takes a long time.

Verify that the **OSType** of the image version that you are trying to create the VM or virtual machine scale set from has the same **OSType** of the source managed image that you used to create the image version.

Unable to share resources

The sharing of shared image gallery, image definition, and image version resources across subscriptions is enabled using [Role-Based Access Control](#) (RBAC).

Replication is slow

Use the **--expand ReplicationStatus** flag to check if the replication to all the specified target regions has been completed. If not, wait for up to 6 hours for the job to complete. If it fails, trigger the command again to create and replicate the image version. If there are a lot of target regions the image version is being replicated to, consider doing the replication in phases.

Azure limits and quotas

[Azure limits and quotas](#) apply to all shared image gallery, image definition, and image version resources. Make sure you are within the limits for your subscriptions.

Next steps

Learn more about [shared image galleries](#).

Preview: Creating and using proximity placement groups using PowerShell

1/19/2020 • 2 minutes to read • [Edit Online](#)

To get VMs as close as possible, achieving the lowest possible latency, you should deploy your scale set within a [proximity placement group](#).

A proximity placement group is a logical grouping used to make sure that Azure compute resources are physically located close to each other. Proximity placement groups are useful for workloads where low latency is a requirement.

IMPORTANT

Proximity Placement Groups is currently in public preview. This preview version is provided without a service level agreement, and it's not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

Proximity placement groups are not available in these regions during the preview: **Japan East, Australia East and India Central**.

Create a proximity placement group

Create a proximity placement group using the [New-AzProximityPlacementGroup](#) cmdlet.

```
$resourceGroup = "myPPGResourceGroup"
$location = "East US"
$ppgName = "myPPG"
New-AzResourceGroup -Name $resourceGroup -Location $location
$ppg = New-AzProximityPlacementGroup ` 
    -Location $location ` 
    -Name $ppgName ` 
    -ResourceGroupName $resourceGroup ` 
    -ProximityPlacementGroupType Standard
```

List proximity placement groups

You can list all of the proximity placement groups using the [Get-AzProximityPlacementGroup](#) cmdlet.

```
Get-AzProximityPlacementGroup
```

Create a scale set

Create a scale in the proximity placement group using `-ProximityPlacementGroup $ppg.Id` to refer to the proximity placement group ID when you use [New-AzVMSS](#) to create the scale set.

```
$scaleSetName = "myVM"

New-AzVmss `

-ResourceGroupName $resourceGroup `

-Location $location `

-VMSScaleSetName $scaleSetName `

-VirtualNetworkName "myVnet" `

-SubnetName "mySubnet" `

-PublicIpAddressName "myPublicIPAddress" `

-LoadBalancerName "myLoadBalancer" `

-UpgradePolicyMode "Automatic" `

-ProximityPlacementGroup $ppg.Id
```

You can see the instance in the placement group using [Get-AzProximityPlacementGroup](#).

```
Get-AzProximityPlacementGroup `

-ResourceId $ppg.Id | Format-Table `

-Wrap `

-Property VirtualMachineScaleSets
```

Next steps

You can also use the [Azure CLI](#) to create proximity placement groups.

Vertical autoscale with virtual machine scale sets

1/19/2020 • 4 minutes to read • [Edit Online](#)

This article describes how to vertically scale Azure [Virtual Machine Scale Sets](#) with or without reprovisioning.

Vertical scaling, also known as *scale up* and *scale down*, means increasing or decreasing virtual machine (VM) sizes in response to a workload. Compare this behavior with [horizontal scaling](#), also referred to as *scale out* and *scale in*, where the number of VMs is altered depending on the workload.

Reprovisioning means removing an existing VM and replacing it with a new one. When you increase or decrease the size of VMs in a virtual machine scale set, in some cases you want to resize existing VMs and retain your data, while in other cases you need to deploy new VMs of the new size. This document covers both cases.

Vertical scaling can be useful when:

- A service built on virtual machines is under-utilized (for example at weekends). Reducing the VM size can reduce monthly costs.
- Increasing VM size to cope with larger demand without creating additional VMs.

You can set up vertical scaling to be triggered based on metric based alerts from your virtual machine scale set.

When the alert is activated, it fires a webhook that triggers a runbook that can scale your scale set up or down.

Vertical scaling can be configured by following these steps:

1. Create an Azure Automation account with run-as capability.
2. Import Azure Automation Vertical Scale runbooks for virtual machine scale sets into your subscription.
3. Add a webhook to your runbook.
4. Add an alert to your virtual machine scale set using a webhook notification.

NOTE

Because of the size of the first Virtual Machine, the sizes it can be scaled to, may be limited due to the availability of the other sizes in the cluster current Virtual Machine is deployed in. In the published automation runbooks used in this article we take care of this case and only scale within the below VM size pairs. This means that a Standard_D1v2 Virtual Machine will not suddenly be scaled up to Standard_G5 or scaled down to Basic_A0. Also constrained Virtual Machine sizes scale up/down is not supported. You can choose to scale between the following pairs of sizes:

VM SIZES SCALING PAIR	
Basic_A0	Basic_A4
Standard_A0	Standard_A4
Standard_A5	Standard_A7
Standard_A8	Standard_A9
Standard_A10	Standard_A11
Standard_A1_v2	Standard_A8_v2
Standard_A2m_v2	Standard_A8m_v2
Standard_B1s	Standard_B2s

VM SIZES SCALING PAIR	
Standard_B1ms	Standard_B8ms
Standard_D1	Standard_D4
Standard_D11	Standard_D14
Standard_DS1	Standard_DS4
Standard_DS11	Standard_DS14
Standard_D1_v2	Standard_D5_v2
Standard_D11_v2	Standard_D14_v2
Standard_DS1_v2	Standard_DS5_v2
Standard_DS11_v2	Standard_DS14_v2
Standard_D2_v3	Standard_D64_v3
Standard_D2s_v3	Standard_D64s_v3
Standard_DC2s	Standard_DC4s
Standard_E2_v3	Standard_E64_v3
Standard_E2s_v3	Standard_E64s_v3
Standard_F1	Standard_F16
Standard_F1s	Standard_F16s
Standard_F2sv2	Standard_F72sv2
Standard_G1	Standard_G5
Standard_GS1	Standard_GS5
Standard_H8	Standard_H16
Standard_H8m	Standard_H16m
Standard_L4s	Standard_L32s
Standard_L8s_v2	Standard_L80s_v2
Standard_M8ms	Standard_M128ms
Standard_M32ls	Standard_M64ls
Standard_M64s	Standard_M128s
Standard_M64	Standard_M128
Standard_M64m	Standard_M128m
Standard_NC6	Standard_NC24

VM SIZES SCALING PAIR	
Standard_NC6s_v2	Standard_NC24s_v2
Standard_NC6s_v3	Standard_NC24s_v3
Standard_ND6s	Standard_ND24s
Standard_NV6	Standard_NV24
Standard_NV6s_v2	Standard_NV24s_v2
Standard_NV12s_v3	Standard_NV48s_v3

Create an Azure Automation Account with run-as capability

The first thing you need to do is create an Azure Automation account that hosts the runbooks used to scale the virtual machine scale set instances. Recently [Azure Automation](#) introduced the "Run As account" feature that makes setting up the Service Principal for automatically running the runbooks on a user's behalf. For more information, see:

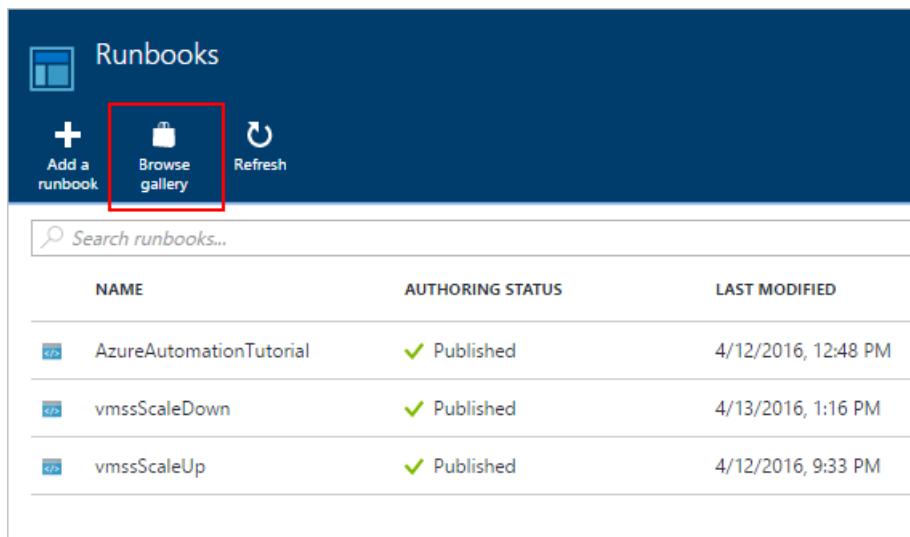
- [Authenticate Runbooks with Azure Run As account](#)

Import Azure Automation Vertical Scale runbooks into your subscription

The runbooks needed to vertically scale your virtual machine scale sets are already published in the Azure Automation Runbook Gallery. To import them into your subscription follow the steps in this article:

- [Runbook and module galleries for Azure Automation](#)

Choose the Browse Gallery option from the Runbooks menu:



The screenshot shows the Azure portal interface for Runbooks. At the top, there's a blue header bar with the 'Runbooks' title. Below it, there are three main buttons: 'Add a runbook' (with a plus sign icon), 'Browse gallery' (with a shopping bag icon), and 'Refresh' (with a circular arrow icon). The 'Browse gallery' button is highlighted with a red rectangular box. Below these buttons is a search bar with the placeholder text 'Search runbooks...'. The main area is a table listing runbooks, with columns for 'NAME', 'AUTHORING STATUS', and 'LAST MODIFIED'. Three runbooks are listed: 'AzureAutomationTutorial' (Published, 4/12/2016, 12:48 PM), 'vmssScaleDown' (Published, 4/13/2016, 1:16 PM), and 'vmssScaleUp' (Published, 4/12/2016, 9:33 PM).

NAME	AUTHORING STATUS	LAST MODIFIED
AzureAutomationTutorial	✓ Published	4/12/2016, 12:48 PM
vmssScaleDown	✓ Published	4/13/2016, 1:16 PM
vmssScaleUp	✓ Published	4/12/2016, 9:33 PM

The runbooks that need to be imported are shown. Select the runbook based on whether you want vertical scaling with or without reprovisioning:

Browse Gallery

Filter

scale set

Scale Down Virtual Machine Scale Set Instances
PowerShell Runbook
</>
This Azure Automation runbook will scale down Virtual Machine Scale Set instances. This is intended to run only in Azure Automation service. An Azure Automation account with the "Run as" feature is required.
Tags:

Created by: Kay Singh
Ratings: 0 of 5
0 downloads
Last update: 4/13/2016

Scale Up Virtual Machine Scale Set Instances
PowerShell Runbook
</>
This Azure Automation runbook will scale down Virtual Machine Scale Set instances. This is intended to run only in Azure Automation service. An Azure Automation account with the "Run as" feature is required.
Tags:

Created by: Kay Singh
Ratings: 0 of 5
0 downloads
Last update: 4/13/2016

Scale Down and Reprovision Virtual Machine Scale Set Instances
PowerShell Runbook
</>
This Azure Automation runbook will scale down and reprovision Virtual Machine Scale Set instances. This is intended to run only in Azure Automation service. An Azure Automation account with the "Run as" feature is required.
Tags:

Created by: Kay Singh
Ratings: 0 of 5
0 downloads
Last update: 4/13/2016

Scale Up and Reprovision Virtual Machine Scale Set Instances
PowerShell Runbook
</>
This Azure Automation runbook will scale down and reprovision Virtual Machine Scale Set instances. This is intended to run only in Azure Automation service. An Azure Automation account with the "Run as" feature is required.
Tags:

Created by: Kay Singh
Ratings: 0 of 5
0 downloads
Last update: 4/13/2016

Add a webhook to your runbook

Once you've imported the runbooks, add a webhook to the runbook so it can be triggered by an alert from a virtual machine scale set. The details of creating a webhook for your Runbook are described in this article:

- [Azure Automation webhooks](#)

NOTE

Make sure you copy the webhook URI before closing the webhook dialog as you will need this address in the next section.

Add an alert to your virtual machine scale set

Below is a PowerShell script that shows how to add an alert to a virtual machine scale set. Refer to the following article to get the name of the metric to fire the alert on: [Azure Monitor autoscaling common metrics](#).

```

$actionEmail = New-AzAlertRuleEmail -CustomEmail user@contoso.com
$actionWebhook = New-AzAlertRuleWebhook -ServiceUri <uri-of-the-webhook>
$threshold = <value-of-the-threshold>
$rg = <resource-group-name>
$id = <resource-id-to-add-the-alert-to>
$location = <location-of-the-resource>
$alertName = <name-of-the-resource>
$metricName = <metric-to-fire-the-alert-on>
$timeWindow = <time-window-in-hh:mm:ss-format>
$condition = <condition-for-the-threshold> # Other valid values are LessThanOrEqual, GreaterThan,
GreaterThanOrEqual
$description = <description-for-the-alert>

Add-AzMetricAlertRule -Name $alertName `

    -Location $location `

    -ResourceGroup $rg `

    -TargetResourceId $id `

    -MetricName $metricName `

    -Operator $condition `

    -Threshold $threshold `

    -WindowSize $timeWindow `

    -TimeAggregationOperator Average `

    -Actions $actionEmail, $actionWebhook `

    -Description $description

```

NOTE

It is recommended to configure a reasonable time window for the alert in order to avoid triggering vertical scaling, and any associated service interruption, too often. Consider a window of least 20-30 minutes or more. Consider horizontal scaling if you need to avoid any interruption.

For more information on how to create alerts, see the following articles:

- [Azure Monitor PowerShell quickstart samples](#)
- [Azure Monitor Cross-platform CLI quickstart samples](#)

Summary

This article showed simple vertical scaling examples. With these building blocks - Automation account, runbooks, webhooks, alerts - you can connect a rich variety of events with a customized set of actions.

Using Virtual Machine Scale Sets with the Azure DSC Extension

1/19/2020 • 2 minutes to read • [Edit Online](#)

[Virtual Machine Scale Sets](#) can be used with the [Azure Desired State Configuration \(DSC\)](#) extension handler. Virtual machine scale sets provide a way to deploy and manage large numbers of virtual machines, and can elastically scale in and out in response to load. DSC is used to configure the VMs as they come online so they are running the production software.

Differences between deploying to Virtual Machines and Virtual Machine Scale Sets

The underlying template structure for a virtual machine scale set is slightly different from a single VM. Specifically, a single VM deploys extensions under the "virtualMachines" node. There is an entry of type "extensions" where DSC is added to the template

```
"resources": [
    {
        "name": "Microsoft.PowerShell.DSC",
        "type": "extensions",
        "location": "[resourceGroup().location]",
        "apiVersion": "2015-06-15",
        "dependsOn": [
            "[concat('Microsoft.Compute/virtualMachines/', variables('vmName'))]"
        ],
        "tags": {
            "displayName": "dscExtension"
        },
        "properties": {
            "publisher": "Microsoft.PowerShell",
            "type": "DSC",
            "typeHandlerVersion": "2.20",
            "autoUpgradeMinorVersion": false,
            "forceUpdateTag": "[parameters('dscExtensionUpdateTagVersion')]",
            "settings": {
                "configuration": {
                    "url": "[concat(parameters('_artifactsLocation'), '/',
variables('dscExtensionArchiveFolder'), '/', variables('dscExtensionArchiveFileName'))]",
                    "script": "DscExtension.ps1",
                    "function": "Main"
                },
                "configurationArguments": {
                    "nodeName": "[variables('vmName')]"
                }
            },
            "protectedSettings": {
                "configurationUrlSasToken": "[parameters('_artifactsLocationSasToken')]"
            }
        }
    }
]
```

A virtual machine scale set node has a "properties" section with the "VirtualMachineProfile", "extensionProfile" attribute. DSC is added under "extensions"

```

"extensionProfile": {
    "extensions": [
        {
            "name": "Microsoft.Powershell.DSC",
            "properties": {
                "publisher": "Microsoft.Powershell",
                "type": "DSC",
                "typeHandlerVersion": "2.20",
                "autoUpgradeMinorVersion": false,
                "forceUpdateTag": "[parameters('DscExtensionUpdateTagVersion')]",
                "settings": {
                    "configuration": {
                        "url": "[concat(parameters('_artifactsLocation'), '/',
variables('DscExtensionArchiveFolder'), '/', variables('DscExtensionArchiveFileName'))]",
                        "script": "DscExtension.ps1",
                        "function": "Main"
                    },
                    "configurationArguments": {
                        "nodeName": "localhost"
                    }
                },
                "protectedSettings": {
                    "configurationUrlsAsToken": "[parameters('_artifactsLocationSasToken')]"
                }
            }
        }
    ]
}

```

Behavior for a Virtual Machine Scale Set

The behavior for a virtual machine scale set is identical to the behavior for a single VM. When a new VM is created, it is automatically provisioned with the DSC extension. If a newer version of the WMF is required by the extension, the VM reboots before coming online. Once it is online, it downloads the DSC configuration .zip and provision it on the VM. More details can be found in [the Azure DSC Extension Overview](#).

Next steps

Examine the [Azure Resource Manager template for the DSC extension](#).

Learn how the [DSC extension securely handles credentials](#).

For more information on the Azure DSC extension handler, see [Introduction to the Azure Desired State Configuration extension handler](#).

For more information about PowerShell DSC, [visit the PowerShell documentation center](#).

Convert a scale set template to a managed disk scale set template

1/19/2020 • 4 minutes to read • [Edit Online](#)

Customers with a Resource Manager template for creating a scale set not using managed disk may wish to modify it to use managed disk. This article shows how to use managed disks, using as an example a pull request from the [Azure Quickstart Templates](#), a community-driven repo for sample Resource Manager templates. The full pull request can be seen here: <https://github.com/Azure/azure-quickstart-templates/pull/2998>, and the relevant parts of the diff are below, along with explanations:

Making the OS disks managed

In the following diff, several variables related to storage account and disk properties are removed. Storage account type is no longer necessary (Standard_LRS is the default), but you could specify it if desired. Only Standard_LRS and Premium_LRS are supported with managed disk. New storage account suffix, unique string array, and sa count were used in the old template to generate storage account names. These variables are no longer necessary in the new template because managed disk automatically creates storage accounts on the customer's behalf. Similarly, vhd container name and OS disk name are no longer necessary because managed disk automatically names the underlying storage blob containers and disks.

```
"variables": {
-   "storageAccountType": "Standard_LRS",
-   "namingInfix": "[toLower(substring(concat(parameters('vmssName'), uniqueString(resourceGroup().id)), 0, 9))]",
-   "longNamingInfix": "[toLower(parameters('vmssName'))]",
-   "newStorageAccountSuffix": "[concat(variables('namingInfix'), 'sa')]",
-   "uniqueStringArray": [
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '0')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '1')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '2')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '3')))]",
-     "[concat(uniqueString(concat(resourceGroup().id, variables('newStorageAccountSuffix'), '4')))]"
-   ],
-   "saCount": "[length(variables('uniqueStringArray'))]",
-   "vhdContainerName": "[concat(variables('namingInfix'), 'vhd')]",
-   "osDiskName": "[concat(variables('namingInfix'), 'osdisk')]",
-   "addressPrefix": "10.0.0.0/16",
-   "subnetPrefix": "10.0.0.0/24",
-   "virtualNetworkName": "[concat(variables('namingInfix'), 'vnet')]"}
```

In the following diff, you compute API is updated to version 2016-04-30-preview, which is the earliest required version for managed disk support with scale sets. You could use unmanaged disks in the new API version with the old syntax if desired. If you only update the compute API version and don't change anything else, the template should continue to work as before.

```

@@ -86,7 +74,7 @@
        "version": "latest"
    },
    "imageReference": "[variables('osType')]",
-   "computeApiVersion": "2016-03-30",
+   "computeApiVersion": "2016-04-30-preview",
    "networkApiVersion": "2016-03-30",
    "storageApiVersion": "2015-06-15"
},

```

In the following diff, the storage account resource is removed from the resources array completely. The resource is no longer needed as managed disk creates them automatically.

```

@@ -113,19 +101,6 @@
        }
    },
{
-   "type": "Microsoft.Storage/storageAccounts",
-   "name": "[concat(variables('uniqueStringArray')[copyIndex()], variables('newStorageAccountSuffix'))]",
-   "location": "[resourceGroup().location]",
-   "apiVersion": "[variables('storageApiVersion')]",
-   "copy": {
-       "name": "storageLoop",
-       "count": "[variables('saCount')]"
-   },
-   "properties": {
-       "accountType": "[variables('storageAccountType')]"
-   }
},
{
    "type": "Microsoft.Network/publicIPAddresses",
    "name": "[variables('publicIPAddressName')]",
    "location": "[resourceGroup().location]",

```

In the following diff, we can see that we are removing the depends on clause referring from the scale set to the loop that was creating storage accounts. In the old template, this was ensuring that the storage accounts were created before the scale set began creation, but this clause is no longer necessary with managed disk. The vhd containers property is also removed, along with the OS disk name property as these properties are automatically handled under the hood by managed disk. You could add `"managedDisk": { "storageAccountType": "Premium_LRS" }` in the "osDisk" configuration if you wanted premium OS disks. Only VMs with an uppercase or lowercase 's' in the VM sku can use premium disks.

```

@@ -183,7 +158,6 @@
    "location": "[resourceGroup().location]",
    "apiVersion": "[variables('computeApiVersion')]",
    "dependsOn": [
-     "storageLoop",
      "[concat('Microsoft.Network/loadBalancers/', variables('loadBalancerName'))]",
      "[concat('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]"
    ],
@@ -200,16 +174,8 @@
    "virtualMachineProfile": {
      "storageProfile": {
        "osDisk": {
-         "vhdContainers": [
-           "[concat('https://', variables('uniqueStringArray')[0], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[1], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[2], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[3], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]",
-           "[concat('https://', variables('uniqueStringArray')[4], variables('newStorageAccountSuffix'),
'.blob.core.windows.net/', variables('vhdContainerName'))]"
-         ],
-         "name": "[variables('osDiskName')]",
        },
        "imageReference": "[variables('imageReference')]"
      },

```

There is no explicit property in the scale set configuration for whether to use managed or unmanaged disk. The scale set knows which to use based on the properties that are present in the storage profile. Thus, it is important when modifying the template to ensure that the right properties are in the storage profile of the scale set.

Data disks

With the changes above, the scale set uses managed disks for the OS disk, but what about data disks? To add data disks, add the "dataDisks" property under "storageProfile" at the same level as "osDisk". The value of the property is a JSON list of objects, each of which has properties "lun" (which must be unique per data disk on a VM), "createOption" ("empty" is currently the only supported option), and "diskSizeGB" (the size of the disk in gigabytes; must be greater than 0 and less than 1024) as in the following example:

```

"dataDisks": [
  {
    "lun": "1",
    "createOption": "empty",
    "diskSizeGB": "1023"
  }
]

```

If you specify `n` disks in this array, each VM in the scale set gets `n` data disks. Do note, however, that these data disks are raw devices. They are not formatted. It is up to the customer to attach, partition, and format the disks before using them. Optionally, you could also specify `"managedDisk": { "storageAccountType": "Premium_LRS" }` in each data disk object to specify that it should be a premium data disk. Only VMs with an uppercase or lowercase 's' in the VM sku can use premium disks.

To learn more about using data disks with scale sets, see [this article](#).

Next steps

For example Resource Manager templates using scale sets, search for "vmss" in the [Azure Quickstart Templates GitHub repo](#).

For general information, check out the [main landing page for scale sets](#).

Planned maintenance notifications for virtual machine scale sets

1/19/2020 • 11 minutes to read • [Edit Online](#)

Azure periodically performs updates to improve the reliability, performance, and security of the host infrastructure for virtual machines (VMs). Updates might include patching the hosting environment or upgrading and decommissioning hardware. Most updates don't affect the hosted VMs. However, updates affect VMs in these scenarios:

- If the maintenance doesn't require a reboot, Azure uses in-place migration to pause the VM while the host is updated. Maintenance operations that don't require a reboot are applied fault domain by fault domain. Progress is stopped if any warning health signals are received.
- If maintenance requires a reboot, you get a notice of when the maintenance is planned. In these cases, you are given a time window that is typically 35 days where you can start the maintenance yourself, when it works for you.

Planned maintenance that requires a reboot is scheduled in waves. Each wave has different scope (regions):

- A wave starts with a notification to customers. By default, notification is sent to the subscription owner and co-owners. You can add recipients and messaging options like email, SMS, and webhooks to the notifications by using Azure [Activity Log alerts](#).
- With notification, a *self-service window* is made available. During this window that is typically 35 days, you can find which of your VMs are included in the wave. You can proactively start maintenance according to your own scheduling needs.
- After the self-service window, a *scheduled maintenance window* begins. At some point during this window, Azure schedules and applies the required maintenance to your VM.

The goal in having two windows is to give you enough time to start maintenance and reboot your VM while knowing when Azure will automatically start maintenance.

You can use the Azure portal, PowerShell, the REST API, and the Azure CLI to query for maintenance windows for your virtual machine scale set VMs, and to start self-service maintenance.

Should you start maintenance during the self-service window?

The following guidelines can help you decide whether to start maintenance at a time that you choose.

NOTE

Self-service maintenance might not be available for all of your VMs. To determine whether proactive redeploy is available for your VM, look for **Start now** in the maintenance status. Currently, self-service maintenance isn't available for Azure Cloud Services (Web/Worker Role) and Azure Service Fabric.

Self-service maintenance isn't recommended for deployments that use *availability sets*. Availability sets are highly available setups in which only one update domain is affected at any time. For availability sets:

- Let Azure trigger the maintenance. For maintenance that requires a reboot, maintenance is done update domain by update domain. Update domains don't necessarily receive the maintenance sequentially. There's a 30-minute pause between update domains.

- If a temporary loss of some of your capacity (1/update domain count) is a concern, you can easily compensate for the loss by allocating additional instances during the maintenance period.
- For maintenance that doesn't require a reboot, updates are applied at the fault domain level.

Don't use self-service maintenance in the following scenarios:

- If you shut down your VMs frequently, either manually, by using DevTest Labs, by using auto-shutdown, or by following a schedule. Self-service maintenance in these scenarios might revert the maintenance status and cause additional downtime.
- On short-lived VMs that you know will be deleted before the end of the maintenance wave.
- For workloads with a large state stored in the local (ephemeral) disk that you want to maintain after update.
- If you resize your VM often. This scenario might revert the maintenance status.
- If you have adopted scheduled events that enable proactive failover or graceful shutdown of your workload 15 minutes before maintenance shutdown begins.

Do use self-service maintenance if you plan to run your VM uninterrupted during the scheduled maintenance phase and none of the preceding counterindications apply.

It's best to use self-service maintenance in the following cases:

- You need to communicate an exact maintenance window to management or your customer.
- You need to complete the maintenance by a specific date.
- You need to control the sequence of maintenance, for example, in a multi-tier application, to guarantee safe recovery.
- You need more than 30 minutes of VM recovery time between two update domains. To control the time between update domains, you must trigger maintenance on your VMs one update domain at a time.

View virtual machine scale sets that are affected by maintenance in the portal

When a planned maintenance wave is scheduled, you can view the list of virtual machine scale sets that are affected by the upcoming maintenance wave by using the Azure portal.

1. Sign in to the [Azure portal](#).
2. In the left menu, select **All services**, and then select **Virtual machine scale sets**.
3. Under **Virtual machine scale sets**, select **Edit columns** to open the list of available columns.
4. In the **Available columns** section, select **Self-service maintenance**, and then move it to the **Selected columns** list. Select **Apply**.

To make the **Self-service maintenance** item easier to find, you can change the drop-down option in the **Available columns** section from **All** to **Properties**.

The **Self-service maintenance** column now appears in the list of virtual machine scale sets. Each virtual machine scale set can have one of the following values for the self-service maintenance column:

VALUE	DESCRIPTION
Yes	At least one VM in your virtual machine scale set is in a self-service window. You can start maintenance at any time during this self-service window.
No	No VMs are in a self-service window in the affected virtual machine scale set.

VALUE	DESCRIPTION
-	Your virtual machines scale sets aren't part of a planned maintenance wave.

Notification and alerts in the portal

Azure communicates a schedule for planned maintenance by sending an email to the subscription owner and co-owners group. You can add recipients and channels to this communication by creating Activity Log alerts. For more information, see [Monitor subscription activity with the Azure Activity Log](#).

1. Sign in to the [Azure portal](#).
2. In the left menu, select **Monitor**.
3. In the **Monitor - Alerts (classic)** pane, select **+Add activity log alert**.
4. On the **Add activity log alert** page, select or enter the requested information. In **Criteria**, make sure that you set the following values:
 - **Event category:** Select **Service Health**.
 - **Services:** Select **Virtual Machine Scale Sets and Virtual Machines**.
 - **Type:** Select **Planned maintenance**.

To learn more about how to configure Activity Log alerts, see [Create Activity Log alerts](#)

Start maintenance on your virtual machine scale set from the portal

You can see more maintenance-related details in the overview of virtual machine scale sets. If at least one VM in the virtual machine scale set is included in the planned maintenance wave, a new notification ribbon is added near the top of the page. Select the notification ribbon to go to the **Maintenance** page.

On the **Maintenance** page, you can see which VM instance is affected by the planned maintenance. To start maintenance, select the check box that corresponds to the affected VM. Then, select **Start maintenance**.

After you start maintenance, the affected VMs in your virtual machine scale set undergo maintenance and are temporarily unavailable. If you missed the self-service window, you can still see the time window when your virtual machine scale set will be maintained by Azure.

Check maintenance status by using PowerShell

You can use Azure PowerShell to see when VMs in your virtual machine scale sets are scheduled for maintenance. Planned maintenance information is available by using the `Get-AzVmss` cmdlet when you use the `-InstanceView` parameter.

Maintenance information is returned only if maintenance is planned. If no maintenance is scheduled that affects the VM instance, the cmdlet doesn't return any maintenance information.

```
Get-AzVmss -ResourceGroupName rgName -VMScaleSetName vmssName -InstanceId id -InstanceView
```

The following properties are returned under **MaintenanceRedeployStatus**:

VALUE	DESCRIPTION
IsCustomerInitiatedMaintenanceAllowed	Indicates whether you can start maintenance on the VM at this time.

VALUE	DESCRIPTION
PreMaintenanceWindowStartTime	The beginning of the maintenance self-service window when you can initiate maintenance on your VM.
PreMaintenanceWindowEndTime	The end of the maintenance self-service window when you can initiate maintenance on your VM.
MaintenanceWindowStartTime	The beginning of the maintenance scheduled in which Azure initiates maintenance on your VM.
MaintenanceWindowEndTime	The end of the maintenance scheduled window in which Azure initiates maintenance on your VM.
LastOperationResultCode	The result of the last attempt to initiate maintenance on the VM.

Start maintenance on your VM instance by using PowerShell

You can start maintenance on a VM if **IsCustomerInitiatedMaintenanceAllowed** is set to **true**. Use the [Set-AzVmss](#) cmdlet with `-PerformMaintenance` parameter.

```
Set-AzVmss -ResourceGroupName rgName -VMScaleSetName vmssName -InstanceId id -PerformMaintenance
```

Check maintenance status by using the CLI

You can view planned maintenance information by using [az vmss list-instances](#).

Maintenance information is returned only if maintenance is planned. If no maintenance that affects the VM instance is scheduled, the command doesn't return any maintenance information.

```
az vmss list-instances -g rgName -n vmssName --expand instanceView
```

The following properties are returned under **MaintenanceRedeployStatus** for each VM instance:

VALUE	DESCRIPTION
IsCustomerInitiatedMaintenanceAllowed	Indicates whether you can start maintenance on the VM at this time.
PreMaintenanceWindowStartTime	The beginning of the maintenance self-service window when you can initiate maintenance on your VM.
PreMaintenanceWindowEndTime	The end of the maintenance self-service window when you can initiate maintenance on your VM.
MaintenanceWindowStartTime	The beginning of the maintenance scheduled in which Azure initiates maintenance on your VM.
MaintenanceWindowEndTime	The end of the maintenance scheduled window in which Azure initiates maintenance on your VM.
LastOperationResultCode	The result of the last attempt to initiate maintenance on the VM.

Start maintenance on your VM instance by using the CLI

The following call initiates maintenance on a VM instance if `IsCustomerInitiatedMaintenanceAllowed` is set to **true**:

```
az vmss perform-maintenance -g rgName -n vmssName --instance-ids id
```

FAQ

Q: Why do you need to reboot my VMs now?

A: Although most updates and upgrades to the Azure platform don't affect VM availability, in some cases, we can't avoid rebooting VMs hosted in Azure. We have accumulated several changes that require us to restart our servers that will result in VM reboot.

Q: If I follow your recommendations for high availability by using an availability set, am I safe?

A: Virtual machines deployed in an availability set or in virtual machine scale sets use update domains. When performing maintenance, Azure honors the update domain constraint and doesn't reboot VMs from a different update domain (within the same availability set). Azure also waits for at least 30 minutes before moving to the next group of VMs.

For more information about high availability, see [Regions and availability for virtual machines in Azure](#).

Q: How can I be notified about planned maintenance?

A: A planned maintenance wave starts by setting a schedule to one or more Azure regions. Soon after, an email notification is sent to the subscription owners (one email per subscription). You can add channels and recipients for this notification by using Activity Log alerts. If you deploy a VM to a region in which planned maintenance is already scheduled, you don't receive the notification. Instead, check the maintenance state of the VM.

Q: I don't see any indication of planned maintenance in the portal, PowerShell, or the CLI. What's wrong?

A: Information related to planned maintenance is available during a planned maintenance wave only for the VMs that are affected by the planned maintenance. If you don't see data, the maintenance wave might already be finished (or not started), or your VM might already be hosted on an updated server.

Q: Is there a way to know exactly when my VM will be affected?

A: When we set the schedule, we define a time window of several days. The exact sequencing of servers (and VMs) within this window is unknown. If you want to know the exact time your VMs will be updated, you can use [scheduled events](#). When you use scheduled events, you can query from within the VM and receive a 15-minute notification before a VM reboot.

Q: How long will it take you to reboot my VM?

A: Depending on the size of your VM, reboot might take up to several minutes during the self-service maintenance window. During the Azure-initiated reboots in the scheduled maintenance window, the reboot typically takes about 25 minutes. If you use Cloud Services (Web/Worker Role), virtual machine scale sets, or availability sets, you are given 30 minutes between each group of VMs (update domain) during the scheduled maintenance window.

Q: I don't see any maintenance information on my VMs. What went wrong?

A: There are several reasons why you might not see any maintenance information on your VMs:

- You are using a subscription marked as *Microsoft Internal*.
- Your VMs aren't scheduled for maintenance. It might be that the maintenance wave ended, was canceled, or was modified so that your VMs are no longer affected by it.

- You don't have the **Maintenance** column added to your VM list view. Although we added this column to the default view, if you configure your view to see non-default columns, you must manually add the **Maintenance** column to your VM list view.

Q: My VM is scheduled for maintenance for the second time. Why?

A: In several use cases, your VM is scheduled for maintenance after you have already completed your maintenance and redeployed:

- We have canceled the maintenance wave and restarted it with a different payload. It might be that we've detected a faulted payload and we simply need to deploy an additional payload.
- Your VM was *service healed* to another node due to a hardware fault.
- You have selected to stop (deallocate) and restart the VM.
- You have **auto shutdown** turned on for the VM.

Next steps

Learn how to register for maintenance events from within the VM by using [scheduled events](#).

Azure classic CLI commands

2/28/2020 • 42 minutes to read • [Edit Online](#)

IMPORTANT

Classic VMs will be retired on March 1, 2023.

If you use IaaS resources from ASM, please complete your migration by March 1, 2023. We encourage you to make the switch sooner to take advantage of the many feature enhancements in Azure Resource Manager.

For more information, see [Migrate your IaaS resources to Azure Resource Manager by March 1, 2023](#).

This topic describes how to install the Azure classic CLI. The classic CLI is deprecated and should only be used with the classic deployment model. For all other deployments, use the [Azure CLI](#).

This article provides syntax and options for Azure classic command-line interface (CLI) commands you'd commonly use to create and manage Azure resources. This is not a complete reference, and your CLI version may show slightly different commands or parameters.

To get started, first [install the Azure classic CLI](#) and [connect to your Azure subscription](#).

For current command syntax and options at the command line in Resource Manager mode, type `azure help` or, to display help for a specific command, `azure help [command]`. Also find CLI examples in the documentation for creating and managing specific Azure services.

Optional parameters are shown in square brackets (for example, `[parameter]`). All other parameters are required.

In addition to command-specific optional parameters documented here, there are three optional parameters that can be used to display detailed output such as request options and status codes. The `-v` parameter provides verbose output, and the `-vv` parameter provides even more detailed verbose output. The `--json` option outputs the result in raw json format.

Setting the Resource Manager mode

Use the following command to enable Azure CLI Resource Manager mode commands.

```
azure config mode arm
```

NOTE

The CLI's Azure Resource Manager mode and Azure Service Management mode are mutually exclusive. That is, resources created in one mode cannot be managed from the other mode.

Account information

Your Azure subscription information is used by the tool to connect to your account.

List the imported subscriptions

```
account list [options]
```

Show details about a subscription

```
account show [options] [subscriptionNameOrId]
```

Set the current subscription

```
account set [options] <subscriptionNameOrId>
```

Remove a subscription or environment, or clear all of the stored account and environment info

```
account clear [options]
```

Commands to manage your account environment

```
account env list [options]
account env show [options] [environment]
account env add [options] [environment]
account env set [options] [environment]
account env delete [options] [environment]
```

Active Directory objects

Commands to display active directory applications

```
ad app create [options]
ad app delete [options] <object-id>
```

Commands to display active directory groups

```
ad group list [options]
ad group show [options]
```

Commands to provide an active directory sub group or member info

```
ad group member list [options] [objectId]
```

Commands to display active directory service principals

```
ad sp list [options]
ad sp show [options]
ad sp create [options] <application-id>
ad sp delete [options] <object-id>
```

Commands to display active directory users

```
ad user list [options]
ad user show [options]
```

Availability sets

Creates an availability set within a resource group

```
availset create [options] <resource-group> <name> <location> [tags]
```

Lists the availability sets within a resource group

```
availset list [options] <resource-group>
```

Gets one availability set within a resource group

```
availset show [options] <resource-group> <name>
```

Deletes one availability set within a resource group

```
availset delete [options] <resource-group> <name>
```

Local settings

List Azure CLI configuration settings

```
config list [options]
```

Delete a config setting

```
config delete [options] <name>
```

Update a config setting

```
config set <name> <value>
```

Sets the Azure CLI working mode to either arm or asm

```
config mode [options] <modename>
```

Account features

List all features available for your subscription

```
feature list [options]
```

Shows a feature

```
feature show [options] <providerName> <featureName>
```

Registers a previewed feature of a resource provider

```
feature register [options] <providerName> <featureName>
```

Resource groups

Creates a resource group

```
group create [options] <name> <location>
```

Set tags to a resource group

```
group set [options] <name> <tags>
```

Deletes a resource group

```
group delete [options] <name>
```

Lists the resource groups for your subscription

```
group list [options]
```

Shows a resource group for your subscription

```
group show [options] <name>
```

Commands to manage resource group logs

```
group log show [options] [name]
```

Commands to manage your deployment in a resource group

```
group deployment create [options] [resource-group] [name]
group deployment list [options] <resource-group> [state]
group deployment show [options] <resource-group> [deployment-name]
group deployment stop [options] <resource-group> [deployment-name]
```

Commands to manage your local or gallery resource group template

```
group template list [options]
group template show [options] <name>
group template download [options] [name] [file]
group template validate [options] <resource-group>
```

HDInsight clusters

Commands to create or add to a cluster configuration file

```
hdinsight config create [options] <configFilePath> <overwrite>
hdinsight config add-config-values [options] <configFilePath>
hdinsight config add-script-action [options] <configFilePath>
```

Example: Create a configuration file that contains a script action to run when creating a cluster.

```
hdinsight config create "C:\myFiles\configFile.config"
hdinsight config add-script-action --configFilePath "C:\myFiles\configFile.config" --nodeType HeadNode --uri
<scriptActionURI> --name myScriptAction --parameters "-param value"
```

Command to create a cluster in a resource group

```
hdinsight cluster create [options] <clusterName>
```

Example: Create a Storm on Linux cluster

```
azure hdinsight cluster create -g myarmgroup -l westus -y Linux --clusterType Storm --version 3.2 --
defaultStorageAccountName mystorageaccount --defaultStorageAccountKey <defaultStorageAccountKey> --
defaultStorageContainer mycontainer --userName admin --password <clusterPassword> --sshUserName sshuser --
sshPassword <sshPassword> --workerNodeCount 1 myNewCluster01

info: Executing command hdinsight cluster create
+ Submitting the request to create cluster...
info: hdinsight cluster create command OK
```

Example: Create a cluster with a script action

```
azure hdinsight cluster create -g myarmgroup -l westus -y Linux --clusterType Hadoop --version 3.2 --
defaultStorageAccountName mystorageaccount --defaultStorageAccountKey <defaultStorageAccountKey> --
defaultStorageContainer mycontainer --userName admin --password <clusterPassword> --sshUserName sshuser --
sshPassword <sshPassword> --workerNodeCount 1 -configurationPath "C:\myFiles\configFile.config" myNewCluster01

info: Executing command hdinsight cluster create
+ Submitting the request to create cluster...
info: hdinsight cluster create command OK
```

Parameter options:

```

-h, --help
-v, --verbose
-vv
--json
-g --resource-group <resource-group>
-c, --clusterName <clusterName>
-l, --location <location>
-y, --osType <osType>
'Windows' or 'Linux'
--version <version>
--clusterType <clusterType>
Hadoop | HBase | Spark | Storm
--defaultStorageAccountName <storageAccountName>
storage
--defaultStorageAccountKey <storageAccountKey>
HDInsight storage
--defaultStorageContainer <storageContainer>
HDInsight default storage
--headNodeSize <headNodeSize>
--workerNodeCount <workerNodeCount>
--workerNodeSize <workerNodeSize>
--zookeeperNodeSize <zookeeperNodeSize>
--userName <userName>
--password <password>
--sshUserName <sshUserName>
--sshPassword <sshPassword>
--sshPublicKey <sshPublicKey>
--rdpUserName <rdpUserName>
--rdpPassword <rdpPassword>
--rdpAccessExpiry <rdpAccessExpiry>
For example 12/12/2015 (only for Windows clusters)
--virtualNetworkId <virtualNetworkId>
Value is a GUID for Windows cluster and ARM resource ID for Linux cluster
--subnetName <subnetName>
--additionalStorageAccounts <additionalStorageAccounts>
Can be multiple.
In the format of 'accountName#accountKey'.
For example, --additionalStorageAccounts "acc1#key1;acc2#key2"
--hiveMetastoreServerName <hiveMetastoreServerName>
metastore for Hive
--hiveMetastoreDatabaseName <hiveMetastoreDatabaseName>
for Hive
--hiveMetastoreUserName <hiveMetastoreUserName>
metastore for Hive
--hiveMetastorePassword <hiveMetastorePassword>
metastore for Hive
--oozieMetastoreServerName <oozieMetastoreServerName>
metastore for Oozie
--oozieMetastoreDatabaseName <oozieMetastoreDatabaseName>
for Oozie
--oozieMetastoreUserName <oozieMetastoreUserName>
metastore for Oozie
--oozieMetastorePassword <oozieMetastorePassword>
metastore for Oozie
--configurationPath <configurationPath>
-s, --subscription <id>
--tags <tags>
Can be multiple.
In the format of 'name=value'.
Name is required and value is optional.
For example, --tags tag1=value1;tag2

```

output usage information
use verbose output
more verbose with debug output
use json output
The name of the resource group
HDInsight cluster name
Data center location for the cluster
HDInsight cluster operating system
HDInsight cluster version
HDInsight cluster type.
Storage account url to use for default HDInsight
Key to the storage account to use for default
Container in the storage account to use for
(Optional) Head node size for the cluster
Number of worker nodes to use for the cluster
(Optional) Worker node size for the cluster
(Optional) Zookeeper node size for the cluster
Cluster username
Cluster password
SSH username (only for Linux clusters)
SSH password (only for Linux clusters)
SSH public key (only for Linux clusters)
RDP username (only for Windows clusters)
RDP password (only for Windows clusters)
RDP access expiry.
(Optional) Virtual network ID for the cluster.
(Optional) Subnet for the cluster
(Optional) Additional storage accounts.
(Optional) SQL Server name for the external metastore
(Optional) Database name for the external metastore
(Optional) Database username for the external metastore
(Optional) Database password for the external metastore
(Optional) SQL Server name for the external metastore
(Optional) Database name for the external metastore
(Optional) Database username for the external metastore
(Optional) Database password for the external metastore
(Optional) HDInsight cluster configuration file path
The subscription id
Tags to set to the cluster.

Command to delete a cluster

```
hdinsight cluster delete [options] <clusterName>
```

Command to show cluster details

```
hdinsight cluster show [options] <clusterName>
```

Command to list all clusters (in a specific resource group, if provided)

```
hdinsight cluster list [options]
```

Command to resize a cluster

```
hdinsight cluster resize [options] <clusterName> <targetInstanceCount>
```

Command to enable HTTP access for a cluster

```
hdinsight cluster enable-http-access [options] <clusterName> <userName> <password>
```

Command to disable HTTP access for a cluster

```
hdinsight cluster disable-http-access [options] <clusterName>
```

Command to enable RDP access for a cluster

```
hdinsight cluster enable-rdp-access [options] <clusterName> <rdpUserName> <rdpPassword> <rdpExpiryDate>
```

Command to disable HTTP access for a cluster

```
hdinsight cluster disable-rdp-access [options] <clusterName>
```

Insights (events, alert rules, autoscale settings, metrics)

Retrieve operation logs for a subscription, a correlationId, a resource group, resource, or resource provider

```
insights logs list [options]
```

Locations

List the available locations

```
location list [options]
```

Network resources

Commands to manage virtual networks

```
network vnet create [options] <resource-group> <name> <location>
```

Creates a virtual network. In the following example we create a virtual network named newvnet for resource group myresourcegroup in the West US region.

```
azure network vnet create myresourcegroup newvnet "west us"
info:   Executing command network vnet create
+ Looking up virtual network "newvnet"
+ Creating virtual network "newvnet"
  Loading virtual network state
data:   Id:
/subscriptions/#####
data:   Name:          newvnet
data:   Type:          Microsoft.Network/virtualNetworks
data:   Location:      westus
data:   Tags:
data:   Provisioning state: Succeeded
data:   Address prefixes:
data:     10.0.0.0/8
data:   DNS servers:
data:   Subnets:
data:
info:   network vnet create command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-n, --name <name>	the name of the virtual network
-l, --location <location>	the location
-a, --address-prefixes <address-prefixes>	the comma separated list of address prefixes for this virtual network For example -a 10.0.0.0/24,10.0.1.0/24. Default value is 10.0.0.0/8
-d, --dns-servers <dns-servers>	the comma separated list of DNS servers IP addresses
-t, --tags <tags>	the tags set on this virtual network. Can be multiple. In the format of "name=value". Name is required and value is optional. For example, -t tag1=value1;tag2
-s, --subscription <subscription>	the subscription identifier

```
network vnet set [options] <resource-group> <name>
```

Updates a virtual network configuration within a resource group.

```
azure network vnet set myresourcegroup newvnet

info: Executing command network vnet set
+ Looking up virtual network "newvnet"
+ Updating virtual network "newvnet"
+ Loading virtual network state
data: Id: /subscriptions/#####/resourceGroups/myresourcegroup/providers/Microsoft.Network/virtualNetworks/newvnet
data: Name: newvnet
data: Type: Microsoft.Network/virtualNetworks
data: Location: westus
data: Tags:
data: Provisioning state: Succeeded
data: Address prefixes:
data: 10.0.0.0/8
data: DNS servers:
data: Subnets:
data:
info: network vnet set command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the virtual network
-a, --address-prefixes <address-prefixes>  the comma separated list of address prefixes for this virtual
network.
For example -a 10.0.0.0/24,10.0.1.0/24.
This list will be appended to the current list of address prefixes.
The address prefixes in this list should not overlap between them.
The address prefixes in this list should not overlap with existing address prefixes in the vnet.

-d, --dns-servers [dns-servers]           the comma separated list of DNS servers IP addresses.
This list will be appended to the current list of DNS server IP addresses.

-t, --tags <tags>                         the tags set on this virtual network.
Can be multiple. In the format of "name=value".
Name is required and value is optional. For example, -t tag1=value1;tag2.
This list will be appended to the current list of tags

--no-tags                                 remove all existing tags
-s, --subscription <subscription>        the subscription identifier
```

```
network vnet list [options] <resource-group>
```

The command lists all virtual networks in a resource group.

```
C:\>azure network vnet list myresourcegroup

info: Executing command network vnet list
+ Listing virtual networks
data: ID
Name Location Address prefixes DNS servers
data: -----
-----
data: /subscriptions/#####
wvnet newvnet westus 10.0.0.0/8
info: network vnet list command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-s, --subscription <subscription>	the subscription identifier

```
network vnet show [options] <resource-group> <name>
```

The command shows the virtual network properties in a resource group.

```
azure network vnet show -g myresourcegroup -n newvnet

info: Executing command network vnet show
+ Looking up virtual network "newvnet"
data: Id:
/subscriptions/#####
data: Name: newvnet
data: Type: Microsoft.Network/virtualNetworks
data: Location: westus
data: Tags:
data: Provisioning state: Succeeded
data: Address prefixes:
data: 10.0.0.0/8
data: DNS servers:
data: Subnets:
data:
info: network vnet show command OK
```

```
network vnet delete [options] <resource-group> <name>
```

The command removes a virtual network.

```
azure network vnet delete myresourcegroup newvnetX

info: Executing command network vnet delete
+ Looking up virtual network "newvnetX"
Delete virtual network newvnetX? [y/n] y
+ Deleting virtual network "newvnetX"
info: network vnet delete command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-n, --name <name>	the name of the virtual network
-q, --quiet	quiet mode, do not ask for delete confirmation
-s, --subscription <subscription>	the subscription identifier

Commands to manage virtual network subnets

```
network vnet subnet create [options] <resource-group> <vnet-name> <name>
```

Adds another subnet to an existing virtual network.

```
azure network vnet subnet create -g myresourcegroup --vnet-name newvnet -n subnet --address-prefix 10.0.1.0/24

info: Executing command network vnet subnet create
+ Looking up the subnet "subnet"
+ Creating subnet "subnet"
+ Looking up the subnet "subnet"
data: Id: /subscriptions/################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/virtualNetworks/newvnet/subnets/subnet
data: Name: subnet
data: Type: Microsoft.Network/virtualNetworks/subnets
data: Provisioning state: Succeeded
data: Address prefix: 10.0.1.0/24
info: network vnet subnet create command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-e, --vnet-name <vnet-name>	the name of the virtual network
-n, --name <name>	the name of the subnet
-a, --address-prefix <address-prefix>	the address prefix
-w, --network-security-group-id <network-security-group-id>	the network security group identifier. e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Network/networkSecurityGroups/<nsg-name>
-o, --network-security-group-name <network-security-group-name>	the network security group name
-s, --subscription <subscription>	the subscription identifier

```
network vnet subnet set [options] <resource-group> <vnet-name> <name>
```

Sets a specific virtual network subnet within a resource group.

```
C:\>azure network vnet subnet set -g myresourcegroup --vnet-name newvnet -n subnet1

info: Executing command network vnet subnet set
+ Looking up the subnet "subnet1"
+ Setting subnet "subnet1"
+ Looking up the subnet "subnet1"
data: Id:
/subscriptions/#####resourceGroups/myresourcegroup/providers/Microsoft.Network/virtualNetworks/newvnet/subnets/subnet1
data: Name: subnet1
data: Type: Microsoft.Network/virtualNetworks/subnets
data: Provisioning state: Succeeded
data: Address prefix: 10.0.1.0/24
info: network vnet subnet set command OK
```

```
network vnet subnet list [options] <resource-group> <vnet-name>
```

Lists all the virtual network subnets for a specific virtual network within a resource group.

```
azure network vnet subnet set -g myresourcegroup --vnet-name newvnet -n subnet1

info: Executing command network vnet subnet set
+ Looking up the subnet "subnet1"
+ Setting subnet "subnet1"
+ Looking up the subnet "subnet1"
data: Id:
/subscriptions/#####resourceGroups/myresourcegroup/providers/Microsoft.Network/virtualNetworks/newvnet/subnets/subnet1
data: Name: subnet1
data: Type: Microsoft.Network/virtualNetworks/subnets
data: Provisioning state: Succeeded
data: Address prefix: 10.0.1.0/24
info: network vnet subnet set command OK
```

```
network vnet subnet show [options] <resource-group> <vnet-name> <name>
```

Displays virtual network subnet properties

```
azure network vnet subnet show -g myresourcegroup --vnet-name newvnet -n subnet1

info: Executing command network vnet subnet show
+ Looking up the subnet "subnet1"
data: Id:
/subscriptions/#####resourceGroups/myresourcegroup/providers/Microsoft.Network/virtualNetworks/newvnet/subnets/subnet1
data: Name: subnet1
data: Type: Microsoft.Network/virtualNetworks/subnets
data: Provisioning state: Succeeded
data: Address prefix: 10.0.1.0/24
info: network vnet subnet show command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-e, --vnet-name <vnet-name>                the name of the virtual network
-n, --name <name>                          the name of the subnet
-s, --subscription <subscription>          the subscription identifier
```

```
network vnet subnet delete [options] <resource-group> <vnet-name> <subnet-name>
```

Removes a subnet from an existing virtual network.

```
azure network vnet subnet delete -g myresourcegroup --vnet-name newvnet -n subnet1

info: Executing command network vnet subnet delete
+ Looking up the subnet "subnet1"
Delete subnet "subnet1"? [y/n] y
+ Deleting subnet "subnet1"
info: network vnet subnet delete command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-e, --vnet-name <vnet-name>                the name of the virtual network
-n, --name <name>                          the subnet name
-s, --subscription <subscription>          the subscription identifier
-q, --quiet                                quiet mode, do not ask for delete confirmation
```

Commands to manage load balancers

```
network lb create [options] <resource-group> <name> <location>
```

Creates a load balancer set.

```
azure network lb create -g myresourcegroup -n mylb -l westus

info: Executing command network lb create
+ Looking up the load balancer "mylb"
+ Creating load balancer "mylb"
+ Looking up the load balancer "mylb"
data: Id:
/subscriptions/#####
/resourceGroups/myresourcegroup/providers/Microsoft.Network/loadBalancers/mylb
data: Name:                      mylb
data: Type:                       Microsoft.Network/loadBalancers
data: Location:                  westus
data: Provisioning state:        Succeeded
info: network lb create command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the load balancer
-l, --location <location>                 the location
-t, --tags <tags>                         the list of tags.
                                            Can be multiple. In the format of "name=value".
                                            Name is required and value is optional. For example, -t tag1=value1;tag2
-s, --subscription <subscription>         the subscription identifier
```

```
network lb list [options] <resource-group>
```

Lists Load balancer resources within a resource group.

```
azure network lb list myresourcegroup

info: Executing command network lb list
+ Getting the load balancers
data: Name  Location
data: ----  -----
data: mylb  westus
info: network lb list command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-s, --subscription <subscription>         the subscription identifier
```

```
network lb show [options] <resource-group> <name>
```

Displays load balancer information of a specific load balancer within a resource group

```
azure network lb show myresourcegroup mylb -v

info: Executing command network lb show
verbose: Looking up the load balancer "mylb"
data: Id:
/subscriptions/########################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/loadB
alancers/mylb
data: Name:                      mylb
data: Type:                       Microsoft.Network/loadBalancers
data: Location:                  westus
data: Provisioning state:       Succeeded
info: network lb show command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the load balancer
-s, --subscription <subscription>         the subscription identifier
```

```
network lb delete [options] <resource-group> <name>
```

Delete load balancer resources.

```
azure network lb delete myresourcegroup mylb

info: Executing command network lb delete
+ Looking up the load balancer "mylb"
Delete load balancer "mylb"? [y/n] y
+ Deleting load balancer "mylb"
info: network lb delete command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the load balancer
-q, --quiet                               quiet mode, do not ask for delete confirmation
-s, --subscription <subscription>         the subscription identifier
```

Commands to manage probes of a load balancer

```
network lb probe create [options] <resource-group> <lb-name> <name>
```

Create the probe configuration for health status in the load balancer. Keep in mind to run this command, your load balancer requires a frontend-ip resource (Check out command "azure network frontend-ip" to assign an ip address to load balancer).

```
azure network lb probe create -g myresourcegroup --lb-name mylb -n mylbprobe --protocol tcp --port 80 -i 300

info: Executing command network lb probe create
+ Looking up the load balancer "mylb"
+ Updating load balancer "mylb"
info: network lb probe create command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the probe
-p, --protocol <protocol>	the probe protocol
-o, --port <port>	the probe port
-f, --path <path>	the probe path
-i, --interval <interval>	the probe interval in seconds
-c, --count <count>	the number of probes
-s, --subscription <subscription>	the subscription identifier

```
network lb probe set [options] <resource-group> <lb-name> <name>
```

Updates an existing load balancer probe with new values for it.

```
azure network lb probe set -g myresourcegroup -l mylb -n mylbprobe -p mylbprobe1 -p TCP -o 443 -i 300

info: Executing command network lb probe set
+ Looking up the load balancer "mylb"
+ Updating load balancer "mylb"
info: network lb probe set command OK
```

Parameter options

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the probe
-e, --new-probe-name <new-probe-name>	the new name of the probe
-p, --protocol <protocol>	the new value for probe protocol
-o, --port <port>	the new value for probe port
-f, --path <path>	the new value for probe path
-i, --interval <interval>	the new value for probe interval in seconds
-c, --count <count>	the new value for number of probes
-s, --subscription <subscription>	the subscription identifier

```
network lb probe list [options] <resource-group> <lb-name>
```

List the probe properties for a load balancer set.

```
C:\>azure network lb probe list -g myresourcegroup -l mylb

info: Executing command network lb probe list
+ Looking up the load balancer "mylb"
data: Name      Protocol Port Path Interval Count
data: -----  -----
data: mylbprobe Tcp      443     300      2
info: network lb probe list command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-l, --lb-name <lb-name>                   the name of the load balancer
-s, --subscription <subscription>        the subscription identifier
```

```
network lb probe delete [options] <resource-group> <lb-name> <name>
```

Removes the probe created for the load balancer.

```
azure network lb probe delete -g myresourcegroup -l mylb -n mylbprobe

info: Executing command network lb probe delete
+ Looking up the load balancer "mylb"
Delete a probe "mylbprobe?" [y/n] y
+ Updating load balancer "mylb"
info: network lb probe delete command OK
```

Commands to manage frontend ip configurations of a load balancer

```
network lb frontend-ip create [options] <resource-group> <lb-name> <name>
```

Creates a frontend IP configuration to an existing load balancer set.

```
azure network lb frontend-ip create -g myresourcegroup --lb-name mylb -n myfrontendip -o Dynamic -e subnet -m newvnet

info: Executing command network lb frontend-ip create
+ Looking up the load balancer "mylb"
+ Looking up the subnet "subnet"
+ Creating frontend IP configuration "myfrontendip"
+ Looking up the load balancer "mylb"
data: Id: /subscriptions/#####/resourceGroups/Myresourcegroup/providers/Microsoft.Network/loadBalancers/mylb/frontendIPConfigurations/myfrontendip
data: Name: myfrontendip
data: Type: Microsoft.Network/loadBalancers/frontendIPConfigurations
data: Provisioning state: Succeeded
data: Private IP allocation method: Dynamic
data: Private IP address: 10.0.1.4
data: Subnet: /subscriptions/#####/resourceGroups/myresourcegroup/providers/Microsoft.Network/virtualNetworks/newvnet/subnets/subnet
data: Public IP address:
data: Inbound NAT rules
data: Outbound NAT rules
data: Load balancing rules
data:
info: network lb frontend-ip create command OK
```

```
network lb frontend-ip set [options] <resource-group> <lb-name> <name>
```

Updates an existing configuration of a frontend IP. The command below adds a public IP called mypubip5 to an existing load balancer frontend IP named myfrontendip.

```

azure network lb frontend-ip set -g myresourcegroup --lb-name mylb -n myfrontendip -i mypubip5

info: Executing command network lb frontend-ip set
+ Looking up the load balancer "mylb"
+ Looking up the public ip "mypubip5"
+ Updating load balancer "mylb"
+ Looking up the load balancer "mylb"
data: Id:
/subscriptions/################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/loadB
alancers/mylb/frontendIPConfigurations/myfrontendip
data: Name: myfrontendip
data: Type: Microsoft.Network/loadBalancers/frontendIPConfigurations
data: Provisioning state: Succeeded
data: Private IP allocation method: Dynamic
data: Private IP address:
data: Subnet:
data: Public IP address:
id=/subscriptions/################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/pu
blicIPAddresses/mypubip5
data: Inbound NAT rules
data: Outbound NAT rules
data: Load balancing rules
data:
info: network lb frontend-ip set command OK

```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the frontend ip configuration
-a, --private-ip-address <private-ip-address>	the private ip address
-o, --private-ip-allocation-method <private-ip-allocation-method>	the private ip allocation method [Static, Dynamic]
-u, --public-ip-id <public-ip-id>	the public ip identifier.
e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Network/publicIPAddresses/<public-ip-name>	
-i, --public-ip-name <public-ip-name>	the public ip name.
This public ip must exist in the same resource group as the lb.	
Please use public-ip-id if that is not the case.	
-b, --subnet-id <subnet-id>	the subnet id.
e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Network/VirtualNetworks/<vnet-name>/subnets/<subnet-name>	
-e, --subnet-name <subnet-name>	the subnet name
-m, --vnet-name <vnet-name>	the virtual network name.
This virtual network must exist in the same resource group as the lb.	
Please use subnet-id if that is not the case.	
-s, --subscription <subscription>	the subscription identifier

```
network lb frontend-ip list [options] <resource-group> <lb-name>
```

Lists all the frontend IP resources configured for the load balancer.

```
azure network lb frontend-ip list -g myresourcegroup -l mylb

info: Executing command network lb frontend-ip list
+ Looking up the load balancer "mylb"
data: Name Provisioning state Private IP allocation method Subnet
data: -----
data: myprivateip Succeeded Dynamic
info: network lb frontend-ip list command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-s, --subscription <subscription>	the subscription identifier

```
network lb frontend-ip delete [options] <resource-group> <lb-name> <name>
```

Deletes the frontend IP object associated to load balancer

```
network lb frontend-ip delete -g myresourcegroup -l mylb -n myfrontendip
info: Executing command network lb frontend-ip delete
+ Looking up the load balancer "mylb"
Delete frontend ip configuration "myfrontendip"? [y/n] y
+ Updating load balancer "mylb"
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the frontend ip configuration
-q, --quiet	quiet mode, do not ask for delete confirmation
-s, --subscription <subscription>	the subscription identifier

Commands to manage backend address pools of a load balancer

```
network lb address-pool create [options] <resource-group> <lb-name> <name>
```

Create a backend address pool for a load balancer.

```
azure network lb address-pool create -g myresourcegroup --lb-name mylb -n myaddresspool

info: Executing command network lb address-pool create
+ Looking up the load balancer "mylb"
+ Updating load balancer "mylb"
+ Looking up the load balancer "mylb"
data: Id: /subscriptions/#####/resourceGroups/myresourgroup/providers/Microsoft.Network/loadBalancers/mylb/backendAddressPools/myaddresspool
data: Name: myaddresspool
data: Type: Microsoft.Network/loadBalancers/backendAddressPools
data: Provisioning state: Succeeded
data: Backend IP configurations:
data: Load balancing rules:
data:
info: network lb address-pool create command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the backend address pool
-s, --subscription <subscription>	the subscription identifier

```
network lb address-pool list [options] <resource-group> <lb-name>
```

List backend IP address pool range for a specific resource group

```
azure network lb address-pool list -g myresourcegroup -l mylb

info: Executing command network lb address-pool list
+ Looking up the load balancer "mylb"
data: Name Provisioning state
data: ----- -----
data: mybackendpool Succeeded
info: network lb address-pool list command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-s, --subscription <subscription>	the subscription identifier

```
network lb address-pool delete [options] <resource-group> <lb-name> <name>
```

Removes the backend IP pool range resource from load balancer.

```
azure network lb address-pool delete -g myresourcegroup -l mylb -n mybackendpool

info: Executing command network lb address-pool delete
+ Looking up the load balancer "mylb"
Delete backend address pool "mybackendpool"? [y/n] y
+ Updating load balancer "mylb"
info: network lb address-pool delete command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the backend address pool
-q, --quiet	quiet mode, do not ask for delete confirmation
-s, --subscription <subscription>	the subscription identifier

Commands to manage load balancer rules

```
network lb rule create [options] <resource-group> <lb-name> <name>
```

Create load balancer rules.

You can create a load balancer rule configuring the frontend endpoint for the load balancer and the backend address pool range to receive the incoming network traffic. Settings also include the ports for frontend IP endpoint and ports for the backend address pool range.

The following example shows how to create a load balancer rule, the frontend endpoint listening to port 80 TCP and load balancing network traffic sending to port 8080 for the backend address pool range.

```
azure network lb rule create -g myresourcegroup -l mylb -n mylbrule -p tcp -f 80 -b 8080 -i 10

info: Executing command network lb rule create
+ Looking up the load balancer "mylb"
+ Updating load balancer "mylb"
+ Loading rule state
data: Id:
/subscriptions/####################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/loadBalancers/mylb/loadBalancingRules/mylbrule
data: Name: mylbrule
data: Type: Microsoft.Network/loadBalancers/loadBalancingRules
data: Provisioning state: Succeeded
data: Frontend IP configuration:
/subscriptions/####################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/loadBalancers/mylb/frontendIPConfigurations/myfrontendip
data: Backend address pool:
id=/subscriptions/####################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/loadBalancers/mylb/backendAddressPools/mybackendpool
data: Protocol: Tcp
data: Frontend port: 80
data: Backend port: 8080
data: Enable floating IP: false
data: Idle timeout in minutes: 10
data: Probes
data:
info: network lb rule create command OK
```

```
network lb rule set [options] <resource-group> <lb-name> <name>
```

Updates an existing load balancer rule set in a specific resource group. In the following example, we changed the rule name from mylbrule to mynewlbrule.

```
azure network lb rule set -g myresourcegroup -l mylb -n mylbrule -r mynewlbrule -p tcp -f 80 -b 8080 -i 10 -t myfrontendip -o mybackendpool

info: Executing command network lb rule set
+ Looking up the load balancer "mylb"
+ Updating load balancer "mylb"
+ Loading rule state
data: Id:
/subscriptions/#####resourceGroups/yresourcegroup/providers/Microsoft.Network/loadBalancers/mylb/loadBalancingRules/mynewlbrule
data: Name: mynewlbrule
data: Type: Microsoft.Network/loadBalancers/loadBalancingRules
data: Provisioning state: Succeeded
data: Frontend IP configuration:
/subscriptions/#####resourceGroups/yresourcegroup/providers/Microsoft.Network/loadBalancers/mylb/frontendIPConfigurations/myfrontendip
data: Backend address pool:
id=/subscriptions/#####resourceGroups/yresourcegroup/providers/Microsoft.Network/loadBalancers/mylb/backendAddressPools/mybackendpool
data: Protocol: Tcp
data: Frontend port: 80
data: Backend port: 8080
data: Enable floating IP: false
data: Idle timeout in minutes: 10
data: Probes
data:
info: network lb rule set command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the rule
-r, --new-rule-name <new-rule-name>	new rule name
-p, --protocol <protocol>	the rule protocol
-f, --frontend-port <frontend-port>	the frontend port
-b, --backend-port <backend-port>	the backend port
-e, --enable-floating-ip <enable-floating-ip>	enable floating point ip
-i, --idle-timeout <idle-timeout>	the idle timeout in minutes
-a, --probe-name [probe-name]	the name of the probe defined in the same load balancer
-t, --frontend-ip-name <frontend-ip-name>	the name of the frontend ip configuration in the same load balancer
-o, --backend-address-pool <backend-address-pool>	name of the backend address pool defined in the same load balancer
-s, --subscription <subscription>	the subscription identifier

```
network lb rule list [options] <resource-group> <lb-name>
```

Lists all load balancer rules configured for a load balancer in a specific resource group.

```
azure network lb rule list -g myresourcegroup -l mylb

info: Executing command network lb rule list
+ Looking up the load balancer "mylb"
data: Name Provisioning state Protocol Frontend port Backend port Enable floating IP Idle
timeout in minutes Backend address pool Probe data

data: mynewlrule Succeeded Tcp 80 8080 false 10
/subscriptions/#####resourceGroups/myresourcegroup/providers/Microsoft.Network/loadB
alancers/mylb/backendAddressPools/mybackendpool
info: network lb rule list command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-s, --subscription <subscription>	the subscription identifier


```
network lb rule delete [options] <resource-group> <lb-name> <name>
```

Deletes a load balancer rule.

```
azure network lb rule delete -g myresourcegroup -l mylb -n mynewlrule

info: Executing command network lb rule delete
+ Looking up the load balancer "mylb"
Delete load balancing rule mynewlrule? [y/n] y
+ Updating load balancer "mylb"
info: network lb rule delete command OK
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the rule
-q, --quiet	quiet mode, do not ask for delete confirmation
-s, --subscription <subscription>	the subscription identifier

Commands to manage load balancer inbound NAT rules

```
network lb inbound-nat-rule create [options] <resource-group> <lb-name> <name>
```

Creates an inbound NAT rule for load balancer.

In the following example we created a NAT rule from frontend IP (which was previously defined using the "azure network frontend-ip" command) with an inbound listening port and outbound port that the load balancer uses to send the network traffic.

```

azure network lb inbound-nat-rule create -g myresourcegroup -l mylb -n myinboundnat -p tcp -f 80 -b 8080 -i
myfrontendip

info: Executing command network lb inbound-nat-rule create
+ Looking up the load balancer "mylb"
+ Updating load balancer "mylb"
+ Looking up the load balancer "mylb"
data: Id:
/subscriptions/####################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/loadB
alancers/mylb/inboundNatRules/myinboundnat
data: Name: myinboundnat
data: Type: Microsoft.Network/loadBalancers/inboundNatRules
data: Provisioning state: Succeeded
data: Frontend IP Configuration:
id=/subscriptions/####################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/lo
adBalancers/mylb/frontendIPConfigurations/myfrontendip
data: Backend IP configuration
data: Protocol: Tcp
data: Frontend port: 80
data: Backend port: 8080
data: Enable floating IP: false
info: network lb inbound-nat-rule create command OK

```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the inbound NAT rule
-p, --protocol <protocol>	the rule protocol [tcp,udp]
-f, --frontend-port <frontend-port>	the frontend port [0-65535]
-b, --backend-port <backend-port>	the backend port [0-65535]
-e, --enable-floating-ip <enable-floating-ip>	enable floating point ip [true,false]
-i, --frontend-ip <frontend-ip>	the name of the frontend ip configuration
-m, --vm-id <vm-id>	the VM id.
e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group- name>/providers/Microsoft.Compute/virtualMachines/<vm-name>	
-a, --vm-name <vm-name>	the VM name.This VM must exist in the same resource group as the lb.
Please use vm-id if that is not the case.	
this parameter will be ignored if --vm-id is specified	
-s, --subscription <subscription>	the subscription identifier

```
network lb inbound-nat-rule set [options] <resource-group> <lb-name> <name>
```

Updates an existing inbound nat rule. In the following example, we changed the inbound listening port from 80 to 81.

```

azure network lb inbound-nat-rule set -g group-1 -l mylb -n myinboundnat -p tcp -f 81 -b 8080 -i myfrontendip

info: Executing command network lb inbound-nat-rule set
+ Looking up the load balancer "mylb"
+ Updating load balancer "mylb"
+ Looking up the load balancer "mylb"
data: Id: /subscriptions/#####resourceGroups/group-1/providers/Microsoft.Network/loadBalancers/mylb/inboundNatRules/myinboundnat
data: Name: myinboundnat
data: Type: Microsoft.Network/loadBalancers/inboundNatRules
data: Provisioning state: Succeeded
data: Frontend IP Configuration: id=/subscriptions/#####resourceGroups/group-1/providers/Microsoft.Network/loadBalancers/mylb/frontendIPConfigurations/myfrontendip
data: Backend IP configuration
data: Protocol: Tcp
data: Frontend port: 81
data: Backend port: 8080
data: Enable floating IP: false
info: network lb inbound-nat-rule set command OK

```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-l, --lb-name <lb-name>	the name of the load balancer
-n, --name <name>	the name of the inbound NAT rule
-p, --protocol <protocol>	the rule protocol [tcp,udp]
-f, --frontend-port <frontend-port>	the frontend port [0-65535]
-b, --backend-port <backend-port>	the backend port [0-65535]
-e, --enable-floating-ip <enable-floating-ip>	enable floating point ip [true,false]
-i, --frontend-ip <frontend-ip>	the name of the frontend ip configuration
-m, --vm-id [vm-id]	the VM id.
e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Compute/virtualMachines/<vm-name>	
-a, --vm-name <vm-name>	the VM name. This virtual machine must exist in the same resource group as the lb. Please use vm-id if that is not the case
-s, --subscription <subscription>	the subscription identifier

```
network lb inbound-nat-rule list [options] <resource-group> <lb-name>
```

Lists all inbound nat rules for load balancer.

```

azure network lb inbound-nat-rule list -g myresourcegroup -l mylb

info: Executing command network lb inbound-nat-rule list
+ Looking up the load balancer "mylb"
data: Name Provisioning state Protocol Frontend port Backend port Enable floating IP Idle
timeout in minutes Backend IP configuration
data: ----- -----
----- ---
-----
data: myinboundnat Succeeded Tcp 81 8080 false 4
info: network lb inbound-nat-rule list command OK

```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-l, --lb-name <lb-name>                   the name of the load balancer
-s, --subscription <subscription>        the subscription identifier
```

```
network lb inbound-nat-rule delete [options] <resource-group> <lb-name> <name>
```

Deletes NAT rule for the load balancer in a specific resource group.

```
azure network lb inbound-nat-rule delete -g myresourcegroup -l mylb -n myinboundnat

info: Executing command network lb inbound-nat-rule delete
+ Looking up the load balancer "mylb"
Delete inbound NAT rule "myinboundnat?" [y/n] y
+ Updating load balancer "mylb"
info: network lb inbound-nat-rule delete command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-l, --lb-name <lb-name>                   the name of the load balancer
-n, --name <name>                         the name of the inbound NAT rule
-q, --quiet                               quiet mode, do not ask for delete confirmation
-s, --subscription <subscription>        the subscription identifier
```

Commands to manage public ip addresses

```
network public-ip create [options] <resource-group> <name> <location>
```

Creates a public ip resource. You will create the public ip resource and associate to a domain name.

```
azure network public-ip create -g myresourcegroup -n mytestpublicip1 -l eastus -d azureclitest -a "Dynamic"
info: Executing command network public-ip create
+ Looking up the public ip "mytestpublicip1"
+ Creating public ip address "mytestpublicip1"
+ Looking up the public ip "mytestpublicip1"
data: Id:
/subscriptions/########################################/resourceGroups/myresourcegroup/providers/Microsoft.Network/publicIPAddresses/mytestpublicip1
data: Name:                      mytestpublicip1
data: Type:                      Microsoft.Network/publicIPAddresses
data: Location:                 eastus
data: Provisioning state:       Succeeded
data: Allocation method:         Dynamic
data: Idle timeout:             4
data: Domain name label:        azureclitest
data: FQDN:                      azureclitest.eastus.cloudapp.azure.com
info: network public-ip create command OK
```

Parameter options:

```

-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the public ip
-l, --location <location>                 the location
-d, --domain-name-label <domain-name-label> the domain name label.
This set DNS to <domain-name-label>.<location>.cloudapp.azure.com
-a, --allocation-method <allocation-method> the allocation method [Static][Dynamic]
-i, --idletimeout <idletimeout>           the idle timeout in minutes
-f, --reverse-fqdn <reverse-fqdn>        the reverse fqdn
-t, --tags <tags>                         the list of tags.
Can be multiple. In the format of "name=value".
Name is required and value is optional.
For example, -t tag1=value1;tag2
-s, --subscription <subscription>         the subscription identifier

```

```
network public-ip set [options] <resource-group> <name>
```

Updates the properties of an existing public ip resource. In the following example we changed the public IP address from Dynamic to Static.

```

azure network public-ip set -g group-1 -n mytestpublicip1 -d azureclitest -a "Static"
info:   Executing command network public-ip set
+ Looking up the public ip "mytestpublicip1"
+ Updating public ip address "mytestpublicip1"
+ Looking up the public ip "mytestpublicip1"
data:   Id:
/subscriptions/#####resourceGroups/myresourcegroup/providers/Microsoft.Network/publicIPAddresses/mytestpublicip1
data:   Name:          mytestpublicip1
data:   Type:          Microsoft.Network/publicIPAddresses
data:   Location:      eastus
data:   Provisioning state: Succeeded
data:   Allocation method: Static
data:   Idle timeout:   4
data:   IP Address:    (static IP address)
data:   Domain name label: azureclitest
data:   FQDN:          azureclitest.eastus.cloudapp.azure.com
info:   network public-ip set command OK

```

Parameter options:

```

-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the public ip
-d, --domain-name-label <domain-name-label> the domain name label.
This set DNS to <domain-name-label>.<location>.cloudapp.azure.com
-a, --allocation-method <allocation-method> the allocation method [Static][Dynamic]
-i, --idletimeout <idletimeout>           the idle timeout in minutes
-f, --reverse-fqdn <reverse-fqdn>        the reverse fqdn
-t, --tags <tags>                         the list of tags.
Can be multiple. In the format of "name=value".
Name is required and value is optional.
For example, -t tag1=value1;tag2
--no-tags                                 remove all existing tags
-s, --subscription <subscription>         the subscription identifier

```

```
network public-ip list [options] <resource-group>
```

Lists all public IP resources within a resource group.

```
azure network public-ip list -g myresourcegroup

info: Executing command network public-ip list
+ Getting the public ip addresses
data: Name Location Allocation IP Address Idle timeout DNS Name
data: -----
data: mypubip5 westus Dynamic 4 "domain
name".westus.cloudapp.azure.com
data: myPublicIP eastus Dynamic 4 "domain
name".eastus.cloudapp.azure.com
data: mytestpublicip eastus Dynamic 4 "domain
name".eastus.cloudapp.azure.com
data: mytestpublicip1 eastus Static (Static IP address) 4
azureclitest.eastus.cloudapp.azure.com
```

Parameter options:

-h, --help	output usage information
-v, --verbose	use verbose output
--json	use json output
-g, --resource-group <resource-group>	the name of the resource group
-s, --subscription <subscription>	the subscription identifier

```
network public-ip show [options] <resource-group> <name>
```

Displays public ip properties for a public ip resource within a resource group.

```
azure network public-ip show -g myresourcegroup -n mytestpublicip

info: Executing command network public-ip show
+ Looking up the public ip "mytestpublicip1"
data: Id:
/subscriptions/#####
/resourceGroups/myresourcegroup/providers/Microsoft.Network/publicIPAddresses/mytestpublicip
data: Name: mytestpublicip
data: Type: Microsoft.Network/publicIPAddresses
data: Location: eastus
data: Provisioning state: Succeeded
data: Allocation method: Static
data: Idle timeout: 4
data: IP Address: (static IP address)
data: Domain name label: azureclitest
data: FQDN: azureclitest.eastus.cloudapp.azure.com
info: network public-ip show command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the public IP
-s, --subscription <subscription>         the subscription identifier
```

```
network public-ip delete [options] <resource-group> <name>
```

Deletes public ip resource.

```
azure network public-ip delete -g group-1 -n mypublicipname
info:   Executing command network public-ip delete
+ Looking up the public ip "mypublicipname"
Delete public ip address "mypublicipname"? [y/n] y
+ Deleting public ip address "mypublicipname"
info:   network public-ip delete command OK
```

Parameter options:

```
-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the public IP
-q, --quiet                               quiet mode, do not ask for delete confirmation
-s, --subscription <subscription>         the subscription identifier
```

Commands to manage network interfaces

```
network nic create [options] <resource-group> <name> <location>
```

Creates a resource called network interface (NIC) which can be used for load balancers or associate to a Virtual Machine.

```
azure network nic create -g myresourcegroup -l eastus -n testnic1 --subnet-name subnet-1 --subnet-vnet-name
myvnet

info:   Executing command network nic create
+ Looking up the network interface "testnic1"
+ Looking up the subnet "subnet-1"
+ Creating network interface "testnic1"
+ Looking up the network interface "testnic1"
data:   Id:          /subscriptions/c4a17ddf-aa84-491c-b6f9-b90d882299f7/resourceGroups/group-
1/providers/Microsoft.Network/networkInterfaces/testnic1
data:   Name:        testnic1
data:   Type:        Microsoft.Network/networkInterfaces
data:   Location:   eastus
data:   Provisioning state: Succeeded
data:   IP configurations:
data:     Name:           NIC-config
data:     Provisioning state: Succeeded
data:     Private IP address: 10.0.0.5
data:     Private IP Allocation Method: Dynamic
data:     Subnet:          /subscriptions/c4a17ddf-aa84-491c-b6f9-
b90d882299f7/resourceGroups/group-1/providers/Microsoft.Network/virtualNetworks/myVNET/subnets/Subnet-1
```

Parameter options:

```

-h, --help                                output usage information
-v, --verbose                             use verbose output
--json                                    use json output
-g, --resource-group <resource-group>    the name of the resource group
-n, --name <name>                          the name of the network interface
-l, --location <location>                 the location
-w, --network-security-group-id <network-security-group-id> the network security group identifier.
e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Network/networkSecurityGroups/<nsg-name>
-o, --network-security-group-name <network-security-group-name> the network security group name.
This network security group must exist in the same resource group as the nic.
Please use network-security-group-id if that is not the case.
-i, --public-ip-id <public-ip-id>          the public IP identifier.
e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Network/publicIPAddresses/<public-ip-name>
-p, --public-ip-name <public-ip-name>        the public IP name.
This public ip must exist in the same resource group as the nic.
Please use public-ip-id if that is not the case.
-a, --private-ip-address <private-ip-address> the private IP address
-u, --subnet-id <subnet-id>                  the subnet identifier.
e.g. /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Network/virtualNetworks/<vnet-name>/subnets/<subnet-name>
--subnet-name <subnet-name>                  the subnet name
-m, --subnet-vnet-name <subnet-vnet-name>    the vnet name under which subnet-name exists
-t, --tags <tags>                           the comma separated list of tags.
Can be multiple. In the format of "name=value".
Name is required and value is optional.
For example, -t tag1=value1;tag2
-s, --subscription <subscription>            the subscription identifier
data:
info:   network nic create command OK

```

```

network nic set [options] <resource-group> <name>
network nic list [options] <resource-group>
network nic show [options] <resource-group> <name>
network nic delete [options] <resource-group> <name>

```

Commands to manage network security groups

```

network nsg create [options] <resource-group> <name> <location>
network nsg set [options] <resource-group> <name>
network nsg list [options] <resource-group>
network nsg show [options] <resource-group> <name>
network nsg delete [options] <resource-group> <name>

```

Commands to manage network security group rules

```

network nsg rule create [options] <resource-group> <nsg-name> <name>
network nsg rule set [options] <resource-group> <nsg-name> <name>
network nsg rule list [options] <resource-group> <nsg-name>
network nsg rule show [options] <resource-group> <nsg-name> <name>
network nsg rule delete [options] <resource-group> <nsg-name> <name>

```

Commands to manage traffic manager profile

```
network traffic-manager profile create [options] <resource-group> <name>
network traffic-manager profile set [options] <resource-group> <name>
network traffic-manager profile list [options] <resource-group>
network traffic-manager profile show [options] <resource-group> <name>
network traffic-manager profile delete [options] <resource-group> <name>
network traffic-manager profile is-dns-available [options] <resource-group> <relative-dns-name>
```

Commands to manage traffic manager endpoints

```
network traffic-manager profile endpoint create [options] <resource-group> <profile-name> <name> <endpoint-location>
network traffic-manager profile endpoint set [options] <resource-group> <profile-name> <name>
network traffic-manager profile endpoint delete [options] <resource-group> <profile-name> <name>
```

Commands to manage virtual network gateways

```
network gateway list [options] <resource-group>
```

Resource provider registrations

List currently registered providers in Resource Manager

```
provider list [options]
```

Show details about the requested provider namespace

```
provider show [options] <namespace>
```

Register provider with the subscription

```
provider register [options] <namespace>
```

Unregister provider with the subscription

```
provider unregister [options] <namespace>
```

Resources

Creates a resource in a resource group

```
resource create [options] <resource-group> <name> <resource-type> <location> <api-version>
```

Updates a resource in a resource group without any templates or parameters

```
resource set [options] <resource-group> <name> <resource-type> <properties> <api-version>
```

Lists the resources

```
resource list [options] [resource-group]
```

Gets one resource within a resource group or subscription

```
resource show [options] <resource-group> <name> <resource-type> <api-version>
```

Deletes a resource in a resource group

```
resource delete [options] <resource-group> <name> <resource-type> <api-version>
```

Azure roles

Get all available role definitions

```
role list [options]
```

Get an available role definition

```
role show [options] [name]
```

Commands to manage your role assignment

```
role assignment create [options] [objectId] [upn] [mail] [spn] [role] [scope] [resource-group] [resource-type] [resource-name]
role assignment list [options] [objectId] [upn] [mail] [spn] [role] [scope] [resource-group] [resource-type] [resource-name]
role assignment delete [options] [objectId] [upn] [mail] [spn] [role] [scope] [resource-group] [resource-type] [resource-name]
```

Storage objects

Commands to manage your Storage accounts

```
storage account list [options]
storage account show [options] <name>
storage account create [options] <name>
storage account set [options] <name>
storage account delete [options] <name>
```

Commands to manage your Storage account keys

```
storage account keys list [options] <name>
storage account keys renew [options] <name>
```

Commands to show your Storage connection string

```
storage account connectionstring show [options] <name>
```

Commands to manage your Storage containers

```
storage container list [options] [prefix]
storage container show [options] [container]
storage container create [options] [container]
storage container delete [options] [container]
storage container set [options] [container]
```

Commands to manage shared access signatures of your Storage container

```
storage container sas create [options] [container] [permissions] [expiry]
```

Commands to manage stored access policies of your Storage container

```
storage container policy create [options] [container] [name]
storage container policy show [options] [container] [name]
storage container policy list [options] [container]
storage container policy set [options] [container] [name]
storage container policy delete [options] [container] [name]
```

Commands to manage your Storage blobs

```
storage blob list [options] [container] [prefix]
storage blob show [options] [container] [blob]
storage blob delete [options] [container] [blob]
storage blob upload [options] [file] [container] [blob]
storage blob download [options] [container] [blob] [destination]
```

Commands to manage your blob copy operations

```
storage blob copy start [options] [sourceUri] [destContainer]
storage blob copy show [options] [container] [blob]
storage blob copy stop [options] [container] [blob] [copyid]
```

Commands to manage shared access signature of your Storage blob

```
storage blob sas create [options] [container] [blob] [permissions] [expiry]
```

Commands to manage your Storage file shares

```
storage share create [options] [share]
storage share show [options] [share]
storage share delete [options] [share]
storage share list [options] [prefix]
```

Commands to manage your Storage files

```
storage file list [options] [share] [path]
storage file delete [options] [share] [path]
storage file upload [options] [source] [share] [path]
storage file download [options] [share] [path] [destination]
```

Commands to manage your Storage file directory

```
storage directory create [options] [share] [path]
storage directory delete [options] [share] [path]
```

Commands to manage your Storage queues

```
storage queue create [options] [queue]
storage queue list [options] [prefix]
storage queue show [options] [queue]
storage queue delete [options] [queue]
```

Commands to manage shared access signatures of your Storage queue

```
storage queue sas create [options] [queue] [permissions] [expiry]
```

Commands to manage stored access policies of your Storage queue

```
storage queue policy create [options] [queue] [name]
storage queue policy show [options] [queue] [name]
storage queue policy list [options] [queue]
storage queue policy set [options] [queue] [name]
storage queue policy delete [options] [queue] [name]
```

Commands to manage your Storage logging properties

```
storage logging show [options]
storage logging set [options]
```

Commands to manage your Storage metrics properties

```
storage metrics show [options]
storage metrics set [options]
```

Commands to manage your Storage tables

```
storage table create [options] [table]
storage table list [options] [prefix]
storage table show [options] [table]
storage table delete [options] [table]
```

Commands to manage shared access signatures of your Storage table

```
storage table sas create [options] [table] [permissions] [expiry]
```

Commands to manage stored access policies of your Storage table

```
storage table policy create [options] [table] [name]
storage table policy show [options] [table] [name]
storage table policy list [options] [table]
storage table policy set [options] [table] [name]
storage table policy delete [options] [table] [name]
```

Tags

Add a tag

```
tag create [options] <name> <value>
```

Remove an entire tag or a tag value

```
tag delete [options] <name> <value>
```

Lists the tag information

```
tag list [options]
```

Get a tag

```
tag show [options] [name]
```

Virtual Machines

Create a VM

```
vm create [options] <resource-group> <name> <location> <os-type>
```

Create a VM with default resources

```
vm quick-create [options] <resource-group> <name> <location> <os-type> <image-urn> <admin-username> <admin-password>
```

TIP

Starting with CLI version 0.10, you can provide a short alias such as "UbuntuLTS" or "Win2012R2Datacenter" as the `image-urn` for some popular Marketplace images. Run `azure help vm quick-create` for options. Additionally, starting with version 0.10, `azure vm quick-create` uses premium storage by default if it's available in the selected region.

List the virtual machines within an account

```
vm list [options]
```

Get one virtual machine within a resource group

```
vm show [options] <resource-group> <name>
```

Delete one virtual machine within a resource group

```
vm delete [options] <resource-group> <name>
```

Shutdown one virtual machine within a resource group

```
vm stop [options] <resource-group> <name>
```

Restart one virtual machine within a resource group

```
vm restart [options] <resource-group> <name>
```

Start one virtual machine within a resource group

```
vm start [options] <resource-group> <name>
```

Shutdown one virtual machine within a resource group and releases the compute resources

```
vm deallocate [options] <resource-group> <name>
```

List available virtual machine sizes

```
vm sizes [options]
```

Capture the VM as OS Image or VM Image

```
vm capture [options] <resource-group> <name> <vhd-name-prefix>
```

Set the state of the VM to Generalized

```
vm generalize [options] <resource-group> <name>
```

Get instance view of the VM

```
vm get-instance-view [options] <resource-group> <name>
```

Enable you to reset Remote Desktop Access or SSH settings on a Virtual Machine and to reset the password for the account that has administrator or sudo authority

```
vm reset-access [options] <resource-group> <name>
```

Update VM with new data

```
vm set [options] <resource-group> <name>
```

Commands to manage your Virtual Machine data disks

```
vm disk attach-new [options] <resource-group> <vm-name> <size-in-gb> [vhd-name]
vm disk detach [options] <resource-group> <vm-name> <lun>
vm disk attach [options] <resource-group> <vm-name> [vhd-url]
```

Commands to manage VM resource extensions

```
vm extension set [options] <resource-group> <vm-name> <name> <publisher-name> <version>
vm extension get [options] <resource-group> <vm-name>
```

Commands to manage your Docker Virtual Machine

```
vm docker create [options] <resource-group> <name> <location> <os-type>
```

Commands to manage VM images

```
vm image list-publishers [options] <location>
vm image list-offers [options] <location> <publisher>
vm image list-skus [options] <location> <publisher> <offer>
vm image list [options] <location> <publisher> [<offer>] [<sku>]
```