

Contents

[Linux VMs Documentation](#)

[Overview](#)

[About Virtual Machines](#)

[Quickstarts](#)

[Create VM - Azure CLI](#)

[Create VM - Portal](#)

[Create VM - Azure PowerShell](#)

[Tutorials](#)

[1 - Create / manage VMs](#)

[2 - Create / manage disks](#)

[3 - Automate configuration](#)

[4 - Create VM images](#)

[5 - Highly available VMs](#)

[6 - Create a scale set](#)

[7 - Load balance VMs](#)

[8 - Manage networking](#)

[9 - Backup virtual machines](#)

[10 - Govern VMs](#)

[11 - Monitor and update VMs](#)

[12 - Manage VM security](#)

[13 - Deploy Jenkins](#)

[14 - CI/CD with Azure Pipelines](#)

[15a - Create LAMP stack](#)

[15b - Create LEMP stack](#)

[15c - Create MEAN stack](#)

[16 - Secure web server with SSL](#)

[Samples](#)

[Azure CLI](#)

[Azure PowerShell](#)

Concepts

Azure Resource Manager

Regions and availability

VM types and sizes

Generation 2 VMs

General purpose

B-series burstable

Compute optimized

Memory optimized

Constrained vCPUs

Storage optimized

Optimize performance

Accelerated compute

GPU optimized

Setup GPU drivers

High performance compute

Previous generations

Azure compute units (ACU)

Benchmark scores

Endorsed distros

Maintenance and updates

Disk storage

Introduction to managed disks

Select a disk type for IaaS VMs

Designing for high performance

Scalability targets for disks

Backup and disaster recovery for disks

Ephemeral OS disks

Networking

Scale sets

Infrastructure automation

Security and policy

[States and lifecycle](#)

[Monitoring](#)

[Backup and recovery](#)

[Deployment considerations](#)

[Infrastructure guidelines](#)

[vCPU quotas](#)

[How-to guides](#)

[Create VMs](#)

[Use the CLI](#)

[Use a template](#)

[Use REST API](#)

[Copy or clone a VM](#)

[Secure VMs](#)

[Just-in-time access](#)

[Encrypt](#)

[Use access controls](#)

[Use policies](#)

[Create a Key Vault](#)

[Create and use SSH keys](#)

[On Linux or macOS](#)

[On Windows](#)

[Detailed steps](#)

[Protect VMs](#)

[Back up VMs](#)

[Back up a single VM](#)

[Back up multiple VMs](#)

[Restore a disk](#)

[Restore individual files](#)

[Set up disaster recovery for VMs](#)

[Enable disaster recovery for a VM](#)

[Run a disaster recovery drill for a VM](#)

[Fail over a VM to another region](#)

Manage VMs

[VM usage](#)

[Common CLI tasks](#)

[Change VM size](#)

[Swap the OS disk](#)

[Time sync](#)

[Tag a VM](#)

[Run scripts on a VM](#)

[Custom Script Extension](#)

[Run Command](#)

[Use Remote Desktop](#)

[Join VM to Azure Active Directory](#)

[Red Hat Enterprise Linux](#)

[CentOS](#)

[Ubuntu](#)

[Sign in with Azure Active Directory credentials](#)

[Updates and patches](#)

[Red Hat Update Infrastructure](#)

[Azure VM agent](#)

[Overview](#)

[Agent update](#)

[Planned maintenance](#)

[Mitigating speculative execution](#)

[Scheduled events](#)

[Monitor metadata](#)

[Get usage metrics with REST](#)

Manage costs

[Prepay for VMs - Azure reservations](#)

[Prepay for Azure software plans](#)

[What are Azure reservations?](#)

[VM instance size flexibility](#)

Use Images

[Shared image galleries](#)

[Overview](#)

[CLI](#)

[Portal](#)

[Share images across tenants](#)

[Troubleshoot shared images](#)

[Image builder \(preview\)](#)

[Overview](#)

[Use Azure CLI](#)

[Template reference](#)

[Build for image galleries](#)

[Update an existing image](#)

[Troubleshoot](#)

[Find and use images](#)

[Create custom image](#)

[Generic steps](#)

[Ubuntu](#)

[CentOS](#)

[Red Hat](#)

[Debian](#)

[SUSE](#)

[Oracle Linux](#)

[OpenBSD](#)

[FreeBSD](#)

[Capture VM to image](#)

[Build image with Packer](#)

[RHEL images in Azure](#)

[Download existing disk](#)

[Availability and scale](#)

[Autoscale](#)

[High availability](#)

[Vertically scale](#)

Create VM in availability zone

Use automation tools

Ansible

Install and configure

Create a Linux VM

Manage a Linux VM

Terraform

Install and configure

Create a complete VM

Cloud-init

Cloud-init overview

Configure VM hostname

Update packages in a VM

Add a user on a VM

Configure swapfile

Run existing bash script

Prepare existing VM for cloud-init

Jenkins

Create a Jenkins server

Scale with VM agents

Publish artifacts to Storage

Secure Jenkins

Run containers

Create Docker host

Use Docker Machine

Use Docker Compose

Run applications

Cloud Foundry

Overview

Deploy your first app

Cassandra

OpenShift

- [OpenShift overview](#)
- [OpenShift prerequisites](#)
- [OpenShift Container Platform](#)
- [OpenShift Marketplace Self-Managed](#)
- [Azure Stack](#)
- [OpenShift post-deployment tasks](#)
- [Troubleshooting deployments](#)
- [SAP on Azure](#)
- [Oracle](#)
- [Elasticsearch](#)
- [FreeBSD Packet Filter](#)
- [Databases](#)
 - [MySQL on SUSE](#)
 - [MongoDB](#)
 - [PostgreSQL](#)
 - [MS SQL on Linux](#)
- [High Performance Computing \(HPC\)](#)
- [IBM Db2 pureScale](#)
 - [Architecture](#)
 - [Deployment](#)
- [Manage storage](#)
 - [Add a disk](#)
 - [Azure CLI](#)
 - [Azure portal](#)
 - [Detach a disk](#)
 - [Deploy disks with template](#)
 - [Resize a disk](#)
 - [Snapshot a disk](#)
 - [Back up unmanaged disks](#)
 - [Migration and conversion](#)
 - [Migrate to Premium storage with Azure Site Recovery](#)
 - [Convert to Managed Disks](#)

- [Convert disk between Standard and Premium](#)
- [Copy files to a VM](#)
- [Performance](#)
 - [Using write accelerator](#)
 - [Enable ultra SSDs](#)
 - [Benchmark a disk](#)
 - [Optimizing performance](#)
 - [Configure software RAID](#)
 - [Configure LVM](#)
- [Find unattached disks](#)
- [Use File storage](#)
- [Disks FAQs](#)
- [Manage networking](#)
 - [Create virtual network](#)
 - [Open ports to a VM](#)
 - [Assign public IP address](#)
 - [Use multiple NICs](#)
 - [Use accelerated networking](#)
 - [Assign public DNS name](#)
 - [Find and delete unattached NICs](#)
 - [DNS resolution](#)
 - [Use internal DNS](#)
- [Configure managed identities](#)
 - [Portal](#)
 - [CLI](#)
 - [PowerShell](#)
 - [Azure Resource Manager Template](#)
 - [REST](#)
 - [Azure SDKs](#)
- [Use VM extensions](#)
- [Move and migrate VMs](#)
- [Change subscription or resource group](#)

- [Move VMs to another region](#)
- [Move to an availability zone](#)
- [Migrate AWS and on-premises VMs](#)
 - [Migrate from Amazon Web Services \(AWS\) to Azure](#)
 - [Upload on-prem VM](#)
 - [Use Azure Site Recovery](#)
- [Migrate from Classic to Azure Resource Manager](#)
 - [Deep dive on migration](#)
 - [Plan for migration](#)
 - [Migrate using the CLI](#)
 - [Common migration errors](#)
 - [Community tools for migration](#)
- [FAQ](#)

Reference

- [Azure CLI](#)
- [PowerShell](#)
- [.NET](#)
- [Java](#)
- [Node.js](#)
- [Python](#)
- [REST](#)
- [Resource Manager template](#)

Resources

- [Author templates](#)
- [Build your skills with Microsoft Learn](#)
- [Azure Roadmap](#)
- [Azure Quickstart templates](#)
- [Pricing](#)
- [Regional availability](#)
- [Stack Overflow](#)
- [Videos](#)
- [FAQ](#)

Troubleshoot

Azure and Linux

3/14/2019 • 7 minutes to read • [Edit Online](#)

Microsoft Azure is a growing collection of integrated public cloud services including analytics, virtual machines, databases, mobile, networking, storage, and web—ideal for hosting your solutions. Microsoft Azure provides a scalable computing platform that allows you to only pay for what you use, when you want it - without having to invest in on-premises hardware. Azure is ready when you are to scale your solutions up and out to whatever scale you require to service the needs of your clients.

If you are familiar with the various features of Amazon's AWS, you can examine the [Azure vs AWS definition mapping document](#).

Regions

Microsoft Azure resources are distributed across multiple geographical regions around the world. A "region" represents multiple data centers in a single geographical area. Azure currently (as of August 2018) has 42 regions generally available around the world with an additional 12 regions announced - more global regions than any other cloud provider. An updated list of existing and newly announced regions can be found in the following page:

- [Azure Regions](#)

Availability

Azure announced an industry leading single instance virtual machine Service Level Agreement of 99.9% provided you deploy the VM with premium storage for all disks. In order for your deployment to qualify for the standard 99.95% VM Service Level Agreement, you still need to deploy two or more VMs running your workload inside of an availability set. An availability set ensures that your VMs are distributed across multiple fault domains in the Azure data centers as well as deployed onto hosts with different maintenance windows. The full [Azure SLA](#) explains the guaranteed availability of Azure as a whole.

Managed Disks

Managed Disks handles Azure Storage account creation and management in the background for you, and ensures that you do not have to worry about the scalability limits of the storage account. You specify the disk size and the performance tier (Standard or Premium), and Azure creates and manages the disk. As you add disks or scale the VM up and down, you don't have to worry about the storage being used. If you're creating new VMs, [use the Azure CLI](#) or the Azure portal to create VMs with Managed OS and data disks. If you have VMs with unmanaged disks, you can [convert your VMs to be backed with Managed Disks](#).

You can also manage your custom images in one storage account per Azure region, and use them to create hundreds of VMs in the same subscription. For more information about Managed Disks, see the [Managed Disks Overview](#).

Azure Virtual Machines & Instances

Microsoft Azure supports running a number of popular Linux distributions provided and maintained by a number of partners. You can find distributions such as Red Hat Enterprise, CentOS, SUSE Linux Enterprise, Debian, Ubuntu, CoreOS, RancherOS, FreeBSD, and more in the Azure Marketplace. Microsoft actively works with various Linux communities to add even more flavors to the [Azure endorsed Linux Distros](#) list.

If your preferred Linux distro of choice is not currently present in the gallery, you can "Bring your own Linux" VM

by [creating and uploading a Linux VHD in Azure](#).

Azure virtual machines allow you to deploy a wide range of computing solutions in an agile way. You can deploy virtually any workload and any language on nearly any operating system - Windows, Linux, or a custom created one from any one of the growing list of partners. Still don't see what you are looking for? Don't worry - you can also bring your own images from on-premises.

VM Sizes

The [size](#) of the VM that you use is determined by the workload that you want to run. The size that you choose then determines factors such as processing power, memory, and storage capacity. Azure offers a wide variety of sizes to support many types of uses.

Azure charges an [hourly price](#) based on the VM's size and operating system. For partial hours, Azure charges only for the minutes used. Storage is priced and charged separately.

Automation

To achieve a proper DevOps culture, all infrastructure must be code. When all the infrastructure lives in code it can easily be recreated (Phoenix Servers). Azure works with all the major automation tooling like Ansible, Chef, SaltStack, and Puppet. Azure also has its own tooling for automation:

- [Azure Templates](#)
- [Azure VMAccess](#)

Azure is rolling out support for [cloud-init](#) across most Linux Distros that support it. Currently Canonical's Ubuntu VMs are deployed with cloud-init enabled by default. Red Hat's RHEL, CentOS, and Fedora support cloud-init, however the Azure images maintained by Red Hat do not currently have cloud-init installed. To use cloud-init on a Red Hat family OS, you must create a custom image with cloud-init installed.

- [Using cloud-init on Azure Linux VMs](#)

Quotas

Each Azure Subscription has default quota limits in place that could impact the deployment of a large number of VMs for your project. The current limit on a per subscription basis is 20 VMs per region. Quota limits can be raised quickly and easily by filing a support ticket requesting a limit increase. For more details on quota limits:

- [Azure Subscription Service Limits](#)

Partners

Microsoft works closely with partners to ensure the images available are updated and optimized for an Azure runtime. For more information on Azure partners, see the following links:

- Linux on Azure - [Endorsed Distributions](#)
- SUSE - [Azure Marketplace - SUSE Linux Enterprise Server](#)
- Red Hat - [Azure Marketplace - Red Hat Enterprise Linux 7.2](#)
- Canonical - [Azure Marketplace - Ubuntu Server 16.04 LTS](#)
- Debian - [Azure Marketplace - Debian 8 "Jessie"](#)
- FreeBSD - [Azure Marketplace - FreeBSD 10.4](#)
- CoreOS - [Azure Marketplace - CoreOS \(Stable\)](#)
- RancherOS - [Azure Marketplace - RancherOS](#)
- Bitnami - [Bitnami Library for Azure](#)
- Mesosphere - [Azure Marketplace - Mesosphere DC/OS on Azure](#)

- Docker - [Azure Marketplace](#) - Azure Container Service with Docker Swarm
- Jenkins - [Azure Marketplace](#) - CloudBees Jenkins Platform

Getting started with Linux on Azure

To begin using Azure, you need an Azure account, the Azure CLI installed, and a pair of SSH public and private keys.

Sign up for an account

The first step in using the Azure Cloud is to sign up for an Azure account. Go to the [Azure Account Signup](#) page to get started.

Install the CLI

With your new Azure account, you can get started immediately using the Azure portal, which is a web-based admin panel. To manage the Azure Cloud via the command line, you install the `azure-cli`. Install the [Azure CLI](#) on your Mac or Linux workstation.

Create an SSH key pair

Now you have an Azure account, the Azure web portal, and the Azure CLI. The next step is to create an SSH key pair that is used to SSH into Linux without using a password. [Create SSH keys on Linux and Mac](#) to enable password-less logins and better security.

Create a VM using the CLI

Creating a Linux VM using the CLI is a quick way to deploy a VM without leaving the terminal you are working in. Everything you can specify on the web portal is available via a command-line flag or switch.

- [Create a Linux VM using the CLI](#)

Create a VM in the portal

Creating a Linux VM in the Azure web portal is a way to easily point and click through the various options to get to a deployment. Instead of using command-line flags or switches, you are able to view a nice web layout of various options and settings. Everything available via the command-line interface is also available in the portal.

- [Create a Linux VM using the Portal](#)

Log in using SSH without a password

The VM is now running on Azure and you are ready to log in. Using passwords to log in via SSH is insecure and time consuming. Using SSH keys is the most secure way and also the quickest way to log in. When you create your Linux VM via the portal or the CLI, you have two authentication choices. If you choose a password for SSH, Azure configures the VM to allow logins via passwords. If you chose to use an SSH public key, Azure configures the VM to only allow logins via SSH keys and disables password logins. To secure your Linux VM by only allowing SSH key logins, use the SSH public key option during the VM creation in the portal or CLI.

Related Azure components

Storage

- [Introduction to Microsoft Azure Storage](#)
- [Add a disk to a Linux VM using the azure-cli](#)
- [How to attach a data disk to a Linux VM in the Azure portal](#)

Networking

- [Virtual Network Overview](#)

- [IP addresses in Azure](#)
- [Opening ports to a Linux VM in Azure](#)
- [Create a Fully Qualified Domain Name in the Azure portal](#)

Containers

- [Virtual Machines and Containers in Azure](#)
- [Azure Container Service introduction](#)
- [Deploy an Azure Container Service cluster](#)

Next steps

You now have an overview of Linux on Azure. The next step is to dive in and create a few VMs!

- [Explore the growing list of sample scripts for common tasks via AzureCLI](#)

Quickstart: Create a Linux virtual machine with the Azure CLI

2/4/2019 • 3 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 deploy a Linux virtual machine (VM) in Azure. In this tutorial, we will be installing Ubuntu 16.04 LTS. To show the VM in action, you'll connect to it using SSH and install the NGINX web server.

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

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, this quickstart requires Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a resource group

Create a resource group with the [az group create](#) command. An Azure resource group is a logical container into which Azure resources are deployed and managed. The following example creates a resource group named *myResourceGroup* in the *eastus* location:

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

Create virtual machine

Create a VM with the [az vm create](#) command.

The following example creates a VM named *myVM* and adds a user account named *azureuser*. The `-generate-ssh-keys` parameter is used to automatically generate an SSH key, and put it in the default key location (`~/.ssh`). To use a specific set of keys instead, use the `--ssh-key-value` option.

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

It takes a few minutes to create the VM and supporting resources. The following example output shows the VM create operation was successful.

```
{  
    "fqdns": "",  
    "id":  
        "/subscriptions/<guid>/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",  
    "location": "eastus",  
    "macAddress": "00-0D-3A-23-9A-49",  
    "powerState": "VM running",  
    "privateIpAddress": "10.0.0.4",  
    "publicIpAddress": "40.68.254.142",  
    "resourceGroup": "myResourceGroup"  
}
```

Note your own `publicIpAddress` in the output from your VM. This address is used to access the VM in the next steps.

Open port 80 for web traffic

By default, only SSH connections are opened when you create a Linux VM in Azure. Use [az vm open-port](#) to open TCP port 80 for use with the NGINX web server:

```
az vm open-port --port 80 --resource-group myResourceGroup --name myVM
```

Connect to virtual machine

SSH to your VM as normal. Replace **publicIpAddress** with the public IP address of your VM as noted in the previous output from your VM:

```
ssh azureuser@publicIpAddress
```

Install web server

To see your VM in action, install the NGINX web server. Update your package sources and then install the latest NGINX package.

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

When done, type `exit` to leave the SSH session.

View the web server in action

Use a web browser of your choice to view the default NGINX welcome page. Use the public IP address of your VM as the web address. The following example shows the default NGINX web site:



Clean up resources

When no longer needed, you can use the `az group delete` command to remove the resource group, VM, and all related resources.

```
az group delete --name myResourceGroup
```

Next steps

In this quickstart, you deployed a simple virtual machine, open a network port for web traffic, and installed a basic web server. To learn more about Azure virtual machines, continue to the tutorial for Linux VMs.

[Azure Linux virtual machine tutorials](#)

Quickstart: Create a Linux virtual machine in the Azure portal

10/31/2018 • 4 minutes to read • [Edit Online](#)

Azure virtual machines (VMs) can be created through the Azure portal. The Azure portal is a browser-based user interface to create VMs and their associated resources. This quickstart shows you how to use the Azure portal to deploy a Linux virtual machine (VM) running Ubuntu 16.04 LTS. To see your VM in action, you also SSH to the VM and install the NGINX web server.

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

Create SSH key pair

You need an SSH key pair to complete this quickstart. If you already have an SSH key pair, you can skip this step.

Open a bash shell and use [ssh-keygen](#) to create an SSH key pair. If you don't have a bash shell on your local computer, you can use the [Azure Cloud Shell](#).

```
ssh-keygen -t rsa -b 2048
```

The above command generates public and private keys with the default name of `id_rsa` in the `~/.ssh` directory. The command returns the full path to the public key. Use the path to the public key to display its contents with `cat`.

```
cat ~/.ssh/id_rsa.pub
```

Save the output of this command. You will need it when configuring your administrator account to log in to your VM.

For more detailed information on how to create SSH key pairs, including the use of PuTTy, see [How to use SSH keys with Windows](#).

If you create your SSH key pair using the Cloud Shell, it will be stored in an Azure File Share that is [automatically mounted by the Cloud Shell](#). Don't delete this file share or storage account until after you have retrieved your keys or you will lose access to the VM.

Sign in to Azure

Sign in to the [Azure portal](#).

Create virtual machine

1. Choose **Create a resource** in the upper left corner of the Azure portal.
2. In the search box above the list of Azure Marketplace resources, search for and select **Ubuntu Server 16.04 LTS** by Canonical, then choose **Create**.
3. In the **Basics** tab, under **Project details**, make sure the correct subscription is selected and then choose to **Create new** under **Resource group**. In the pop-up, type *myResourceGroup* for the name of the resource group and then choose **OK**.

Create a virtual machine

[Basics](#) [Disks](#) [Networking](#) [Management](#) [Guest config](#) [Tags](#) [Review + create](#)

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.

Looking for classic VMs? [Create VM from Azure Marketplace](#)

PROJECT DETAILS

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

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

4. Under **Instance details**, type *myVM* for the **Virtual machine name** and choose *East US* for your **Region**. Leave the other defaults.

INSTANCE DETAILS	
* Virtual machine name <small>i</small>	<input type="text" value="myVM"/> <small>✓</small>
* Region <small>i</small>	<input type="text" value="East US"/> <small>✓</small>
Availability options	<input type="text" value="None"/> <small>✓</small>
* Image <small>i</small>	<input type="text" value="Ubuntu Server 16.04 LTS"/> <small>✓</small>
Browse all images and disks	
* Size <small>i</small>	Standard D2s v3 2 vcpus, 8 GB memory Change size

5. Under **Administrator account**, select **SSH public key**, type your user name, then paste your public key into the text box. Remove any leading or trailing white space in your public key.

ADMINISTRATOR ACCOUNT	
Authentication type	<input type="radio"/> Password <input checked="" type="radio"/> SSH public key
* Username <small>i</small>	<input type="text" value="azureuser"/> <small>✓</small>
* SSH public key <small>i</small>	<input type="text"/>
Login with Azure Active Directory (Preview) <small>i</small> <input type="radio"/> On <input checked="" type="radio"/> Off	

6. Under **Inbound port rules > Public inbound ports**, choose **Allow selected ports** and then select **SSH (22)** and **HTTP (80)** from the drop-down.

INBOUND PORT RULES

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

* Public inbound ports None Allow selected ports

*

⚠️ These ports will be exposed to the internet. Use the Advanced controls to limit inbound traffic to known IP addresses. You can also update inbound traffic rules later.

Review + create **Previous** **Next : Disks >**

7. Leave the remaining defaults and then select the **Review + create** button at the bottom of the page.
8. On the **Create a virtual machine** page, you can see the details about the VM you are about to create. When you are ready, select **Create**.

It will take a few minutes for your VM to be deployed. When the deployment is finished, move on to the next section.

Connect to virtual machine

Create an SSH connection with the VM.

1. Select the **Connect** button on the overview page for your VM.

Microsoft Azure Resource groups > myResourceGroup > myVM

myVM Virtual machine

Search (Ctrl+ /)

Connect Start Restart Stop

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Essentials

Resource group (change)
myResourceGroup
Status
Running
Location
West US
Subscription name (change)
Windows Azure MSDN - Visual Studio Ultimate

2. In the **Connect to virtual machine** page, keep the default options to connect by IP address over port 22. In **Login using VM local account** a connection command is shown. Click the button to copy the command. The following example shows what the SSH connection command looks like:

```
ssh azureuser@10.111.12.123
```

3. Using the same bash shell you used to create your SSH key pair (like the [Azure Cloud Shell](#) or your local bash shell) paste the SSH connection command into the shell to create an SSH session.

Install web server

To see your VM in action, install the NGINX web server. From your SSH session, update your package sources and then install the latest NGINX package.

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

When done, type `exit` to leave the SSH session.

View the web server in action

Use a web browser of your choice to view the default NGINX welcome page. Enter the public IP address of the VM as the web address. The public IP address can be found on the VM overview page or as part of the SSH connection string you used earlier.



Clean up resources

When no longer needed, you can delete the resource group, virtual machine, and all related resources. To do so, select the resource group for the virtual machine, select **Delete**, then confirm the name of the resource group to delete.

Next steps

In this quickstart, you deployed a simple virtual machine, created a Network Security Group and rule, and installed a basic web server. To learn more about Azure virtual machines, continue to the tutorial for Linux VMs.

[Azure Linux virtual machine tutorials](#)

Quickstart: Create a Linux virtual machine in Azure with PowerShell

2/8/2019 • 5 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 quickstart shows you how to use the Azure PowerShell module to deploy a Linux virtual machine (VM) in Azure. This quickstart uses the Ubuntu 16.04 LTS marketplace image from Canonical. To see your VM in action, you'll also SSH to the VM and install the NGINX web server.

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

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. Select **Copy** to copy the blocks of code, paste it into the Cloud Shell, and press enter to run it.

Create SSH key pair

You need an SSH key pair to complete this quickstart. If you already have an SSH key pair, you can skip this step.

Open a bash shell and use [ssh-keygen](#) to create an SSH key pair. If you don't have a bash shell on your local computer, you can use the [Azure Cloud Shell](#).

```
ssh-keygen -t rsa -b 2048
```

For more detailed information on how to create SSH key pairs, including the use of PuTTy, see [How to use SSH keys with Windows](#).

If you create your SSH key pair using the Cloud Shell, it will be stored in a container image in a [storage account that is automatically created by Cloud Shell](#). Don't delete the storage account, or the files share within it, until after you have retrieved your keys or you will lose access to the VM.

Create a resource group

Create an Azure resource group with [New-AzResourceGroup](#). A resource group is a logical container into which Azure resources are deployed and managed:

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

Create virtual network resources

Create a virtual network, subnet, and a public IP address. These resources are used to provide network connectivity to the VM and connect it to the internet:

```

# Create a subnet configuration
$subnetConfig = New-AzVirtualNetworkSubnetConfig ` 
    -Name "mySubnet" ` 
    -AddressPrefix 192.168.1.0/24

# Create a virtual network
$vnet = New-AzVirtualNetwork ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Location "EastUS" ` 
    -Name "myVNET" ` 
    -AddressPrefix 192.168.0.0/16 ` 
    -Subnet $subnetConfig

# Create a public IP address and specify a DNS name
$pip = New-AzPublicIpAddress ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Location "EastUS" ` 
    -AllocationMethod Static ` 
    -IdleTimeoutInMinutes 4 ` 
    -Name "mypublicdns$(Get-Random)"

```

Create an Azure Network Security Group and traffic rule. The Network Security Group secures the VM with inbound and outbound rules. In the following example, an inbound rule is created for TCP port 22 that allows SSH connections. To allow incoming web traffic, an inbound rule for TCP port 80 is also created.

```

# Create an inbound network security group rule for port 22
$nsgRuleSSH = New-AzNetworkSecurityRuleConfig ` 
    -Name "myNetworkSecurityGroupRuleSSH" ` 
    -Protocol "Tcp" ` 
    -Direction "Inbound" ` 
    -Priority 1000 ` 
    -SourceAddressPrefix * ` 
    -SourcePortRange * ` 
    -DestinationAddressPrefix * ` 
    -DestinationPortRange 22 ` 
    -Access "Allow"

# Create an inbound network security group rule for port 80
$nsgRuleWeb = New-AzNetworkSecurityRuleConfig ` 
    -Name "myNetworkSecurityGroupRuleWWW" ` 
    -Protocol "Tcp" ` 
    -Direction "Inbound" ` 
    -Priority 1001 ` 
    -SourceAddressPrefix * ` 
    -SourcePortRange * ` 
    -DestinationAddressPrefix * ` 
    -DestinationPortRange 80 ` 
    -Access "Allow"

# Create a network security group
$nsg = New-AzNetworkSecurityGroup ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Location "EastUS" ` 
    -Name "myNetworkSecurityGroup" ` 
    -SecurityRules $nsgRuleSSH,$nsgRuleWeb

```

Create a virtual network interface card (NIC) with [New-AzNetworkInterface](#). The virtual NIC connects the VM to a subnet, Network Security Group, and public IP address.

```
# Create a virtual network card and associate with public IP address and NSG
$nic = New-AzNetworkInterface ` 
    -Name "myNic" ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Location "EastUS" ` 
    -SubnetId $vnet.Subnets[0].Id ` 
    -PublicIpAddressId $pip.Id ` 
    -NetworkSecurityGroupId $nsg.Id
```

Create a virtual machine

To create a VM in PowerShell, you create a configuration that has settings like the image to use, size, and authentication options. Then the configuration is used to build the VM.

Define the SSH credentials, OS information, and VM size. In this example, the SSH key is stored in

`~/.ssh/id_rsa.pub`.

```
# Define a credential object
$securePassword = ConvertTo-SecureString ' ' -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)

# Create a virtual machine configuration
$vmConfig = New-AzVMConfig ` 
    -VMName "myVM" ` 
    -VMSize "Standard_D1" | ` 
Set-AzVMOperatingSystem ` 
    -Linux ` 
    -ComputerName "myVM" ` 
    -Credential $cred ` 
    -DisablePasswordAuthentication | ` 
Set-AzVMSourceImage ` 
    -PublisherName "Canonical" ` 
    -Offer "UbuntuServer" ` 
    -Skus "16.04-LTS" ` 
    -Version "latest" | ` 
Add-AzVMNetworkInterface ` 
    -Id $nic.Id

# Configure the SSH key
$sshPublicKey = cat ~/.ssh/id_rsa.pub
Add-AzVMSShPublicKey ` 
    -VM $vmConfig ` 
    -KeyData $sshPublicKey ` 
    -Path "/home/azureuser/.ssh/authorized_keys"
```

Now, combine the previous configuration definitions to create with [New-AzVM](#):

```
New-AzVM ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Location eastus -VM $vmConfig
```

It will take a few minutes for your VM to be deployed. When the deployment is finished, move on to the next section.

Connect to the VM

Create an SSH connection with the VM using the public IP address. To see the public IP address of the VM, use the [Get-AzPublicIpAddress](#) cmdlet:

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

Using the same bash shell you used to create your SSH key pair (like the [Azure Cloud Shell](#) or your local bash shell) paste the SSH connection command into the shell to create an SSH session.

```
ssh azureuser@10.111.12.123
```

When prompted, the login user name is *azureuser*. If a passphrase is used with your SSH keys, you need to enter that when prompted.

Install NGINX

To see your VM in action, install the NGINX web server. From your SSH session, update your package sources and then install the latest NGINX package.

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

When done, type `exit` to leave the SSH session.

View the web server in action

Use a web browser of your choice to view the default NGINX welcome page. Enter the public IP address of the VM as the web address. The public IP address can be found on the VM overview page or as part of the SSH connection string you used earlier.



Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) cmdlet to remove the resource group, VM, and all related resources:

```
Remove-AzResourceGroup -Name "myResourceGroup"
```

Next steps

In this quickstart, you deployed a simple virtual machine, created a Network Security Group and rule, and installed a basic web server. To learn more about Azure virtual machines, continue to the tutorial for Linux VMs.

Tutorial: Create and Manage Linux VMs with the Azure CLI

6/8/2019 • 9 minutes to read • [Edit Online](#)

Azure virtual machines provide a fully configurable and flexible computing environment. This tutorial covers basic Azure virtual machine deployment items such as selecting a VM size, selecting a VM image, and deploying a VM. You learn how to:

- Create and connect to a VM
- Select and use VM images
- View and use specific VM sizes
- Resize a VM
- View and understand VM state

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create resource group

Create a resource group with the `az group create` command.

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. In this example, a resource group named *myResourceGroupVM* is created in the *eastus* region.

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

The resource group is specified when creating or modifying a VM, which can be seen throughout this tutorial.

Create virtual machine

Create a virtual machine with the [az vm create](#) command.

When you create a virtual machine, several options are available such as operating system image, disk sizing, and administrative credentials. The following example creates a VM named *myVM* that runs Ubuntu Server. A user account named *azureuser* is created on the VM, and SSH keys are generated if they do not exist in the default key location (*~/ssh*):

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

It may take a few minutes to create the VM. Once the VM has been created, the Azure CLI outputs information about the VM. Take note of the `publicIpAddress`, this address can be used to access the virtual machine..

```
{
  "fqdns": "",
  "id": "/subscriptions/d5b9d4b7-6fc1-0000-0000-
000000000000/resourceGroups/myResourceGroupVM/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "52.174.34.95",
  "resourceGroup": "myResourceGroupVM"
}
```

Connect to VM

You can now connect to the VM with SSH in the Azure Cloud Shell or from your local computer. Replace the example IP address with the `publicIpAddress` noted in the previous step.

```
ssh azureuser@52.174.34.95
```

Once logged in to the VM, you can install and configure applications. When you are finished, you close the SSH session as normal:

```
exit
```

Understand VM images

The Azure marketplace includes many images that can be used to create VMs. In the previous steps, a virtual machine was created using an Ubuntu image. In this step, the Azure CLI is used to search the marketplace for a CentOS image, which is then used to deploy a second virtual machine.

To see a list of the most commonly used images, use the `az vm image list` command.

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

The command output returns the most popular VM images on Azure.

Offer UrnAlias	Publisher Version	Sku	Urn
WindowsServer Datacenter:latest	MicrosoftWindowsServer Win2016Datacenter	2016-Datacenter latest	MicrosoftWindowsServer:WindowsServer:2016-Datacenter:latest
WindowsServer Datacenter:latest	MicrosoftWindowsServer Win2012R2Datacenter	2012-R2-Datacenter latest	MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:latest
WindowsServer SP1:latest	MicrosoftWindowsServer Win2008R2SP1	2008-R2-SP1 latest	MicrosoftWindowsServer:WindowsServer:2008-R2-SP1:latest
WindowsServer Datacenter:latest	MicrosoftWindowsServer Win2012Datacenter	2012-Datacenter latest	MicrosoftWindowsServer:WindowsServer:2012-Datacenter:latest
UbuntuServer Canonical		16.04-LTS	Canonical:UbuntuServer:16.04-LTS:latest
UbuntuLTS	latest		
CentOS	OpenLogic	7.3	OpenLogic:CentOS:7.3:latest
CentOS	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		
Debian	credativ	8	credativ:Debian:8:latest
Debian	latest		
CoreOS	CoreOS	Stable	CoreOS:CoreOS:Stable:latest
CoreOS	latest		

A full list can be seen by adding the `--all` argument. The image list can also be filtered by `--publisher` or `--offer`. In this example, the list is filtered for all images with an offer that matches *CentOS*.

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

Partial output:

Offer	Publisher	Sku	Urn	Version
CentOS	OpenLogic	6.5	OpenLogic:CentOS:6.5:6.5.201501	6.5.201501
CentOS	OpenLogic	6.5	OpenLogic:CentOS:6.5:6.5.201503	6.5.201503
CentOS	OpenLogic	6.5	OpenLogic:CentOS:6.5:6.5.201506	6.5.201506
CentOS	OpenLogic	6.5	OpenLogic:CentOS:6.5:6.5.20150904	6.5.20150904
CentOS	OpenLogic	6.5	OpenLogic:CentOS:6.5:6.5.20160309	6.5.20160309
CentOS	OpenLogic	6.5	OpenLogic:CentOS:6.5:6.5.20170207	6.5.20170207

To deploy a VM using a specific image, take note of the value in the *Urn* column, which consists of the publisher, offer, SKU, and optionally a version number to [identify](#) the image. When specifying the image, the image version number can be replaced with "latest", which selects the latest version of the distribution. In this example, the `--image` argument is used to specify the latest version of a CentOS 6.5 image.

```
az vm create --resource-group myResourceGroupVM --name myVM2 --image OpenLogic:CentOS:6.5:latest --generate-ssh-keys
```

Understand VM sizes

A virtual machine size determines the amount of compute resources such as CPU, GPU, and memory that are made available to the virtual machine. Virtual machines need to be sized appropriately for the expected work load. If workload increases, an existing virtual machine can be resized.

VM Sizes

The following table categorizes sizes into use cases.

TYPE	COMMON SIZES	DESCRIPTION
General purpose	B, Dsv3, Dv3, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory. Ideal for dev / test and small to medium applications and data solutions.
Compute optimized	Fsv2	High CPU-to-memory. Good for medium traffic applications, network appliances, and batch processes.
Memory optimized	Esv3, Ev3, M, DSv2, Dv2	High memory-to-core. Great for relational databases, medium to large caches, and in-memory analytics.
Storage optimized	Lsv2, Ls	High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases.
GPU	NV, NVv2, NC, NCv2, NCv3, ND	Specialized VMs targeted for heavy graphic rendering and video editing.
High performance	H	Our most powerful CPU VMs with optional high-throughput network interfaces (RDMA).

Find available VM sizes

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

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

Partial output:

ResourceDiskSizeInMb	MaxDataDiskCount	MemoryInMb	Name	NumberOfCores	OsDiskSizeInMb
7168	2	3584	Standard_DS1	1	1047552
14336	4	7168	Standard_DS2	2	1047552
28672	8	14336	Standard_DS3	4	1047552
57344	16	28672	Standard_DS4	8	1047552
114688	4	14336	Standard_DS11	2	1047552
229376	8	28672	Standard_DS12	4	1047552
57344	16	57344	Standard_DS13	8	1047552
138240	32	114688	Standard_DS14	16	1047552
291840	1	768	Standard_A0	1	1047552
71680	2	1792	Standard_A1	1	1047552
138240	4	3584	Standard_A2	2	1047552
291840	8	7168	Standard_A3	4	1047552
619520	4	14336	Standard_A5	2	1047552
619520	16	14336	Standard_A4	8	1047552
291840	8	28672	Standard_A6	4	1047552
619520	16	57344	Standard_A7	8	1047552

Create VM with specific size

In the previous VM creation example, a size was not provided, which results in a default size. A VM size can be selected at creation time using `az vm create` and the `--size` argument.

```
az vm create \
  --resource-group myResourceGroupVM \
  --name myVM3 \
  --image UbuntuLTS \
  --size Standard_F4s \
  --generate-ssh-keys
```

Resize a VM

After a VM has been deployed, it can be resized to increase or decrease resource allocation. You can view the current size of a VM with `az vm show`:

```
az vm show --resource-group myResourceGroupVM --name myVM --query hardwareProfile.vmSize
```

Before resizing a VM, check if the desired size is available on the current Azure cluster. The `az vm list-vm-resize-options` command returns the list of sizes.

```
az vm list-vm-resize-options --resource-group myResourceGroupVM --name myVM --query [].name
```

If the desired size is available, the VM can be resized from a powered-on state, however it is rebooted during the operation. Use the [az vm resize](#) command to perform the resize.

```
az vm resize --resource-group myResourceGroupVM --name myVM --size Standard_DS4_v2
```

If the desired size is not on the current cluster, the VM needs to be deallocated before the resize operation can occur. Use the [az vm deallocate](#) command to stop and deallocate the VM. Note, when the VM is powered back on, any data on the temp disk may be removed. The public IP address also changes unless a static IP address is being used.

```
az vm deallocate --resource-group myResourceGroupVM --name myVM
```

Once deallocated, the resize can occur.

```
az vm resize --resource-group myResourceGroupVM --name myVM --size Standard_GS1
```

After the resize, the VM can be started.

```
az vm start --resource-group myResourceGroupVM --name myVM
```

VM power states

An Azure VM can have one of many power states. This state represents the current state of the VM from the standpoint of the hypervisor.

Power states

POWER STATE	DESCRIPTION
Starting	Indicates the virtual machine is being started.
Running	Indicates that the virtual machine is running.
Stopping	Indicates that the virtual machine is being stopped.
Stopped	Indicates that the virtual machine is stopped. Virtual machines in the stopped state still incur compute charges.
Deallocating	Indicates that the virtual machine is being deallocated.
Deallocated	Indicates that the virtual machine is removed from the hypervisor but still available in the control plane. Virtual machines in the Deallocated state do not incur compute charges.
-	Indicates that the power state of the virtual machine is unknown.

Find the power state

To retrieve the state of a particular VM, use the [az vm get-instance-view](#) command. Be sure to specify a valid name for a virtual machine and resource group.

```
az vm get-instance-view \
--name myVM \
--resource-group myResourceGroupVM \
--query instanceView.statuses[1] --output table
```

Output:

Code	DisplayStatus	Level
PowerState/running	VM running	Info

Management tasks

During the life-cycle of a virtual machine, you may want to run management tasks such as starting, stopping, or deleting a virtual machine. Additionally, you may want to create scripts to automate repetitive or complex tasks. Using the Azure CLI, many common management tasks can be run from the command line or in scripts.

Get IP address

This command returns the private and public IP addresses of a virtual machine.

```
az vm list-ip-addresses --resource-group myResourceGroupVM --name myVM --output table
```

Stop virtual machine

```
az vm stop --resource-group myResourceGroupVM --name myVM
```

Start virtual machine

```
az vm start --resource-group myResourceGroupVM --name myVM
```

Delete resource group

Deleting a resource group also deletes all resources contained within, such as the VM, virtual network, and disk. 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 myResourceGroupVM --no-wait --yes
```

Next steps

In this tutorial, you learned about basic VM creation and management such as how to:

- Create and connect to a VM
- Select and use VM images
- View and use specific VM sizes
- Resize a VM
- View and understand VM state

Advance to the next tutorial to learn about VM disks.

[Create and Manage VM disks](#)

Tutorial - Manage Azure disks with the Azure CLI

6/13/2019 • 7 minutes to read • [Edit Online](#)

Azure virtual machines (VMs) use disks to store the operating system, applications, and data. When you create a VM, it is important to choose a disk size and configuration appropriate to the expected workload. This tutorial shows you how to deploy and manage VM disks. You learn about:

- OS disks and temporary disks
- Data disks
- Standard and Premium disks
- Disk performance
- Attaching and preparing data disks
- Resizing disks
- Disk snapshots

Default Azure disks

When an Azure virtual machine is created, two disks are automatically attached to the virtual machine.

Operating system disk - Operating system disks can be sized up to 2 TB, and hosts the VMs 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** be used for applications or data. For applications and data, use data disks, which are detailed later in this tutorial.

Temporary disk - Temporary disks use a solid-state drive that is located on the same Azure host as the VM. Temp disks are highly performant and may be used for operations such as temporary data processing. However, if the VM 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 size. Temporary disks are labeled `/dev/sdb` and have a mountpoint of `/mnt`.

Azure data disks

To install applications and store data, additional data disks can be added. Data disks should be used in any situation where durable and responsive data storage is desired. The size of the virtual machine determines how many data disks can be attached to a VM. For each VM vCPU, four data disks can be attached.

VM disk types

Azure provides two types of disks, standard and Premium.

Standard disk

Standard Storage is backed by HDDs, and delivers cost-effective storage while still being performant. 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. Perfect for VMs running production workload. 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 SSD Sizes	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
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 120	Up to 240	Up to 500	Up to 1,100	Up to 2,300	Up to 5,000	Up to 7,500	Up to 7,500	Up to 16,000	Up to 18,000	Up to 20,000
Throughput per disk	Up to 25 MiB/s	Up to 50 MiB/s	Up to 100 MiB/s	Up to 125 MiB/s	Up to 150 MiB/s	Up to 200 MiB/s	Up to 250 MiB/s	Up to 250 MiB/s	Up to 500 MiB/s	Up to 750 MiB/s	Up to 900 MiB/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](#).

Launch Azure Cloud Shell

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 Cloud Shell, 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.

Create and attach disks

Data disks can be created and attached at VM creation time or to an existing VM.

Attach disk at VM creation

Create a resource group with the [az group create](#) command.

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

Create a VM using the [az vm create](#) command. The following example creates a VM named *myVM*, adds a user account named *azureuser*, and generates SSH keys if they do not exist. The `--data-disk-sizes-gb` argument is used to specify that an additional disk should be created and attached to the virtual machine. To create and attach more than one disk, use a space-delimited list of disk size values. In the following example, a VM is created with two data disks, both 128 GB. Because the disk sizes are 128 GB, these disks are both configured as P10s, which provide maximum 500 IOPS per disk.

```
az vm create \
--resource-group myResourceGroupDisk \
--name myVM \
--image UbuntuLTS \
--size Standard_DS2_v2 \
--generate-ssh-keys \
--data-disk-sizes-gb 128 128
```

Attach disk to existing VM

To create and attach a new disk to an existing virtual machine, use the `az vm disk attach` command. The following example creates a premium disk, 128 gigabytes in size, and attaches it to the VM created in the last step.

```
az vm disk attach \
    --resource-group myResourceGroupDisk \
    --vm-name myVM \
    --name myDataDisk \
    --size-gb 128 \
    --sku Premium_LRS \
    --new
```

Prepare data disks

Once a disk has been attached to the virtual machine, the operating system needs to be configured to use the disk. The following example shows how to manually configure a disk. This process can also be automated using cloud-init, which is covered in a [later tutorial](#).

Create an SSH connection with the virtual machine. Replace the example IP address with the public IP of the virtual machine.

```
ssh 10.101.10.10
```

Partition the disk with `fdisk`.

```
(echo n; echo p; echo 1; echo ; echo ; echo w) | sudo fdisk /dev/sdc
```

Write a file system to the partition by using the `mkfs` command.

```
sudo mkfs -t ext4 /dev/sdc1
```

Mount the new disk so that it is accessible in the operating system.

```
sudo mkdir /datadrive && sudo mount /dev/sdc1 /datadrive
```

The disk can now be accessed through the `datadrive` mountpoint, which can be verified by running the `df -h` command.

```
df -h
```

The output shows the new drive mounted on `/datadrive`.

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda1	30G	1.4G	28G	5%	/
/dev/sdb1	6.8G	16M	6.4G	1%	/mnt
/dev/sdc1	50G	52M	47G	1%	/datadrive

To ensure that the drive is remounted after a reboot, it must be added to the `/etc/fstab` file. To do so, get the UUID of the disk with the `blkid` utility.

```
sudo -i blkid
```

The output displays the UUID of the drive, `/dev/sdc1` in this case.

```
/dev/sdc1: UUID="33333333-3b3b-3c3c-3d3d-3e3e3e3e3e" TYPE="ext4"
```

Add a line similar to the following to the `/etc/fstab` file.

```
UUID=33333333-3b3b-3c3c-3d3d-3e3e3e3e3e /datadrive ext4 defaults,nofail 1 2
```

Now that the disk has been configured, close the SSH session.

```
exit
```

Take a disk snapshot

When you take a disk snapshot, Azure creates a read only, point-in-time copy of the disk. Azure VM snapshots are useful to quickly save the state of a VM before you make configuration changes. In the event of an issue or error, VM can be restored using a snapshot. When a VM has more than one disk, a snapshot is taken of each disk independently of the others. To take application consistent backups, consider stopping the VM before you take disk snapshots. Alternatively, use the [Azure Backup service](#), which enables you to perform automated backups while the VM is running.

Create snapshot

Before you create a virtual machine disk snapshot, the ID or name of the disk is needed. Use the [az vm show](#) command to return the disk ID. In this example, the disk ID is stored in a variable so that it can be used in a later step.

```
osdiskid=$(az vm show \
    -g myResourceGroupDisk \
    -n myVM \
    --query "storageProfile.osDisk.managedDisk.id" \
    -o tsv)
```

Now that you have the ID of the virtual machine disk, the following command creates a snapshot of the disk.

```
az snapshot create \
    --resource-group myResourceGroupDisk \
    --source "$osdiskid" \
    --name osDisk-backup
```

Create disk from snapshot

This snapshot can then be converted into a disk, which can be used to recreate the virtual machine.

```
az disk create \
    --resource-group myResourceGroupDisk \
    --name mySnapshotDisk \
    --source osDisk-backup
```

Restore virtual machine from snapshot

To demonstrate virtual machine recovery, delete the existing virtual machine.

```
az vm delete \
--resource-group myResourceGroupDisk \
--name myVM
```

Create a new virtual machine from the snapshot disk.

```
az vm create \
--resource-group myResourceGroupDisk \
--name myVM \
--attach-os-disk mySnapshotDisk \
--os-type linux
```

Reattach data disk

All data disks need to be reattached to the virtual machine.

First find the data disk name using the [az disk list](#) command. This example places the name of the disk in a variable named *datadisk*, which is used in the next step.

```
datadisk=$(az disk list \
-g myResourceGroupDisk \
--query "[?contains(name,'myVM')].[id]" \
-o tsv)
```

Use the [az vm disk attach](#) command to attach the disk.

```
az vm disk attach \
-g myResourceGroupDisk \
--vm-name myVM \
--name $datadisk
```

Next steps

In this tutorial, you learned about VM disks topics such as:

- OS disks and temporary disks
- Data disks
- Standard and Premium disks
- Disk performance
- Attaching and preparing data disks
- Resizing disks
- Disk snapshots

Advance to the next tutorial to learn about automating VM configuration.

[Automate VM configuration](#)

Tutorial - How to use cloud-init to customize a Linux virtual machine in Azure on first boot

6/13/2019 • 9 minutes to read • [Edit Online](#)

In a previous tutorial, you learned how to SSH to a virtual machine (VM) and manually install NGINX. To create VMs in a quick and consistent manner, some form of automation is typically desired. A common approach to customize a VM on first boot is to use [cloud-init](#). In this tutorial you learn how to:

- Create a cloud-init config file
- Create a VM that uses a cloud-init file
- View a running Node.js app after the VM is created
- Use Key Vault to securely store certificates
- Automate secure deployments of NGINX with cloud-init

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Cloud-init overview

[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.

We are working with our partners to get cloud-init included and working in the images that they provide to Azure. The following table outlines the current cloud-init availability on Azure platform images:

ALIAS	PUBLISHER	OFFER	SKU	VERSION
UbuntuLTS	Canonical	UbuntuServer	16.04-LTS	latest
UbuntuLTS	Canonical	UbuntuServer	14.04.5-LTS	latest
CoreOS	CoreOS	CoreOS	Stable	latest
	OpenLogic	CentOS	7-CI	latest
	RedHat	RHEL	7-RAW-CI	latest

Create cloud-init config file

To see cloud-init in action, create a VM that installs NGINX and runs a simple 'Hello World' Node.js app. The following cloud-init configuration installs the required packages, creates a Node.js app, then initialize and starts the app.

In your current shell, create a file named *cloud-init.txt* 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 cloud-init.txt
```

 to create the file and see a list of available editors. Make sure that the whole cloud-init file is copied correctly, especially the first line:

```

#cloud-config
package_upgrade: true
packages:
- nginx
- nodejs
- npm
write_files:
- owner: www-data:www-data
  path: /etc/nginx/sites-available/default
  content: |
    server {
      listen 80;
      location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
      }
    }
- owner: azureuser:azureuser
  path: /home/azureuser/myapp/index.js
  content: |
    var express = require('express')
    var app = express()
    var os = require('os');
    app.get('/', function (req, res) {
      res.send('Hello World from host ' + os.hostname() + '!')
    })
    app.listen(3000, function () {
      console.log('Hello world app listening on port 3000!')
    })
runcmd:
- service nginx restart
- cd "/home/azureuser/myapp"
- npm init
- npm install express -y
- nodejs index.js

```

For more information about cloud-init configuration options, see [cloud-init config examples](#).

Create virtual machine

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

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

Now create a VM with [az vm create](#). Use the `--custom-data` parameter to pass in your cloud-init config file. Provide the full path to the *cloud-init.txt* config if you saved the file outside of your present working directory. The following example creates a VM named *myVM*:

```
az vm create \
--resource-group myResourceGroupAutomate \
--name myVM \
--image UbuntuLTS \
--admin-username azureuser \
--generate-ssh-keys \
--custom-data cloud-init.txt
```

It takes a few minutes for the VM to be created, the packages to install, and the app to start. There are background tasks that continue to run after the Azure CLI returns you to the prompt. It may be another couple of minutes before you can access the app. When the VM has been created, take note of the `publicIpAddress` displayed by the Azure CLI. This address is used to access the Node.js app via a web browser.

To allow web traffic to reach your VM, open port 80 from the Internet with [az vm open-port](#):

```
az vm open-port --port 80 --resource-group myResourceGroupAutomate --name myVM
```

Test web app

Now you can open a web browser and enter `http://<publicIpAddress>` in the address bar. Provide your own public IP address from the VM create process. Your Node.js app is displayed as shown in the following example:



Inject certificates from Key Vault

This optional section shows how you can securely store certificates in Azure Key Vault and inject them during the VM deployment. Rather than using a custom image that includes the certificates baked-in, this process ensures that the most up-to-date certificates are injected to a VM on first boot. During the process, the certificate never leaves the Azure platform or is exposed in a script, command-line history, or template.

Azure Key Vault safeguards cryptographic keys and secrets, such as certificates or passwords. Key Vault helps streamline the key management process and enables you to maintain control of keys that access and encrypt your data. This scenario introduces some Key Vault concepts to create and use a certificate, though is not an exhaustive overview on how to use Key Vault.

The following steps show how you can:

- Create an Azure Key Vault
- Generate or upload a certificate to the Key Vault
- Create a secret from the certificate to inject in to a VM
- Create a VM and inject the certificate

Create an Azure Key Vault

First, create a Key Vault with [az keyvault create](#) and enable it for use when you deploy a VM. Each Key Vault requires a unique name, and should be all lower case. Replace `mykeyvault` in the following example with your own unique Key Vault name:

```
keyvault_name=mykeyvault
az keyvault create \
    --resource-group myResourceGroupAutomate \
    --name $keyvault_name \
    --enabled-for-deployment
```

Generate certificate and store in Key Vault

For production use, you should import a valid certificate signed by trusted provider with [az keyvault certificate import](#). For this tutorial, the following example shows how you can generate a self-signed certificate with [az keyvault certificate create](#) that uses the default certificate policy:

```
az keyvault certificate create \
--vault-name $keyvault_name \
--name mycert \
--policy "$(az keyvault certificate get-default-policy)"
```

Prepare certificate for use with VM

To use the certificate during the VM create process, obtain the ID of your certificate with [az keyvault secret list-versions](#). The VM needs the certificate in a certain format to inject it on boot, so convert the certificate with [az vm secret format](#). The following example assigns the output of these commands to variables for ease of use in the next steps:

```
secret=$(az keyvault secret list-versions \
--vault-name $keyvault_name \
--name mycert \
--query "[?attributes.enabled].id" --output tsv)
vm_secret=$(az vm secret format --secret "$secret")
```

Create cloud-init config to secure NGINX

When you create a VM, certificates and keys are stored in the protected `/var/lib/waagent` directory. To automate adding the certificate to the VM and configuring NGINX, you can use an updated cloud-init config from the previous example.

Create a file named `cloud-init-secured.txt` and paste the following configuration. Again, if you use the Cloud Shell, create the cloud-init config file there and not on your local machine. Use [sensible-editor cloud-init-secured.txt](#) to create the file and see a list of available editors. Make sure that the whole cloud-init file is copied correctly, especially the first line:

```

#cloud-config
package_upgrade: true
packages:
- nginx
- nodejs
- npm
write_files:
- owner: www-data:www-data
  path: /etc/nginx/sites-available/default
  content: |
    server {
      listen 80;
      listen 443 ssl;
      ssl_certificate /etc/nginx/ssl/mycert.cert;
      ssl_certificate_key /etc/nginx/ssl/mycert.prv;
      location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
      }
    }
- owner: azureuser:azureuser
  path: /home/azureuser/myapp/index.js
  content: |
    var express = require('express')
    var app = express()
    var os = require('os');
    app.get('/', function (req, res) {
      res.send('Hello World from host ' + os.hostname() + '!')
    })
    app.listen(3000, function () {
      console.log('Hello world app listening on port 3000!')
    })
runcmd:
- secretsname=$(find /var/lib/waagent/ -name "*.prv" | cut -c -57)
- mkdir /etc/nginx/ssl
- cp $secretsname.crt /etc/nginx/ssl/mycert.cert
- cp $secretsname.prv /etc/nginx/ssl/mycert.prv
- service nginx restart
- cd "/home/azureuser/myapp"
- npm init
- npm install express -y
- nodejs index.js

```

Create secure VM

Now create a VM with `az vm create`. The certificate data is injected from Key Vault with the `--secrets` parameter. As in the previous example, you also pass in the cloud-init config with the `--custom-data` parameter:

```

az vm create \
--resource-group myResourceGroupAutomate \
--name myVMSecured \
--image UbuntuLTS \
--admin-username azureuser \
--generate-ssh-keys \
--custom-data cloud-init-secured.txt \
--secrets "$vm_secret"

```

It takes a few minutes for the VM to be created, the packages to install, and the app to start. There are background tasks that continue to run after the Azure CLI returns you to the prompt. It may be another couple of minutes before you can access the app. When the VM has been created, take note of the `publicIpAddress`

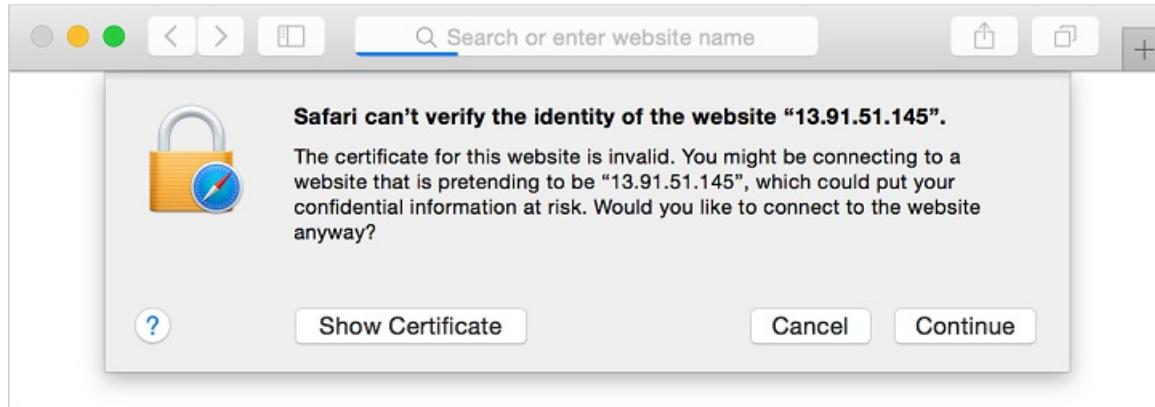
displayed by the Azure CLI. This address is used to access the Node.js app via a web browser.

To allow secure web traffic to reach your VM, open port 443 from the Internet with [az vm open-port](#):

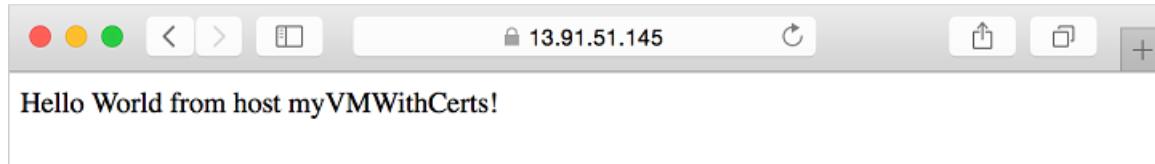
```
az vm open-port \
--resource-group myResourceGroupAutomate \
--name myVMSecured \
--port 443
```

Test secure web app

Now you can open a web browser and enter `https://<publicIpAddress>` in the address bar. Provide your own public IP address as shown in the output of the previous VM create process. Accept the security warning if you used a self-signed certificate:



Your secured NGINX site and Node.js app is then displayed as in the following example:



Next steps

In this tutorial, you configured VMs on first boot with cloud-init. You learned how to:

- Create a cloud-init config file
- Create a VM that uses a cloud-init file
- View a running Node.js app after the VM is created
- Use Key Vault to securely store certificates
- Automate secure deployments of NGINX with cloud-init

Advance to the next tutorial to learn how to create custom VM images.

[Create custom VM images](#)

Tutorial: Create a custom image of an Azure VM with the Azure CLI

2/5/2019 • 4 minutes to read • [Edit Online](#)

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. In this tutorial, you create your own custom image of an Azure virtual machine. You learn how to:

- Deprovision and generalize VMs
- Create a custom image
- Create a VM from a custom image
- List all the images in your subscription
- Delete an image

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

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 tutorial, you must have an existing virtual machine. If needed, this [script sample](#) can create one for you. When working through the tutorial, replace the resource group and VM names where needed.

Create a custom image

To create an image of a virtual machine, you need to prepare the VM by deprovisioning, deallocating, and then marking the source VM as generalized. Once the VM has been prepared, you can create an image.

Deprovision the VM

Deprovisioning generalizes the VM by removing machine-specific information. This generalization makes it possible to deploy many VMs from a single image. During deprovisioning, 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 the VM and manages provisioning and interacting with the Azure Fabric Controller. For more information, see the [Azure Linux Agent user guide](#).

Connect to your VM using SSH and run the command to deprovision the VM. With the `+user` argument, the last provisioned user account and any associated data are also deleted. Replace the example IP address with the public IP address of your VM.

SSH to the VM.

```
ssh azureuser@52.174.34.95
```

Deprovision the VM.

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

Close the SSH session.

```
exit
```

Deallocate and mark the VM as generalized

To create an image, the VM needs to be deallocated. Deallocate the VM using [az vm deallocate](#).

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

Finally, set the state of the VM as generalized with [az vm generalize](#) so the Azure platform knows the VM has been generalized. You can only create an image from a generalized VM.

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

Create the image

Now you can create an image of the VM by using [az image create](#). The following example creates an image named `myImage` from a VM named `myVM`.

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

Create VMs from the image

Now that you have an image, you can create one or more new VMs from the image using [az vm create](#). The following example creates a VM named *myVMfromImage* from the image named *myImage*.

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

Image management

Here are some examples of common image management tasks and how to complete them using the Azure CLI.

List all images by name in a table format.

```
az image list \
--resource-group myResourceGroup
```

Delete an image. This example deletes the image named *myOldImage* from the *myResourceGroup*.

```
az image delete \
--name myOldImage \
--resource-group myResourceGroup
```

Next steps

In this tutorial, you created a custom VM image. You learned how to:

- Deprovision and generalize VMs
- Create a custom image
- Create a VM from a custom image
- List all the images in your subscription
- Delete an image

Advance to the next tutorial to learn about highly available virtual machines.

[Create highly available VMs.](#)

Tutorial: Create and deploy highly available virtual machines with the Azure CLI

2/5/2019 • 5 minutes to read • [Edit Online](#)

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

In this tutorial, you learn how to:

- Create an availability set
- Create a VM in an availability set
- Check available VM sizes

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Availability set overview

An Availability Set is a logical grouping capability that you can use in Azure to ensure that the VM resources you place within it are isolated from each other when they are deployed within an Azure datacenter. Azure ensures that the VMs you place within an Availability Set run across multiple physical servers, compute racks, storage units, and network switches. If a hardware or Azure software failure occurs, only a subset of your VMs are impacted, and your overall application stays up and continues to be available to your customers. Availability Sets are an essential capability when you want to build reliable cloud solutions.

Let's consider a typical VM-based solution where you might have four front-end web servers and use two back-end VMs that host a database. With Azure, you'd want to define two availability sets before you deploy your VMs: one availability set for the "web" tier and one availability set for the "database" tier. When you create a new VM you can then specify the availability set as a parameter to the `az vm create` command, and Azure automatically ensures that the VMs you create within the available set are isolated across multiple physical hardware resources. If the physical hardware that one of your Web Server or Database Server VMs is running on has a problem, you know that the other instances of your Web Server and Database VMs remain running because they are on different hardware.

Use Availability Sets when you want to deploy reliable VM-based solutions within Azure.

Create an availability set

You can create an availability set using `az vm availability-set create`. In this example, the number of update and fault domains is set to 2 for the availability set named `myAvailabilitySet` in the `myResourceGroupAvailability` resource group.

First, create a resource group with `az group create`, then create the availability set:

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

az vm availability-set create \
    --resource-group myResourceGroupAvailability \
    --name myAvailabilitySet \
    --platform-fault-domain-count 2 \
    --platform-update-domain-count 2
```

Availability Sets allow you to isolate resources across fault domains and update domains. A **fault domain** represents an isolated collection of server + network + storage resources. In the preceding example, the availability set is distributed across at least two fault domains when the VMs are deployed. The availability set is also distributed across two **update domains**. Two update domains ensure that when Azure performs software updates, the VM resources are isolated, preventing all the software that runs on the VM from being updated at the same time.

Create VMs inside an availability set

VMs must be created within the availability set to make sure they are correctly distributed across the hardware. An existing VM cannot be added to an availability set after it is created.

When a VM is created with `az vm create`, use the `--availability-set` parameter to specify the name of the availability set.

```
for i in `seq 1 2`; do
    az vm create \
        --resource-group myResourceGroupAvailability \
        --name myVM$i \
        --availability-set myAvailabilitySet \
        --size Standard_DS1_v2 \
        --vnet-name myVnet \
        --subnet mySubnet \
        --image UbuntuLTS \
        --admin-username azureuser \
        --generate-ssh-keys
done
```

There are now two virtual machines within the availability set. Because they are in the same availability set, Azure ensures that the VMs and all their resources (including data disks) are distributed across isolated physical

hardware. This distribution helps ensure much higher availability of the overall VM solution.

The availability set distribution can be viewed in the portal by going to Resource Groups > myResourceGroupAvailability > myAvailabilitySet. The VMs are distributed across the two fault and update domains, as shown in the following example:

NAME	STATUS	FAULT DOMAIN	UPDATE DOMAIN
myVM1	Running	0	0
myVM2	Running	1	1

Check for available VM sizes

Additional VMs can be added to the availability set later, where VM sizes are available on the hardware. Use [az vm availability-set list-sizes](#) to list all the available sizes on the hardware cluster for the availability set:

```
az vm availability-set list-sizes \
--resource-group myResourceGroupAvailability \
--name myAvailabilitySet \
--output table
```

Next steps

In this tutorial, you learned how to:

- Create an availability set
- Create a VM in an availability set
- Check available VM sizes

Advance to the next tutorial to learn about virtual machine scale sets.

[Create a virtual machine scale set](#)

Tutorial: Create a virtual machine scale set and deploy a highly available app on Linux with the Azure CLI

1/24/2019 • 8 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 such as CPU, memory demand, or network traffic. In this tutorial, you deploy a virtual machine scale set in Azure. You learn how to:

- Use cloud-init to create an app to scale
- Create a virtual machine scale set
- Increase or decrease the number of instances in a scale set
- Create autoscale rules
- View connection info for scale set instances
- Use data disks in a scale set

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `Powershell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Scale Set overview

A virtual machine scale set allows you to deploy and manage a set of identical, auto-scaling virtual machines. VMs in a scale set are distributed across logic fault and update domains in one or more *placement groups*. These are groups of similarly configured VMs, similar to [availability sets](#).

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

Scale sets support up to 1,000 VMs when you use an Azure platform image. For workloads with significant installation or VM customization requirements, you may wish to [Create a custom VM image](#). You can create up to 300 VMs in a scale set when using a custom image.

Create an app to scale

For production use, you may wish to [Create a custom VM image](#) that includes your application installed and configured. For this tutorial, let's customize the VMs on first boot to quickly see a scale set in action.

In a previous tutorial, you learned [How to customize a Linux virtual machine on first boot](#) with cloud-init. You can use the same cloud-init configuration file to install NGINX and run a simple 'Hello World' Node.js app.

In your current shell, create a file named *cloud-init.txt* and paste the following configuration. For example, create the file in the Cloud Shell not on your local machine. Enter `sensible-editor cloud-init.txt` to create the file and see a list of available editors. Make sure that the whole cloud-init file is copied correctly, especially the first line:

```
#cloud-config
package_upgrade: true
packages:
- nginx
- nodejs
- npm
write_files:
- owner: www-data:www-data
- path: /etc/nginx/sites-available/default
  content: |
    server {
      listen 80;
      location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
      }
    }
- owner: azureuser:azureuser
- path: /home/azureuser/myapp/index.js
  content: |
    var express = require('express')
    var app = express()
    var os = require('os');
    app.get('/', function (req, res) {
      res.send('Hello World from host ' + os.hostname() + '!')
    })
    app.listen(3000, function () {
      console.log('Hello world app listening on port 3000!')
    })
runcmd:
- service nginx restart
- cd "/home/azureuser/myapp"
- npm init
- npm install express -y
- nodejs index.js
```

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 *myResourceGroupScaleSet* in the *eastus* location:

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

Now create a virtual machine scale set with [az vmss create](#). The following example creates a scale set named *myScaleSet*, uses the cloud-init file to customize the VM, and generates SSH keys if they do not exist:

```
az vmss create \
--resource-group myResourceGroupScaleSet \
--name myScaleSet \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--custom-data cloud-init.txt \
--admin-username azureuser \
--generate-ssh-keys
```

It takes a few minutes to create and configure all the scale set resources and VMs. There are background tasks that continue to run after the Azure CLI returns you to the prompt. It may be another couple of minutes before you can access the app.

Allow web traffic

A load balancer was created automatically as part of the virtual machine scale set. The load balancer distributes traffic across a set of defined VMs using load balancer rules. You can learn more about load balancer concepts and configuration in the next tutorial, [How to load balance virtual machines in Azure](#).

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

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

Test your app

To see your Node.js app on the web, 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 myResourceGroupScaleSet \
--name myScaleSetLBPublicIP \
--query [ipAddress] \
--output tsv
```

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



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

Management tasks

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

View VMs in a scale set

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

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

The output is similar to the following example:

InstanceId	LatestModelApplied	Location	Name	ProvisioningState	ResourceGroup
VmId					
1	True	eastus	myScaleSet_1	Succeeded	MYRESOURCEGROUPSCALESET
c72ddc34-6c41-4a53-b89e-dd24f27b30ab			myScaleSet_3	Succeeded	MYRESOURCEGROUPSCALESET
3	True	eastus			
44266022-65c3-49c5-92dd-88ffa64f95da					

Manually increase or decrease VM instances

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 myResourceGroupScaleSet \
--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 3:

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

Get connection info

To obtain connection information about the VMs in your scale sets, use [az vmss list-instance-connection-info](#). This command outputs the public IP address and port for each VM that allows you to connect with SSH:

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

Use data disks with scale sets

You can create and use data disks with scale sets. In a previous tutorial, you learned how to [Manage Azure disks](#) that outlines the best practices and performance improvements for building apps on data disks rather than the OS disk.

Create scale set with data disks

To create a scale set and attach data disks, add the `--data-disk-sizes-gb` parameter to the [az vmss create](#) command. The following example creates a scale set with 50Gb data disks attached to each instance:

```
az vmss create \
--resource-group myResourceGroupScaleSet \
--name myScaleSetDisks \
--image UbuntuLTS \
--upgrade-policy-mode automatic \
--custom-data cloud-init.txt \
--admin-username azureuser \
--generate-ssh-keys \
--data-disk-sizes-gb 50
```

When instances are removed from a scale set, any attached data disks are also removed.

Add data disks

To add a data disk to instances in your scale set, use [az vmss disk attach](#). The following example adds a 50Gb disk to each instance:

```
az vmss disk attach \
--resource-group myResourceGroupScaleSet \
--name myScaleSet \
--size-gb 50 \
--lun 2
```

Detach data disks

To remove a data disk to instances in your scale set, use [az vmss disk detach](#). The following example removes the data disk at LUN 2 from each instance:

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

Next steps

In this tutorial, you created a virtual machine scale set. You learned how to:

- Use cloud-init to create an app to scale
- Create a virtual machine scale set
- Increase or decrease the number of instances in a scale set
- Create autoscale rules

- View connection info for scale set instances
- Use data disks in a scale set

Advance to the next tutorial to learn more about load balancing concepts for virtual machines.

[Load balance virtual machines](#)

Tutorial: Load balance Linux virtual machines in Azure to create a highly available application with the Azure CLI

2/5/2019 • 10 minutes to read • [Edit Online](#)

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

- Create an Azure load balancer
- Create a load balancer health probe
- Create load balancer traffic rules
- Use cloud-init to create a basic Node.js app
- Create virtual machines and attach to a load balancer
- View a load balancer in action
- Add and remove VMs from a load balancer

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Azure load balancer overview

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

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

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

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

If you followed the previous tutorial to [create a virtual machine scale set](#), a load balancer was created for you. All these components were configured for you as part of the scale set.

Create Azure load balancer

This section details how you can create and configure each component of the load balancer. Before you can create your load balancer, create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroupLoadBalancer* in the *eastus* location:

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

Create a public IP address

To access your app on the Internet, you need a public IP address for the load balancer. Create a public IP address with [az network public-ip create](#). The following example creates a public IP address named *myPublicIP* in the *myResourceGroupLoadBalancer* resource group:

```
az network public-ip create \
--resource-group myResourceGroupLoadBalancer \
--name myPublicIP
```

Create a load balancer

Create a load balancer with [az network lb create](#). The following example creates a load balancer named *myLoadBalancer* and assigns the *myPublicIP* address to the front-end IP configuration:

```
az network lb create \
--resource-group myResourceGroupLoadBalancer \
--name myLoadBalancer \
--frontend-ip-name myFrontEndPool \
--backend-pool-name myBackEndPool \
--public-ip-address myPublicIP
```

Create a health probe

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

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

To create a TCP health probe, you use [az network lb probe create](#). The following example creates a health probe named *myHealthProbe*:

```
az network lb probe create \
--resource-group myResourceGroupLoadBalancer \
--lb-name myLoadBalancer \
--name myHealthProbe \
--protocol tcp \
--port 80
```

Create a load balancer rule

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

Create a load balancer rule with [az network lb rule create](#). The following example creates a rule named *myLoadBalancerRule*, uses the *myHealthProbe* health probe, and balances traffic on port 80:

```
az network lb rule create \
--resource-group myResourceGroupLoadBalancer \
--lb-name myLoadBalancer \
--name myLoadBalancerRule \
--protocol tcp \
--frontend-port 80 \
--backend-port 80 \
--frontend-ip-name myFrontEndPool \
--backend-pool-name myBackEndPool \
--probe-name myHealthProbe
```

Configure virtual network

Before you deploy some VMs and can test your balancer, create the supporting virtual network resources. For more information about virtual networks, see the [Manage Azure Virtual Networks](#) tutorial.

Create network resources

Create a virtual network with [az network vnet create](#). The following example creates a virtual network named *myVnet* with a subnet named *mySubnet*:

```
az network vnet create \
--resource-group myResourceGroupLoadBalancer \
--name myVnet \
--subnet-name mySubnet
```

To add a network security group, you use [az network nsg create](#). The following example creates a network security group named *myNetworkSecurityGroup*:

```
az network nsg create \
--resource-group myResourceGroupLoadBalancer \
--name myNetworkSecurityGroup
```

Create a network security group rule with [az network nsg rule create](#). The following example creates a network security group rule named *myNetworkSecurityGroupRule*:

```
az network nsg rule create \
--resource-group myResourceGroupLoadBalancer \
--nsg-name myNetworkSecurityGroup \
--name myNetworkSecurityGroupRule \
--priority 1001 \
--protocol tcp \
--destination-port-range 80
```

Virtual NICs are created with [az network nic create](#). The following example creates three virtual NICs. (One virtual NIC for each VM you create for your app in the following steps). You can create additional virtual NICs and VMs at any time and add them to the load balancer:

```
for i in `seq 1 3`; do
    az network nic create \
        --resource-group myResourceGroupLoadBalancer \
        --name myNic$i \
        --vnet-name myVnet \
        --subnet mySubnet \
        --network-security-group myNetworkSecurityGroup \
        --lb-name myLoadBalancer \
        --lb-address-pools myBackEndPool
done
```

When all three virtual NICs are created, continue on to the next step

Create virtual machines

Create cloud-init config

In a previous tutorial on [How to customize a Linux virtual machine on first boot](#), you learned how to automate VM customization with cloud-init. You can use the same cloud-init configuration file to install NGINX and run a simple 'Hello World' Node.js app in the next step. To see the load balancer in action, at the end of the tutorial you access this simple app in a web browser.

In your current shell, create a file named *cloud-init.txt* and paste the following configuration. For example, create the file in the Cloud Shell not on your local machine. Enter `sensible-editor cloud-init.txt` to create the file and see a list of available editors. Make sure that the whole cloud-init file is copied correctly, especially the first line:

```

#cloud-config
package_upgrade: true
packages:
- nginx
- nodejs
- npm
write_files:
- owner: www-data:www-data
- path: /etc/nginx/sites-available/default
  content: |
    server {
      listen 80;
      location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
      }
    }
- owner: azureuser:azureuser
- path: /home/azureuser/myapp/index.js
  content: |
    var express = require('express')
    var app = express()
    var os = require('os');
    app.get('/', function (req, res) {
      res.send('Hello World from host ' + os.hostname() + '!')
    })
    app.listen(3000, function () {
      console.log('Hello world app listening on port 3000!')
    })
runcmd:
- service nginx restart
- cd "/home/azureuser/myapp"
- npm init
- npm install express -y
- nodejs index.js

```

Create virtual machines

To improve the high availability of your app, place your VMs in an availability set. For more information about availability sets, see the previous [How to create highly available virtual machines](#) tutorial.

Create an availability set with `az vm availability-set create`. The following example creates an availability set named `myAvailabilitySet`:

```

az vm availability-set create \
--resource-group myResourceGroupLoadBalancer \
--name myAvailabilitySet

```

Now you can create the VMs with `az vm create`. The following example creates three VMs and generates SSH keys if they do not already exist:

```

for i in `seq 1 3`; do
    az vm create \
        --resource-group myResourceGroupLoadBalancer \
        --name myVM$i \
        --availability-set myAvailabilitySet \
        --nics myNic$i \
        --image UbuntuLTS \
        --admin-username azureuser \
        --generate-ssh-keys \
        --custom-data cloud-init.txt \
        --no-wait
done

```

There are background tasks that continue to run after the Azure CLI returns you to the prompt. The `--no-wait` parameter does not wait for all the tasks to complete. It may be another couple of minutes before you can access the app. The load balancer health probe automatically detects when the app is running on each VM. Once the app is running, the load balancer rule starts to distribute traffic.

Test load balancer

Obtain the public IP address of your load balancer with [az network public-ip show](#). The following example obtains the IP address for *myPublicIP* created earlier:

```

az network public-ip show \
    --resource-group myResourceGroupLoadBalancer \
    --name myPublicIP \
    --query [ipAddress] \
    --output tsv

```

You can then enter the public IP address in to a web browser. Remember - it takes a few minutes for the VMs to be ready before the load balancer starts to distribute traffic to them. The app is displayed, including the hostname of the VM that the load balancer distributed traffic to as in the following example:



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

Add and remove VMs

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

Remove a VM from the load balancer

You can remove a VM from the backend address pool with [az network nic ip-config address-pool remove](#). The following example removes the virtual NIC for **myVM2** from *myLoadBalancer*:

```
az network nic ip-config address-pool remove \
--resource-group myResourceGroupLoadBalancer \
--nic-name myNic2 \
--ip-config-name ipConfig1 \
--lb-name myLoadBalancer \
--address-pool myBackEndPool
```

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

To view a list of VMs with virtual NICs connected to the load balancer, use [az network lb address-pool show](#). Query and filter on the ID of the virtual NIC as follows:

```
az network lb address-pool show \
--resource-group myResourceGroupLoadBalancer \
--lb-name myLoadBalancer \
--name myBackEndPool \
--query backendIpConfigurations \
--output tsv | cut -f4
```

The output is similar to the following example, which shows that the virtual NIC for VM 2 is no longer part of the backend address pool:

```
/subscriptions/<guid>/resourceGroups/myResourceGroupLoadBalancer/providers/Microsoft.Network/networkInterfaces
/myNic1/ipConfigurations/ipconfig1
/subscriptions/<guid>/resourceGroups/myResourceGroupLoadBalancer/providers/Microsoft.Network/networkInterfaces
/myNic3/ipConfigurations/ipconfig1
```

Add a VM to the load balancer

After performing VM maintenance, or if you need to expand capacity, you can add a VM to the backend address pool with [az network nic ip-config address-pool add](#). The following example adds the virtual NIC for **myVM2** to *myLoadBalancer*:

```
az network nic ip-config address-pool add \
--resource-group myResourceGroupLoadBalancer \
--nic-name myNic2 \
--ip-config-name ipConfig1 \
--lb-name myLoadBalancer \
--address-pool myBackEndPool
```

To verify that the virtual NIC is connected to the backend address pool, use [az network lb address-pool show](#) again from the preceding step.

Next steps

In this tutorial, you created a load balancer and attached VMs to it. You learned how to:

- Create an Azure load balancer
- Create a load balancer health probe
- Create load balancer traffic rules
- Use cloud-init to create a basic Node.js app
- Create virtual machines and attach to a load balancer
- View a load balancer in action

- Add and remove VMs from a load balancer

Advance to the next tutorial to learn more about Azure virtual network components.

[Manage VMs and virtual networks](#)

Tutorial: Create and manage Azure virtual networks for Linux virtual machines with the Azure CLI

2/5/2019 • 11 minutes to read • [Edit Online](#)

Azure virtual machines use Azure networking for internal and external network communication. This tutorial walks through deploying two virtual machines and configuring Azure networking for these VMs. The examples in this tutorial assume that the VMs are hosting a web application with a database back-end, however an application is not deployed in the tutorial. In this tutorial, you learn how to:

- Create a virtual network and subnet
- Create a public IP address
- Create a front-end VM
- Secure network traffic
- Create a back-end VM

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

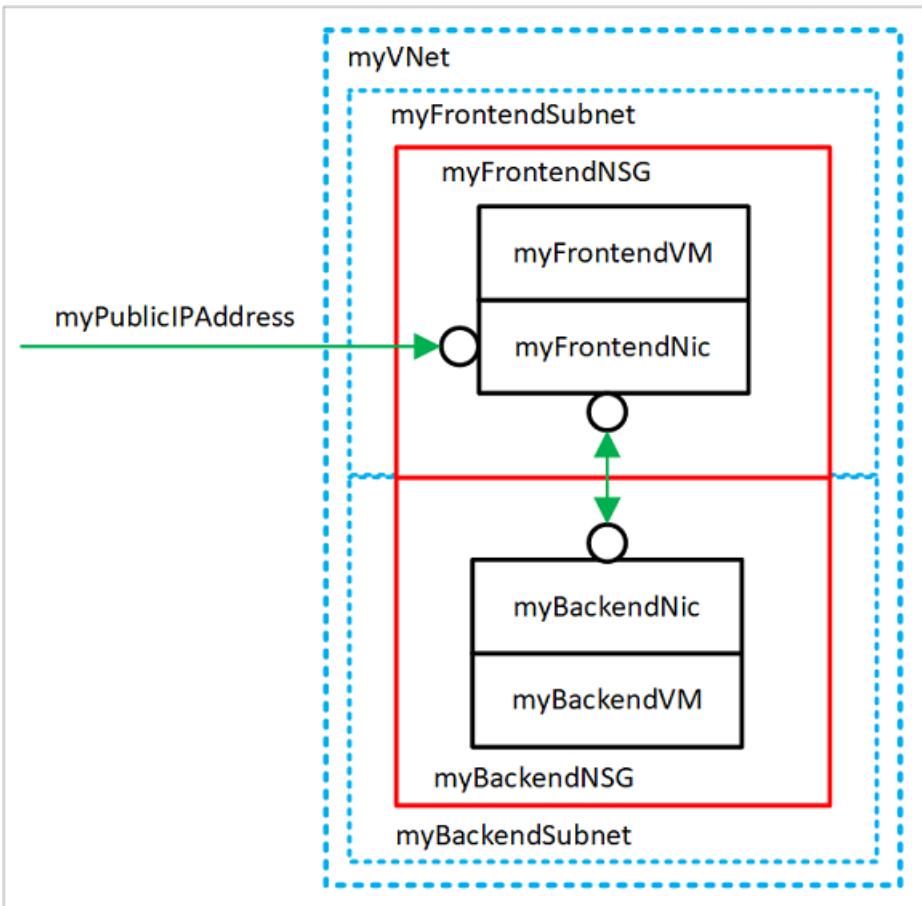
OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

VM networking overview

Azure virtual networks enable secure network connections between virtual machines, the internet, and other Azure services such as Azure SQL database. Virtual networks are broken down into logical segments called subnets. Subnets are used to control network flow, and as a security boundary. When deploying a VM, it generally includes a virtual network interface, which is attached to a subnet.

As you complete the tutorial, the following virtual network resources are created:



- *myVNet* - The virtual network that the VMs use to communicate with each other and the internet.
- *myFrontendSubnet* - The subnet in *myVNet* used by the front-end resources.
- *myPublicIPAddress* - The public IP address used to access *myFrontendVM* from the internet.
- *myFrontendNic* - The network interface used by *myFrontendVM* to communicate with *myBackendVM*.
- *myFrontendVM* - The VM used to communicate between the internet and *myBackendVM*.
- *myBackendNSG* - The network security group that controls communication between the *myFrontendVM* and *myBackendVM*.
- *myBackendSubnet* - The subnet associated with *myBackendNSG* and used by the back-end resources.
- *myBackendNic* - The network interface used by *myBackendVM* to communicate with *myFrontendVM*.
- *myBackendVM* - The VM that uses port 22 and 3306 to communicate with *myFrontendVM*.

Create a virtual network and subnet

For this tutorial, a single virtual network is created with two subnets. A front-end subnet for hosting a web application, and a back-end subnet for hosting a database server.

Before you can create a virtual network, create a resource group with [az group create](#). The following example creates a resource group named *myRGNetwork* in the *eastus* location.

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

Create virtual network

Use the [az network vnet create](#) command to create a virtual network. In this example, the network is named *mvVNet* and is given an address prefix of *10.0.0.0/16*. A subnet is also created with a name of *myFrontendSubnet* and a prefix of *10.0.1.0/24*. Later in this tutorial a front-end VM is connected to this subnet.

```
az network vnet create \
--resource-group myRGNetwork \
--name myVNet \
--address-prefix 10.0.0.0/16 \
--subnet-name myFrontendSubnet \
--subnet-prefix 10.0.1.0/24
```

Create subnet

A new subnet is added to the virtual network using the [az network vnet subnet create](#) command. In this example, the subnet is named *myBackendSubnet* and is given an address prefix of *10.0.2.0/24*. This subnet is used with all back-end services.

```
az network vnet subnet create \
--resource-group myRGNetwork \
--vnet-name myVNet \
--name myBackendSubnet \
--address-prefix 10.0.2.0/24
```

At this point, a network has been created and segmented into two subnets, one for front-end services, and another for back-end services. In the next section, virtual machines are created and connected to these subnets.

Create a public IP address

A public IP address allows Azure resources to be accessible on the internet. The allocation method of the public IP address can be configured as dynamic or static. By default, a public IP address is dynamically allocated. Dynamic IP addresses are released when a VM is deallocated. This behavior causes the IP address to change during any operation that includes a VM deallocation.

The allocation method can be set to static, which ensures that the IP address remains assigned to a VM, even during a deallocated state. When using a statically allocated IP address, the IP address itself cannot be specified. Instead, it is allocated from a pool of available addresses.

```
az network public-ip create --resource-group myRGNetwork --name myPublicIPAddress
```

When creating a VM with the [az vm create](#) command, the default public IP address allocation method is dynamic.

When creating a virtual machine using the [az vm create](#) command, include the

`--public-ip-address-allocation static` argument to assign a static public IP address. This operation is not demonstrated in this tutorial, however in the next section a dynamically allocated IP address is changed to a statically allocated address.

Change allocation method

The IP address allocation method can be changed using the [az network public-ip update](#) command. In this example, the IP address allocation method of the front-end VM is changed to static.

First, deallocate the VM.

```
az vm deallocate --resource-group myRGNetwork --name myFrontendVM
```

Use the [az network public-ip update](#) command to update the allocation method. In this case, the

`--allocation-method` is being set to *static*.

```
az network public-ip update --resource-group myRGNetwork --name myPublicIPAddress --allocation-method static
```

Start the VM.

```
az vm start --resource-group myRGNetwork --name myFrontendVM --no-wait
```

No public IP address

Often, a VM does not need to be accessible over the internet. To create a VM without a public IP address, use the `--public-ip-address ""` argument with an empty set of double quotes. This configuration is demonstrated later in this tutorial.

Create a front-end VM

Use the `az vm create` command to create the VM named *myFrontendVM* using *myPublicIPAddress*.

```
az vm create \
--resource-group myRGNetwork \
--name myFrontendVM \
--vnet-name myVNet \
--subnet myFrontendSubnet \
--nsg myFrontendNSG \
--public-ip-address myPublicIPAddress \
--image UbuntuLTS \
--generate-ssh-keys
```

Secure network traffic

A network security group (NSG) contains a list of security rules that allow or deny network traffic to resources connected to Azure Virtual Networks (VNet). NSGs can be associated to subnets or individual network interfaces. When an NSG is associated with a network interface, it applies only the associated VM. When an NSG is associated to a subnet, the rules apply to all resources connected to the subnet.

Network security group rules

NSG rules define networking ports over which traffic is allowed or denied. The rules can include source and destination IP address ranges so that traffic is controlled between specific systems or subnets. NSG rules also include a priority (between 1—and 4096). Rules are evaluated in the order of priority. A rule with a priority of 100 is evaluated before a rule with priority 200.

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.

The default rules for NSGs are:

- **Virtual network** - Traffic originating and ending in a virtual network is allowed both in inbound and outbound directions.
- **Internet** - Outbound traffic is allowed, but inbound traffic is blocked.
- **Load balancer** - Allow Azure's load balancer to probe the health of your VMs and role instances. If you are not using a load balanced set, you can override this rule.

Create network security groups

A network security group can be created at the same time as a VM using the `az vm create` command. When doing so, the NSG is associated with the VMs network interface and an NSG rule is auto created to allow traffic on port 22 from any source. Earlier in this tutorial, the front-end NSG was auto-created with the front-end VM. An NSG rule was also auto created for port 22.

In some cases, it may be helpful to pre-create an NSG, such as when default SSH rules should not be created, or when the NSG should be attached to a subnet.

Use the [az network nsg create](#) command to create a network security group.

```
az network nsg create --resource-group myRGNetwork --name myBackendNSG
```

Instead of associating the NSG to a network interface, it is associated with a subnet. In this configuration, any VM that is attached to the subnet inherits the NSG rules.

Update the existing subnet named *myBackendSubnet* with the new NSG.

```
az network vnet subnet update \
--resource-group myRGNetwork \
--vnet-name myVNet \
--name myBackendSubnet \
--network-security-group myBackendNSG
```

Secure incoming traffic

When the front-end VM was created, an NSG rule was created to allow incoming traffic on port 22. This rule allows SSH connections to the VM. For this example, traffic should also be allowed on port 80. This configuration allows a web application to be accessed on the VM.

Use the [az network nsg rule create](#) command to create a rule for port 80.

```
az network nsg rule create \
--resource-group myRGNetwork \
--nsg-name myFrontendNSG \
--name http \
--access allow \
--protocol Tcp \
--direction Inbound \
--priority 200 \
--source-address-prefix "*" \
--source-port-range "*" \
--destination-address-prefix "*" \
--destination-port-range 80
```

The front-end VM is only accessible on port 22 and port 80. All other incoming traffic is blocked at the network security group. It may be helpful to visualize the NSG rule configurations. Return the NSG rule configuration with the [az network rule list](#) command.

```
az network nsg rule list --resource-group myRGNetwork --nsg-name myFrontendNSG --output table
```

Secure VM to VM traffic

Network security group rules can also apply between VMs. For this example, the front-end VM needs to communicate with the back-end VM on port 22 and 3306. This configuration allows SSH connections from the front-end VM, and also allow an application on the front-end VM to communicate with a back-end MySQL database. All other traffic should be blocked between the front-end and back-end virtual machines.

Use the [az network nsg rule create](#) command to create a rule for port 22. Notice that the `--source-address-prefix` argument specifies a value of `10.0.1.0/24`. This configuration ensures that only traffic from the front-end subnet is allowed through the NSG.

```
az network nsg rule create \
--resource-group myRGNetwork \
--nsg-name myBackendNSG \
--name SSH \
--access Allow \
--protocol Tcp \
--direction Inbound \
--priority 100 \
--source-address-prefix 10.0.1.0/24 \
--source-port-range "*" \
--destination-address-prefix "*" \
--destination-port-range "22"
```

Now add a rule for MySQL traffic on port 3306.

```
az network nsg rule create \
--resource-group myRGNetwork \
--nsg-name myBackendNSG \
--name MySQL \
--access Allow \
--protocol Tcp \
--direction Inbound \
--priority 200 \
--source-address-prefix 10.0.1.0/24 \
--source-port-range "*" \
--destination-address-prefix "*" \
--destination-port-range "3306"
```

Finally, because NSGs have a default rule allowing all traffic between VMs in the same VNet, a rule can be created for the back-end NSGs to block all traffic. Notice here that the `--priority` is given a value of *300*, which is lower than both the NSG and MySQL rules. This configuration ensures that SSH and MySQL traffic is still allowed through the NSG.

```
az network nsg rule create \
--resource-group myRGNetwork \
--nsg-name myBackendNSG \
--name denyAll \
--access Deny \
--protocol Tcp \
--direction Inbound \
--priority 300 \
--source-address-prefix "*" \
--source-port-range "*" \
--destination-address-prefix "*" \
--destination-port-range "*"
```

Create back-end VM

Now create a virtual machine, which is attached to the *myBackendSubnet*. Notice that the `--nsg` argument has a value of empty double quotes. An NSG does not need to be created with the VM. The VM is attached to the back-end subnet, which is protected with the pre-created back-end NSG. This NSG applies to the VM. Also, notice here that the `--public-ip-address` argument has a value of empty double quotes. This configuration creates a VM without a public IP address.

```
az vm create \
--resource-group myRGNetwork \
--name myBackendVM \
--vnet-name myVNet \
--subnet myBackendSubnet \
--public-ip-address "" \
--nsg "" \
--image UbuntuLTS \
--generate-ssh-keys
```

The back-end VM is only accessible on port 22 and port 3306 from the front-end subnet. All other incoming traffic is blocked at the network security group. It may be helpful to visualize the NSG rule configurations. Return the NSG rule configuration with the [az network rule list](#) command.

```
az network nsg rule list --resource-group myRGNetwork --nsg-name myBackendNSG --output table
```

Next steps

In this tutorial, you created and secured Azure networks as related to virtual machines. You learned how to:

- Create a virtual network and subnet
- Create a public IP address
- Create a front-end VM
- Secure network traffic
- Create back-end VM

Advance to the next tutorial to learn about securing data on virtual machines using Azure backup.

[Back up Linux virtual machines in Azure](#)

Tutorial: Back up and restore files for Linux virtual machines in Azure

4/26/2018 • 5 minutes to read • [Edit Online](#)

You can protect your data by taking backups at regular intervals. Azure Backup creates recovery points that are stored in geo-redundant recovery vaults. When you restore from a recovery point, you can restore the whole VM or specific files. This article explains how to restore a single file to a Linux VM running nginx. If you don't already have a VM to use, you can create one using the [Linux quickstart](#). In this tutorial you learn how to:

- Create a backup of a VM
- Schedule a daily backup
- Restore a file from a backup

Backup overview

When the Azure Backup service initiates a backup, it triggers the backup extension to take a point-in-time snapshot. The Azure Backup service uses the *VMSnapshotLinux* extension in Linux. The extension is installed during the first VM backup if the VM is running. If the VM is not running, the Backup service takes a snapshot of the underlying storage (since no application writes occur while the VM is stopped).

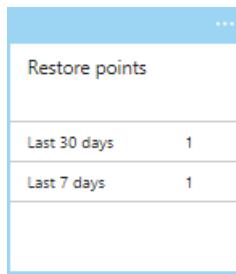
By default, Azure Backup takes a file system consistent backup for Linux VM but it can be configured to take [application consistent backup using pre-script and post-script framework](#). Once the Azure Backup service takes the snapshot, the data is transferred to the vault. To maximize efficiency, the service identifies and transfers only the blocks of data that have changed since the previous backup.

When the data transfer is complete, the snapshot is removed and a recovery point is created.

Create a backup

Create a scheduled daily backup to a Recovery Services Vault:

1. Sign in to the [Azure portal](#).
2. In the menu on the left, select **Virtual machines**.
3. From the list, select a VM to back up.
4. On the VM blade, in the **Settings** section, click **Backup**. The **Enable backup** blade opens.
5. In **Recovery Services vault**, click **Create new** and provide the name for the new vault. A new vault is created in the same Resource Group and location as the virtual machine.
6. Click **Backup policy**. For this example, keep the defaults and click **OK**.
7. On the **Enable backup** blade, click **Enable Backup**. This creates a daily backup based on the default schedule.
8. To create an initial recovery point, on the **Backup** blade click **Backup now**.
9. On the **Backup Now** blade, click the calendar icon, use the calendar control to select the last day this recovery point is retained, and click **Backup**.
10. In the **Backup** blade for your VM, you see the number of recovery points that are complete.



The first backup takes about 20 minutes. Proceed to the next part of this tutorial after your backup is finished.

Restore a file

If you accidentally delete or make changes to a file, you can use File Recovery to recover the file from your backup vault. File Recovery uses a script that runs on the VM, to mount the recovery point as a local drive. These drives remain mounted for 12 hours so that you can copy files from the recovery point and restore them to the VM.

In this example, we show how to recover the default nginx web page /var/www/html/index.nginx-debian.html. The public IP address of our VM in this example is 13.69.75.209. You can find the IP address of your vm using:

```
az vm show --resource-group myResourceGroup --name myVM -d --query [publicIps] --o tsv
```

1. On your local computer, open a browser and type in the public IP address of your VM to see the default nginx web page.



2. SSH into your VM.

```
ssh 13.69.75.209
```

3. Delete /var/www/html/index.nginx-debian.html.

```
sudo rm /var/www/html/index.nginx-debian.html
```

4. On your local computer, refresh the browser by hitting CTRL + F5 to see that default nginx page is gone.



5. On your local computer, sign in to the [Azure portal](#).
6. In the menu on the left, select **Virtual machines**.
7. From the list, select the VM.
8. On the VM blade, in the **Settings** section, click **Backup**. The **Backup** blade opens.
9. In the menu at the top of the blade, select **File Recovery**. The **File Recovery** blade opens.
10. In **Step 1: Select recovery point**, select a recovery point from the drop-down.
11. In **Step 2: Download script to browse and recover files**, click the **Download Executable** button. Save the downloaded file to your local computer.
12. Click **Download script** to download the script file locally.
13. Open a Bash prompt and type the following, replacing *Linux_myVM_05-05-2017.sh* with the correct path and filename for the script that you downloaded, *azureuser* with the username for the VM and *13.69.75.209* with the public IP address for your VM.

```
scp Linux_myVM_05-05-2017.sh azureuser@13.69.75.209:
```

14. On your local computer, open an SSH connection to the VM.

```
ssh 13.69.75.209
```

15. On your VM, add execute permissions to the script file.

```
chmod +x Linux_myVM_05-05-2017.sh
```

16. On your VM, run the script to mount the recovery point as a filesystem.

```
./Linux_myVM_05-05-2017.sh
```

17. The output from the script gives you the path for the mount point. The output looks similar to this:

Microsoft Azure VM Backup - File Recovery

Connecting to recovery point using ISCSI service...

Connection succeeded!

Please wait while we attach volumes of the recovery point to this machine...

***** Volumes of the recovery point and their mount paths on this machine *****

Sr.No.	Disk	Volume	MountPath
--------	------	--------	-----------

1)	/dev/sdc	/dev/sdc1	/home/azureuser/myVM-20170505191055/Volume1
----	----------	-----------	---

***** Open File Explorer to browse for files. *****

After recovery, to remove the disks and close the connection to the recovery point, please click 'Unmount Disks' in step 3 of the portal.

Please enter 'q/Q' to exit...

18. On your VM, copy the nginx default web page from the mount point back to where you deleted the file.

```
sudo cp ~/myVM-20170505191055/Volume1/var/www/html/index.nginx-debian.html /var/www/html/
```

19. On your local computer, open the browser tab where you are connected to the IP address of the VM showing the nginx default page. Press CTRL + F5 to refresh the browser page. You should now see that the default page is working again.



20. On your local computer, go back to the browser tab for the Azure portal and in **Step 3: Unmount the disks after recovery** click the **Unmount Disks** button. If you forget to do this step, the connection to the mountpoint is automatically closed after 12 hours. After those 12 hours, you need to download a new script to create a new mountpoint.

Next steps

In this tutorial, you learned how to:

- Create a backup of a VM
- Schedule a daily backup
- Restore a file from a backup

Advance to the next tutorial to learn about monitoring virtual machines.

[Govern virtual machines](#)

Tutorial: Learn about Linux virtual machine governance with Azure CLI

4/25/2019 • 11 minutes to read • [Edit Online](#)

When deploying resources to Azure, you have tremendous flexibility when deciding what types of resources to deploy, where they are located, and how to set them up. However, that flexibility may open more options than you would like to allow in your organization. As you consider deploying resources to Azure, you might be wondering:

- How do I meet legal requirements for data sovereignty in certain countries/regions?
- How do I control costs?
- How do I ensure that someone does not inadvertently change a critical system?
- How do I track resource costs and bill it accurately?

This article addresses those questions. Specifically, you:

- Assign users to roles and assign the roles to a scope so users have permission to perform expected actions but not more actions.
- Apply policies that prescribe conventions for resources in your subscription.
- Lock resources that are critical to your system.
- Tag resources so you can track them by values that make sense to your organization.

This article focuses on the tasks you take to implement governance. For a broader discussion of the concepts, see [Governance in Azure](#).

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `Powershell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

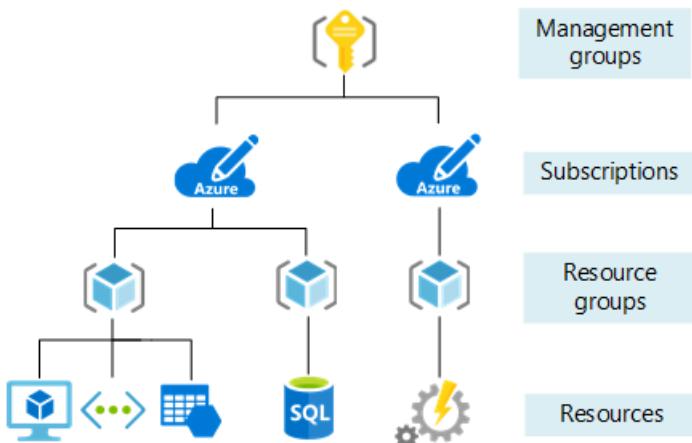
You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use Azure CLI locally, this tutorial requires that you're running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Understand scope

Before creating any items, let's review the concept of scope. Azure provides four levels of management: management groups, subscription, resource group, and resource. [Management groups](#) are in a preview release. 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. When you apply a setting to the subscription, that setting is applied to all resource groups and resources in your subscription. When you apply a setting on the resource group, that setting is applied to the resource group and all its resources. However, another resource group does not have that setting.

Usually, it makes sense to apply critical settings at higher levels and project-specific requirements at lower levels. For example, you might want to make sure all resources for your organization are deployed to certain regions. To accomplish this requirement, apply a policy to the subscription that specifies the allowed locations. As other users in your organization add new resource groups and resources, the allowed locations are automatically enforced.

In this tutorial, you apply all management settings to a resource group so you can easily remove those settings when done.

Let's create that resource group.

```
az group create --name myResourceGroup --location "East US"
```

Currently, the resource group is empty.

Role-based access control

You want to make sure users in your organization have the right level of access to these resources. You don't want to grant unlimited access to users, but you also need to make sure they can do their work. [Role-based access control](#) enables you to manage which users have permission to complete specific actions at a scope.

To create and remove role assignments, users must have `Microsoft.Authorization/roleAssignments/*` access. This access is granted through the Owner or User Access Administrator roles.

For managing virtual machine solutions, there are three resource-specific roles that provide commonly needed access:

- [Virtual Machine Contributor](#)
- [Network Contributor](#)
- [Storage Account Contributor](#)

Instead of assigning roles to individual users, it's often easier to use an Azure Active Directory group that has users

who need to take similar actions. Then, assign that group to the appropriate role. For this article, either use an existing group for managing the virtual machine, or use the portal to [create an Azure Active Directory group](#).

After creating a new group or finding an existing one, use the [az role assignment create](#) command to assign the new Azure Active Directory group to the Virtual Machine Contributor role for the resource group.

```
adgroupId=$(az ad group show --group <your-group-name> --query objectId --output tsv)

az role assignment create --assignee-object-id $adgroupId --role "Virtual Machine Contributor" --resource-group myResourceGroup
```

If you receive an error stating **Principal <guid> does not exist in the directory**, the new group hasn't propagated throughout Azure Active Directory. Try running the command again.

Typically, you repeat the process for *Network Contributor* and *Storage Account Contributor* to make sure users are assigned to manage the deployed resources. In this article, you can skip those steps.

Azure Policy

[Azure Policy](#) helps you make sure all resources in subscription meet corporate standards. Your subscription already has several policy definitions. To see the available policy definitions, use the [az policy definition list](#) command:

```
az policy definition list --query "[].[displayName, policyType, name]" --output table
```

You see the existing policy definitions. The policy type is either **BuiltIn** or **Custom**. Look through the definitions for ones that describe a condition you want assign. In this article, you assign policies that:

- Limit the locations for all resources.
- Limit the SKUs for virtual machines.
- Audit virtual machines that don't use managed disks.

In the following example, you retrieve three policy definitions based on the display name. You use the [az policy assignment create](#) command to assign those definitions to the resource group. For some policies, you provide parameter values to specify the allowed values.

```

# Get policy definitions for allowed locations, allowed SKUs, and auditing VMs that don't use managed disks
locationDefinition=$(az policy definition list --query "[?displayName=='Allowed locations'].name | [0]" --output tsv)
skuDefinition=$(az policy definition list --query "[?displayName=='Allowed virtual machine SKUs'].name | [0]" --output tsv)
auditDefinition=$(az policy definition list --query "[?displayName=='Audit VMs that do not use managed disks'].name | [0]" --output tsv)

# Assign policy for allowed locations
az policy assignment create --name "Set permitted locations" \
--resource-group myResourceGroup \
--policy $locationDefinition \
--params '{
    "listOfAllowedLocations": {
        "value": [
            "eastus",
            "eastus2"
        ]
    }
}'

# Assign policy for allowed SKUs
az policy assignment create --name "Set permitted VM SKUs" \
--resource-group myResourceGroup \
--policy $skuDefinition \
--params '{
    "listOfAllowedSKUs": {
        "value": [
            "Standard_DS1_v2",
            "Standard_E2s_v2"
        ]
    }
}'

# Assign policy for auditing unmanaged disks
az policy assignment create --name "Audit unmanaged disks" \
--resource-group myResourceGroup \
--policy $auditDefinition

```

The preceding example assumes you already know the parameters for a policy. If you need to view the parameters, use:

```
az policy definition show --name $locationDefinition --query parameters
```

Deploy the virtual machine

You have assigned roles and policies, so you're ready to deploy your solution. The default size is Standard_DS1_v2, which is one of your allowed SKUs. The command creates SSH keys if they don't exist in a default location.

```
az vm create --resource-group myResourceGroup --name myVM --image UbuntuLTS --generate-ssh-keys
```

After your deployment finishes, you can apply more management settings to the solution.

Lock resources

[Resource locks](#) prevent users in your organization from accidentally deleting or modifying critical resources. Unlike role-based access control, resource locks apply a restriction across all users and roles. You can set the lock level to *CanNotDelete* or *ReadOnly*.

To create or delete management locks, you must have access to `Microsoft.Authorization/locks/*` actions. Of the built-in roles, only **Owner** and **User Access Administrator** are granted those actions.

To lock the virtual machine and network security group, use the [az lock create](#) command:

```
# Add CanNotDelete lock to the VM
az lock create --name LockVM \
--lock-type CanNotDelete \
--resource-group myResourceGroup \
--resource-name myVM \
--resource-type Microsoft.Compute/virtualMachines

# Add CanNotDelete lock to the network security group
az lock create --name LockNSG \
--lock-type CanNotDelete \
--resource-group myResourceGroup \
--resource-name myVMNSG \
--resource-type Microsoft.Network/networkSecurityGroups
```

To test the locks, try running the following command:

```
az group delete --name myResourceGroup
```

You see an error stating that the delete operation can't be completed because of a lock. The resource group can only be deleted if you specifically remove the locks. That step is shown in [Clean up resources](#).

Tag resources

You apply [tags](#) to your Azure resources to logically organize them by categories. Each tag consists of a name and a value. For example, you can apply the name "Environment" and the value "Production" to all the resources in production.

To add two tags to a resource group, use the [az group update](#) command:

```
az group update -n myResourceGroup --set tags.Environment=Test tags.Dept=IT
```

Let's suppose you want to add a third tag. Run the command again with the new tag. It is appended to the existing tags.

```
az group update -n myResourceGroup --set tags.Project=Documentation
```

Resources don't inherit tags from the resource group. Currently, your resource group has three tags but the resources do not have any tags. To apply all tags from a resource group to its resources, and retain existing tags on resources, use the following script:

```

# Get the tags for the resource group
jsontag=$(az group show -n myResourceGroup --query tags)

# Reformat from JSON to space-delimited and equals sign
t=$(echo $jsontag | tr -d '"{},' | sed 's/: /=g')

# Get the resource IDs for all resources in the resource group
r=$(az resource list -g myResourceGroup --query [].id --output tsv)

# Loop through each resource ID
for resid in $r
do
    # Get the tags for this resource
    jsonrtag=$(az resource show --id $resid --query tags)

    # Reformat from JSON to space-delimited and equals sign
    rt=$(echo $jsonrtag | tr -d '"{},' | sed 's/: /=g')

    # Reapply the updated tags to this resource
    az resource tag --tags $t$rt --id $resid
done

```

Alternatively, you can apply tags from the resource group to the resources without keeping the existing tags:

```

# Get the tags for the resource group
jsontag=$(az group show -n myResourceGroup --query tags)

# Reformat from JSON to space-delimited and equals sign
t=$(echo $jsontag | tr -d '"{},' | sed 's/: /=g')

# Get the resource IDs for all resources in the resource group
r=$(az resource list -g myResourceGroup --query [].id --output tsv)

# Loop through each resource ID
for resid in $r
do
    # Apply tags from resource group to this resource
    az resource tag --tags $t --id $resid
done

```

To combine several values in a single tag, use a JSON string.

```
az group update -n myResourceGroup --set tags.CostCenter='{"Dept":"IT","Environment":"Test"}'
```

To remove all tags on a resource group, use:

```
az group update -n myResourceGroup --remove tags
```

To apply tags to a virtual machine, use the [az resource tag](#) command. Any existing tags on the resource aren't retained.

```
az resource tag -n myVM \
-g myResourceGroup \
--tags Dept=IT Environment=Test Project=Documentation \
--resource-type "Microsoft.Compute/virtualMachines"
```

Find resources by tag

To find resources with a tag name and value, use the [az resource list](#) command:

```
az resource list --tag Environment=Test --query [].name
```

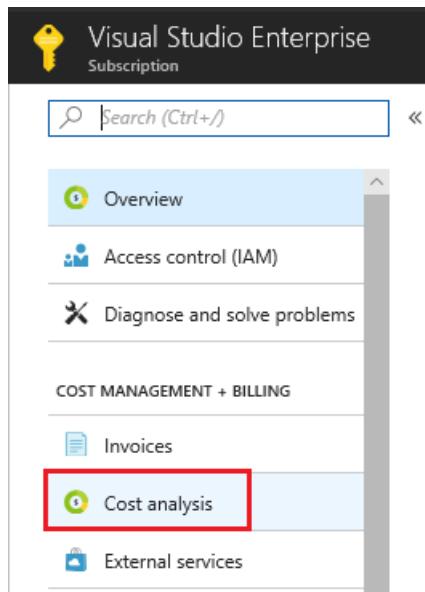
You can use the returned values for management tasks like stopping all virtual machines with a tag value.

```
az vm stop --ids $(az resource list --tag Environment=Test --query "[? type=='Microsoft.Compute/virtualMachines'].id" --output tsv)
```

View costs by tag values

After applying tags to resources, you can view costs for resources with those tags. It takes a while for cost analysis to show the latest usage, so you may not see the costs yet. When the costs are available, you can view costs for resources across resource groups in your subscription. Users must have [subscription level access to billing information](#) to see the costs.

To view costs by tag in the portal, select your subscription and select **Cost Analysis**.



Then, filter by the tag value, and select **Apply**.

A screenshot of the "Costs by service" blade in the Azure portal. It shows the following fields: Subscription (Visual Studio Enterprise), Resource type (3 selected), Resource group (2 selected). Under "Timespan" is "Current period". There are "Apply" and "Download" buttons. A note says: "There is a delay between the time when reported here may be delayed. Amount reaches the billing system. Due to this, costs me recent usage. Taxes are not included." On the left, there's a "Total cost" of "0.24 USD". A "Tag" dropdown is open, showing a list of tags: "Environment: Test" (selected), "Select all", "Project: Documentation", "Dept: IT", "-- No Tags --". A tooltip for "Environment: Test" says: "reaches the billing system. Due to this, costs me recent usage. Taxes are not included."

You can also use the [Azure Billing APIs](#) to programmatically view costs.

Clean up resources

The locked network security group can't be deleted until the lock is removed. To remove the lock, retrieve the IDs

of the locks and provide them to the [az lock delete](#) command:

```
vmlock=$(az lock show --name LockVM \
--resource-group myResourceGroup \
--resource-type Microsoft.Compute/virtualMachines \
--resource-name myVM --output tsv --query id)
nsglock=$(az lock show --name LockNSG \
--resource-group myResourceGroup \
--resource-type Microsoft.Network/networkSecurityGroups \
--resource-name myVMNSG --output tsv --query id)
az lock delete --ids $vmlock $nsglock
```

When no longer needed, you can use the [az group delete](#) command to remove the resource group, VM, and all related resources. Exit the SSH session to your VM, then delete the resources as follows:

```
az group delete --name myResourceGroup
```

Next steps

In this tutorial, you created a custom VM image. You learned how to:

- Assign users to a role
- Apply policies that enforce standards
- Protect critical resources with locks
- Tag resources for billing and management

Advance to the next tutorial to learn about how highly available virtual machines.

[Monitor virtual machines](#)

Tutorial: Monitor and update a Linux virtual machine in Azure

6/6/2019 • 14 minutes to read • [Edit Online](#)

To ensure your virtual machines (VMs) in Azure are running correctly, you can review boot diagnostics, performance metrics and manage package updates. In this tutorial, you learn how to:

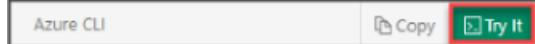
- Enable boot diagnostics on the VM
- View boot diagnostics
- View host metrics
- Enable diagnostics extension on the VM
- View VM metrics
- Create alerts based on diagnostic metrics
- Manage package updates
- Monitor changes and inventory
- Set up advanced monitoring

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create VM

To see diagnostics and metrics in action, you need a VM. First, create a resource group with `az group create`. The following example creates a resource group named `myResourceGroupMonitor` in the `eastus` location.

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

Now create a VM with [az vm create](#). The following example creates a VM named *myVM* and generates SSH keys if they do not already exist in `~/.ssh/`:

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

Enable boot diagnostics

As Linux VMs boot, the boot diagnostic extension 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 Linux VM using the Azure CLI.

Before enabling boot diagnostics, a storage account needs to be created for storing boot logs. Storage accounts must have a globally unique name, be between 3 and 24 characters, and must contain only numbers and lowercase letters. Create a storage account with the [az storage account create](#) command. In this example, a random string is used to create a unique storage account name.

```
storageacct=mydiagdata$RANDOM

az storage account create \
--resource-group myResourceGroupMonitor \
--name $storageacct \
--sku Standard_LRS \
--location eastus
```

When enabling boot diagnostics, the URI to the blob storage container is needed. The following command queries the storage account to return this URI. The URI value is stored in a variable names *bloburi*, which is used in the next step.

```
bloburi=$(az storage account show --resource-group myResourceGroupMonitor --name $storageacct --query
'primaryEndpoints.blob' -o tsv)
```

Now enable boot diagnostics with [az vm boot-diagnostics enable](#). The `--storage` value is the blob URI collected in the previous step.

```
az vm boot-diagnostics enable \
--resource-group myResourceGroupMonitor \
--name myVM \
--storage $bloburi
```

View boot diagnostics

When boot diagnostics are enabled, each time you stop and start the VM, information about the boot process is written to a log file. For this example, first deallocate the VM with the [az vm deallocate](#) command as follows:

```
az vm deallocate --resource-group myResourceGroupMonitor --name myVM
```

Now start the VM with the `az vm start` command as follows:

```
az vm start --resource-group myResourceGroupMonitor --name myVM
```

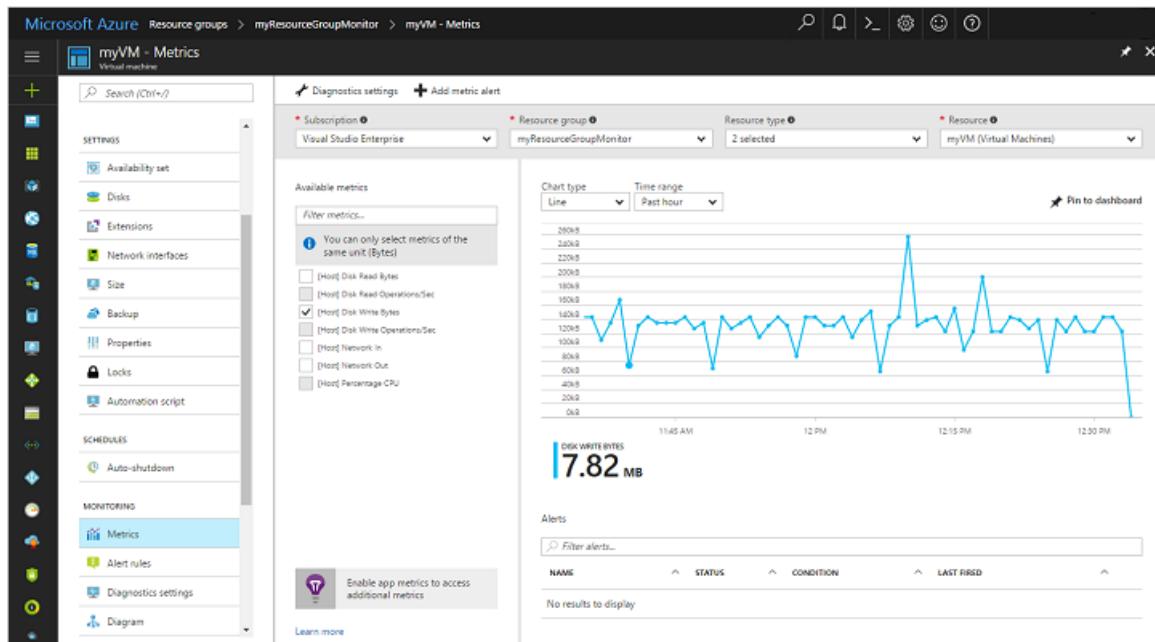
You can get the boot diagnostic data for *myVM* with the `az vm boot-diagnostics get-boot-log` command as follows:

```
az vm boot-diagnostics get-boot-log --resource-group myResourceGroupMonitor --name myVM
```

View host metrics

A Linux VM has a dedicated host in Azure that it interacts with. Metrics are automatically collected for the host and can be viewed in the Azure portal as follows:

1. In the Azure portal, select **Resource Groups**, choose **myResourceGroupMonitor**, and then select **myVM** in the resource list.
2. To see how the host VM is performing, select **Metrics** on the VM window, then choose any of the *[Host]* metrics under **Available metrics**.



Install diagnostics extension

The basic host metrics are available, but to see more granular and VM-specific metrics, you need to install the Azure diagnostics extension on the VM. The Azure diagnostics extension allows additional monitoring and diagnostics data to be retrieved from the VM. You can view these performance metrics and create alerts based on how the VM performs. The diagnostic extension is installed through the Azure portal as follows:

1. In the Azure portal, choose **Resource Groups**, select **myResourceGroupMonitor**, and then select **myVM** in the resource list.
2. Select **Diagnosis settings**. In the *Pick a storage account* drop-down menu, if not already selected, choose the *mydiagdata[1234]* account created in the previous section.
3. Select the **Enable guest-level monitoring** button.

Azure Monitoring collects host-level metrics – like CPU utilization, disk and network usage – for all virtual machines without any additional software. For more insight into this virtual machine, you can collect guest-level metrics, logs, and other diagnostic data using the Azure Diagnostics agent.

To get started now, choose a storage account below where diagnostic data will be sent and then click the button labeled 'Enable guest-level diagnostics'.

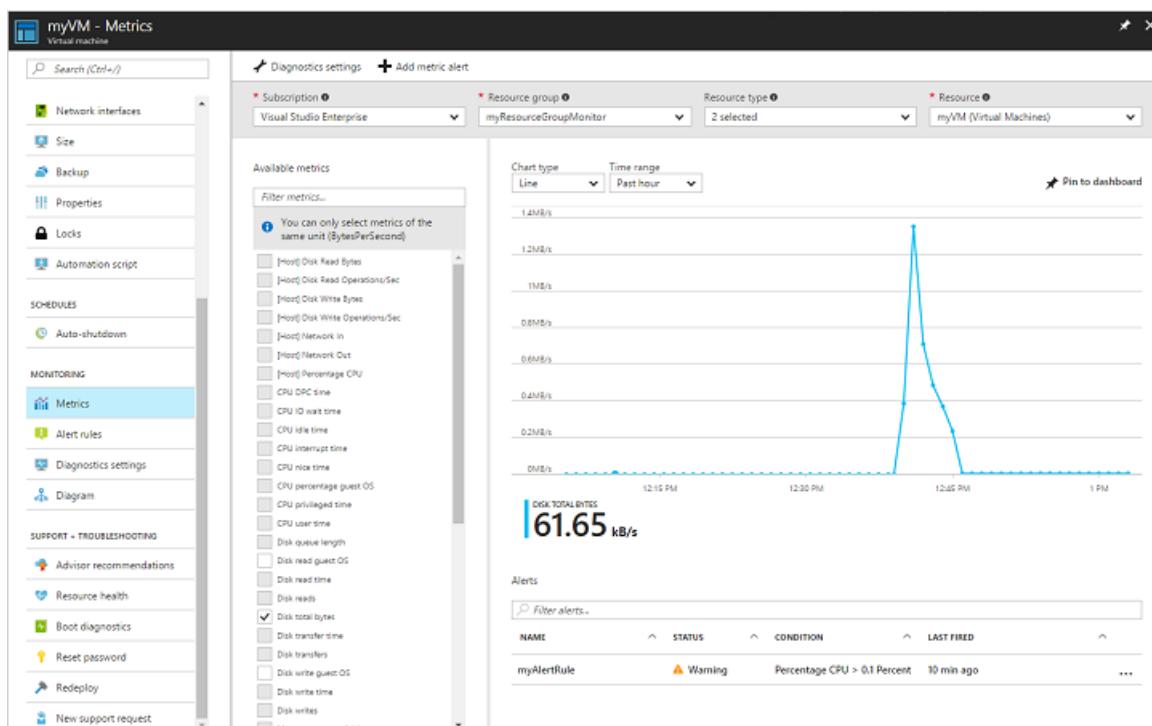
* Pick a storage account
mydiagdata26456

Enable guest-level monitoring

View VM metrics

You can view the VM metrics in the same way that you viewed the host VM metrics:

1. In the Azure portal, choose **Resource Groups**, select **myResourceGroupMonitor**, and then select **myVM** in the resource list.
2. To see how the VM is performing, select **Metrics** on the VM window, and then select any of the [Guest] diagnostics metrics under **Available metrics**.



Create alerts

You can create alerts based on specific performance metrics. Alerts can be used to notify you when average CPU usage exceeds a certain threshold or available free disk space drops below a certain amount, for example. Alerts are displayed in the Azure portal or can be sent via email. You can also trigger Azure Automation runbooks or Azure Logic Apps in response to alerts being generated.

The following example creates an alert for average CPU usage.

1. In the Azure portal, select **Resource Groups**, select **myResourceGroupMonitor**, and then select **myVM** in the resource list.
2. Select **Alerts (classic)**, then choose to **Add metric alert (classic)** across the top of the alerts window.
3. Provide a **Name** for your alert, such as *myAlertRule*

4. To trigger an alert when CPU percentage exceeds 1.0 for five minutes, leave all the other defaults selected.
5. Optionally, check the box for *Email owners, contributors, and readers* to send email notification. The default action is to present a notification in the portal.
6. Select the **OK** button.

Manage software updates

Update management allows you to manage updates and patches for your Azure Linux VMs. Directly from your VM, you can quickly assess the status of available updates, schedule installation of required updates, and review deployment results to verify updates were applied successfully to the VM.

For pricing information, see [Automation pricing for Update management](#)

Enable Update management

Enable Update management for your VM:

1. On the left-hand side of the screen, select **Virtual machines**.
2. From the list, select a VM.
3. On the VM screen, in the **Operations** section, select **Update management**. The **Enable Update Management** screen opens.

Validation is performed to determine if Update management is enabled for this VM. The validation includes checks for a Log Analytics workspace and linked Automation account, and if the solution is in the workspace.

A [Log Analytics](#) workspace is used to collect data that is generated by features and services such as Update management. The workspace provides a single location to review and analyze data from multiple sources. To perform additional actions on VMs that require updates, Azure Automation allows you to run runbooks against VMs, such as download and apply updates.

The validation process also checks to see if the VM is provisioned with the Log Analytics agent and Automation hybrid runbook worker. This agent is used to communicate with the VM and obtain information about the update status.

Choose the Log Analytics workspace and automation account and select **Enable** to enable the solution. The solution takes up to 15 minutes to enable.

If any of the following prerequisites were found to be missing during onboarding, they're automatically added:

- A [Log Analytics workspace](#)
- An [Automation account](#)
- A [Hybrid runbook worker](#) is enabled on the VM

The **Update Management** screen opens. Configure the location, Log Analytics workspace and Automation account to use and select **Enable**. If the fields are grayed out, that means another automation solution is enabled for the VM and the same workspace and Automation account must be used.

Enabling the solution can take up to 15 minutes. During this time, you shouldn't close the browser window. After the solution is enabled, information about missing updates on the VM flows to Azure Monitor logs. It can take between 30 minutes and 6 hours for the data to be available for analysis.

View update assessment

After **Update management** is enabled, the **Update management** screen appears. After the evaluation of updates is complete, you see a list of missing updates on the **Missing updates** tab.

UPDATE NAME	CLASSIFICATION	INFORMATION LINK
apport	✖ Critical updates	
sudo	✖ Critical updates	
linux-firmware	✖ Critical updates	
dosfstools	ⓘ Security updates	
bash	ⓘ Security updates	
cpio	ⓘ Security updates	
curl	ⓘ Security updates	
patch	ⓘ Security updates	

Schedule an update deployment

To install updates, schedule a deployment that follows your release schedule and service window. You can choose which update types to include in the deployment. For example, you can include critical or security updates and exclude update rollups.

Schedule a new Update Deployment for the VM by clicking **Schedule update deployment** at the top of the **Update management** screen. In the **New update deployment** screen, specify the following information:

To create a new update deployment, select **Schedule update deployment**. The **New update deployment** page opens. Enter values for the properties described in the following table and then click **Create**:

PROPERTY	DESCRIPTION
Name	Unique name to identify the update deployment.
Operating System	Linux or Windows
Groups to update	<p>For Azure machines, define a query based on a combination of subscription, resource groups, locations, and tags to build a dynamic group of Azure VMs to include in your deployment. For Non-Azure machines, select an existing saved search to select a group of Non-Azure machines to include in the deployment.</p> <p>To learn more, see Dynamic Groups</p>
Machines to update	<p>Select a Saved search, Imported group, or pick Machine from the drop-down and select individual machines. If you choose Machines, the readiness of the machine is shown in the UPDATE AGENT READINESS column.</p> <p>To learn about the different methods of creating computer groups in Azure Monitor logs, see Computer groups in Azure Monitor logs</p>
Update classifications	Select all the update classifications that you need
Include/exclude updates	This opens the Include/Exclude page. Updates to be included or excluded are on separate tabs. For more information on how inclusion is handled, see inclusion behavior
Schedule settings	Select the time to start, and select either Once or recurring for the recurrence
Pre-scripts + Post-scripts	Select the scripts to run before and after your deployment
Maintenance window	Number of minutes set for updates. The value can't be less than 30 minutes and no more than 6 hours
Reboot control	<p>Determines how reboots should be handled. Available options are:</p> <ul style="list-style-type: none"> Reboot if required (Default) Always reboot Never reboot Only reboot - will not install updates

Update Deployments can also be created programmatically. To learn how to create an Update Deployment with the REST API, see [Software Update Configurations - Create](#). There is also a sample runbook that can be used to create a weekly Update Deployment. To learn more about this runbook, see [Create a weekly update deployment for one or more VMs in a resource group](#).

After you have completed configuring the schedule, click **Create** button and you return to the status dashboard. Notice that the **Scheduled** table shows the deployment schedule you created.

View results of an update deployment

After the scheduled deployment starts, you can see the status for that deployment on the **Update deployments** tab on the **Update management** screen. If it is currently running, its status shows as **In progress**. After it

completes, if successful, it changes to **Succeeded**. If there is a failure with one or more updates in the deployment, the status is **Partially failed**. Select the completed update deployment to see the dashboard for that update deployment.

The screenshot shows the Azure portal interface for a completed update deployment. At the top, it displays the deployment name 'Marketing10' and the target VM 'PatchNewLinuxVM'. Below this, a summary table shows the deployment status as 'Succeeded' with a green checkmark, and the start and end times. The 'Update results' section features a large green circle containing the number '89 Updates'. To the right, a detailed table lists 18 individual updates with their names and statuses: NetworkManager-libnm.x86_64, NetworkManager-tui.x86_64, NetworkManager-team.x86_64, NetworkManager.x86_64, and WALinuxAgent.noarch, all marked as 'Succeeded' with green checkmarks. Below the table are navigation arrows for page 1 through 18. At the bottom, there are tabs for 'Diagnostics and Logs' (with 'All Logs' selected), 'Output' (with a play icon), and 'Errors' (with a red X icon).

In **Update results** tile is a summary of the total number of updates and deployment results on the VM. In the table to the right is a detailed breakdown of each update and the installation results, which could be one of the following values:

- **Not attempted** - the update was not installed because there was insufficient time available based on the maintenance window duration defined.
- **Succeeded** - the update succeeded
- **Failed** - the update failed

Select **All logs** to see all log entries that the deployment created.

Select the **Output** tile to see job stream of the runbook responsible for managing the update deployment on the target VM.

Select **Errors** to see detailed information about any errors from the deployment.

Monitor changes and inventory

You can collect and view inventory for software, files, Linux daemons, Windows Services, and Windows registry keys on your computers. Tracking the configurations of your machines can help you pinpoint operational issues across your environment and better understand the state of your machines.

Enable Change and Inventory management

Enable Change and Inventory management for your VM:

1. On the left-hand side of the screen, select **Virtual machines**.
2. From the list, select a VM.
3. On the VM screen, in the **Operations** section, select **Inventory** or **Change tracking**. The **Enable Change Tracking and Inventory** screen opens.

Configure the location, Log Analytics workspace and Automation account to use and select **Enable**. If the fields are grayed out, that means another automation solution is enabled for the VM and the same workspace and Automation account must be used. Even though the solutions are separate on the menu, they are the same solution. Enabling one enables both for your VM.

After the solution has been enabled, it may take some time while inventory is being collected on the VM before data appears.

Track changes

On your VM, select **Change Tracking** under **OPERATIONS**. Select **Edit Settings**, the **Change Tracking** page is displayed. Select the type of setting you want to track and then select **+ Add** to configure the settings. The available option Linux is **Linux Files**

For detailed information on Change Tracking see, [Troubleshoot changes on a VM](#)

View inventory

On your VM, select **Inventory** under **OPERATIONS**. On the **Software** tab, there is a table list the software that had been found. The high-level details for each software record are viewable in the table. These details include the software name, version, publisher, last refreshed time.

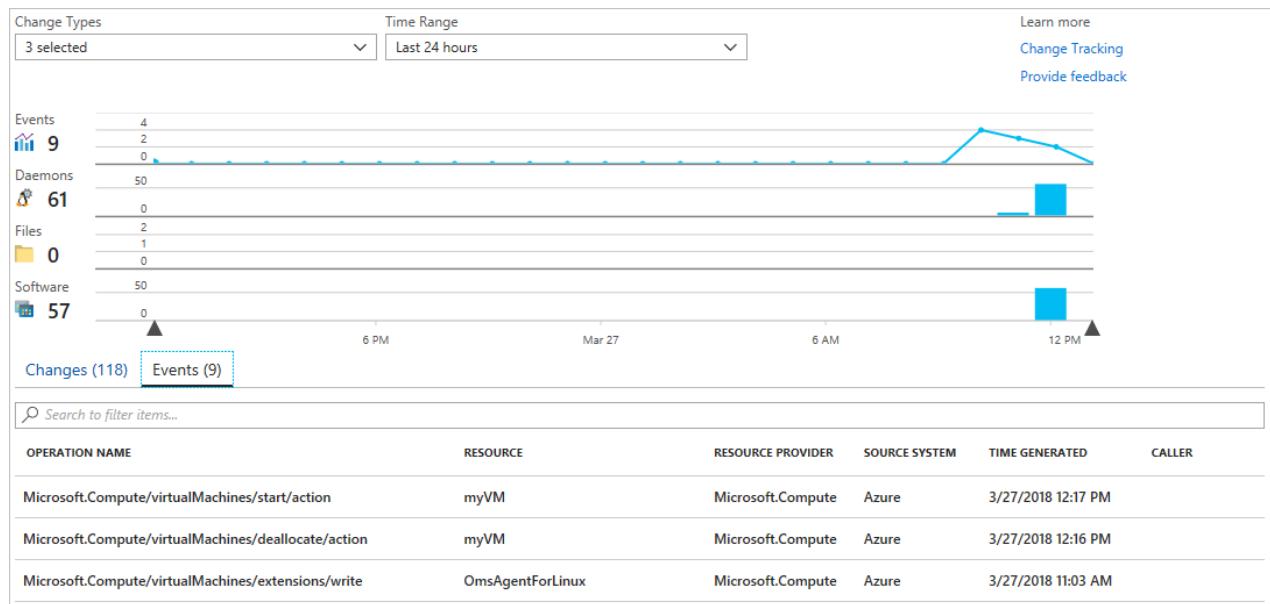
New software 1				Learn more
				Inventory
				Provide feedback
Last 24 hours	Software	Files	Linux Daemons	
	<input type="text"/> Search to filter items...			
NAME	VERSION	PUBLISHER	LAST REFRESHED TIME	...
accounts-service	0.6.40-2ubuntu11.3	Ubuntu Developers <ubun...	3/27/2018 11:51 AM	...
acl	2.2.52-3	Ubuntu Developers <ubun...	3/27/2018 11:51 AM	...
acpid	1:2.0.26-1ubuntu2	Ubuntu Developers <ubun...	3/27/2018 11:51 AM	...
adduser	3.113+nmu3ubuntu4	Ubuntu Core Developers <...	3/27/2018 11:51 AM	...
apparmor	2.10.95-0ubuntu2.8	Ubuntu Developers <ubun...	3/27/2018 11:51 AM	...
apport	2.20.1-0ubuntu2.15	Martin Pitt <martin.pitt@u...	3/27/2018 11:51 AM	...
apport-symptoms	0.20	Ubuntu Developers <ubun...	3/27/2018 11:51 AM	...
apt	1.2.25	Ubuntu Developers <ubun...	3/27/2018 11:51 AM	...

Monitor Activity logs and changes

From the **Change tracking** page on your VM, select **Manage Activity Log Connection**. This task opens the **Azure Activity log** page. Select **Connect** to connect Change tracking to the Azure activity log for your VM.

With this setting enabled, navigate to the **Overview** page for your VM and select **Stop** to stop your VM. When prompted, select **Yes** to stop the VM. When it is deallocated, select **Start** to restart your VM.

Stopping and starting a VM logs an event in its activity log. Navigate back to the **Change tracking** page. Select the **Events** tab at the bottom of the page. After a while, the events shown in the chart and the table. Each event can be selected to view detailed information on the event.

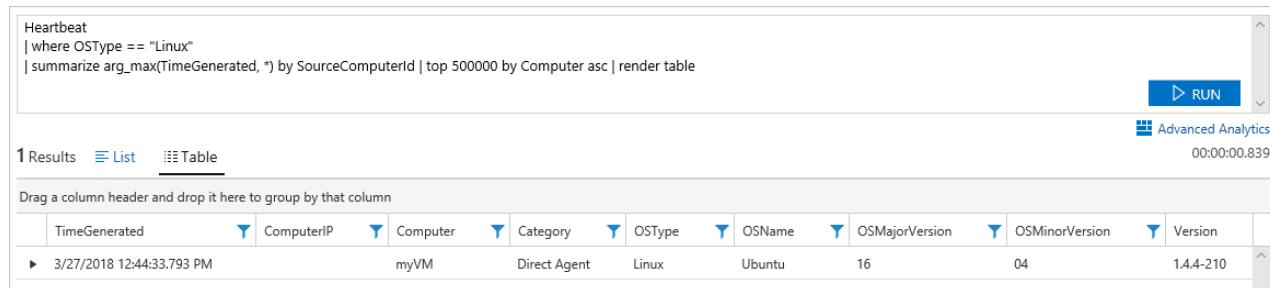


The chart shows changes that have occurred over time. After you have added an Activity Log connection, the line graph at the top displays Azure Activity Log events. Each row of bar graphs represents a different trackable Change type. These types are Linux daemons, files, and software. The change tab shows the details for the changes shown in the visualization in descending order of time that the change occurred (most recent first).

Advanced monitoring

You can do more advanced monitoring of your VM by using a solution like [Azure Monitor for VMs](#), which monitors your Azure virtual machines (VM) at scale by analyzing the performance and health of your Windows and Linux VMs, including their different processes and interconnected dependencies on other resources and external processes. Configuration management of your Azure VMs is delivered with the [Azure Automation](#) Change Tracking and Inventory solution to easily identify changes in your environment. Managing update compliance is provided with the Azure Automation Update Management solution.

From the Log Analytics workspace the VM is connected to, you can also retrieve, consolidate, and analyze collected data with the [rich query language](#).



Next steps

In this tutorial, you configured, reviewed, and managed updates for a VM. You learned how to:

- Enable boot diagnostics on the VM
- View boot diagnostics
- View host metrics
- Enable diagnostics extension on the VM
- View VM metrics
- Create alerts based on diagnostic metrics
- Manage package updates
- Monitor changes and inventory
- Set up advanced monitoring

Advance to the next tutorial to learn about Azure Security Center.

[Manage VM security](#)

Tutorial: Use Azure Security Center to monitor Linux virtual machines

2/21/2019 • 4 minutes to read • [Edit Online](#)

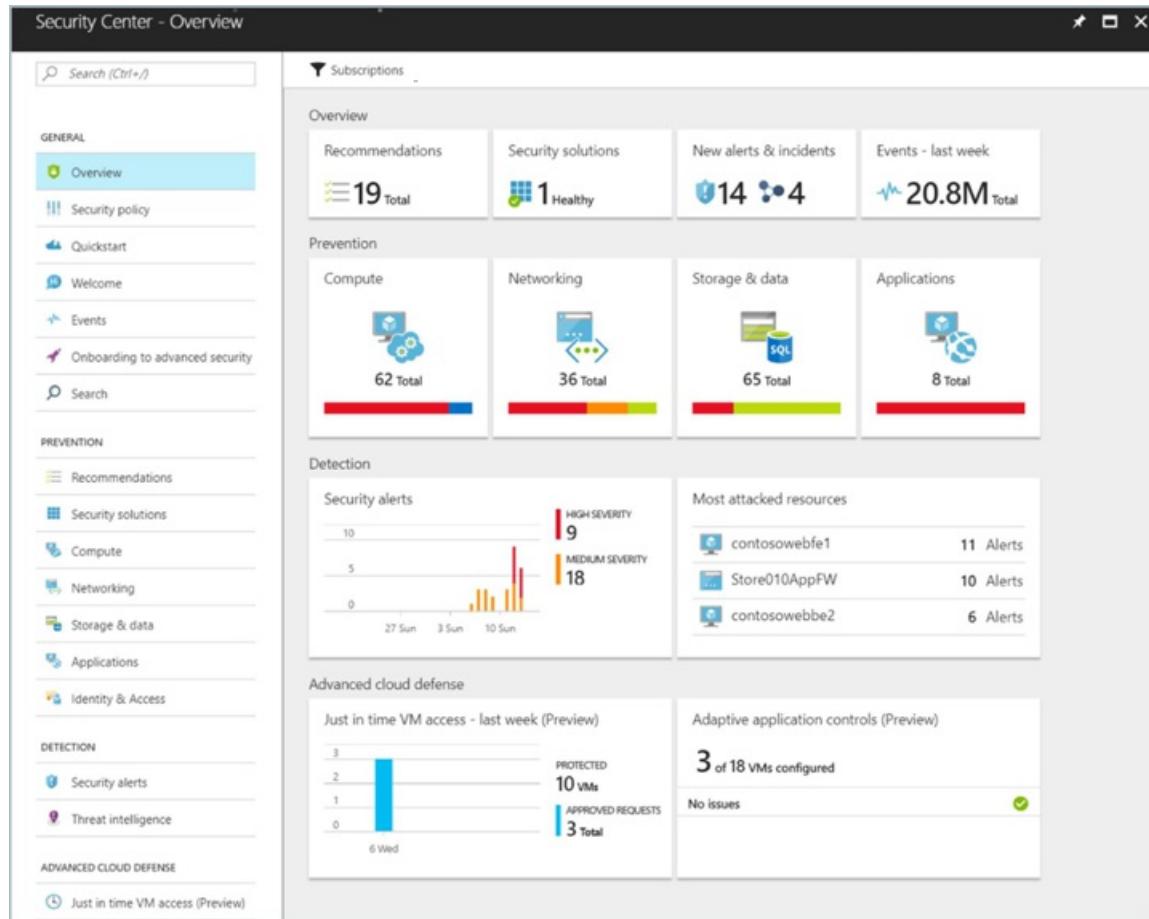
Azure Security Center can help you gain visibility into your Azure resource security practices. Security Center offers integrated security monitoring. It can detect threats that otherwise might go unnoticed. In this tutorial, you learn about Azure Security Center, and how to:

- Set up data collection
- Set up security policies
- View and fix configuration health issues
- Review detected threats

Security Center overview

Security Center identifies potential virtual machine (VM) configuration issues and targeted security threats. These might include VMs that are missing network security groups, unencrypted disks, and brute-force Remote Desktop Protocol (RDP) attacks. The information is shown on the Security Center dashboard in easy-to-read graphs.

To access the Security Center dashboard, in the Azure portal, on the menu, select **Security Center**. On the dashboard, you can see the security health of your Azure environment, find a count of current recommendations, and view the current state of threat alerts. You can expand each high-level chart to see more detail.



Security Center goes beyond data discovery to provide recommendations for issues that it detects. For example, if a VM was deployed without an attached network security group, Security Center displays a recommendation, with

remediation steps you can take. You get automated remediation without leaving the context of Security Center.

The screenshot shows the 'Recommendations' blade in the Azure Security Center. At the top, there's a 'Filter' button. Below it is a table with columns: DESCRIPTION, RESOURCE, STATE, and SEVERITY. The table lists several recommendations:

DESCRIPTION	RESOURCE	STATE	SEVERITY	...
Install Endpoint Protection	2016OMSAA	Open	! High	...
Add a Next Generation Firewall	2 endpoints	Open	! High	...
Enable Network Security Groups on sub...	2 subnets	Open	! High	...
Apply disk encryption	3 virtual mac...	Open	! High	...
Enable encryption for Azure Storage Acc...	9 storage acc...	Open	! High	...
Restrict access through Internet facing e...	2 virtual mac...	Open	⚠ Medium	...
Add a vulnerability assessment solution	2016OMSAA	Open	⚠ Medium	...
Remediate OS vulnerabilities (by Micros...	2016OMSLin...	Open	ℹ Low	...

Set up data collection

Before you can get visibility into VM security configurations, you need to set up Security Center data collection. This involves turning on data collection which automatically installs the Microsoft Monitoring Agent on all the VMs in your subscription.

1. On the Security Center dashboard, click **Security policy**, and then select your subscription.
2. For **Data collection**, in **Auto Provisioning** select **On**.
3. For **Default workspace configuration** leave it as **Use workspace(s) created by Security Center (default)**.
4. Under **Security Events** keep the default option of **Common**.
5. Click **Save** at the top of the page.

The Security Center data collection agent is then installed on all VMs, and data collection begins.

Set up a security policy

Security policies are used to define the items for which Security Center collects data and makes recommendations. You can apply different security policies to different sets of Azure resources. Although by default Azure resources are evaluated against all policy items, you can turn off individual policy items for all Azure resources or for a resource group. For in-depth information about Security Center security policies, see [Set security policies in Azure Security Center](#).

To set up a security policy for an entire subscription:

1. On the Security Center dashboard, select **Security policy** and then select your subscription.
2. On the **Security policy** blade, select **Security policy**.
3. On the **Security policy - Security policy** blade, turn on or turn off policy items that you want to apply to the subscription.
4. When you're finished selecting your settings, select **Save** at the top of the blade.

The screenshot shows the 'Security policy - Security policy' blade in the Azure Security Center. On the left, there's a sidebar titled 'POLICY COMPONENTS' with options: Data Collection (selected), Security policy (highlighted with a blue background), Email notifications, and Pricing tier. The main content area is titled 'Show recommendations for' and lists various security components with toggle switches:

Setting	Value	Status
System updates	On	Off
Security configurations	On	Off
Endpoint protection	On	Off
Disk encryption	On	Off
Network security groups	On	Off
Web application firewall	On	Off
Next generation firewall	On	Off
Vulnerability Assessment	On	Off
Storage Encryption	On	Off
JIT Network Access	On	Off
Adaptive Application Controls	On	Off
SQL auditing & Threat detection	On	Off
SQL Encryption	On	Off

View VM configuration health

After you've turned on data collection and set a security policy, Security Center begins to provide alerts and recommendations. As VMs are deployed, the data collection agent is installed. Security Center is then populated with data for the new VMs. For in-depth information about VM configuration health, see [Protect your VMs in Security Center](#).

As data is collected, the resource health for each VM and related Azure resource is aggregated. The information is shown in an easy-to-read chart.

To view resource health:

1. On the Security Center dashboard, under **Prevention**, select **Compute**.
2. On the **Compute** blade, select **VMs and computers**. This view provides a summary of the configuration status for all your VMs.

The screenshot shows the Microsoft Security Center Compute dashboard. At the top, there are three main navigation items: 'Overview' (with a bar chart icon), 'VMs and computers' (with a computer monitor icon), and 'Cloud services' (with a cloud icon). Below these is a search bar labeled 'Search by name'. The main table displays four VMs: 'myVM', 'BackEnd', 'Database', and 'FrontEnd'. Each row includes columns for 'NAME', 'MONITORED' (green checkmark), 'SYSTEM UPDATES' (red exclamation mark), 'ENDPOINT PROTECT...' (green checkmark), 'VULNERABILITIES' (orange triangle), and 'DISK ENCRYPTION' (red exclamation mark).

To see all recommendations for a VM, select the VM.

Remediate configuration issues

After Security Center begins to populate with configuration data, recommendations are made based on the security policy you set up. For instance, if a VM was set up without an associated network security group, a recommendation is made to create one.

To see a list of all recommendations:

1. On the Security Center dashboard, select **Recommendations**.
2. Select a specific recommendation. A list of all resources for which the recommendation applies appears.
3. To apply a recommendation, select the resource.
4. Follow the instructions for remediation steps.

In many cases, Security Center provides actionable steps you can take to address a recommendation without leaving Security Center. In the following example, Security Center detects a network security group that has an unrestricted inbound rule. On the recommendation page, you can select the **Edit inbound rules** button. The UI that is needed to modify the rule appears.

The screenshot shows the 'Edit inbound rules' dialog for the 'FrontEnd' VM. It includes sections for 'Network security group info' (Network Security Group: 'FrontEnd', Location: 'eastus'), a warning about unrestricted inbound rules, and a table of 'Related inbound rules' (Priority 1000: 'FrontEnd3389' - Allow, Priority 1001: 'FrontEnd5985' - Allow). Below this is an 'Associated with' section showing the VM itself.

VIRTUAL MACHINE	IP	STATE	SEVERITY	...
BackEnd	168.62.41.154	Open	Medium	...
Database	23.96.5.203	Open	Medium	...
FrontEnd	40.76.193.8	Open	Medium	...
myVM	168.62.166.52	Open	Medium	...

PRIORITY	NAME	SOURCE	SERVICE	ACTIONS
1000	FrontEnd3389	*	Tcp	Allow
1001	FrontEnd5985	*	Tcp	Allow

As recommendations are remediated, they are marked as resolved.

View detected threats

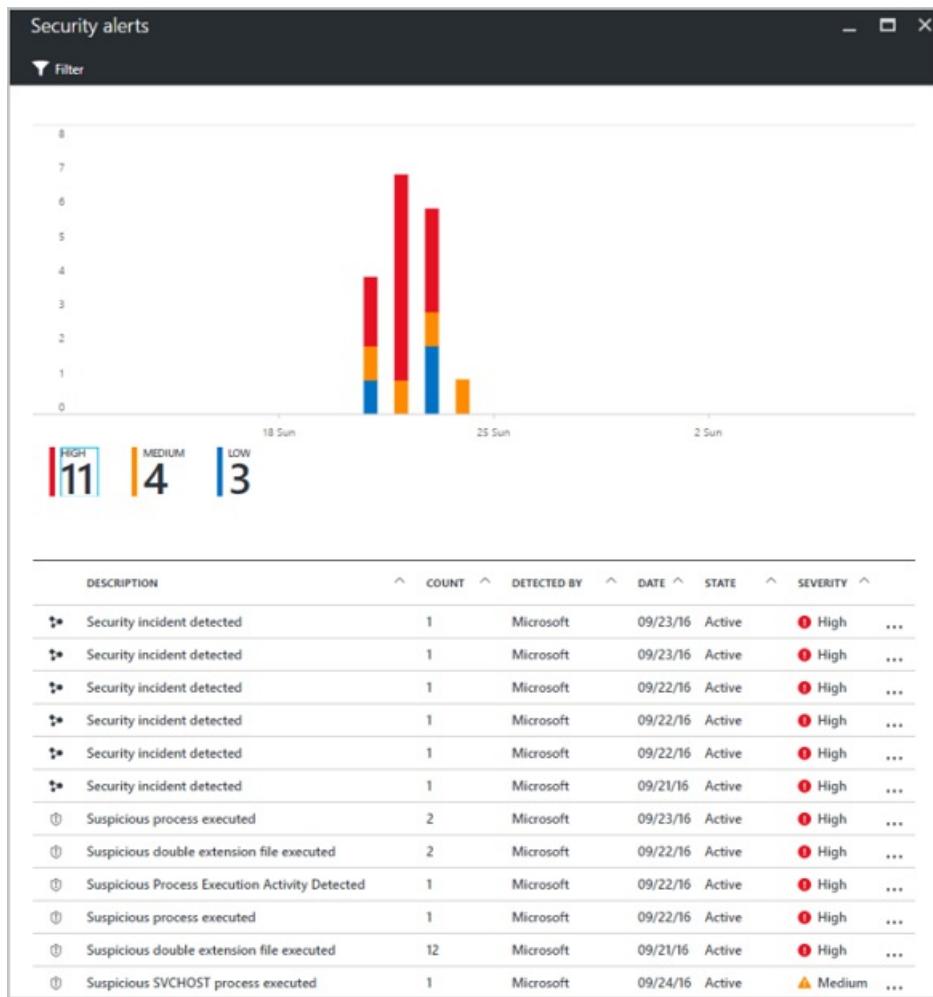
In addition to resource configuration recommendations, Security Center displays threat detection alerts. The security alerts feature aggregates data collected from each VM, Azure networking logs, and connected partner solutions to detect security threats against Azure resources. For in-depth information about Security Center threat detection capabilities, see [Azure Security Center detection capabilities](#).

The security alerts feature requires the Security Center pricing tier to be increased from *Free* to *Standard*. A **free trial** is available when you move to this higher pricing tier.

To change the pricing tier:

1. On the Security Center dashboard, click **Security policy**, and then select your subscription.
2. Select **Pricing tier**.
3. Select **Standard** and then click **Save** at the top of the blade.

After you've changed the pricing tier, the security alerts graph begins to populate as security threats are detected.



Select an alert to view information. For example, you can see a description of the threat, the detection time, all threat attempts, and the recommended remediation. In the following example, an RDP brute-force attack was detected, with 294 failed RDP attempts. A recommended resolution is provided.

Failed RDP Brute Force Attack	
myVMWindows	
DESCRIPTION	
DESCRIPTION	Several Remote Desktop login attempts were detected from 5.9.57.202, none of them succeeded. Event logs analysis shows that in the last 59 minutes there were 198 failed attempts. Some of the failed login attempts aimed at 2 existing user(s).
DETECTION TIME	Saturday, April 29, 2017, 10:59:56 PM
SEVERITY	 Medium
STATE	Active
ATTACKED RESOURCE	myVMWindows
SUBSCRIPTION	Free Trial (248352d0-5fc9-4c2e-8db3-d8b3462a0020)
DETECTED BY	 Microsoft
ACTION TAKEN	Detected
ALERT START TIME (UTC)	04/30/2017 05:00:06
NON-EXISTENT USERS	97
SUCCESSFUL LOGINS	0
FAILED USER LOGONS	server, administrator
REPORTS	Report: RDP Brute Forcing
REMEDIATION STEPS	
REMEDIATION STEPS	<ol style="list-style-type: none"> 1. If available, add the source IP to NSG block list for 24 hours (see https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-nsg/) 2. Enforce the use of strong passwords and do not reuse them across multiple VMs and services (see http://windows.microsoft.com/en-us/Windows7/Tips-for-creating-strong-passwords-and-passphrases) 3. Create an allow list for RDP access in NSG (see https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-nsg/)

Next steps

In this tutorial, you set up Azure Security Center, and then reviewed VMs in Security Center. You learned how to:

- Set up data collection
- Set up security policies
- View and fix configuration health issues
- Review detected threats

Advance to the next tutorial to learn more about creating a CI/CD pipeline with Jenkins, GitHub, and Docker.

[Create CI/CD infrastructure with Jenkins, GitHub, and Docker](#)

Tutorial: Create a development infrastructure on a Linux VM in Azure with Jenkins, GitHub, and Docker

3/14/2019 • 9 minutes to read • [Edit Online](#)

To automate the build and test phase of application development, you can use a continuous integration and deployment (CI/CD) pipeline. In this tutorial, you create a CI/CD pipeline on an Azure VM including how to:

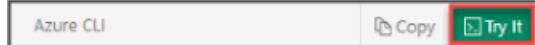
- Create a Jenkins VM
- Install and configure Jenkins
- Create webhook integration between GitHub and Jenkins
- Create and trigger Jenkins build jobs from GitHub commits
- Create a Docker image for your app
- Verify GitHub commits build new Docker image and updates running app

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create Jenkins instance

In a previous tutorial on [How to customize a Linux virtual machine on first boot](#), you learned how to automate VM customization with cloud-init. This tutorial uses a cloud-init file to install Jenkins and Docker on a VM. Jenkins is a popular open-source automation server that integrates seamlessly with Azure to enable continuous integration (CI) and continuous delivery (CD). For more tutorials on how to use Jenkins, see the [Jenkins in Azure hub](#).

In your current shell, create a file named `cloud-init-jenkins.txt` and paste the following configuration. For example, create the file in the Cloud Shell not on your local machine. Enter `sensible-editor cloud-init-jenkins.txt` to

create the file and see a list of available editors. Make sure that the whole cloud-init file is copied correctly, especially the first line:

```
#cloud-config
package_upgrade: true
write_files:
  - path: /etc/systemd/system/docker.service.d/docker.conf
    content: |
      [Service]
      ExecStart=
      ExecStart=/usr/bin/dockerd
  - path: /etc/docker/daemon.json
    content: |
      {
        "hosts": ["fd://","tcp://127.0.0.1:2375"]
      }
runcmd:
  - apt install openjdk-8-jre-headless -y
  - wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
  - sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
  - apt-get update && apt-get install jenkins -y
  - curl -sSL https://get.docker.com/ | sh
  - usermod -aG docker azureuser
  - usermod -aG docker jenkins
  - service jenkins restart
```

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

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

Now create a VM with [az vm create](#). Use the `--custom-data` parameter to pass in your cloud-init config file. Provide the full path to *cloud-init-jenkins.txt* if you saved the file outside of your present working directory.

```
az vm create --resource-group myResourceGroupJenkins \
  --name myVM \
  --image UbuntuLTS \
  --admin-username azureuser \
  --generate-ssh-keys \
  --custom-data cloud-init-jenkins.txt
```

It takes a few minutes for the VM to be created and configured.

To allow web traffic to reach your VM, use [az vm open-port](#) to open port *8080* for Jenkins traffic and port *1337* for the Node.js app that is used to run a sample app:

```
az vm open-port --resource-group myResourceGroupJenkins --name myVM --port 8080 --priority 1001
az vm open-port --resource-group myResourceGroupJenkins --name myVM --port 1337 --priority 1002
```

Configure Jenkins

To access your Jenkins instance, obtain the public IP address of your VM:

```
az vm show --resource-group myResourceGroupJenkins --name myVM -d --query [publicIps] --o tsv
```

For security purposes, you need to enter the initial admin password that is stored in a text file on your VM to start the Jenkins install. Use the public IP address obtained in the previous step to SSH to your VM:

```
ssh azureuser@<publicIps>
```

Verify Jenkins is running using the `service` command:

```
$ service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
  Active: active (exited) since Tue 2019-02-12 16:16:11 UTC; 55s ago
    Docs: man:systemd-sysv-generator(8)
   Tasks: 0 (limit: 4103)
  CGroup: /system.slice/jenkins.service

Feb 12 16:16:10 myVM systemd[1]: Starting LSB: Start Jenkins at boot time...
...
...
```

View the `initialAdminPassword` for your Jenkins install and copy it:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

If the file isn't available yet, wait a couple more minutes for cloud-init to complete the Jenkins and Docker install.

Now open a web browser and go to `http://<publicIps>:8080`. Complete the initial Jenkins setup as follows:

- Choose **Select plugins to install**
- Search for *Github* in the text box across the top. Check the box for *Github*, then select **Install**
- Create the first admin user. Enter a username, such as **admin**, then provide your own secure password. Finally, type a full name and e-mail address.
- Select **Save and Finish**
- Once Jenkins is ready, select **Start using Jenkins**
 - If your web browser displays a blank page when you start using Jenkins, restart the Jenkins service.
From your SSH session, type `sudo service jenkins restart`, then refresh your web browser.
- If needed, log in to Jenkins with the username and password you created.

Create GitHub webhook

To configure the integration with GitHub, open the [Node.js Hello World sample app](#) from the Azure samples repo.

To fork the repo to your own GitHub account, select the **Fork** button in the top right-hand corner.

Create a webhook inside the fork you created:

- Select **Settings**, then select **Webhooks** on the left-hand side.
- Choose **Add webhook**, then enter *Jenkins* in filter box.
- For the **Payload URL**, enter `http://<publicIps>:8080/github-webhook/`. Make sure you include the trailing /
- For **Content type**, select *application/x-www-form-urlencoded*.
- For **Which events would you like to trigger this webhook?**, select *Just the push event*.
- Set **Active** to checked.
- Click **Add webhook**.

The screenshot shows the GitHub fork settings for the repository 'nodejs-docs-hello-world'. The 'Webhooks' tab is selected in the sidebar. The main area displays the 'Add webhook' configuration page. It includes fields for 'Payload URL' (set to 'http://13.82.180.34/github-webhook/'), 'Content type' (set to 'application/x-www-form-urlencoded'), and a 'Secret' field. Below these, there's a section for selecting events: 'Just the push event.' is selected. There's also a checkbox for 'Active' which is checked, with a note: 'We will deliver event details when this hook is triggered.' At the bottom is a green 'Add webhook' button.

Create Jenkins job

To have Jenkins respond to an event in GitHub such as committing code, create a Jenkins job. Use the URLs for your own GitHub fork.

In your Jenkins website, select **Create new jobs** from the home page:

- Enter *HelloWorld* as job name. Choose **Freestyle project**, then select **OK**.
- Under the **General** section, select **GitHub project** and enter your forked repo URL, such as <https://github.com/cynthn/nodejs-docs-hello-world>
- Under the **Source code management** section, select **Git**, enter your forked repo .git URL, such as <https://github.com/cynthn/nodejs-docs-hello-world.git>
- Under the **Build Triggers** section, select **GitHub hook trigger for GITscm polling**.
- Under the **Build** section, choose **Add build step**. Select **Execute shell**, then enter `echo "Test"` in the command window.
- Select **Save** at the bottom of the jobs window.

Test GitHub integration

To test the GitHub integration with Jenkins, commit a change in your fork.

Back in GitHub web UI, select your forked repo, and then select the **index.js** file. Select the pencil icon to edit this file so line 6 reads:

```
response.end("Hello World!");
```

To commit your changes, select the **Commit changes** button at the bottom.

In Jenkins, a new build starts under the **Build history** section of the bottom left-hand corner of your job page. Choose the build number link and select **Console output** on the left-hand side. You can view the steps Jenkins takes as your code is pulled from GitHub and the build action outputs the message `Test` to the console. Each time a commit is made in GitHub, the webhook reaches out to Jenkins and triggers a new build in this way.

Define Docker build image

To see the Node.js app running based on your GitHub commits, let's build a Docker image to run the app. The image is built from a Dockerfile that defines how to configure the container that runs the app.

From the SSH connection to your VM, change to the Jenkins workspace directory named after the job you created in a previous step. In this example, that was named *HelloWorld*.

```
cd /var/lib/jenkins/workspace/HelloWorld
```

Create a file in this workspace directory with `sudo sensible-editor Dockerfile` and paste the following contents. Make sure that the whole Dockerfile is copied correctly, especially the first line:

```
FROM node:alpine

EXPOSE 1337

WORKDIR /var/www
COPY package.json /var/www/
RUN npm install
COPY index.js /var/www/
```

This Dockerfile uses the base Node.js image using Alpine Linux, exposes port 1337 that the Hello World app runs on, then copies the app files and initializes it.

Create Jenkins build rules

In a previous step, you created a basic Jenkins build rule that output a message to the console. Let's create the build step to use our Dockerfile and run the app.

Back in your Jenkins instance, select the job you created in a previous step. Select **Configure** on the left-hand side and scroll down to the **Build** section:

- Remove your existing `echo "Test"` build step. Select the red cross on the top right-hand corner of the existing build step box.
- Choose **Add build step**, then select **Execute shell**
- In the **Command** box, enter the following Docker commands, then select **Save**:

```
docker build --tag helloworld:$BUILD_NUMBER .
docker stop helloworld && docker rm helloworld
docker run --name helloworld -p 1337:1337 helloworld:$BUILD_NUMBER node /var/www/index.js &
```

The Docker build steps create an image and tag it with the Jenkins build number so you can maintain a history of images. Any existing containers running the app are stopped and then removed. A new container is then started using the image and runs your Node.js app based on the latest commits in GitHub.

Test your pipeline

To see the whole pipeline in action, edit the `index.js` file in your forked GitHub repo again and select **Commit change**. A new job starts in Jenkins based on the webhook for GitHub. It takes a few seconds to create the Docker image and start your app in a new container.

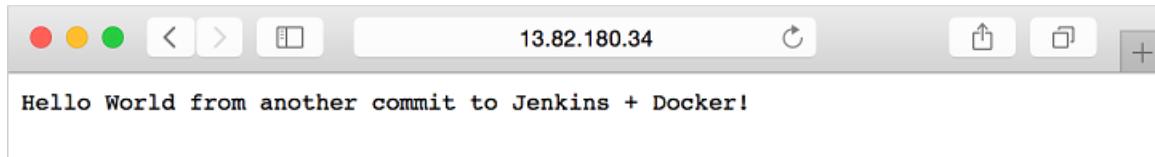
If needed, obtain the public IP address of your VM again:

```
az vm show --resource-group myResourceGroupJenkins --name myVM -d --query [publicIps] --o tsv
```

Open a web browser and enter `http://<publicIps>:1337`. Your Node.js app is displayed and reflects the latest commits in your GitHub fork as follows:



Now make another edit to the `index.js` file in GitHub and commit the change. Wait a few seconds for the job to complete in Jenkins, then refresh your web browser to see the updated version of your app running in a new container as follows:



Next steps

In this tutorial, you configured GitHub to run a Jenkins build job on each code commit and then deploy a Docker container to test your app. You learned how to:

- Create a Jenkins VM
- Install and configure Jenkins
- Create webhook integration between GitHub and Jenkins
- Create and trigger Jenkins build jobs from GitHub commits
- Create a Docker image for your app
- Verify GitHub commits build new Docker image and updates running app

Advance to the next tutorial to learn more about how to integrate Jenkins with Azure DevOps Services.

[Deploy apps with Jenkins and Azure DevOps Services](#)

Tutorial: Deploy your app to Linux virtual machines in Azure with using Jenkins and Azure DevOps Services

3/14/2019 • 7 minutes to read • [Edit Online](#)

Continuous integration (CI) and continuous deployment (CD) form a pipeline by which you can build, release, and deploy your code. Azure DevOps Services provides a complete, fully featured set of CI/CD automation tools for deployment to Azure. Jenkins is a popular third-party CI/CD server-based tool that also provides CI/CD automation. You can use Azure DevOps Services and Jenkins together to customize how you deliver your cloud app or service.

In this tutorial, you use Jenkins to build a Node.js web app. You then use Azure DevOps to deploy it to a [deployment group](#) that contains Linux virtual machines (VMs). You learn how to:

- Get the sample app.
- Configure Jenkins plug-ins.
- Configure a Jenkins Freestyle project for Node.js.
- Configure Jenkins for Azure DevOps Services integration.
- Create a Jenkins service endpoint.
- Create a deployment group for the Azure virtual machines.
- Create an Azure Pipelines release pipeline.
- Execute manual and CI-triggered deployments.

Before you begin

- You need access to a Jenkins server. If you have not yet created a Jenkins server, see [Create a Jenkins master on an Azure virtual machine](#).
- Sign in to your Azure DevOps Services organization (<https://{{yourorganization}}.visualstudio.com>). You can get a [free Azure DevOps Services organization](#).

NOTE

For more information, see [Connect to Azure DevOps Services](#).

- You need a Linux virtual machine for a deployment target. For more information, see [Create and manage Linux VMs with the Azure CLI](#).
- Open inbound port 80 for your virtual machine. For more information, see [Create network security groups using the Azure portal](#).

Get the sample app

You need an app to deploy, stored in a Git repository. For this tutorial, we recommend that you use [this sample app available from GitHub](#). This tutorial contains a sample script that's used for installing Node.js and an application. If you want to work with your own repository, you should configure a similar sample.

Create a fork of this app and take note of the location (URL) for use in later steps of this tutorial. For more information, see [Fork a repo](#).

NOTE

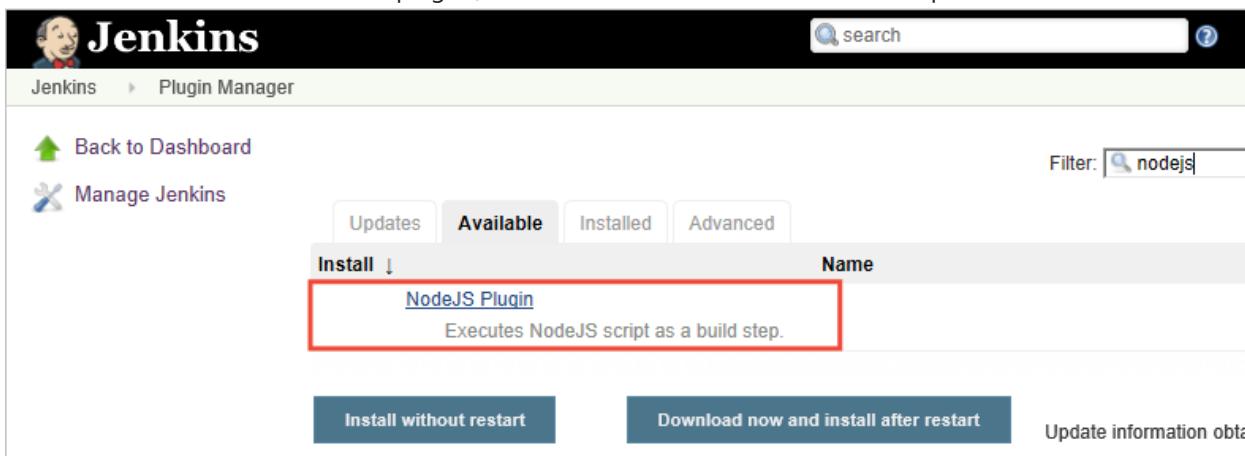
The app was built through [Yeoman](#). It uses Express, bower, and grunt. And it has some npm packages as dependencies. The sample also contains a script that sets up Nginx and deploys the app. It is executed on the virtual machines. Specifically, the script:

1. Installs Node, Nginx, and PM2.
2. Configures Nginx and PM2.
3. Starts the Node app.

Configure Jenkins plug-ins

First, you must configure two Jenkins plug-ins: **NodeJS** and **VS Team Services Continuous Deployment**.

1. Open your Jenkins account and select **Manage Jenkins**.
2. On the **Manage Jenkins** page, select **Manage Plugins**.
3. Filter the list to locate the **NodeJS** plug-in, and select the **Install without restart** option.



The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and a help icon. Below the navigation bar, the title 'Plugin Manager' is displayed. On the left, there are links for 'Back to Dashboard' and 'Manage Jenkins'. In the center, there are four tabs: 'Updates' (disabled), 'Available' (selected), 'Installed', and 'Advanced'. A filter bar at the top right contains a search icon and the text 'nodejs'. Below the tabs, a table lists available plugins. One plugin, 'NodeJS Plugin', is highlighted with a red border. Its details are shown in a tooltip: 'Executes NodeJS script as a build step.' At the bottom of the table, there are two buttons: 'Install without restart' (highlighted in red) and 'Download now and install after restart'. To the right of these buttons, a link says 'Update information obtained'.

4. Filter the list to find the **VS Team Services Continuous Deployment** plug-in and select the **Install without restart** option.
5. Go back to the Jenkins dashboard and select **Manage Jenkins**.
6. Select **Global Tool Configuration**. Find **NodeJS** and select **NodeJS installations**.
7. Select the **Install automatically** option, and then enter a **Name** value.
8. Select **Save**.

Configure a Jenkins Freestyle project for Node.js

1. Select **New Item**. Enter an item name.
2. Select **Freestyle project**. Select **OK**.
3. On the **Source Code Management** tab, select **Git** and enter the details of the repository and the branch that contain your app code.

- On the **Build Triggers** tab, select **Poll SCM** and enter the schedule `H/03 * * * *` to poll the Git repository for changes every three minutes.
- On the **Build Environment** tab, select **Provide Node & npm bin/ folder PATH** and select the **NodeJS Installation** value. Leave **npmrc file** set to **use system default**.
- On the **Build** tab, select **Execute shell** and enter the command `npm install` to ensure that all dependencies are updated.

Configure Jenkins for Azure DevOps Services integration

NOTE

Ensure that the personal access token (PAT) you use for the following steps contains the *Release* (read, write, execute and manage) permission in Azure DevOps Services.

- Create a PAT in your Azure DevOps Services organization if you don't already have one. Jenkins requires this information to access your Azure DevOps Services organization. Be sure to store the token information for upcoming steps in this section.
To learn how to generate a token, read [How do I create a personal access token for Azure DevOps Services?](#).
- In the **Post-build Actions** tab, select **Add post-build action**. Select **Archive the artifacts**.
- For **Files to archive**, enter `**/*` to include all files.
- To create another action, select **Add post-build action**.
- Select **Trigger release in TFS/Team Services**. Enter the URI for your Azure DevOps Services organization, such as `https://<your-organization-name>.visualstudio.com`.
- Enter the **Project** name.
- Choose a name for the release pipeline. (You create this release pipeline later in Azure DevOps Services.)

8. Choose credentials to connect to your Azure DevOps Services or Team Foundation Server environment:

- Leave **Username** blank if you are using Azure DevOps Services.
- Enter a username and password if you are using an on-premises version of Team Foundation Server.

Trigger release in TFS/Team Services	
Collection url	<input type="text" value="https://adventworks.visualstudio.com"/>
Team project	<input type="text" value="AW"/>
Release definition	<input type="text" value="Adventworks Linux CD"/>
Username	<input type="text" value="admin"/>
Password or PAT	<input type="password" value="*****"/>

9. Save the Jenkins project.

Create a Jenkins service endpoint

A service endpoint allows Azure DevOps Services to connect to Jenkins.

1. Open the **Services** page in Azure DevOps Services, open the **New Service Endpoint** list, and select **Jenkins**.

The screenshot shows the 'Endpoints' section of the Azure DevOps Services 'Services' page. A red box highlights the 'Jenkins' option in the list, which is currently selected. Other options shown include 'Generic', 'GitHub', and 'Kubernetes'. The top navigation bar has a 'Fabrikam' dropdown, a search icon, and tabs for Home, Code, Work, Build & Release, Test, and Services (which is the active tab). A red box also highlights the 'Services' tab itself.

2. Enter a name for the connection.
3. Enter the URL of your Jenkins server, and select the **Accept untrusted SSL certificates** option. An example URL is <http://YourJenkinsURL.westcentralus.cloudapp.azure.com>.
4. Enter the username and password for your Jenkins account.
5. Select **Verify connection** to check that the information is correct.
6. Select **OK** to create the service endpoint.

Create a deployment group for Azure virtual machines

You need a [deployment group](#) to register the Azure DevOps Services agent so the release pipeline can be deployed to your virtual machine. Deployment groups make it easy to define logical groups of target machines for deployment, and to install the required agent on each machine.

NOTE

In the following procedure, be sure to install the prerequisites and *don't run the script with sudo privileges*.

1. Open the **Releases** tab of the **Build & Release** hub, open **Deployment groups**, and select **+ New**.
2. Enter a name for the deployment group, and an optional description. Then select **Create**.
3. Choose the operating system for your deployment target virtual machine. For example, select **Ubuntu 16.04+**.

4. Select **Use a personal access token in the script for authentication.**
5. Select the **System prerequisites** link. Install the prerequisites for your operating system.
6. Select **Copy script to clipboard** to copy the script.
7. Log in to your deployment target virtual machine and run the script. Don't run the script with sudo privileges.
8. After the installation, you are prompted for deployment group tags. Accept the defaults.
9. In Azure DevOps Services, check for your newly registered virtual machine in **Targets** under **Deployment Groups**.

Create an Azure Pipelines release pipeline

A release pipeline specifies the process that Azure Pipelines uses to deploy the app. In this example, you execute a shell script.

To create the release pipeline in Azure Pipelines:

1. Open the **Releases** tab of the **Build & Release** hub, and select **Create release pipeline**.
2. Select the **Empty** template by choosing to start with an **Empty process**.
3. In the **Artifacts** section, select **+ Add Artifact** and choose **Jenkins** for **Source type**. Select your Jenkins service endpoint connection. Then select the Jenkins source job and select **Add**.
4. Select the ellipsis next to **Environment 1**. Select **Add deployment group phase**.
5. Choose your deployment group.
6. Select **+** to add a task to **Deployment group phase**.
7. Select the **Shell Script** task and select **Add**. The **Shell Script** task provides the configuration for a script to run on each server in order to install Node.js and start the app.
8. For **Script Path**, enter `$(System.DefaultWorkingDirectory)/Fabrikam-Node/deployscript.sh`.
9. Select **Advanced**, and then enable **Specify Working Directory**.
10. For **Working Directory**, enter `$(System.DefaultWorkingDirectory)/Fabrikam-Node`.
11. Edit the name of the release pipeline to the name that you specified on the **Post-build Actions** tab of the build in Jenkins. Jenkins requires this name to be able to trigger a new release when the source artifacts are updated.
12. Select **Save** and select **OK** to save the release pipeline.

Execute manual and CI-triggered deployments

1. Select **+** **Release** and select **Create Release**.
2. Select the build that you completed in the highlighted drop-down list, and select **Queue**.
3. Choose the release link in the pop-up message. For example: "Release **Release-1** has been created."
4. Open the **Logs** tab to watch the release console output.
5. In your browser, open the URL of one of the servers that you added to your deployment group. For example, enter `http://{your-server-ip-address}`.
6. Go to the source Git repository and modify the contents of the **h1** heading in the file `app/views/index.jade` with some changed text.
7. Commit your change.
8. After a few minutes, you will see a new release created on the **Releases** page of Azure DevOps. Open the release to see the deployment taking place. Congratulations!

Troubleshooting the Jenkins plugin

If you encounter any bugs with the Jenkins plugins, file an issue in the [Jenkins JIRA](#) for the specific component.

Next steps

In this tutorial, you automated the deployment of an app to Azure by using Jenkins for build and Azure DevOps Services for release. You learned how to:

- Build your app in Jenkins.
- Configure Jenkins for Azure DevOps Services integration.
- Create a deployment group for the Azure virtual machines.
- Create a release pipeline that configures the VMs and deploys the app.

To learn about how to deploy a LAMP (Linux, Apache, MySQL, and PHP) stack, advance to the next tutorial.

[Deploy LAMP stack](#)

Tutorial: Install a LAMP web server on a Linux virtual machine in Azure

2/5/2019 • 6 minutes to read • [Edit Online](#)

This article walks you through how to deploy an Apache web server, MySQL, and PHP (the LAMP stack) on an Ubuntu VM in Azure. If you prefer the NGINX web server, see the [LEMP stack](#) tutorial. To see the LAMP server in action, you can optionally install and configure a WordPress site. In this tutorial you learn how to:

- Create an Ubuntu VM (the 'L' in the LAMP stack)
- Open port 80 for web traffic
- Install Apache, MySQL, and PHP
- Verify installation and configuration
- Install WordPress on the LAMP server

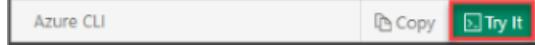
This setup is for quick tests or proof of concept. For more on the LAMP stack, including recommendations for a production environment, see the [Ubuntu documentation](#).

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a resource group

Create a resource group with the `az group create` command. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Create a virtual machine

Create a VM with the [az vm create](#) command.

The following example creates a VM named *myVM* and creates SSH keys if they do not already exist in a default key location. To use a specific set of keys, use the `--ssh-key-value` option. The command also sets *azureuser* as an administrator user name. You use this name later to connect to the VM.

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

When the VM has been created, the Azure CLI shows information similar to the following example. Take note of the `publicIpAddress`. This address is used to access the VM in later steps.

```
{
  "fqdns": "",
  "id": "/subscriptions/<subscription
ID>/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "40.68.254.142",
  "resourceGroup": "myResourceGroup"
}
```

Open port 80 for web traffic

By default, only SSH connections are allowed into Linux VMs deployed in Azure. Because this VM is going to be a web server, you need to open port 80 from the internet. Use the [az vm open-port](#) command to open the desired port.

```
az vm open-port --port 80 --resource-group myResourceGroup --name myVM
```

SSH into your VM

If you don't already know the public IP address of your VM, run the [az network public-ip list](#) command. You need this IP address for several later steps.

```
az network public-ip list --resource-group myResourceGroup --query [].ipAddress
```

Use the following command to create an SSH session with the virtual machine. Substitute the correct public IP address of your virtual machine. In this example, the IP address is *40.68.254.142*. *azureuser* is the administrator user name set when you created the VM.

```
ssh azureuser@40.68.254.142
```

Install Apache, MySQL, and PHP

Run the following command to update Ubuntu package sources and install Apache, MySQL, and PHP. Note the caret (^) at the end of the command, which is part of the `lamp-server^` package name.

```
sudo apt update && sudo apt install lamp-server^
```

You are prompted to install the packages and other dependencies. This process installs the minimum required PHP extensions needed to use PHP with MySQL.

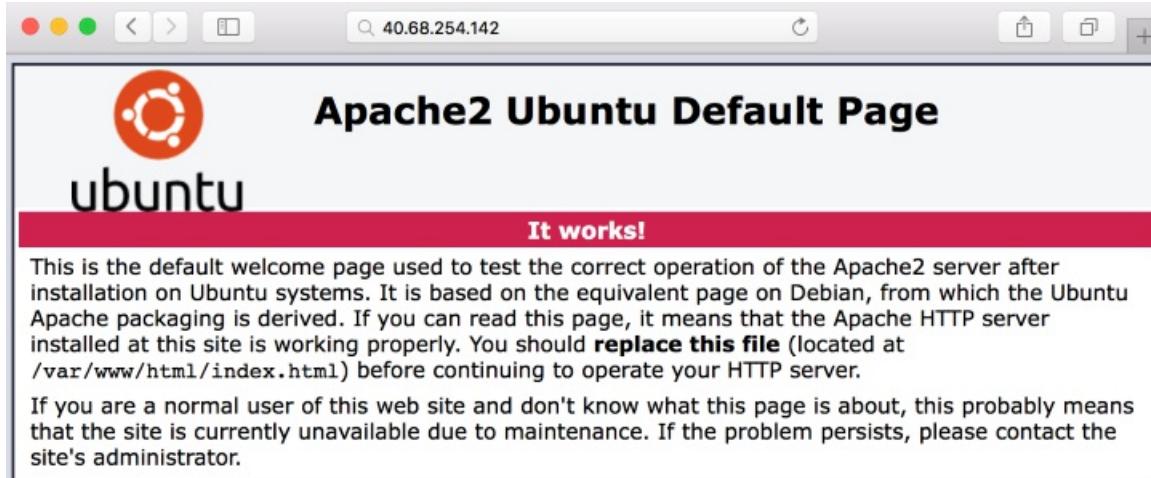
Verify installation and configuration

Verify Apache

Check the version of Apache with the following command:

```
apache2 -v
```

With Apache installed, and port 80 open to your VM, the web server can now be accessed from the internet. To view the Apache2 Ubuntu Default Page, open a web browser, and enter the public IP address of the VM. Use the public IP address you used to SSH to the VM:



Verify and secure MySQL

Check the version of MySQL with the following command (note the capital `V` parameter):

```
mysql -V
```

To help secure the installation of MySQL, including setting a root password, run the `mysql_secure_installation` script.

```
sudo mysql_secure_installation
```

You can optionally set up the Validate Password Plugin (recommended). Then, set a password for the MySQL root user, and configure the remaining security settings for your environment. We recommend that you answer "Y" (yes) to all questions.

If you want to try MySQL features (create a MySQL database, add users, or change configuration settings), login to MySQL. This step is not required to complete this tutorial.

```
sudo mysql -u root -p
```

When done, exit the mysql prompt by typing `\q`.

Verify PHP

Check the version of PHP with the following command:

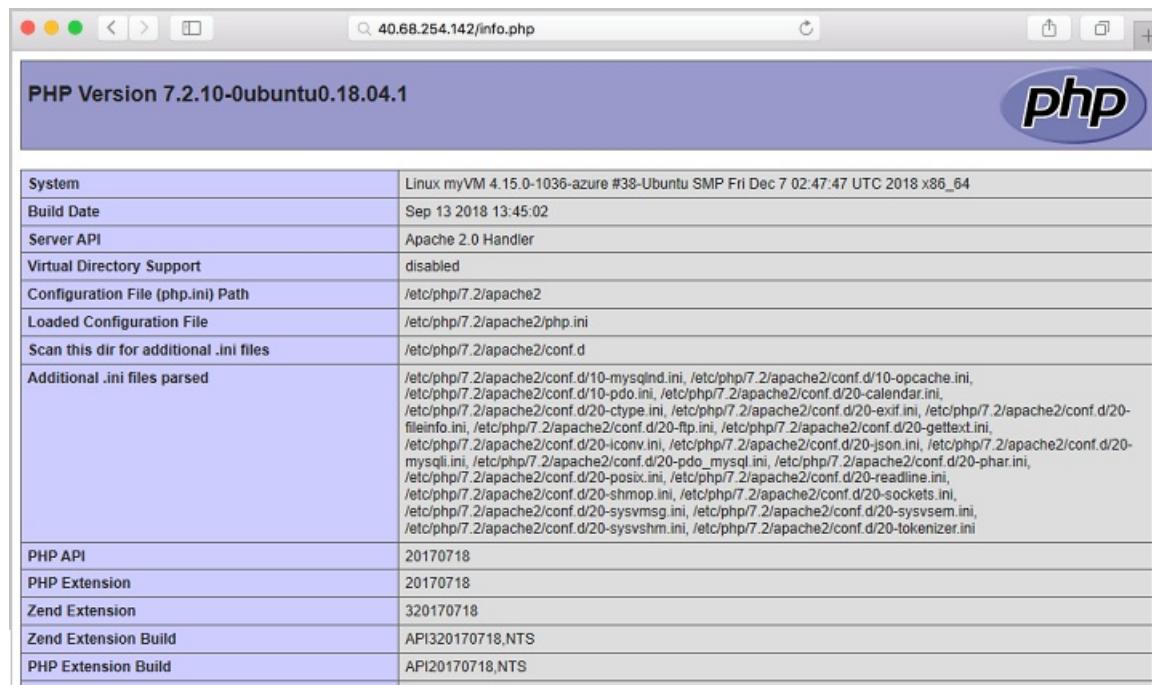
```
php -v
```

If you want to test further, create a quick PHP info page to view in a browser. The following command creates the PHP info page:

```
sudo sh -c 'echo "<?php phpinfo(); ?>" > /var/www/html/info.php'
```

Now you can check the PHP info page you created. Open a browser and go to

`http://yourPublicIPAddress/info.php`. Substitute the public IP address of your VM. It should look similar to this image.



Install WordPress

If you want to try your stack, install a sample app. As an example, the following steps install the open source [WordPress](#) platform to create websites and blogs. Other workloads to try include [Drupal](#) and [Moodle](#).

This WordPress setup is only for proof of concept. To install the latest WordPress in production with recommended security settings, see the [WordPress documentation](#).

Install the WordPress package

Run the following command:

```
sudo apt install wordpress
```

Configure WordPress

Configure WordPress to use MySQL and PHP.

In a working directory, create a text file `wordpress.sql` to configure the MySQL database for WordPress:

```
sudo sensible-editor wordpress.sql
```

Add the following commands, substituting a database password of your choice for *yourPassword* (leave other values unchanged). If you previously set up a MySQL security policy to validate password strength, make sure the password meets the strength requirements. Save the file.

```
CREATE DATABASE wordpress;
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
ON wordpress.*
TO wordpress@localhost
IDENTIFIED BY 'yourPassword';
FLUSH PRIVILEGES;
```

Run the following command to create the database:

```
cat wordpress.sql | sudo mysql --defaults-extra-file=/etc/mysql/debian.cnf
```

Because the file `wordpress.sql` contains database credentials, delete it after use:

```
sudo rm wordpress.sql
```

To configure PHP, run the following command to open a text editor of your choice and create the file `/etc/wordpress/config-localhost.php`:

```
sudo sensible-editor /etc/wordpress/config-localhost.php
```

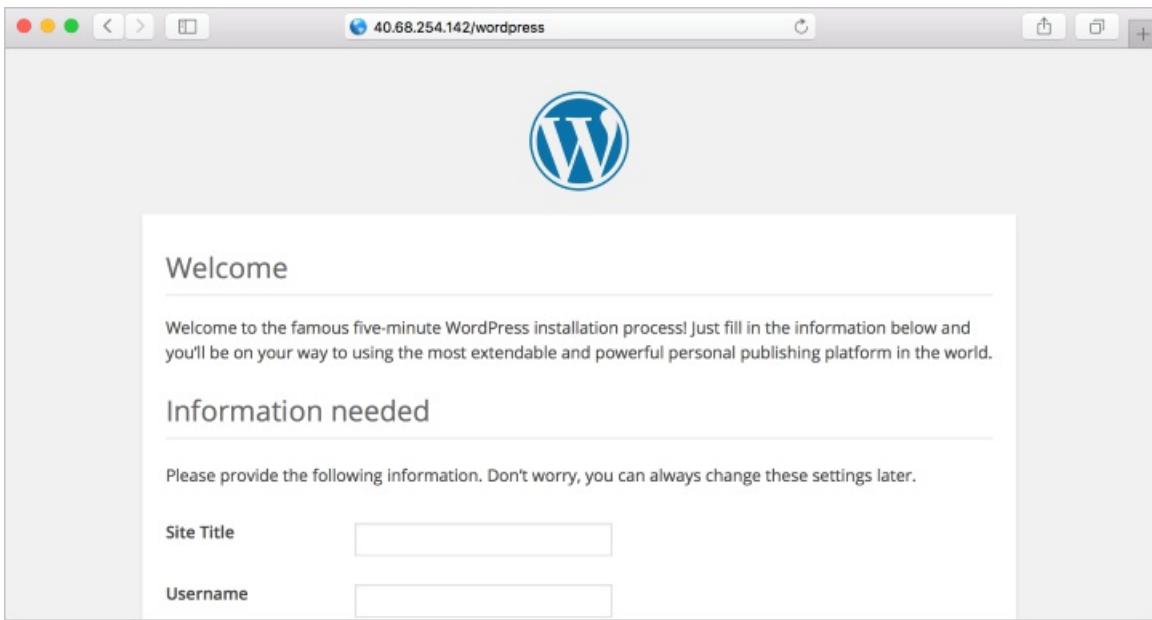
Copy the following lines to the file, substituting your WordPress database password for *yourPassword* (leave other values unchanged). Then save the file.

```
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'yourPassword');
define('DB_HOST', 'localhost');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');
?>
```

Move the WordPress installation to the web server document root:

```
sudo ln -s /usr/share/wordpress /var/www/html/wordpress
sudo mv /etc/wordpress/config-localhost.php /etc/wordpress/config-default.php
```

Now you can complete the WordPress setup and publish on the platform. Open a browser and go to `http://yourPublicIPAddress/wordpress`. Substitute the public IP address of your VM. It should look similar to this image.



Next steps

In this tutorial, you deployed a LAMP server in Azure. You learned how to:

- Create an Ubuntu VM
- Open port 80 for web traffic
- Install Apache, MySQL, and PHP
- Verify installation and configuration
- Install WordPress on the LAMP server

Advance to the next tutorial to learn how to secure web servers with SSL certificates.

[Secure web server with SSL](#)

Tutorial: Install a LEMP web server on a Linux virtual machine in Azure

1/31/2019 • 7 minutes to read • [Edit Online](#)

This article walks you through how to deploy an NGINX web server, MySQL, and PHP (the LEMP stack) on an Ubuntu VM in Azure. The LEMP stack is an alternative to the popular [LAMP stack](#), which you can also install in Azure. To see the LEMP server in action, you can optionally install and configure a WordPress site. In this tutorial you learn how to:

- Create an Ubuntu VM (the 'L' in the LEMP stack)
- Open port 80 for web traffic
- Install NGINX, MySQL, and PHP
- Verify installation and configuration
- Install WordPress on the LEMP server

This setup is for quick tests or proof of concept.

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a resource group

Create a resource group with the `az group create` command. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Create a virtual machine

Create a VM with the [az vm create](#) command.

The following example creates a VM named *myVM* and creates SSH keys if they do not already exist in a default key location. To use a specific set of keys, use the `--ssh-key-value` option. The command also sets *azureuser* as an administrator user name. You use this name later to connect to the VM.

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

When the VM has been created, the Azure CLI shows information similar to the following example. Take note of the `publicIpAddress`. This address is used to access the VM in later steps.

```
{
  "fqdns": "",
  "id": "/subscriptions/<subscription
ID>/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "40.68.254.142",
  "resourceGroup": "myResourceGroup"
}
```

Open port 80 for web traffic

By default, only SSH connections are allowed into Linux VMs deployed in Azure. Because this VM is going to be a web server, you need to open port 80 from the internet. Use the [az vm open-port](#) command to open the desired port.

```
az vm open-port --port 80 --resource-group myResourceGroup --name myVM
```

SSH into your VM

If you don't already know the public IP address of your VM, run the [az network public-ip list](#) command. You need this IP address for several later steps.

```
az network public-ip list --resource-group myResourceGroup --query [].ipAddress
```

Use the following command to create an SSH session with the virtual machine. Substitute the correct public IP address of your virtual machine. In this example, the IP address is *40.68.254.142*. *azureuser* is the administrator user name set when you created the VM.

```
ssh azureuser@40.68.254.142
```

Install NGINX, MySQL, and PHP

Run the following command to update Ubuntu package sources and install NGINX, MySQL, and PHP.

```
sudo apt update && sudo apt install nginx && sudo apt install mysql-server php-mysql php-fpm
```

You are prompted to install the packages and other dependencies. This process installs the minimum required PHP extensions needed to use PHP with MySQL.

Verify installation and configuration

Verify NGINX

Check the version of NGINX with the following command:

```
nginx -v
```

With NGINX installed, and port 80 open to your VM, the web server can now be accessed from the internet. To view the NGINX welcome page, open a web browser, and enter the public IP address of the VM. Use the public IP address you used to SSH to the VM:



Verify and secure MySQL

Check the version of MySQL with the following command (note the capital `V` parameter):

```
mysql -V
```

To help secure the installation of MySQL, including setting a root password, run the `mysql_secure_installation` script.

```
sudo mysql_secure_installation
```

You can optionally set up the Validate Password Plugin (recommended). Then, set a password for the MySQL root user, and configure the remaining security settings for your environment. We recommend that you answer "Y" (yes) to all questions.

If you want to try MySQL features (create a MySQL database, add users, or change configuration settings), login to MySQL. This step is not required to complete this tutorial.

```
sudo mysql -u root -p
```

When done, exit the mysql prompt by typing `\q`.

Verify PHP

Check the version of PHP with the following command:

```
php -v
```

Configure NGINX to use the PHP FastCGI Process Manager (PHP-FPM). Run the following commands to back up the original NGINX server block config file and then edit the original file in an editor of your choice:

```
sudo cp /etc/nginx/sites-available/default /etc/nginx/sites-available/default_backup  
sudo sensible-editor /etc/nginx/sites-available/default
```

In the editor, replace the contents of `/etc/nginx/sites-available/default` with the following. See the comments for explanation of the settings. Substitute the public IP address of your VM for `yourPublicIPAddress`, confirm the PHP version in `fastcgi_pass`, and leave the remaining settings. Then save the file.

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    root /var/www/html;  
    # Homepage of website is index.php  
    index index.php;  
  
    server_name yourPublicIPAddress;  
  
    location / {  
        try_files $uri $uri/ =404;  
    }  
  
    # Include FastCGI configuration for NGINX  
    location ~ \.php$ {  
        include snippets/fastcgi-php.conf;  
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;  
    }  
}
```

Check the NGINX configuration for syntax errors:

```
sudo nginx -t
```

If the syntax is correct, restart NGINX with the following command:

```
sudo service nginx restart
```

If you want to test further, create a quick PHP info page to view in a browser. The following command creates the PHP info page:

```
sudo sh -c 'echo "<?php phpinfo(); ?>" > /var/www/html/info.php'
```

Now you can check the PHP info page you created. Open a browser and go to

<http://yourPublicIPAddress/info.php>. Substitute the public IP address of your VM. It should look similar to this image.

PHP Version 7.2.10-0ubuntu0.18.04.1	
System	Linux myVM 4.15.0-1036-azure #38-Ubuntu SMP Fri Dec 7 02:47:47 UTC 2018 x86_64
Build Date	Sep 13 2018 13:45:02
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlind.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-finfo.info, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysqli.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS

Install WordPress

If you want to try your stack, install a sample app. As an example, the following steps install the open source [WordPress](#) platform to create websites and blogs. Other workloads to try include [Drupal](#) and [Moodle](#).

This WordPress setup is only for proof of concept. To install the latest WordPress in production with recommended security settings, see the [WordPress documentation](#).

Install the WordPress package

Run the following command:

```
sudo apt install wordpress
```

Configure WordPress

Configure WordPress to use MySQL and PHP.

In a working directory, create a text file `wordpress.sql` to configure the MySQL database for WordPress:

```
sudo sensible-editor wordpress.sql
```

Add the following commands, substituting a database password of your choice for `yourPassword` (leave other values unchanged). If you previously set up a MySQL security policy to validate password strength, make sure the password meets the strength requirements. Save the file.

```
CREATE DATABASE wordpress;
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
ON wordpress.*  
TO wordpress@localhost
IDENTIFIED BY 'yourPassword';
FLUSH PRIVILEGES;
```

Run the following command to create the database:

```
cat wordpress.sql | sudo mysql --defaults-extra-file=/etc/mysql/debian.cnf
```

Because the file `wordpress.sql` contains database credentials, delete it after use:

```
sudo rm wordpress.sql
```

To configure PHP, run the following command to open a text editor of your choice and create the file `/etc/wordpress/config-localhost.php`:

```
sudo sensible-editor /etc/wordpress/config-localhost.php
```

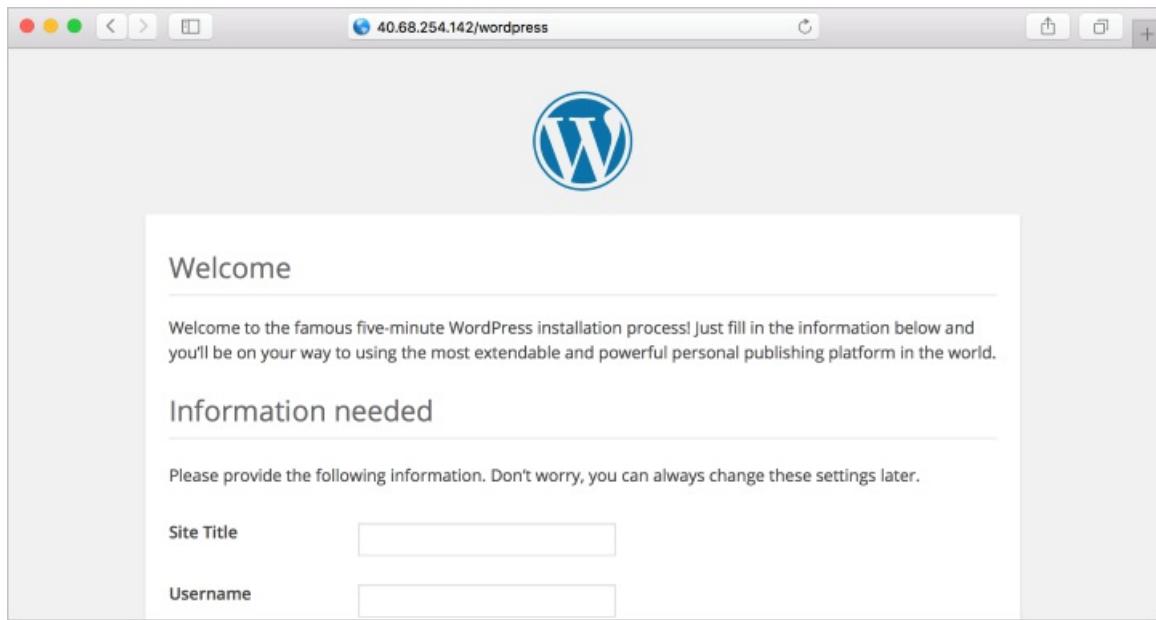
Copy the following lines to the file, substituting your WordPress database password for `yourPassword` (leave other values unchanged). Then save the file.

```
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'yourPassword');
define('DB_HOST', 'localhost');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');
?>
```

Move the WordPress installation to the web server document root:

```
sudo ln -s /usr/share/wordpress /var/www/html/wordpress
sudo mv /etc/wordpress/config-localhost.php /etc/wordpress/config-default.php
```

Now you can complete the WordPress setup and publish on the platform. Open a browser and go to `http://yourPublicIPAddress/wordpress`. Substitute the public IP address of your VM. It should look similar to this image.



Next steps

In this tutorial, you deployed a LEMP server in Azure. You learned how to:

- Create an Ubuntu VM
- Open port 80 for web traffic
- Install NGINX, MySQL, and PHP
- Verify installation and configuration
- Install WordPress on the LEMP stack

Advance to the next tutorial to learn how to secure web servers with SSL certificates.

[Secure web server with SSL](#)

Tutorial: Create a MongoDB, Express, AngularJS, and Node.js (MEAN) stack on a Linux virtual machine in Azure

3/21/2019 • 7 minutes to read • [Edit Online](#)

This tutorial shows you how to implement a MongoDB, Express, AngularJS, and Nodejs (MEAN) stack on a Linux virtual machine (VM) in Azure. The MEAN stack that you create enables adding, deleting, and listing books in a database. You learn how to:

- Create a Linux VM
- Install Nodejs
- Install MongoDB and set up the server
- Install Express and set up routes to the server
- Access the routes with AngularJS
- Run the application

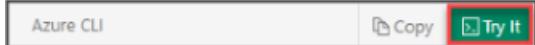
Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `Powershell` shells to run a variety of tools to work with Azure services.

Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a Linux VM

Create a resource group with the `az group create` command and create a Linux VM with the `az vm create` command. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example uses the Azure CLI to create a resource group named `myResourceGroupMEAN` in the `eastus` location. A VM is created named `myVM` with SSH keys if they do not already exist in a default key location.

To use a specific set of keys, use the `--ssh-key-value` option.

```
az group create --name myResourceGroupMEAN --location eastus
az vm create \
    --resource-group myResourceGroupMEAN \
    --name myVM \
    --image UbuntuLTS \
    --admin-username azureuser \
    --admin-password 'Azure12345678!' \
    --generate-ssh-keys
az vm open-port --port 3300 --resource-group myResourceGroupMEAN --name myVM
```

When the VM has been created, the Azure CLI shows information similar to the following example:

```
{
  "fqdns": "",
  "id": "/subscriptions/{subscription-
id}/resourceGroups/myResourceGroupMEAN/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "13.72.77.9",
  "resourceGroup": "myResourceGroupMEAN"
}
```

Take note of the `publicIpAddress`. This address is used to access the VM.

Use the following command to create an SSH session with the VM. Make sure to use the correct public IP address. In our example above our IP address was 13.72.77.9.

```
ssh azureuser@13.72.77.9
```

Install Node.js

[Node.js](#) is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js is used in this tutorial to set up the Express routes and AngularJS controllers.

On the VM, using the bash shell that you opened with SSH, install Node.js.

```
sudo apt-get install -y nodejs
```

Install MongoDB and set up the server

[MongoDB](#) stores data in flexible, JSON-like documents. Fields in a database can vary from document to document and data structure can be changed over time. For our example application, we are adding book records to MongoDB that contain book name, isbn number, author, and number of pages.

1. On the VM, using the bash shell that you opened with SSH, set the MongoDB key.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
0C49F3730359A14518585931BC711F9BA15703C6
echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4 multiverse" | sudo
tee /etc/apt/sources.list.d/mongodb-org-3.4.list
```

2. Update the package manager with the key.

```
sudo apt-get update
```

3. Install MongoDB.

```
sudo apt-get install -y mongodb
```

4. Start the server.

```
sudo service mongodb start
```

5. We also need to install the [body-parser](#) package to help us process the JSON passed in requests to the server.

Install the npm package manager.

```
sudo apt-get install npm
```

Install the body parser package.

```
sudo npm install body-parser
```

6. Create a folder named *Books* and add a file to it named *server.js* that contains the configuration for the web server.

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(express.static(__dirname + '/public'));
app.use(bodyParser.json());
require('./apps/routes')(app);
app.set('port', 3300);
app.listen(app.get('port'), function() {
  console.log('Server up: http://localhost:' + app.get('port'));
});
```

Install Express and set up routes to the server

[Express](#) is a minimal and flexible Node.js web application framework that provides features for web and mobile applications. Express is used in this tutorial to pass book information to and from our MongoDB database.

[Mongoose](#) provides a straight-forward, schema-based solution to model your application data. Mongoose is used in this tutorial to provide a book schema for the database.

1. Install Express and Mongoose.

```
sudo npm install express mongoose
```

2. In the *Books* folder, create a folder named *apps* and add a file named *routes.js* with the express routes defined.

```

var Book = require('./models/book');
module.exports = function(app) {
  app.get('/book', function(req, res) {
    Book.find({}, function(err, result) {
      if (err) throw err;
      res.json(result);
    });
  });
  app.post('/book', function(req, res) {
    var book = new Book({
      name: req.body.name,
      isbn: req.body.isbn,
      author: req.body.author,
      pages: req.body.pages
    });
    book.save(function(err, result) {
      if (err) throw err;
      res.json({
        message: "Successfully added book",
        book: result
      });
    });
  });
  app.delete("/book/:isbn", function(req, res) {
    Book.findOneAndRemove(req.query, function(err, result) {
      if (err) throw err;
      res.json({
        message: "Successfully deleted the book",
        book: result
      });
    });
  });
  var path = require('path');
  app.get('*', function(req, res) {
    res.sendfile(path.join(__dirname + '/public', 'index.html'));
  });
};

```

- In the `apps` folder, create a folder named `models` and add a file named `book.js` with the book model configuration defined.

```

var mongoose = require('mongoose');
var dbHost = 'mongodb://localhost:27017/test';
mongoose.connect(dbHost);
mongoose.connection;
mongoose.set('debug', true);
var bookSchema = mongoose.Schema({
  name: String,
  isbn: {type: String, index: true},
  author: String,
  pages: Number
});
var Book = mongoose.model('Book', bookSchema);
module.exports = mongoose.model('Book', bookSchema);

```

Access the routes with AngularJS

[AngularJS](#) provides a web framework for creating dynamic views in your web applications. In this tutorial, we use AngularJS to connect our web page with Express and perform actions on our book database.

- Change the directory back up to `Books` (`cd ../../`), and then create a folder named `public` and add a file named `script.js` with the controller configuration defined.

```

var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
  $http({
    method: 'GET',
    url: '/book'
  }).then(function successCallback(response) {
    $scope.books = response.data;
  }, function errorCallback(response) {
    console.log('Error: ' + response);
  });
  $scope.del_book = function(book) {
    $http({
      method: 'DELETE',
      url: '/book/:isbn',
      params: {'isbn': book.isbn}
    }).then(function successCallback(response) {
      console.log(response);
    }, function errorCallback(response) {
      console.log('Error: ' + response);
    });
  };
  $scope.add_book = function() {
    var body = '{ "name": "' + $scope.Name +
    '", "isbn": "' + $scope.Isbn +
    '", "author": "' + $scope.Author +
    '", "pages": "' + $scope.Pages + '" }';
    $http({
      method: 'POST',
      url: '/book',
      data: body
    }).then(function successCallback(response) {
      console.log(response);
    }, function errorCallback(response) {
      console.log('Error: ' + response);
    });
  };
});

```

2. In the *public* folder, create a file named *index.html* with the web page defined.

```

<!doctype html>
<html ng-app="myApp" ng-controller="myCtrl">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
    <script src="script.js"></script>
  </head>
  <body>
    <div>
      <table>
        <tr>
          <td>Name:</td>
          <td><input type="text" ng-model="Name"></td>
        </tr>
        <tr>
          <td>Isbn:</td>
          <td><input type="text" ng-model="Isbn"></td>
        </tr>
        <tr>
          <td>Author:</td>
          <td><input type="text" ng-model="Author"></td>
        </tr>
        <tr>
          <td>Pages:</td>
          <td><input type="number" ng-model="Pages"></td>
        </tr>
      </table>
      <button ng-click="add_book()">Add</button>
    </div>
    <hr>
    <div>
      <table>
        <tr>
          <th>Name</th>
          <th>Isbn</th>
          <th>Author</th>
          <th>Pages</th>
        </tr>
        <tr ng-repeat="book in books">
          <td><input type="button" value="Delete" data-ng-click="del_book(book)"></td>
          <td>{{book.name}}</td>
          <td>{{book.isbn}}</td>
          <td>{{book.author}}</td>
          <td>{{book.pages}}</td>
        </tr>
      </table>
    </div>
  </body>
</html>

```

Run the application

1. Change the directory back up to *Books* (`cd ..`) and start the server by running this command:

```
nodejs server.js
```

2. Open a web browser to the address that you recorded for the VM. For example, `http://13.72.77.9:3300`. You should see something like the following page:

The screenshot shows a web browser window with the URL `13.72.77.9:3300`. The page contains a form for adding a book record. The fields are labeled 'Name', 'Isbn', 'Author', and 'Pages', each with a corresponding input box. Below the form is a table header with columns 'Name', 'Isbn', 'Author', and 'Pages'. A single row of data is shown below the header.

Name	Isbn	Author	Pages
MyBook	000001	Me	200

3. Enter data into the textboxes and click **Add**. For example:

The screenshot shows the same web browser window after data has been entered into the form. The 'Name' field now contains 'MyBook', 'Isbn' contains '000001', 'Author' contains 'Me', and 'Pages' contains '200'. The 'Add' button is visible at the bottom of the form. The table below the form now displays the entered data.

Name	Isbn	Author	Pages
MyBook	000001	Me	200

4. After refreshing the page, you should see something like this page:

The screenshot shows the browser after a refresh. The form is empty again. Below it, a table lists the single book record that was added previously. The 'Delete' button is visible next to the record.

Name	Isbn	Author	Pages
MyBook	000001	Me	200

5. You could click **Delete** and remove the book record from the database.

Next steps

In this tutorial, you created a web application that keeps track of book records using a MEAN stack on a Linux VM. You learned how to:

- Create a Linux VM
- Install Node.js
- Install MongoDB and set up the server
- Install Express and set up routes to the server
- Access the routes with AngularJS
- Run the application

Advance to the next tutorial to learn how to secure web servers with SSL certificates.

[Secure web server with SSL](#)

Tutorial: Secure a web server on a Linux virtual machine in Azure with SSL certificates stored in Key Vault

4/11/2019 • 5 minutes to read • [Edit Online](#)

To secure web servers, a Secure Sockets Layer (SSL) certificate can be used to encrypt web traffic. These SSL certificates can be stored in Azure Key Vault, and allow secure deployments of certificates to Linux virtual machines (VMs) in Azure. In this tutorial you learn how to:

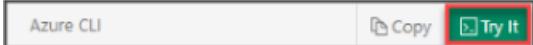
- Create an Azure Key Vault
- Generate or upload a certificate to the Key Vault
- Create a VM and install the NGINX web server
- Inject the certificate into the VM and configure NGINX with an SSL binding

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Overview

Azure Key Vault safeguards cryptographic keys and secrets, such as certificates or passwords. Key Vault helps streamline the certificate management process and enables you to maintain control of keys that access those certificates. You can create a self-signed certificate inside Key Vault, or upload an existing, trusted certificate that you already own.

Rather than using a custom VM image that includes certificates baked-in, you inject certificates into a running VM. This process ensures that the most up-to-date certificates are installed on a web server during deployment. If

you renew or replace a certificate, you don't also have to create a new custom VM image. The latest certificates are automatically injected as you create additional VMs. During the whole process, the certificates never leave the Azure platform or are exposed in a script, command-line history, or template.

Create an Azure Key Vault

Before you can create a Key Vault and certificates, create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroupSecureWeb* in the *eastus* location:

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

Next, create a Key Vault with [az keyvault create](#) and enable it for use when you deploy a VM. Each Key Vault requires a unique name, and should be all lowercase. Replace *<mykeyvault>* in the following example with your own unique Key Vault name:

```
keyvault_name=<mykeyvault>
az keyvault create \
    --resource-group myResourceGroupSecureWeb \
    --name $keyvault_name \
    --enabled-for-deployment
```

Generate a certificate and store in Key Vault

For production use, you should import a valid certificate signed by trusted provider with [az keyvault certificate import](#). For this tutorial, the following example shows how you can generate a self-signed certificate with [az keyvault certificate create](#) that uses the default certificate policy:

```
az keyvault certificate create \
    --vault-name $keyvault_name \
    --name mycert \
    --policy "$(az keyvault certificate get-default-policy)"
```

Prepare a certificate for use with a VM

To use the certificate during the VM create process, obtain the ID of your certificate with [az keyvault secret list-versions](#). Convert the certificate with [az vm secret format](#). The following example assigns the output of these commands to variables for ease of use in the next steps:

```
secret=$(az keyvault secret list-versions \
    --vault-name $keyvault_name \
    --name mycert \
    --query "[?attributes.enabled].id" --output tsv)
vm_secret=$(az vm secret format --secrets "$secret")
```

Create a cloud-init config to secure NGINX

[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.

When you create a VM, certificates and keys are stored in the protected */var/lib/waagent/* directory. To automate adding the certificate to the VM and configuring the web server, use cloud-init. In this example, you install and configure the NGINX web server. You can use the same process to install and configure Apache.

Create a file named *cloud-init-web-server.txt* and paste the following configuration:

```
#cloud-config
package_upgrade: true
packages:
- nginx
write_files:
- owner: www-data:www-data
- path: /etc/nginx/sites-available/default
content: |
  server {
    listen 443 ssl;
    ssl_certificate /etc/nginx/ssl/mycert.cert;
    ssl_certificate_key /etc/nginx/ssl/mycert.prv;
  }
runcmd:
- secretsname=$(find /var/lib/waagent/ -name "*.prv" | cut -c -57)
- mkdir /etc/nginx/ssl
- cp $secretsname.crt /etc/nginx/ssl/mycert.cert
- cp $secretsname.prv /etc/nginx/ssl/mycert.prv
- service nginx restart
```

Create a secure VM

Now create a VM with [az vm create](#). The certificate data is injected from Key Vault with the `--secrets` parameter. You pass in the cloud-init config with the `--custom-data` parameter:

```
az vm create \
--resource-group myResourceGroupSecureWeb \
--name myVM \
--image UbuntuLTS \
--admin-username azureuser \
--generate-ssh-keys \
--custom-data cloud-init-web-server.txt \
--secrets "$vm_secret"
```

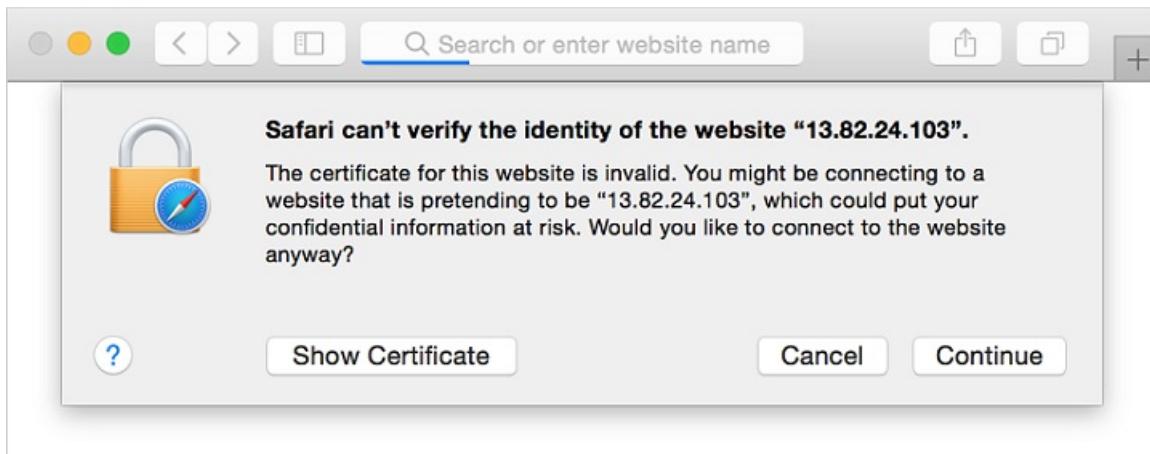
It takes a few minutes for the VM to be created, the packages to install, and the app to start. When the VM has been created, take note of the `publicIpAddress` displayed by the Azure CLI. This address is used to access your site in a web browser.

To allow secure web traffic to reach your VM, open port 443 from the Internet with [az vm open-port](#):

```
az vm open-port \
--resource-group myResourceGroupSecureWeb \
--name myVM \
--port 443
```

Test the secure web app

Now you can open a web browser and enter `https://<publicIpAddress>` in the address bar. Provide your own public IP address from the VM create process. Accept the security warning if you used a self-signed certificate:



Your secured NGINX site is then displayed as in the following example:



Next steps

In this tutorial, you secured an NGINX web server with an SSL certificate stored in Azure Key Vault. You learned how to:

- Create an Azure Key Vault
- Generate or upload a certificate to the Key Vault
- Create a VM and install the NGINX web server
- Inject the certificate into the VM and configure NGINX with an SSL binding

Follow this link to see pre-built virtual machine script samples.

[Linux virtual machine script samples](#)

Azure CLI Samples for Linux virtual machines

4/3/2019 • 2 minutes to read • [Edit Online](#)

The following table includes links to bash scripts built using the Azure CLI.

Create virtual machines	
Create a virtual machine	Creates a Linux virtual machine with minimal configuration.
Create a fully configured virtual machine	Creates a resource group, virtual machine, and all related resources.
Create highly available virtual machines	Creates several virtual machines in a highly available and load balanced configuration.
Create a VM and run configuration script	Creates a virtual machine and uses the Azure Custom Script extension to install NGINX.
Create a VM with WordPress installed	Creates a virtual machine and uses the Azure Custom Script extension to install WordPress.
Create a VM from a managed OS disk	Creates a virtual machine by attaching an existing Managed Disk as OS disk.
Create a VM from a snapshot	Creates a virtual machine from a snapshot by first creating a managed disk from snapshot and then attaching the new managed disk as OS disk.
Manage storage	
Create managed disk from a VHD	Creates a managed disk from a specialized VHD as an OS disk or from a data VHD as data disk.
Create a managed disk from a snapshot	Creates a managed disk from a snapshot.
Copy managed disk to same or different subscription	Copies managed disk to same or different subscription but in the same region as the parent managed disk.
Export a snapshot as VHD to a storage account	Exports a managed snapshot as VHD to a storage account in different region.
Export the VHD of a managed disk to a storage account	Exports the underlying VHD of a managed disk to a storage account in different region.
Copy snapshot to same or different subscription	Copies snapshot to same or different subscription but in the same region as the parent snapshot.
Network virtual machines	

Secure network traffic between virtual machines	Creates two virtual machines, all related resources, and an internal and external network security groups (NSG).
Secure virtual machines	
Encrypt a VM and data disks	Creates an Azure Key Vault, encryption key, and service principal, then encrypts a VM.
Monitor virtual machines	
Monitor a VM with Azure Monitor logs	Creates a virtual machine, installs the Log Analytics agent, and enrolls the VM in an Log Analytics workspace.
Troubleshoot virtual machines	
Troubleshoot a VMs operating system disk	Mounts the operating system disk from one VM as a data disk on a second VM.

Azure Virtual Machine PowerShell samples

3/6/2019 • 2 minutes to read • [Edit Online](#)

The following table includes links to PowerShell scripts samples that create and manage Linux virtual machines.

Create virtual machines	
Create a fully configured virtual machine	Creates a resource group, virtual machine, and all related resources.
Create a VM with Docker enabled	Creates a virtual machine, configures this VM as a Docker host, and runs an NGINX container.
Create a VM and run configuration script	Creates a virtual machine and uses the Azure Custom Script extension to install NGINX.
Create a VM with WordPress installed	Creates a virtual machine and uses the Azure Custom Script extension to install WordPress.
Create a VM from a managed OS disk	Creates a virtual machine by attaching an existing Managed Disk as OS disk.
Create a VM from a snapshot	Creates a virtual machine from a snapshot by first creating a managed disk from the snapshot and then attaching the new managed disk as OS disk.
Manage storage	
Create a managed disk from a VHD in the same or a different subscription	Creates a managed disk from a specialized VHD as an OS disk, or from a data VHD as a data disk, in the same or a different subscription.
Create a managed disk from a snapshot	Creates a managed disk from a snapshot.
Export a snapshot as a VHD to a storage account	Exports a managed snapshot as a VHD to a storage account in a different region.
Export the VHD of a managed disk to a storage account	Exports the underlying VHD of a managed disk to a storage account in a different region.
Create a snapshot from a VHD	Creates a snapshot from a VHD and then uses that snapshot to create multiple identical managed disks quickly.
Copy a snapshot to the same or a different subscription	Copies snapshot to the same or a different subscription that is in the same region as the parent snapshot.
Monitor virtual machines	
Monitor a VM with Azure Monitor logs	Creates a virtual machine, installs the Log Analytics agent, and enrolls the VM in a Log Analytics workspace.

Copy a managed disk to the same or a different subscription	Copies a managed disk to the same or a different subscription that is in the same region as the parent managed disk.

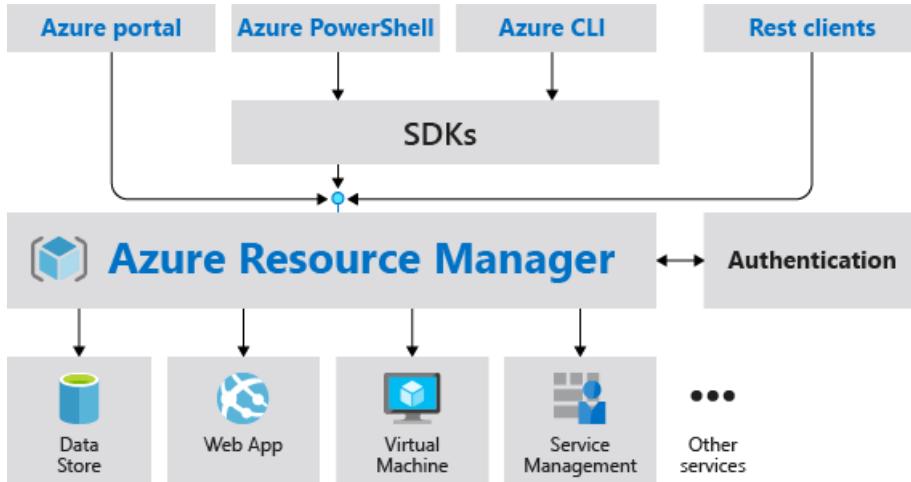
Azure Resource Manager overview

6/18/2019 • 12 minutes to read • [Edit Online](#)

Azure Resource Manager is the deployment and management service for Azure. It provides a consistent management layer that enables you to create, update, and delete resources in your Azure subscription. You can use its access control, auditing, and tagging features to secure and organize your resources after deployment.

When you take actions through the portal, PowerShell, Azure CLI, REST APIs, or client SDKs, the Azure Resource Manager API handles your request. Because all requests are handled through the same API, you see consistent results and capabilities in all the different tools. 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.

The following image shows how all the tools interact with the Azure Resource Manager API. The API passes requests to the Resource Manager service, which authenticates and authorizes the requests. Resource Manager then routes the requests to the appropriate service.



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 how to allocate resources to resource groups 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](#).
- **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](#).
- **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.

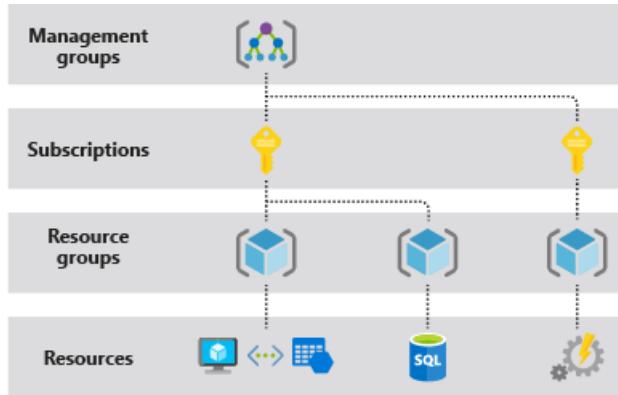
The benefits of using Resource Manager

Resource Manager provides several benefits:

- You can deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.
- You can repeatedly deploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.
- You can manage your infrastructure through declarative templates rather than scripts.
- You can define the dependencies between resources so they're deployed in the correct order.
- You can apply access control to all services in your resource group because Role-Based Access Control (RBAC) is natively integrated into the management platform.
- You can apply tags to resources to logically organize all the resources in your subscription.
- You can 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.

Guidance

The following suggestions help you take full advantage of Resource Manager when working with your solutions.

- Define and deploy your infrastructure through the declarative syntax in Resource Manager templates, rather than through imperative commands.
- Define all deployment and configuration steps in the template. You should have no manual steps for setting up your solution.
- Run imperative commands to manage your resources, such as to start or stop an app or machine.
- Arrange resources with the same lifecycle in a resource group. Use tags for all other organizing of resources.

For guidance on how enterprises can use Resource Manager to effectively manage subscriptions, see [Azure enterprise scaffold - prescriptive subscription governance](#).

For recommendations on creating Resource Manager templates, see [Azure Resource Manager template best](#)

practices.

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. Therefore, 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](#).

Resource providers

Each resource provider offers a set of resources and operations for working with those resources. For example, if you want to store keys and secrets, you work with the **Microsoft.KeyVault** resource provider. This resource provider offers a resource type called **vaults** for creating the key vault.

The name of a resource type is in the format: **{resource-provider}/{resource-type}**. The resource type for a key vault is **Microsoft.KeyVault/vaults**.

Before getting started with deploying your resources, you should gain an understanding of the available resource providers. Knowing the names of resource providers and resources helps you define resources you want to deploy to Azure. Also, you need to know the valid locations and API versions for each resource type. For more information, see [Resource providers and types](#).

For all the operations offered by resource providers, see the [Azure REST APIs](#).

Template deployment

With Resource Manager, you can create a template (in JSON format) that defines the infrastructure and configuration of your Azure solution. By using a template, you can repeatedly deploy your solution throughout its lifecycle and have confidence your resources are deployed in a consistent state.

To learn about the format of the template and how you construct it, see [Understand the structure and syntax of Azure Resource Manager Templates](#). To view the JSON syntax for resources types, see [Define resources in Azure Resource Manager templates](#).

Resource Manager processes the template like any other request. It parses the template and converts its syntax

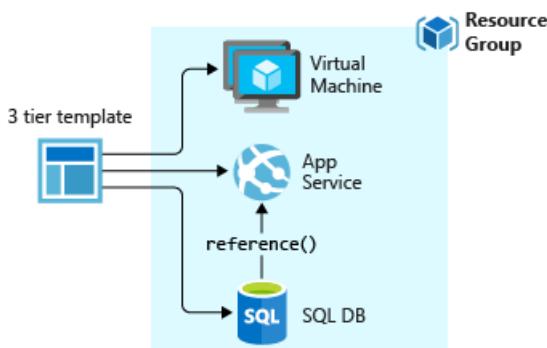
into REST API operations for the appropriate resource providers. For example, when Resource Manager receives a template with the following resource definition:

```
"resources": [
  {
    "apiVersion": "2016-01-01",
    "type": "Microsoft.Storage/storageAccounts",
    "name": "mystorageaccount",
    "location": "westus",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {}
  }
]
```

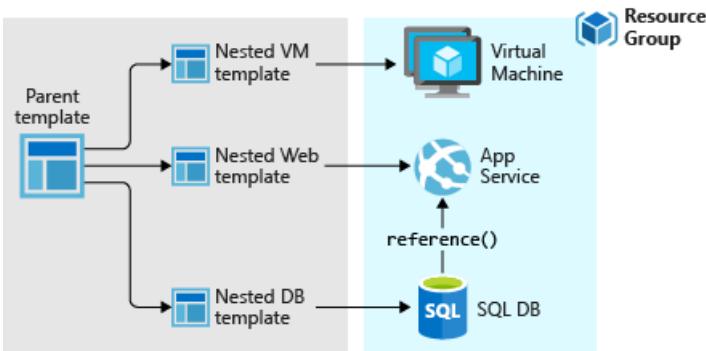
It converts the definition to the following REST API operation, which is sent to the Microsoft.Storage resource provider:

```
PUT
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/mystorageaccount?api-version=2016-01-01
REQUEST BODY
{
  "location": "westus",
  "properties": {},
  "sku": {
    "name": "Standard_LRS"
  },
  "kind": "Storage"
}
```

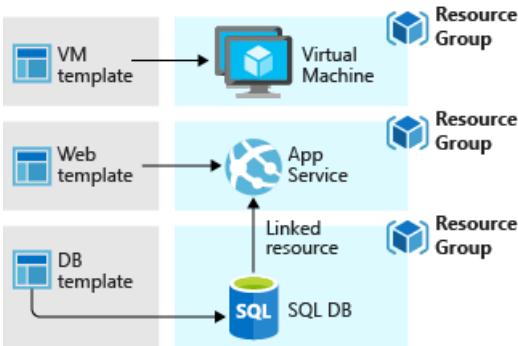
How you define templates and resource groups is entirely up to you and how you want to manage your solution. For example, you can deploy your three tier application through a single template to a single resource group.



But, you don't have to define your entire infrastructure in a single template. Often, it makes sense to divide your deployment requirements into a set of targeted, purpose-specific templates. You can easily reuse these templates for different solutions. To deploy a particular solution, you create a master template that links all the required templates. The following image shows how to deploy a three tier solution through a parent template that includes three nested templates.



If you envision your tiers having separate lifecycles, you can deploy your three tiers to separate resource groups. Notice the resources can still be linked to resources in other resource groups.



For information about nested templates, see [Using linked templates with Azure Resource Manager](#).

Azure Resource Manager analyzes dependencies to ensure resources are created in the correct order. If one resource relies on a value from another resource (such as a virtual machine needing a storage account for disks), you set a dependency. For more information, see [Defining dependencies in Azure Resource Manager templates](#).

You can also use the template for updates to the infrastructure. For example, you can add a resource to your solution and add configuration rules for the resources that are already deployed. If the template defines a resource that already exists, Resource Manager updates the existing resource instead of creating a new one.

Resource Manager provides extensions for scenarios when you need additional operations such as installing particular software that isn't included in the setup. If you're already using a configuration management service, like DSC, Chef or Puppet, you can continue working with that service by using extensions. For information about virtual machine extensions, see [About virtual machine extensions and features](#).

When you create a solution from the portal, the solution automatically includes a deployment template. You don't have to create your template from scratch because you can start with the template for your solution and customize it to meet your specific needs. For a sample, see [Quickstart: Create and deploy Azure Resource Manager templates by using the Azure portal](#). You can also retrieve a template for an existing resource group by either exporting the current state of the resource group, or viewing the template used for a particular deployment. Viewing the [exported template](#) is a helpful way to learn about the template syntax.

Finally, the template becomes part of the source code for your app. You can check it in to your source code repository and update it as your app evolves. You can edit the template through Visual Studio.

After defining your template, you're ready to deploy the resources to Azure. To deploy the resources, see:

- [Deploy resources with Resource Manager templates and Azure PowerShell](#)
- [Deploy resources with Resource Manager templates and Azure CLI](#)
- [Deploy resources with Resource Manager templates and Azure portal](#)
- [Deploy resources with Resource Manager templates and Resource Manager REST API](#)

Safe deployment practices

When deploying a complex service to Azure, you might need to deploy your service to multiple regions, and check its health before proceeding to the next step. Use [Azure Deployment Manager](#) to coordinate a staged rollout of the service. By staging the rollout of your service, you can find potential problems before it has been deployed to all regions. If you don't need these precautions, the deployment operations in the preceding section are the better option.

Deployment Manager is currently in public preview.

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.

Quickstarts and tutorials

Use the following quickstarts and tutorials to learn how to develop resource manager templates:

- Quickstarts

TITLE	DESCRIPTION
Use the Azure portal	Generate a template using the portal, and understand the process of editing and deploying the template.
Use Visual Studio Code	Use Visual Studio Code to create and edit templates, and how to use the Azure Cloud shell to deploy templates.
Use Visual Studio	Use Visual Studio to create, edit, and deploy templates.

- Tutorials

TITLE	DESCRIPTION
Utilize template reference	Utilize the template reference documentation to develop templates. In the tutorial, you find the storage account schema, and use the information to create an encrypted storage account.
Create multiple instances	Create multiple instances of Azure resources. In the tutorial, you create multiple instances of storage account.
Set resource deployment order	Define resource dependencies. In the tutorial, you create a virtual network, a virtual machine, and the dependent Azure resources. You learn how the dependencies are defined.

TITLE	DESCRIPTION
Use conditions	Deploy resources based on some parameter values. In the tutorial, you define a template to create a new storage account or use an existing storage account based on the value of a parameter.
Integrate key vault	Retrieve secrets/passwords from Azure Key Vault. In the tutorial, you create a virtual machine. The virtual machine administrator password is retrieved from a Key Vault.
Create linked templates	Modularize templates, and call other templates from a template. In the tutorial, you create a virtual network, a virtual machine, and the dependent resources. The dependent storage account is defined in a linked template.
Deploy virtual machine extensions	Perform post-deployment tasks by using extensions. In the tutorial, you deploy a customer script extension to install web server on the virtual machine.
Deploy SQL extensions	Perform post-deployment tasks by using extensions. In the tutorial, you deploy a customer script extension to install web server on the virtual machine.
Secure artifacts	Secure the artifacts needed to complete the deployments. In the tutorial, you learn how to secure the artifact used in the Deploy SQL extensions tutorial.
Use safe deployment practices	Use Azure Deployment manager.
Tutorial: Troubleshoot Resource Manager template deployments	Troubleshoot template deployment issues.

These tutorials can be used individually, or as a series to learn the major Resource Manager template development concepts.

Next steps

In this article, you learned how to use Azure Resource Manager for deployment, management, and access control of resources on Azure. Proceed to the next article to learn how to create your first Azure Resource Manager template.

[Quickstart: Create and deploy Azure Resource Manager templates by using the Azure portal](#)

Regions and availability for virtual machines in Azure

7/9/2018 • 7 minutes to read • [Edit Online](#)

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. 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?

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 [Microsoft Azure in China](#).
- **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 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 recommend 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.

An availability set is composed of two additional groupings that protect against hardware failures and allow updates to safely be applied - fault domains (FDs) and update domains (UDs). You can read more about how to manage the availability of [Linux VMs](#) or [Windows VMs](#).

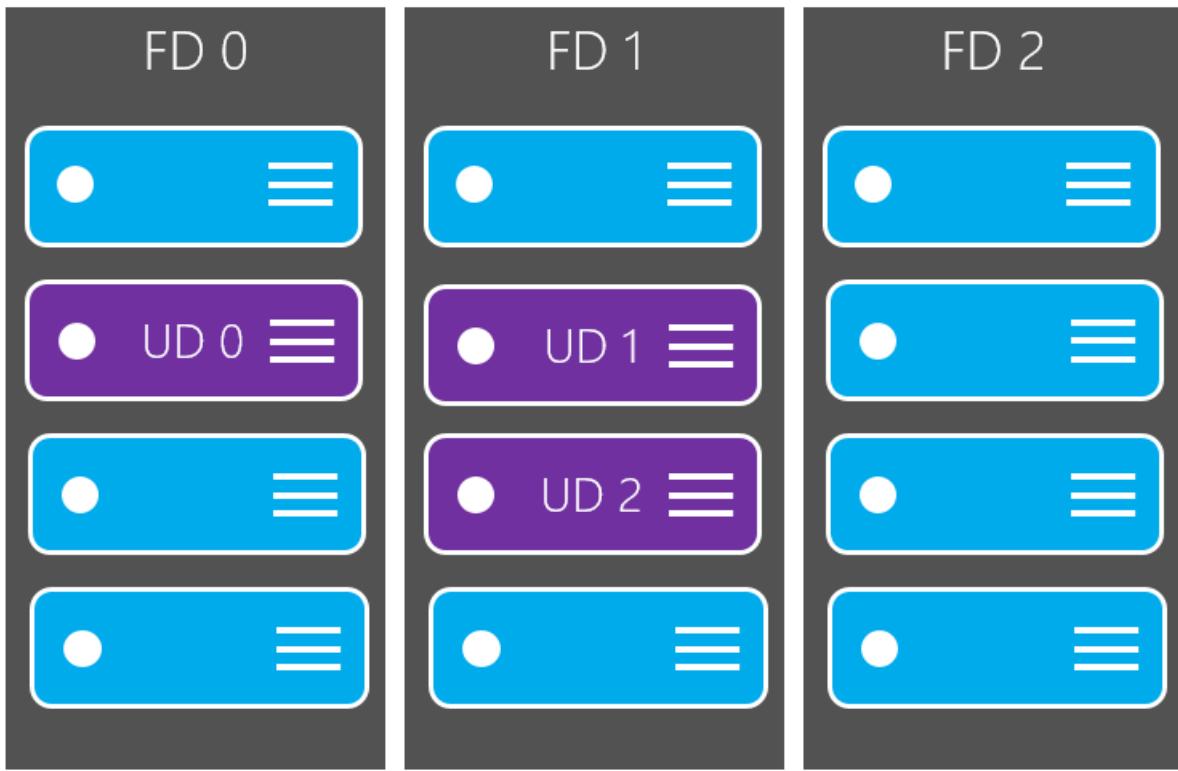
When allocating multiple compute resources which do not use the high availability constructs of fault domains there is a high probability of anti-affinity, however this anti-affinity is not guaranteed.

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. As you create VMs within an availability set, the Azure platform automatically distributes your VMs across these fault domains. This approach limits the impact of potential physical hardware failures, network outages, or power interruptions.

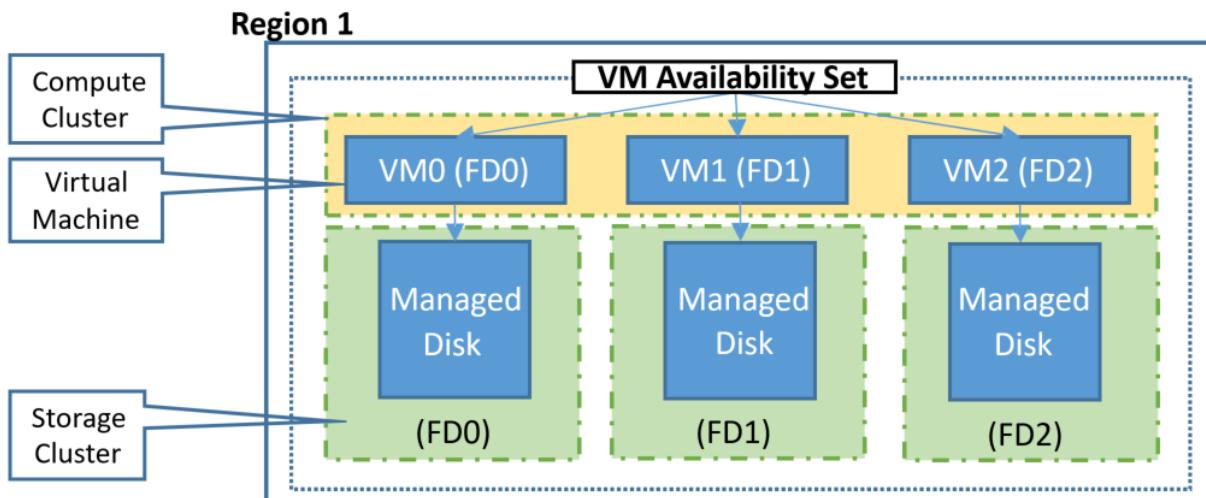
Update domains

An update domain is a logical group of underlying hardware that can undergo maintenance or be rebooted at the same time. As you create VMs within an availability set, the Azure platform automatically distributes your VMs across these update domains. 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 planned maintenance, but only one update domain is rebooted at a time.



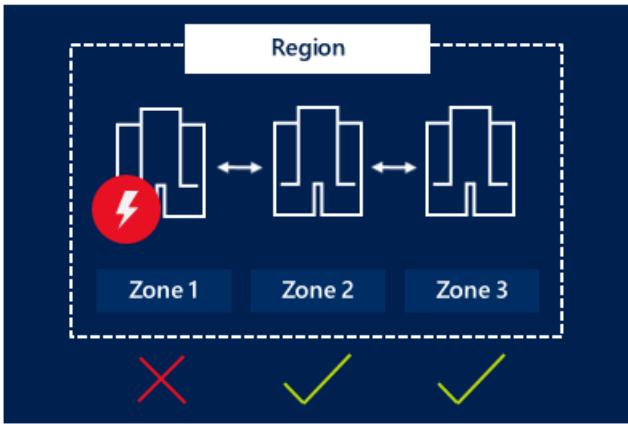
Managed Disk fault domains

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](#).



Availability zones

[Availability zones](#), an alternative to availability sets, 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.

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](#).

Sizes for Linux virtual machines in Azure

6/13/2019 • 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 Linux 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 [Windows virtual machines](#).

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, 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, 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	NV, NVv2, NC, NCv2, NCv3, ND, NDv2 (Preview)	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	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](#).
- For availability of VM sizes in Azure regions, see [Products available by region](#).
- To see general limits on Azure VMs, see [Azure subscription and service limits, quotas, and constraints](#).
- 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 Linux 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](#)
- [High performance compute](#)
- Check the [Previous generation](#) page for A Standard, Dv1 (D1-4 and D11-14 v1), and A8-A11 series

Support for generation 2 VMs (preview) on Azure

6/16/2019 • 4 minutes to read • [Edit Online](#)

IMPORTANT

Azure support for generation 2 VMs is currently in 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](#).

Support for generation 2 virtual machines (VMs) is now available in preview in Azure. You can't change a virtual machine's generation after you've created it, so review the considerations on this page before you choose a generation.

Generation 2 VMs support key features that aren't supported in generation 1 VMs. These features include increased memory, Intel Software Guard Extensions (Intel SGX), and virtualized persistent memory (vPMEM). Generation 2 VMs also have some features that aren't supported in Azure yet. For more information, see the [Features and capabilities](#) section.

Generation 2 VMs use the new UEFI-based boot architecture rather than the BIOS-based architecture used by generation 1 VMs. Compared to generation 1 VMs, generation 2 VMs might have improved boot and installation times. For an overview of generation 2 VMs and some of the differences between generation 1 and generation 2, see [Should I create a generation 1 or 2 virtual machine in Hyper-V?](#).

Generation 2 VM sizes

Generation 1 VMs are supported by all VM sizes in Azure. Azure now offers preview generation 2 support for the following selected VM series:

- [Dsv2-series](#) and [Dsv3-series](#)
- [Esv3-series](#)
- [Fsv2-series](#)
- [GS-series](#)
- [Ls-series](#) and [Lsv2-series](#)
- [Mv2-series](#)

Generation 2 VM images in Azure Marketplace

Generation 2 VMs support the following Marketplace images:

- Windows Server 2019 Datacenter
- Windows Server 2016 Datacenter
- Windows Server 2012 R2 Datacenter
- Windows Server 2012 Datacenter

On-premises vs. Azure generation 2 VMs

Azure doesn't currently support some of the features that on-premises Hyper-V supports for generation 2 VMs.

GENERATION 2 FEATURE	ON-PREMISES HYPER-V	AZURE
Secure boot	✓□	□
Shielded VM	✓□	□
vTPM	✓□	□
Virtualization-based security (VBS)	✓□	□
VHDX format	✓□	□

Features and capabilities

Generation 1 vs. generation 2 features

FEATURE	GENERATION 1	GENERATION 2
Boot	PCAT	UEFI
Disk controllers	IDE	SCSI
VM sizes	All VM sizes	Only VMs that support premium storage

Generation 1 vs. generation 2 capabilities

CAPABILITY	GENERATION 1	GENERATION 2
OS disk > 2 TB	□	✓□
Custom disk/image/swap OS	✓□	✓□
Virtual machine scale set support	✓□	✓□
ASR/backup	✓□	□
Shared image gallery	✓□	□
Azure disk encryption	✓□	□

Creating a generation 2 VM

Marketplace image

In the Azure portal or Azure CLI, you can create generation 2 VMs from a Marketplace image that supports UEFI boot.

The `windowsserver-gen2preview` offer contains Windows generation 2 images only. This packaging avoids confusion between generation 1 and generation 2 images. To create a generation 2 VM, select **Images** from this offer and follow the standard process to create the VM.

Currently, Marketplace offers the following Windows generation 2 images:

- 2019-datacenter-gen2
- 2016-datacenter-gen2
- 2012-r2-datacenter-gen2
- 2012-datacenter-gen2

See the [Features and capabilities](#) section for a current list of supported Marketplace images.

Managed image or managed disk

You can create a generation 2 VM from a managed image or managed disk in the same way you would create a generation 1 VM.

Virtual machine scale sets

You can also create generation 2 VMs by using virtual machine scale sets. In the Azure CLI, use Azure scale sets to create generation 2 VMs.

Frequently asked questions

- **Are generation 2 VMs available in all Azure regions?**

Yes. But not all [generation 2 VM sizes](#) are available in every region. The availability of the generation 2 VM depends on the availability of the VM size.

- **Is there a price difference between generation 1 and generation 2 VMs?**

No.

- **How do I increase the OS disk size?**

OS disks larger than 2 TB are new to generation 2 VMs. By default, OS disks are smaller than 2 TB for generation 2 VMs. You can increase the disk size up to a recommended maximum of 4 TB. Use the Azure CLI or the Azure portal to increase the OS disk size. For information about how to expand disks programmatically, see [Resize a disk](#).

To increase the OS disk size from the Azure portal:

1. In the Azure portal, go to the VM properties page.
2. To shut down and deallocate the VM, select the **Stop** button.
3. In the **Disk**s section, select the OS disk you want to increase.
4. In the **Disk**s section, select **Configuration**, and update the **Size** to the value you want.
5. Go back to the VM properties page and **Start** the VM.

You might see a warning for OS disks larger than 2 TB. The warning doesn't apply to generation 2 VMs. However, OS disk sizes larger than 4 TB are *not recommended*.

- **Do generation 2 VMs support accelerated networking?**

Yes. For more information, see [Create a VM with accelerated networking](#).

- **Is VHDX supported on generation 2?**

No, generation 2 VMs support only VHD.

- **Do generation 2 VMs support Azure Ultra Disk Storage?**

Yes.

- **Can I migrate a VM from generation 1 to generation 2?**

No, you can't change the generation of a VM after you create it. If you need to switch between VM generations, create a new VM of a different generation.

Next steps

- Learn about [generation 2 virtual machines in Hyper-V](#).

General purpose virtual machine sizes

10/8/2018 • 9 minutes to read • [Edit Online](#)

General purpose VM sizes provide balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers. This article provides information about the number of vCPUs, data disks and NICs as well as storage throughput for sizes in this grouping.

- The [DC-series](#) is a new family of virtual machines in Azure that can help protect the confidentiality and integrity of your data and code while it's processed in the public cloud. These machines are backed by the latest generation of 3.7GHz Intel XEON E-2176G Processor with SGX technology. With the Intel Turbo Boost Technology these machines can go up to 4.7GHz. DC series instances enable customers to build secure enclave-based applications to protect their code and data while it's in use.
- The Av2-series VMs can be deployed on a variety of hardware types and processors. A-series VMs have CPU performance and memory configurations best suited for entry level workloads like development and test. The size is throttled, based upon the hardware, to offer consistent processor performance for the running instance, regardless of the hardware it is deployed on. To determine the physical hardware on which this size is deployed, query the virtual hardware from within the Virtual Machine.

Example use cases include development and test servers, low traffic web servers, small to medium databases, proof-of-concepts, and code repositories.

- Dv2-series, a follow-on to the original D-series, features a more powerful CPU and optimal CPU-to-memory configuration making them suitable for most production workloads. The Dv2-series CPU is about 35% faster than the D-series CPU. It is based on the latest generation Intel Xeon® E5-2673 v3 2.4 GHz (Haswell) or E5-2673 v4 2.3 GHz (Broadwell) processors, and with the Intel Turbo Boost Technology 2.0, can go up to 3.1 GHz. The Dv2-series has the same memory and disk configurations as the D-series.
- The Dv3-series features the 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor or the latest 2.3 GHz Intel XEON ® E5-2673 v4 (Broadwell) processor in a hyper-threaded configuration, providing a better value proposition for most general purpose workloads. Memory has been expanded (from ~3.5 GiB/vCPU to 4 GiB/vCPU) while disk and network limits have been adjusted on a per core basis to align with the move to hyperthreading. The Dv3 no longer has the high memory VM sizes of the D/Dv2 families, those have been moved to the new Ev3 family.

Example D-series use cases include enterprise-grade applications, relational databases, in-memory caching, and analytics.

B-series

Premium Storage: Supported

Premium Storage caching: Not Supported

The B-series burstable VMs are ideal for workloads that do not need the full performance of the CPU continuously, like web servers, small databases and development and test environments. These workloads typically have burstable performance requirements. The B-Series provides these customers the ability to purchase a VM size with a price conscious baseline performance that allows the VM instance to build up credits when the VM is utilizing less than its base performance. When the VM has accumulated credit, the VM can burst above the VM's baseline using up to 100% of the CPU when your application requires the higher CPU performance.

Example use cases include development and test servers, low-traffic web servers, small databases, micro services,

servers for proof-of-concepts, build servers.

SIZE	VCPU	MEM ORY: GIB	TEMP STOR AGE (SSD) GIB	BASE CPU PERF OF VM	MAX CPU PERF OF VM	INITIAL CREDITS	CREDITS BANKED / HOUR	MAX BANKED CREDITS	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs
Standard_B1ls ¹	1	0.5	4	5%	100 %	30	3	72	2	200 / 10	160 / 10	2
Standard_B1s	1	1	4	10%	100 %	30	6	144	2	400 / 10	320 / 10	2
Standard_B1ms	1	2	4	20%	100 %	30	12	288	2	800 / 10	640 / 10	2
Standard_B2s	2	4	8	40%	200 %	60	24	576	4	1600 / 15	1280 / 15	3
Standard_B2ms	2	8	16	60%	200 %	60	36	864	4	2400 / 22.5	1920 / 22.5	3
Standard_B4ms	4	16	32	90%	400 %	120	54	1296	8	3600 / 35	2880 / 35	4
Standard_B8ms	8	32	64	135 %	800 %	240	81	1944	16	4320 / 50	4320 / 50	4
Standard_B12ms	12	48	96	202 %	1200 %	360	121	2909	16	6480 / 75	4320 / 50	6
Standard_B16ms	16	64	128	270 %	1600 %	480	162	3888	32	8640 / 100	4320 / 50	8
Standard_B20ms	20	80	160	337 %	2000 %	600	203	4860	32	10800 / 125	4320 / 50	8

¹ B1ls is supported only on Linux

ACU: 160-190

Premium Storage: Supported

Premium Storage caching: Supported

Dsv3-series sizes are based on the 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor or the latest 2.3 GHz Intel XEON ® E5-2673 v4 (Broadwell) processor that can achieve 3.5GHz with Intel Turbo Boost Technology 2.0 and use premium storage. The Dsv3-series sizes offer a combination of vCPU, memory, and temporary storage for most production workloads.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D 2s_v3	2	8	16	4	4000 / 32 (50)	3200 / 48	2 / 1000
Standard_D 4s_v3	4	16	32	8	8000 / 64 (100)	6400 / 96	2 / 2000
Standard_D 8s_v3	8	32	64	16	16000 / 128 (200)	12800 / 192	4 / 4000
Standard_D 16s_v3	16	64	128	32	32000 / 256 (400)	25600 / 384	8 / 8000
Standard_D 32s_v3	32	128	256	32	64000 / 512 (800)	51200 / 768	8 / 16000
Standard_D 64s_v3	64	256	512	32	128000 / 1024 (1600)	80000 / 1200	8 / 30000

¹ Dsv3-series VM's feature Intel® Hyper-Threading Technology

Dv3-series ¹

ACU: 160-190

Premium Storage: Not Supported

Premium Storage caching: Not Supported

Dv3-series sizes are based on the 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor or 2.3 GHz Intel XEON ® E5-2673 v4 (Broadwell) processor that can achieve 3.5GHz with Intel Turbo Boost Technology 2.0. The Dv3-series sizes offer a combination of vCPU, memory, and temporary storage for most production workloads.

Data disk storage is billed separately from virtual machines. To use premium storage disks, use the Dsv3 sizes. The pricing and billing meters for Dsv3 sizes are the same as Dv3-series.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX NICs / NETWORK BANDWIDTH
Standard_D2_v3	2	8	50	4	3000/46/23	2 / 1000
Standard_D4_v3	4	16	100	8	6000/93/46	2 / 2000
Standard_D8_v3	8	32	200	16	12000/187/93	4 / 4000
Standard_D16_v3	16	64	400	32	24000/375/187	8 / 8000
Standard_D32_v3	32	128	800	32	48000/750/375	8 / 16000
Standard_D64_v3	64	256	1600	32	96000/1000/500	8 / 30000

¹ Dv3-series VM's feature Intel® Hyper-Threading Technology

DSv2-series

ACU: 210-250

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D S1_v2	1	3.5	7	4	4000 / 32 (43)	3200 / 48	2 / 750
Standard_D S2_v2	2	7	14	8	8000 / 64 (86)	6400 / 96	2 / 1500
Standard_D S3_v2	4	14	28	16	16000 / 128 (172)	12800 / 192	4 / 3000
Standard_D S4_v2	8	28	56	32	32000 / 256 (344)	25600 / 384	8 / 6000
Standard_D S5_v2	16	56	112	64	64000 / 512 (688)	51200 / 768	8 / 12000

Dv2-series

ACU: 210-250

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS	THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D1_v2	1	3.5	50	3000 / 46 / 23	4	4x500	2 / 750
Standard_D2_v2	2	7	100	6000 / 93 / 46	8	8x500	2 / 1500
Standard_D3_v2	4	14	200	12000 / 187 / 93	16	16x500	4 / 3000
Standard_D4_v2	8	28	400	24000 / 375 / 187	32	32x500	8 / 6000
Standard_D5_v2	16	56	800	48000 / 750 / 375	64	64x500	8 / 12000

Av2-series

ACU: 100

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS / THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_A1_v2	1	2	10	1000 / 20 / 10	2 / 2x500	2 / 250
Standard_A2_v2	2	4	20	2000 / 40 / 20	4 / 4x500	2 / 500
Standard_A4_v2	4	8	40	4000 / 80 / 40	8 / 8x500	4 / 1000
Standard_A8_v2	8	16	80	8000 / 160 / 80	16 / 16x500	8 / 2000

SIZE	VCPU	MEMORY: GiB	TEMP STORAGE (SSD) GiB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS / THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_A2 m_v2	2	16	20	2000 / 40 / 20	4 / 4x500	2 / 500
Standard_A4 m_v2	4	32	40	4000 / 80 / 40	8 / 8x500	4 / 1000
Standard_A8 m_v2	8	64	80	8000 / 160 / 80	16 / 16x500	8 / 2000

DC-series

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPU	MEMORY: GiB	TEMP STORAGE (SSD) GiB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GiB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D C2s	2	8	100	2	4000 / 32 (43)	3200 / 48	2 / 1500
Standard_D C4s	4	16	200	4	8000 / 64 (86)	6400 / 96	2 / 3000

Size table definitions

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller. For example, 1023 GiB = 1098.4 GB
- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6 bytes/sec.
- Data disks can operate in cached or uncached modes. For cached data disk operation, the host cache mode is set to **ReadOnly** or **ReadWrite**. For uncached data disk operation, the host cache mode is set to **None**.
- If you want to get the best performance for your VMs, you should limit the number of data disks to 2 disks per vCPU.
- Expected network bandwidth** is the maximum aggregated [bandwidth allocated per VM type](#) across all NICs, for all destinations. Upper limits are not guaranteed, but are intended to provide guidance for selecting the right VM type for the intended application. Actual network performance will depend on a variety of factors including network congestion, application loads, and network settings. For information on optimizing network throughput, see [Optimizing network throughput for Windows and Linux](#). To achieve the expected network performance on Linux or Windows, it may be necessary to select a specific version or optimize your VM. For more information, see [How to reliably test for virtual machine throughput](#).

Other sizes

- [Compute optimized](#)
- [Memory optimized](#)
- [Storage optimized](#)
- [GPU](#)
- [High performance compute](#)
- [Previous generations](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

B-series burstable virtual machine sizes

6/28/2019 • 7 minutes to read • [Edit Online](#)

The B-series VM family allows you to choose which VM size provides you the necessary base level performance for your workload, with the ability to burst CPU performance up to 100% of an Intel® Broadwell E5-2673 v4 2.3 GHz, or an Intel® Haswell 2.4 GHz E5-2673 v3 processor vCPU.

The B-series VMs are ideal for workloads that do not need the full performance of the CPU continuously, like web servers, proof of concepts, small databases and development build environments. These workloads typically have burstable performance requirements. The B-series provides you with the ability to purchase a VM size with baseline performance and the VM instance builds up credits when it is using less than its baseline. When the VM has accumulated credit, the VM can burst above the baseline using up to 100% of the vCPU when your application requires higher CPU performance.

The B-series comes in the following VM sizes:

SIZE	VCPU	MEM ORY: GIB	TEMP STOR AGE (SSD) GIB	BASE CPU PERF OF VM	MAX CPU PERF OF VM	INITI AL CREDI TS	CREDI TS BANK ED / HOUR	MAX BANK ED CREDI TS	MAX DATA DISKS	MAX CACH ED AND TEMP STOR AGE	MAX UNCA CHED DISK THRO UGHP UT: IOPS / MBPS	MAX NICS
Standard_B_1s ¹	1	0.5	4	5%	100%	30	3	72	2	200 / 10	160 / 10	2
Standard_B_1s	1	1	4	10%	100%	30	6	144	2	400 / 10	320 / 10	2
Standard_B_1ms	1	2	4	20%	100%	30	12	288	2	800 / 10	640 / 10	2
Standard_B_2s	2	4	8	40%	200%	60	24	576	4	1600 / 15	1280 / 15	3
Standard_B_2ms	2	8	16	60%	200%	60	36	864	4	2400 / 22.5	1920 / 22.5	3
Standard_B_4ms	4	16	32	90%	400%	120	54	1296	8	3600 / 35	2880 / 35	4
Standard_B_8ms	8	32	64	135%	800%	240	81	1944	16	4320 / 50	4320 / 50	4

SIZE	VCPU	MEM ORY: GiB	TEMP STOR AGE (SSD) GiB	BASE CPU PERF OF VM	MAX CPU PERF OF VM	INITI AL CREDI TS	CREDI TS BANK ED / HOUR	MAX BANK ED CREDI TS	MAX DATA DISKS	MAX CACH ED AND TEMP STOR AGE	MAX UNCA CHED DISK	MAX NICS
										THRO UGHP UT:	IOPS / MBPS	
Standard_B 12ms	12	48	96	202%	1200 %	360	121	2909	16	6480 / 75	4320 / 50	6
Standard_B 16ms	16	64	128	270%	1600 %	480	162	3888	32	8640 / 100	4320 / 50	8
Standard_B 20ms	20	80	160	337%	2000 %	600	203	4860	32	1080 0 / 125	4320 / 50	8

¹ B11s is supported only on Linux

Workload example

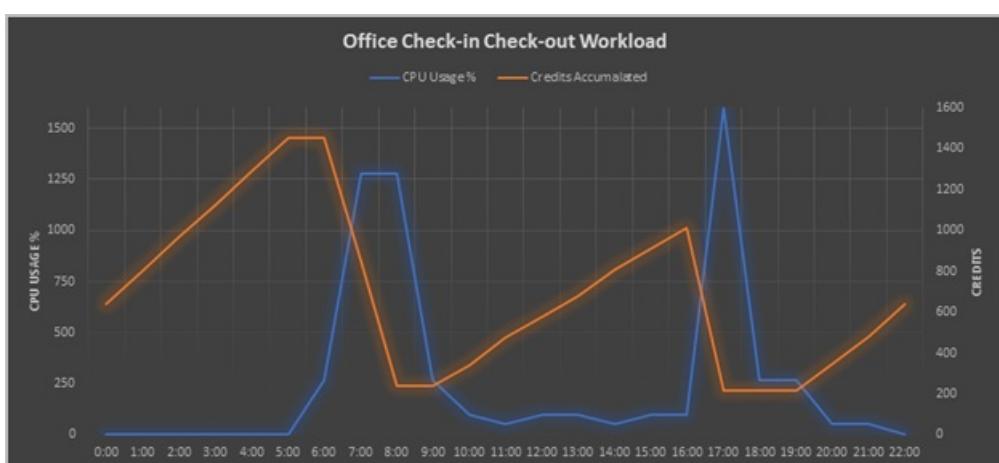
Consider an office check-in/out application. The application needs CPU bursts during business hours, but not a lot of computing power during off hours. In this example, the workload requires a 16vCPU virtual machine with 64GiB of RAM to work efficiently.

The table shows the hourly traffic data and the chart is a visual representation of that traffic.

B16 characteristics:

Max CPU perf: 16vCPU * 100% = 1600%

Baseline: 270%



SCENARIO	TIME	CPU USAGE (%)	CREDITS ACCUMULATED ¹	CREDITS AVAILABLE
B16ms Deployment	Deployment	Deployment	480 (Initial Credits)	480
No traffic	0:00	0	162	642

SCENARIO	TIME	CPU USAGE (%)	CREDITS ACCUMULATED	CREDITS AVAILABLE
No traffic	1:00	0	162	804
No traffic	2:00	0	162	966
No traffic	3:00	0	162	1128
No traffic	4:00	0	162	1290
No traffic	5:00	0	162	1452
Low Traffic	6:00	270	0	1452
Employees come to office (app needs 80% vCPU)	7:00	1280	-606	846
Employees continue coming to office (app needs 80% vCPU)	8:00	1280	-606	240
Low Traffic	9:00	270	0	240
Low Traffic	10:00	100	102	342
Low Traffic	11:00	50	132	474
Low Traffic	12:00	100	102	576
Low Traffic	13:00	100	102	678
Low Traffic	14:00	50	132	810
Low Traffic	15:00	100	102	912
Low Traffic	16:00	100	102	1014
Employees checking out (app needs 100% vCPU)	17:00	1600	-798	216
Low Traffic	18:00	270	0	216
Low Traffic	19:00	270	0	216
Low Traffic	20:00	50	132	348
Low Traffic	21:00	50	132	480
No traffic	22:00	0	162	642
No traffic	23:00	0	162	804

¹ Credits accumulated/credits used in an hour is equivalent to:

$$((\text{Base CPU perf of VM} - \text{CPU Usage}) / 100) * 60 \text{ minutes}$$

For a D16s_v3 which has 16 vCPUs and 64 GiB of memory the hourly rate is \$0.936 per hour (monthly \$673.92) and for B16ms with 16 vCPUs and 64 GiB memory the rate is \$0.794 per hour (monthly \$547.86). **This results in 15% savings!**

Q & A

Q: How do you get 135% baseline performance from a VM?

A: The 135% is shared amongst the 8 vCPU's that make up the VM size. For example, if your application uses 4 of the 8 cores working on batch processing and each of those 4 vCPU's are running at 30% utilization the total amount of VM CPU performance would equal 120%. Meaning that your VM would be building credit time based on the 15% delta from your baseline performance. But it also means that when you have credits available that same VM can use 100% of all 8 vCPU's giving that VM a Max CPU performance of 800%.

Q: How can I monitor my credit balance and consumption

A: We will be introducing 2 new metrics in the coming weeks, the **Credit** metric will allow you to view how many credits your VM has banked and the **ConsumedCredit** metric will show how many CPU credits your VM has consumed from the bank. You will be able to view these metrics from the metrics pane in the portal or programmatically through the Azure Monitor APIs.

For more information on how to access the metrics data for Azure, see [Overview of metrics in Microsoft Azure](#).

Q: How are credits accumulated?

A: The VM accumulation and consumption rates are set such that a VM running at exactly its base performance level will have neither a net accumulation or consumption of bursting credits. A VM will have a net increase in credits whenever it is running below its base performance level and will have a net decrease in credits whenever the VM is utilizing the CPU more than its base performance level.

Example: I deploy a VM using the B1ms size for my small time and attendance database application. This size allows my application to use up to 20% of a vCPU as my baseline, which is 0.2 credits per minute I can use or bank.

My application is busy at the beginning and end of my employees work day, between 7:00-9:00 AM and 4:00 - 6:00PM. During the other 20 hours of the day, my application is typically at idle, only using 10% of the vCPU. For the non-peak hours, I earn 0.2 credits per minute but only consume 0.1 credits per minute, so my VM will bank $0.2 - 0.1 = 0.1$ credits per minute. For the 20 hours that I am off-peak, I will bank $0.1 \times 60 \times 20 = 120$ credits.

During peak hours my application averages 60% vCPU utilization, I still earn 0.2 credits per minute but I consume 0.6 credits per minute, for a net cost of 0.4 credits a minute or $0.4 \times 60 = 24$ credits per hour. I have 4 hours per day of peak usage, so it costs $4 \times 24 = 96$ credits for my peak usage.

If I take the 120 credits I earned off-peak and subtract the 96 credits I used for my peak times, I bank an additional 24 credits per day that I can use for other bursts of activity.

Q: How can I calculate credits accumulated and used?

A: You can use the following formula:

$$(\text{Base CPU perf of VM} - \text{CPU Usage}) / 100 = \text{Credits bank or use per minute}$$

e.g. in above instance your baseline is 20% and if you use 10% of the CPU you are accumulating $(20\% - 10\%) / 100 = 0.1$ credit per minute.

Q: Does the B-Series support Premium Storage data disks?

A: Yes, all B-Series sizes support Premium Storage data disks.

Q: Why is my remaining credit set to 0 after a redeploy or a stop/start?

A : When a VM is "REDPLOYED" and the VM moves to another node, the accumulated credit is lost. If the VM is stopped/started, but remains on the same node, the VM retains the accumulated credit. Whenever the VM starts fresh on a node, it gets an initial credit, for Standard_B8ms it is 240 mins.

Q: What happens if I deploy an unsupported OS image on B1ls?

A : B1ls only supports Linux images and if you deploy any another OS image you might not get the best customer experience.

Other sizes

- [General purpose](#)
- [Compute optimized](#)
- [Memory optimized](#)
- [Storage optimized](#)
- [GPU optimized](#)
- [High performance compute](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

Compute optimized virtual machine sizes

7/6/2018 • 3 minutes to read • [Edit Online](#)

Compute optimized VM sizes have a high CPU-to-memory ratio and are good for medium traffic web servers, network appliances, batch processes, and application servers. This article provides information about the number of vCPUs, data disks, and NICs as well as storage throughput and network bandwidth for each size in this grouping.

Fsv2-series is based on the Intel® Xeon® Platinum 8168 processor, featuring a sustained all core Turbo clock speed of 3.4GHz and a maximum single-core turbo frequency of 3.7 GHz. Intel® AVX-512 instructions, which are new on Intel Scalable Processors, will provide up to a 2X performance boost to vector processing workloads on both single and double precision floating point operations. In other words, they are really fast for any computational workload.

At a lower per-hour list price, the Fsv2-series is the best value in price-performance in the Azure portfolio based on the Azure Compute Unit (ACU) per vCPU.

Fsv2-series¹

ACU: 195 - 210

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPU'S	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_F_2s_v2	2	4	16	4	4000 / 31 (32)	3200 / 47	2 / 875
Standard_F_4s_v2	4	8	32	8	8000 / 63 (64)	6400 / 95	2 / 1750
Standard_F_8s_v2	8	16	64	16	16000 / 127 (128)	12800 / 190	4 / 3500
Standard_F_16s_v2	16	32	128	32	32000 / 255 (256)	25600 / 380	4 / 7000
Standard_F_32s_v2	32	64	256	32	64000 / 512 (512)	51200 / 750	8 / 14000
Standard_F_64s_v2	64	128	512	32	128000 / 1024 (1024)	80000 / 1100	8 / 28000

SIZE	VCPUs	MEMORY: GiB	TEMP STORAGE (SSD) GiB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GiB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_F72s_v2 ^{2, 3}	72	144	576	32	144000 / 1152 (1520)	80000 / 1100	8 / 30000

¹ Fsv2-series VM's feature Intel® Hyper-Threading Technology

² More than 64 vCPU's require one of these supported guest OSes: Windows Server 2016, Ubuntu 16.04 LTS, SLES 12 SP2, and Red Hat Enterprise Linux, CentOS 7.3, or Oracle Linux 7.3 with LIS 4.2.1

³ Instance is isolated to hardware dedicated to a single customer.

Size table definitions

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller. For example, 1023 GiB = 1098.4 GB
- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6 bytes/sec.
- Data disks can operate in cached or uncached modes. For cached data disk operation, the host cache mode is set to **ReadOnly** or **ReadWrite**. For uncached data disk operation, the host cache mode is set to **None**.
- If you want to get the best performance for your VMs, you should limit the number of data disks to 2 disks per vCPU.
- Expected network bandwidth** is the maximum aggregated [bandwidth allocated per VM type](#) across all NICs, for all destinations. Upper limits are not guaranteed, but are intended to provide guidance for selecting the right VM type for the intended application. Actual network performance will depend on a variety of factors including network congestion, application loads, and network settings. For information on optimizing network throughput, see [Optimizing network throughput for Windows and Linux](#). To achieve the expected network performance on Linux or Windows, it may be necessary to select a specific version or optimize your VM. For more information, see [How to reliably test for virtual machine throughput](#).

Other sizes

- [General purpose](#)
- [Memory optimized](#)
- [Storage optimized](#)
- [GPU](#)
- [High performance compute](#)
- [Previous generations](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

Memory optimized virtual machine sizes

5/17/2019 • 11 minutes to read • [Edit Online](#)

Memory optimized VM sizes offer a high memory-to-CPU ratio that are great for relational database servers, medium to large caches, and in-memory analytics. This article provides information about the number of vCPUs, data disks and NICs as well as storage throughput and network bandwidth for each size in this grouping.

- The Mv2-Series offers the highest vCPU count (up to 208 vCPUs) and largest memory (up to 5.7 TiB) of any VM in the cloud. It's ideal for extremely large databases or other applications that benefit from high vCPU counts and large amounts of memory.
- The M-Series offers a high vCPU count (up to 128 vCPUs) and a large amount of memory (up to 3.8 TiB). It's also ideal for extremely large databases or other applications that benefit from high vCPU counts and large amounts of memory.
- Dv2-series, G-series, and the DSv2/GS counterparts are ideal for applications that demand faster vCPUs, better temporary storage performance, or have higher memory demands. They offer a powerful combination for many enterprise-grade applications.
- Dv2-series, a follow-on to the original D-series, features a more powerful CPU. The Dv2-series CPU is about 35% faster than the D-series CPU. It is based on the latest generation 2.4 GHz Intel Xeon® E5-2673 v3 2.4 GHz (Haswell) or E5-2673 v4 2.3 GHz (Broadwell) processors, and with the Intel Turbo Boost Technology 2.0, can go up to 3.1 GHz. The Dv2-series has the same memory and disk configurations as the D-series.
- The Ev3-series features the E5-2673 v4 2.3 GHz (Broadwell) processor in a hyper-threaded configuration, providing a better value proposition for most general purpose workloads, and bringing the Ev3 into alignment with the general purpose VMs of most other clouds. Memory has been expanded (from 7 GiB/vCPU to 8 GiB/vCPU) while disk and network limits have been adjusted on a per core basis to align with the move to hyperthreading. The Ev3 is the follow up to the high memory VM sizes of the D/Dv2 families.
- Azure Compute offers virtual machine sizes that are isolated to a specific hardware type and dedicated to a single customer. These virtual machine sizes are best suited for workloads that require a high degree of isolation from other customers for workloads involving elements like compliance and regulatory requirements. Customers can also choose to further subdivide the resources of these isolated virtual machines by using [Azure support for nested virtual machines](#). Please see the tables of virtual machine families below for your isolated VM options.

ESv3-series

ACU: 160-190¹

Premium Storage: Supported

Premium Storage caching: Supported

ESv3-series instances are based on the 2.3 GHz Intel XEON® E5-2673 v4 (Broadwell) processor and can achieve 3.5GHz with Intel Turbo Boost Technology 2.0 and use premium storage. Ev3-series instances are ideal for memory-intensive enterprise applications.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_E_2s_v3	2	16	32	4	4000 / 32 (50)	3200 / 48	2 / 1000
Standard_E_4s_v3 ²	4	32	64	8	8000 / 64 (100)	6400 / 96	2 / 2000
Standard_E_8s_v3 ²	8	64	128	16	16000 / 128 (200)	12800 / 192	4 / 4000
Standard_E_16s_v3 ²	16	128	256	32	32000 / 256 (400)	25600 / 384	8 / 8000
Standard_E_20s_v3	20	160	320	32	40000 / 320 (400)	32000 / 480	8 / 10000
Standard_E_32s_v3 ²	32	256	512	32	64000 / 512 (800)	51200 / 768	8 / 16000
Standard_E_64s_v3 ²	64	432	864	32	128000 / 1024 (1600)	80000 / 1200	8 / 30000
Standard_E_64is_v3 ³	64	432	864	32	128000 / 1024 (1600)	80000 / 1200	8 / 30000

¹ Esv3-series VM's feature Intel® Hyper-Threading Technology.

² Constrained core sizes available.

³ Instance is isolated to hardware dedicated to a single customer.

Ev3-series

ACU: 160 - 190¹

Premium Storage: Not Supported

Premium Storage caching: Not Supported

Ev3-series instances are based on the 2.3 GHz Intel XEON ® E5-2673 v4 (Broadwell) processor and can achieve 3.5GHz with Intel Turbo Boost Technology 2.0. Ev3-series instances are ideal for memory-intensive enterprise applications.

Data disk storage is billed separately from virtual machines. To use premium storage disks, use the ESv3 sizes. The pricing and billing meters for ESv3 sizes are the same as Ev3-series.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX NICs / NETWORK BANDWIDTH
Standard_E2_v3	2	16	50	4	3000/46/23	2 / 1000
Standard_E4_v3	4	32	100	8	6000/93/46	2 / 2000
Standard_E8_v3	8	64	200	16	12000/187/93	4 / 4000
Standard_E16_v3	16	128	400	32	24000/375/187	8 / 8000
Standard_E20_v3	20	160	500	32	30000/469/234	8 / 10000
Standard_E32_v3	32	256	800	32	48000/750/375	8 / 16000
Standard_E64_v3	64	432	1600	32	96000/1000/500	8 / 30000
Standard_E64i_v3 ^{2, 3}	64	432	1600	32	96000/1000/500	8 / 30000

¹ Ev3-series VM's feature Intel® Hyper-Threading Technology.

² Constrained core sizes available.

³ Instance is isolated to hardware dedicated to a single customer.

Mv2-series

Premium Storage: Supported

Premium Storage caching: Supported

Write Accelerator: [Supported](#)

The Mv2-series features high throughput, low latency, directly mapped local NVMe storage running on a hyper-threaded Intel® Xeon® Platinum 8180M 2.5GHz (Skylake) processor with an all core base frequency of 2.5 GHz and a max turbo frequency of 3.8 GHz. All Mv2-series virtual machine sizes can use both standard and premium persistent disks. Mv2-series instances are memory optimized VM sizes providing unparalleled computational performance to support large in-memory databases and workloads, with a high memory-to-CPU ratio that is ideal for relational database servers, large caches, and in-memory analytics.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_M208ms_v2 ^{1, 2}	208	5700	4096	64	80000 / 800 (7040)	40000 / 1000	8 / 16000
Standard_M208s_v2 ^{1, 2}	208	2850	4096	64	80000 / 800 (7040)	40000 / 1000	8 / 16000

Mv2-series VM's feature Intel® Hyper-Threading Technology

¹ These large VMs require one of these supported guest OSes: Windows Server 2016, Windows Server 2019, SLES 12 SP4, SLES 15.

² Mv2-series VMs are generation 2 only. If you're using Linux, see the following section for how to find and select a SUSE Linux image.

Find a SUSE image

To select an appropriate SUSE Linux image in the Azure portal:

1. In the Azure portal, select **Create a resource**
2. Search for "SUSE SAP"
3. SLES for SAP generation 2 images are available as either pay-as-you-go, or bring your own subscription (BYOS). In the search results, expand the desired image category:
 - SUSE Linux Enterprise Server (SLES) for SAP
 - SUSE Linux Enterprise Server (SLES) for SAP (BYOS)
4. SUSE images compatible with the Mv2-series are prefixed with the name **GEN2:**. The following SUSE images are available for Mv2-series VMs:
 - GEN2: SUSE Linux Enterprise Server (SLES) 12 SP4 for SAP Applications
 - GEN2: SUSE Linux Enterprise Server (SLES) 15 for SAP Applications
 - GEN2: SUSE Linux Enterprise Server (SLES) 12 SP4 for SAP Applications (BYOS)
 - GEN2: SUSE Linux Enterprise Server (SLES) 15 for SAP Applications (BYOS)

Select a SUSE image via Azure CLI

To see a list of the currently available SLES for SAP image for Mv2-series VMs, use the following

az vm image list command:

```
az vm image list --output table --publisher SUSE --sku gen2 --all
```

The command outputs the currently available Generation 2 VMs available from SUSE for Mv2-series VMs.

Example output:

Offer	Publisher	Sku	Urn	Version
SLES-SAP	SUSE	gen2-12-sp4	SUSE:SLES-SAP:gen2-12-sp4:2019.05.13	2019.05.13
SLES-SAP	SUSE	gen2-15	SUSE:SLES-SAP:gen2-15:2019.05.13	2019.05.13
SLES-SAP-BYOS	SUSE	gen2-12-sp4	SUSE:SLES-SAP-BYOS:gen2-12-sp4:2019.05.13	2019.05.13
SLES-SAP-BYOS	SUSE	gen2-15	SUSE:SLES-SAP-BYOS:gen2-15:2019.05.13	2019.05.13

M-series

ACU: 160-180 ¹

Premium Storage: Supported

Premium Storage caching: Supported

Write Accelerator: [Supported](#)

SIZE	VCPUs	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_M8ms ³	8	218.75	256	8	10000 / 100 (793)	5000 / 125	4 / 2000
Standard_M16ms ³	16	437.5	512	16	20000 / 200 (1587)	10000 / 250	8 / 4000
Standard_M32ts	32	192	1024	32	40000 / 400 (3174)	20000 / 500	8 / 8000
Standard_M32ls	32	256	1024	32	40000 / 400 (3174)	20000 / 500	8 / 8000
Standard_M32ms ³	32	875	1024	32	40000 / 400 (3174)	20000 / 500	8 / 8000
Standard_M64s	64	1024	2048	64	80000 / 800 (6348)	40000 / 1000	8 / 16000
Standard_M64ls	64	512	2048	64	80000 / 800 (6348)	40000 / 1000	8 / 16000
Standard_M64ms ³	64	1792	2048	64	80000 / 800 (6348)	40000 / 1000	8 / 16000
Standard_M128s ²	128	2048	4096	64	160000 / 1600 (12696)	80000 / 2000	8 / 30000
Standard_M128ms ^{2, 4}	128	3892	4096	64	160000 / 1600 (12696)	80000 / 2000	8 / 30000

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_M64	64	1024	7168	64	80000 / 800 (1228)	40000 / 1000	8 / 16000
Standard_M64m	64	1792	7168	64	80000 / 800 (1228)	40000 / 1000	8 / 16000
Standard_M128 ²	128	2048	14336	64	250000 / 1600 (2456)	80000 / 2000	8 / 32000
Standard_M128m ²	128	3892	14336	64	250000 / 1600 (2456)	80000 / 2000	8 / 32000

¹ M-series VM's feature Intel® Hyper-Threading Technology

² More than 64 vCPU's require one of these supported guest OSes: Windows Server 2016, Ubuntu 16.04 LTS, SLES 12 SP2, and Red Hat Enterprise Linux, CentOS 7.3 or Oracle Linux 7.3 with LIS 4.2.1.

³ Constrained core sizes available.

⁴ Instance is isolated to hardware dedicated to a single customer.

DSv2-series 11-15

ACU: 210 - 250¹

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D S11_v2 ³	2	14	28	8	8000 / 64 (72)	6400 / 96	2 / 1500
Standard_D S12_v2 ³	4	28	56	16	16000 / 128 (144)	12800 / 192	4 / 3000
Standard_D S13_v2 ³	8	56	112	32	32000 / 256 (288)	25600 / 384	8 / 6000
Standard_D S14_v2 ³	16	112	224	64	64000 / 512 (576)	51200 / 768	8 / 12000

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D S15_v2 ²	20	140	280	64	80000 / 640 (720)	64000 / 960	8 / 25000 ⁴

¹ The maximum disk throughput (IOPS or MBps) possible with a DSv2 series VM may be limited by the number, size and striping of the attached disk(s). For details, see [Designing for high performance](#).

² Instance is isolated to hardware dedicated to a single customer.

³ Constrained core sizes available.

⁴ 25000 Mbps with Accelerated Networking.

Dv2-series 11-15

ACU: 210 - 250

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS / THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D11_v2	2	14	100	6000 / 93 / 46	8 / 8x500	2 / 1500
Standard_D12_v2	4	28	200	12000 / 187 / 93	16 / 16x500	4 / 3000
Standard_D13_v2	8	56	400	24000 / 375 / 187	32 / 32x500	8 / 6000
Standard_D14_v2	16	112	800	48000 / 750 / 375	64 / 64x500	8 / 12000
Standard_D15_v2 ¹	20	140	1000	60000 / 937 / 468	64 / 64x500	8 / 25000 ²

¹ Instance is isolated to hardware dedicated to a single customer.

² 25000 Mbps with Accelerated Networking.

Size table definitions

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller. For example, 1023 GiB = 1098.4 GB
- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6

bytes/sec.

- Data disks can operate in cached or uncached modes. For cached data disk operation, the host cache mode is set to **ReadOnly** or **ReadWrite**. For uncached data disk operation, the host cache mode is set to **None**.
- If you want to get the best performance for your VMs, you should limit the number of data disks to 2 disks per vCPU.
- **Expected network bandwidth** is the maximum aggregated [bandwidth allocated per VM type](#) across all NICs, for all destinations. Upper limits are not guaranteed, but are intended to provide guidance for selecting the right VM type for the intended application. Actual network performance will depend on a variety of factors including network congestion, application loads, and network settings. For information on optimizing network throughput, see [Optimizing network throughput for Windows and Linux](#). To achieve the expected network performance on Linux or Windows, it may be necessary to select a specific version or optimize your VM. For more information, see [How to reliably test for virtual machine throughput](#).

Other sizes

- [General purpose](#)
- [Compute optimized](#)
- [Storage optimized](#)
- [GPU](#)
- [High performance compute](#)
- [Previous generations](#)

Next steps

- Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.
- Learn how to [Create and Manage Linux VMs with the Azure CLI](#)

Constrained vCPU capable VM sizes

3/9/2018 • 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	VCPU	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.

Storage optimized virtual machine sizes

5/9/2019 • 4 minutes to read • [Edit Online](#)

Storage optimized VM sizes offer high disk throughput and IO, and are ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases. Examples include Cassandra, MongoDB, Cloudera and Redis. This article provides information about the number of vCPUs, data disks and NICs as well as local storage throughput and network bandwidth for each optimized size.

The Lsv2-series features high throughput, low latency, directly mapped local NVMe storage running on the [AMD EPYC™ 7551 processor](#) with an all core boost of 2.55GHz and a max boost of 3.0GHz. The Lsv2-series VMs come in sizes from 8 to 80 vCPU in a simultaneous multi-threading configuration. There is 8 GiB of memory per vCPU, and one 1.92TB NVMe SSD M.2 device per 8 vCPUs, with up to 19.2TB (10x1.92TB) available on the L80s v2.

NOTE

The Lsv2-series VMs are optimized to use the local disk on the node attached directly to the VM rather than using durable data disks. This allows for greater IOPs / throughput for your workloads. The Lsv2 and Ls-series do not support the creation of a local cache to increase the IOPs achievable by durable data disks.

The high throughput and IOPs of the local disk makes the Lsv2 and Ls-series VMs ideal for NoSQL stores such as Apache Cassandra and MongoDB which replicate data across multiple VMs to achieve persistence in the event of the failure of a single VM.

To learn more, see [Optimize performance on the Lsv2-series virtual machines](#).

Lsv2-series

ACU: 150-175

Premium Storage: Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY (GiB)	TEMP DISK ¹ (GiB)	NVME DISKS ²	NVME DISK THROUGHPUT ³ (READ IOPS / MBPS)	MAX UNCACHE D DATA DISK THROUGHPUT (IOPS/MBPS) ⁴	MAX DATA DISKS	MAX NICs / EXPECTED NETWORK BANDWID TH (MBPS)
Standard_L8s_v2	8	64	80	1x1.92 TB	400000 / 2000	8000/160	16	2 / 3200
Standard_L16s_v2	16	128	160	2x1.92 TB	800000 / 4000	16000/320	32	4 / 6400
Standard_L32s_v2	32	256	320	4x1.92 TB	1.5M / 8000	32000/640	32	8 / 12800
Standard_L64s_v2	64	512	640	8x1.92 TB	2.9M / 16000	64000/1280	32	8 / 16000+

SIZE	VCPUs	MEMORY (GiB)	TEMP DISK (GiB)	NVME DISKS	NVME DISK THROUGHPUT (READ IOPS / MBPS)	MAX UNCACHE D DATA DISK THROUGHPUT (IOPS/MBPS)	MAX DATA DISKS	MAX NICs / EXPECTED NETWORK BANDWIDTH (Mbps)
Standard_L80s_v2 ⁵	80	640	800	10x1.92TB	3.8M / 20000	80000/1400	32	8 / 16000+

¹ Lsv2-series VMs have a standard SCSI based temp resource disk for OS paging/swap file use (D: on Windows, /dev/sdb on Linux). This disk provides 80 GiB of storage, 4,000 IOPS, and 80 MBps transfer rate for every 8 vCPUs (e.g. Standard_L80s_v2 provides 800 GiB at 40,000 IOPS and 800 MBPS). This ensures the NVMe drives can be fully dedicated to application use. This disk is Ephemeral, and all data will be lost on stop/deallocate.

² Local NVMe Disks are ephemeral, data will be lost on these disks if you stop/deallocate your VM.

³ Hyper-V NVMe Direct technology provides unthrottled access to local NVMe drives mapped securely into the guest VM space. Achieving maximum performance requires using either the latest WS2019 build or Ubuntu 18.04 or 16.04 from the Azure Marketplace. Write performance varies based on IO size, drive load, and capacity utilization.

⁴ Lsv2-series VMs do not provide host cache for data disk as it does not benefit the Lsv2 workloads. However, Lsv2 VMs can accommodate Azure's Ephemeral VM OS disk option (up to 30 GiB).

⁵ VMs with more than 64 vCPUs require one of these supported guest operating systems:

- Windows Server 2016 or later
- Ubuntu 16.04 LTS or later, with Azure tuned kernel (4.15 kernel or later)
- SLES 12 SP2 or later
- RHEL or CentOS version 6.7 thru 6.10, with Microsoft-provided LIS package 4.3.1 (or later) installed
- RHEL or CentOS version 7.3, with Microsoft-provided LIS package 4.2.1 (or later) installed
- RHEL or CentOS version 7.6 or later
- Oracle Linux with UEM4 or later
- Debian 9 with the backports kernel, Debian 10 or later
- CoreOS with a 4.14 kernel or later

Size table definitions

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller. For example, 1023 GiB = 1098.4 GB
- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6 bytes/sec.
- If you want to get the best performance for your VMs, you should limit the number of data disks to 2 disks per vCPU.
- **Expected network bandwidth** is the maximum aggregated [bandwidth allocated per VM type](#) across all NICs, for all destinations. Upper limits are not guaranteed, but are intended to provide guidance for selecting the right VM type for the intended application. Actual network performance will depend on a variety of factors including network congestion, application loads, and network settings. For information on optimizing network throughput, see [Optimizing network throughput for Windows and Linux](#). To achieve the expected network performance on Linux or Windows, it may be necessary to select a specific version or optimize your VM. For more information, see [How to reliably test for virtual machine throughput](#).

Other sizes

- [General purpose](#)
- [Compute optimized](#)
- [Memory optimized](#)
- [GPU](#)
- [High performance compute](#)
- [Previous generations](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

Learn how to [Optimize performance on the Lsv2-series virtual machines](#).

Optimize performance on the Lsv2-series virtual machines

4/19/2019 • 6 minutes to read • [Edit Online](#)

Lsv2-series virtual machines support a variety of workloads that need high I/O and throughput on local storage across a wide range of applications and industries. The Lsv2-series is ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases, including Cassandra, MongoDB, Cloudera, and Redis.

The design of the Lsv2-series Virtual Machines (VMs) maximizes the AMD EPYC™ 7551 processor to provide the best performance between the processor, memory, NVMe devices, and the VMs. In addition to maximizing the hardware performance, Lsv2-series VMs are designed to work with the needs of Linux operating systems for better performance with the hardware and the software.

Tuning the software and hardware resulted in the optimized version of [Canonical's Ubuntu 18.04 and 16.04](#), released in early December 2018 to the Azure Marketplace, which supports maximum performance on the NVMe devices in Lsv2-series VMs.

This article provides tips and suggestions to ensure your workloads and applications achieve the maximum performance designed into the VMs. The information on this page will be continuously updated as more Lsv2 optimized images are added to the Azure Marketplace.

AMD EYPC™ chipset architecture

Lsv2-series VMs use AMD EYPC™ server processors based on the Zen microarchitecture. AMD developed Infinity Fabric (IF) for EYPC™ as scalable interconnect for its NUMA model that could be used for on-die, on-package, and multi-package communications. Compared with QPI (Quick-Path Interconnect) and UPI (Ultra-Path Interconnect) used on Intel modern monolithic-die processors, AMD's many-NUMA small-die architecture may bring both performance benefits as well as challenges. The actual impact of memory bandwidth and latency constraints could vary depending on the type of workloads running.

Tips to maximize performance

- If you are uploading a custom Linux GuestOS for your workload, note that Accelerated Networking will be **OFF** by default. If you intend to enable Accelerated Networking, enable it at the time of VM creation for best performance.
- The hardware that powers the Lsv2-series VMs utilizes NVMe devices with eight I/O Queue Pairs (QPs). Every NVMe device I/O queue is actually a pair: a submission queue and a completion queue. The NVMe driver is set up to optimize the utilization of these eight I/O QPs by distributing I/O's in a round robin schedule. To gain max performance, run eight jobs per device to match.
- Avoid mixing NVMe admin commands (for example, NVMe SMART info query, etc.) with NVMe I/O commands during active workloads. Lsv2 NVMe devices are backed by Hyper-V NVMe Direct technology, which switches into "slow mode" whenever any NVMe admin commands are pending. Lsv2 users could see a dramatic performance drop in NVMe I/O performance if that happens.
- Lsv2 users should not rely on device NUMA information (all 0) reported from within the VM for data drives to decide the NUMA affinity for their apps. The recommended way for better performance is to spread workloads across CPUs if possible.
- The maximum supported queue depth per I/O queue pair for Lsv2 VM NVMe device is 1024 (vs. Amazon

i3 QD 32 limit). Lsv2 users should limit their (synthetic) benchmarking workloads to queue depth 1024 or lower to avoid triggering queue full conditions, which can reduce performance.

Utilizing local NVMe storage

Local storage on the 1.92 TB NVMe disk on all Lsv2 VMs is ephemeral. During a successful standard reboot of the VM, the data on the local NVMe disk will persist. The data will not persist on the NVMe if the VM is redeployed, de-allocated, or deleted. Data will not persist if another issue causes the VM, or the hardware it is running on, to become unhealthy. When this happens, any data on the old host is securely erased.

There will also be cases when the VM needs to be moved to a different host machine, for example, during a planned maintenance operation. Planned maintenance operations and some hardware failures can be anticipated with [Scheduled Events](#). Scheduled Events should be used to stay updated on any predicted maintenance and recovery operations.

In the case that a planned maintenance event requires the VM to be recreated on a new host with empty local disks, the data will need to be resynchronized (again, with any data on the old host being securely erased). This occurs because Lsv2-series VMs do not currently support live migration on the local NVMe disk.

There are two modes for planned maintenance.

Standard VM customer-controlled maintenance

- The VM is moved to an updated host during a 30-day window.
- Lsv2 local storage data could be lost, so backing-up data prior to the event is recommended.

Automatic maintenance

- Occurs if the customer does not execute customer-controlled maintenance, or in the event of emergency procedures such as a security zero-day event.
- Intended to preserve customer data, but there is a small risk of a VM freeze or reboot.
- Lsv2 local storage data could be lost, so backing-up data prior to the event is recommended.

For any upcoming service events, use the controlled maintenance process to select a time most convenient to you for the update. Prior to the event, you may back up your data in premium storage. After the maintenance event completes, you can return your data to the refreshed Lsv2 VMs local NVMe storage.

Scenarios that maintain data on local NVMe disks include:

- The VM is running and healthy.
- The VM is rebooted in place (by you or Azure).
- The VM is paused (stopped without de-allocation).
- The majority of the planned maintenance servicing operations.

Scenarios that securely erase data to protect the customer include:

- The VM is redeployed, stopped (de-allocated), or deleted (by you).
- The VM becomes unhealthy and has to service heal to another node due to a hardware issue.
- A small number of the planned maintenance servicing operations that requires the VM to be reallocated to another host for servicing.

To learn more about options for backing up data in local storage, see [Backup and disaster recovery for Azure IaaS disks](#).

Frequently asked questions

- **How do I start deploying Lsv2-series VMs?**

Much like any other VM, use the [Portal](#), [Azure CLI](#), or [PowerShell](#) to create a VM.

- **Will a single NVMe disk failure cause all VMs on the host to fail?**

If a disk failure is detected on the hardware node, the hardware is in a failed state. When this occurs, all VMs on the node are automatically de-allocated and moved to a healthy node. For Lsv2-series VMs, this means that the customer's data on the failing node is also securely erased and will need to be recreated by the customer on the new node. As noted, before live migration becomes available on Lsv2, the data on the failing node will be proactively moved with the VMs as they are transferred to another node.

- **Do I need to make any adjustments to rq_affinity for performance?**

The rq_affinity setting is a minor adjustment when using the absolute maximum input/output operations per second (IOPS). Once everything else is working well, then try to set rq_affinity to 0 to see if it makes a difference.

- **Do I need to change the blk_mq settings?**

RHEL/CentOS 7.x automatically uses blk-mq for the NVMe devices. No configuration changes or settings are necessary. The scsi_mod.use_blk_mq setting is for SCSI only and was used during Lsv2 Preview because the NVMe devices were visible in the guest VMs as SCSI devices. Currently, the NVMe devices are visible as NVMe devices, so the SCSI blk-mq setting is irrelevant.

- **Do I need to change "fio"?**

To get maximum IOPS with a performance measuring tool like 'fio' in the L64v2 and L80v2 VM sizes, set "rq_affinity" to 0 on each NVMe device. For example, this command line will set "rq_affinity" to zero for all 10 NVMe devices in an L80v2 VM:

```
for i in `seq 0 9`; do echo 0 >/sys/block/nvme${i}n1/queue/rq_affinity; done
```

Also note that the best performance is obtained when I/O is done directly to each of the raw NVMe devices with no partitioning, no file systems, no RAID 0 config, etc. Before starting a testing session, ensure the configuration is in a known fresh/clean state by running `blkdiscard` on each of the NVMe devices.

Next steps

- See specifications for all VMs optimized for storage performance on Azure

GPU optimized virtual machine sizes

6/11/2019 • 10 minutes to read • [Edit Online](#)

GPU optimized VM sizes are specialized virtual machines available with single or multiple NVIDIA GPUs. These sizes are designed for compute-intensive, graphics-intensive, and visualization workloads. This article provides information about the number and type of GPUs, vCPUs, data disks, and NICs. Storage throughput and network bandwidth are also included for each size in this grouping.

- **NC, NCv2, NCv3** sizes are optimized for compute-intensive and network-intensive applications and algorithms. Some examples are CUDA- and OpenCL-based applications and simulations, AI, and Deep Learning. The NCv3-series is focused on high-performance computing workloads featuring NVIDIA's Tesla V100 GPU. The NC-series uses the Intel Xeon E5-2690 v3 2.60GHz v3 (Haswell) processor, and the NCv2-series and NCv3-series VMs use the Intel Xeon E5-2690 v4 (Broadwell) processor.
- **ND, and NDv2** The ND-series is focused on training and inference scenarios for deep learning. It uses the NVIDIA Tesla P40 GPU and the Intel Xeon E5-2690 v4 (Broadwell) processor. The NDv2-series uses the Intel Xeon Platinum 8168 (Skylake) processor.
- **NV and NVv3** sizes are optimized and designed for remote visualization, streaming, gaming, encoding, and VDI scenarios using frameworks such as OpenGL and DirectX. These VMs are backed by the NVIDIA Tesla M60 GPU.

NC-series

Premium Storage: Not Supported

Premium Storage caching: Not Supported

NC-series VMs are powered by the [NVIDIA Tesla K80](#) card and the Intel Xeon E5-2690 v3 (Haswell) processor. Users can crunch through data faster by leveraging CUDA for energy exploration applications, crash simulations, ray traced rendering, deep learning, and more. The NC24r configuration provides a low latency, high-throughput network interface optimized for tightly coupled parallel computing workloads.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	GPU	GPU MEMORY: GIB	MAX DATA DISKS	MAX NICs
Standard_N_C6	6	56	340	1	12	24	1
Standard_N_C12	12	112	680	2	24	48	2
Standard_N_C24	24	224	1440	4	48	64	4
Standard_N_C24r*	24	224	1440	4	48	64	4

1 GPU = one-half K80 card.

*RDMA capable

NCv2-series

Premium Storage: Supported

Premium Storage caching: Supported

NCv2-series VMs are powered by [NVIDIA Tesla P100](#) GPUs. These GPUs can provide more than 2x the computational performance of the NC-series. Customers can take advantage of these updated GPUs for traditional HPC workloads such as reservoir modeling, DNA sequencing, protein analysis, Monte Carlo simulations, and others. In addition to the GPUs, the NCv2-series VMs are also powered by Intel Xeon E5-2690 v4 (Broadwell) CPUs.

The NC24rs v2 configuration provides a low latency, high-throughput network interface optimized for tightly coupled parallel computing workloads.

IMPORTANT

For this size family, the vCPU (core) quota in your subscription is initially set to 0 in each region. [Request a vCPU quota increase](#) for this family in an [available region](#).

SIZE	VCPUs	MEMORY: GIB	TEMP STORAGE (SSD) GIB	GPU	GPU MEMORY: GIB	MAX DATA DISKS	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs
Standard_NC6s_v2	6	112	736	1	16	12	20000/200	4
Standard_NC12s_v2	12	224	1474	2	32	24	40000 / 400	8
Standard_NC24s_v2	24	448	2948	4	64	32	80000 / 800	8
Standard_NC24rs_v2*	24	448	2948	4	64	32	80000 / 800	8

1 GPU = one P100 card.

*RDMA capable

NCv3-series

Premium Storage: Supported

Premium Storage caching: Supported

NCv3-series VMs are powered by [NVIDIA Tesla V100](#) GPUs. These GPUs can provide 1.5x the computational performance of the NCv2-series. Customers can take advantage of these updated GPUs for traditional HPC workloads such as reservoir modeling, DNA sequencing, protein analysis, Monte Carlo simulations, and others. The NC24rs v3 configuration provides a low latency, high-throughput network interface optimized for tightly coupled parallel computing workloads. In addition to the GPUs, the NCv3-series VMs are also powered by Intel Xeon E5-2690 v4 (Broadwell) CPUs.

IMPORTANT

For this size family, the vCPU (core) quota in your subscription is initially set to 0 in each region. [Request a vCPU quota increase](#) for this family in an [available region](#).

SIZE	VCPUs	MEMORY: GiB	TEMP STORAGE (SSD) GiB	GPU	GPU MEMORY: GiB	MAX DATA DISKS	MAX UNCACHE D DISK THROUGHPUT: IOPS / MBPS	MAX NICs
Standard_NC6s_v3	6	112	736	1	16	12	20000 / 200	4
Standard_NC12s_v3	12	224	1474	2	32	24	40000 / 400	8
Standard_NC24s_v3	24	448	2948	4	64	32	80000 / 800	8
Standard_NC24rs_v3*	24	448	2948	4	64	32	80000 / 800	8

1 GPU = one V100 card.

*RDMA capable

NDv2-series (Preview)

Premium Storage: Supported

Premium Storage caching: Supported

Infiniband: Not supported

NDv2-series virtual machine is a new addition to the GPU family designed for the needs of the HPC, AI, and machine learning workloads. It's powered by 8 NVIDIA Tesla V100 NVLINK interconnected GPUs and 40 Intel Xeon Platinum 8168 (Skylake) cores and 672 GiB of system memory. NDv2 instance provides excellent FP32 and FP64 performance for HPC and AI workloads utilizing Cuda, TensorFlow, Pytorch, Caffe, and other frameworks.

[Sign-up and get access to these machines during preview.](#)

SIZE	VCPUs	GPU	MEMORY	NICs (MAX)	TEMP STORAGE (SSD) GiB	MAX. DATA DISKS	MAX UNCACHE D DISK THROUGHPUT: IOPS / MBPS	MAX NETWORK BANDWIDTH
Standard_ND40s_v2	40	8 V100 (NVLink)	672 GiB	8	2948	32	80000 / 800	24000 Mbps

ND-series

Premium Storage: Supported

Premium Storage caching: Supported

The ND-series virtual machines are a new addition to the GPU family designed for AI, and Deep Learning workloads. They offer excellent performance for training and inference. ND instances are powered by [NVIDIA Tesla P40](#) GPUs and Intel Xeon E5-2690 v4 (Broadwell) CPUs. These instances provide excellent performance for single-precision floating point operations, for AI workloads utilizing Microsoft Cognitive Toolkit, TensorFlow, Caffe, and other frameworks. The ND-series also offers a much larger GPU memory size (24 GB), enabling to fit much larger neural net models. Like the NC-series, the ND-series offers a configuration with a secondary low-latency, high-throughput network through RDMA, and InfiniBand connectivity so you can run large-scale training jobs spanning many GPUs.

IMPORTANT

For this size family, the vCPU (core) quota per region in your subscription is initially set to 0. [Request a vCPU quota increase](#) for this family in an [available region](#).

SIZE	VCPUs	MEMORY: GIB	TEMP STORAGE (SSD) GIB	GPU	GPU MEMORY: GIB	MAX DATA DISKS	MAX UNCACHE D DISK THROUGHPUT: IOPS / MBPS	MAX NICs
Standard_ND6s	6	112	736	1	24	12	20000 / 200	4
Standard_ND12s	12	224	1474	2	48	24	40000 / 400	8
Standard_ND24s	24	448	2948	4	96	32	80000 / 800	8
Standard_ND24rs*	24	448	2948	4	96	32	80000 / 800	8

1 GPU = one P40 card.

*RDMA capable

NV-series

Premium Storage: Not Supported

Premium Storage caching: Not Supported

The NV-series virtual machines are powered by [NVIDIA Tesla M60](#) GPUs and NVIDIA GRID technology for desktop accelerated applications and virtual desktops where customers are able to visualize their data or simulations. Users are able to visualize their graphics intensive workflows on the NV instances to get superior graphics capability and additionally run single precision workloads such as encoding and rendering. NV-series VMs are also powered by Intel Xeon E5-2690 v3 (Haswell) CPUs.

Each GPU in NV instances comes with a GRID license. This license gives you the flexibility to use an NV instance as a virtual workstation for a single user, or 25 concurrent users can connect to the VM for a virtual application

scenario.

SIZE	VCPU	MEMORY : GIB	TEMP STORAGE (SSD) GIB	GPU	GPU MEMORY : GIB	MAX DATA DISKS	MAX NICs	VIRTUAL WORKSTATIONS	VIRTUAL APPLICATIONS
Standard_NV6	6	56	340	1	8	24	1	1	25
Standard_NV12	12	112	680	2	16	48	2	2	50
Standard_NV24	24	224	1440	4	32	64	4	4	100

1 GPU = one-half M60 card.

NVv3-series (Preview)¹

Premium Storage: Supported

Premium Storage caching: Supported

The NVv3-series virtual machines are powered by [NVIDIA Tesla M60](#) GPUs and NVIDIA GRID technology with Intel E5-2690 v4 (Broadwell) CPUs. These virtual machines are targeted for GPU accelerated graphics applications and virtual desktops where customers want to visualize their data, simulate results to view, work on CAD, or render and stream content. Additionally, these virtual machines can run single precision workloads such as encoding and rendering. NVv3 virtual machines support Premium Storage and come with twice the system memory (RAM) when compared with its predecessor NV-series.

Each GPU in NVv3 instances comes with a GRID license. This license gives you the flexibility to use an NV instance as a virtual workstation for a single user, or 25 concurrent users can connect to the VM for a virtual application scenario.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	GPU	GPU MEMORY: GIB	MAX DATA DISKS	MAX UNCAC HED DISK THROU GPUT: IOPS / MBPS	MAX NICs	VIRTUA L WORKS TATION S	VIRTUA L APPLIC ATIONS
Standard_NV12s_v3	12	112	320	1	8	12	20000 / 200	4	1	25
Standard_NV24s_v3	24	224	640	2	16	24	40000 / 400	8	2	50
Standard_NV48s_v3	48	448	1280	4	32	32	80000 / 800	8	4	100

1 GPU = one-half M60 card.

¹ NVv3-series VMs feature Intel Hyper-Threading Technology

Size table definitions

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller. For example, 1023 GiB = 1098.4 GB
- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6 bytes/sec.
- Data disks can operate in cached or uncached modes. For cached data disk operation, the host cache mode is set to **ReadOnly** or **ReadWrite**. For uncached data disk operation, the host cache mode is set to **None**.
- If you want to get the best performance for your VMs, you should limit the number of data disks to 2 disks per vCPU.
- **Expected network bandwidth** is the maximum aggregated [bandwidth allocated per VM type](#) across all NICs, for all destinations. Upper limits are not guaranteed, but are intended to provide guidance for selecting the right VM type for the intended application. Actual network performance will depend on a variety of factors including network congestion, application loads, and network settings. For information on optimizing network throughput, see [Optimizing network throughput for Windows and Linux](#). To achieve the expected network performance on Linux or Windows, it may be necessary to select a specific version or optimize your VM. For more information, see [How to reliably test for virtual machine throughput](#).

Supported distributions and drivers

To take advantage of the GPU capabilities of Azure N-series VMs running Linux, 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 the Azure CLI or Azure Resource Manager templates. See the [NVIDIA GPU Driver Extension documentation](#) for supported distributions and deployment steps. For general information about VM extensions, see [Azure virtual machine extensions and features](#).

If you choose to install NVIDIA GPU drivers manually, see [N-series GPU driver setup for Linux](#) for supported distributions, drivers, and installation and verification steps.

Deployment considerations

- For availability of N-series VMs, see [Products available by region](#).
- N-series VMs can only be deployed in the Resource Manager deployment model.
- N-series VMs differ in the type of Azure Storage they support for their disks. NC and NV VMs only support VM disks that are backed by Standard Disk Storage (HDD). NCv2, NCv3, ND, NDv2, and NVv2 VMs only support VM disks that are backed by Premium Disk Storage (SSD).
- If you want to deploy more than a few N-series VMs, consider a pay-as-you-go subscription or other purchase options. If you're using an [Azure free account](#), you can use only a limited number of Azure compute cores.
- You might need to increase the cores quota (per region) in your Azure subscription, and increase the separate quota for NC, NCv2, NCv3, ND, NDv2, NV, or NVv2 cores. To request a quota increase, [open an online customer support request](#) at no charge. Default limits may vary depending on your subscription category.
- You shouldn't install X server or other systems that use the [Nouveau](#) driver on Ubuntu NC VMs. Before installing NVIDIA GPU drivers, you need to disable the [Nouveau](#) driver.

Other sizes

- General purpose
- Compute optimized
- Memory optimized
- Storage optimized
- High performance compute
- Previous generations

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

Install NVIDIA GPU drivers on N-series VMs running Linux

6/3/2019 • 8 minutes to read • [Edit Online](#)

To take advantage of the GPU capabilities of Azure N-series VMs running Linux, 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 the Azure CLI or Azure Resource Manager templates. See the [NVIDIA GPU Driver Extension documentation](#) for supported distributions and deployment steps.

If you choose to install GPU drivers manually, this article provides supported distributions, drivers, and installation and verification steps. Manual driver setup information is also available for [Windows VMs](#).

For N-series VM specs, storage capacities, and disk details, see [GPU Linux VM sizes](#).

Supported distributions and drivers

NVIDIA CUDA drivers

NVIDIA CUDA drivers for NC, NCv2, NCv3, ND, and NDv2-series VMs (optional for NV-series) are supported only on the Linux distributions listed in the following table. CUDA driver information is current at time of publication. For the latest CUDA drivers, visit the [NVIDIA](#) website. Ensure that you install or upgrade to the latest CUDA drivers for your distribution.

TIP

As an alternative to manual CUDA driver installation on a Linux VM, you can deploy an Azure [Data Science Virtual Machine](#) image. The DSVM editions for Ubuntu 16.04 LTS or CentOS 7.4 pre-install NVIDIA CUDA drivers, the CUDA Deep Neural Network Library, and other tools.

DISTRIBUTION	DRIVER
Ubuntu 16.04 LTS, 18.04 LTS	NVIDIA CUDA 10.1, driver branch R418
Red Hat Enterprise Linux 7.3, 7.4, 7.5, 7.6	
CentOS-based 7.3, 7.4, 7.5, 7.6, CentOS-based 7.4 HPC	

NVIDIA GRID drivers

Microsoft redistributes NVIDIA GRID driver installers for NV and NVv2-series VMs used as virtual workstations or for virtual applications. Install only these GRID drivers on Azure NV 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 an NVIDIA vGPU software license server.

DISTRIBUTION	DRIVER
Ubuntu 16.04 LTS, 18.04 LTS	NVIDIA GRID 8.0, driver branch R418
Red Hat Enterprise Linux 7.3, 7.4, 7.5, 7.6	
CentOS-based 7.3, 7.4, 7.5, 7.6	

WARNING

Installation of third-party software on Red Hat products can affect the Red Hat support terms. See the [Red Hat Knowledgebase article](#).

Install CUDA drivers on N-series VMs

Here are steps to install CUDA drivers from the NVIDIA CUDA Toolkit on N-series VMs.

C and C++ developers can optionally install the full Toolkit to build GPU-accelerated applications. For more information, see the [CUDA Installation Guide](#).

To install CUDA drivers, make an SSH connection to each VM. To verify that the system has a CUDA-capable GPU, run the following command:

```
lspci | grep -i NVIDIA
```

You will see output similar to the following example (showing an NVIDIA Tesla K80 card):

```
af8a:00:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
```

Then run installation commands specific for your distribution.

Ubuntu

- Download and install the CUDA drivers from the NVIDIA website. For example, for Ubuntu 16.04 LTS:

```
CUDA_REPO_PKG=cuda-repo-ubuntu1604_10.0.130-1_amd64.deb

wget -O /tmp/${CUDA_REPO_PKG}
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/${CUDA_REPO_PKG}

sudo dpkg -i /tmp/${CUDA_REPO_PKG}

sudo apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80.pub

rm -f /tmp/${CUDA_REPO_PKG}

sudo apt-get update

sudo apt-get install cuda-drivers
```

The installation can take several minutes.

- To optionally install the complete CUDA toolkit, type:

```
sudo apt-get install cuda
```

3. Reboot the VM and proceed to verify the installation.

CUDA driver updates

We recommend that you periodically update CUDA drivers after deployment.

```
sudo apt-get update  
sudo apt-get upgrade -y  
sudo apt-get dist-upgrade -y  
sudo apt-get install cuda-drivers  
sudo reboot
```

CentOS or Red Hat Enterprise Linux

1. Update the kernel (recommended). If you choose not to update the kernel, ensure that the versions of `kernel-devel` and `dkms` are appropriate for your kernel.

```
sudo yum install kernel kernel-tools kernel-headers kernel-devel  
sudo reboot
```

2. Install the latest [Linux Integration Services for Hyper-V and Azure](#).

```
wget https://aka.ms/lis  
tar xvzf lis  
cd LISISO  
sudo ./install.sh  
sudo reboot
```

3. Reconnect to the VM and continue installation with the following commands:

```
sudo rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm  
sudo yum install dkms  
CUDA_REPO_PKG=cuda-repo-rhel7-10.0.130-1.x86_64.rpm  
wget http://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/${CUDA_REPO_PKG} -O /tmp/${CUDA_REPO_PKG}  
sudo rpm -ivh /tmp/${CUDA_REPO_PKG}  
rm -f /tmp/${CUDA_REPO_PKG}  
sudo yum install cuda-drivers
```

The installation can take several minutes.

4. To optionally install the complete CUDA toolkit, type:

```
sudo yum install cuda
```

5. Reboot the VM and proceed to verify the installation.

Verify driver installation

To query the GPU device state, SSH to the VM and run the `nvidia-smi` command-line utility installed with the driver.

If the driver is installed, you will see output similar to the following. Note that **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.

```
Tue Oct 10 20:48:53 2017
+-----+
| NVIDIA-SMI 384.81                    Driver Version: 384.81 |
+-----+
| GPU  Name Persistence-M  Bus-Id Disp.A  Volatile Uncorr. ECC |
| Fan  Temp Perf Pwr:Usage/Cap | Memory-Usage | GPU-Util Compute M. |
|=====+=====+=====+=====+=====+=====+=====+=====+=====|
| 0   Tesla K80      off    000007D1:00:00.0 off        0          |
| N/A  51C     P0    58W / 149W    0MiB / 11439MiB   0%       Default |
+-----+
+-----+
| Processes:                               GPU Memory Usage |
| GPU PID  Type Process name               |
|=====+=====+=====+=====|
| 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 VM scale set. The RDMA network supports Message Passing Interface (MPI) traffic for applications running with Intel MPI 5.x or a later version. Additional requirements follow:

Distributions

Deploy RDMA-capable N-series VMs from one of the images in the Azure Marketplace that supports RDMA connectivity on N-series VMs:

- **Ubuntu 16.04 LTS** - Configure RDMA drivers on the VM and register with Intel to download Intel MPI:

1. Install dapl, rdmacm, ibverbs, and mlx4

```
sudo apt-get update
sudo apt-get install libdap12 libmlx4-1
```

2. In `/etc/waagent.conf`, enable RDMA by uncommenting the following configuration lines. You need root access to edit this file.

```
OS.EnableRDMA=y
OS.UpdateRdmaDriver=y
```

3. Add or change the following memory settings in KB in the `/etc/security/limits.conf` file. You need root access to edit this file. For testing purposes you can set memlock to unlimited. For example:

```
<User or group name> hard memlock unlimited .
```

```
<User or group name> hard     memlock <memory required for your application in KB>  
<User or group name> soft     memlock <memory required for your application in KB>
```

4. Install Intel MPI Library. Either [purchase and download](#) the library from Intel or download the [free evaluation version](#).

```
wget http://registrationcenter-download.intel.com/akdlm/irc_nas/tec/9278/l_mpi_p_5.1.3.223.tgz
```

Only Intel MPI 5.x runtimes are supported.

For installation steps, see the [Intel MPI Library Installation Guide](#).

5. Enable ptrace for non-root non-debugger processes (needed for the most recent versions of Intel MPI).

```
echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope
```

- **CentOS-based 7.4 HPC** - RDMA drivers and Intel MPI 5.1 are installed on the VM.

Install GRID drivers on NV or NVv2-series VMs

To install NVIDIA GRID drivers on NV or NVv2-series VMs, make an SSH connection to each VM and follow the steps for your Linux distribution.

Ubuntu

1. Run the `lspci` command. Verify that the NVIDIA M60 card or cards are visible as PCI devices.
2. Install updates.

```
sudo apt-get update  
sudo apt-get upgrade -y  
sudo apt-get dist-upgrade -y  
sudo apt-get install build-essential ubuntu-desktop -y
```

3. Disable the Nouveau kernel driver, which is incompatible with the NVIDIA driver. (Only use the NVIDIA driver on NV or NVv2 VMs.) To do this, create a file in `/etc/modprobe.d` named `nouveau.conf` with the following contents:

```
blacklist nouveau  
blacklist lbm-nouveau
```

4. Reboot the VM and reconnect. Exit X server:

```
sudo systemctl stop lightdm.service
```

5. Download and install the GRID driver:

```
wget -O NVIDIA-Linux-x86_64-grid.run https://go.microsoft.com/fwlink/?LinkId=874272  
chmod +x NVIDIA-Linux-x86_64-grid.run  
sudo ./NVIDIA-Linux-x86_64-grid.run
```

6. When you're asked whether you want to run the nvidia-xconfig utility to update your X configuration file, select **Yes**.
7. After installation completes, copy /etc/nvidia/gridd.conf.template to a new file gridd.conf at location /etc/nvidia/

```
sudo cp /etc/nvidia/gridd.conf.template /etc/nvidia/gridd.conf
```

8. Add the following to `/etc/nvidia/gridd.conf`:

```
IgnoreSP=FALSE
```

9. Reboot the VM and proceed to verify the installation.

CentOS or Red Hat Enterprise Linux

1. Update the kernel and DKMS (recommended). If you choose not to update the kernel, ensure that the versions of `kernel-devel` and `dkms` are appropriate for your kernel.

```
sudo yum update  
  
sudo yum install kernel-devel  
  
sudo rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm  
  
sudo yum install dkms
```

2. Disable the Nouveau kernel driver, which is incompatible with the NVIDIA driver. (Only use the NVIDIA driver on NV or NV2 VMs.) To do this, create a file in `/etc/modprobe.d` named `nouveau.conf` with the following contents:

```
blacklist nouveau  
  
blacklist lbm-nouveau
```

3. Reboot the VM, reconnect, and install the latest [Linux Integration Services for Hyper-V and Azure](#).

```
wget https://aka.ms/lis  
  
tar xvzf lis  
  
cd LISISO  
  
sudo ./install.sh  
  
sudo reboot
```

4. Reconnect to the VM and run the `lspci` command. Verify that the NVIDIA M60 card or cards are visible as PCI devices.

5. Download and install the GRID driver:

```
wget -O NVIDIA-Linux-x86_64-grid.run https://go.microsoft.com/fwlink/?LinkId=874272  
chmod +x NVIDIA-Linux-x86_64-grid.run  
sudo ./NVIDIA-Linux-x86_64-grid.run
```

6. When you're asked whether you want to run the nvidia-xconfig utility to update your X configuration file, select **Yes**.

7. After installation completes, copy /etc/nvidia/gridd.conf.template to a new file gridd.conf at location /etc/nvidia/

```
sudo cp /etc/nvidia/gridd.conf.template /etc/nvidia/gridd.conf
```

8. Add the following to `/etc/nvidia/gridd.conf`:

```
IgnoreSP=False
```

9. Reboot the VM and proceed to verify the installation.

Verify driver installation

To query the GPU device state, SSH to the VM and run the [nvidia-smi](#) command-line utility installed with the driver.

If the driver is installed, you will see output similar to the following. Note that **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.

```
[azureuser@danlepnvr3 nvidia]$ nvidia-smi  
Wed May 24 00:19:43 2017  
+-----+  
| NVIDIA-SMI 367.92 | Driver Version: 367.92 |  
+-----+  
| GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC |  
| Fan Temp Perf Pwr:Usage/Cap | Memory-Usage | GPU-Util Compute M. |  
+-----+-----+-----+-----+-----+-----+-----+  
| 0 Tesla M60 off | 8342:00:00.0 off | 0MiB / 8123MiB | 0% Default |  
| N/A 31C P0 39W / 150W | | | |  
+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
| Processes: GPU Memory |  
| GPU PID Type Process name Usage |  
+-----+-----+-----+-----+  
| No running processes found |  
+-----+
```

X11 server

If you need an X11 server for remote connections to an NV or NVv2 VM, [x11vnc](#) is recommended because it allows hardware acceleration of graphics. The BusID of the M60 device must be manually added to the X11 configuration file (usually, `/etc/X11/xorg.conf`). Add a "Device" section similar to the following:

```

Section "Device"
    Identifier      "Device0"
    Driver          "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName       "Tesla M60"
    BusID           "PCI:0@your-BusID:0:0"
EndSection

```

Additionally, update your `"Screen"` section to use this device.

The decimal BusID can be found by running

```
nvidia-xconfig --query-gpu-info | awk '/PCI BusID/{print $4}'
```

The BusID can change when a VM gets reallocated or rebooted. Therefore, you may want to create a script to update the BusID in the X11 configuration when a VM is rebooted. For example, create a script named `busidupdate.sh` (or another name you choose) with contents similar to the following:

```

#!/bin/bash
XCONFIG="/etc/X11/xorg.conf"
OLDBUSID=`awk '/BusID/{gsub(/\//, "", $2); print $2}' ${XCONFIG}`
NEWBUSID=`nvidia-xconfig --query-gpu-info | awk '/PCI BusID/{print $4}'` 

if [[ "${OLDBUSID}" == "${NEWBUSID}" ]]; then
    echo "NVIDIA BUSID not changed - nothing to do"
else
    echo "NVIDIA BUSID changed from \"${OLDBUSID}\" to \"${NEWBUSID}\": Updating ${XCONFIG}"
    sed -e 's|BusID.*|BusID      '\\"${NEWBUSID}"'|' -i ${XCONFIG}
fi

```

Then, create an entry for your update script in `/etc/rc.d/rc3.d` so the script is invoked as root on boot.

Troubleshooting

- You can set persistence mode using `nvidia-smi` so the output of the command is faster when you need to query cards. To set persistence mode, execute `nvidia-smi -pm 1`. Note that if the VM is restarted, the mode setting goes away. You can always script the mode setting to execute upon startup.

Next steps

- To capture a Linux VM image with your installed NVIDIA drivers, see [How to generalize and capture a Linux virtual machine](#).

High performance compute virtual machine sizes

6/11/2019 • 10 minutes to read • [Edit Online](#)

Azure H-series virtual machines (VMs) are designed to deliver leadership-class performance, MPI scalability, and cost efficiency for a variety of real-world HPC workloads.

HB-series VMs are optimized for applications driven by memory bandwidth, such as fluid dynamics, explicit finite element analysis, and weather modeling. HB VMs feature 60 AMD EPYC 7551 processor cores, 4 GB of RAM per CPU core, and no hyperthreading. The AMD EPYC platform provides more than 260 GB/sec of memory bandwidth.

HC-series VMs are optimized for applications driven by dense computation, such as implicit finite element analysis, molecular dynamics, and computational chemistry. HC VMs feature 44 Intel Xeon Platinum 8168 processor cores, 8 GB of RAM per CPU core, and no hyperthreading. The Intel Xeon Platinum platform supports Intel's rich ecosystem of software tools such as the Intel Math Kernel Library.

Both HB and HC VMs feature 100 Gb/sec Mellanox EDR InfiniBand in a non-blocking fat tree configuration for consistent RDMA performance. HB and HC VMs support standard Mellanox/OFED drivers such that all MPI types and versions, as well as RDMA verbs, are supported as well.

H-series VMs are optimized for applications driven by high CPU frequencies or large memory per core requirements. H-series VMs feature 8 or 16 Intel Xeon E5 2667 v3 processor cores, 7 or 14 GB of RAM per CPU core, and no hyperthreading. H-series features 56 Gb/sec Mellanox FDR InfiniBand in a non-blocking fat tree configuration for consistent RDMA performance. H-series VMs support Intel MPI 5.x and MS-MPI.

HB-series

ACU: 199-216

Premium Storage: Supported

Premium Storage Caching: Supported

SIZE	VCPU	PROC ESSO R	MEM ORY (GB)	MEM ORY BAND WIDT H GB/S	BASE CPU FREQ UENC Y (GHZ)	ALL- CORE S FREQ UENC Y (GHZ, PEAK)	SING LE- CORE FREQ UENC Y (GHZ, PEAK)	RDM A PERF ORM ANCE (GB/S)	MPI SUPP ORT	TEMP STOR AGE (GB)	MAX DATA DISK S	MAX ETHE RNET NICS
Standard _HB6 0rs	60	AMD EPYC 7551	240	263	2.0	2.55	2.55	100	All	700	4	1

HC-series

ACU: 297-315

Premium Storage: Supported

Premium Storage Caching: Supported

SIZE	VCPU	PROCESSOR	MEMORY (GB)	MEMORY BANDWIDTH GB/S	BASE CPU FREQ (GHZ)	ALL-CORES FREQ UENCY (GHZ, PEAK)	SINGLE-CORE FREQ UENCY (GHZ, PEAK)	RDMA PERFORMANCE (GB/S)	MPI SUPPORT	TEMP STORAGE (GB)	MAX DATA DISKS	MAX ETHERNET NICs
Standard_HC4_4rs	44	Intel Xeon Platinum 8168	352	191	2.7	3.4	3.7	100	All	700	4	1

H-series

ACU: 290-300

Premium Storage: Not Supported

Premium Storage Caching: Not Supported

SIZE	VCPU	PROCESSOR	MEMORY (GB)	MEMORY BANDWIDTH GB/S	BASE CPU FREQ (GHZ)	ALL-CORES FREQ UENCY (GHZ, PEAK)	SINGLE-CORE FREQ UENCY (GHZ, PEAK)	RDMA PERFORMANCE (GB/S)	MPI SUPPORT	TEMP STORAGE (GB)	MAX DATA DISKS	MAX ETHERNET NICs
Standard_H8	8	Intel Xeon E5 2667 v3	56	40	3.2	3.3	3.6	-	Intel 5.x, MS-MPI	1000	32	2
Standard_H16	16	Intel Xeon E5 2667 v3	112	80	3.2	3.3	3.6	-	Intel 5.x, MS-MPI	2000	64	4

SIZE	VCPU	PROC ESSO R	MEM ORY (GB)	MEM BAND WIDT H GB/S	BASE CPU FREQ UENC Y (GHZ)	ALL-CORE S FREQ UENC Y (GHZ, PEAK)	SING LE-CORE FREQ UENC Y (GHZ, PEAK)	RDMA PERFOR MANCE (GB/S)	MPI SUPP ORT	TEMP STOR AGE (GB)	MAX DATA DISK S	MAX ETHE RNET NICs
Standard _H16r ¹	16	Intel Xeon E5 2667 v3	112	80	3.2	3.3	3.6	56	Intel 5.x, MS-MPI	2000	64	4
Standard _H16mr ¹	16	Intel Xeon E5 2667 v3	224	80	3.2	3.3	3.6	56	Intel 5.x, MS-MPI	2000	64	4

¹ For MPI applications, dedicated RDMA backend network is enabled by FDR InfiniBand network.

Size table definitions

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller. For example, 1023 GiB = 1098.4 GB
- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6 bytes/sec.
- Data disks can operate in cached or uncached modes. For cached data disk operation, the host cache mode is set to **ReadOnly** or **ReadWrite**. For uncached data disk operation, the host cache mode is set to **None**.
- If you want to get the best performance for your VMs, you should limit the number of data disks to 2 disks per vCPU.
- Expected network bandwidth** is the maximum aggregated [bandwidth allocated per VM type](#) across all NICs, for all destinations. Upper limits are not guaranteed, but are intended to provide guidance for selecting the right VM type for the intended application. Actual network performance will depend on a variety of factors including network congestion, application loads, and network settings. For information on optimizing network throughput, see [Optimizing network throughput for Windows and Linux](#). To achieve the expected network performance on Linux or Windows, it may be necessary to select a specific version or optimize your VM. For more information, see [How to reliably test for virtual machine throughput](#).

Deployment considerations

- Azure subscription** – To deploy more than a few compute-intensive instances, consider a pay-as-you-go subscription or other purchase options. If you're using an [Azure free account](#), you can use only a limited number of Azure compute cores.
- Pricing and availability** – These VM sizes are offered only in the Standard pricing tier. Check [Products available by region](#) for availability in Azure regions.
- Cores quota** – You might need to increase the cores quota in your Azure subscription from the default value. Your subscription might also limit the number of cores you can deploy in certain VM size families, including the H-series. To request a quota increase, [open an online customer support request](#) at no charge. (Default limits may vary depending on your subscription category.)

NOTE

Contact Azure Support if you have large-scale capacity needs. Azure quotas are credit limits, not capacity guarantees. Regardless of your quota, you are only charged for cores that you use.

- **Virtual network** – An Azure [virtual network](#) is not required to use the compute-intensive instances. However, for many deployments you need at least a cloud-based Azure virtual network, or a site-to-site connection if you need to access on-premises resources. When needed, create a new virtual network to deploy the instances. Adding compute-intensive VMs to a virtual network in an affinity group is not supported.
- **Resizing** – Because of their specialized hardware, you can only resize compute-intensive instances within the same size family (H-series or compute-intensive A-series). For example, you can only resize an H-series VM from one H-series size to another. In addition, resizing from a non-compute-intensive size to a compute-intensive size is not supported.

RDMA-capable instances

A subset of the compute-intensive instances (A8, A9, H16r, H16mr, HB and HC) feature a network interface for remote direct memory access (RDMA) connectivity. Selected N-series sizes designated with 'r' such as the NC24rs configurations (NC24rs_v2 and NC24rs_v3) are also RDMA-capable. This interface is in addition to the standard Azure network interface available to other VM sizes.

This interface allows the RDMA-capable instances to communicate over an InfiniBand (IB) network, operating at EDR rates for HB, HC, FDR rates for H16r, H16mr, and RDMA-capable N-series virtual machines, and QDR rates for A8 and A9 virtual machines. These RDMA capabilities can boost the scalability and performance of certain Message Passing Interface (MPI) applications. For more information on speed, see the details in the tables on this page.

NOTE

In Azure, IP over IB is only supported on the SR-IOV enabled VMs (currently HB and HC). RDMA over IB is supported for all RDMA-capable instances.

MPI

The SR-IOV enabled VM sizes on Azure allow almost any flavor of MPI to be used. On non-SR-IOV enabled VMs, only Intel MPI 5.x versions are supported. Later versions (2017, 2018) of the Intel MPI runtime library may or may not be compatible with the Azure Linux RDMA drivers.

Supported OS images

The Azure Marketplace has many Linux distributions that support RDMA connectivity:

- **CentOS-based HPC** - For non-SR-IOV enabled VMs, CentOS-based version 6.5 HPC or a later version, up to 7.5 are suitable. For H-series VMs, versions 7.1 to 7.5 are recommended. RDMA drivers and Intel MPI 5.1 are installed on the VM. For SR-IOV VMs, CentOS-HPC 7.6 comes optimized and pre-loaded with the RDMA drivers and various MPI packages installed. For other RHEL/CentOS VM images, add the InfiniBandLinux extension to enable InfiniBand. This Linux VM extension installs Mellanox OFED drivers (on SR-IOV VMs) for RDMA connectivity. The following PowerShell cmdlet installs the latest version (version 1.0) of the InfiniBandDriverLinux extension on an existing RDMA-capable VM. The RDMA-capable VM is named *myVM* and is deployed in the resource group named *myResourceGroup* in the *West US* region as follows:

```
Set-AzVMExtension -ResourceGroupName "myResourceGroup" -Location "westus" -VMName "myVM" -  
ExtensionName "InfiniBandDriverLinux" -Publisher "Microsoft.HpcCompute" -Type "InfiniBandDriverLinux"  
-TypeHandlerVersion "1.0"
```

Alternatively, VM extensions can be included in Azure Resource Manager templates for easy deployment with the following JSON element:

```
"properties":{  
    "publisher": "Microsoft.HpcCompute",  
    "type": "InfiniBandDriverLinux",  
    "typeHandlerVersion": "1.0",  
}
```

NOTE

On the CentOS-based HPC images, kernel updates are disabled in the **yum** configuration file. This is because Linux RDMA drivers are distributed as an RPM package, and driver updates might not work if the kernel is updated.

- **SUSE Linux Enterprise Server** - SLES 12 SP3 for HPC, SLES 12 SP3 for HPC (Premium), SLES 12 SP1 for HPC, SLES 12 SP1 for HPC (Premium), SLES 12 SP4 and SLES 15. RDMA drivers are installed and Intel MPI packages are distributed on the VM. Install MPI by running the following command:

```
sudo rpm -v -i --nodeps /opt/intelMPI/intel_mpi_packages/*.rpm
```

- **Ubuntu** - Ubuntu Server 16.04 LTS, 18.04 LTS. Configure RDMA drivers on the VM and register with Intel to download Intel MPI:

1. Install dapl, rdmacm, ibverbs, and mlx4

```
sudo apt-get update  
  
sudo apt-get install libdap12 libmlx4-1
```

2. In /etc/waagent.conf, enable RDMA by uncommenting the following configuration lines. You need root access to edit this file.

```
OS.EnableRDMA=y  
  
OS.UpdateRdmaDriver=y
```

3. Add or change the following memory settings in KB in the /etc/security/limits.conf file. You need root access to edit this file. For testing purposes you can set memlock to unlimited. For example:

```
<User or group name> hard memlock unlimited .
```

```
<User or group name> hard     memlock <memory required for your application in KB>  
  
<User or group name> soft     memlock <memory required for your application in KB>
```

4. Install Intel MPI Library. Either [purchase and download](#) the library from Intel or download the [free evaluation version](#).

```
wget http://registrationcenter-download.intel.com/akdlm/irc_nas/tec/9278/l_mpi_p_5.1.3.223.tgz
```

Only Intel MPI 5.x runtimes are supported.

For installation steps, see the [Intel MPI Library Installation Guide](#).

5. Enable ptrace for non-root non-debugger processes (needed for the most recent versions of Intel MPI).

```
echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope
```

For more details on enabling InfiniBand, setting up MPI, see [Enable InfiniBand](#).

Cluster configuration options

Azure provides several options to create clusters of Linux HPC VMs that can communicate using the RDMA network, including:

- **Virtual machines** - Deploy the RDMA-capable HPC VMs in the same availability set (when you use the Azure Resource Manager deployment model). If you use the classic deployment model, deploy the VMs in the same cloud service.
- **Virtual machine scale sets** - In a virtual machine scale set, ensure that you limit the deployment to a single placement group. For example, in a Resource Manager template, set the `singlePlacementGroup` property to `true`.
- **Azure CycleCloud** - Create an HPC cluster in [Azure CycleCloud](#) to run MPI jobs on Linux nodes.
- **Azure Batch** - Create an [Azure Batch](#) pool to run MPI workloads on Linux compute nodes. For more information, see [Use RDMA-capable or GPU-enabled instances in Batch pools](#). Also see the [Batch Shipyard](#) project, for running container-based workloads on Batch.
- **Microsoft HPC Pack** - [HPC Pack](#) supports several Linux distributions to run on compute nodes deployed in RDMA-capable Azure VMs, managed by a Windows Server head node. For an example deployment, see [Create HPC Pack Linux RDMA Cluster in Azure](#).

Network considerations

- On non-SR-IOV, RDMA-enabled Linux VMs in Azure, eth1 is reserved for RDMA network traffic. Do not change any eth1 settings or any information in the configuration file referring to this network.
- On SR-IOV enabled VMs (HB and HC-series), ib0 is reserved for RDMA network traffic.
- The RDMA network in Azure reserves the address space 172.16.0.0/16. To run MPI applications on instances deployed in an Azure virtual network, make sure that the virtual network address space does not overlap the RDMA network.
- Depending on your choice of cluster management tool, additional system configuration may be needed to run MPI jobs. For example, on a cluster of VMs, you may need to establish trust among the cluster nodes by generating SSH keys or by establishing passwordless SSH logins.

Other sizes

- [General purpose](#)
- [Compute optimized](#)
- [Memory optimized](#)
- [Storage optimized](#)
- [GPU](#)

- [Previous generations](#)

Next steps

- Learn more about how to setup, optimize and scale [HPC workloads](#) on Azure.
- Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

Previous generations of virtual machine sizes

4/12/2019 • 12 minutes to read • [Edit Online](#)

This section provides information on previous generations of virtual machine sizes. These sizes can still be used, but there are newer generations available.

F-series

F-series is based on the 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor, which can achieve clock speeds as high as 3.1 GHz with the Intel Turbo Boost Technology 2.0. This is the same CPU performance as the Dv2-series of VMs.

F-series VMs are an excellent choice for workloads that demand faster CPUs but do not need as much memory or temporary storage per vCPU. Workloads such as analytics, gaming servers, web servers, and batch processing will benefit from the value of the F-series.

ACU: 210 - 250

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPUs	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS / THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_F1	1	2	16	3000 / 46 / 23	4 / 4x500	2 / 750
Standard_F2	2	4	32	6000 / 93 / 46	8 / 8x500	2 / 1500
Standard_F4	4	8	64	12000 / 187 / 93	16 / 16x500	4 / 3000
Standard_F8	8	16	128	24000 / 375 / 187	32 / 32x500	8 / 6000
Standard_F16	16	32	256	48000 / 750 / 375	64 / 64x500	8 / 12000

Fs-series¹

The Fs-series provides all the advantages of the F-series, in addition to Premium storage.

ACU: 210 - 250

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPUs	MEMORY: GiB	TEMP STORAGE (SSD) GiB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GiB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_F_1s	1	2	4	4	4000 / 32 (12)	3200 / 48	2 / 750
Standard_F_2s	2	4	8	8	8000 / 64 (24)	6400 / 96	2 / 1500
Standard_F_4s	4	8	16	16	16000 / 128 (48)	12800 / 192	4 / 3000
Standard_F_8s	8	16	32	32	32000 / 256 (96)	25600 / 384	8 / 6000
Standard_F_16s	16	32	64	64	64000 / 512 (192)	51200 / 768	8 / 12000

MBps = 10^6 bytes per second, and GiB = 1024^3 bytes.

¹ The maximum disk throughput (IOPS or MBps) possible with a Fs series VM may be limited by the number, size, and striping of the attached disk(s). For details, see [Designing for high performance](#).

Ls-series

The Ls-series offers up to 32 vCPUs, using the [Intel® Xeon® processor E5 v3 family](#). The Ls-series gets the same CPU performance as the G/GS-Series and comes with 8 GiB of memory per vCPU.

The Ls-series does not support the creation of a local cache to increase the IOPS achievable by durable data disks. The high throughput and IOPS of the local disk makes Ls-series VMs ideal for NoSQL stores such as Apache Cassandra and MongoDB which replicate data across multiple VMs to achieve persistence in the event of the failure of a single VM.

ACU: 180-240

Premium Storage: Supported

Premium Storage caching: Not Supported

SIZE	VCPUs	MEMORY (GiB)	TEMP STORAGE (GiB)	MAX DATA DISKS	MAX TEMP STORAGE THROUGHPUT (IOPS / MBPS)	MAX UNCACHED DISK THROUGHPUT (IOPS / MBPS)	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_L_4s	4	32	678	16	20000 / 200	5000 / 125	2 / 4000
Standard_L_8s	8	64	1388	32	40000 / 400	10000 / 250	4 / 8000

SIZE	VCPU	MEMORY (GiB)	TEMP STORAGE (GiB)	MAX DATA DISKS	MAX TEMP STORAGE THROUGHPUT (IOPS / MBPS)	MAX UNCACHED DISK THROUGHPUT (IOPS / MBPS)	MAX NICs / EXPECTED NETWORK BANDWIDTH (Mbps)
Standard_L16s	16	128	2807	64	80000 / 800	20000 / 500	8 / 16000
Standard_L32s ¹	32	256	5630	64	160000 / 1600	40000 / 1000	8 / 20000

The maximum disk throughput possible with Ls-series VMs may be limited by the number, size, and striping of any attached disks. For details, see [Designing for high performance](#).

¹ Instance is isolated to hardware dedicated to a single customer.

NVv2-series (Preview)

Newer size recommendation: [NVv3-series \(Preview\)](#)

The NVv2-series virtual machines are powered by [NVIDIA Tesla M60](#) GPUs and NVIDIA GRID technology with Intel Broadwell CPUs. These virtual machines are targeted for GPU accelerated graphics applications and virtual desktops where customers want to visualize their data, simulate results to view, work on CAD, or render and stream content. Additionally, these virtual machines can run single precision workloads such as encoding and rendering. NVv2 virtual machines support Premium Storage and come with twice the system memory (RAM) when compared with its predecessor NV-series.

Each GPU in NVv2 instances comes with a GRID license. This license gives you the flexibility to use an NV instance as a virtual workstation for a single user, or 25 concurrent users can connect to the VM for a virtual application scenario.

SIZE	VCPU	MEMORY : GiB	TEMP STORAGE (SSD) GiB	GPU	GPU MEMORY : GiB	MAX DATA DISKS	MAX NICs	VIRTUAL WORKSTATIONS	VIRTUAL APPLICATIONS
Standard_NV6s_v2	6	112	320	1	8	12	4	1	25
Standard_NV12s_v2	12	224	640	2	16	24	8	2	50
Standard_NV24s_v2	24	448	1280	4	32	32	8	4	100

Standard A0 - A4 using CLI and PowerShell

In the classic deployment model, some VM size names are slightly different in CLI and PowerShell:

- Standard_A0 is ExtraSmall
- Standard_A1 is Small
- Standard_A2 is Medium
- Standard_A3 is Large
- Standard_A4 is ExtraLarge

Size table definitions

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller. For example, 1023 GiB = 1098.4 GB
- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6 bytes/sec.
- Data disks can operate in cached or uncached modes. For cached data disk operation, the host cache mode is set to **ReadOnly** or **ReadWrite**. For uncached data disk operation, the host cache mode is set to **None**.
- If you want to get the best performance for your VMs, you should limit the number of data disks to 2 disks per vCPU.
- **Expected network bandwidth** is the maximum aggregated [bandwidth allocated per VM type](#) across all NICs, for all destinations. Upper limits are not guaranteed, but are intended to provide guidance for selecting the right VM type for the intended application. Actual network performance will depend on a variety of factors including network congestion, application loads, and network settings. For information on optimizing network throughput, see [Optimizing network throughput for Windows and Linux](#). To achieve the expected network performance on Linux or Windows, it may be necessary to select a specific version or optimize your VM. For more information, see [How to reliably test for virtual machine throughput](#).

Older generations of virtual machine sizes

This section provides information on older generations of virtual machine sizes. These sizes are still supported but will not receive additional capacity. There are newer or alternative sizes that are generally available. Please refer to [Sizes for Windows virtual machines in Azure](#) or [Sizes for Linux virtual machines in Azure](#) to choose the VM sizes that will best fit your need.

For more information on resizing a Linux VM, see [Resize a Linux virtual machine using Azure CLI](#). If you're using Windows VMs and prefer to use PowerShell, see [Resize a Windows VM](#).

Basic A

Newer size recommendation: [Av2-series](#)

Premium Storage: Not Supported

Premium Storage caching: Not Supported

The basic tier sizes are primarily for development workloads and other applications that don't require load balancing, auto-scaling, or memory-intensive virtual machines.

SIZE – SIZE\NAME	VCPUs	MEMORY	NICs (MAX)	MAX TEMPORARY DISK SIZE	MAX. DATA DISKS (1023 GB EACH)	MAX. IOPS (300 PER DISK)
A0\Basic_A0	1	768 MB	2	20 GB	1	1x300
A1\Basic_A1	1	1.75 GB	2	40 GB	2	2x300
A2\Basic_A2	2	3.5 GB	2	60 GB	4	4x300
A3\Basic_A3	4	7 GB	2	120 GB	8	8x300
A4\Basic_A4	8	14 GB	2	240 GB	16	16x300

A-series

Newer size recommendation: [Av2-series](#)

ACU: 50-100

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (HDD): GIB	MAX DATA DISKS	MAX DATA DISK THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_A0 ¹	1	0.768	20	1	1x500	2 / 100
Standard_A1	1	1.75	70	2	2x500	2 / 500
Standard_A2	2	3.5	135	4	4x500	2 / 500
Standard_A3	4	7	285	8	8x500	2 / 1000
Standard_A4	8	14	605	16	16x500	4 / 2000
Standard_A5	2	14	135	4	4x500	2 / 500
Standard_A6	4	28	285	8	8x500	2 / 1000
Standard_A7	8	56	605	16	16x500	4 / 2000

¹ The A0 size is over-subscribed on the physical hardware. For this specific size only, other customer deployments may impact the performance of your running workload. The relative performance is outlined below as the expected baseline, subject to an approximate variability of 15 percent.

A-series - compute-intensive instances

Newer size recommendation: [Av2-series](#)

ACU: 225

Premium Storage: Not Supported

Premium Storage caching: Not Supported

The A8-A11 and H-series sizes are also known as *compute-intensive instances*. The hardware that runs these sizes is designed and optimized for compute-intensive and network-intensive applications, including high-performance computing (HPC) cluster applications, modeling, and simulations. The A8-A11 series uses Intel Xeon E5-2670 @ 2.6 GHZ and the H-series uses Intel Xeon E5-2667 v3 @ 3.2 GHz.

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (HDD): GIB	MAX DATA DISKS	MAX DATA DISK THROUGHPUT: IOPS	MAX NICs
Standard_A8 ¹	8	56	382	32	32x500	2
Standard_A9 ¹	16	112	382	64	64x500	4
Standard_A10	8	56	382	32	32x500	2
Standard_A11	16	112	382	64	64x500	4

¹For MPI applications, dedicated RDMA backend network is enabled by FDR InfiniBand network, which delivers ultra-low-latency and high bandwidth.

D-series

Newer size recommendation: [Dv3-series](#)

ACU: 160-250¹

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS / THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D1	1	3.5	50	3000 / 46 / 23	4 / 4x500	2 / 500
Standard_D2	2	7	100	6000 / 93 / 46	8 / 8x500	2 / 1000
Standard_D3	4	14	200	12000 / 187 / 93	16 / 16x500	4 / 2000
Standard_D4	8	28	400	24000 / 375 / 187	32 / 32x500	8 / 4000

¹ VM Family can run on one of the following CPU's: 2.2 GHz Intel Xeon® E5-2660 v2, 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) or 2.3 GHz Intel XEON® E5-2673 v4 (Broadwell)

D-series - memory optimized

Newer size recommendation: [Dv3-series](#)

ACU: 160-250¹

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS / THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D11	2	14	100	6000 / 93 / 46	8 / 8x500	2 / 1000
Standard_D12	4	28	200	12000 / 187 / 93	16 / 16x500	4 / 2000
Standard_D13	8	56	400	24000 / 375 / 187	32 / 32x500	8 / 4000
Standard_D14	16	112	800	48000 / 750 / 375	64 / 64x500	8 / 8000

¹ VM Family can run on one of the following CPU's: 2.2 GHz Intel Xeon® E5-2660 v2, 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) or 2.3 GHz Intel XEON® E5-2673 v4 (Broadwell)

DS-series

Newer size recommendation: [DSv3-series](#)

ACU: 160-250 ¹

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D S1	1	3.5	7	4	4,000 / 32 (43)	3,200 / 32	2 / 500
Standard_D S2	2	7	14	8	8,000 / 64 (86)	6,400 / 64	2 / 1000
Standard_D S3	4	14	28	16	16,000 / 128 (172)	12,800 / 128	4 / 2000
Standard_D S4	8	28	56	32	32,000 / 256 (344)	25,600 / 256	8 / 4000

¹ VM Family can run on one of the following CPU's: 2.2 GHz Intel Xeon® E5-2660 v2, 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) or 2.3 GHz Intel XEON® E5-2673 v4 (Broadwell)

DS-series - memory optimized

Newer size recommendation: [DSv3-series](#)

ACU: 160-250^{1,2}

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_D S11	2	14	28	8	8,000 / 64 (72)	6,400 / 64	2 / 1000
Standard_D S12	4	28	56	16	16,000 / 128 (144)	12,800 / 128	4 / 2000
Standard_D S13	8	56	112	32	32,000 / 256 (288)	25,600 / 256	8 / 4000
Standard_D S14	16	112	224	64	64,000 / 512 (576)	51,200 / 512	8 / 8000

¹ The maximum disk throughput (IOPS or MBps) possible with a DS series VM may be limited by the number, size and striping of the attached disk(s). For details, see [Designing for high performance](#).

² VM Family can run on one of the following CPU's: 2.2 GHz Intel Xeon® E5-2660 v2, 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) or 2.3 GHz Intel XEON® E5-2673 v4 (Broadwell)

GS-series

ACU: 180 - 240¹

Premium Storage: Supported

Premium Storage caching: Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_G S1	2	28	56	8	10,000 / 100 (264)	5,000 / 125	2 / 2000
Standard_G S2	4	56	112	16	20,000 / 200 (528)	10,000 / 250	2 / 4000

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX UNCACHED DISK THROUGHPUT: IOPS / MBPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_G S3	8	112	224	32	40,000 / 400 (1,056)	20,000 / 500	4 / 8000
Standard_G S4 ³	16	224	448	64	80,000 / 800 (2,112)	40,000 / 1,000	8 / 16000
Standard_G S5 ^{2,3}	32	448	896	64	160,000 / 1,600 (4,224)	80,000 / 2,000	8 / 20000

¹ The maximum disk throughput (IOPS or MBps) possible with a GS series VM may be limited by the number, size and striping of the attached disk(s). For details, see [Designing for high performance](#).

² Instance is isolated to hardware dedicated to a single customer.

³ Constrained core sizes available.

G-series

ACU: 180 - 240

Premium Storage: Not Supported

Premium Storage caching: Not Supported

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS / THROUGHPUT: IOPS	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_G1	2	28	384	6000 / 93 / 46	8 / 8 x 500	2 / 2000
Standard_G2	4	56	768	12000 / 187 / 93	16 / 16 x 500	2 / 4000
Standard_G3	8	112	1,536	24000 / 375 / 187	32 / 32 x 500	4 / 8000
Standard_G4	16	224	3,072	48000 / 750 / 375	64 / 64 x 500	8 / 16000
Standard_G5 ¹	32	448	6,144	96000 / 1500 / 750	64 / 64 x 500	8 / 20000

¹ Instance is isolated to hardware dedicated to a single customer.

Other sizes

- [General purpose](#)
- [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.

Azure compute unit (ACU)

2/15/2019 • 2 minutes to read • [Edit Online](#)

The concept of the Azure Compute Unit (ACU) provides a way of comparing compute (CPU) performance across Azure SKUs. This will help you easily identify which SKU is most likely to satisfy your performance needs. ACU is currently standardized on a Small (Standard_A1) VM being 100 and all other SKUs then represent approximately how much faster that SKU can run a standard benchmark.

IMPORTANT

The ACU is only a guideline. The results for your workload may vary.

SKU FAMILY	ACU \ VCPU	VCPUs: CORE
A0	50	1:1
A1 - A4	100	1:1
A5 - A7	100	1:1
A1_v2 - A8_v2	100	1:1
A2m_v2 - A8m_v2	100	1:1
A8 - A11	225*	1:1
D1 - D14	160 - 250	1:1
D1_v2 - D15_v2	210 - 250*	1:1
DS1 - DS14	160 - 250	1:1
DS1_v2 - DS15_v2	210 - 250*	1:1
D_v3	160 - 190*	2:1***
Ds_v3	160 - 190*	2:1***
E_v3	160 - 190*	2:1***
Es_v3	160 - 190*	2:1***
F2s_v2 - F72s_v2	195 - 210*	2:1***
F1 - F16	210 - 250*	1:1
F1s - F16s	210 - 250*	1:1

SKU FAMILY	ACU \ VCPU	VCPU: CORE
G1 - G5	180 - 240*	1:1
GS1 - GS5	180 - 240*	1:1
H	290 - 300*	1:1
HB	199 - 216**	1:1
HC	297 - 315*	1:1
L4s - L32s	180 - 240*	1:1
L8s_v2 - L80s_v2	150 - 175**	2:1
M	160 - 180	2:1***

*ACUs use Intel® Turbo technology to increase CPU frequency and provide a performance increase. The amount of the performance increase can vary based on the VM size, workload, and other workloads running on the same host.

**ACUs use AMD® Boost technology to increase CPU frequency and provide a performance increase. The amount of the performance increase can vary based on the VM size, workload, and other workloads running on the same host.

***Hyper-threaded and capable of running nested virtualization

Here are links to more information about the different sizes:

- [General-purpose](#)
- [Memory optimized](#)
- [Compute optimized](#)
- [GPU optimized](#)
- [High performance compute](#)
- [Storage optimized](#)

Compute benchmark scores for Linux VMs

4/28/2019 • 21 minutes to read • [Edit Online](#)

The following CoreMark benchmark scores show compute performance for Azure's high-performance VM lineup running Ubuntu. Compute benchmark scores are also available for [Windows VMs](#).

Av2 - General Compute

(3/15/2019 12:06:55 AM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_A1_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	1	1	1.9	6,483	120	1.85%	273
Standard_A1_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	1.9	6,059	208	3.43%	217
Standard_A1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	1.9	6,367	453	7.12%	217
Standard_A2_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	3.9	13,161	194	1.48%	266
Standard_A2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.9	12,067	401	3.32%	203
Standard_A2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	3.9	12,527	797	6.37%	238

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_A2m_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	15.7	13,167	179	1.36%	273
Standard_A2m_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	15.7	12,133	336	2.77%	210
Standard_A2m_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	15.7	12,401	656	5.29%	224
Standard_A4_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	7.8	26,307	231	0.88%	231
Standard_A4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	7.8	24,552	720	2.93%	224
Standard_A4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	7.8	24,963	1,625	6.51%	252
Standard_A4m_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	31.4	26,238	292	1.11%	259
Standard_A4m_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	31.4	24,250	491	2.02%	189

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_A4m_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	31.4	24,725	1,553	6.28%	259
Standard_A8_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	15.7	53,237	687	1.29%	266
Standard_A8_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	15.7	49,655	585	1.18%	147
Standard_A8_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	15.7	49,005	2,162	4.41%	294
Standard_A8m_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	2	62.9	52,627	902	1.71%	266
Standard_A8m_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	62.9	49,838	633	1.27%	182
Standard_A8m_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	62.9	49,123	2,483	5.05%	259

B - Burstable

(3/15/2019 12:27:08 AM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
---------	-----	-------	------------	-------------	-----------	--------	---------	-------

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_B1ms	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	1.9	13,593	307	2.26%	28
Standard_B1ms	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	1.9	14,069	495	3.52%	672
Standard_B1s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	0.9	13,736	211	1.54%	28
Standard_B1s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	0.9	13,965	457	3.27%	672
Standard_B2ms	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	7.8	27,361	1,110	4.06%	28
Standard_B2ms	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	7.8	27,432	771	2.81%	672
Standard_B2s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.9	27,488	822	2.99%	28
Standard_B2s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	3.9	27,548	864	3.14%	672

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_B4ms	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	15.7	54,951	1,868	3.40%	28
Standard_B4ms	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	15.7	54,051	1,260	2.33%	672
Standard_B8ms	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	31.4	111,929	1,562	1.40%	35
Standard_B8ms	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	31.4	109,537	1,354	1.24%	665

DSv3 - General Compute + Premium Storage

(3/12/2019 6:52:03 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_D2s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	7.8	20,153	838	4.16%	147
Standard_D2s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	7.8	20,903	1,324	6.33%	553
Standard_D4s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	15.7	39,502	1,257	3.18%	189

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D4s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	15.7	40,547	1,935	4.77%	511
Standard_D8s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	31.4	80,191	1,054	1.31%	168
Standard_D8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	31.4	79,884	3,073	3.85%	532
Standard_D16s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	62.9	160,319	1,213	0.76%	105
Standard_D16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	62.9	156,325	2,176	1.39%	588
Standard_D32s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	32	2	125.9	315,457	2,647	0.84%	105
Standard_D32s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	1	125.9	312,058	1,661	0.53%	595
Standard_D64s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	251.9	627,378	4,447	0.71%	700

Dv3 - General Compute

(3/12/2019 6:54:27 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D2_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	7.8	20,359	799	3.93%	154
Standard_D2_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	7.8	20,737	1,422	6.86%	546
Standard_D4_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	15.7	40,095	1,501	3.74%	147
Standard_D4_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	15.7	41,147	2,706	6.58%	546
Standard_D8_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	31.4	80,383	1,486	1.85%	133
Standard_D8_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	31.4	80,511	3,916	4.86%	560
Standard_D16_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	62.9	160,932	2,200	1.37%	140

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D16_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	62.9	158,679	4,550	2.87%	560
Standard_D32_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	32	2	125.9	314,208	4,250	1.35%	189
Standard_D32_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	1	125.9	312,472	3,173	1.02%	511
Standard_D64_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	251.9	627,470	9,651	1.54%	700

DSv2 - Storage Optimized

(3/15/2019 12:53:13 AM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS1_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	3.4	14,642	600	4.10%	259
Standard_DS1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	3.4	14,808	904	6.10%	434
Standard_DS2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	6.8	28,654	877	3.06%	301

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_DS2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	6.8	29,089	1,421	4.89%	406
Standard_DS3_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	13.7	57,255	1,633	2.85%	238
Standard_DS3_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	13.7	57,255	2,265	3.96%	462
Standard_DS4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	27.5	116,681	1,097	0.94%	231
Standard_DS4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	27.5	112,512	1,261	1.12%	462
Standard_DS5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	55.0	225,661	2,370	1.05%	189
Standard_DS5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	55.0	229,145	2,878	1.26%	21
Standard_DS5_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	55.0	226,818	1,797	0.79%	497

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS11_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	13.7	28,571	920	3.22%	238
Standard_DS11_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	13.7	29,049	1,614	5.56%	469
Standard_DS11-1_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	13.7	14,594	617	4.23%	287
Standard_DS11-1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	13.7	14,951	852	5.70%	413
Standard_DS12_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	27.5	57,503	1,398	2.43%	217
Standard_DS12_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	27.5	57,082	2,372	4.16%	483
Standard_DS12-1_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	27.5	14,698	564	3.84%	238
Standard_DS12-1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	27.5	15,127	941	6.22%	462

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS12-2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	27.5	28,711	981	3.42%	259
Standard_DS12-2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	27.5	29,305	1,241	4.24%	441
Standard_DS13_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	55.0	116,875	1,286	1.10%	203
Standard_DS13_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	55.0	112,318	1,356	1.21%	504
Standard_DS13-2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	55.0	29,105	1,154	3.97%	224
Standard_DS13-2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	55.0	29,936	1,720	5.75%	483
Standard_DS13-4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	55.0	56,992	1,814	3.18%	280
Standard_DS13-4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	55.0	57,781	2,122	3.67%	427

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS14_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	110.2	224,149	3,450	1.54%	196
Standard_DS14_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	110.2	227,108	1,267	0.56%	504
Standard_DS14-4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	2	110.2	56,211	2,154	3.83%	189
Standard_DS14-4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	110.2	59,651	2,560	4.29%	518
Standard_DS14-8_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	2	110.2	112,280	4,430	3.95%	196
Standard_DS14-8_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	110.2	113,375	1,442	1.27%	511
Standard_DS15_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	20	2	137.7	279,359	4,032	1.44%	665

Dv2 - General Compute

(3/12/2019 6:53:48 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D1_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	3.4	14,730	663	4.50%	385
Standard_D1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	3.4	15,057	1,319	8.76%	322
Standard_D2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	6.8	29,395	1,073	3.65%	329
Standard_D2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	6.8	29,564	2,145	7.26%	378
Standard_D3_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	13.7	58,150	1,340	2.30%	343
Standard_D3_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	13.7	57,820	2,944	5.09%	364
Standard_D4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	27.5	117,448	1,612	1.37%	308
Standard_D4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	27.5	114,082	3,369	2.95%	399

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	55.0	226,370	4,722	2.09%	147
Standard_D5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	55.0	225,035	5,026	2.23%	119
Standard_D5_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	55.0	227,883	3,259	1.43%	441
Standard_D11_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	13.7	29,260	1,012	3.46%	308
Standard_D11_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	13.7	29,306	1,763	6.02%	399
Standard_D12_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	27.5	58,322	1,391	2.39%	329
Standard_D12_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	27.5	57,999	3,533	6.09%	371
Standard_D13_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	55.0	117,218	1,514	1.29%	329

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D13_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	55.0	114,344	3,307	2.89%	378
Standard_D14_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	110.2	224,348	5,477	2.44%	280
Standard_D14_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	110.2	228,221	2,733	1.20%	427
Standard_D15_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	20	2	137.7	281,494	7,976	2.83%	672

Esv3 - Memory Optimized + Premium Storage

(3/12/2019 7:17:33 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_E2s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	15.7	20,957	1,200	5.73%	672
Standard_E4s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	31.4	40,420	1,993	4.93%	672
Standard_E4-2s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	31.4	20,774	1,133	5.45%	672

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_E8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	62.9	80,153	3,308	4.13%	665
Standard_E8-2s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	62.9	21,178	1,334	6.30%	665
Standard_E8-4s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	62.9	40,614	2,216	5.46%	672
Standard_E16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	125.9	156,137	2,160	1.38%	672
Standard_E16-4s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	125.9	41,950	2,309	5.50%	637
Standard_E16-8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	125.9	81,196	3,179	3.91%	658
Standard_E20s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	20	1	157.4	196,619	1,325	0.67%	672
Standard_E32s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	251.9	304,707	5,719	1.88%	672

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_E32-8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	2	251.9	83,576	3,693	4.42%	672
Standard_E32-16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	2	251.9	158,023	4,317	2.73%	672
Standard_E64s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	425.2	628,540	3,982	0.63%	49
Standard_E64-16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	2	425.2	169,611	3,265	1.92%	42
Standard_E64-32s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	425.2	307,584	3,569	1.16%	56

Eisv3 - Memory Opt + Premium Storage (isolated)

(4/11/2019 10:07:29 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_E64is_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	425.2	627,745	4,062	0.65%	196

Ev3 - Memory Optimized

(3/12/2019 6:52:13 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_E2_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	15.7	21,171	1,772	8.37%	693
Standard_E4_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	31.4	41,181	3,148	7.64%	700
Standard_E8_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	62.9	81,211	5,055	6.22%	700
Standard_E16_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	125.9	158,152	4,033	2.55%	700
Standard_E20_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	20	1	157.4	197,739	2,731	1.38%	693
Standard_E32_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	251.9	307,286	8,353	2.72%	700
Standard_E64_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	425.2	628,451	9,235	1.47%	707

EiV3 - Memory Optimized (isolated)

(3/12/2019 6:57:51 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_E64i_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	425.2	625,855	4,881	0.78%	7
Standard_E64i_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	425.2	629,151	9,756	1.55%	217

Fsv2 - Compute + Storage Optimized

(3/12/2019 6:51:35 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_F2s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	2	1	3.9	28,219	1,843	6.53%	700
Standard_F4s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	4	1	7.8	53,911	1,002	1.86%	707
Standard_F8s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	8	1	15.7	106,467	1,101	1.03%	707
Standard_F16s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	16	1	31.4	211,311	1,724	0.82%	707
Standard_F32s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	32	1	62.9	423,175	4,346	1.03%	707

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_F64s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	64	2	125.9	829,537	21,574	2.60%	707
Standard_F72s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	72	2	141.7	933,800	26,783	2.87%	707

Fs - Compute and Storage Optimized

(3/15/2019 12:12:51 AM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_F1s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	1.9	14,552	504	3.46%	350
Standard_F1s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	1.9	14,784	858	5.80%	357
Standard_F2s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.9	28,664	895	3.12%	245
Standard_F2s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	3.9	29,188	1,228	4.21%	455
Standard_F4s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	7.8	57,192	1,700	2.97%	259

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_F4s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	7.8	57,412	2,215	3.86%	448
Standard_F8s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	15.7	117,008	1,139	0.97%	259
Standard_F8s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	15.7	112,610	1,595	1.42%	441
Standard_F16s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	31.4	225,444	2,328	1.03%	210
Standard_F16s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	31.4	228,919	3,380	1.48%	28
Standard_F16s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	31.4	227,015	1,543	0.68%	462

F - Compute Optimized

(3/12/2019 6:53:59 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_F1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	1.9	14,937	593	3.97%	350

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_F1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	1.9	15,460	1,326	8.58%	350
Standard_F2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.9	29,324	1,196	4.08%	343
Standard_F2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	3.9	29,299	1,908	6.51%	364
Standard_F4	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	7.8	58,314	1,245	2.14%	364
Standard_F4	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	7.8	58,280	3,581	6.14%	336
Standard_F8	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	15.7	117,516	1,460	1.24%	308
Standard_F8	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	15.7	114,361	3,868	3.38%	399
Standard_F16	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	31.4	226,487	4,140	1.83%	154

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_F16	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	31.4	226,683	4,723	2.08%	133
Standard_F16	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	31.4	228,592	2,371	1.04%	392

GS - Storage Optimized

(3/12/2019 10:22:33 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_GS1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	2	1	27.5	28,835	2,222	7.71%	287
Standard_GS2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	4	1	55.0	55,568	3,139	5.65%	287
Standard_GS3	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	110.2	106,567	2,188	2.05%	287
Standard_GS4	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	1	220.4	210,586	4,130	1.96%	287
Standard_GS4-4	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	4	1	220.4	58,598	2,670	4.56%	287

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_GS4-8	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	220.4	108,234	2,392	2.21%	287
Standard_GS5	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	32	2	440.9	399,835	8,694	2.17%	287
Standard_GS5-8	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	2	440.9	116,643	2,354	2.02%	287
Standard_GS5-16	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	2	440.9	210,984	2,995	1.42%	287

G - Compute Optimized

(3/12/2019 10:23:51 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_G1	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	2	1	27.5	32,808	2,679	8.17%	287
Standard_G2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	4	1	55.0	62,907	4,465	7.10%	287
Standard_G3	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	110.2	113,471	4,346	3.83%	287

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_G4	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	1	220.4	212,092	2,857	1.35%	287
Standard_G5	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	32	2	440.9	403,315	6,947	1.72%	273

H - High Performance Compute (HPC)

(3/12/2019 10:50:51 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_H8	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	8	1	55.0	149,859	734	0.49%	175
Standard_H8m	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	8	1	110.2	149,931	657	0.44%	147
Standard_H16	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	110.2	282,083	6,738	2.39%	84
Standard_H16m	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	220.4	282,106	7,013	2.49%	84
Standard_H16mr	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	220.4	282,167	6,889	2.44%	84

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_H16r	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	110.2	280,837	6,587	2.35%	84

Lv2 - Storage Optimized

(3/14/2019 5:49:04 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_L8s_v2	AMD EPYC 7551 32-Core Processor	8	1	62.9	80,528	404	0.50%	119
Standard_L16s_v2	AMD EPYC 7551 32-Core Processor	16	2	125.9	154,829	3,708	2.40%	119
Standard_L32s_v2	AMD EPYC 7551 32-Core Processor	32	4	251.9	310,811	7,751	2.49%	119
Standard_L64s_v2	AMD EPYC 7551 32-Core Processor	64	8	503.9	595,140	14,572	2.45%	112
Standard_L80s_v2	AMD EPYC 7551 32-Core Processor	80	10	629.9	773,171	19,559	2.53%	119

Ls - Storage Optimized

(3/12/2019 10:22:29 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
---------	-----	-------	------------	-------------	-----------	--------	---------	-------

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_L4s	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	4	1	31.4	56,488	2,916	5.16%	287
Standard_L8s	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	62.9	107,017	2,323	2.17%	287
Standard_L16s	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	1	125.9	210,865	3,653	1.73%	280
Standard_L32s	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	32	2	251.9	399,963	9,254	2.31%	287

M - Memory Optimized

(4/11/2019 7:30:39 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_M8-2ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	2	1	215.2	22,605	29	0.13%	42
Standard_M8-4ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	4	1	215.2	44,488	183	0.41%	42
Standard_M16-4ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	4	1	430.6	44,451	269	0.61%	42

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_M16-8ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	8	1	430.6	88,238	1,243	1.41%	42
Standard_M32-8ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	8	1	861.2	88,521	1,353	1.53%	42
Standard_M32-16ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	16	1	861.2	174,674	3,104	1.78%	42
Standard_M64	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	1,007.9	683,022	11,929	1.75%	42
Standard_M64-16ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	16	2	1,763.9	169,386	4,737	2.80%	42
Standard_M64-32ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	32	2	1,763.9	337,599	4,738	1.40%	42
Standard_M64m	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	1,763.9	677,466	14,478	2.14%	42
Standard_M64ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	1,763.9	675,342	16,577	2.45%	42

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_M64s	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	1,007.9	674,785	15,983	2.37%	42
Standard_M128	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	128	4	2,015.9	1,334,218	21,126	1.58%	42
Standard_M128-32ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	32	4	3,831.1	334,873	5,005	1.49%	42
Standard_M128-64ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	4	3,831.1	667,808	18,145	2.72%	42
Standard_M128m	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	128	4	3,831.1	1,335,873	19,642	1.47%	42
Standard_M128ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	128	4	3,831.1	1,329,151	24,295	1.83%	42
Standard_M128s	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	128	4	2,015.9	1,329,923	20,117	1.51%	42
Standard_M16ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	16	1	430.6	174,686	2,704	1.55%	35

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_M32ls	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	32	1	251.9	344,069	3,372	0.98%	42
Standard_M32ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	32	1	861.2	343,226	4,884	1.42%	84
Standard_M32ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	32	2	861.2	336,526	4,396	1.31%	7
Standard_M32ts	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	32	1	188.9	341,112	5,491	1.61%	35
Standard_M64ls	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	503.9	676,026	18,078	2.67%	42
Standard_M8ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	8	1	215.2	88,066	1,391	1.58%	42

NCSv3 - GPU Enabled

(3/21/2019 5:48:37 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_NC6s_v3	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	6	1	110.2	106,929	353	0.33%	49

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_NC12s_v3	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	12	1	220.4	213,585	875	0.41%	42
Standard_NC24rs_v3	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	24	2	440.9	403,554	6,705	1.66%	42
Standard_NC24s_v3	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	24	2	440.9	403,874	7,603	1.88%	42

NCSV2 - GPU Enabled

(3/12/2019 11:19:19 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	Avg Score	STDDEV	STDDEV%	#RUNS
Standard_NC6s_v2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	6	1	110.2	107,115	321	0.30%	63
Standard_NC12s_v2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	12	1	220.4	213,814	656	0.31%	63
Standard_NC24rs_v2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	24	2	440.9	401,728	6,995	1.74%	63
Standard_NC24s_v2	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	24	2	440.9	402,808	7,923	1.97%	63

NC - GPU Enabled

(3/12/2019 11:08:03 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_NC6	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	6	1	55.0	102,211	658	0.64%	259
Standard_NC12	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	12	1	110.2	203,523	2,293	1.13%	259
Standard_NC24	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	24	2	220.4	382,897	8,712	2.28%	259
Standard_NC24r	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	24	2	220.4	383,171	9,166	2.39%	259

NDs- GPU Enabled

(3/12/2019 11:19:10 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_ND6s	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	6	1	110.2	107,095	353	0.33%	63
Standard_ND12s	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	12	1	220.4	212,298	3,457	1.63%	63

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_ND24rs	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	24	2	440.9	402,749	7,838	1.95%	56
Standard_ND24s	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz	24	2	440.9	401,822	7,776	1.94%	63

NV - GPU Enabled

(3/12/2019 11:08:13 PM pbi 3897709)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_NV6	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	6	1	55.0	101,728	2,094	2.06%	259
Standard_NV12	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	12	1	110.2	203,903	1,724	0.85%	252
Standard_NV24	Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz	24	2	220.4	379,879	8,737	2.30%	259

About CoreMark

Linux numbers were computed by running [CoreMark](#) on Ubuntu. CoreMark was configured with the number of threads set to the number of virtual CPUs, and concurrency set to PThreads. The target number of iterations was adjusted based on expected performance to provide a runtime of at least 20 seconds (typically much longer). The final score represents the number of iterations completed divided by the number of seconds it took to run the test. Each test was run at least seven times on each VM. Test run dates shown above. Tests run on multiple VMs across Azure public regions the VM was supported in on the date run. Basic A and B (Burstable) series not shown because performance is variable. N series not shown as they are GPU centric and Coremark doesn't measure GPU performance.

Next steps

- For storage capacities, disk details, and additional considerations for choosing among VM sizes, see [Sizes for virtual machines](#).
- To run the CoreMark scripts on Linux VMs, download the [CoreMark script pack](#).

Endorsed Linux distributions on Azure

5/17/2019 • 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+	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+	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.

Maintenance for virtual machines in Azure

5/2/2019 • 5 minutes to read • [Edit Online](#)

Azure periodically updates its platform to improve the reliability, performance, and security of the host infrastructure for virtual machines. The purpose of these updates ranges from patching software components in the hosting environment to upgrading networking components or decommissioning hardware.

Updates rarely affect the hosted VMs. When updates do have an effect, Azure chooses the least impactful method for updates:

- If the update doesn't require a reboot, the VM is paused while the host is updated, or the VM is live-migrated to an already updated host.
- If maintenance requires a reboot, you're notified of the planned maintenance. Azure also provides a time window in which you can start the maintenance yourself, at a time that works for you. The self-maintenance window is typically 30 days unless the maintenance is urgent. Azure is investing in technologies to reduce the number of cases in which planned platform maintenance requires the VMs to be rebooted.

This page describes how Azure performs both types of maintenance. For more information about unplanned events (outages), see [Manage the availability of VMs for Windows](#) or the corresponding article for [Linux](#).

Within a VM, you can get notifications about upcoming maintenance by [using Scheduled Events for Windows](#) or for [Linux](#).

For instructions on managing planned maintenance, see [Handling planned maintenance notifications for Linux](#) or the corresponding article for [Windows](#).

Maintenance that doesn't require a reboot

As stated earlier, most platform updates don't affect customer VMs. When a no-impact update isn't possible, Azure chooses the update mechanism that's least impactful to customer VMs.

Most nonzero-impact maintenance pauses the VM for less than 10 seconds. In certain cases, Azure uses memory-preserving maintenance mechanisms. These mechanisms pause the VM for up to 30 seconds and preserve the memory in RAM. The VM is then resumed, and its clock is automatically synchronized.

Memory-preserving maintenance works for more than 90 percent of Azure VMs. It doesn't work for G, M, N, and H series. Azure increasingly uses live-migration technologies and improves memory-preserving maintenance mechanisms to reduce the pause durations.

These maintenance operations that don't require a reboot are applied one fault domain at a time. They stop if they receive any warning health signals.

These types of updates can affect some applications. When the VM is live-migrated to a different host, some sensitive workloads might show a slight performance degradation in the few minutes leading up to the VM pause. To prepare for VM maintenance and reduce impact during Azure maintenance, try [using Scheduled Events for Windows](#) or [Linux](#) for such applications. Azure is working on maintenance-control features for these sensitive applications.

Live migration

Live migration is an operation that doesn't require a reboot and that preserves memory for the VM. It causes a pause or freeze, typically lasting no more than 5 seconds. Except for G, M, N, and H series, all infrastructure as a service (IaaS) VMs, are eligible for live migration. Eligible VMs represent more than 90 percent of the IaaS VMs.

that are deployed to the Azure fleet.

The Azure platform starts live migration in the following scenarios:

- Planned maintenance
- Hardware failure
- Allocation optimizations

Some planned-maintenance scenarios use live migration, and you can use Scheduled Events to know in advance when live migration operations will start.

Live migration can also be used to move VMs when Azure Machine Learning algorithms predict an impending hardware failure or when you want to optimize VM allocations. For more information about predictive modeling that detects instances of degraded hardware, see [Improving Azure VM resiliency with predictive machine learning and live migration](#). Live-migration notifications appear in the Azure portal in the Monitor and Service Health logs as well as in Scheduled Events if you use these services.

Maintenance that requires a reboot

In the rare case where VMs need to be rebooted for planned maintenance, you'll be notified in advance. Planned maintenance has two phases: the self-service phase and a scheduled maintenance phase.

During the *self-service phase*, which typically lasts four weeks, you start the maintenance on your VMs. As part of the self-service, you can query each VM to see its status and the result of your last maintenance request.

When you start self-service maintenance, your VM is redeployed to an already updated node. Because the VM reboots, the temporary disk is lost and dynamic IP addresses associated with the virtual network interface are updated.

If an error arises during self-service maintenance, the operation stops, the VM isn't updated, and you get the option to retry the self-service maintenance.

When the self-service phase ends, the *scheduled maintenance phase* begins. During this phase, you can still query for the maintenance phase, but you can't start the maintenance yourself.

For more information on managing maintenance that requires a reboot, see [Handling planned maintenance notifications for Linux](#) or the corresponding article for [Windows](#).

Availability considerations during scheduled maintenance

If you decide to wait until the scheduled maintenance phase, there are a few things you should consider to maintain the highest availability of your VMs.

Paired regions

Each Azure region is paired with another region within the same geographical vicinity. Together, they make a region pair. During the scheduled maintenance phase, Azure updates only the VMs in a single region of a region pair. For example, while updating the VM in North Central US, Azure doesn't update any VM in South Central US at the same time. However, other regions such as North Europe can be under maintenance at the same time as East US. Understanding how region pairs work can help you better distribute your VMs across regions. For more information, see [Azure region pairs](#).

Availability sets and scale sets

When deploying a workload on Azure VMs, you can create the VMs within an *availability set* to provide high availability to your application. Using availability sets, you can ensure that during either an outage or maintenance events that require a reboot, at least one VM is available.

Within an availability set, individual VMs are spread across up to 20 update domains (UDs). During scheduled maintenance, only one UD is updated at any given time. UDs aren't necessarily updated sequentially.

Virtual machine *scale sets* are an Azure compute resource that you can use to deploy and manage a set of identical VMs as a single resource. The scale set is automatically deployed across UDs, like VMs in an availability set. As with availability sets, when you use scale sets, only one UD is updated at any given time during scheduled maintenance.

For more information about setting up your VMs for high availability, see [Manage the availability of your VMs for Windows](#) or the corresponding article for [Linux](#).

Next steps

For information on planned maintenance impacting virtual machines, see [Handling planned maintenance notifications](#).

Introduction to Azure managed disks

4/22/2019 • 9 minutes to read • [Edit Online](#)

An Azure managed disk is a virtual hard disk (VHD). You can think of it like a physical disk in an on-premises server but, virtualized. Azure managed disks are stored as page blobs, which are a random IO storage object in Azure. We call a managed disk 'managed' because it is an abstraction over page blobs, blob containers, and Azure storage accounts. With managed disks, all you have to do is provision the disk, and Azure takes care of the rest.

When you select to use Azure managed disks with your workloads, Azure creates and manages the disk for you. The available types of disks are ultra disks (Preview), premium solid-state drives (SSD), standard SSD, and standard hard disk drives (HDD). For more 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 supports [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.

Encryption

Managed disks offer two different kinds of encryption. The first is Storage Service Encryption (SSE), which is performed by the storage service. The second one is Azure Disk Encryption, which you can enable on the OS and data disks for your VMs.

Storage Service Encryption (SSE)

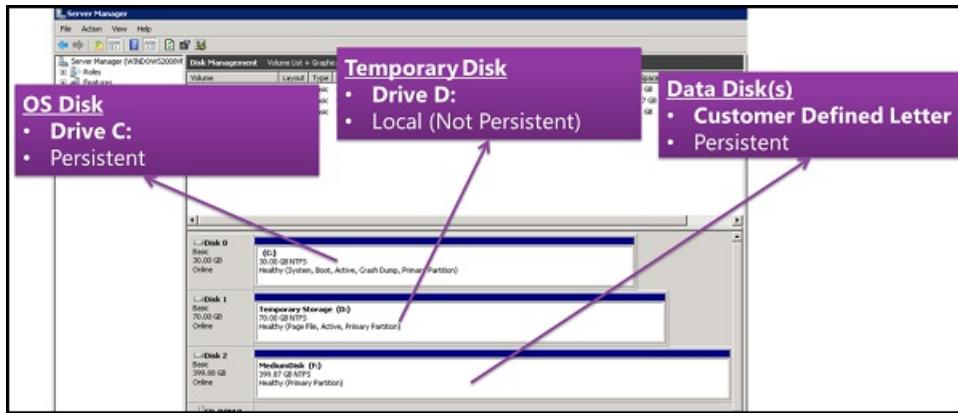
[Azure Storage Service Encryption](#) provides encryption-at-rest and safeguards your data to meet your organizational security and compliance commitments. SSE is enabled by default for all managed disks, snapshots, and images in all the regions where managed disks are available. Visit the [Managed Disks FAQ page](#) for more details.

Azure Disk Encryption (ADE)

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 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. They 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.

To learn more about how to create snapshots with managed disks, see the following resources:

- [Create copy of VHD stored as a managed disk using snapshots in Windows](#)
- [Create copy of VHD stored as a managed disk using snapshots 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 (sysprep) 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

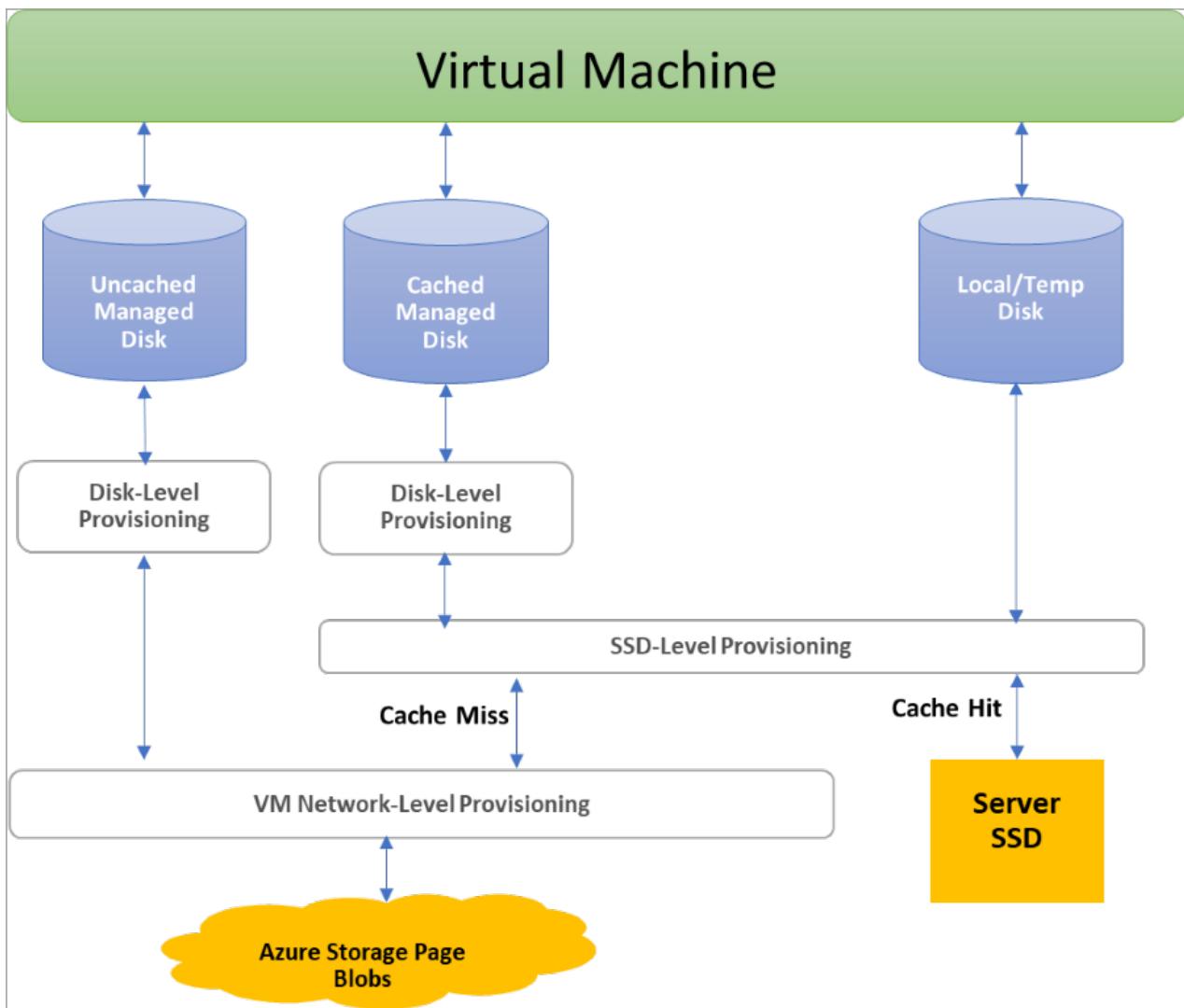
It's important to understand the difference between images and snapshots. With managed disks, you can take an image of a generalized VM that has been deallocated. This image includes all of the disks attached to the VM. You can use this image to create a VM, and it includes all of the disks.

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

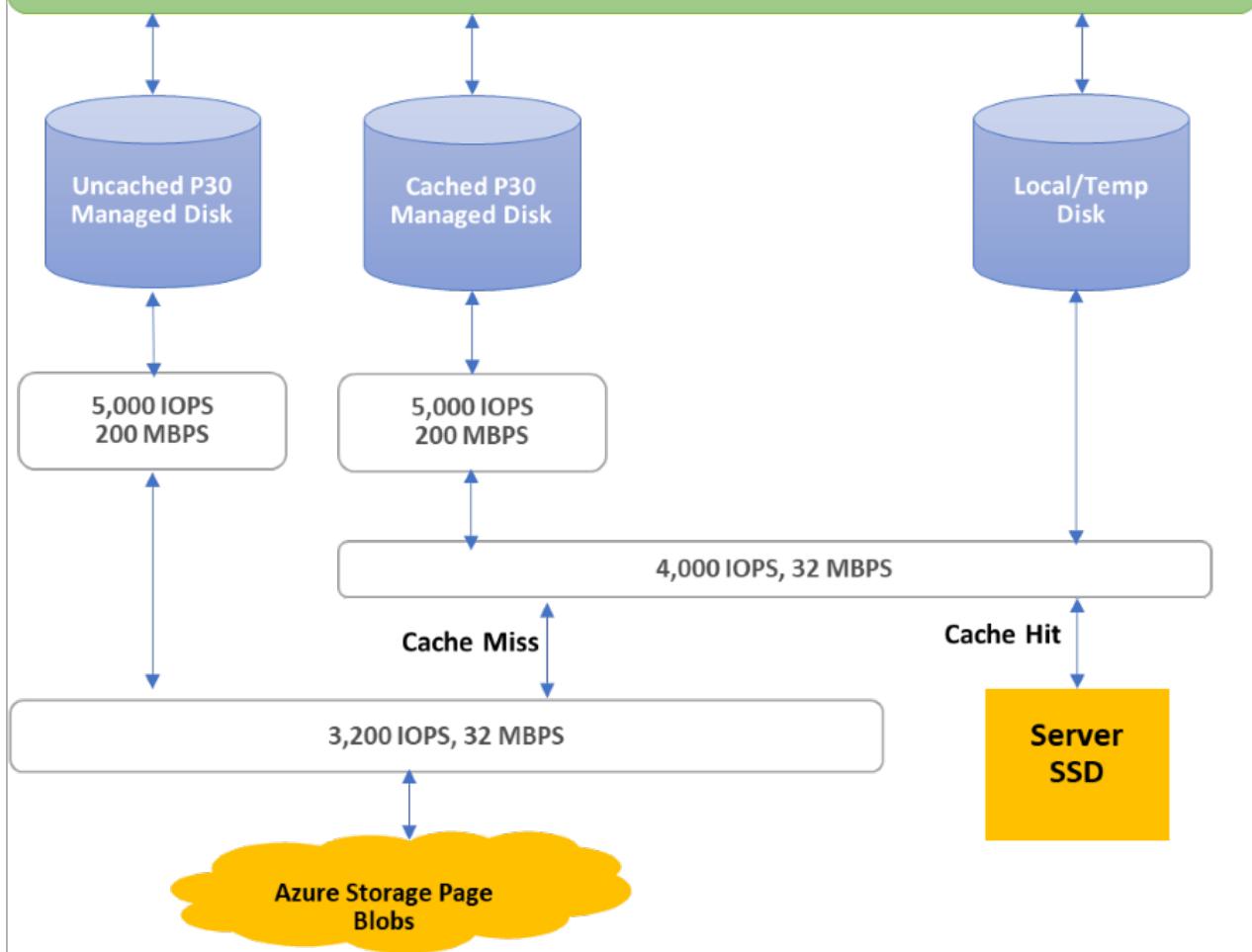
The following diagram depicts real-time allocation of bandwidth and IOPS for disks, using a three-level provisioning system:



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

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?

5/10/2019 • 10 minutes to read • [Edit Online](#)

Azure managed disks currently offers four disk types, three of which are generally available (GA) and one that is currently in preview. These four disk types each have their own appropriate target customer scenarios.

Disk comparison

The following table provides a comparison of ultra solid-state-drives (SSD) (preview), premium SSD, standard SSD, and standard hard disk drives (HDD) for managed disks to help you decide what to use.

	ULTRA SSD (PREVIEW)	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
Disk size	65,536 gibibyte (GiB) (Preview)	32,767 GiB	32,767 GiB	32,767 GiB
Max throughput	2,000 MiB/s (Preview)	900 MiB/s	750 MiB/s	500 MiB/s
Max IOPS	160,000 (Preview)	20,000	6,000	2,000

Ultra SSD (preview)

Azure ultra SSD (preview) delivers high throughput, high IOPS, and consistent low latency disk storage for Azure IaaS VMs. Some additional benefits of ultra SSD include the ability to dynamically change the performance of the disk, along with your workloads, without the need to restart your virtual machines. Ultra SSDs are suited for data-intensive workloads such as SAP HANA, top tier databases, and transaction-heavy workloads. Ultra SSD 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 SSD 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 SSD are:

- Disk capacity: Ultra SSD capacity ranges from 4 GiB up to 64 TiB.
- Disk IOPS: Ultra SSD 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. The minimum disk IOPS are 100 IOPS.
- Disk throughput: With ultra SSD, 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 disk throughput is 1 MiB.

- Ultra SSDs 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.

Disk size

DISK SIZE (GiB)	IOPS CAPS	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

Transactions

For ultra 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.

Preview scope and limitations

During preview, ultra SSD:

- Are supported in East US 2 in a single availability zone
- 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 ES/DS v3 VMs
- Are only available as data disks and only support 4k physical sector size
- Can only be created as empty disks
- Currently can only be deployed using Azure Resource Manager templates, CLI, PowerShell, and the Python SDK.
- Cannot be deployed with the Azure portal (yet).
- Does not yet support disk snapshots, VM images, availability sets, virtual machine scale sets, and Azure disk encryption.
- Does not yet support integration with Azure Backup or Azure Site Recovery.
- As with [most previews](#), this feature should not be used for production workloads until general availability (GA).

If you would like to start using ultra SSDs, see our article on the subject: [Enabling Azure ultra SSDs](#).

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

PREMIUM SSD SIZES	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
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 120	Up to 240	Up to 500	Up to 1,100	Up to 2,300	Up to 5,000	Up to 7,500	Up to 7,500	Up to 16,000	Up to 18,000	Up to 20,000
Throughput per disk	Up to 25 MiB/sec	Up to 50 MiB/sec	Up to 100 MiB/sec	Up to 125 MiB/sec	Up to 150 MiB/sec	Up to 200 MiB/sec	Up to 250 MiB/sec	Up to 250 MiB/sec	Up to 500 MiB/sec	Up to 750 MiB/sec	Up to 900 MiB/sec

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.

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

STANDARD SSD SIZES	E4	E6	E10	E15	E20	E30	E40	E50	E60	E70	E80
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 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/s ec	Up to 50 MiB/s ec	Up to 60 MiB/s ec	Up to 400 MiB/s ec	Up to 600 MiB/s ec	Up to 750 MiB/s ec					

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 SSD VM reservation fee

Azure VMs have the capability to indicate if they are compatible with ultra SSD. 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.

Refer to the [Azure Disks pricing page](#) for ultra disk Disks pricing details.

Azure premium storage: design for high performance

4/23/2019 • 36 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 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 [Azure Storage Scalability and Performance Targets](#) articles.

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			
Min. Latency			

PERFORMANCE REQUIREMENTS	50 PERCENTILE	90 PERCENTILE	99 PERCENTILE
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
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

COUNTER	DESCRIPTION	PERFMON	IOSTAT
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 section at the end of this article 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](#).

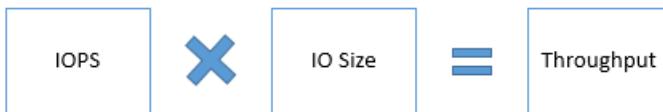
	IOPS	THROUGHPUT	LATENCY
Example Scenario	Enterprise OLTP application requiring very high transactions per second rate.	Enterprise Data warehousing application processing large amounts of data.	Near real-time applications requiring instant responses to user requests, like online gaming.
Performance factors			
IO size	Smaller IO size yields higher IOPS.	Larger IO size to yields higher Throughput.	

	IOPS	THROUGHPUT	LATENCY
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 premium 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.

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. Table below 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 section at the end of this article 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, DSv2, 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, DSv2 series and a GS series:

VM SIZE	CPU CORES	MEMORY	VM DISK SIZES	MAX. DATA DISKS	CACHE SIZE	IOPS	BANDWIDTH CACHE IO LIMITS
Standard_DS14	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_GS5	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 eight GA disk sizes and three disk sizes that are in preview, currently. 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 11 disks sizes and their capabilities. P4, P6, P15, P60, P70, and P80 sizes are currently only supported for Managed Disks.

PREMIUM DISKS TYPE	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
Disk size	32 GiB	64 GiB	128 GiB	256 GiB	512 GB	1,024 GiB (1 TiB)	2,048 GiB (2 TiB)	4,095 GiB (4 TiB)	8,192 GiB (8 TiB)	16,384 GiB (16 TiB)	32,767 GiB (32 TiB)
IOPS per disk	120	240	500	1100	2300	5000	7500	7500	12,500	15,000	20,000

PREMIUM DISKS TYPE	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
Throughput per disk	25 MiB per second	50 MiB per second	100 MiB per second	125 MiB per second	150 MiB per second	200 MiB per second	250 MiB per second	250 MiB per second	480 MiB per second	750 MiB per second	750 MiB per second

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 larger than 4 TiB. If multiple disks are attached to your VM, each disk that is 4 TiB or smaller 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.

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 later.

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	3.2.0-75.110+	Ubuntu-12_04_5-LTS-amd64-server-20150119-en-us-30GB
Ubuntu	14.04	3.13.0-44.73+	Ubuntu-14_04_1-LTS-amd64-server-20150123-en-us-30GB
Debian	7.x, 8.x	3.16.7-ckt4-1+	
SUSE	SLES 12	3.12.36-38.1+	suse-sles-12-priority-v20150213 suse-sles-12-v20150213
SUSE	SLES 11 SP4	3.0.101-0.63.1+	
CoreOS	584.0.0+	3.18.4+	CoreOS 584.0.0
CentOS	6.5, 6.6, 6.7, 7.0		LIS4 required <i>See note in the next section</i>
CentOS	7.1+	3.10.0-229.1.2.el7+	LIS4 recommended <i>See note in the next section</i>

DISTRIBUTION	VERSION	SUPPORTED KERNEL	DETAILS
Red Hat Enterprise Linux (RHEL)	6.8+, 7.2+		
Oracle	6.0+, 7.2+		UEK4 or RHCK
Oracle	7.0-7.1		UEK4 or RHCK w/ LIS 4.1+
Oracle	6.4-6.7		UEK4 or RHCK w/ LIS 4.1+

LIS drivers for OpenLogic CentOS

If you're running OpenLogic CentOS VMs, run the following command to install the latest drivers:

```
sudo rpm -e hypervkvpd ## (Might return an error if not installed. That's OK.)
sudo yum install microsoft-hyper-v
```

To activate the new drivers, restart the VM.

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.

$$\text{IOPS} \times \text{Latency} = \text{Queue Depth}$$

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.

$$\text{QD per Disk} \times \text{No. of Columns per Volume} = \text{QD of Striped Volume}$$

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

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](#)

Scalability and performance targets for VM disks on Linux

2/15/2019 • 4 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

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Standard managed disks	25,000	50,000
Standard SSD managed disks	25,000	50,000
Premium managed disks	25,000	50,000
Standard_LRS snapshots	25,000	50,000
Standard_ZRS snapshots	25,000	50,000
Managed image	25,000	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 [Linux 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 SIZES	E4	E6	E10	E15	E20	E30	E40	E50	E60	E70	E80
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 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					

Premium SSD managed disks: Per-disk limits

PREMIUM SSD SIZES	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70	P80
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 120	Up to 240	Up to 500	Up to 1,100	Up to 2,300	Up to 5,000	Up to 7,500	Up to 7,500	Up to 16,000	Up to 18,000	Up to 20,000
Throughput per disk	Up to 25 MiB/sec	Up to 50 MiB/sec	Up to 100 MiB/sec	Up to 125 MiB/sec	Up to 150 MiB/sec	Up to 200 MiB/sec	Up to 250 MiB/sec	Up to 250 MiB/sec	Up to 500 MiB/sec	Up to 750 MiB/sec	Up to 900 MiB/sec

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

4/23/2019 • 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.

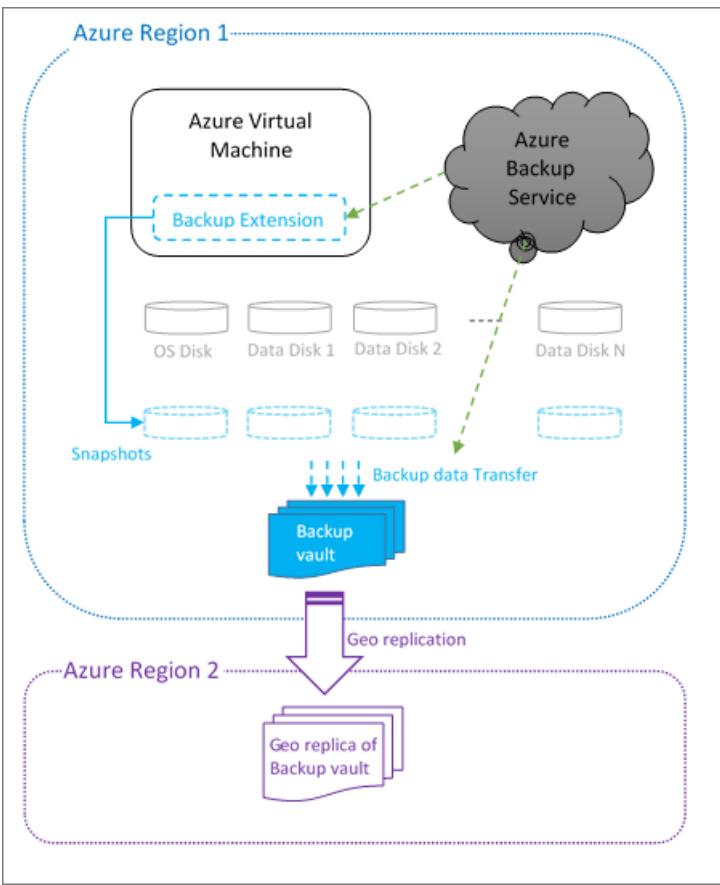
Your choices for high availability, backup, and DR at application or infrastructure levels can be represented as follows:

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, only file-consistent backups are possible, because Linux does not have functionality equivalent to the volume shadow service.



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](#).

5. Make sure that the VM allows network connectivity for the backup service to function. Follow the instructions for [network connectivity](#).
6. 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 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:

1. Freeze all the disks.
2. Flush all the pending writes.
3. [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.

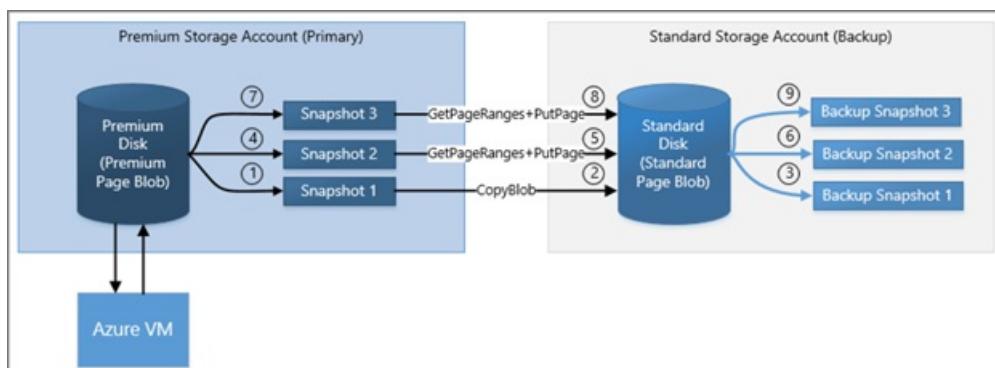
If you use geo-redundant storage or read-access geo-redundant storage for your disks, then the snapshots are replicated to the secondary region automatically. There can be a few minutes of lag before the replication. If the primary datacenter goes down before the snapshots finish replicating, you cannot 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.

Preview: Ephemeral OS disks for Azure VMs

5/6/2019 • 4 minutes to read • [Edit Online](#)

Ephemeral OS disks are created on the local Virtual Machine (VM) storage and not persisted to the remote Azure Storage. Ephemeral OS disks work well for stateless workloads, where applications are tolerant of individual VM failures, but are more concerned about the time it takes for large-scale deployments or time to reimagine the individual VM instances. It is also suitable for applications, deployed using the classic deployment model, to move to the Resource Manager deployment model. With Ephemeral OS disk, you would observe lower read/write latency to the OS disk and faster VM reimaging. In addition, Ephemeral OS disk is free, you incur no storage cost for OS disk.

The key features of ephemeral disks are:

- They can be used with both Marketplace images and custom images.
- You can deploy VM Images up to the size of the VM Cache.
- Ability to fast reset or reimagine their VMs to the original boot state.
- Lower run-time latency similar to a temporary disk.
- No cost for the OS disk.

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 - 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

Scale set 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 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

Currently, the only method to reimagine a Virtual Machine instance with ephemeral OS disk is through using REST API. For scale sets, reimaging is already available through Powershell, CLI, and the portal.

```
POST https://management.azure.com/subscriptions/{sub-
id}/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: For preview, we will support platform and/or 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

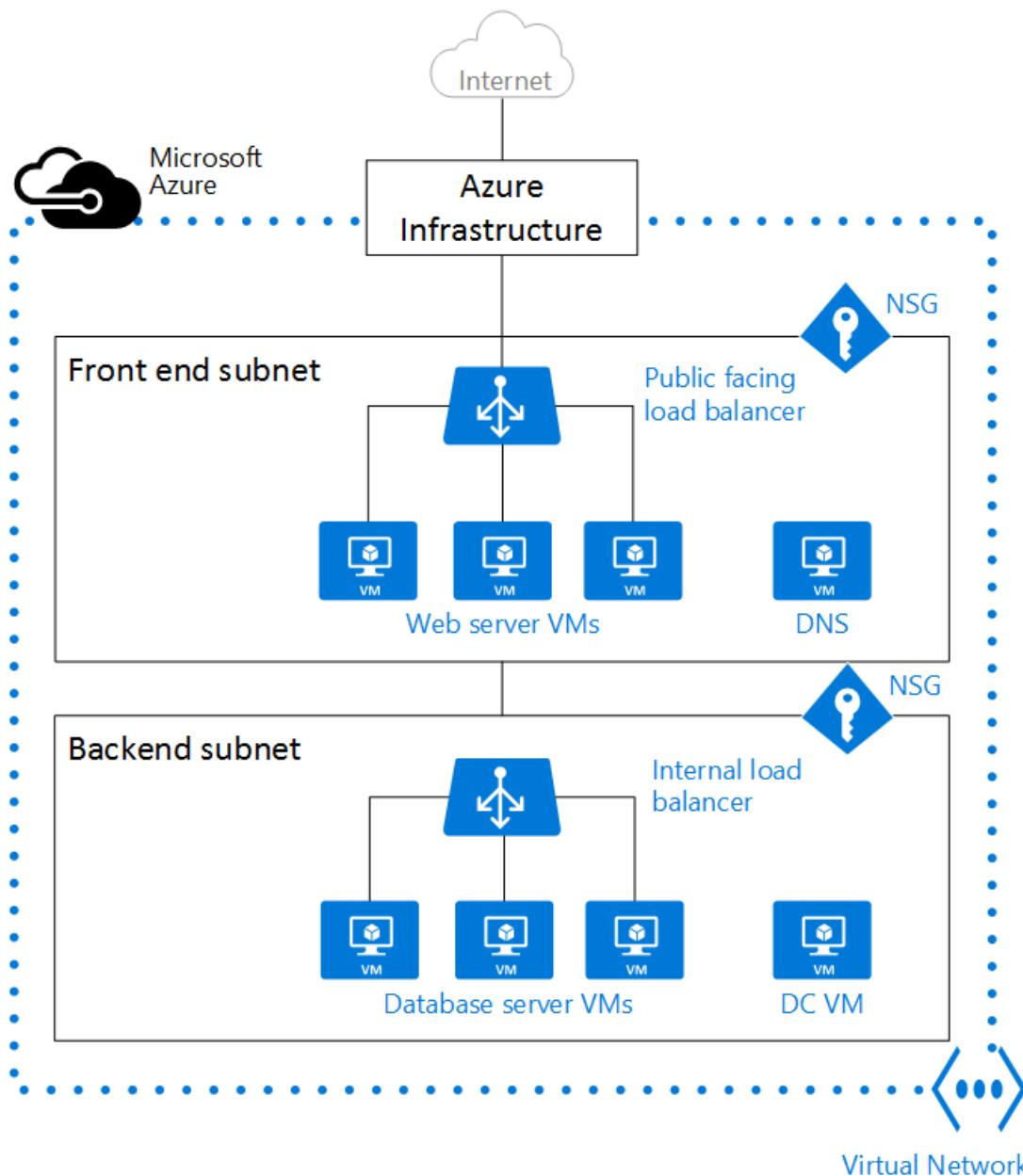
You can create a VM with an ephemeral OS disk using the [Azure CLI](#).

Virtual networks and virtual machines in Azure

1/23/2019 • 14 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

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](#).

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 -PublicIpAddress 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 <code>New-AzPublicIpAddress</code> with the -AllocationMethod parameter as Dynamic or Static.
Azure CLI	You use <code>az network public-ip create</code> 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
--------	-------------

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.
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 .

METHOD	DESCRIPTION
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.
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 Add-AzVMNetworkInterface 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](#).

What are virtual machine scale sets?

5/28/2019 • 3 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.

Next steps

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

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

Use infrastructure automation tools with virtual machines in Azure

1/24/2019 • 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

3/12/2019 • 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.

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](#).

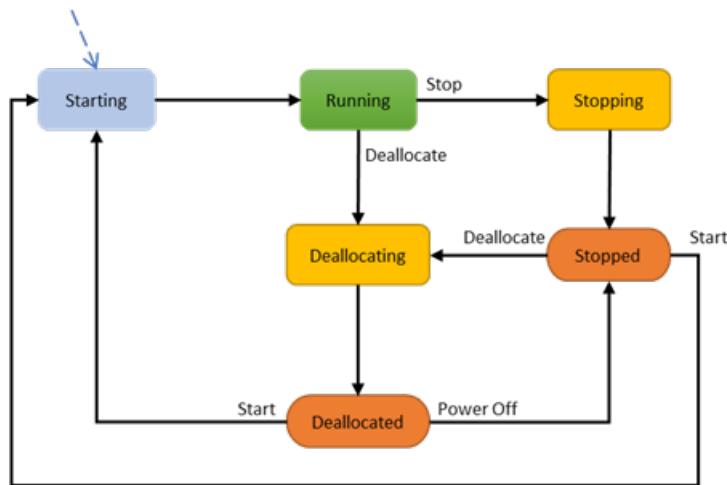
Virtual machines lifecycle and states

8/10/2018 • 3 minutes to read • [Edit Online](#)

Azure Virtual Machines (VMs) go through different states that can be categorized into *provisioning* and *power* states. The purpose of this article is to describe these states and specifically highlight when customers are billed for instance usage.

Power states

The power state represents the last known state of the VM.



The following table provides a description of each instance state and indicates whether it is billed for instance usage or not.

STATE	DESCRIPTION	INSTANCE USAGE BILLING
Starting	VM is starting up. <pre>"statuses": [{ "code": "PowerState/starting", "level": "Info", "displayStatus": "VM starting" }]</pre>	Not billed
Running	Normal working state for a VM <pre>"statuses": [{ "code": "PowerState/running", "level": "Info", "displayStatus": "VM running" }]</pre>	Billed

Stopping	This is a transitional state. When completed, it will show as **Stopped**. <pre>"statuses": [{ "code": "PowerState/stopping", "level": "Info", "displayStatus": "VM stopping" }]</pre>	Billed
Stopped	The VM has been shut down from within the guest OS or using the PowerOff APIs. Hardware is still allocated to the VM and it remains on the host. <pre>"statuses": [{ "code": "PowerState/stopped", "level": "Info", "displayStatus": "VM stopped" }]</pre>	Billed*
Deallocating	Transitional state. When completed, the VM will show as **Deallocated**. <pre>"statuses": [{ "code": "PowerState/deallocating", "level": "Info", "displayStatus": "VM deallocating" }]</pre>	Not billed*
Deallocated	The VM has been stopped successfully and removed from the host. <pre>"statuses": [{ "code": "PowerState/deallocated", "level": "Info", "displayStatus": "VM deallocated" }]</pre>	Not billed

*Some Azure resources, such as Disks and Networking, incur charges. Software licenses on the instance do not incur charges.

Provisioning states

A provisioning state is the status of a user-initiated, control-plane operation on the VM. These states are separate from the power state of a VM.

- **Create** – VM creation.
- **Update** – updates the model for an existing VM. Some non-model changes to VM such as Start/Restart also fall under update.
- **Delete** – VM deletion.

- **Deallocate** – is where a VM is stopped and removed from the host. Deallocating a VM is considered an update, so it will display provisioning states related to updating.

Here are the transitional operation states after the platform has accepted a user-initiated action:

States	Description
Creating	<pre>"statuses": [{ "code": "ProvisioningState/creating", "level": "Info", "displayStatus": "Creating" }]</pre>
Updating	<pre>"statuses": [{ "code": "ProvisioningState/updating", "level": "Info", "displayStatus": "Updating" }]</pre>
Deleting	<pre>"statuses": [{ "code": "ProvisioningState/deleting", "level": "Info", "displayStatus": "Deleting" }]</pre>
OS provisioning states	<p>If a VM is created with an OS image and not with a specialized image, then following substates can be observed:</p> <ol style="list-style-type: none"> 1. OSProvisioningInProgress – The VM is running, and installation of guest OS is in progress. <pre>"statuses": [{ "code": "ProvisioningState/creating/OSProvisioningInProgress", "level": "Info", "displayStatus": "OS Provisioning In progress" }]</pre> <ol style="list-style-type: none"> 2. OSProvisioningComplete – Short-lived state. The VM quickly transitions to **Success** unless any extensions need to be installed. Installing extensions can take time. <pre>"statuses": [{ "code": "ProvisioningState/creating/OSProvisioningComplete", "level": "Info", "displayStatus": "OS Provisioning Complete" }]</pre> <p>Note: OS Provisioning can transition to **Failed** if there is an OS failure or the OS doesn't install in time. Customers will be billed for the deployed VM on the infrastructure.</p>

Once the operation is complete, the VM will transition into one of the following states:

- **Succeeded** – the user-initiated actions have completed.

```
"statuses": [
{
  "code": "ProvisioningState/succeeded",
  "level": "Info",
  "displayStatus": "Provisioning succeeded",
  "time": "time"
}
]
```

- **Failed** – represents a failed operation. Refer to the error codes to get more information and possible solutions.

```
"statuses": [
{
  "code": "ProvisioningState/failed/InternalOperationError",
  "level": "Error",
  "displayStatus": "Provisioning failed",
  "message": "Operation abandoned due to internal error. Please try again later.",
  "time": "time"
}
]
```

VM instance view

The instance view API provides VM running-state information. For more information, see the [Virtual Machines - Instance View](#) API documentation.

Azure Resources explorer provides a simple UI for viewing the VM running state: [Resource Explorer](#).

Provisioning states are visible on VM properties and instance view. Power states are available in instance view of VM.

Next steps

To learn more about monitoring your VM, see [How to monitor virtual machines in Azure](#).

How to monitor virtual machines in Azure

3/12/2019 • 5 minutes to read • [Edit Online](#)

You can take advantage of many opportunities to monitor your VMs by collecting, viewing, and analyzing diagnostic and log data. To do simple [monitoring](#) of your VM, you can use the Overview screen for the VM in the Azure portal. You can use [extensions](#) to configure diagnostics on your VMs to collect additional metric data. You can also use more advanced monitoring options, such as [Application Insights](#) and [Log Analytics](#).

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 logs to store the data and set up alerts to let you know when performance isn't quite 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 PowerShell](#), or the [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 PowerBI.
- Analyze it in PowerBI using the [PowerBI 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 Diagnostic Logs](#) are logs emitted by your VM that provide rich, frequent data about its operation. Diagnostic 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 PowerBI.
- Analyze them with [Log Analytics](#).

Advanced monitoring

- [Azure Monitor](#) is a service that monitors your cloud and on-premises environments to maintain their availability and performance. It delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources they depend on. You can install an extension on a [Linux VM](#) or a [Windows VM](#) that installs the Log Analytics agent to collect log data and store in a Log Analytics workspace.

For Windows and Linux VMs, the recommended method for collecting logs is by installing the Log Analytics agent. The easiest way to install the Log Analytics agent on a VM is through the [Log Analytics VM Extension](#). Using the extension simplifies the installation process and automatically configures the agent to send data to the Log Analytics workspace that you specify. The agent is also upgraded automatically, ensuring that you have the latest features and fixes.

- [Network Watcher](#) enables you to monitor your VM and its associated resources as they relate to the network that they are in. You can install the Network Watcher Agent extension on a [Linux VM](#) or a [Windows VM](#).
- [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 their different processes and interconnected dependencies on other resources and external processes.

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](#).

Backup and restore options for Linux virtual machines in Azure

4/9/2018 • 2 minutes to read • [Edit Online](#)

You can protect your data by taking backups at regular intervals. There are several backup options available for VMs, depending on your use-case.

Azure Backup

For backing up Azure VMs running production workloads, use Azure Backup. Azure Backup supports application-consistent backups for both Windows and Linux VMs. Azure Backup creates recovery points that are stored in geo-redundant recovery vaults. When you restore from a recovery point, you can restore the whole VM or just specific files.

For a simple, hands-on introduction to Azure Backup for Azure VMs, see the "Back up Azure virtual machines" tutorial for [Linux](#) or [Windows](#).

For more information on how Azure Backup works, see [Plan your VM backup infrastructure in Azure](#)

Azure Site Recovery

Azure Site Recovery protects your VMs from a major disaster scenario, when a whole region experiences an outage due to major natural disaster or widespread service interruption. You can configure Azure Site Recovery for your VMs so that you can recover your application with a single click in matter of minutes. You can replicate to an Azure region of your choice, it is not restricted to paired regions.

You can run disaster-recovery drills with on-demand test failovers, without affecting your production workloads or ongoing replication. Create recovery plans to orchestrate failover and failback of the entire application running on multiple VMs. The recovery plan feature is integrated with Azure automation runbooks.

You can get started by [replicating your virtual machines](#).

Managed snapshots

In development and test environments, snapshots provide a quick and simple option for backing up VMs that use Managed Disks. A managed snapshot is a read-only full copy of a managed disk. Snapshots exist independent of the source disk and can be used to create new managed disks for rebuilding a VM. They are billed based on the used portion of the disk. For example, if you create a snapshot of a managed disk with provisioned capacity of 64 GB and actual used data size of 10 GB, snapshot will be billed only for the used data size of 10 GB.

For more information on creating snapshots, see:

- [Create copy of VHD stored as a Managed Disk using Snapshots in Windows](#)
- [Create copy of VHD stored as a Managed Disk using Snapshots in Linux](#)

Next steps

You can try out Azure Backup by following the "Back up Windows virtual machines tutorial" for [Linux](#) or [Windows](#).

Example Azure infrastructure walkthrough for Linux VMs

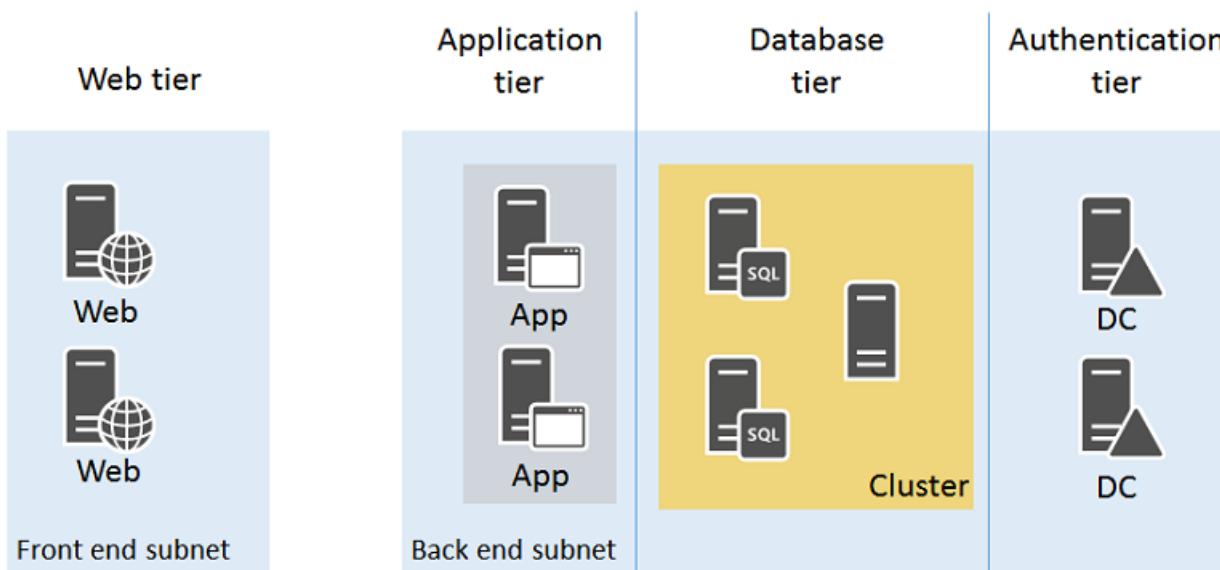
3/14/2019 • 3 minutes to read • [Edit Online](#)

This article walks through building out an example application infrastructure. We detail designing an infrastructure for a simple on-line store that brings together all the guidelines and decisions around naming conventions, availability sets, virtual networks and load balancers, and actually deploying your virtual machines (VMs).

Example workload

Adventure Works Cycles wants to build an on-line store application in Azure that consists of:

- Two nginx servers running the client front-end in a web tier
- Two nginx servers processing data and orders in an application tier
- Two MongoDB servers part of a sharded cluster for storing product data and orders in a database tier
- Two Active Directory domain controllers for customer accounts and suppliers in an authentication tier
- All the servers are located in two subnets:
 - a front-end subnet for the web servers
 - a back-end subnet for the application servers, MongoDB cluster, and domain controllers



Incoming secure web traffic must be load-balanced among the web servers as customers browse the on-line store. Order processing traffic in the form of HTTP requests from the web servers must be load-balanced among the application servers. Additionally, the infrastructure must be designed for high availability.

The resulting design must incorporate:

- An Azure subscription and account
- A single resource group
- Azure Managed Disks
- A virtual network with two subnets
- Availability sets for the VMs with a similar role
- Virtual machines

All the above follow these naming conventions:

- Adventure Works Cycles uses **[IT workload]-[location]-[Azure resource]** as a prefix
 - For this example, "azos" (Azure On-line Store) is the IT workload name and "use" (East US 2) is the location
- Virtual networks use AZOS-USE-VN**[number]**
- Availability sets use azos-use-as-**[role]**
- Virtual machine names use azos-use-vm-**[vmname]**

Azure subscriptions and accounts

Adventure Works Cycles is using their Enterprise subscription, named Adventure Works Enterprise Subscription, to provide billing for this IT workload.

Storage

Adventure Works Cycles determined that they should use Azure Managed Disks. When creating VMs, both storage available storage tiers are used:

- **Standard storage** for the web servers, application servers, and domain controllers and their data disks.
- **Premium storage** for the MongoDB sharded cluster servers and their data disks.

Virtual network and subnets

Because the virtual network does not need ongoing connectivity to the Adventure Work Cycles on-premises network, they decided on a cloud-only virtual network.

They created a cloud-only virtual network with the following settings using the Azure portal:

- Name: AZOS-USE-VN01
- Location: East US 2
- Virtual network address space: 10.0.0.0/8
- First subnet:
 - Name: FrontEnd
 - Address space: 10.0.1.0/24
- Second subnet:
 - Name: BackEnd
 - Address space: 10.0.2.0/24

Availability sets

To maintain high availability of all four tiers of their on-line store, Adventure Works Cycles decided on four availability sets:

- **azos-use-as-web** for the web servers
- **azos-use-as-app** for the application servers
- **azos-use-as-db** for the servers in the MongoDB sharded cluster
- **azos-use-as-dc** for the domain controllers

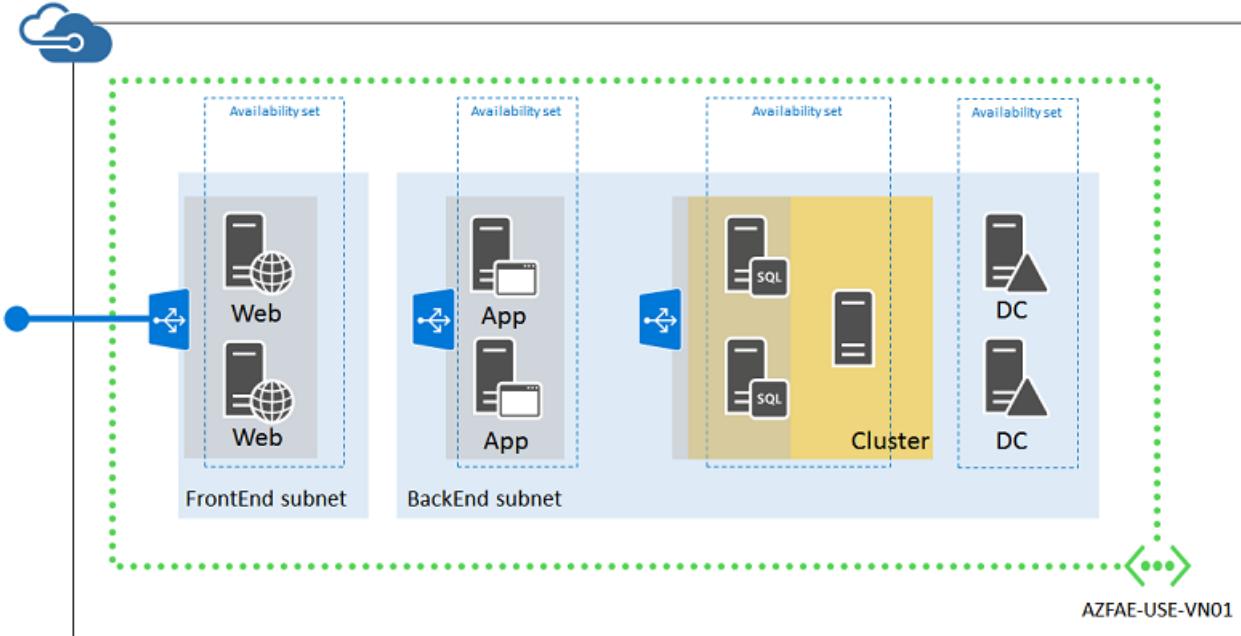
Virtual machines

Adventure Works Cycles decided on the following names for their Azure VMs:

- **azos-use-vm-web01** for the first web server

- **azos-use-vm-web02** for the second web server
- **azos-use-vm-app01** for the first application server
- **azos-use-vm-app02** for the second application server
- **azos-use-vm-db01** for the first MongoDB server in the cluster
- **azos-use-vm-db02** for the second MongoDB server in the cluster
- **azos-use-vm-dc01** for the first domain controller
- **azos-use-vm-dc02** for the second domain controller

Here is the resulting configuration.



This configuration incorporates:

- A cloud-only virtual network with two subnets (FrontEnd and BackEnd)
- Azure Managed Disks using both Standard and Premium disks
- Four availability sets, one for each tier of the on-line store
- The virtual machines for the four tiers
- An external load balanced set for HTTPS-based web traffic from the Internet to the web servers
- An internal load balanced set for unencrypted web traffic from the web servers to the application servers
- A single resource group

Virtual machine vCPU quotas

2/6/2019 • 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 the Azure CLI.

Check usage

You can check your quota usage using `az vm list-usage`.

```
az vm list-usage --location "East US" -o table
```

The output should look something like this:

Name	CurrentValue	Limit
Availability Sets	0	2000
Total Regional vCPUs	29	100
Virtual Machines	7	10000
Virtual Machine Scale Sets	0	2000
Standard DSv3 Family vCPUs	8	100
Standard DSv2 Family vCPUs	3	100
Standard Dv3 Family vCPUs	2	100
Standard D Family vCPUs	8	100
Standard Dv2 Family vCPUs	8	100
Basic A Family vCPUs	0	100
Standard A0-A7 Family vCPUs	0	100
Standard A8-A11 Family vCPUs	0	100
Standard DS Family vCPUs	0	100
Standard G Family vCPUs	0	100
Standard GS Family vCPUs	0	100
Standard F Family vCPUs	0	100
Standard FS Family vCPUs	0	100
Standard Storage Managed Disks	5	10000
Premium Storage Managed Disks	5	10000

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 Azure reservations are deployable in the subscription. For example, if a specific subscription has 10 Standard_D1 reservations the usages limit for Standard_D1 reservations 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 either 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](#).

Create a complete Linux virtual machine with the Azure CLI

3/14/2019 • 10 minutes to read • [Edit Online](#)

To quickly create a virtual machine (VM) in Azure, you can use a single Azure CLI command that uses default values to create any required supporting resources. Resources such as a virtual network, public IP address, and network security group rules are automatically created. For more control of your environment in production use, you may create these resources ahead of time and then add your VMs to them. This article guides you through how to create a VM and each of the supporting resources one by one.

Make sure that you have installed the latest [Azure CLI](#) and logged to an Azure account in with `az login`.

In the following examples, replace example parameter names with your own values. Example parameter names include `myResourceGroup`, `myVnet`, and `myVM`.

Create 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 and supporting virtual network resources. Create the 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
```

By default, the output of Azure CLI commands is in JSON (JavaScript Object Notation). To change the default output to a list or table, for example, use [az configure --output](#). You can also add `--output` to any command for a one time change in output format. The following example shows the JSON output from the `az group create` command:

```
{
  "id": "/subscriptions/guid/resourceGroups/myResourceGroup",
  "location": "eastus",
  "name": "myResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

Create a virtual network and subnet

Next you create a virtual network in Azure and a subnet in to which you can create your VMs. Use [az network vnet create](#) to create a virtual network named `myVnet` with the `192.168.0.0/16` address prefix. You also add a subnet named `mySubnet` with the address prefix of `192.168.1.0/24`:

```
az network vnet create \
    --resource-group myResourceGroup \
    --name myVnet \
    --address-prefix 192.168.0.0/16 \
    --subnet-name mySubnet \
    --subnet-prefix 192.168.1.0/24
```

The output shows the subnet is logically created inside the virtual network:

```
{
  "addressSpace": {
    "addressPrefixes": [
      "192.168.0.0/16"
    ]
  },
  "dhcpOptions": {
    "dnsServers": []
  },
  "etag": "W/\"e95496fc-f417-426e-a4d8-c9e4d27fc2ee\"",
  "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/myVnet",
  "location": "eastus",
  "name": "myVnet",
  "provisioningState": "Succeeded",
  "resourceGroup": "myResourceGroup",
  "resourceGuid": "ed62fd03-e9de-430b-84df-8a3b87cacdbb",
  "subnets": [
    {
      "addressPrefix": "192.168.1.0/24",
      "etag": "W/\"e95496fc-f417-426e-a4d8-c9e4d27fc2ee\"",
      "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/myVnet/subnets/mySubnet",
      "ipConfigurations": null,
      "name": "mySubnet",
      "networkSecurityGroup": null,
      "provisioningState": "Succeeded",
      "resourceGroup": "myResourceGroup",
      "resourceNavigationLinks": null,
      "routeTable": null
    }
  ],
  "tags": {},
  "type": "Microsoft.Network/virtualNetworks",
  "virtualNetworkPeerings": null
}
```

Create a public IP address

Now let's create a public IP address with [az network public-ip create](#). This public IP address enables you to connect to your VMs from the Internet. Because the default address is dynamic, create a named DNS entry with the `--domain-name-label` parameter. The following example creates a public IP named *myPublicIP* with the DNS name of *mypublicdns*. Because the DNS name must be unique, provide your own unique DNS name:

```
az network public-ip create \
    --resource-group myResourceGroup \
    --name myPublicIP \
    --dns-name mypublicdns
```

Output:

```
{  
  "publicIp": {  
    "dnsSettings": {  
      "domainNameLabel": "mypublicdns",  
      "fqdn": "mypublicdns.eastus.cloudapp.azure.com",  
      "reverseFqdn": null  
    },  
    "etag": "W/\"2632aa72-3d2d-4529-b38e-b622b4202925\"",  
    "id":  
      "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP",  
      "idleTimeoutInMinutes": 4,  
      "ipAddress": null,  
      "ipConfiguration": null,  
      "location": "eastus",  
      "name": "myPublicIP",  
      "provisioningState": "Succeeded",  
      "publicIpAddressVersion": "IPv4",  
      "publicIpAllocationMethod": "Dynamic",  
      "resourceGroup": "myResourceGroup",  
      "resourceGuid": "4c65de38-71f5-4684-be10-75e605b3e41f",  
      "tags": null,  
      "type": "Microsoft.Network/publicIPAddresses"  
    },  
  },  
}
```

Create a network security group

To control the flow of traffic in and out of your VMs, you apply a network security group to a virtual NIC or subnet.

The following example uses [az network nsg create](#) to create a network security group named *myNetworkSecurityGroup*:

```
az network nsg create \  
  --resource-group myResourceGroup \  
  --name myNetworkSecurityGroup
```

You define rules that allow or deny specific traffic. To allow inbound connections on port 22 (to enable SSH access), create an inbound rule with [az network nsg rule create](#). The following example creates a rule named *myNetworkSecurityGroupRuleSSH*:

```
az network nsg rule create \  
  --resource-group myResourceGroup \  
  --nsg-name myNetworkSecurityGroup \  
  --name myNetworkSecurityGroupRuleSSH \  
  --protocol tcp \  
  --priority 1000 \  
  --destination-port-range 22 \  
  --access allow
```

To allow inbound connections on port 80 (for web traffic), add another network security group rule. The following example creates a rule named *myNetworkSecurityGroupRuleHTTP*:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNetworkSecurityGroup \
--name myNetworkSecurityGroupRuleWeb \
--protocol tcp \
--priority 1001 \
--destination-port-range 80 \
--access allow
```

Examine the network security group and rules with [az network nsg show](#):

```
az network nsg show --resource-group myResourceGroup --name myNetworkSecurityGroup
```

Output:

```
{
  "defaultSecurityRules": [
    {
      "access": "Allow",
      "description": "Allow inbound traffic from all VMs in VNET",
      "destinationAddressPrefix": "VirtualNetwork",
      "destinationPortRange": "*",
      "direction": "Inbound",
      "etag": "W/\"3371b313-ea9f-4687-a336-a8ebdfd80523\"",
      "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup/defaultSecurityRules/AllowVnetInBound",
      "name": "AllowVnetInBound",
      "priority": 65000,
      "protocol": "*",
      "provisioningState": "Succeeded",
      "resourceGroup": "myResourceGroup",
      "sourceAddressPrefix": "VirtualNetwork",
      "sourcePortRange": "*"
    },
    {
      "access": "Allow",
      "description": "Allow inbound traffic from azure load balancer",
      "destinationAddressPrefix": "*",
      "destinationPortRange": "*",
      "direction": "Inbound",
      "etag": "W/\"3371b313-ea9f-4687-a336-a8ebdfd80523\"",
      "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup/defaultSecurityRules/AllowAzureLoadBalancerInBou",
      "name": "AllowAzureLoadBalancerInBound",
      "priority": 65001,
      "protocol": "*",
      "provisioningState": "Succeeded",
      "resourceGroup": "myResourceGroup",
      "sourceAddressPrefix": "AzureLoadBalancer",
      "sourcePortRange": "*"
    },
    {
      "access": "Deny",
      "description": "Deny all inbound traffic",
      "destinationAddressPrefix": "*",
      "destinationPortRange": "*",
      "direction": "Inbound",
      "etag": "W/\"3371b313-ea9f-4687-a336-a8ebdfd80523\"",
      "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup/defaultSecurityRules/DenyAllInBound",
      "name": "DenyAllInBound",
```

```
"priority": 65500,
"protocol": "*",
"provisioningState": "Succeeded",
"resourceGroup": "myResourceGroup",
"sourceAddressPrefix": "*",
"sourcePortRange": "*"
},
{
"access": "Allow",
"description": "Allow outbound traffic from all VMs to all VMs in VNET",
"destinationAddressPrefix": "VirtualNetwork",
"destinationPortRange": "*",
"direction": "Outbound",
"etag": "W/\"3371b313-ea9f-4687-a336-a8ebdfd80523\",
"id":
"/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetwor
kSecurityGroup/defaultSecurityRules/AllowVnetOutBound",
"name": "AllowVnetOutBound",
"priority": 65000,
"protocol": "*",
"provisioningState": "Succeeded",
"resourceGroup": "myResourceGroup",
"sourceAddressPrefix": "VirtualNetwork",
"sourcePortRange": "*"
},
{
"access": "Allow",
"description": "Allow outbound traffic from all VMs to Internet",
"destinationAddressPrefix": "Internet",
"destinationPortRange": "*",
"direction": "Outbound",
"etag": "W/\"3371b313-ea9f-4687-a336-a8ebdfd80523\",
"id":
"/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetwor
kSecurityGroup/defaultSecurityRules/AllowInternetOutBound",
"name": "AllowInternetOutBound",
"priority": 65001,
"protocol": "*",
"provisioningState": "Succeeded",
"resourceGroup": "myResourceGroup",
"sourceAddressPrefix": "*",
"sourcePortRange": "*"
},
{
"access": "Deny",
"description": "Deny all outbound traffic",
"destinationAddressPrefix": "*",
"destinationPortRange": "*",
"direction": "Outbound",
"etag": "W/\"3371b313-ea9f-4687-a336-a8ebdfd80523\",
"id":
"/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetwor
kSecurityGroup/defaultSecurityRules/DenyAllOutBound",
"name": "DenyAllOutBound",
"priority": 65500,
"protocol": "*",
"provisioningState": "Succeeded",
"resourceGroup": "myResourceGroup",
"sourceAddressPrefix": "*",
"sourcePortRange": "*"
}
],
"etag": "W/\"3371b313-ea9f-4687-a336-a8ebdfd80523\",
"id":
"/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetwor
kSecurityGroup",
"location": "eastus",
"name": "myNetworkSecurityGroup",
"networkInterfaces": null,
```

```

"provisioningState": "Succeeded",
"resourceGroup": "myResourceGroup",
"resourceGuid": "47a9964e-23a3-438a-a726-8d60ebbb1c3c",
"securityRules": [
  {
    "access": "Allow",
    "description": null,
    "destinationAddressPrefix": "*",
    "destinationPortRange": "22",
    "direction": "Inbound",
    "etag": "W/\"9e344b60-0daa-40a6-84f9-0ebbe4a4b640\"",
    "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup/securityRules/myNetworkSecurityGroupRuleSSH",
    "name": "myNetworkSecurityGroupRuleSSH",
    "priority": 1000,
    "protocol": "Tcp",
    "provisioningState": "Succeeded",
    "resourceGroup": "myResourceGroup",
    "sourceAddressPrefix": "*",
    "sourcePortRange": "*"
  },
  {
    "access": "Allow",
    "description": null,
    "destinationAddressPrefix": "*",
    "destinationPortRange": "80",
    "direction": "Inbound",
    "etag": "W/\"9e344b60-0daa-40a6-84f9-0ebbe4a4b640\"",
    "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup/securityRules/myNetworkSecurityGroupRuleWeb",
    "name": "myNetworkSecurityGroupRuleWeb",
    "priority": 1001,
    "protocol": "Tcp",
    "provisioningState": "Succeeded",
    "resourceGroup": "myResourceGroup",
    "sourceAddressPrefix": "*",
    "sourcePortRange": "*"
  }
],
"subnets": null,
"tags": null,
"type": "Microsoft.Network/networkSecurityGroups"
}

```

Create a virtual NIC

Virtual network interface cards (NICs) are programmatically available because you can apply rules to their use. Depending on the [VM size](#), you can attach multiple virtual NICs to a VM. In the following [az network nic create](#) command, you create a NIC named *myNic* and associate it with your network security group. The public IP address *myPublicIP* is also associated with the virtual NIC.

```

az network nic create \
  --resource-group myResourceGroup \
  --name myNic \
  --vnet-name myVnet \
  --subnet mySubnet \
  --public-ip-address myPublicIP \
  --network-security-group myNetworkSecurityGroup

```

Output:

```
{
  "NewNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": [],
      "internalDnsNameLabel": null,
      "internalDomainNameSuffix": "brqlt10lvoedgkeuomc4pm5tb.bx.internal.cloudapp.net",
      "internalFqdn": null
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "etag": "W/\"04b5ab44-d8f4-422a-9541-e5ae7de8466d\"",
    "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myNic",
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "etag": "W/\"04b5ab44-d8f4-422a-9541-e5ae7de8466d\"",
        "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myNic/ipConfigurations/ipconfig1",
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatRules": null,
        "name": "ipconfig1",
        "primary": true,
        "privateIpAddress": "192.168.1.4",
        "privateIpAddressVersion": "IPv4",
        "privateIpAllocationMethod": "Dynamic",
        "provisioningState": "Succeeded",
        "publicIpAddress": {
          "dnsSettings": null,
          "etag": null,
          "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP",
          "idleTimeoutInMinutes": null,
          "ipAddress": null,
          "ipConfiguration": null,
          "location": null,
          "name": null,
          "provisioningState": null,
          "publicIpAddressVersion": null,
          "publicIpAllocationMethod": null,
          "resourceGroup": "myResourceGroup",
          "resourceGuid": null,
          "tags": null,
          "type": null
        },
        "resourceGroup": "myResourceGroup",
        "subnet": {
          "addressPrefix": null,
          "etag": null,
          "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/myVnet/subnets/mySubnet",
          "ipConfigurations": null,
          "name": null,
          "networkSecurityGroup": null,
          "provisioningState": null,
          "resourceGroup": "myResourceGroup",
          "resourceNavigationLinks": null,
          "routeTable": null
        }
      }
    ],
    "location": "eastus",
    "macAddress": null,
    "name": "myNic",
    "networkSecurityGroup": {
      "defaultSecurityRules": null,
      "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNSG"
    }
  }
}
```

```
        "etag": null,
        "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup",
        "location": null,
        "name": null,
        "networkInterfaces": null,
        "provisioningState": null,
        "resourceGroup": "myResourceGroup",
        "resourceGuid": null,
        "securityRules": null,
        "subnets": null,
        "tags": null,
        "type": null
    },
    "primary": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "myResourceGroup",
    "resourceGuid": "b3dbaa0e-2cf2-43be-a814-5cc49fea3304",
    "tags": null,
    "type": "Microsoft.Network/networkInterfaces",
    "virtualMachine": null
}
}
```

Create an availability set

Availability sets help spread your VMs across fault domains and update domains. Even though you only create one VM right now, it's best practice to use availability sets to make it easier to expand in the future.

Fault domains define a grouping of virtual machines that share a common power source and network switch. By default, the virtual machines that are configured within your availability set are separated across up to three fault domains. A hardware issue in one of these fault domains does not affect every VM that is running your app.

Update domains indicate groups of virtual machines and underlying physical hardware that can be rebooted at the same time. During planned maintenance, the order in which update domains are rebooted might not be sequential, but only one update domain is rebooted at a time.

Azure automatically distributes VMs across the fault and update domains when placing them in an availability set. For more information, see [managing the availability of VMs](#).

Create an availability set for your VM with [az vm availability-set create](#). The following example creates an availability set named *myAvailabilitySet*:

```
az vm availability-set create \
--resource-group myResourceGroup \
--name myAvailabilitySet
```

The output notes fault domains and update domains:

```
{  
  "id":  
    "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Compute/availabilitySets/myAvailabilitySet",  
  "location": "eastus",  
  "managed": null,  
  "name": "myAvailabilitySet",  
  "platformFaultDomainCount": 2,  
  "platformUpdateDomainCount": 5,  
  "resourceGroup": "myResourceGroup",  
  "sku": {  
    "capacity": null,  
    "managed": true,  
    "tier": null  
  },  
  "statuses": null,  
  "tags": {},  
  "type": "Microsoft.Compute/availabilitySets",  
  "virtualMachines": []  
}
```

Create a VM

You've created the network resources to support Internet-accessible VMs. Now create a VM and secure it with an SSH key. In this example, let's create an Ubuntu VM based on the most recent LTS. You can find additional images with [az vm image list](#), as described in [finding Azure VM images](#).

Specify an SSH key to use for authentication. If you do not have an SSH public key pair, you can [create them](#) or use the `--generate-ssh-keys` parameter to create them for you. If you already have a key pair, this parameter uses existing keys in `~/.ssh`.

Create the VM by bringing all the resources and information together with the [az vm create](#) command. The following example creates a VM named *myVM*:

```
az vm create \  
  --resource-group myResourceGroup \  
  --name myVM \  
  --location eastus \  
  --availability-set myAvailabilitySet \  

```

SSH to your VM with the DNS entry you provided when you created the public IP address. This `fqdn` is shown in the output as you create your VM:

```
{  
  "fqdns": "mypublicdns.eastus.cloudapp.azure.com",  
  "id": "/subscriptions/guid/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",  
  "location": "eastus",  
  "macAddress": "00-0D-3A-13-71-C8",  
  "powerState": "VM running",  
  "privateIpAddress": "192.168.1.5",  
  "publicIpAddress": "13.90.94.252",  
  "resourceGroup": "myResourceGroup"  
}
```

```
ssh azureuser@mypublicdns.eastus.cloudapp.azure.com
```

Output:

```
The authenticity of host 'mypublicdns.eastus.cloudapp.azure.com (13.90.94.252)' can't be established.  
ECDSA key fingerprint is SHA256:SyIINP80Um6XRTvWiFaNz+H+1jcrKB1IIiNgCDDJRj6A.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'mypublicdns.eastus.cloudapp.azure.com,13.90.94.252' (ECDSA) to the list of known  
hosts.  
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.11.0-1016-azure x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
  
Get cloud support with Ubuntu Advantage Cloud Guest:  
https://www.ubuntu.com/business/services/cloud  
  
0 packages can be updated.  
0 updates are security updates.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
azureuser@myVM:~$
```

You can install NGINX and see the traffic flow to the VM. Install NGINX as follows:

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

To see the default NGINX site in action, open your web browser and enter your FQDN:



Export as a template

What if you now want to create an additional development environment with the same parameters, or a production environment that matches it? Resource Manager uses JSON templates that define all the parameters for your environment. You build out entire environments by referencing this JSON template. You can [build JSON](#)

templates manually or export an existing environment to create the JSON template for you. Use [az group export](#) to export your resource group as follows:

```
az group export --name myResourceGroup > myResourceGroup.json
```

This command creates the `myResourceGroup.json` file in your current working directory. When you create an environment from this template, you are prompted for all the resource names. You can populate these names in your template file by adding the `--include-parameter-default-value` parameter to the `az group export` command. Edit your JSON template to specify the resource names, or [create a parameters.json file](#) that specifies the resource names.

To create an environment from your template, use [az group deployment create](#) as follows:

```
az group deployment create \  
  --resource-group myNewResourceGroup \  
  --template-file myResourceGroup.json
```

You might want to read [more about how to deploy from templates](#). Learn about how to incrementally update environments, use the parameters file, and access templates from a single storage location.

Next steps

Now you're ready to begin working with multiple networking components and VMs. You can use this sample environment to build out your application by using the core components introduced here.

How to create a Linux virtual machine with Azure Resource Manager templates

6/20/2019 • 4 minutes to read • [Edit Online](#)

Learn how to create a Linux virtual machine (VM) by using an Azure Resource Manager template and the Azure CLI from the Azure Cloud shell. To create a Windows virtual machine, see [Create a Windows virtual machine from a Resource Manager template](#).

Templates overview

Azure Resource Manager templates are JSON files that define the infrastructure and configuration of your Azure solution. By using a template, you can repeatedly deploy your solution throughout its lifecycle and have confidence your resources are deployed in a consistent state. To learn more about the format of the template and how you construct it, see [Quickstart: Create and deploy Azure Resource Manager templates by using the Azure portal](#). To view the JSON syntax for resources types, see [Define resources in Azure Resource Manager templates](#).

Create a virtual machine

Creating an Azure virtual machine usually includes two steps:

1. 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.
2. Create a virtual machine.

The following example creates a VM from an [Azure Quickstart template](#). Only SSH authentication is allowed for this deployment. When prompted, provide the value of your own SSH public key, such as the contents of `~/.ssh/id_rsa.pub`. If you need to create an SSH key pair, see [How to create and use an SSH key pair for Linux VMs in Azure](#). Here is a copy of the template:

```
{  
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json",  
  "contentVersion": "1.0.0.0",  
  "parameters": {  
    "projectName": {  
      "type": "string",  
      "metadata": {  
        "description": "Specifies a name for generating resource names."  
      }  
    },  
    "location": {  
      "type": "string",  
      "defaultValue": "[resourceGroup().location]",  
      "metadata": {  
        "description": "Specifies the location for all resources."  
      }  
    },  
    "adminUsername": {  
      "type": "string",  
      "metadata": {  
        "description": "Specifies a username for the Virtual Machine."  
      }  
    },  
    "adminPublicKey": {  
      "type": "string",  
      "metadata": {  
        "description": "Specifies the public key for SSH authentication."  
      }  
    }  
  },  
  "resources": [  
    {  
      "type": "Microsoft.Compute/virtualMachines",  
      "name": "[parameters('projectName')]",  
      "location": "[parameters('location')]",  
      "properties": {  
        "osProfile": {  
          "computerName": "[parameters('projectName')]",  
          "adminUsername": "[parameters('adminUsername')]",  
          "linuxConfiguration": {  
            "metadata": {  
              "ssh": {  
                "publicKey": "[parameters('adminPublicKey')]"  
              }  
            }  
          }  
        },  
        "hardwareProfile": {  
          "vmSize": "Standard_DS1_v2"  
        }  
      }  
    }  
  ]  
}
```

```

        "description": "Specifies the SSH rsa public key file as a string. Use \"ssh-keygen -t rsa -b 2048\" to generate your SSH key pairs."
    }
}
},
"variables": {
    "vNetName": "[concat(parameters(' projectName'), '-vnet')]",
    "vNetAddressPrefixes": "10.0.0.0/16",
    "vNetSubnetName": "default",
    "vNetSubnetAddressPrefix": "10.0.0.0/24",
    "vmName": "[concat(parameters(' projectName'), '-vm')]",
    "publicIPAddressName": "[concat(parameters(' projectName'), '-ip')]",
    "networkInterfaceName": "[concat(parameters(' projectName'), '-nic')]",
    "networkSecurityGroupName": "[concat(parameters(' projectName'), '-nsg')]"
},
"resources": [
{
    "type": "Microsoft.Network/networkSecurityGroups",
    "apiVersion": "2018-11-01",
    "name": "[variables('networkSecurityGroupName')]",
    "location": "[parameters('location')]",
    "properties": {
        "securityRules": [
            {
                "name": "ssh_rule",
                "properties": {
                    "description": "Locks inbound down to ssh default port 22.",
                    "protocol": "Tcp",
                    "sourcePortRange": "*",
                    "destinationPortRange": "22",
                    "sourceAddressPrefix": "*",
                    "destinationAddressPrefix": "*",
                    "access": "Allow",
                    "priority": 123,
                    "direction": "Inbound"
                }
            }
        ]
    }
},
{
    "type": "Microsoft.Network/publicIPAddresses",
    "apiVersion": "2018-11-01",
    "name": "[variables('publicIPAddressName')]",
    "location": "[parameters('location')]",
    "properties": {
        "publicIPAllocationMethod": "Dynamic"
    },
    "sku": {
        "name": "Basic"
    }
},
{
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2018-11-01",
    "name": "[variables('vNetName')]",
    "location": "[parameters('location')]",
    "properties": {
        "addressSpace": {
            "addressPrefixes": [
                "[variables('vNetAddressPrefixes')]"
            ]
        },
        "subnets": [
            {
                "name": "[variables('vNetSubnetName')]",
                "properties": {
                    "addressPrefix": "[variables('vNetSubnetAddressPrefix')]"
                }
            }
        ]
    }
}
]
```

```

        }
    ]
}
},
{
    "type": "Microsoft.Network/networkInterfaces",
    "apiVersion": "2018-11-01",
    "name": "[variables('networkInterfaceName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses', variables('publicIPAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks', variables('vNetName'))]",
        "[resourceId('Microsoft.Network/networkSecurityGroups', variables('networkSecurityGroupName'))]"
    ],
    "properties": {
        "ipConfigurations": [
            {
                "name": "ipconfig1",
                "properties": {
                    "privateIPAllocationMethod": "Dynamic",
                    "publicIPAddress": {
                        "id": "[resourceId('Microsoft.Network/publicIPAddresses', variables('publicIPAddressName'))]"
                    },
                    "subnet": {
                        "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('vNetName'), variables('vNetSubnetName'))]"
                    }
                }
            }
        ]
    }
},
{
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2018-10-01",
    "name": "[variables('vmName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces', variables('networkInterfaceName'))]"
    ],
    "properties": {
        "hardwareProfile": {
            "vmSize": "Standard_D2s_v3"
        },
        "osProfile": {
            "computerName": "[variables('vmName')]",
            "adminUsername": "[parameters('adminUsername')]",
            "linuxConfiguration": {
                "disablePasswordAuthentication": true,
                "ssh": {
                    "publicKeys": [
                        {
                            "path": "[concat('/home/', parameters('adminUsername'), '/.ssh/authorized_keys')]",
                            "keyData": "[parameters('adminPublicKey')]"
                        }
                    ]
                }
            }
        },
        "storageProfile": {
            "imageReference": {
                "publisher": "Canonical",
                "offer": "UbuntuServer",
                "sku": "18.04-LTS",
                "version": "latest"
            },
            "osDisk": {
                "createOption": "fromImage"
            }
        }
    }
}

```

```

        },
      "networkProfile": {
        "networkInterfaces": [
          {
            "id": "[resourceId('Microsoft.Network/networkInterfaces', variables('networkInterfaceName'))]"
          }
        ]
      }
    ],
  "outputs": {
    "adminUsername": {
      "type": "string",
      "value": "[parameters('adminUsername')]"
    }
  }
}

```

To run the CLI script, Select **Try it** to open the Azure Cloud shell. To paste the script, right-click the shell, and then select **Paste**:

```

echo "Enter the Resource Group name:" &&
read resourceGroupName &&
echo "Enter the location (i.e. centralus):" &&
read location &&
echo "Enter the project name (used for generating resource names):" &&
read projectName &&
echo "Enter the administrator username:" &&
read username &&
echo "Enter the SSH public key:" &&
read key &&
az group create --name $resourceGroupName --location "$location" &&
az group deployment create --resource-group $resourceGroupName --template-uri
https://raw.githubusercontent.com/azure/azure-quickstart-templates/master/101-vm-sshkey/azuredeploy.json -->
parameters projectName=$projectName adminUsername=$username adminPublicKey="$key" &&
az vm show --resource-group $resourceGroupName --name "$projectName-vm" --show-details --query publicIps -->
output tsv

```

The last Azure CLI command shows the public IP address of the newly created VM. You need the public IP address to connect to the virtual machine. See the next section of this article.

In the previous example, you specified a template stored in GitHub. You can also download or create a template and specify the local path with the `--template-file` parameter.

Here are some additional resources:

- To learn how to develop Resource Manager templates, see [Azure Resource Manager documentation](#).
- To see the Azure virtual machine schemas, see [Azure template reference](#).
- To see more virtual machine template samples, see [Azure Quickstart templates](#).

Connect to virtual machine

You can then SSH to your VM as normal. Provide your own public IP address from the preceding command:

```
ssh <adminUsername>@<ipAddress>
```

Next steps

In this example, you created a basic Linux VM. For more Resource Manager templates that include application frameworks or create more complex environments, browse the [Azure Quickstart templates](#).

To learn more about creating templates, view the JSON syntax and properties for the resources types you deployed:

- [Microsoft.Network/networkSecurityGroups](#)
- [Microsoft.Network/publicIPAddresses](#)
- [Microsoft.Network/virtualNetworks](#)
- [Microsoft.Network/networkInterfaces](#)
- [Microsoft.Compute/virtualMachines](#)

Create a Linux virtual machine that uses SSH authentication with the REST API

3/5/2019 • 3 minutes to read • [Edit Online](#)

A Linux virtual machine (VM) in Azure consists of various resources such as disks and network interfaces and defines parameters such as location, size and operating system image and authentication settings.

You can create a Linux VM via the Azure portal, Azure CLI 2.0, many Azure SDKs, Azure Resource Manager templates and many third-party tools such as Ansible or Terraform. All these tools ultimately use the REST API to create the Linux VM.

This article shows you how to use the REST API to create a Linux VM running Ubuntu 18.04-LTS with managed disks and SSH authentication.

Before you start

Before you create and submit the request, you will need:

- The `{subscription-id}` for your subscription
 - If you have multiple subscriptions, see [Working with multiple subscriptions](#)
- A `{resourceGroupName}` you've created ahead of time
- A [virtual network interface](#) in the same resource group
- An SSH key pair (you can [generate a new one](#) if you don't have one)

Request basics

To create or update a virtual machine, use the following *PUT* operation:

```
PUT https://management.azure.com/subscriptions/{subscription-
id}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmName}?api-version=2017-
12-01
```

In addition to the `{subscription-id}` and `{resourceGroupName}` parameters, you'll need to specify the `{vmName}` (`api-version` is optional, but this article was tested with `api-version=2017-12-01`)

The following headers are required:

REQUEST HEADER	DESCRIPTION
<code>Content-Type:</code>	Required. Set to <code>application/json</code> .
<code>Authorization:</code>	Required. Set to a valid <code>Bearer</code> access token .

For general information about working with REST API requests, see [Components of a REST API request/response](#).

Create the request body

The following common definitions are used to build a request body:

NAME	REQUIRED	TYPE	DESCRIPTION
location	True	string	Resource location.
name		string	Name for the virtual machine.
properties.hardwareProfile		HardwareProfile	Specifies the hardware settings for the virtual machine.
properties.storageProfile		StorageProfile	Specifies the storage settings for the virtual machine disks.
properties.osProfile		OSProfile	Specifies the operating system settings for the virtual machine.
properties.networkProfile		NetworkProfile	Specifies the network interfaces of the virtual machine.

An example request body is below. Make sure you specify the VM name in the `{computerName}` and `{name}` parameters, the name of the network interface you've created under `networkInterfaces`, your username in `adminUsername` and `path`, and the *public* portion of your SSH keypair (located in, for example, `~/.ssh/id_rsa.pub`) in `keyData`. Other parameters you might want to modify include `location` and `vmSize`.

```
{
  "location": "eastus",
  "name": "{vmName}",
  "properties": {
    "hardwareProfile": {
      "vmSize": "Standard_DS1_v2"
    },
    "storageProfile": {
      "imageReference": {
        "sku": "18.04-LTS",
        "publisher": "Canonical",
        "version": "latest",
        "offer": "UbuntuServer"
      },
      "osDisk": {
        "caching": "ReadWrite",
        "managedDisk": {
          "storageAccountType": "Premium_LRS"
        },
        "name": "myVMosdisk",
        "createOption": "FromImage"
      }
    },
    "osProfile": {
      "adminUsername": "{your-username}",
      "computerName": "{vmName}",
      "linuxConfiguration": {
        "ssh": {
          "publicKeys": [
            {
              "path": "/home/{your-username}/.ssh/authorized_keys",
              "keyData": "ssh-rsa AAAAB3NzaC1{snip}mf69/J1"
            }
          ]
        },
        "disablePasswordAuthentication": true
      }
    },
    "networkProfile": {
      "networkInterfaces": [
        {
          "id": "/subscriptions/{subscription-
id}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkInterfaces/{existing-nic-name}",
          "properties": {
            "primary": true
          }
        }
      ]
    }
  }
}
```

For a complete list of the available definitions in the request body, see [Virtual machines create or update request body definitions](#).

Sending the request

You may use the client of your preference for sending this HTTP request. You may also use an [in-browser tool](#) by clicking the **Try it** button.

Responses

There are two successful responses for the operation to create or update a virtual machine:

NAME	TYPE	DESCRIPTION
200 OK	VirtualMachine	OK
201 Created	VirtualMachine	Created

A condensed *201 Created* response from the previous example request body that creates a VM shows a *vmId* has been assigned and the *provisioningState* is *Creating*:

```
{
  "vmId": "e0de9b84-a506-4b95-9623-00a425d05c90",
  "provisioningState": "Creating"
}
```

For more information about REST API responses, see [Process the response message](#).

Next steps

For more information on the Azure REST APIs or other management tools such as Azure CLI or Azure PowerShell, see the following:

- [Azure Compute provider REST API](#)
- [Get started with Azure REST API](#)
- [Azure CLI](#)
- [Azure PowerShell module](#)

Create a copy of a Linux VM by using Azure CLI and Managed Disks

2/22/2019 • 2 minutes to read • [Edit Online](#)

This article shows you how to create a copy of your Azure virtual machine (VM) running Linux by using the Azure CLI and the Azure Resource Manager deployment model.

You can also [upload and create a VM from a VHD](#).

Prerequisites

- Install the [Azure CLI](#).
- Sign in to an Azure account with [az login](#).
- Have an Azure VM to use as the source for your copy.

Stop the source VM

Deallocate the source VM by using [az vm deallocate](#). The following example deallocates the VM named *myVM* in the resource group *myResourceGroup*:

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

Copy the source VM

To copy a VM, you create a copy of the underlying virtual hard disk. This process creates a specialized virtual hard disk (VHD) as a Managed Disk that contains the same configuration and settings as the source VM.

For more information about Azure Managed Disks, see [Azure Managed Disks overview](#).

1. List each VM and the name of its OS disk with [az vm list](#). The following example lists all VMs in the resource group named *myResourceGroup*:

```
az vm list -g myResourceGroup \
    --query '[].{Name:name,DiskName:storageProfile.osDisk.name}' \
    --output table
```

The output is similar to the following example:

Name	DiskName
myVM	myDisk

2. Copy the disk by creating a new managed disk and by using [az disk create](#). The following example creates a disk named *myCopiedDisk* from the managed disk named *myDisk*:

```
az disk create --resource-group myResourceGroup \
--name myCopiedDisk --source myDisk
```

3. Verify the managed disks now in your resource group by using [az disk list](#). The following example lists the managed disks in the resource group named *myResourceGroup*:

```
az disk list --resource-group myResourceGroup --output table
```

Set up a virtual network

The following optional steps create a new virtual network, subnet, public IP address, and virtual network interface card (NIC).

If you are copying a VM for troubleshooting purposes or additional deployments, you might not want to use a VM in an existing virtual network.

If you want to create a virtual network infrastructure for your copied VMs, follow the next few steps. If you don't want to create a virtual network, skip to [Create a VM](#).

1. Create the virtual network by using [az network vnet create](#). The following example creates a virtual network named *myVnet* and a subnet named *mySubnet*:

```
az network vnet create --resource-group myResourceGroup \
--location eastus --name myVnet \
--address-prefix 192.168.0.0/16 \
--subnet-name mySubnet \
--subnet-prefix 192.168.1.0/24
```

2. Create a public IP by using [az network public-ip create](#). The following example creates a public IP named *myPublicIP* with the DNS name of *mypublicdns*. (Because the DNS name must be unique, provide a unique name.)

```
az network public-ip create --resource-group myResourceGroup \
--location eastus --name myPublicIP --dns-name mypublicdns \
--allocation-method static --idle-timeout 4
```

3. Create the NIC by using [az network nic create](#). The following example creates a NIC named *myNic* that's attached to the *mySubnet* subnet:

```
az network nic create --resource-group myResourceGroup \
--location eastus --name myNic \
--vnet-name myVnet --subnet mySubnet \
--public-ip-address myPublicIP
```

Create a VM

Create a VM by using [az vm create](#).

Specify the copied managed disk to use as the OS disk (`--attach-os-disk`), as follows:

```
az vm create --resource-group myResourceGroup \
--name myCopiedVM --nics myNic \
--size Standard_DS1_v2 --os-type Linux \
--attach-os-disk myCopiedDisk
```

Next steps

To learn how to use Azure CLI to manage your new VM, see [Azure CLI commands for the Azure Resource Manager](#).

Manage virtual machine access using just-in-time

6/20/2019 • 11 minutes to read • [Edit Online](#)

Just-in-time (JIT) virtual machine (VM) access can be used to lock down inbound traffic to your Azure VMs, reducing exposure to attacks while providing easy access to connect to VMs when needed.

NOTE

The just-in-time feature is available on the Standard tier of Security Center. See [Pricing](#) to learn more about Security Center's pricing tiers.

NOTE

Security Center just-in-time VM access currently supports only VMs deployed through Azure Resource Manager. To learn more about the classic and Resource Manager deployment models see [Azure Resource Manager vs. classic deployment](#).

Attack scenario

Brute force attacks commonly target management ports as a means to gain access to a VM. If successful, an attacker can take control over the VM and establish a foothold into your environment.

One way to reduce exposure to a brute force attack is to limit the amount of time that a port is open. Management ports do not need to be open at all times. They only need to be open while you are connected to the VM, for example to perform management or maintenance tasks. When just-in-time is enabled, Security Center uses [network security group](#) (NSG) and Azure Firewall rules, which restrict access to management ports so they cannot be targeted by attackers.



How does JIT access work?

When just-in-time is enabled, Security Center locks down inbound traffic to your Azure VMs by creating an NSG rule. You select the ports on the VM to which inbound traffic will be locked down. These ports are controlled by the just-in-time solution.

When a user requests access to a VM, Security Center checks that the user has [Role-Based Access Control \(RBAC\)](#) permissions that permit them to successfully request access to a VM. If the request is approved, Security Center automatically configures the Network Security Groups (NSGs) and Azure Firewall to allow inbound traffic to the selected ports and requested source IP addresses or ranges, for the amount of time that was specified. After the time has expired, Security Center restores the NSGs to their previous states. Those connections that are already established are not being interrupted, however.

NOTE

If a JIT access request is approved for a VM behind an Azure Firewall, then Security Center automatically changes both the NSG and firewall policy rules. For the amount of time that was specified, the rules allow inbound traffic to the selected ports and requested source IP addresses or ranges. After the time is over, Security Center restores the firewall and NSG rules to their previous states.

Permissions needed to configure and use JIT

TO ENABLE A USER TO:	PERMISSIONS TO SET
Configure or edit a JIT policy for a VM	<p>Assign these actions to the role: On the scope of a subscription or Resource Group that is associated with the VM:</p> <p><code>Microsoft.Security/locations/jitNetworkAccessPolicies/write</code> On the scope of a subscription or Resource Group or VM:</p> <p><code>Microsoft.Compute/virtualMachines/write</code></p>

TO ENABLE A USER TO:	PERMISSIONS TO SET
Request JIT access to a VM	<p>Assign these actions to the user: On the scope of a subscription or Resource Group that is associated with the VM:</p> <pre>Microsoft.Security/locations/{the_location_of_the_VM}/jitNetworkAccessPolicies</pre> <p>On the scope of a Subscription or Resource Group or VM:</p> <pre>Microsoft.Compute/virtualMachines/read</pre>

Configure JIT on a VM

There are 3 ways to configure a JIT policy on a VM:

- [Configure JIT access in Azure Security Center](#)
- [Configure JIT access in an Azure VM blade](#)
- [Configure a JIT policy on a VM programmatically](#)

Configure JIT in ASC

From ASC, you can configure a JIT policy and request access to a VM using a JIT policy

Configure JIT access on a VM in ASC

1. Open the **Security Center** dashboard.
2. In the left pane, select **Just-in-time VM access**.

The screenshot shows the Azure Security Center - Overview dashboard for the subscription 'Contoso IT - demo'. The left sidebar navigation includes sections like Policy & Compliance, Resource Security Hygiene, Threat Protection, Automation & Orchestration, and Advanced Cloud Defense. The 'Just in time VM access' option is highlighted with a red box. The main content area displays four main sections: Policy & compliance, Resource security hygiene, Threat protection, and Resource health monitoring. The 'Policy & compliance' section shows subscription coverage (1 total, 1 covered standard, 0 covered free, 0 not covered), policy compliance (36%), and least compliant subscriptions (Contoso IT - demo at 36%). The 'Resource security hygiene' section shows recommendations (36 total, 15 high severity, 12 medium severity, 9 low severity) and resource health monitoring for Compute & apps, Data & storage, Networking, and Identity & access. The 'Threat protection' section shows security alerts by severity (30 total, 4 high severity, 23 medium severity, 3 low severity) and security alerts over time. The 'Resource health monitoring' section shows the status of various service categories.

The **Just-in-time VM access** window opens.

Virtual machines

Configured Recommended No recommendation

VMs for which we recommend you to apply the just in time VM access control.

61 VMs

[Enable JIT on 2 VMs](#)

Search to filter items...

VIRTUAL MACHINE	STATE	SEVERITY
AA-Contoso-01	Open	High
App01	Open	High
App03	Open	High
App04	Open	High
App05	Open	High
App06	Open	High
App07	Open	High
App08	Open	High
App09	Open	High

Just-in-time VM access provides information on the state of your VMs:

- **Configured** - VMs that have been configured to support just-in-time VM access. The data presented is for the last week and includes for each VM the number of approved requests, last access date and time, and last user.
- **Recommended** - VMs that can support just-in-time VM access but have not been configured to. We recommend that you enable just-in-time VM access control for these VMs.
- **No recommendation** - Reasons that can cause a VM not to be recommended are:
 - Missing NSG - The just-in-time solution requires an NSG to be in place.
 - Classic VM - Security Center just-in-time VM access currently supports only VMs deployed through Azure Resource Manager. A classic deployment is not supported by the just-in-time solution.
 - Other - A VM is in this category if the just-in-time solution is turned off in the security policy of the subscription or the resource group, or if the VM is missing a public IP and doesn't have an NSG in place.

3. Select the **Recommended** tab.
4. Under **VIRTUAL MACHINE**, click the VMs that you want to enable. This puts a checkmark next to a VM.
5. Click **Enable JIT on VMs**. This blade displays the default ports recommended by Azure Security Center:
 - 22 - SSH
 - 3389 - RDP
 - 5985 - WinRM
 - 5986 - WinRM
6. You can also configure custom ports:
 - a. Click **Add**. The **Add port configuration** window opens.
 - b. For each port you choose to configure, both default and custom, you can customize the following settings:
 - **Protocol type**- The protocol that is allowed on this port when a request is approved.
 - **Allowed source IP addresses**- The IP ranges that are allowed on this port when a request is approved.
 - **Maximum request time**- The maximum time window during which a specific port can be opened.
 - c. Click **OK**.
7. Click **Save**.

NOTE

When JIT VM Access is enabled for a VM, Azure Security Center creates "deny all inbound traffic" rules for the selected ports in the network security groups associated and Azure Firewall with it. If other rules had been created for the selected ports, then the existing rules take priority over the new "deny all inbound traffic" rules. If there are no existing rules on the selected ports, then the new "deny all inbound traffic" rules take top priority in the Network Security Groups and Azure Firewall.

Request JIT access via ASC

To request access to a VM via ASC:

1. Under **Just in time VM access**, select the **Configured** tab.
2. Under **Virtual Machine**, click the VMs that you want to request access for. This puts a checkmark next to the VM.
 - The icon in the **Connection Details** column indicates whether JIT is enabled on the NSG or FW. If it's enabled on both, only the Firewall icon appears.
 - The **Connection Details** column provides the correct information required to connect the VM, as well as indicates the opened ports.

Virtual machines						
	Configured	Recommended	No recommendation			
VMs for which the just in time VM access control is already in place. Presented data is for the last week.						
92 VMs						
<input type="checkbox"/> Search to filter items...						
VIRTUAL MACHINE	APPROVED	LAST ACCESS	CONNECTION DETAILS	LAST USER		
af-vm	1 Requests	5/7/19, 11:05 AM	13.64.24.215:13389	user@contoso.com	...	
svrworkload2	1 Requests	5/7/19, 11:30 AM	20.185.107.87:10022	user@contoso.com	...	
sc2019	2 Requests	5/7/19, 11:39 AM	3 Ports	user@contoso.com	...	
bengr-jit-mul-1	1 Requests	Active now	Ports: 5986, 22, 3389	user@contoso.com	...	
LBWeb0	0 Requests	N/A	-	N/A	...	
LBWeb1	0 Requests	N/A	-	N/A	...	
muliport1	0 Requests	N/A	-	N/A	...	
<input checked="" type="checkbox"/> muliport0	0 Requests	N/A	-	N/A	...	
WebApp1	0 Requests	N/A	-	N/A	...	
WinVM	0 Requests	N/A	-	N/A	...	
vm2	0 Requests	N/A	-	N/A	...	
BarWaff2Jun3	0 Requests	N/A	-	N/A	...	
bengr-jit-mul-2	0 Requests	N/A	-	N/A	...	
ChkpJun3	0 Requests	N/A	-	N/A	...	
...	

3. Click **Request access**. The **Request access** window opens.

Request access

Please select the ports that you would like to open per virtual machine.

PORT	TOGGLE	ALLOWED SOURCE IP	IP RANGE	TIMERANGE
▼ vm1				
22	On Off	My IP IP Range	No range	
3389	On Off	My IP IP Range	No range	
5985	On Off	My IP IP Range	No range	
5986	On Off	My IP IP Range	No range	
▼ vm2				
22	On Off	My IP IP Range	No range	
3389	On Off	My IP IP Range	No range	
5985	On Off	My IP IP Range	No range	
5986	On Off	My IP IP Range	No range	

Open ports

- Under **Request access**, for each VM, configure the ports that you want to open and the source IP addresses that the port is opened on and the time window for which the port will be open. It will only be possible to request access to the ports that are configured in the just-in-time policy. Each port has a maximum allowed time derived from the just-in-time policy.
- Click **Open ports**.

NOTE

If a user who is requesting access is behind a proxy, the option **My IP** may not work. You may need to define the full IP address range of the organization.

Edit a JIT access policy via ASC

You can change a VM's existing just-in-time policy by adding and configuring a new port to protect for that VM, or by changing any other setting related to an already protected port.

To edit an existing just-in-time policy of a VM:

- In the **Configured** tab, under **VMs**, select a VM to which to add a port by clicking on the three dots within the row for that VM.
- Select **Edit**.
- Under **JIT VM access configuration**, you can either edit the existing settings of an already protected port or add a new custom port.

The screenshot shows the JIT VM access configuration for a virtual machine named 'vm1'. It includes a table for defining port rules:

PORT	PROT...	ALLOWED SOU...	IP RANGE	TIME RANGE
22	Any	Per request	N/A	3 hours
3389	Any	Per request	N/A	3 hours

Audit JIT access activity in ASC

You can gain insights into VM activities using log search. To view logs:

- Under **Just-in-time VM access**, select the **Configured** tab.
- Under **VMs**, select a VM to view information about by clicking on the three dots within the row for that VM and select **Activity Log** in the menu. The **Activity log** opens.

The screenshot shows the JIT VM access configuration for a virtual machine named 'vm1'. It includes a table for defining port rules:

PORT	PROT...	ALLOWED SOU...	IP RANGE	TIME RANGE
22	Any	Per request	N/A	3 hours
3389	Any	Per request	N/A	3 hours

Activity log provides a filtered view of previous operations for that VM along with time, date, and subscription.

You can download the log information by selecting **Click here to download all the items as CSV**.

Modify the filters and click **Apply** to create a search and log.

Configure JIT access in an Azure VM blade

For your convenience, you can connect to a VM using JIT directly from within the VM blade in Azure.

Configure JIT access on a VM via the Azure VM blade

To make it easy to roll out just-in-time access across your VMs, you can set a VM to allow only just-in-time access directly from within the VM.

1. In the Azure portal, select **Virtual machines**.
2. Click on the virtual machine you want to limit to just-in-time access.
3. In the menu, click **Configuration**.
4. Under **Just-in-time-access** click **Enable just-in-time policy**.

This enables just-in-time access for the VM using the following settings:

- Windows servers:
 - RDP port 3389
 - 3 hours of maximum allowed access
 - Allowed source IP addresses is set to Any
- Linux servers:
 - SSH port 22
 - 3 hours of maximum allowed access
 - Allowed source IP addresses is set to Any

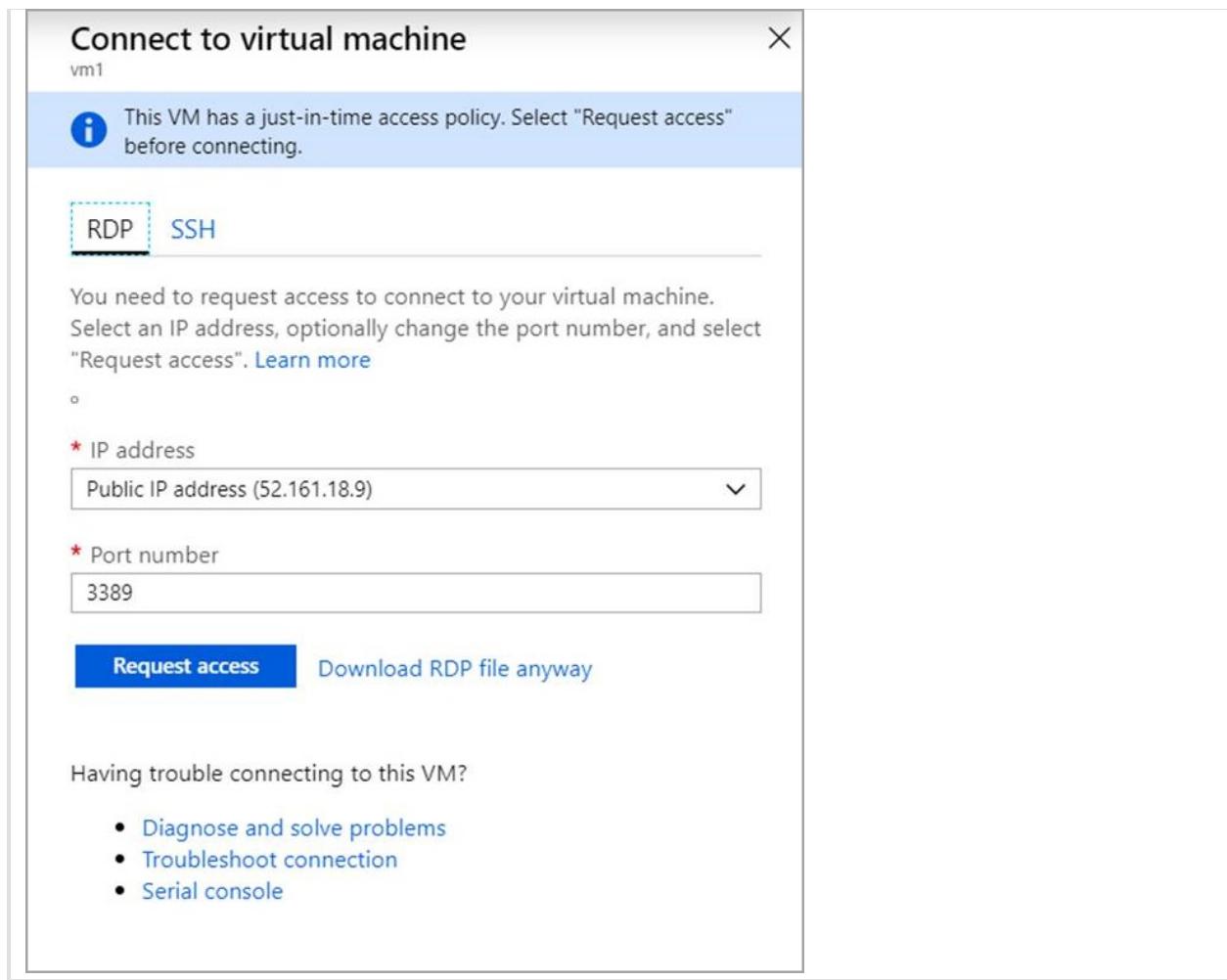
If a VM already has just-in-time enabled, when you go to its configuration page you will be able to see that just-in-time is enabled and you can use the link to open the policy in Azure Security Center to view and change the settings.

The screenshot shows the Azure portal interface. On the left, there's a sidebar titled 'Virtual machines' with a list of VMs. One VM, 'vm-contoso-us', is selected and highlighted with a red box. On the right, the main content area is titled 'vm-contoso-us - Configuration'. It contains several sections: 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Settings' (with sub-options like 'Networking', 'Disks', 'Size', 'Security', 'Extensions', 'Continuous delivery (Preview)', 'Availability set'), and 'Identity (Preview)'. At the bottom of the configuration list, there's a section titled 'Just-in-time access' with a sub-section 'To improve security, enable a just-in-time access policy.' containing a blue 'Enable just-in-time policy' button. Below this, there's a section for 'Azure hybrid benefit' with a 'Use existing Windows license' link and 'No' and 'Yes' buttons. The 'Configuration' section at the very bottom is also highlighted with a red box.

Request JIT access to a VM via the Azure VM blade

In the Azure portal, when you try to connect to a VM, Azure checks to see if you have a just-in-time access policy configured on that VM.

- If you do have a JIT policy configured on the VM, you can click **Request access** to enable you to have access in accordance with the JIT policy set for the VM.



Having trouble connecting to this VM?

- [Diagnose and solve problems](#)
- [Troubleshoot connection](#)
- [Serial console](#)

The access is requested with the following default parameters:

- **source IP:** 'Any' (*) (cannot be changed)
- **time range:** 3 hours (cannot be changed)
- **port number** RDP port 3389 for Windows / port 22 for Linux (can be changed)

NOTE

After a request is approved for a VM protected by Azure Firewall, Security Center provides the user with the proper connection details (the port mapping from the DNAT table) to use to connect to the VM.

- If you do not have JIT configured on a VM, you will be prompted to configure a JIT policy it.

To connect to your virtual machine via RDP, select an IP address, optionally change the port number, and download the RDP file.

* IP address
Public IP address (40.124.37.238)

* Port number
3389

Download RDP File

i Inbound traffic to the Public IP address may be blocked. You can update inbound port rules in the **VM Networking** page.

wrench You can troubleshoot VM connection issues by opening the **Diagnose and solve problems** page.

Configure a JIT policy on a VM programmatically

You can set up and use just-in-time via REST APIs and via PowerShell.

JIT VM access via REST APIs

The just-in-time VM access feature can be used via the Azure Security Center API. You can get information about configured VMs, add new ones, request access to a VM, and more, via this API. See [Jit Network Access Policies](#), to learn more about the just-in-time REST API.

JIT VM access via PowerShell

To use the just-in-time VM access solution via PowerShell, use the official Azure Security Center PowerShell cmdlets, and specifically `Set-AzJitNetworkAccessPolicy`.

The following example sets a just-in-time VM access policy on a specific VM, and sets the following:

1. Close ports 22 and 3389.
2. Set a maximum time window of 3 hours for each so they can be opened per approved request.
3. Allows the user who is requesting access to control the source IP addresses and allows the user to establish a successful session upon an approved just-in-time access request.

Run the following in PowerShell to accomplish this:

1. Assign a variable that holds the just-in-time VM access policy for a VM:

```
$JitPolicy = (@{  
    id="/subscriptions/SUBSCRIPTIONID/resourceGroups/RESOURCEGROUP/providers/Microsoft.Compute/virtualMachines/VMNAME"  
    ports=@{  
        number=22;  
        protocol="*";  
        allowedSourceAddressPrefix=@("*");  
        maxRequestAccessDuration="PT3H"}},  
    @{  
        number=3389;  
        protocol="*";  
        allowedSourceAddressPrefix=@("*");  
        maxRequestAccessDuration="PT3H"}})}
```

2. Insert the VM just-in-time VM access policy to an array:

```
$JitPolicyArr=@($JitPolicy)
```

3. Configure the just-in-time VM access policy on the selected VM:

```
Set-AzJitNetworkAccessPolicy -Kind "Basic" -Location "LOCATION" -Name "default" -ResourceGroupName "RESOURCEGROUP" -VirtualMachine $JitPolicyArr
```

Request access to a VM via PowerShell

In the following example, you can see a just-in-time VM access request to a specific VM in which port 22 is requested to be opened for a specific IP address and for a specific amount of time:

Run the following in PowerShell:

1. Configure the VM request access properties

```
$JitPolicyVm1 = (@{  
    id="/SUBSCRIPTIONID/resourceGroups/RESOURCEGROUP/providers/Microsoft.Compute/virtualMachines/VMNAME"  
    ports=@{  
        number=22;  
        endTimeUtc="2018-09-17T17:00:00.3658798Z";  
        allowedSourceAddressPrefix=@("IPV4ADDRESS")}}})
```

2. Insert the VM access request parameters in an array:

```
$JitPolicyArr=@($JitPolicyVm1)
```

3. Send the request access (use the resource ID you got in step 1)

```
Start-AzJitNetworkAccessPolicy -ResourceId  
"/subscriptions/SUBSCRIPTIONID/resourceGroups/RESOURCEGROUP/providers/Microsoft.Security/locations/LOCATION/jitNetworkAccessPolicies/default" -VirtualMachine $JitPolicyArr
```

For more information, see the PowerShell cmdlet documentation.

Next steps

In this article, you learned how just-in-time VM access in Security Center helps you control access to your Azure virtual machines.

To learn more about Security Center, see the following:

- [Setting security policies](#) — Learn how to configure security policies for your Azure subscriptions and resource groups.
- [Managing security recommendations](#) — Learn how recommendations help you protect your Azure resources.
- [Security health monitoring](#) — Learn how to monitor the health of your Azure resources.
- [Managing and responding to security alerts](#) — Learn how to manage and respond to security alerts.
- [Monitoring partner solutions](#) — Learn how to monitor the health status of your partner solutions.
- [Security Center FAQ](#) — Find frequently asked questions about using the service.
- [Azure Security blog](#) — Find blog posts about Azure security and compliance.

How to encrypt a Linux virtual machine in Azure

2/1/2019 • 5 minutes to read • [Edit Online](#)

For enhanced virtual machine (VM) security and compliance, virtual disks and the VM itself can be encrypted. VMs are encrypted using cryptographic keys that are secured in an Azure Key Vault. You control these cryptographic keys and can audit their use. This article details how to encrypt virtual disks on a Linux VM using the Azure CLI.

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 choose to install and use the CLI locally, this article requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Overview of disk encryption

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. You retain control of these cryptographic keys and can audit their use. These cryptographic keys are used to encrypt and decrypt virtual disks attached to your VM.

The process for encrypting a VM is as follows:

1. Create a cryptographic key in an Azure Key Vault.
2. Configure the cryptographic key to be usable for encrypting disks.
3. Enable disk encryption for your virtual disks.
4. The required cryptographic keys are requested from Azure Key Vault.
5. The virtual disks are encrypted using the provided cryptographic key.

Requirements and limitations

Supported scenarios and requirements for disk encryption:

- The following Linux server SKUs - Ubuntu, CentOS, SUSE and SUSE Linux Enterprise Server (SLES), and Red Hat Enterprise Linux.
- All resources (such as Key Vault, Storage account, and VM) must be in the same Azure region and subscription.
- Standard A, D, DS, G, GS, etc., series VMs.
- Updating the cryptographic keys on an already encrypted Linux VM.

Disk encryption is not currently supported in the following scenarios:

- Basic tier VMs.
- VMs created using the Classic deployment model.
- Disabling OS disk encryption on Linux VMs.

- Use of custom Linux images.

For more information on supported scenarios and limitations, see [Azure Disk Encryption for IaaS VMs](#)

Create an Azure Key Vault and keys

In the following examples, replace example parameter names with your own values. Example parameter names include *myResourceGroup*, *myKey*, and *myVM*.

The first step is to create an Azure Key Vault to store your cryptographic keys. Azure Key Vault can store keys, secrets, or passwords that allow you to securely implement them in your applications and services. For virtual disk encryption, you use Key Vault to store a cryptographic key that is used to encrypt or decrypt your virtual disks.

Enable the Azure Key Vault provider within your Azure subscription with [az provider register](#) and create a resource group with [az group create](#). The following example creates a resource group name *myResourceGroup* in the **eastus** location:

```
az provider register -n Microsoft.KeyVault
resourcegroup="myResourceGroup"
az group create --name $resourcegroup --location eastus
```

The Azure Key Vault containing the cryptographic keys and associated compute resources such as storage and the VM itself must reside in the same region. Create an Azure Key Vault with [az keyvault create](#) and enable the Key Vault for use with disk encryption. Specify a unique Key Vault name for *keyvault_name* as follows:

```
keyvault_name=myvaultname$RANDOM
az keyvault create \
--name $keyvault_name \
--resource-group $resourcegroup \
--location eastus \
--enabled-for-disk-encryption True
```

You can store cryptographic keys using software or Hardware Security Model (HSM) protection. Using an HSM requires a premium Key Vault. There is an additional cost to creating a premium Key Vault rather than standard Key Vault that stores software-protected keys. To create a premium Key Vault, in the preceding step add **--sku Premium** to the command. The following example uses software-protected keys since you created a standard Key Vault.

For both protection models, the Azure platform needs to be granted access to request the cryptographic keys when the VM boots to decrypt the virtual disks. Create a cryptographic key in your Key Vault with [az keyvault key create](#). The following example creates a key named *myKey*:

```
az keyvault key create \
--vault-name $keyvault_name \
--name myKey \
--protection software
```

Create a virtual machine

Create a VM with [az vm create](#) and attach a 5Gb data disk. Only certain marketplace images support disk encryption. The following example creates a VM named *myVM* using an *Ubuntu 16.04 LTS* image:

```
az vm create \
--resource-group $resourcegroup \
--name myVM \
--image Canonical:UbuntuServer:16.04-LTS:latest \
--admin-username azureuser \
--generate-ssh-keys \
--data-disk-sizes-gb 5
```

SSH to your VM using the *publicIpAddress* shown in the output of the preceding command. Create a partition and filesystem, then mount the data disk. For more information, see [Connect to a Linux VM to mount the new disk](#). Close your SSH session.

Encrypt the virtual machine

Encrypt your VM with [az vm encryption enable](#):

```
az vm encryption enable \
--resource-group $resourcegroup \
--name myVM \
--disk-encryption-keyvault $keyvault_name \
--key-encryption-key myKey \
--volume-type all
```

It takes some time for the disk encryption process to complete. Monitor the status of the process with [az vm encryption show](#):

```
az vm encryption show --resource-group $resourcegroup --name myVM --query 'status'
```

When complete, the output will look similar to the following example:

```
[  
 {  
   "code": "ProvisioningState/succeeded",  
   "displayStatus": "Provisioning succeeded",  
   "level": "Info",  
   "message": "Encryption succeeded for all volumes",  
   "time": null  
 }  
]
```

Add additional data disks

Once you have encrypted your data disks, you can add additional virtual disks to your VM and encrypt them.

Once the data disk has been added to the VM, rerun the command to encrypt the virtual disks as follows:

```
az vm encryption enable \
--resource-group $resourcegroup \
--name myVM \
--disk-encryption-keyvault $keyvault_name \
--key-encryption-key myKey \
--volume-type data
```

Next steps

- For more information about managing Azure Key Vault, including deleting cryptographic keys and vaults, see [Manage Key Vault using CLI](#).
- For more information about disk encryption, such as preparing an encrypted custom VM to upload to Azure, see [Azure Disk Encryption](#).

What is role-based access control (RBAC) for Azure resources?

6/19/2019 • 7 minutes to read • [Edit Online](#)

Access management for cloud resources is a critical function for any organization that is using the cloud. Role-based access control (RBAC) helps you manage who has access to Azure resources, what they can do with those resources, and what areas they have access to.

RBAC is an authorization system built on [Azure Resource Manager](#) that provides fine-grained access management of Azure resources.

What can I do with RBAC?

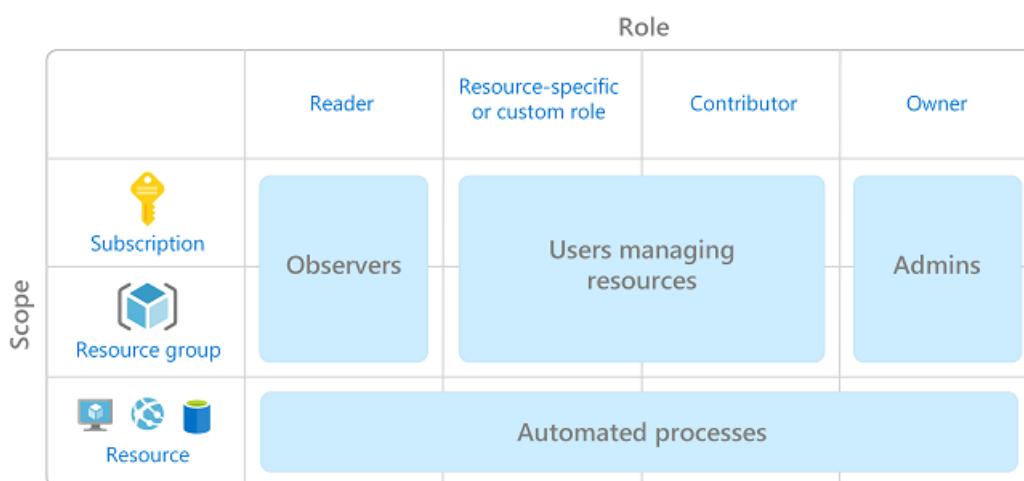
Here are some examples of what you can do with RBAC:

- Allow one user to manage virtual machines in a subscription and another user to manage virtual networks
- Allow a DBA group to manage SQL databases in a subscription
- Allow a user to manage all resources in a resource group, such as virtual machines, websites, and subnets
- Allow an application to access all resources in a resource group

Best practice for using RBAC

Using RBAC, you can segregate duties within your team and grant only the amount of access to users that they need to perform their jobs. Instead of giving everybody unrestricted permissions in your Azure subscription or resources, you can allow only certain actions at a particular scope.

When planning your access control strategy, it's a best practice to grant users the least privilege to get their work done. The following diagram shows a suggested pattern for using RBAC.



How RBAC works

The way you control access to resources using RBAC is to create role assignments. This is a key concept to understand – it's how permissions are enforced. A role assignment consists of three elements: security principal, role definition, and scope.

Security principal

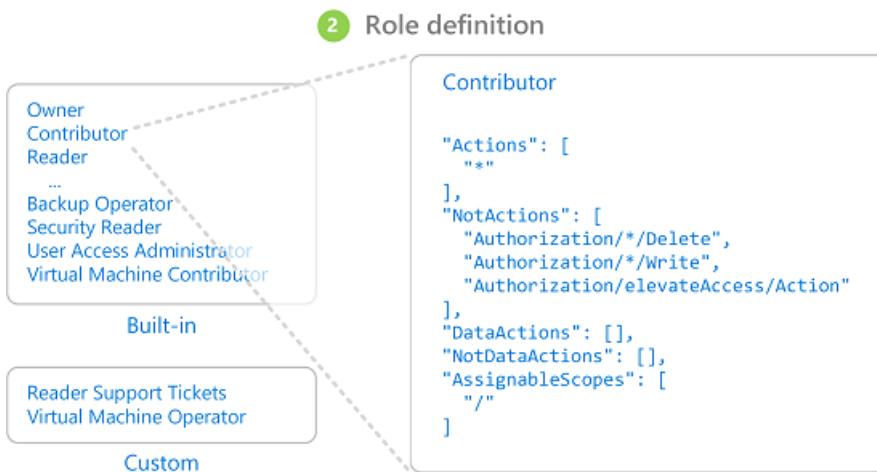
A *security principal* is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources.



- User - An individual who has a profile in Azure Active Directory. You can also assign roles to users in other tenants. For information about users in other organizations, see [Azure Active Directory B2B](#).
- Group - A set of users created in Azure Active Directory. When you assign a role to a group, all users within that group have that role.
- Service principal - A security identity used by applications or services to access specific Azure resources. You can think of it as a *user identity* (username and password or certificate) for an application.
- Managed identity - An identity in Azure Active Directory that is automatically managed by Azure. You typically use [managed identities](#) when developing cloud applications to manage the credentials for authenticating to Azure services.

Role definition

A *role definition* is a collection of permissions. It's sometimes just called a *role*. A role definition lists the operations that can be performed, such as read, write, and delete. Roles can be high-level, like owner, or specific, like virtual machine reader.



Azure includes several [built-in roles](#) that you can use. The following lists four fundamental built-in roles. The first three apply to all resource types.

- **Owner** - Has full access to all resources including the right to delegate access to others.
- **Contributor** - Can create and manage all types of Azure resources but can't grant access to others.
- **Reader** - Can view existing Azure resources.
- **User Access Administrator** - Lets you manage user access to Azure resources.

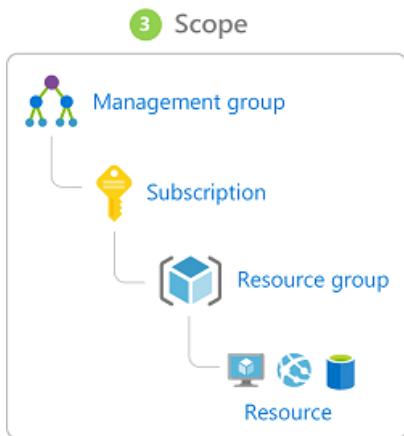
The rest of the built-in roles allow management of specific Azure resources. For example, the [Virtual Machine Contributor](#) role allows a user to create and manage virtual machines. If the built-in roles don't meet the specific needs of your organization, you can create your own [custom roles for Azure resources](#).

Azure has data operations that enable you to grant access to data within an object. For example, if a user has read data access to a storage account, then they can read the blobs or messages within that storage account. For more information, see [Understand role definitions for Azure resources](#).

Scope

Scope is the set of resources that the access applies to. When you assign a role, you can further limit the actions allowed by defining a scope. This is helpful if you want to make someone a [Website Contributor](#), but only for one resource group.

In Azure, you can specify a scope at multiple levels: [management group](#), subscription, resource group, or resource. Scopes are structured in a parent-child relationship.



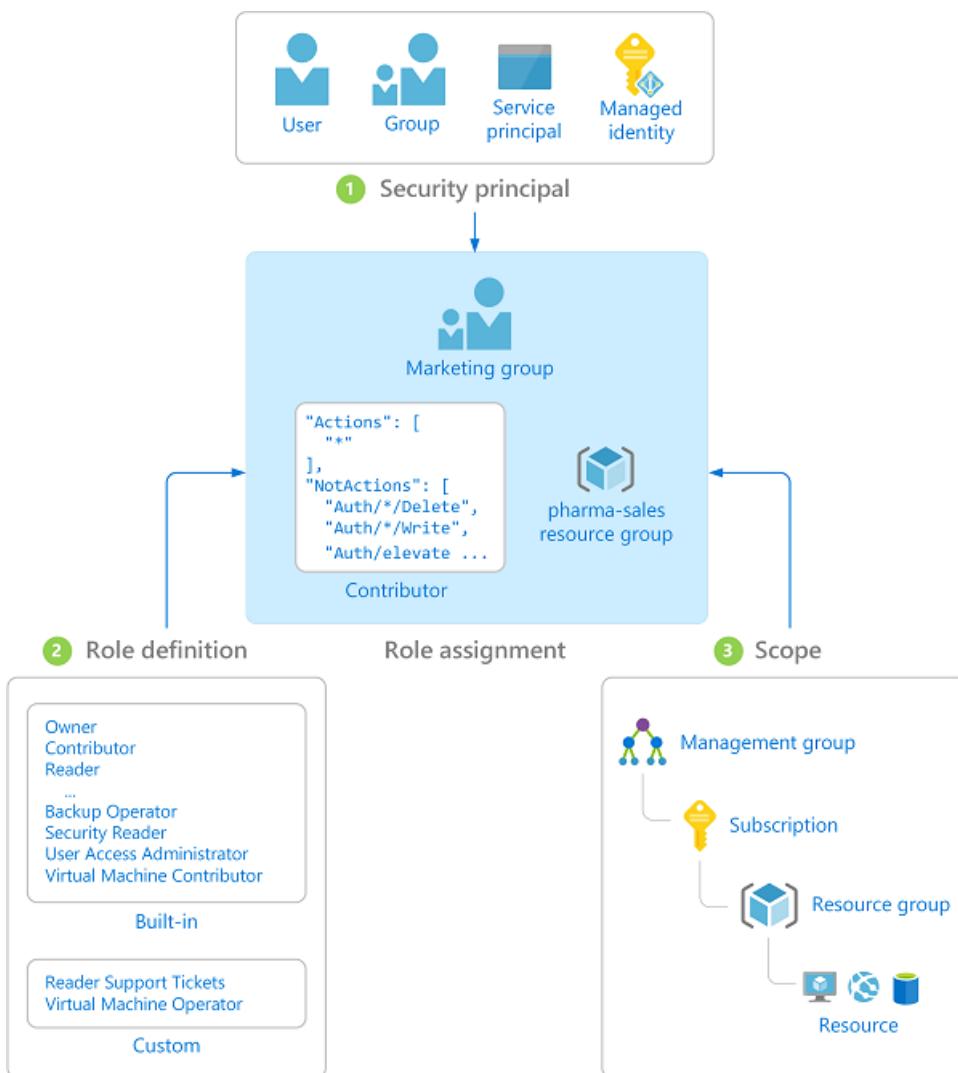
When you grant access at a parent scope, those permissions are inherited to the child scopes. For example:

- If you assign the [Owner](#) role to a user at the management group scope, that user can manage everything in all subscriptions in the management group.
- If you assign the [Reader](#) role to a group at the subscription scope, the members of that group can view every resource group and resource in the subscription.
- If you assign the [Contributor](#) role to an application at the resource group scope, it can manage resources of all types in that resource group, but not other resource groups in the subscription.

Role assignments

A *role assignment* is the process of attaching a role definition to a user, group, service principal, or managed identity at a particular scope for the purpose of granting access. Access is granted by creating a role assignment, and access is revoked by removing a role assignment.

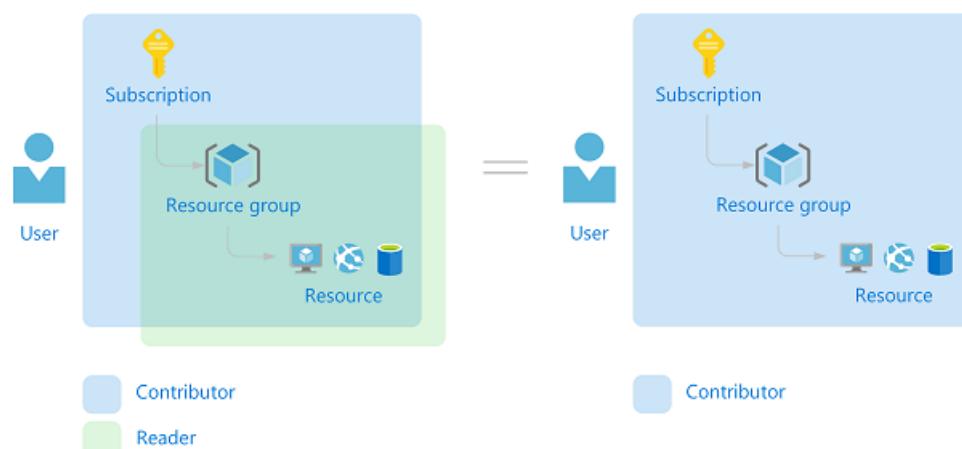
The following diagram shows an example of a role assignment. In this example, the Marketing group has been assigned the [Contributor](#) role for the pharma-sales resource group. This means that users in the Marketing group can create or manage any Azure resource in the pharma-sales resource group. Marketing users do not have access to resources outside the pharma-sales resource group, unless they are part of another role assignment.



You can create role assignments using the Azure portal, Azure CLI, Azure PowerShell, Azure SDKs, or REST APIs. You can have up to 2000 role assignments in each subscription. To create and remove role assignments, you must have `Microsoft.Authorization/roleAssignments/*` permission. This permission is granted through the [Owner](#) or [User Access Administrator](#) roles.

Multiple role assignments

So what happens if you have multiple overlapping role assignments? RBAC is an additive model, so your effective permissions are the addition of your role assignments. Consider the following example where a user is granted the Contributor role at the subscription scope and the Reader role on a resource group. The addition of the Contributor permissions and the Reader permissions is effectively the Contributor role for the resource group. Therefore, in this case, the Reader role assignment has no impact.



Deny assignments

Previously, RBAC was an allow-only model with no deny, but now RBAC supports deny assignments in a limited way. Similar to a role assignment, a *deny assignment* attaches a set of deny actions to a user, group, service principal, or managed identity at a particular scope for the purpose of denying access. A role assignment defines a set of actions that are *allowed*, while a deny assignment defines a set of actions that are *not allowed*. In other words, deny assignments block users from performing specified actions even if a role assignment grants them access. Deny assignments take precedence over role assignments. For more information, see [Understand deny assignments for Azure resources](#).

How RBAC determines if a user has access to a resource

The following are the high-level steps that RBAC uses to determine if you have access to a resource on the management plane. This is helpful to understand if you are trying to troubleshoot an access issue.

1. A user (or service principal) acquires a token for Azure Resource Manager.

The token includes the user's group memberships (including transitive group memberships).

2. The user makes a REST API call to Azure Resource Manager with the token attached.
3. Azure Resource Manager retrieves all the role assignments and deny assignments that apply to the resource upon which the action is being taken.
4. Azure Resource Manager narrows the role assignments that apply to this user or their group and determines what roles the user has for this resource.
5. Azure Resource Manager determines if the action in the API call is included in the roles the user has for this resource.
6. If the user doesn't have a role with the action at the requested scope, access is not granted. Otherwise, Azure Resource Manager checks if a deny assignment applies.
7. If a deny assignment applies, access is blocked. Otherwise access is granted.

License requirements

Using this feature is free and included in your Azure subscription.

Next steps

- [Quickstart: View the access a user has to Azure resources using the Azure portal](#)
- [Manage access to Azure resources using RBAC and the Azure portal](#)
- [Understand the different roles in Azure](#)
- [Enterprise Cloud Adoption: Resource access management in Azure](#)

Apply policies to Linux VMs with Azure Resource Manager

5/21/2019 • 2 minutes to read • [Edit Online](#)

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. In this article, we describe how you can use Azure Resource Manager policies to define the desired behavior for your organization's Virtual Machines.

For an introduction to policies, see [What is Azure Policy?](#).

Permitted Virtual Machines

To ensure that virtual machines for your organization are compatible with an application, you can restrict the permitted operating systems. In the following policy example, you allow only Ubuntu 14.04.2-LTS Virtual Machines to be created.

```
{
  "if": {
    "allOf": [
      {
        "field": "type",
        "in": [
          "Microsoft.Compute/disks",
          "Microsoft.Compute/virtualMachines",
          "Microsoft.Compute/VirtualMachineScaleSets"
        ]
      },
      {
        "not": {
          "allOf": [
            {
              "field": "Microsoft.Compute/imagePublisher",
              "in": [
                "Canonical"
              ]
            },
            {
              "field": "Microsoft.Compute/imageOffer",
              "in": [
                "UbuntuServer"
              ]
            },
            {
              "field": "Microsoft.Compute/imageSku",
              "in": [
                "14.04.2-LTS"
              ]
            },
            {
              "field": "Microsoft.Compute/imageVersion",
              "in": [
                "latest"
              ]
            }
          ]
        }
      }
    ],
    "then": {
      "effect": "deny"
    }
  }
}
```

Use a wild card to modify the preceding policy to allow any Ubuntu LTS image:

```
{
  "field": "Microsoft.Compute/virtualMachines/imageSku",
  "like": "*LTS"
}
```

For information about policy fields, see [Policy aliases](#).

Managed disks

To require the use of managed disks, use the following policy:

```
{
  "if": {
    "anyOf": [
      {
        "allOf": [
          {
            "field": "type",
            "equals": "Microsoft.Compute/virtualMachines"
          },
          {
            "field": "Microsoft.Compute/virtualMachines/osDisk.uri",
            "exists": true
          }
        ]
      },
      {
        "allOf": [
          {
            "field": "type",
            "equals": "Microsoft.Compute/VirtualMachineScaleSets"
          },
          {
            "anyOf": [
              {
                "field": "Microsoft.Compute/VirtualMachineScaleSets/osDisk.vhdContainers",
                "exists": true
              },
              {
                "field": "Microsoft.Compute/VirtualMachineScaleSets/osdisk.imageUrl",
                "exists": true
              }
            ]
          }
        ]
      }
    ],
    "then": {
      "effect": "deny"
    }
  }
}
```

Images for Virtual Machines

For security reasons, you can require that only approved custom images are deployed in your environment. You can specify either the resource group that contains the approved images, or the specific approved images.

The following example requires images from an approved resource group:

```
{
  "if": {
    "allOf": [
      {
        "field": "type",
        "in": [
          "Microsoft.Compute/virtualMachines",
          "Microsoft.Compute/VirtualMachineScaleSets"
        ]
      },
      {
        "not": {
          "field": "Microsoft.Compute/imageId",
          "contains": "resourceGroups/CustomImage"
        }
      }
    ],
    "then": {
      "effect": "deny"
    }
  }
}
```

The following example specifies the approved image IDs:

```
{
  "field": "Microsoft.Compute/imageId",
  "in": ["{imageId1}","{imageId2}"]
}
```

Virtual Machine extensions

You may want to forbid usage of certain types of extensions. For example, an extension may not be compatible with certain custom virtual machine images. The following example shows how to block a specific extension. It uses publisher and type to determine which extension to block.

```
{
  "if": {
    "allOf": [
      {
        "field": "type",
        "equals": "Microsoft.Compute/virtualMachines/extensions"
      },
      {
        "field": "Microsoft.Compute/virtualMachines/extensions/publisher",
        "equals": "Microsoft.Compute"
      },
      {
        "field": "Microsoft.Compute/virtualMachines/extensions/type",
        "equals": "{extension-type}"
      }
    ],
    "then": {
      "effect": "deny"
    }
  }
}
```

Next steps

- After defining a policy rule (as shown in the preceding examples), you need to create the policy definition and assign it to a scope. The scope can be a subscription, resource group, or resource. To assign policies, see [Use Azure portal to assign and manage resource policies](#), [Use PowerShell to assign policies](#), or [Use Azure CLI to assign policies](#).
- For an introduction to resource policies, see [What is Azure Policy?](#).
- For guidance on how enterprises can use Resource Manager to effectively manage subscriptions, see [Azure enterprise scaffold - prescriptive subscription governance](#).

How to set up Key Vault for virtual machines with the Azure CLI

5/21/2019 • 2 minutes to read • [Edit Online](#)

In the Azure Resource Manager stack, secrets/certificates are modeled as resources that are provided by Key Vault. To learn more about Azure Key Vault, see [What is Azure Key Vault?](#) In order for Key Vault to be used with Azure Resource Manager VMs, the *EnabledForDeployment* property on Key Vault must be set to true. This article shows you how to set up Key Vault for use with Azure virtual machines (VMs) using the Azure CLI.

To perform these steps, you need the latest [Azure CLI](#) installed and logged in to an Azure account using [az login](#).

Create a Key Vault

Create a key vault and assign the deployment policy with [az keyvault create](#). The following example creates a key vault named `myKeyVault` in the `myResourceGroup` resource group:

```
az keyvault create -l westus -n myKeyVault -g myResourceGroup --enabled-for-deployment true
```

Update a Key Vault for use with VMs

Set the deployment policy on an existing key vault with [az keyvault update](#). The following updates the key vault named `myKeyVault` in the `myResourceGroup` resource group:

```
az keyvault update -n myKeyVault -g myResourceGroup --set properties.enabledForDeployment=true
```

Use templates to set up Key Vault

When you use a template, you need to set the `enabledForDeployment` property to `true` for the Key Vault resource as follows:

```
{
  "type": "Microsoft.KeyVault/vaults",
  "name": "ContosoKeyVault",
  "apiVersion": "2015-06-01",
  "location": "<location-of-key-vault>",
  "properties": {
    "enabledForDeployment": "true",
    ....
    ....
  }
}
```

Next steps

For other options that you can configure when you create a Key Vault by using templates, see [Create a key vault](#).

Quick steps: Create and use an SSH public-private key pair for Linux VMs in Azure

1/30/2019 • 3 minutes to read • [Edit Online](#)

With a secure shell (SSH) key pair, you can create virtual machines (VMs) in Azure that use SSH keys for authentication, eliminating the need for passwords to sign in. This article shows you how to quickly generate and use an SSH public-private key file pair for Linux VMs. You can complete these steps with the Azure Cloud Shell, a macOS or Linux host, the Windows Subsystem for Linux, and other tools that support OpenSSH.

NOTE

VMs created using SSH keys are by default configured with passwords disabled, which greatly increases the difficulty of brute-force guessing attacks.

For more background and examples, see [Detailed steps to create SSH key pairs](#).

For additional ways to generate and use SSH keys on a Windows computer, see [How to use SSH keys with Windows on Azure](#).

Supported SSH key formats

Azure currently supports SSH protocol 2 (SSH-2) RSA public-private key pairs with a minimum length of 2048 bits. Other key formats such as ED25519 and ECDSA are not supported.

Create an SSH key pair

Use the `ssh-keygen` command to generate SSH public and private key files. By default, these files are created in the `~/.ssh` directory. You can specify a different location, and an optional password (*passphrase*) to access the private key file. If an SSH key pair with the same name exists in the given location, those files are overwritten.

The following command creates an SSH key pair using RSA encryption and a bit length of 2048:

```
ssh-keygen -t rsa -b 2048
```

If you use the [Azure CLI](#) to create your VM with the `az vm create` command, you can optionally generate SSH public and private key files using the `--generate-ssh-keys` option. The key files are stored in the `~/.ssh` directory unless specified otherwise with the `--ssh-dest-key-path` option. The `--generate-ssh-keys` option will not overwrite existing key files, instead returning an error. In the following command, replace `VMname` and `RGname` with your own values:

```
az vm create --name VMname --resource-group RGname --generate-ssh-keys
```

Provide an SSH public key when deploying a VM

To create a Linux VM that uses SSH keys for authentication, specify your SSH public key when creating the VM using the Azure portal, Azure CLI, Azure Resource Manager templates, or other methods:

- [Create a Linux virtual machine with the Azure portal](#)

- [Create a Linux virtual machine with the Azure CLI](#)
- [Create a Linux VM using an Azure template](#)

If you're not familiar with the format of an SSH public key, you can display your public key with the following `cat` command, replacing `~/.ssh/id_rsa.pub` with the path and filename of your own public key file if needed:

```
cat ~/.ssh/id_rsa.pub
```

A typical public key value looks like this example:

```
ssh-rsa  
AAAAB3NzaC1yc2EAAQABAAQACQC1/KanayNr+Q7ogR5mKnGpKWRBQU7F3Jjh...  
IaNOWd6zM8hB6UrcRT1Tpwk/SuGMw1Vb40x1EFphBkVEUgB0l0oANIEr...  
XE1iY/sFZLJUJzcRUI0MOfHXAuCjg/qyqqbIuTDFyfg8k0JTtyGFEMQhbX...  
g5xo/ra3mq2imwtOKJpfdtFoMiKhJmSNHSk7vFTeYgg0v2cQ2+vL381cIFX40h+QCzvNF/A...  
7reCBVV1hyxc2L1YUkrq4DHzkxNY5c90GSHXSle9Ys03F1J5ip18f6gPq4xFmo6dV...  
DXYYB182ZCGQzXfz1PDC29cWvgDZEXNHuYr0LmJ...  
DOGhx4VodXAjT09omhQJpE6w1ZbRW...  
username@domainname
```

If you copy and paste the contents of the public key file to use in the Azure portal or a Resource Manager template, make sure you don't copy any trailing whitespace. To copy a public key in macOS, you can pipe the public key file to **pbcopy**. Similarly in Linux, you can pipe the public key file to programs such as **xclip**.

The public key that you place on your Linux VM in Azure is by default stored in `~/.ssh/id_rsa.pub`, unless you specified a different location when you created the key pair. To use the [Azure CLI 2.0](#) to create your VM with an existing public key, specify the value and optionally the location of this public key using the [az vm create](#) command with the `--ssh-key-value` option. In the following command, replace *VMname*, *RGname*, and *keyFile* with your own values:

```
az vm create --name VMname --resource-group RGname --ssh-key-value @keyFile
```

SSH into your VM

With the public key deployed on your Azure VM, and the private key on your local system, SSH into your VM using the IP address or DNS name of your VM. In the following command, replace *azureuser* and *myvm.westus.cloudapp.azure.com* with the administrator user name and the fully qualified domain name (or IP address):

```
ssh azureuser@myvm.westus.cloudapp.azure.com
```

If you specified a passphrase when you created your key pair, enter that passphrase when prompted during the login process. The VM is added to your `~/.ssh/known_hosts` file, and you won't be asked to connect again until either the public key on your Azure VM changes or the server name is removed from `~/.ssh/known_hosts`.

If the VM is using the just-in-time access policy, you need to request access before you can connect to the VM. For more information about the just-in-time policy, see [Manage virtual machine access using the just in time policy](#).

Next steps

- For more information on working with SSH key pairs, see [Detailed steps to create and manage SSH key pairs](#).

- If you have difficulties with SSH connections to Azure VMs, see [Troubleshoot SSH connections to an Azure Linux VM](#).

How to use SSH keys with Windows on Azure

3/14/2019 • 6 minutes to read • [Edit Online](#)

This article describes ways to generate and use *secure shell* (SSH) keys on a Windows computer to create and connect to a Linux virtual machine (VM) in Azure. To use SSH keys from a Linux or macOS client, see the [quick](#) or [detailed](#) guidance.

Overview of SSH and keys

SSH is an encrypted connection protocol that allows secure sign-ins over unsecured connections. SSH is the default connection protocol for Linux VMs hosted in Azure. Although SSH itself provides an encrypted connection, using passwords with SSH connections still leaves the VM vulnerable to brute-force attacks or guessing of passwords. A more secure and preferred method of connecting to a VM using SSH is by using a public-private key pair, also known as *SSH keys*.

- The *public key* is placed on your Linux VM, or any other service that you wish to use with public-key cryptography.
- The *private key* remains on your local system. Protect this private key. Do not share it.

When you use an SSH client to connect to your Linux VM (which has the public key), the remote VM tests the client to make sure it possesses the private key. If the client has the private key, it's granted access to the VM.

Depending on your organization's security policies, you can reuse a single public-private key pair to access multiple Azure VMs and services. You do not need a separate pair of keys for each VM or service you wish to access.

Your public key can be shared with anyone, but only you (or your local security infrastructure) should possess your private key.

Supported SSH key formats

Azure currently supports SSH protocol 2 (SSH-2) RSA public-private key pairs with a minimum length of 2048 bits. Other key formats such as ED25519 and ECDSA are not supported.

Windows packages and SSH clients

You connect to and manage Linux VMs in Azure using an *SSH client*. Computers running Linux or macOS usually have a suite of SSH commands to generate and manage SSH keys and to make SSH connections.

Windows computers do not always have comparable SSH commands installed. Recent versions of Windows 10 provide [OpenSSH client commands](#) to create and manage SSH keys and make SSH connections from a command prompt. Recent Windows 10 versions also include the [Windows Subsystem for Linux](#) to run and access utilities such as an SSH client natively within a Bash shell.

Other common Windows SSH clients you can install locally are included in the following packages:

- [PuTTY](#)
- [Git For Windows](#)
- [MobaXterm](#)
- [Cygwin](#)

You can also use the SSH utilities available in Bash in the [Azure Cloud Shell](#).

- Access Cloud Shell in your web browser at <https://shell.azure.com> or in the [Azure portal](#).
- Access Cloud Shell as a terminal from within Visual Studio Code by installing the [Azure Account extension](#).

Create an SSH key pair

The following sections describe two options to create an SSH key pair on Windows. You can use a shell command (`ssh-keygen`) or a GUI tool (PuTTYgen).

Create SSH keys with ssh-keygen

If you run a command shell on Windows that supports SSH client tools (or you use Azure Cloud Shell), create an SSH key pair using the `ssh-keygen` command. Type the following command, and answer the prompts. If an SSH key pair exists in the chosen location, those files are overwritten.

```
ssh-keygen -t rsa -b 2048
```

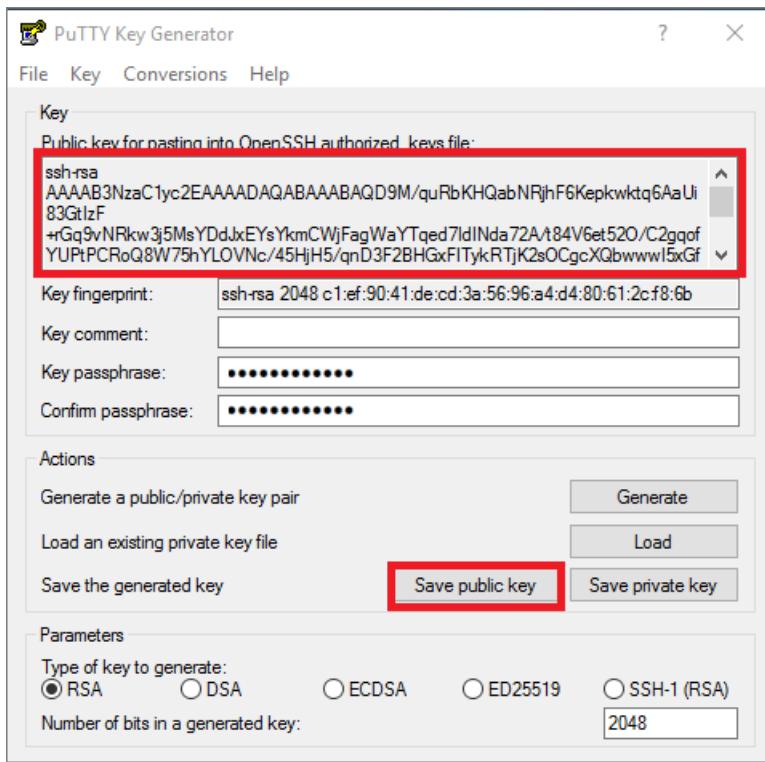
For more background and information, see the [quick](#) or [detailed](#) steps to create SSH keys using `ssh-keygen`.

Create SSH keys with PuTTYgen

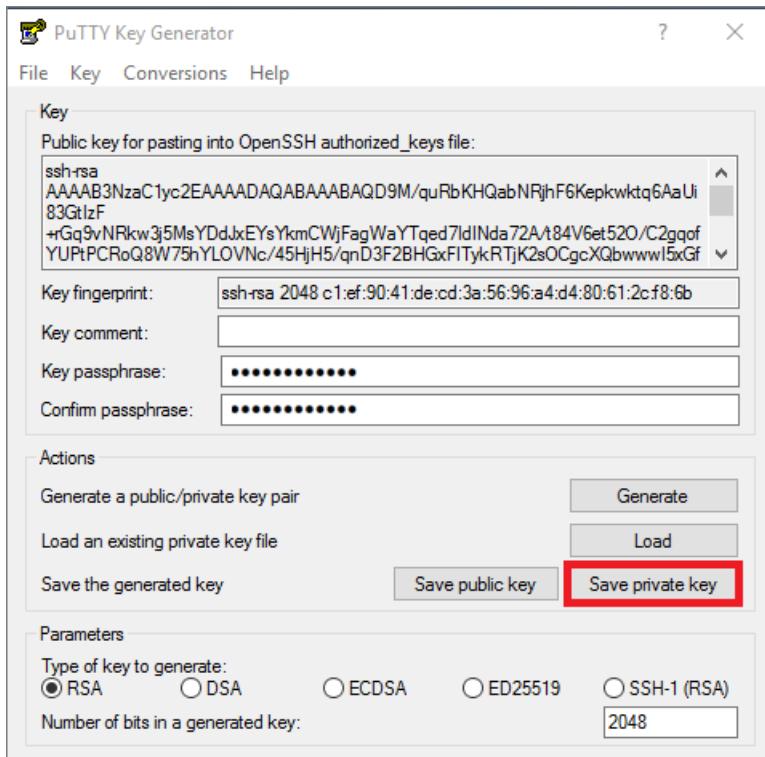
If you prefer to use a GUI-based tool to create SSH keys, you can use the PuTTYgen key generator, included with the [PuTTY download package](#).

To create an SSH RSA key pair with PuTTYgen:

1. Start PuTTYgen.
2. Click **Generate**. By default PuTTYgen generates a 2048-bit SSH-2 RSA key.
3. Move the mouse around in the blank area to provide randomness for the key.
4. After the public key is generated, optionally enter and confirm a passphrase. You will be prompted for the passphrase when you authenticate to the VM with your private SSH key. Without a passphrase, if someone obtains your private key, they can sign in to any VM or service that uses that key. We recommend you create a passphrase. However, if you forget the passphrase, there is no way to recover it.
5. The public key is displayed at the top of the window. You can copy this entire public key and then paste it into the Azure portal or an Azure Resource Manager template when you create a Linux VM. You can also select **Save public key** to save a copy to your computer:



6. Optionally, to save the private key in PuTTY private key format (.ppk file), select **Save private key**. You will need the .ppk file later to use PuTTY to make an SSH connection to the VM.

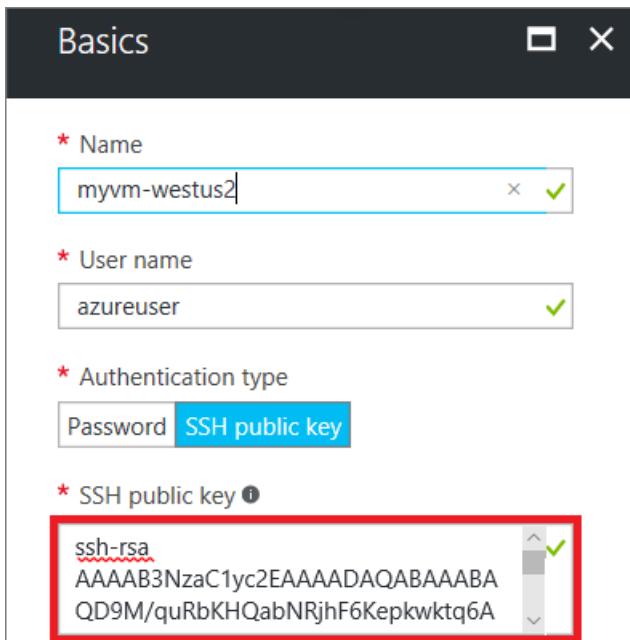


If you want to save the private key in the OpenSSH format, the private key format used by many SSH clients, select **Conversions > Export OpenSSH key**.

Provide an SSH public key when deploying a VM

To create a Linux VM that uses SSH keys for authentication, provide your SSH public key when creating the VM using the Azure portal or other methods.

The following example shows how you would copy and paste this public key into the Azure portal when you create a Linux VM. The public key is typically then stored in the `~/.ssh/authorized_key` directory on your new VM.



Connect to your VM

One way to make an SSH connection to your Linux VM from Windows is to use an SSH client. This is the preferred method if you have an SSH client installed on your Windows system, or if you use the SSH tools in Bash in Azure Cloud Shell. If you prefer a GUI-based tool, you can connect with PuTTY.

Use an SSH client

With the public key deployed on your Azure VM, and the private key on your local system, SSH to your VM using the IP address or DNS name of your VM. Replace `azureuser` and `myvm.westus.cloudapp.azure.com` in the following command with the administrator user name and the fully qualified domain name (or IP address):

```
ssh azureuser@myvm.westus.cloudapp.azure.com
```

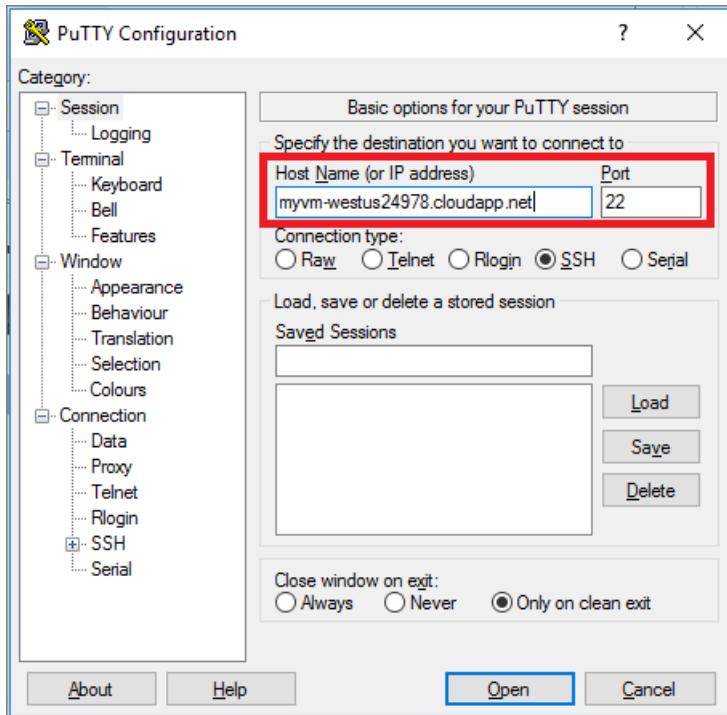
If you configured a passphrase when you created your key pair, enter the passphrase when prompted during the sign-in process.

If the VM is using the just-in-time access policy, you need to request access before you can connect to the VM. For more information about the just-in-time policy, see [Manage virtual machine access using the just in time policy](#).

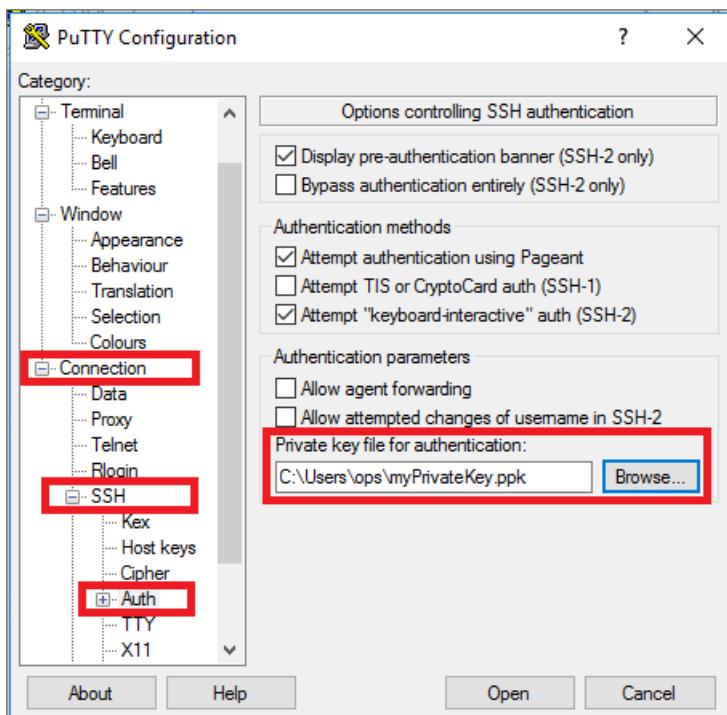
Connect with PuTTY

If you installed the [PuTTY download package](#) and previously generated a PuTTY private key (.ppk) file, you can connect to a Linux VM with PuTTY.

1. Start PuTTy.
2. Fill in the host name or IP address of your VM from the Azure portal:



3. Select the **Connection > SSH > Auth** category. Browse to and select your PuTTY private key (.ppk file):



4. Click **Open** to connect to your VM.

Next steps

- For detailed steps, options, and advanced examples of working with SSH keys, see [Detailed steps to create SSH key pairs](#).
- You can also use PowerShell in Azure Cloud Shell to generate SSH keys and make SSH connections to Linux VMs. See the [PowerShell quickstart](#).
- If you have difficulty using SSH to connect to your Linux VMs, see [Troubleshoot SSH connections to an Azure Linux VM](#).

Detailed steps: Create and manage SSH keys for authentication to a Linux VM in Azure

2/4/2019 • 10 minutes to read • [Edit Online](#)

With a secure shell (SSH) key pair, you can create a Linux virtual machine on Azure that defaults to using SSH keys for authentication, eliminating the need for passwords to sign in. VMs created with the Azure portal, Azure CLI, Resource Manager templates, or other tools can include your SSH public key as part of the deployment, which sets up SSH key authentication for SSH connections.

This article provides detailed background and steps to create and manage an SSH RSA public-private key file pair for SSH client connections. If you want quick commands, see [How to create an SSH public-private key pair for Linux VMs in Azure](#).

For additional ways to generate and use SSH keys on a Windows computer, see [How to use SSH keys with Windows on Azure](#).

Overview of SSH and keys

SSH is an encrypted connection protocol that allows secure sign-ins over unsecured connections. SSH is the default connection protocol for Linux VMs hosted in Azure. Although SSH itself provides an encrypted connection, using passwords with SSH connections still leaves the VM vulnerable to brute-force attacks or guessing of passwords. A more secure and preferred method of connecting to a VM using SSH is by using a public-private key pair, also known as *SSH keys*.

- The *public key* is placed on your Linux VM, or any other service that you wish to use with public-key cryptography.
- The *private key* remains on your local system. Protect this private key. Do not share it.

When you use an SSH client to connect to your Linux VM (which has the public key), the remote VM tests the client to make sure it possesses the private key. If the client has the private key, it's granted access to the VM.

Depending on your organization's security policies, you can reuse a single public-private key pair to access multiple Azure VMs and services. You do not need a separate pair of keys for each VM or service you wish to access.

Your public key can be shared with anyone, but only you (or your local security infrastructure) should possess your private key.

Private key passphrase

The SSH private key should have a very secure passphrase to safeguard it. This passphrase is just to access the private SSH key file and *is not* the user account password. When you add a passphrase to your SSH key, it encrypts the private key using 128-bit AES, so that the private key is useless without the passphrase to decrypt it. If an attacker stole your private key and that key did not have a passphrase, they would be able to use that private key to sign in to any servers that have the corresponding public key. If a private key is protected by a passphrase, it cannot be used by that attacker, providing an additional layer of security for your infrastructure on Azure.

Supported SSH key formats

Azure currently supports SSH protocol 2 (SSH-2) RSA public-private key pairs with a minimum length of 2048 bits. Other key formats such as ED25519 and ECDSA are not supported.

SSH keys use and benefits

When you create an Azure VM by specifying the public key, Azure copies the public key (in the `.pub` format) to the `~/.ssh/authorized_keys` folder on the VM. SSH keys in `~/.ssh/authorized_keys` are used to challenge the client to match the corresponding private key on an SSH connection. In an Azure Linux VM that uses SSH keys for authentication, Azure configures the SSHD server to not allow password sign-in, only SSH keys. Therefore, by creating an Azure Linux VM with SSH keys, you can help secure the VM deployment and save yourself the typical post-deployment configuration step of disabling passwords in the `sshd_config` file.

If you do not wish to use SSH keys, you can set up your Linux VM to use password authentication. If your VM is not exposed to the Internet, using passwords may be sufficient. However, you still need to manage your passwords for each Linux VM and maintain healthy password policies and practices, such as minimum password length and regular updates. Using SSH keys reduces the complexity of managing individual credentials across multiple VMs.

Generate keys with ssh-keygen

To create the keys, a preferred command is `ssh-keygen`, which is available with OpenSSH utilities in the Azure Cloud Shell, a macOS or Linux host, the [Windows Subsystem for Linux](#), and other tools. `ssh-keygen` asks a series of questions and then writes a private key and a matching public key.

SSH keys are by default kept in the `~/.ssh` directory. If you do not have a `~/.ssh` directory, the `ssh-keygen` command creates it for you with the correct permissions.

Basic example

The following `ssh-keygen` command generates 2048-bit SSH RSA public and private key files by default in the `~/.ssh` directory. If an SSH key pair exists in the current location, those files are overwritten.

```
ssh-keygen -t rsa -b 2048
```

Detailed example

The following example shows additional command options to create an SSH RSA key pair. If an SSH key pair exists in the current location, those files are overwritten.

```
ssh-keygen \
-t rsa \
-b 4096 \
-C "azureuser@myserver" \
-f ~/.ssh/mykeys/myprivatekey \
-N mypassphrase
```

Command explained

`ssh-keygen` = the program used to create the keys

`-t rsa` = type of key to create, in this case in the RSA format

`-b 4096` = the number of bits in the key, in this case 4096

`-C "azureuser@myserver"` = a comment appended to the end of the public key file to easily identify it. Normally an email address is used as the comment, but use whatever works best for your infrastructure.

`-f ~/.ssh/mykeys/myprivatekey` = the filename of the private key file, if you choose not to use the default name. A corresponding public key file appended with `.pub` is generated in the same directory. The directory must exist.

`-N mypassphrase` = an additional passphrase used to access the private key file.

Example of ssh-keygen

```
ssh-keygen -t rsa -b 2048 -C "azureuser@myserver"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/azureuser/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/azureuser/.ssh/id_rsa.
Your public key has been saved in /home/azureuser/.ssh/id_rsa.pub.
The key fingerprint is:
14:a3:cb:3e:78:ad:25:cc:55:e9:0c:08:e5:d1:a9:08 azureuser@myserver
The keys randomart image is:
---[ RSA 2048]---+
|   o o. . |
| E. = .o   |
| ..o...    |
| . o....   |
| o S =    |
| . + O    |
| + = =    |
| o +      |
| .         |
+-----+
```

Saved key files

Enter file in which to save the key (/home/azureuser/.ssh/id_rsa): `~/.ssh/id_rsa`

The key pair name for this article. Having a key pair named `id_rsa` is the default; some tools might expect the `id_rsa` private key file name, so having one is a good idea. The directory `~/ssh/` is the default location for SSH key pairs and the SSH config file. If not specified with a full path, `ssh-keygen` creates the keys in the current working directory, not the default `~/ssh`.

List of the `~/ssh` directory

```
ls -al ~/.ssh
-rw----- 1 azureuser staff 1675 Aug 25 18:04 id_rsa
-rw-r--r-- 1 azureuser staff 410 Aug 25 18:04 id_rsa.pub
```

Key passphrase

Enter passphrase (empty for no passphrase):

It is *strongly* recommended to add a passphrase to your private key. Without a passphrase to protect the key file, anyone with the file can use it to sign in to any server that has the corresponding public key. Adding a passphrase offers more protection in case someone is able to gain access to your private key file, giving you time to change the keys.

Generate keys automatically during deployment

If you use the [Azure CLI](#) to create your VM, you can optionally generate SSH public and private key files by running the `az vm create` command with the `--generate-ssh-keys` option. The keys are stored in the `~/ssh` directory. Note that this command option does not overwrite keys if they already exist in that location.

Provide SSH public key when deploying a VM

To create a Linux VM that uses SSH keys for authentication, provide your SSH public key when creating the VM using the Azure portal, CLI, Resource Manager templates, or other methods. When using the portal, you enter the public key itself. If you use the [Azure CLI](#) to create your VM with an existing public key, specify the value or

location of this public key by running the `az vm create` command with the `--ssh-key-value` option.

If you're not familiar with the format of an SSH public key, you can see your public key by running `cat` as follows, replacing `~/.ssh/id_rsa.pub` with your own public key file location:

```
cat ~/.ssh/id_rsa.pub
```

Output is similar to the following (here redacted):

```
ssh-rsa  
XXXXXXXXXXc2EAAAADAXABAAABAXC5Am7+fGZ+5zXBGgXS6GUvmsXCLGc7tX7/rViXk3+eShZzaXnt75gUmT1I2f75zFn2h1AIDGKwf4g12KW  
cZxy81TniUOTjUsV1wPymXUXxESL/UfJKfbdstBhT0dy5EG9rYWA0K43SJmwPhH28BpoLfXXXXXG+/ilsXXXXXKgRLiJ2W19MzXHp8zLxw7r  
9wx3HaV1P4XiFv9U4hGcp8RMI1MP1nNesFlOBpG4pV2bJRTXNxY416F8WZ3C4ku8Xx0o88mxaTpVZ3T1841a1tmNTZCcPkXuMrBjYSjbA8  
npoXAXNwiivyo3X2KMXXXXdXXXXXXXXXXXX/ azureuser@myserver
```

If you copy and paste the contents of the public key file into the Azure portal or a Resource Manager template, make sure you don't copy any additional whitespace or introduce additional line breaks. For example, if you use macOS, you can pipe the public key file (by default, `~/.ssh/id_rsa.pub`) to **pbcopy** to copy the contents (there are other Linux programs that do the same thing, such as `xclip`).

If you prefer to use a public key that is in a multiline format, you can generate an RFC4716 formatted key in a pem container from the public key you previously created.

To create a RFC4716 formatted key from an existing SSH public key:

```
ssh-keygen \  
-f ~/.ssh/id_rsa.pub \  
-e \  
-m RFC4716 > ~/.ssh/id_ssh2.pem
```

SSH to your VM with an SSH client

With the public key deployed on your Azure VM, and the private key on your local system, SSH to your VM using the IP address or DNS name of your VM. Replace *azureuser* and *myvm.westus.cloudapp.azure.com* in the following command with the administrator user name and the fully qualified domain name (or IP address):

```
ssh azureuser@myvm.westus.cloudapp.azure.com
```

If you provided a passphrase when you created your key pair, enter the passphrase when prompted during the sign-in process. (The server is added to your `~/.ssh/known_hosts` folder, and you won't be asked to connect again until the public key on your Azure VM changes or the server name is removed from `~/.ssh/known_hosts`.)

If the VM is using the just-in-time access policy, you need to request access before you can connect to the VM. For more information about the just-in-time policy, see [Manage virtual machine access using the just in time policy](#).

Use ssh-agent to store your private key passphrase

To avoid typing your private key file passphrase with every SSH sign-in, you can use `ssh-agent` to cache your private key file passphrase. If you are using a Mac, the macOS Keychain securely stores the private key passphrase when you invoke `ssh-agent`.

Verify and use `ssh-agent` and `ssh-add` to inform the SSH system about the key files so that you do not need to use the passphrase interactively.

```
eval "$(ssh-agent -s)"
```

Now add the private key to `ssh-agent` using the command `ssh-add`.

```
ssh-add ~/.ssh/id_rsa
```

The private key passphrase is now stored in `ssh-agent`.

Use `ssh-copy-id` to copy the key to an existing VM

If you have already created a VM, you can install the new SSH public key to your Linux VM with a command similar to the following:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub azureuser@myserver
```

Create and configure an SSH config file

You can create and configure an SSH config file (`~/.ssh/config`) to speed up log-ins and to optimize your SSH client behavior.

The following example shows a simple configuration that you can use to quickly sign in as a user to a specific VM using the default SSH private key.

Create the file

```
touch ~/.ssh/config
```

Edit the file to add the new SSH configuration

```
vim ~/.ssh/config
```

Example configuration

Add configuration settings appropriate for your host VM.

```
# Azure Keys
Host myvm
  Hostname 102.160.203.241
  User azureuser
# ./Azure Keys
```

You can add configurations for additional hosts to enable each to use its own dedicated key pair. See [SSH config file](#) for more advanced configuration options.

Now that you have an SSH key pair and a configured SSH config file, you are able to sign in to your Linux VM quickly and securely. When you run the following command, SSH locates and loads any settings from the `Host myvm` block in the SSH config file.

```
ssh myvm
```

The first time you sign in to a server using an SSH key, the command prompts you for the passphrase for that key file.

Next steps

Next up is to create Azure Linux VMs using the new SSH public key. Azure VMs that are created with an SSH public key as the sign-in are better secured than VMs created with the default sign-in method, passwords.

- [Create a Linux virtual machine with the Azure portal](#)
- [Create a Linux virtual machine with the Azure CLI](#)
- [Create a Linux VM using an Azure template](#)

2 minutes to read

Back up a virtual machine in Azure with the CLI

3/28/2019 • 5 minutes to read • [Edit Online](#)

The Azure CLI is used to create and manage Azure resources from the command line or in scripts. You can protect your data by taking backups at regular intervals. Azure Backup creates recovery points that can be stored in geo-redundant recovery vaults. This article details how to back up a virtual machine (VM) in Azure with the Azure CLI. You can also perform these steps with [Azure PowerShell](#) or in the [Azure portal](#).

This quick start enables backup on an existing Azure VM. If you need to create a VM, you can [create a VM with the Azure CLI](#).

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

To install and use the CLI locally, you must run Azure CLI version 2.0.18 or later. To find the CLI version, run `az --version`. If you need to install or upgrade, see [Install the Azure CLI](#).

Create a recovery services vault

A Recovery Services vault is a logical container that stores the backup data for each protected resource, such as Azure VMs. When the backup job for a protected resource runs, it creates a recovery point inside the Recovery Services vault. You can then use one of these recovery points to restore data to a given point in time.

Create a Recovery Services vault with `az backup vault create`. Specify the same resource group and location as the VM you wish to protect. If you used the [VM quickstart](#), then you created:

- a resource group named *myResourceGroup*,
- a VM named *myVM*,
- resources in the *eastus* location.

```
az backup vault create --resource-group myResourceGroup \
--name myRecoveryServicesVault \
--location eastus
```

By default, the Recovery Services vault is set for Geo-Redundant storage. Geo-Redundant storage ensures your backup data is replicated to a secondary Azure region that is hundreds of miles away from the primary region. If the storage redundancy setting needs to be modified, use [az backup vault backup-properties set](#) cmdlet.

```
az backup vault backup-properties set \
--name myRecoveryServicesVault \
--resource-group myResourceGroup \
--backup-storage-redundancy "LocallyRedundant/GeoRedundant"
```

Enable backup for an Azure VM

Create a protection policy to define: when a backup job runs, and how long the recovery points are stored. The default protection policy runs a backup job each day and retains recovery points for 30 days. You can use these default policy values to quickly protect your VM. To enable backup protection for a VM, use [az backup protection enable-for-vm](#). Specify the resource group and VM to protect, then the policy to use:

```
az backup protection enable-for-vm \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--vm myVM \
--policy-name DefaultPolicy
```

NOTE

If the VM is not in the same resource group as that of vault, then myResourceGroup refers to the resource group where vault was created. Instead of VM name, provide the VM ID as indicated below.

```
az backup protection enable-for-vm \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--vm $(az vm show -g VMResourceGroup -n MyVm --query id | tr -d '') \
--policy-name DefaultPolicy
```

Start a backup job

To start a backup now rather than wait for the default policy to run the job at the scheduled time, use [az backup protection backup-now](#). This first backup job creates a full recovery point. Each backup job after this initial backup creates incremental recovery points. Incremental recovery points are storage and time-efficient, as they only transfer changes made since the last backup.

The following parameters are used to back up the VM:

- `--container-name` is the name of your VM
- `--item-name` is the name of your VM
- `--retain-until` value should be set to the last available date, in UTC time format (**dd-mm-yyyy**), that you wish the recovery point to be available

The following example backs up the VM named *myVM* and sets the expiration of the recovery point to October 18,

2017:

```
az backup protection backup-now \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--container-name myVM \
--item-name myVM \
--retain-until 18-10-2017
```

Monitor the backup job

To monitor the status of backup jobs, use [az backup job list](#):

```
az backup job list \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--output table
```

The output is similar to the following example, which shows the backup job is *InProgress*:

Name	Operation	Status	Item Name	Start Time UTC	Duration
a0a8e5e6	Backup	InProgress	myvm	2017-09-19T03:09:21	0:00:48.718366
fe5d0414	ConfigureBackup	Completed	myvm	2017-09-19T03:03:57	0:00:31.191807

When the *Status* of the backup job reports *Completed*, your VM is protected with Recovery Services and has a full recovery point stored.

Clean up deployment

When no longer needed, you can disable protection on the VM, remove the restore points and Recovery Services vault, then delete the resource group and associated VM resources. If you used an existing VM, you can skip the final [az group delete](#) command to leave the resource group and VM in place.

If you want to try a Backup tutorial that explains how to restore data for your VM, go to [Next steps](#).

```
az backup protection disable \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--container-name myVM \
--item-name myVM \
--delete-backup-data true
az backup vault delete \
--resource-group myResourceGroup \
--name myRecoveryServicesVault \
az group delete --name myResourceGroup
```

Next steps

In this quick start, you created a Recovery Services vault, enabled protection on a VM, and created the initial recovery point. To learn more about Azure Backup and Recovery Services, continue to the tutorials.

[Back up multiple Azure VMs](#)

Use Azure portal to back up multiple virtual machines

3/15/2019 • 6 minutes to read • [Edit Online](#)

When you back up data in Azure, you store that data in an Azure resource called a Recovery Services vault. The Recovery Services vault resource is available from the Settings menu of most Azure services. The benefit of having the Recovery Services vault integrated into the Settings menu of most Azure services makes it very easy to back up data. However, individually working with each database or virtual machine in your business is tedious. What if you want to back up the data for all virtual machines in one department, or in one location? It is easy to back up multiple virtual machines by creating a backup policy and applying that policy to the desired virtual machines. This tutorial explains how to:

- Create a Recovery Services vault
- Define a backup policy
- Apply the backup policy to protect multiple virtual machines
- Trigger an on-demand backup job for the protected virtual machines

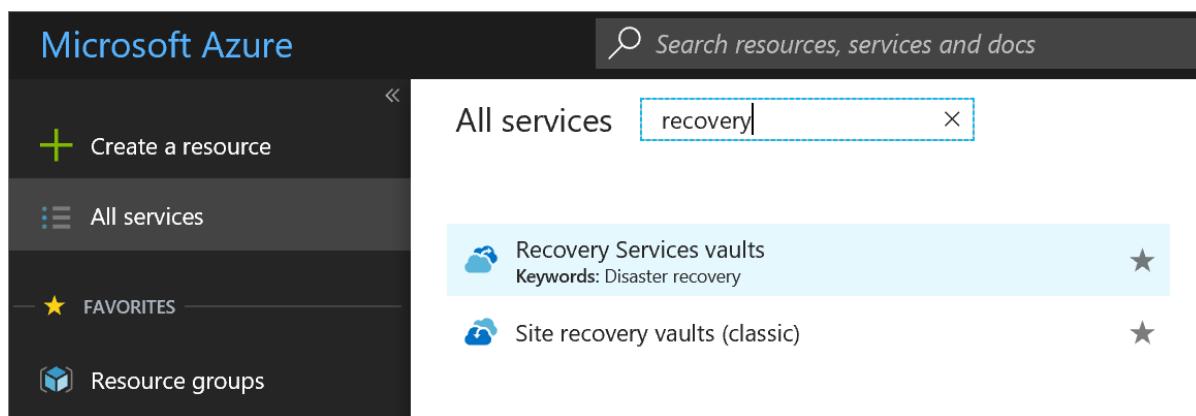
Log in to the Azure portal

Log in to the [Azure portal](#).

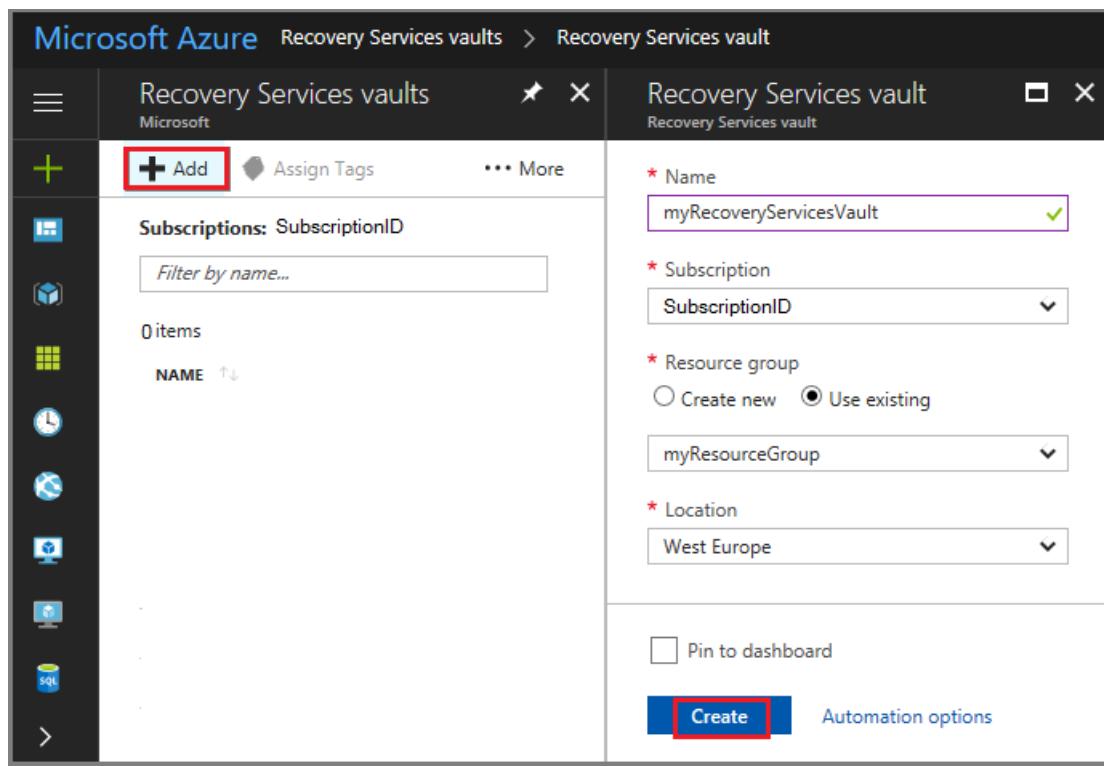
Create a Recovery Services vault

The Recovery Services vault contains the backup data, and the backup policy applied to the protected virtual machines. Backing up virtual machines is a local process. You cannot back up a virtual machine from one location to a Recovery Services vault in another location. So, for each Azure location that has virtual machines to be backed up, at least one Recovery Services vault must exist in that location.

1. On the left-hand menu, select **All services** and in the services list, type *Recovery Services*. As you type, the list of resources filters. When you see Recovery Services vaults in the list, select it to open the Recovery Services vaults menu.



2. In the **Recovery Services vaults** menu, click **Add** to open the Recovery Services vault menu.



3. In the Recovery Services vault menu,

- Type *myRecoveryServicesVault* in **Name**.
- The current subscription ID appears in **Subscription**. If you have additional subscriptions, you could choose another subscription for the new vault.
- For **Resource group** select **Use existing** and choose *myResourceGroup*. If *myResourceGroup* doesn't exist, select **Create new** and type *myResourceGroup*.
- From the **Location** drop-down menu, choose *West Europe*.
- Click **Create** to create your Recovery Services vault.

A Recovery Services vault must be in the same location as the virtual machines being protected. If you have virtual machines in multiple regions, create a Recovery Services vault in each region. This tutorial creates a Recovery Services vault in *West Europe* because that is where *myVM* (the virtual machine created with the quickstart) was created.

It can take several minutes for the Recovery Services vault to be created. Monitor the status notifications in the upper right-hand area of the portal. Once your vault is created, it appears in the list of Recovery Services vaults.

When you create a Recovery Services vault, by default the vault has geo-redundant storage. To provide data resiliency, geo-redundant storage replicates the data multiple times across two Azure regions.

Set backup policy to protect VMs

After creating the Recovery Services vault, the next step is to configure the vault for the type of data, and to set the backup policy. Backup policy is the schedule for how often and when recovery points are taken. Policy also includes the retention range for the recovery points. For this tutorial let's assume your business is a sports complex with a hotel, stadium, and restaurants and concessions, and you are protecting the data on the virtual machines. The following steps create a backup policy for the financial data.

1. From the list of Recovery Services vaults, select **myRecoveryServicesVault** to open its dashboard.

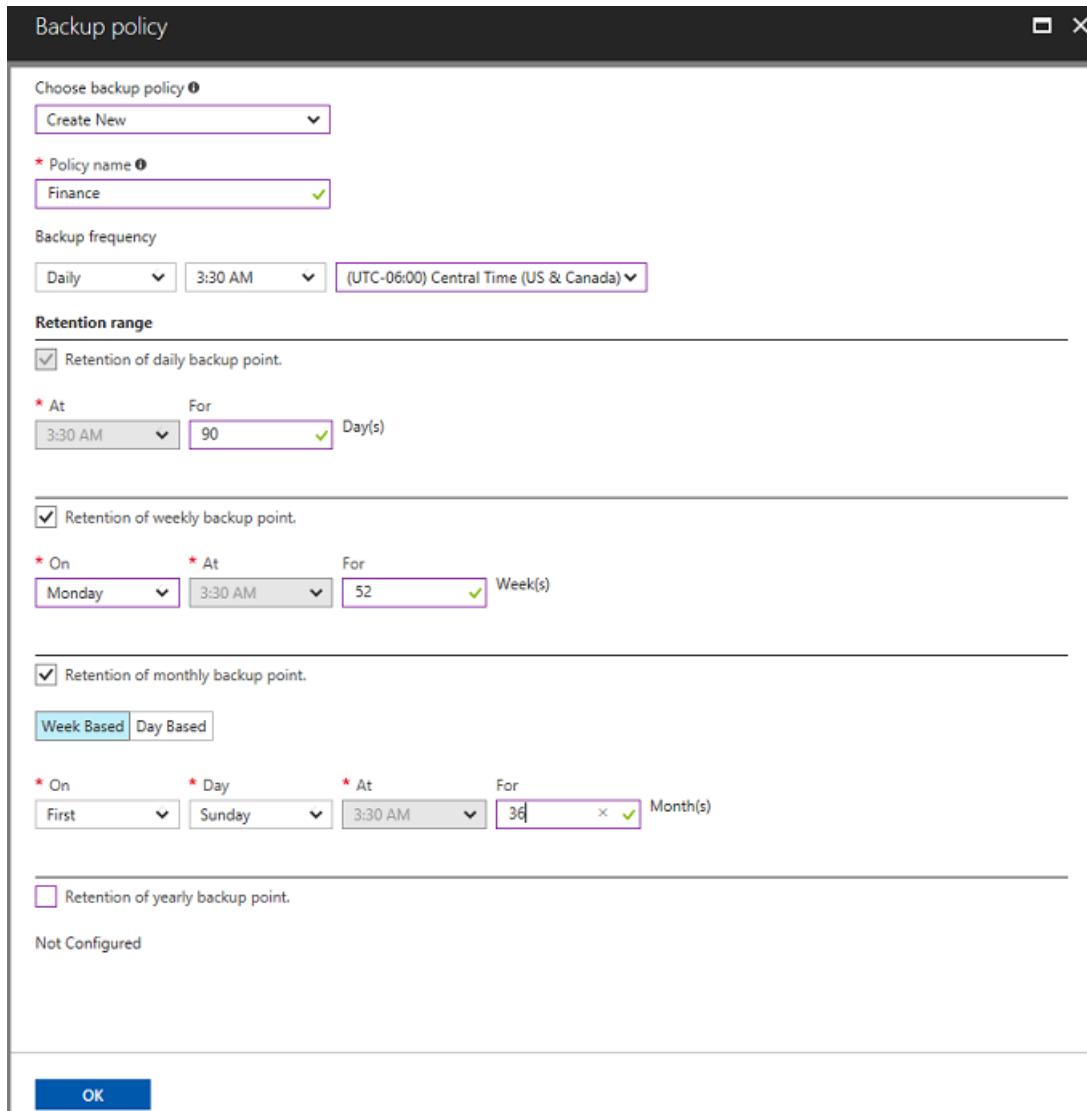
2. On the vault dashboard menu, click **Backup** to open the Backup menu.
3. On the Backup Goal menu, in the **Where is your workload running?** drop-down menu, choose *Azure*. From the **What do you want to backup?** drop-down, choose *Virtual machine*, and click **Backup**.

These actions prepare the Recovery Services vault for interacting with a virtual machine. Recovery Services vaults have a default policy that creates a restore point each day, and retains the restore points for 30 days.

4. To create a new policy, on the Backup policy menu, from the **Choose backup policy** drop-down menu, select *Create New*.

5. In the **Backup policy** menu, for **Policy Name** type *Finance*. Enter the following changes for the Backup policy:

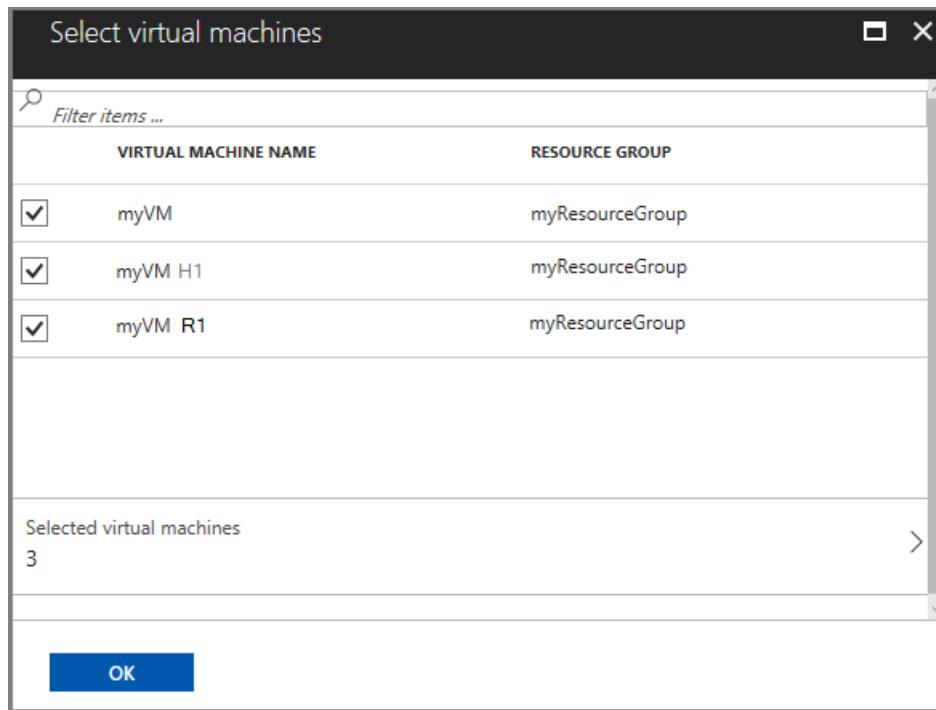
- For **Backup frequency** set the timezone for *Central Time*. Since the sports complex is in Texas, the owner wants the timing to be local. Leave the backup frequency set to Daily at 3:30AM.
- For **Retention of daily backup point**, set the period to 90 days.
- For **Retention of weekly backup point**, use the *Monday* restore point and retain it for 52 weeks.
- For **Retention of monthly backup point**, use the restore point from First Sunday of the month, and retain it for 36 months.
- Deselect the **Retention of yearly backup point** option. The leader of Finance doesn't want to keep data longer than 36 months.
- Click **OK** to create the backup policy.



After creating the backup policy, associate the policy with the virtual machines.

6. In the **Select virtual machines** dialog select *myVM* and click **OK** to deploy the backup policy to the virtual machines.

All virtual machines that are in the same location, and are not already associated with a backup policy, appear. *myVMH1* and *myVMR1* are selected to be associated with the *Finance* policy.



When the deployment completes, you receive a notification that deployment successfully completed.

Initial backup

You have enabled backup for the Recovery Services vaults, but an initial backup has not been created. It is a disaster recovery best practice to trigger the first backup, so that your data is protected.

To run an on-demand backup job:

1. On the vault dashboard, click **3** under **Backup Items**, to open the Backup Items menu.

The **Backup Items** menu opens.

2. On the **Backup Items** menu, click **Azure Virtual Machine** to open the list of virtual machines associated with the vault.

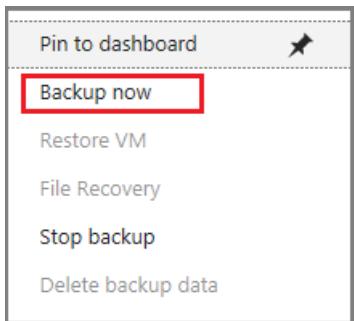
NAME	RESOURCE GROUP	BACKUP PRE-CHECK	LAST BACKUP STATUS	LATEST RESTORE POINT
myVM	myResourceGroup	Passed	Warning(Initial backu...)	Context menu
buntu	rasquill-security	Passed	Warning(Initial backu...)	...
ops	rhelpfiles	Passed	Warning(Initial backu...)	...

The **Backup Items** list opens.

NAME	RESOURCE GROUP	BACKUP PRE-CHECK	LAST BACKUP STATUS	LATEST RESTORE POINT	Context menu
myVM	myResourceGroup	Passed	Warning(Initial backu...)		...
buntu	rasquill-security	Passed	Warning(Initial backu...)		...
ops	rhelpfiles	Passed	Warning(Initial backu...)		...

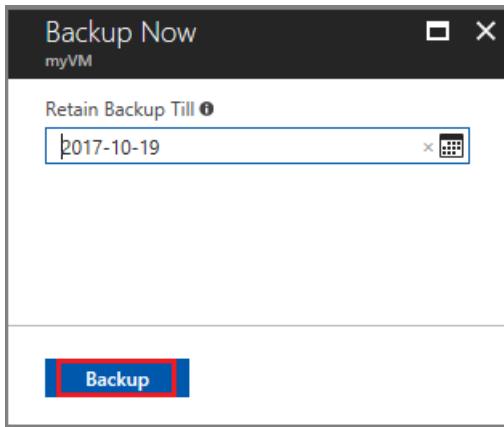
3. On the **Backup Items** list, click the ellipses ... to open the Context menu.

4. On the Context menu, select **Backup now**.



The Backup Now menu opens.

5. On the Backup Now menu, enter the last day to retain the recovery point, and click **Backup**.



Deployment notifications let you know the backup job has been triggered, and that you can monitor the progress of the job on the Backup jobs page. Depending on the size of your virtual machine, creating the initial backup may take a while.

When the initial backup job completes, you can see its status in the Backup job menu. The on-demand backup job created the initial restore point for *myVM*. If you want to back up other virtual machines, repeat these steps for each virtual machine.

NAME	RESOURCE GROUP	BACKUP PRE-CHECK	LAST BACKUP STATUS	LATEST RESTORE POINT	...
myVM	myResourceGroup	Passed	Success	9/19/2017 6:52:32 PM	...

Clean up resources

If you plan to continue on to work with subsequent tutorials, do not clean up the resources created in this tutorial. If you do not plan to continue, use the following steps to delete all resources created by this tutorial in the Azure portal.

1. On the **myRecoveryServicesVault** dashboard, click **3** under **Backup Items**, to open the Backup Items menu.

myRecoveryServicesVault

Recovery Services vault

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

SETTINGS

Properties

Locks

Automation script

GETTING STARTED

Backup

Site Recovery

MONITORING AND REPORTS

Jobs

Alerts and Events

Backup Reports

Backup items

3

Status

Active

Location

West Europe

Subscription name

subscriptionID

Subscription ID

subscription number

Backup management servers

0

Replicated items

0

Monitoring

Backup Alerts (last 24...)

Critical 0

Warning 0

Backup Pre-Check Status (Azure VMs)

CRITICAL 0

TOTAL 0

WARNING 0

Site Recovery Health

Unhealthy serv... 0

Events 0

Updates availa... 0

2. On the **Backup Items** menu, click **Azure Virtual Machine** to open the list of virtual machines associated with the vault.

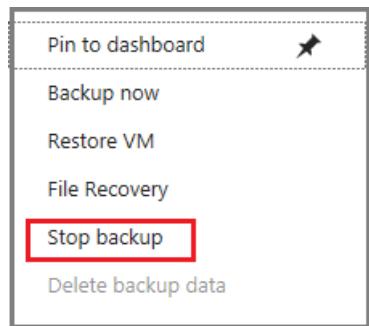
BACKUP MANAGEMENT TYPE	BACKUP ITEM COUNT
Azure Virtual Machine	3
Azure Backup Agent	0
Azure Backup Server	0

The **Backup Items** list opens.

3. In the **Backup Items** menu, click the ellipsis to open the Context menu.

NAME	RESOURCE GROUP	BACKUP PRE-CHECK	LAST BACKUP STATUS	LATEST RESTORE POINT	
myVM	myResourceGroup	Passed	Success	9/19/2017 6:22:37 PM	...
myVM H1	myResourceGroup	Passed	Success	9/19/2017 6:32:35 PM	...
myVM R1	myResourceGroup	Passed	Success	9/19/2017 6:56:17 PM	...

4. On the context menu select **Stop backup** to open Stop Backup menu.



5. In the **Stop Backup** menu, select the upper drop-down menu and choose **Delete Backup Data**.
 6. In the **Type the name of the Backup item** dialog, type *myVM*.
 7. Once the backup item is verified (a checkmark appears), **Stop backup** button is enabled. Click **Stop Backup** to stop the policy and delete the restore points.

8. In the **myRecoveryServicesVault** menu, click **Delete**.

The screenshot shows the Azure portal interface for a Recovery Services vault named 'myRecoveryServicesVault'. The left sidebar includes options like Overview, Activity log, Access control (IAM), and Tags. The main content area has tabs for Backup, Replicate, and Delete, with the Delete tab highlighted by a red box. A survey message at the top encourages users to take a survey about their experience with Azure Backup and Site Recovery. The 'Essentials' section displays the following information:

Resource group	Backup items
myResourceGroup	0
Status	Backup management servers
Active	0

Once the vault is deleted, you return to the list of Recovery Services vaults.

Next steps

In this tutorial you used the Azure portal to:

- Create a Recovery Services vault
- Set the vault to protect virtual machines
- Create a custom backup and retention policy
- Assign the policy to protect multiple virtual machines
- Trigger an on-demand back up for virtual machines

Continue to the next tutorial to restore an Azure virtual machine from disk.

[Restore VMs using CLI](#)

Restore a disk and create a recovered VM in Azure

6/19/2019 • 6 minutes to read • [Edit Online](#)

Azure Backup creates recovery points that are stored in geo-redundant recovery vaults. When you restore from a recovery point, you can restore the whole VM or individual files. This article explains how to restore a complete VM using CLI. In this tutorial you learn how to:

- List and select recovery points
- Restore a disk from a recovery point
- Create a VM from the restored disk

For information on using PowerShell to restore a disk and create a recovered VM, see [Back up and restore Azure VMs with PowerShell](#).

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services.

Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.18 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install the Azure CLI](#).

Prerequisites

This tutorial requires a Linux VM that has been protected with Azure Backup. To simulate an accidental VM deletion and recovery process, you create a VM from a disk in a recovery point. If you need a Linux VM that has been protected with Azure Backup, see [Back up a virtual machine in Azure with the CLI](#).

Backup overview

When Azure initiates a backup, the backup extension on the VM takes a point-in-time snapshot. The backup extension is installed on the VM when the first backup is requested. Azure Backup can also take a snapshot of the underlying storage if the VM is not running when the backup takes place.

By default, Azure Backup takes a file system consistent backup. Once Azure Backup takes the snapshot, the data is transferred to the Recovery Services vault. To maximize efficiency, Azure Backup identifies and transfers only the blocks of data that have changed since the previous backup.

When the data transfer is complete, the snapshot is removed and a recovery point is created.

List available recovery points

To restore a disk, you select a recovery point as the source for the recovery data. As the default policy creates a recovery point each day and retains them for 30 days, you can keep a set of recovery points that allows you to select a particular point in time for recovery.

To see a list of available recovery points, use [az backup recoverypoint list](#). The recovery point **name** is used to recover disks. In this tutorial, we want the most recent recovery point available. The `--query [0].name` parameter selects the most recent recovery point name as follows:

```
az backup recoverypoint list \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--container-name myVM \
--item-name myVM \
--query [0].name \
--output tsv
```

Restore a VM disk

To restore your disk from the recovery point, you first create an Azure storage account. This storage account is used to store the restored disk. In additional steps, the restored disk is used to create a VM.

1. To create a storage account, use [az storage account create](#). The storage account name must be all lowercase, and be globally unique. Replace *mystorageaccount* with your own unique name:

```
az storage account create \
--resource-group myResourceGroup \
--name mystorageaccount \
--sku Standard_LRS
```

2. Restore the disk from your recovery point with [az backup restore restore-disks](#). Replace *mystorageaccount* with the name of the storage account you created in the preceding command. Replace *myRecoveryPointName* with the recovery point name you obtained in the output from the previous [az backup recoverypoint list](#) command:

```
az backup restore restore-disks \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--container-name myVM \
--item-name myVM \
--storage-account mystorageaccount \
--rp-name myRecoveryPointName
```

Monitor the restore job

To monitor the status of restore job, use [az backup job list](#):

```
az backup job list \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--output table
```

The output is similar to the following example, which shows the restore job is *InProgress*:

Name	Operation	Status	Item Name	Start Time UTC	Duration
7f2ad916	Restore	InProgress	myvm	2017-09-19T19:39:52	0:00:34.520850
a0a8e5e6	Backup	Completed	myvm	2017-09-19T03:09:21	0:15:26.155212
fe5d0414	ConfigureBackup	Completed	myvm	2017-09-19T03:03:57	0:00:31.191807

When the *Status* of the restore job reports *Completed*, the disk has been restored to the storage account.

Convert the restored disk to a Managed Disk

The restore job creates an unmanaged disk. In order to create a VM from the disk, it must first be converted to a managed disk.

1. Obtain the connection information for your storage account with [az storage account show-connection-string](#). Replace *mystorageaccount* with the name of your storage account as follows:

```
export AZURE_STORAGE_CONNECTION_STRING=$( az storage account show-connection-string \
--resource-group myResourceGroup \
--output tsv \
--name mystorageaccount )
```

2. Your unmanaged disk is secured in the storage account. The following commands get information about your unmanaged disk and create a variable named *uri* that is used in the next step when you create the Managed Disk.

```
container=$(az storage container list --query [0].name -o tsv)
blob=$(az storage blob list --container-name $container --query [0].name -o tsv)
uri=$(az storage blob url --container-name $container --name $blob -o tsv)
```

3. Now you can create a Managed Disk from your recovered disk with [az disk create](#). The *uri* variable from the preceding step is used as the source for your Managed Disk.

```
az disk create \
--resource-group myResourceGroup \
--name myRestoredDisk \
--source $uri
```

4. As you now have a Managed Disk from your restored disk, clean up the unmanaged disk and storage account with [az storage account delete](#). Replace *mystorageaccount* with the name of your storage account as follows:

```
az storage account delete \
--resource-group myResourceGroup \
--name mystorageaccount
```

Create a VM from the restored disk

The final step is to create a VM from the Managed Disk.

1. Create a VM from your Managed Disk with [az vm create](#) as follows:

```
az vm create \
--resource-group myResourceGroup \
--name myRestoredVM \
--attach-os-disk myRestoredDisk \
--os-type linux
```

2. To confirm that your VM has been created from your recovered disk, list the VMs in your resource group with [az vm list](#) as follows:

```
az vm list --resource-group myResourceGroup --output table
```

Next steps

In this tutorial, you restored a disk from a recovery point and then created a VM from the disk. You learned how to:

- List and select recovery points
- Restore a disk from a recovery point
- Create a VM from the restored disk

Advance to the next tutorial to learn about restoring individual files from a recovery point.

[Restore files to a virtual machine in Azure](#)

Restore files to a virtual machine in Azure

6/19/2019 • 6 minutes to read • [Edit Online](#)

Azure Backup creates recovery points that are stored in geo-redundant recovery vaults. When you restore from a recovery point, you can restore the whole VM or individual files. This article details how to restore individual files. In this tutorial you learn how to:

- List and select recovery points
- Connect a recovery point to a VM
- Restore files from a recovery point

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.18 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install the Azure CLI](#).

Prerequisites

This tutorial requires a Linux VM that has been protected with Azure Backup. To simulate an accidental file deletion and recovery process, you delete a page from a web server. If you need a Linux VM that runs a webserver and has been protected with Azure Backup, see [Back up a virtual machine in Azure with the CLI](#).

Backup overview

When Azure initiates a backup, the backup extension on the VM takes a point-in-time snapshot. The backup extension is installed on the VM when the first backup is requested. Azure Backup can also take a snapshot of the underlying storage if the VM is not running when the backup takes place.

By default, Azure Backup takes a file system consistent backup. Once Azure Backup takes the snapshot, the data is transferred to the Recovery Services vault. To maximize efficiency, Azure Backup identifies and transfers only the

blocks of data that have changed since the previous backup.

When the data transfer is complete, the snapshot is removed and a recovery point is created.

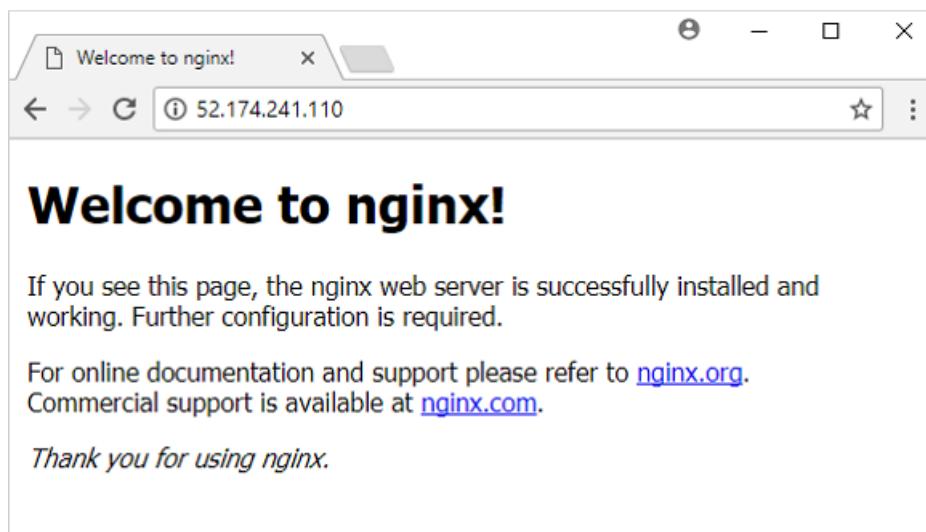
Delete a file from a VM

If you accidentally delete or make changes to a file, you can restore individual files from a recovery point. This process allows you to browse the files backed up in a recovery point and restore only the files you need. In this example, we delete a file from a web server to demonstrate the file-level recovery process.

1. To connect to your VM, obtain the IP address of your VM with [az vm show](#):

```
az vm show --resource-group myResourceGroup --name myVM -d --query [publicIps] --o tsv
```

2. To confirm that your web site currently works, open a web browser to the public IP address of your VM. Leave the web browser window open.



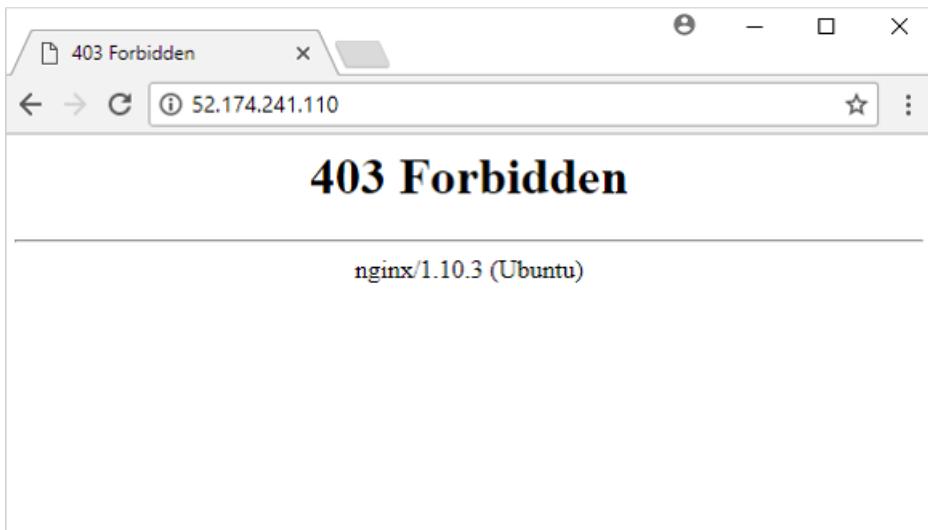
3. Connect to your VM with SSH. Replace *publicIpAddress* with the public IP address that you obtained in a previous command:

```
ssh publicIpAddress
```

4. Delete the default page from the web server at */var/www/html/index.nginx-debian.html* as follows:

```
sudo rm /var/www/html/index.nginx-debian.html
```

5. In your web browser, refresh the web page. The web site no longer loads the page, as shown in the following example:



6. Close the SSH session to your VM as follows:

```
exit
```

Generate file recovery script

To restore your files, Azure Backup provides a script to run on your VM that connects your recovery point as a local drive. You can browse this local drive, restore files to the VM itself, then disconnect the recovery point. Azure Backup continues to back up your data based on the assigned policy for schedule and retention.

1. To list recovery points for your VM, use [az backup recoverypoint list](#). In this example, we select the most recent recovery point for the VM named *myVM* that is protected in *myRecoveryServicesVault*:

```
az backup recoverypoint list \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--container-name myVM \
--item-name myVM \
--query [0].name \
--output tsv
```

2. To obtain the script that connects, or mounts, the recovery point to your VM, use [az backup restore files mount-rp](#). The following example obtains the script for the VM named *myVM* that is protected in *myRecoveryServicesVault*.

Replace *myRecoveryPointName* with the name of the recovery point that you obtained in the preceding command:

```
az backup restore files mount-rp \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--container-name myVM \
--item-name myVM \
--rp-name myRecoveryPointName
```

The script is downloaded and a password is displayed, as in the following example:

```
File downloaded: myVM_we_1571974050985163527.sh. Use password c068a041ce12465
```

3. To transfer the script to your VM, use Secure Copy (SCP). Provide the name of your downloaded script, and

replace *publicIpAddress* with the public IP address of your VM. Make sure you include the trailing `:` at the end of the SCP command as follows:

```
scp myVM_we_1571974050985163527.sh 52.174.241.110:
```

Restore file to your VM

With the recovery script copied to your VM, you can now connect the recovery point and restore files.

1. Connect to your VM with SSH. Replace *publicIpAddress* with the public IP address of your VM as follows:

```
ssh publicIpAddress
```

2. To allow your script to run correctly, add execute permissions with **chmod**. Enter the name of your own script:

```
chmod +x myVM_we_1571974050985163527.sh
```

3. To mount the recovery point, run the script. Enter the name of your own script:

```
./myVM_we_1571974050985163527.sh
```

As the script runs, you are prompted to enter a password to access the recovery point. Enter the password shown in the output from the previous [az backup restore files mount-rp](#) command that generated the recovery script.

The output from the script gives you the path for the recovery point. The following example output shows that the recovery point is mounted at `/home/azureuser/myVM-20170919213536/Volume1`:

```
Microsoft Azure VM Backup - File Recovery

Please enter the password as shown on the portal to securely connect to the recovery point. :
c068a041ce12465

Connecting to recovery point using ISCSI service...

Connection succeeded!

Please wait while we attach volumes of the recovery point to this machine...

***** Volumes of the recovery point and their mount paths on this machine *****

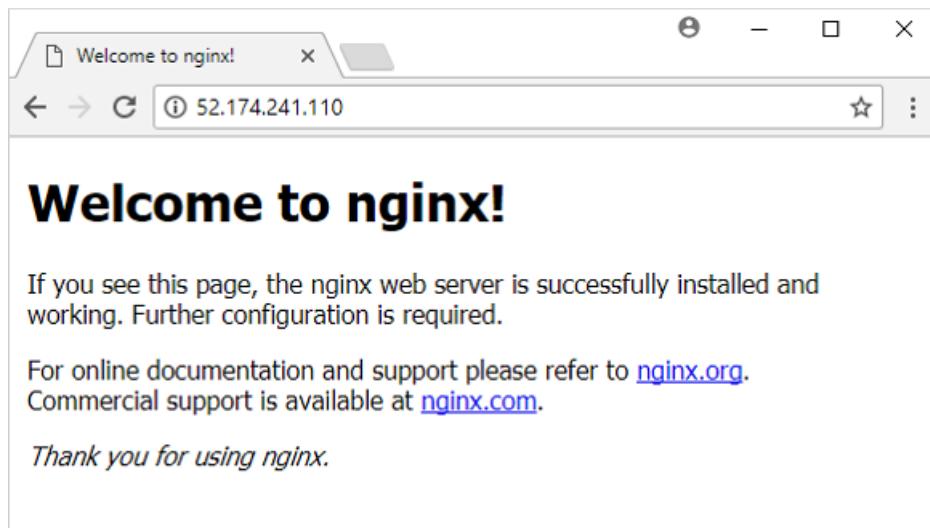
Sr.No. | Disk | Volume | MountPath
1) | /dev/sdc | /dev/sdc1 | /home/azureuser/myVM-20170919213536/Volume1

***** Open File Explorer to browse for files. *****
```

4. Use **cp** to copy the NGINX default web page from the mounted recovery point back to the original file location. Replace the `/home/azureuser/myVM-20170919213536/Volume1` mount point with your own location:

```
sudo cp /home/azureuser/myVM-20170919213536/Volume1/var/www/html/index.nginx-debian.html /var/www/html/
```

5. In your web browser, refresh the web page. The web site now loads correctly again, as shown in the following example:



6. Close the SSH session to your VM as follows:

```
exit
```

7. Unmount the recovery point from your VM with `az backup restore files unmount-rp`. The following example unmounts the recovery point from the VM named *myVM* in *myRecoveryServicesVault*.

Replace *myRecoveryPointName* with the name of your recovery point that you obtained in the previous commands:

```
az backup restore files unmount-rp \
--resource-group myResourceGroup \
--vault-name myRecoveryServicesVault \
--container-name myVM \
--item-name myVM \
--rp-name myRecoveryPointName
```

Next steps

In this tutorial, you connected a recovery point to a VM and restored files for a web server. You learned how to:

- List and select recovery points
- Connect a recovery point to a VM
- Restore files from a recovery point

Advance to the next tutorial to learn about how to back up Windows Server to Azure.

[Back up Windows Server to Azure](#)

About Site Recovery

6/2/2019 • 3 minutes to read • [Edit Online](#)

Welcome to the Azure Site Recovery service! This article provides a quick service overview.

As an organization you need to adopt a business continuity and disaster recovery (BCDR) strategy that keeps your data safe, and your apps and workloads up and running, when planned and unplanned outages occur.

Azure Recovery Services contribute to your BCDR strategy:

- **Site Recovery service:** Site Recovery helps ensure business continuity by keeping business apps and workloads running during outages. Site Recovery replicates workloads running on physical and virtual machines (VMs) from a primary site to a secondary location. When an outage occurs at your primary site, you fail over to secondary location, and access apps from there. After the primary location is running again, you can fail back to it.
- **Backup service:** The [Azure Backup](#) service keeps your data safe and recoverable by backing it up to Azure.

Site Recovery can manage replication for:

- Azure VMs replicating between Azure regions.
- On-premises VMs, Azure Stack VMs and physical servers.

What does Site Recovery provide?

FEATURE	DETAILS
Simple BCDR solution	Using Site Recovery, you can set up and manage replication, failover, and fallback from a single location in the Azure portal.
Azure VM replication	You can set up disaster recovery of Azure VMs from a primary region to a secondary region.
On-premises VM replication	You can replicate on-premises VMs and physical servers to Azure, or to a secondary on-premises datacenter. Replication to Azure eliminates the cost and complexity of maintaining a secondary datacenter.
Workload replication	Replicate any workload running on supported Azure VMs, on-premises Hyper-V and VMware VMs, and Windows/Linux physical servers.
Data resilience	Site Recovery orchestrates replication without intercepting application data. When you replicate to Azure, data is stored in Azure storage, with the resilience that provides. When failover occurs, Azure VMs are created, based on the replicated data.
RTO and RPO targets	Keep recovery time objectives (RTO) and recovery point objectives (RPO) within organizational limits. Site Recovery provides continuous replication for Azure VMs and VMware VMs, and replication frequency as low as 30 seconds for Hyper-V. You can reduce RTO further by integrating with Azure Traffic Manager .

FEATURE	DETAILS
Keep apps consistent over failover	You can replicate using recovery points with application-consistent snapshots. These snapshots capture disk data, all data in memory, and all transactions in process.
Testing without disruption	You can easily run disaster recovery drills, without affecting ongoing replication.
Flexible failovers	You can run planned failovers for expected outages with zero-data loss, or unplanned failovers with minimal data loss (depending on replication frequency) for unexpected disasters. You can easily fail back to your primary site when it's available again.
Customized recovery plans	Using recovery plans, can customize and sequence the failover and recovery of multi-tier applications running on multiple VMs. You group machines together in a recovery plan, and optionally add scripts and manual actions. Recovery plans can be integrated with Azure automation runbooks.
BCDR integration	Site Recovery integrates with other BCDR technologies. For example, you can use Site Recovery to protect the SQL Server backend of corporate workloads, with native support for SQL Server AlwaysOn, to manage the failover of availability groups.
Azure automation integration	A rich Azure Automation library provides production-ready, application-specific scripts that can be downloaded and integrated with Site Recovery.
Network integration	Site Recovery integrates with Azure for simple application network management, including reserving IP addresses, configuring load-balancers, and integrating Azure Traffic Manager for efficient network switchovers.

What can I replicate?

SUPPORTED	DETAILS
Replication scenarios	<p>Replicate Azure VMs from one Azure region to another.</p> <p>Replicate on-premises VMware VMs, Hyper-V VMs, physical servers (Windows and Linux), Azure Stack VMs to Azure.</p>
	<p>Replicate AWS Windows instances to Azure.</p> <p>Replicate on-premises VMware VMs, Hyper-V VMs managed by System Center VMM, and physical servers to a secondary site.</p>
Regions	Review supported regions for Site Recovery.
Replicated machines	Review the replication requirements for Azure VM replication, on-premises VMware VMs and physical servers , and on-premises Hyper-V VMs .

SUPPORTED	DETAILS
Workloads	You can replicate any workload running on a machine that's supported for replication. In addition, the Site Recovery team have performed app-specific testing for a number of apps .

Next steps

- Read more about [workload support](#).
- Get started with [Azure VM replication between regions](#).

Set up disaster recovery for Azure VMs

6/2/2019 • 9 minutes to read • [Edit Online](#)

The [Azure Site Recovery](#) service contributes to your disaster recovery strategy by managing and orchestrating replication, failover, and failback of on-premises machines and Azure virtual machines (VMs).

This tutorial shows you how to set up disaster recovery for Azure VMs by replicating them from one Azure region to another. In this tutorial, you learn how to:

- Create a Recovery Services vault
- Verify target resource settings
- Set up outbound network connectivity for VMs
- Enable replication for a VM

NOTE

This article provides instructions for deploying disaster recovery with the simplest settings. If you want to learn about customized settings, review the articles under the [How To section](#).

Prerequisites

To complete this tutorial:

- Review the [scenario architecture and components](#).
- Review the [support requirements](#) before you start.

Create a Recovery Services vault

Create the vault in any region, except the source region.

1. Sign in to the [Azure portal](#) > **Recovery Services**.
2. Click **Create a resource** > **Management Tools** > **Backup and Site Recovery**.
3. In **Name**, specify a friendly name to identify the vault. If you have more than one subscription, select the appropriate one.
4. Create a resource group or select an existing one. Specify an Azure region. To check supported regions, see geographic availability in [Azure Site Recovery Pricing Details](#).
5. To quickly access the vault from the dashboard, click **Pin to dashboard** and then click **Create**.

The screenshot shows the Azure portal interface for creating a Recovery Services vault. On the left, there's a sidebar titled 'Recovery Services vaults' with a list of existing vaults like 'contosoassessment-Migr...', 'ContosoCorporation-Rec...', 'ContosoDemo', etc. The main panel is titled 'Recovery Services vault' and contains fields for 'Name' (set to 'Vault1'), 'Subscription' (selected as '<subscription-name>'), 'Resource group' (set to 'RG1'), and 'Location' (set to 'West Central US'). A note at the top right says 'Click here to try new preview create vault experience with Tags support.'

The new vault is added to the **Dashboard** under **All resources**, and on the main **Recovery Services vaults** page.

Verify target resource settings

1. Verify that your Azure subscription allows you to create VMs in the target region. Contact support to enable the required quota.
2. Make sure your subscription has enough resources to support VM sizes that match your source VMs. Site Recovery picks the same size, or the closest possible size, for the target VM.

Set up outbound network connectivity for VMs

For Site Recovery to work as expected, you need to modify outbound network connectivity from the VMs that you want to replicate.

NOTE

Site Recovery doesn't support using an authentication proxy to control network connectivity.

Outbound connectivity for URLs

If you're using a URL-based firewall proxy to control outbound connectivity, allow access to these URLs.

URL	DETAILS
*.blob.core.windows.net	Allows data to be written from the VM to the cache storage account in the source region.
login.microsoftonline.com	Provides authorization and authentication to Site Recovery service URLs.
*.hypervrecoverymanager.windowsazure.com	Allows the VM to communicate with the Site Recovery service.
*.servicebus.windows.net	Allows the VM to write Site Recovery monitoring and diagnostics data.

Outbound connectivity for IP address ranges

If you want to control outbound connectivity using IP addresses instead of URLs, allow these addresses for IP-based firewalls, proxy, or NSG rules.

- [Microsoft Azure Datacenter IP Ranges](#)
- [Windows Azure Datacenter IP Ranges in Germany](#)
- [Windows Azure Datacenter IP Ranges in China](#)
- [Office 365 URLs and IP address ranges](#)
- [Site Recovery service endpoint IP addresses](#)

If you're using NSG you can create a storage service tag NSG rules for the source region. [Learn more](#).

Verify Azure VM certificates

Check that the VMs you want to replicate have the latest root certificates. If they don't the VM can't register to Site Recovery, due to security constraints.

- For Windows VMs, install all the latest Windows updates on the VM, so that all the trusted root certificates are on the machine. In a disconnected environment, follow the standard Windows Update and certificate update processes for your organization.
- For Linux VMs, follow the guidance provided by your Linux distributor, to get the latest trusted root certificates and certificate revocation list on the VM.

Set permissions on the account

Azure Site Recovery provides three built-in roles to control Site Recovery management operations.

- **Site Recovery Contributor** - This role has all permissions required to manage Azure Site Recovery operations in a Recovery Services vault. A user with this role, however, can't create or delete a Recovery Services vault or assign access rights to other users. This role is best suited for disaster recovery administrators who can enable and manage disaster recovery for applications or entire organizations.
- **Site Recovery Operator** - This role has permissions to execute and manage Failover and Failback operations. A user with this role can't enable or disable replication, create or delete vaults, register new infrastructure, or assign access rights to other users. This role is best suited for a disaster recovery operator who can fail over virtual machines or applications when instructed by application owners and IT administrators. Post resolution of the disaster, the DR operator can reprotect and failback the virtual machines.
- **Site Recovery Reader** - This role has permissions to view all Site Recovery management operations. This role is best suited for an IT monitoring executive who can monitor the current state of protection and raise support tickets.

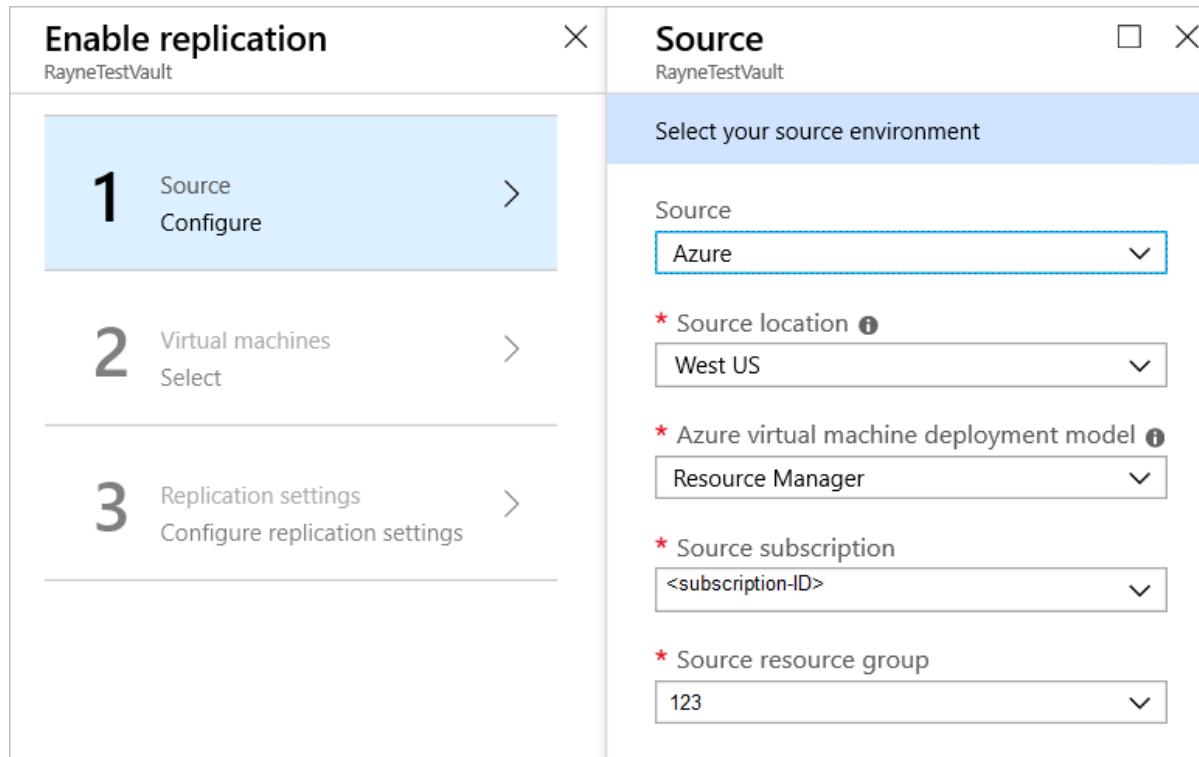
Learn more about [Azure RBAC built-in roles](#).

Enable replication for a VM

Select the source

1. In Recovery Services vaults, click the vault name > **+Replicate**.
2. In **Source**, select **Azure**.
3. In **Source location**, select the source Azure region where your VMs are currently running.
4. Select the **Source subscription** where the virtual machines are running. This can be any subscription within the same Azure Active Directory tenant where your recovery services vault exists.

- Select the **Source resource group**, and click **OK** to save the settings.



Select the VMs

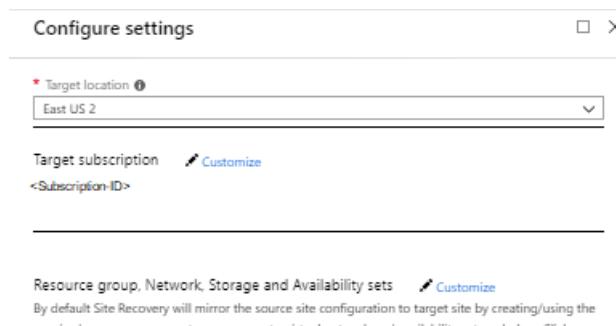
Site Recovery retrieves a list of the VMs associated with the subscription and resource group/cloud service.

- In **Virtual Machines**, select the VMs you want to replicate.
- Click **OK**.

Configure replication settings

Site Recovery creates default settings and replication policy for the target region. You can change the settings as required.

- Click **Settings** to view the target and replication settings.
- To override the default target settings, click **Customize** next to **Resource group, Network, Storage and Availability**.



- Customize target settings as summarized in the table.

SETTING	DETAILS
Target subscription	By default, the target subscription is the same as the source subscription. Click 'Customize' to select a different target subscription within the same Azure Active Directory tenant.

Setting	Details
Target location	<p>The target region used for disaster recovery.</p> <p>We recommend that the target location matches the location of the Site Recovery vault.</p>
Target resource group	<p>The resource group in the target region that holds Azure VMs after failover.</p> <p>By default, Site Recovery creates a new resource group in the target region with an "asr" suffix. The location of the target resource group can be any region except the region in which your source virtual machines are hosted.</p>
Target virtual network	<p>The network in the target region that VMs are located after failover.</p> <p>By default, Site Recovery creates a new virtual network (and subnets) in the target region with an "asr" suffix.</p>
Cache storage accounts	<p>Site Recovery uses a storage account in the source region. Changes to source VMs are sent to this account before replication to the target location.</p> <p>If you are using a firewall-enabled cache storage account, make sure that you enable Allow trusted Microsoft services. Learn more.</p>
Target storage accounts (source VM uses non-managed disks)	<p>By default, Site Recovery creates a new storage account in the target region to mirror the source VM storage account.</p> <p>Enable Allow trusted Microsoft services if you're using a firewall-enabled cache storage account.</p>
Replica managed disks (If source VM uses managed disks)	<p>By default, Site Recovery creates replica managed disks in the target region to mirror the source VM's managed disks with the same storage type (Standard or premium) as the source VM's managed disk. You can only customize Disk type</p>
Target availability sets	<p>By default, Azure Site Recovery creates a new availability set in the target region with name having "asr" suffix for the VMs part of an availability set in source region. In case availability set created by Azure Site Recovery already exists, it is reused.</p>

SETTING	DETAILS
Target availability zones	<p>By default, Site Recovery assigns the same zone number as the source region in target region if the target region supports availability zones.</p> <p>If the target region doesn't support availability zones, the target VMs are configured as single instances by default.</p> <p>Click Customize to configure VMs as part of an availability set in the target region.</p> <p>You can't change the availability type (single instance, availability set or availability zone) after you enable replication. You need to disable and enable replication to change the availability type.</p>

- To customize replication policy settings, click **Customize** next to **Replication policy**, and modify the settings as needed.

SETTING	DETAILS
Replication policy name	Policy name.
Recovery point retention	By default, Site Recovery keeps recovery points for 24 hours. You can configure a value between 1 and 72 hours.
App-consistent snapshot frequency	<p>By default, Site Recovery takes an app-consistent snapshot every 4 hours. You can configure any value between 1 and 12 hours.</p> <p>An app-consistent snapshot is a point-in-time snapshot of the application data inside the VM. Volume Shadow Copy Service (VSS) ensures that app on the VM are in a consistent state when the snapshot is taken.</p>
Replication group	If your application needs multi-VM consistency across VMs, you can create a replication group for those VMs. By default, the selected VMs are not part of any replication group.

- In **Customize**, select **Yes** for multi-VM consistency if you want to add VMs to a new or existing replication group. Then click **OK**.

NOTE

- All the machines in a replication group have shared crash consistent and app-consistent recovery points when failed over.
- Enabling multi-VM consistency can impact workload performance (it's CPU intensive). It should be used only if machines are running the same workload, and you need consistency across multiple machines.
- You can have a maximum of 16 VMs in a replication group.
- If you enable multi-VM consistency, machines in the replication group communicate with each other over port 20004. Make sure there's no firewall blocking the internal communication between the VMs over this port.
- For Linux VMs in a replication group, ensure the outbound traffic on port 20004 is manually opened in accordance with guidance for the Linux version.

Configure encryption settings

If the source VM has Azure disk encryption (ADE) enabled, review the settings.

1. Verify the settings:

- **Disk encryption key vaults:** By default, Site Recovery creates a new key vault on the source VM disk encryption keys, with an "asr" suffix. If the key vault already exists, it is reused.
- **Key encryption key vaults:** By default, Site Recovery creates a new key vault in the target region. The name has an "asr" suffix, and is based on the source VM key encryption keys. If the key vault created by Site Recovery already exists, it's reused.

2. Click **Customize** to select custom vaults.

NOTE

Only Azure VMs running Windows operating systems and [enabled for encryption with Azure AD app](#) are currently supported by Azure Site Recovery.

Track replication status

1. In **Settings**, click **Refresh** to get the latest status.

2. Track progress and status as follows:

- Track progress of the **Enable protection** job in **Settings > Jobs > Site Recovery Jobs**.
- In **Settings > Replicated Items**, you can view the status of VMs and the initial replication progress. Click the VM to drill down into its settings.

Next steps

In this tutorial, you configured disaster recovery for an Azure VM. Now you can initiate a disaster recovery drill to check that failover is working as expected.

[Run a disaster recovery drill](#)

Run a disaster recovery drill for Azure VMs to a secondary Azure region

5/30/2019 • 2 minutes to read • [Edit Online](#)

The [Azure Site Recovery](#) service contributes to your business continuity and disaster recovery (BCDR) strategy by keeping your business apps up and running available during planned and unplanned outages. Site Recovery manages and orchestrates disaster recovery of on-premises machines and Azure virtual machines (VMs), including replication, failover, and recovery.

This tutorial shows you how to run a disaster recovery drill for an Azure VM, from one Azure region to another, with a test failover. A drill validates your replication strategy without data loss or downtime, and doesn't affect your production environment. In this tutorial, you learn how to:

- Check the prerequisites
- Run a test failover for a single VM

NOTE

This tutorial is intended to guide the user through the steps to perform a DR drill with minimal steps; in case you want to learn more about the various aspects associated with performing a DR drill, including networking considerations, automation or troubleshooting, refer to the documents under 'How To' for Azure VMs.

Prerequisites

- Before you run a test failover, we recommend that you verify the VM properties to make sure everything's as expected. Access the VM properties in **Replicated items**. The **Essentials** blade shows information about machines settings and status.
- **We recommend you use a separate Azure VM network for the test failover**, and not the default network that was set up when you enabled replication.

Run a test failover

1. In **Settings > Replicated Items**, click the VM +**Test Failover** icon.
2. In **Test Failover**, Select a recovery point to use for the failover:
 - **Latest processed**: Fails the VM over to the latest recovery point that was processed by the Site Recovery service. The time stamp is shown. With this option, no time is spent processing data, so it provides a low RTO (Recovery Time Objective)
 - **Latest app-consistent**: This option fails over all VMs to the latest app-consistent recovery point. The time stamp is shown.
 - **Custom**: Select any recovery point.
3. Select the target Azure virtual network to which Azure VMs in the secondary region will be connected, after the failover occurs.
4. To start the failover, click **OK**. To track progress, click the VM to open its properties. Or, you can click the **Test Failover** job in the vault name > **Settings > Jobs > Site Recovery jobs**.
5. After the failover finishes, the replica Azure VM appears in the Azure portal > **Virtual Machines**. Make sure that the VM is running, sized appropriately, and connected to the appropriate network.

6. To delete the VMs that were created during the test failover, click **Cleanup test failover** on the replicated item or the recovery plan. In **Notes**, record and save any observations associated with the test failover.

Next steps

[Run a production failover](#)

Fail over and reprotect Azure VMs between regions

5/30/2019 • 2 minutes to read • [Edit Online](#)

This tutorial describes how to fail over an Azure virtual machine (VM) to a secondary Azure region with the [Azure Site Recovery](#) service. After you've failed over, you reprotect the VM. In this tutorial, you learn how to:

- Fail over the Azure VM
- Reprotect the secondary Azure VM, so that it replicates to the primary region.

NOTE

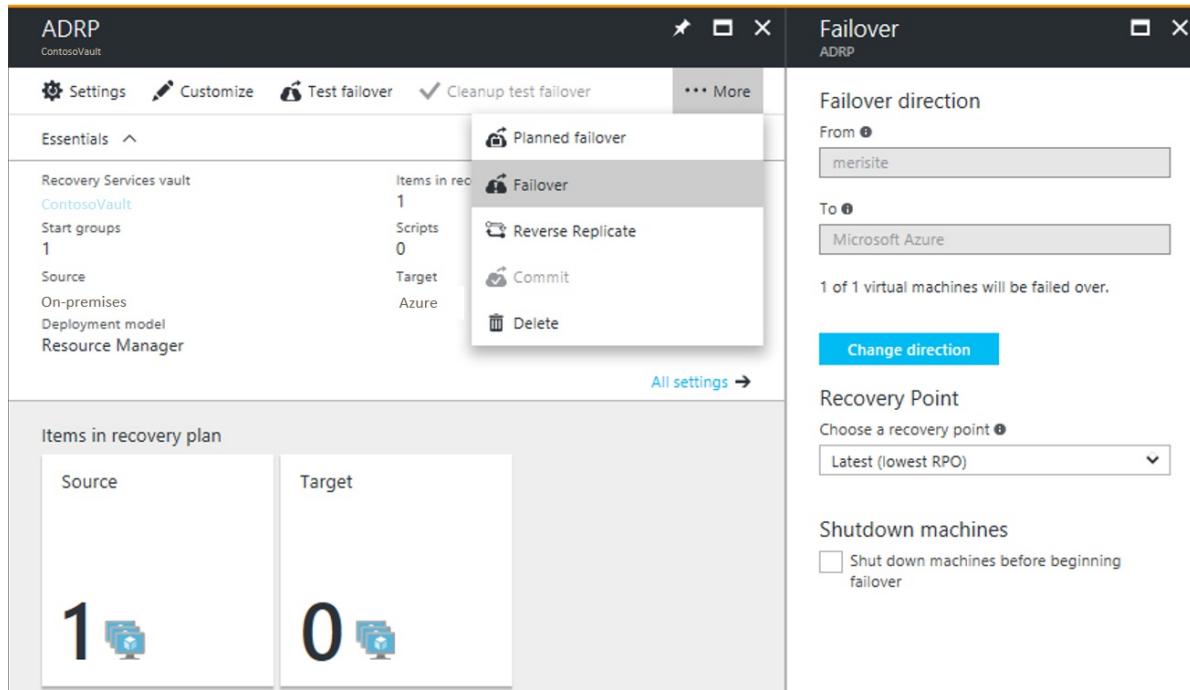
This tutorial contains the simplest path with default settings and minimum customization. For more complex scenarios, use the articles under 'How To' for Azure VMs.

Prerequisites

- Before you start, review [frequently asked questions](#) about failover.
- Make sure that you've completed a [disaster recovery drill](#) to check everything is working as expected.
- Verify the VM properties before you run the test failover. The VM must comply with [Azure requirements](#).

Run a failover to the secondary region

1. In **Replicated items**, select the VM that you want to fail over > **Failover**



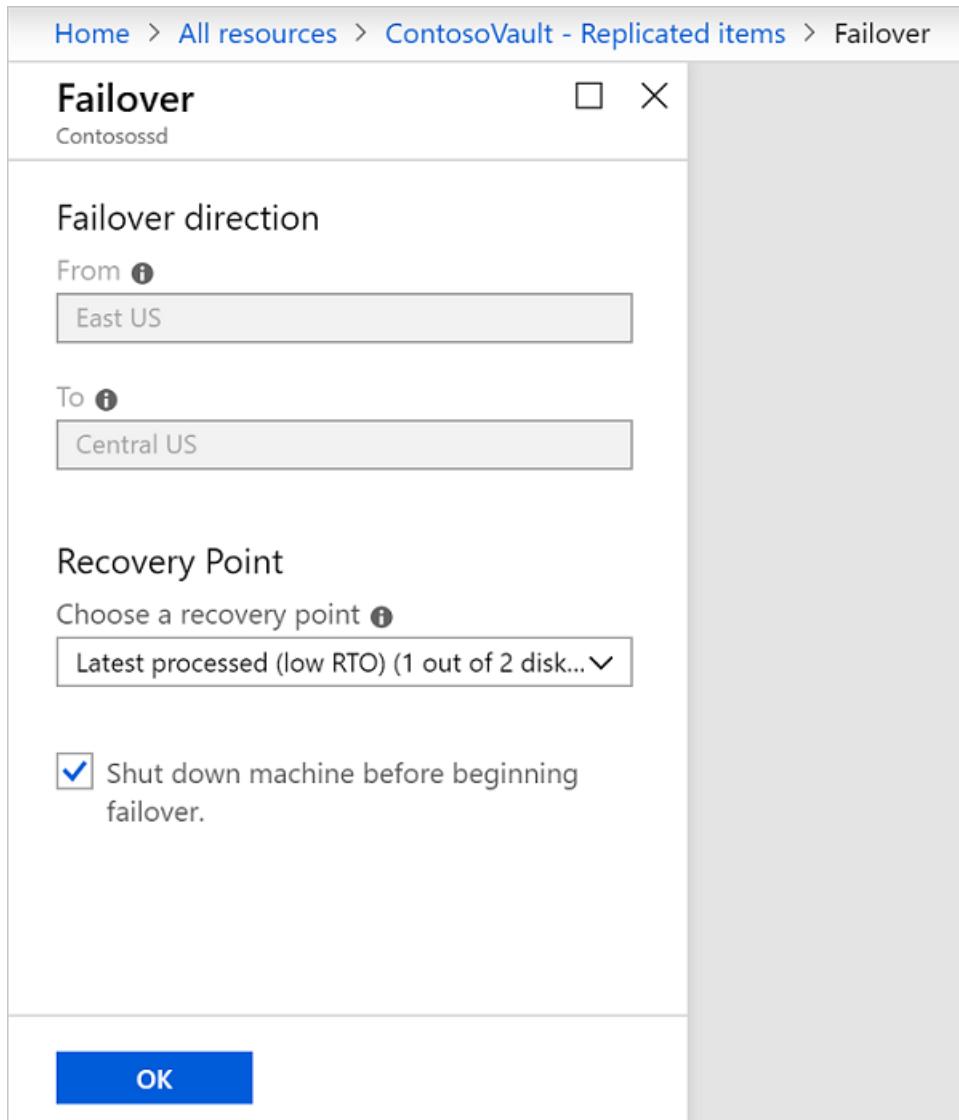
2. In **Failover**, select a **Recovery Point** to fail over to. You can use one of the following options:

- **Latest** (default): Processes all the data in the Site Recovery service and provides the lowest Recovery Point Objective (RPO).
- **Latest processed**: Reverts the virtual machine to the latest recovery point that has been processed by Site Recovery service.
- **Custom**: Fails over to a particular recovery point. This option is useful for performing a test failover.

3. Select **Shut down machine before beginning failover** if you want Site Recovery to attempt to do a shutdown of source VMs before triggering the failover. Failover continues even if shutdown fails. Site Recovery does not clean up the source after failover.
4. Follow the failover progress on the **Jobs** page.
5. After the failover, validate the virtual machine by logging in to it. If you want to go another recovery point for the virtual machine, then you can use **Change recovery point** option.
6. Once you are satisfied with the failed over virtual machine, you can **Commit** the failover. Committing deletes all the recovery points available with the service. You won't now be able to change the recovery point.

NOTE

When you fail over a VM to which you add a disk after you enabled replication for the VM, replication points will show the disks that are available for recovery. For example, if a VM has a single disk and you add a new one, replication points that were created before you added the disk will show that the replication point consists of "1 of 2 disks".

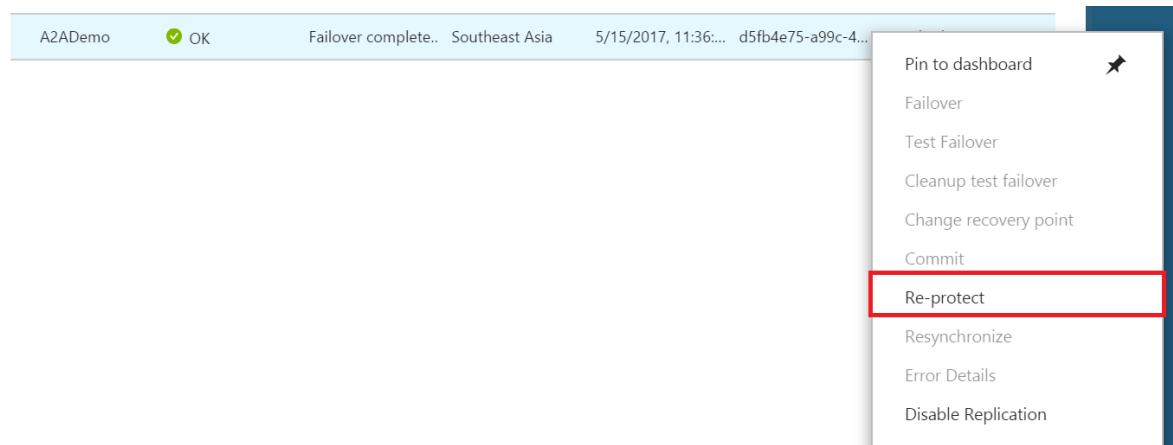


Reprotect the secondary VM

After failover of the VM, you need to reprotect it so that it replicates back to the primary region.

1. Make sure that the VM is in the **Failover committed** state, and check that the primary region is available, and you're able to create and access new resources in it.

2. In **Vault > Replicated items**, right-click the VM that's been failed over, and then select **Re-Protect**.



3. Verify that the direction of protection, secondary to primary region, is already selected.
4. Review the **Resource group, Network, Storage, and Availability sets** information. Any resources marked as new are created as part of the reprotect operation.
5. Click **OK** to trigger a reprotect job. This job seeds the target site with the latest data. Then, it replicates the deltas to the primary region. The VM is now in a protected state.

Next steps

- After reprotecting, [learn how to](#) fail back to the primary region when it's available.
- [Learn more](#) about the reprotection flow.

Understanding Azure virtual machine usage

3/5/2019 • 7 minutes to read • [Edit Online](#)

By analyzing your Azure usage data, powerful consumption insights can be gained – insights that can enable better cost management and allocation throughout your organization. This document provides a deep dive into your Azure Compute consumption details. For more details on general Azure usage, navigate to [Understanding your bill](#).

Download your usage details

To begin, [download your usage details](#). The table below provides the definition and example values of usage for Virtual Machines deployed via the Azure Resource Manager. This document does not contain detailed information for VMs deployed via our classic model.

FIELDS	MEANING	EXAMPLE VALUES
Usage Date	The date when the resource was used.	"11/23/2017"
Meter ID	Identifies the top-level service for which this usage belongs to.	"Virtual Machines"
Meter Sub-Category	The billed meter identifier. <ul style="list-style-type: none">• For Compute Hour usage, there is a meter for each VM Size + OS (Windows, Non-Windows) + Region.• For Premium software usage, there is a meter for each software type. Most premium software images have different meters for each core size. For more information, visit the Compute Pricing Page.	"2005544f-659d-49c9-9094-8e0aea1be3a5"
Meter Name	This is specific for each service in Azure. For compute, it is always "Compute Hours".	"Compute Hours"
Meter Region	Identifies the location of the datacenter for certain services that are priced based on datacenter location.	"JA East"
Unit	Identifies the unit that the service is charged in. Compute resources are billed per hour.	"Hours"
Consumed	The amount of the resource that has been consumed for that day. For Compute, we bill for each minute the VM ran for a given hour (up to 6 decimals of accuracy).	"1", "0.5"

FIELDS	MEANING	EXAMPLE VALUES
Resource Location	Identifies the datacenter where the resource is running.	"JA East"
Consumed Service	The Azure platform service that you used.	"Microsoft.Compute"
Resource Group	The resource group in which the deployed resource is running in. For more information, see Azure Resource Manager overview .	"MyRG"
Instance ID	The identifier for the resource. The identifier contains the name you specify for the resource when it was created. For VMs, the Instance ID will contain the SubscriptionId, ResourceGroupName, and VMName (or scale set name for scale set usage).	<pre>"/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx/ resourceGroups/MyRG/providers/Microsoft.Compute/virtualMachines/MyVM1"</pre> <p>or</p> <pre>"/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx/ resourceGroups/MyRG/providers/Microsoft.Compute/virtualMachineScaleSets/MyVMSS1"</pre>
Tags	Tag you assign to the resource. Use tags to group billing records. Learn how to tag your Virtual Machines . This is available for Resource Manager VMs only.	<pre>"</pre> <pre>{"myDepartment":"RD","myUser":"myName"}</pre>
Additional Info	<p>Service-specific metadata. For VMs, we populate the following data in the additional info field:</p> <ul style="list-style-type: none"> • Image Type- specific image that you ran. Find the full list of supported strings below under Image Types. • Service Type: the size that you deployed. • VMName: name of your VM. This field is only populated for scale set VMs. If you need your VM Name for scale set VMs, you can find that in the Instance ID string above. • UsageType: This specifies the type of usage this represents. <ul style="list-style-type: none"> ◦ ComputeHR is the Compute Hour usage for the underlying VM, like Standard_D1_v2. ◦ ComputeHR_SW is the premium software charge if the VM is using premium software, like Microsoft R Server. 	<p>Virtual Machines</p> <pre>{"ImageType":"Canonical","ServiceType":"Standard_DS1_v2","VMName":"","UsageType":"ComputeHR"}</pre> <p>Virtual Machine Scale Sets</p> <pre>{"ImageType":"Canonical","ServiceType":"Standard_DS1_v2","VMName":"myVM1","UsageType":"ComputeHR"}</pre> <p>Premium Software</p> <pre>{"ImageType":"","ServiceType":"Standard_DS1_v2","VMName":"","UsageType":"ComputeHR_SW"}</pre>

Image Type

For some images in the Azure gallery, the image type is populated in the Additional Info field. This enables users to understand and track what they have deployed on their Virtual Machine. The following values that are populated in this field based on the image you have deployed:

- BitRock
- Canonical
- FreeBSD
- Open Logic
- Oracle
- SLES for SAP
- SQL Server 14 Preview on Windows Server 2012 R2 Preview
- SUSE
- SUSE Premium
- StorSimple Cloud Appliance
- Red Hat
- Red Hat for SAP Business Applications
- Red Hat for SAP HANA
- Windows Client BYOL
- Windows Server BYOL
- Windows Server Preview

Service Type

The service type field in the Additional Info field corresponds to the exact VM size you deployed. Premium storage VMs (SSD-based) and non-premium storage VMs (HDD-based) are priced the same. If you deploy an SSD-based size, like Standard_DS2_v2, you see the non-SSD size ('Standard_D2_v2 VM') in the Meter Sub-Category column and the SSD-size ('Standard_DS2_v2') in the Additional Info field.

Region Names

The region name populated in the Resource Location field in the usage details varies from the region name used in the Azure Resource Manager. Here is a mapping between the region values:

RESOURCE MANAGER REGION NAME	RESOURCE LOCATION IN USAGE DETAILS
australiaeast	AU East
australiasoutheast	AU Southeast
brazilsouth	BR South
CanadaCentral	CA Central
CanadaEast	CA East
CentralIndia	IN Central
centralus	Central US

RESOURCE MANAGER REGION NAME	RESOURCE LOCATION IN USAGE DETAILS
chinaeast	China East
chinanorth	China North
eastasia	East Asia
eastus	East US
eastus2	East US 2
GermanyCentral	DE Central
GermanyNortheast	DE Northeast
japaneast	JA East
japanwest	JA West
KoreaCentral	KR Central
KoreaSouth	KR South
northcentralus	North Central US
northeurope	North Europe
southcentralus	South Central US
southeastasia	Southeast Asia
SouthIndia	IN South
UKNorth	US North
uksouth	UK South
UKSouth2	UK South 2
ukwest	UK West
USDoDCentral	US DoD Central
USDoDEast	US DoD East
USGovArizona	USGov Arizona
usgoviowa	USGov Iowa
USGovTexas	USGov Texas

RESOURCE MANAGER REGION NAME	RESOURCE LOCATION IN USAGE DETAILS
usgovvirginia	USGov Virginia
westcentralus	US West Central
westeurope	West Europe
WestIndia	IN West
westus	West US
westus2	US West 2

Virtual machine usage FAQ

What resources are charged when deploying a VM?

VMs acquire costs for the VM itself, any premium software running on the VM, the storage account\managed disk associated with the VM, and the networking bandwidth transfers from the VM.

How can I tell if a VM is using Azure Hybrid Benefit in the Usage CSV?

If you deploy using the [Azure Hybrid Benefit](#), you are charged the Non-Windows VM rate since you are bringing your own license to the cloud. In your bill, you can distinguish which Resource Manager VMs are running Azure Hybrid Benefit because they have either "Windows_Server BYOL" or "Windows_Client BYOL" in the ImageType column.

How are Basic vs. Standard VM Types differentiated in the Usage CSV?

Both Basic and Standard A-Series VMs are offered. If you deploy a Basic VM, in the Meter Sub Category, it has the string "Basic." If you deploy a Standard A-Series VM, then the VM size appears as "A1 VM" since Standard is the default. To learn more about the differences between Basic and Standard, see the [Pricing Page](#).

What are ExtraSmall, Small, Medium, Large, and ExtraLarge sizes?

ExtraSmall - ExtraLarge are the legacy names for Standard_A0 – Standard_A4. In classic VM usage records, you might see this convention used if you have deployed these sizes.

What is the difference between Meter Region and Resource Location?

The Meter Region is associated with the meter. For some Azure services who use one price for all regions, the Meter Region field could be blank. However, since VMs have dedicated prices per region for Virtual Machines, this field is populated. Similarly, the Resource Location for Virtual Machines is the location where the VM is deployed. The Azure regions in both fields are the same, although they might have a different string convention for the region name.

Why is the ImageType value blank in the Additional Info field?

The ImageType field is only populated for a subset of images. If you did not deploy one of the images above, the ImageType is blank.

Why is the VMName blank in the Additional Info?

The VMName is only populated in the Additional Info field for VMs in a scale set. The InstanceID field contains the VM name for non-scale set VMs.

What does ComputeHR mean in the UsageType field in the Additional Info?

ComputeHR stands for Compute Hour which represents the usage event for the underlying infrastructure cost. If the UsageType is ComputeHR_SW, the usage event represents the premium software charge for the VM.

How do I know if I am charged for premium software?

When exploring which VM Image best fits your needs, be sure to check out the [Azure Marketplace](#). The image has the software plan rate. If you see "Free" for the rate, there is no additional cost for the software.

What is the difference between Microsoft.ClassicCompute and Microsoft.Compute in the Consumed service?

Microsoft.ClassicCompute represents classic resources deployed via the Azure Service Manager. If you deploy via the Resource Manager, then Microsoft.Compute is populated in the consumed service. Learn more about the [Azure Deployment models](#).

Why is the InstanceID field blank for my Virtual Machine usage?

If you deploy via the classic deployment model, the InstanceID string is not available.

Why are the tags for my VMs not flowing to the usage details?

Tags only flow to you the Usage CSV for Resource Manager VMs only. Classic resource tags are not available in the usage details.

How can the consumed quantity be more than 24 hours one day?

In the Classic model, billing for resources is aggregated at the Cloud Service level. If you have more than one VM in a Cloud Service that uses the same billing meter, your usage is aggregated together. VMs deployed via Resource Manager are billed at the VM level, so this aggregation will not apply.

Why is pricing not available for DS/FS/GS/LS sizes on the pricing page?

Premium storage capable VMs are billed at the same rate as non-premium storage capable VMs. Only your storage costs differ. Visit the [storage pricing page](#) for more information.

Next steps

To learn more about your usage details, see [Understand your bill for Microsoft Azure](#).

Common Azure CLI commands for managing Azure resources

9/24/2018 • 2 minutes to read • [Edit Online](#)

The Azure CLI allows you to create and manage your Azure resources on macOS, Linux, and Windows. This article details some of the most common commands to create and manage virtual machines (VMs).

This article requires the Azure CLI version 2.0.4 or later. Run `az --version` to find the version. If you need to upgrade, see [Install Azure CLI](#). You can also use [Cloud Shell](#) from your browser.

Basic Azure Resource Manager commands in Azure CLI

For more detailed help with specific command line switches and options, you can use the online command help and options by typing `az <command> <subcommand> --help`.

Create VMs

TASK	AZURE CLI COMMANDS
Create a resource group	<code>az group create --name myResourceGroup --location eastus</code>
Create a Linux VM	<code>az vm create --resource-group myResourceGroup --name myVM --image ubuntults</code>
Create a Windows VM	<code>az vm create --resource-group myResourceGroup --name myVM --image win2016datacenter</code>

Manage VM state

TASK	AZURE CLI COMMANDS
Start a VM	<code>az vm start --resource-group myResourceGroup --name myVM</code>
Stop a VM	<code>az vm stop --resource-group myResourceGroup --name myVM</code>
Deallocate a VM	<code>az vm deallocate --resource-group myResourceGroup --name myVM</code>
Restart a VM	<code>az vm restart --resource-group myResourceGroup --name myVM</code>
Redeploy a VM	<code>az vm redeploy --resource-group myResourceGroup --name myVM</code>
Delete a VM	<code>az vm delete --resource-group myResourceGroup --name myVM</code>

Get VM info

TASK	AZURE CLI COMMANDS
List VMs	<code>az vm list</code>
Get information about a VM	<code>az vm show --resource-group myResourceGroup --name myVM</code>
Get usage of VM resources	<code>az vm list-usage --location eastus</code>
Get all available VM sizes	<code>az vm list-sizes --location eastus</code>

Disks and images

TASK	AZURE CLI COMMANDS
Add a data disk to a VM	<code>az vm disk attach --resource-group myResourceGroup --vm-name myVM --disk myDataDisk --size-gb 128 --new</code>
Remove a data disk from a VM	<code>az vm disk detach --resource-group myResourceGroup --vm-name myVM --disk myDataDisk</code>
Resize a disk	<code>az disk update --resource-group myResourceGroup --name myDataDisk --size-gb 256</code>
Snapshot a disk	<code>az snapshot create --resource-group myResourceGroup --name mySnapshot --source myDataDisk</code>
Create image of a VM	<code>az image create --resource-group myResourceGroup --source myVM --name myImage</code>
Create VM from image	<code>az vm create --resource-group myResourceGroup --name myNewVM --image myImage</code>

Next steps

For additional examples of the CLI commands, see the [Create and Manage Linux VMs with the Azure CLI](#) tutorial.

Resize a Linux virtual machine using Azure CLI

2/5/2019 • 2 minutes to read • [Edit Online](#)

After you provision a virtual machine (VM), you can scale the VM up or down by changing the [VM size](#). In some cases, you must deallocate the VM first. You need to deallocate the VM if the desired size is not available on the hardware cluster that is hosting the VM. This article details how to resize a Linux VM with the Azure CLI.

Resize a VM

To resize a VM, you need the latest [Azure CLI](#) installed and logged in to an Azure account using [az login](#).

1. View the list of available VM sizes on the hardware cluster where the VM is hosted with [az vm list-vm-resize-options](#). The following example lists VM sizes for the VM named `myVM` in the resource group `myResourceGroup` region:

```
az vm list-vm-resize-options --resource-group myResourceGroup --name myVM --output table
```

2. If the desired VM size is listed, resize the VM with [az vm resize](#). The following example resizes the VM named `myVM` to the `Standard_DS3_v2` size:

```
az vm resize --resource-group myResourceGroup --name myVM --size Standard_DS3_v2
```

The VM restarts during this process. After the restart, your existing OS and data disks are remapped. Anything on the temporary disk is lost.

3. If the desired VM size is not listed, you need to first deallocate the VM with [az vm deallocate](#). This process allows the VM to then be resized to any size available that the region supports and then started. The following steps deallocate, resize, and then start the VM named `myVM` in the resource group named `myResourceGroup`:

```
az vm deallocate --resource-group myResourceGroup --name myVM
az vm resize --resource-group myResourceGroup --name myVM --size Standard_DS3_v2
az vm start --resource-group myResourceGroup --name myVM
```

WARNING

Deallocating the VM also releases any dynamic IP addresses assigned to the VM. The OS and data disks are not affected.

Next steps

For additional scalability, run multiple VM instances and scale out. For more information, see [Automatically scale Linux machines in a Virtual Machine Scale Set](#).

Change the OS disk used by an Azure VM using the CLI

2/4/2019 • 2 minutes to read • [Edit Online](#)

If you have an existing VM, but you want to swap the disk for a backup disk or another OS disk, you can use the Azure CLI to swap the OS disks. You don't have to delete and recreate the VM. You can even use a managed disk in another resource group, as long as it isn't already in use.

The VM does need to be stopped\deallocated, then the resource ID of the managed disk can be replaced with the resource ID of a different managed disk.

Make sure that the VM size and storage type are compatible with the disk you want to attach. For example, if the disk you want to use is in Premium Storage, then the VM needs to be capable of Premium Storage (like a DS-series size).

This article requires Azure CLI version 2.0.25 or greater. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Use [az disk list](#) to get a list of the disks in your resource group.

```
az disk list \
  -g myResourceGroupDisk \
  --query '[*].{diskId:id}' \
  --output table
```

Use [az vm stop](#) to stop\deallocate the VM before swapping the disks.

```
az vm stop \
  -n myVM \
  -g myResourceGroup
```

Use [az vm update](#) with the full resource ID of the new disk for the `--osdisk` parameter

```
az vm update \
  -g myResourceGroup \
  -n myVM \
  --os-disk /subscriptions/<subscription ID>/resourceGroups/swap/providers/Microsoft.Compute/disks/myDisk
```

Restart the VM using [az vm start](#).

```
az vm start \
  -n myVM \
  -g myResourceGroup
```

Next steps

To create a copy of a disk, see [Snapshot a disk](#).

Time sync for Linux VMs in Azure

5/21/2019 • 6 minutes to read • [Edit Online](#)

Time sync is important for security and event correlation. Sometimes it is used for distributed transactions implementation. Time accuracy between multiple computer systems is achieved through synchronization. Synchronization can be affected by multiple things, including reboots and network traffic between the time source and the computer fetching the time.

Azure is backed by infrastructure running Windows Server 2016. Windows Server 2016 has improved algorithms used to correct time and condition the local clock to synchronize with UTC. The Windows Server 2016 Accurate Time feature greatly improved how the VMCTimeSync service that governs VMs with the host for accurate time. Improvements include more accurate initial time on VM start or VM restore and interrupt latency correction.

NOTE

For a quick overview of Windows Time service, take a look at this [high-level overview video](#).

For more information, see [Accurate time for Windows Server 2016](#).

Overview

Accuracy for a computer clock is gauged on how close the computer clock is to the Coordinated Universal Time (UTC) time standard. UTC is defined by a multinational sample of precise atomic clocks that can only be off by one second in 300 years. But, reading UTC directly requires specialized hardware. Instead, time servers are synced to UTC and are accessed from other computers to provide scalability and robustness. Every computer has time synchronization service running that knows what time servers to use and periodically checks if computer clock needs to be corrected and adjusts time if needed.

Azure hosts are synchronized to internal Microsoft time servers that take their time from Microsoft-owned Stratum 1 devices, with GPS antennas. Virtual machines in Azure can either depend on their host to pass the accurate time (*host time*) on to the VM or the VM can directly get time from a time server, or a combination of both.

On stand-alone hardware, the Linux OS only reads the host hardware clock on boot. After that, the clock is maintained using the interrupt timer in the Linux kernel. In this configuration, the clock will drift over time. In newer Linux distributions on Azure, VMs can use the VMCTimeSync provider, included in the Linux integration services (LIS), to query for clock updates from the host more frequently.

Virtual machine interactions with the host can also affect the clock. During [memory preserving maintenance](#), VMs are paused for up to 30 seconds. For example, before maintenance begins the VM clock shows 10:00:00 AM and lasts 28 seconds. After the VM resumes, the clock on the VM would still show 10:00:00 AM, which would be 28 seconds off. To correct for this, the VMCTimeSync service monitors what is happening on the host and prompts for changes to happen on the VMs to compensate.

Without time synchronization working, the clock on the VM would accumulate errors. When there is only one VM, the effect might not be significant unless the workload requires highly accurate timekeeping. But in most cases, we have multiple, interconnected VMs that use time to track transactions and the time needs to be consistent throughout the entire deployment. When time between VMs is different, you could see the following effects:

- Authentication will fail. Security protocols like Kerberos or certificate-dependent technology rely on time being consistent across the systems.
- It's very hard to figure out what has happened in a system if logs (or other data) don't agree on time. The same

event would look like it occurred at different times, making correlation difficult.

- If clock is off, the billing could be calculated incorrectly.

Configuration options

There are generally three ways to configure time sync for your Linux VMs hosted in Azure:

- The default configuration for Azure Marketplace images uses both NTP time and VMICTimeSync host-time.
- Host-only using VMICTimeSync.
- Use another, external time server with or without using VMICTimeSync host-time.

Use the default

By default, most Azure Marketplace images for Linux are configured to sync from two sources:

- NTP as primary, which gets time from an NTP server. For example, Ubuntu 16.04 LTS Marketplace images use ntp.ubuntu.com.
- The VMICTimeSync service as secondary, used to communicate the host time to the VMs and make corrections after the VM is paused for maintenance. Azure hosts use Microsoft-owned Stratum 1 devices to keep accurate time.

In newer Linux distributions, the VMICTimeSync service uses the precision time protocol (PTP), but earlier distributions may not support PTP and will fall-back to NTP for getting time from the host.

To confirm NTP is synchronizing correctly, run the `ntpq -p` command.

Host-only

Because NTP servers like time.windows.com and ntp.ubuntu.com are public, syncing time with them requires sending traffic over the internet. Varying packet delays can negatively affect quality of the time sync. Removing NTP by switching to host-only sync can sometimes improve your time sync results.

Switching to host-only time sync makes sense if you experience time sync issues using the default configuration. Try out the host-only sync to see if that would improve the time sync on your VM.

External time server

If you have specific time sync requirements, there is also an option of using external time servers. External time servers can provide specific time, which can be useful for test scenarios, ensuring time uniformity with machines hosted in non-Microsoft datacenters, or handling leap seconds in a special way.

You can combine an external time server with the VMICTimeSync service to provide results similar to the default configuration. Combining an external time server with VMICTimeSync is the best option for dealing with issues that can be caused when VMs are paused for maintenance.

Tools and resources

There are some basic commands for checking your time synchronization configuration. Documentation for Linux distribution will have more details on the best way to configure time synchronization for that distribution.

Integration services

Check to see if the integration service (hv_utils) is loaded.

```
lsmod | grep hv_utils
```

You should see something similar to this:

```
hv_utils          24418  0
hv_vmbus         397185  7 hv_balloon,hyperv_keyboard,hv_netvsc,hid_hyperv,hv_utils,hyperv_fb,hv_storvsc
```

See if the Hyper-V integration services daemon is running.

```
ps -ef | grep hv
```

You should see something similar to this:

```
root      229      2  0 17:52 ?      00:00:00 [hv_vmbus_con]
root      391      2  0 17:52 ?      00:00:00 [hv_balloon]
```

Check for PTP

With newer versions of Linux, a Precision Time Protocol (PTP) clock source is available as part of the VMICTimeSync provider. On older versions of Red Hat Enterprise Linux or CentOS 7.x the [Linux Integration Services](#) can be downloaded and used to install the updated driver. When using PTP, the Linux device will be of the form /dev/ptpx.

See which PTP clock sources are available.

```
ls /sys/class/ptp
```

In this example, the value returned is *ptp0*, so we use that to check the clock name. To verify the device, check the clock name.

```
cat /sys/class/ptp/ptp0/clock_name
```

This should return **hyperv**.

chrony

On Red Hat Enterprise Linux and CentOS 7.x, [chrony](#) configured to use a PTP source clock. The Network Time Protocol daemon (*ntpd*) doesn't support PTP sources, so using **chrony** is recommended. To enable PTP, update **chrony.conf**.

```
refclock PHC /dev/ptp0 poll 3 dpoll -2 offset 0
```

For more information on Red Hat and NTP, see [Configure NTP](#).

For more information on chrony, see [Using chrony](#).

If both chrony and TimeSync sources are enabled simultaneously, you can mark one as **prefer** which sets the other source as a backup. Because NTP services do not update the clock for large skews except after a long period, the VMICTimeSync will recover the clock from paused VM events far more quickly than NTP-based tools alone.

systemd

On Ubuntu and SUSE time sync is configured using [systemd](#). For more information on Ubuntu, see [Time Synchronization](#). For more information on SUSE, see Section 4.5.8 in [SUSE Linux Enterprise Server 12 SP3 Release Notes](#).

Next steps

For more information, see [Accurate time for Windows Server 2016](#).

How to tag a Linux virtual machine in Azure

9/24/2018 • 2 minutes to read • [Edit Online](#)

This article describes different ways to tag a Linux virtual machine in Azure through the Resource Manager deployment model. Tags are user-defined key/value pairs which can be placed directly on a resource or a resource group. Azure currently supports up to 15 tags per resource and resource group. Tags may be placed on a resource at the time of creation or added to an existing resource. Please note, tags are supported for resources created via the Resource Manager deployment model only.

Tagging a Virtual Machine through Templates

First, let's look at tagging through templates. [This template](#) places tags on the following resources: Compute (Virtual Machine), Storage (Storage Account), and Network (Public IP Address, Virtual Network, and Network Interface). This template is for a Windows VM but can be adapted for Linux VMs.

Click the **Deploy to Azure** button from the [template link](#). This will navigate to the [Azure portal](#) where you can deploy this template.

Simple deployment of a VM with Tags



This template includes the following tags: *Department*, *Application*, and *Created By*. You can add/edit these tags directly in the template if you would like different tag names.

```
"apiVersion": "2015-05-01-preview",
"type": "Microsoft.Compute/virtualMachines",
"name": "[variables('vmName')]",
"location": "[variables('location')]",
"tags": {
    "Department": "[parameters('departmentName')]",
    "Application": "[parameters('applicationName')]",
    "Created By": "[parameters('createdBy')]"
},
```

As you can see, the tags are defined as key/value pairs, separated by a colon (:). The tags must be defined in this format:

```
"tags": {
    "Key1" : "Value1",
    "Key2" : "Value2"
}
```

Save the template file after you finish editing it with the tags of your choice.

Next, in the **Edit Parameters** section, you can fill out the values for your tags.

DEPARTMENTNAME (string) ⓘ

APPLICATIONNAME (string) ⓘ

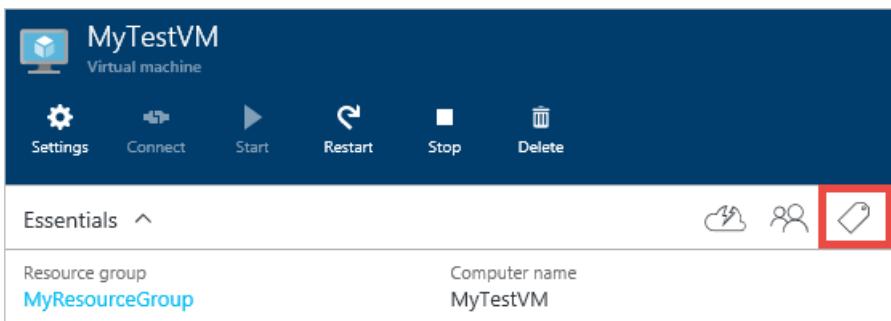
CREATEDBY (string) ⓘ

Click **Create** to deploy this template with your tag values.

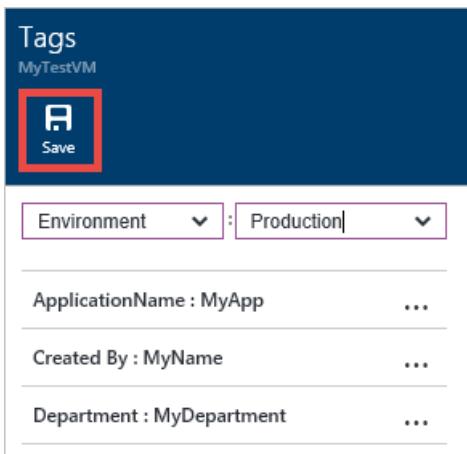
Tagging through the Portal

After creating your resources with tags, you can view, add, and delete tags in the portal.

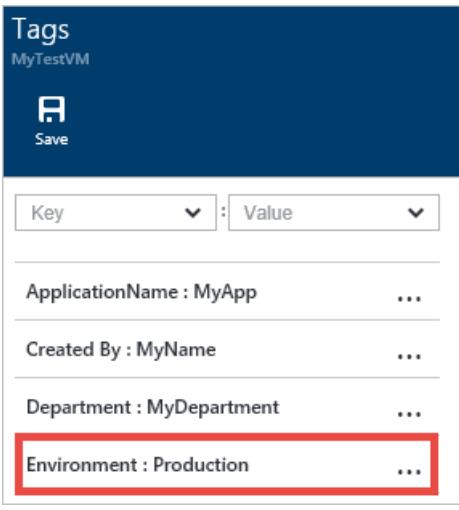
Select the tags icon to view your tags:



Add a new tag through the portal by defining your own Key/Value pair, and save it.



Your new tag should now appear in the list of tags for your resource.



Tagging with Azure CLI

To begin, you need the latest [Azure CLI](#) installed and logged in to an Azure account using `az login`.

You can view all properties for a given Virtual Machine, including the tags, using this command:

```
az vm show --resource-group MyResourceGroup --name MyTestVM
```

To add a new VM tag through the Azure CLI, you can use the `azure vm update` command along with the tag parameter `--set`:

```
az vm update \
--resource-group MyResourceGroup \
--name MyTestVM \
--set tags.myNewTagName1=myNewTagValue1 tags.myNewTagName2=myNewTagValue2
```

To remove tags, you can use the `--remove` parameter in the `azure vm update` command.

```
az vm update --resource-group MyResourceGroup --name MyTestVM --remove tags.myNewTagName1
```

Now that we have applied tags to our resources Azure CLI and the Portal, let's take a look at the usage details to see the tags in the billing portal.

Viewing your tags in the usage details

Tags placed on Compute, Network, and Storage resources in the Resource Manager deployment model will be populated in your usage details in the [billing portal](#).

Click on **Download usage details** to view the usage details in your subscription.

NEXT BILL (ESTIMATED):

\$0.00

DATE PURCHASED
6/24/2014

CURRENT BILLING PERIOD
5/24/2015 - 6/23/2015

 [Download usage details](#)

-  [Contact Microsoft Support](#)
-  [Edit subscription details](#)
-  [Change subscription address](#)
-  [Partner Information](#)
-  [Cancel Subscription](#)

Select your billing statement and the **Version 2** usage details:

Click here to [Understand Your Bill](#).

Current period	View Current Statement	Download Usage	Version 2 - Preview
7/24/2014 - 8/23/2014	Download Usage	Version 1	

From the usage details, you can see all of the tags in the **Tags** column:

Consumed Service	Resource Group	Instance Id	Tags
"Microsoft.Compute"	"MYRESOURCEGROUP"	"/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/MYRESOURCEGROUP/providers/Microsoft.Compute/virtualMachines/MyWindowsVM"	"[{"Department":"MyDepartment","Application":"MyApp1","Created By":"MyName","Type":"Virtual Machine","Environment":"Production","Location":"MyLocation"}]"
"Microsoft.Storage"	"myresourcegroup"	"/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/myresourcegroup/providers/Microsoft.Storage/storageAccounts/mystorageaccount"	"[{"Application":"MyApp1","Created By":"MyName","Department":"MyDepartment","Type":"Storage Account"}]"
"Microsoft.Network"	"MyResourceGroup"	"/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/MyResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP"	"[{"Department":"MyDepartment","Application":"MyApp1","Created By":"MyName","Type":"Public IP"}]"

By analyzing these tags along with usage, organizations will be able to gain new insights into their consumption data.

Next steps

- To learn more about tagging your Azure resources, see [Azure Resource Manager Overview](#) and [Using Tags to organize your Azure Resources](#).
- To see how tags can help you manage your use of Azure resources, see [Understanding your Azure Bill](#) and [Gain insights into your Microsoft Azure resource consumption](#).

Run scripts in your Linux VM

6/28/2019 • 2 minutes to read • [Edit Online](#)

To automate tasks or troubleshoot issues, you may need to run commands in a VM. The following article gives a brief overview of the features that are available to run scripts and commands within your VMs.

Custom Script Extension

The [Custom Script Extension](#) is primarily used for post deployment configuration and software installation.

- Download and run scripts in Azure virtual machines.
- Can be run using Azure Resource Manager templates, Azure CLI, REST API, PowerShell, or Azure portal.
- Script files can be downloaded from Azure storage or GitHub, or provided from your PC when run from the Azure portal.
- Run PowerShell script in Windows machines and Bash script in Linux machines.
- Useful for post deployment configuration, software installation, and other configuration or management tasks.

Run command

The [Run Command](#) feature enables virtual machine and application management and troubleshooting using scripts, and is available even when the machine is not reachable, for example if the guest firewall doesn't have the RDP or SSH port open.

- Run scripts in Azure virtual machines.
- Can be run using [Azure portal](#), [REST API](#), [Azure CLI](#), or [PowerShell](#)
- Quickly run a script and view output and repeat as needed in the Azure portal.
- Script can be typed directly or you can run one of the built-in scripts.
- Run PowerShell script in Windows machines and Bash script in Linux machines.
- Useful for virtual machine and application management and for running scripts in virtual machines that are unreachable.

Hybrid Runbook Worker

The [Hybrid Runbook Worker](#) provides general machine, application, and environment management with user's custom scripts stored in an Automation account.

- Run scripts in Azure and non-Azure machines.
- Can be run using Azure portal, Azure CLI, REST API, PowerShell, webhook.
- Scripts stored and managed in an Automation Account.
- Run PowerShell, PowerShell Workflow, Python, or Graphical runbooks
- No time limit on script run time.
- Multiple scripts can run concurrently.
- Full script output is returned and stored.
- Job history available for 90 days.
- Scripts can run as Local System or with user-supplied credentials.
- Requires [manual installation](#)

Serial console

The [Serial console](#) provides direct access to a VM, similar to having a keyboard connected to the VM.

- Run commands in Azure virtual machines.
- Can be run using a text-based console to the machine in the Azure portal.
- Login to the machine with a local user account.
- Useful when access to the virtual machine is needed regardless of the machine's network or operating system state.

Next steps

Learn more about the different features that are available to run scripts and commands within your VMs.

- [Custom Script Extension](#)
- [Run Command](#)
- [Hybrid Runbook Worker](#)
- [Serial console](#)

Use the Azure Custom Script Extension Version 2 with Linux virtual machines

4/25/2019 • 11 minutes to read • [Edit Online](#)

The Custom Script Extension Version 2 downloads and runs scripts on Azure virtual machines. This extension is useful for post-deployment configuration, software installation, or any other configuration/management task. You can download scripts from Azure Storage or another accessible internet location, or you can provide them to the extension runtime.

The Custom Script Extension integrates with Azure Resource Manager templates. You can also run it by using Azure CLI, PowerShell, or the Azure Virtual Machines REST API.

This article details how to use the Custom Script Extension from Azure CLI, and how to run the extension by using an Azure Resource Manager template. This article also provides troubleshooting steps for Linux systems.

There are two Linux Custom Script Extensions:

- Version 1 - Microsoft.OSTCExtensions.CustomScriptForLinux
- Version 2 - Microsoft.Azure.Extensions.CustomScript

Please switch new and existing deployments to use the new version 2 instead. The new version is intended to be a drop-in replacement. Therefore, the migration is as easy as changing the name and version, you do not need to change your extension configuration.

Operating System

The Custom Script Extension for Linux will run on the extension supported extension OS's, for more information, see this [article](#).

Script Location

You can use the extension to use your Azure Blob storage credentials, to access Azure Blob storage. Alternatively, the script location can be any where, as long as the VM can route to that end point, such as GitHub, internal file server etc.

Internet Connectivity

If you need to download a script externally such as GitHub or Azure Storage, then additional firewall/Network Security Group ports need to be opened. For example if your script is located in Azure Storage, you can allow access using Azure NSG Service Tags for [Storage](#).

If your script is on a local server, then you may still need additional firewall/Network Security Group ports need to be opened.

Tips and Tricks

- The highest failure rate for this extension is due to syntax errors in the script, test the script runs without error, and also put in additional logging into the script to make it easier to find where it failed.
- Write scripts that are idempotent, so if they get run again more than once accidentally, it will not cause system changes.
- Ensure the scripts do not require user input when they run.
- There is 90 mins allowed for the script to run, anything longer will result in a failed provision of the extension.
- Do not put reboots inside the script, this will cause issues with other extensions that are being installed, and post reboot, the extension will not continue after the restart.

- If you have a script that will cause a reboot, then install applications and run scripts etc. You should schedule the reboot using a Cron job, or using tools such as DSC, or Chef, Puppet extensions.
- The extension will only run a script once, if you want to run a script on every boot, then you can use [cloud-init image](#) and use a [Scripts Per Boot](#) module. Alternatively, you can use the script to create a Systemd service unit.
- If you want to schedule when a script will run, you should use the extension to create a Cron job.
- When the script is running, you will only see a 'transitioning' extension status from the Azure portal or CLI. If you want more frequent status updates of a running script, you will need to create your own solution.
- Custom Script extension does not natively support proxy servers, however you can use a file transfer tool that supports proxy servers within your script, such as *Curl*.
- Be aware of non default directory locations that your scripts or commands may rely on, have logic to handle this.

Extension schema

The Custom Script Extension configuration specifies things like script location and the command to be run. You can store this configuration in configuration files, specify it on the command line, or specify it in an Azure Resource Manager template.

You can store sensitive data in a protected configuration, which is encrypted and only decrypted inside the virtual machine. The protected configuration is useful when the execution command includes secrets such as a password.

These items should be treated as sensitive data and specified in the extensions protected setting configuration. Azure VM extension protected setting data is encrypted, and only decrypted on the target virtual machine.

```
{
  "name": "config-app",
  "type": "Microsoft.Compute/virtualMachines/extensions",
  "location": "[resourceGroup().location]",
  "apiVersion": "2015-06-15",
  "dependsOn": [
    "[concat('Microsoft.Compute/virtualMachines/', concat(variables('vmName'),copyindex()))]"
  ],
  "tags": {
    "displayName": "config-app"
  },
  "properties": {
    "publisher": "Microsoft.Azure.Extensions",
    "type": "CustomScript",
    "typeHandlerVersion": "2.0",
    "autoUpgradeMinorVersion": true,
    "settings": {
      "skipDOS2Unix":false,
      "timestamp":123456789
    },
    "protectedSettings": {
      "commandToExecute": "<command-to-execute>",
      "script": "<base64-script-to-execute>",
      "storageAccountName": "<storage-account-name>",
      "storageAccountKey": "<storage-account-key>",
      "fileUris": ["https://.."]
    }
  }
}
```

Property values

NAME	VALUE / EXAMPLE	DATA TYPE

NAME	VALUE / EXAMPLE	DATA TYPE
apiVersion	2015-06-15	date
publisher	Microsoft.Compute.Extensions	string
type	CustomScript	string
typeHandlerVersion	2.0	int
fileUris (e.g.)	https://github.com/MyProject/Archive/MyPythonScript.py	array
commandToExecute (e.g.)	python MyPythonScript.py <my-param1>	string
script	lyEvYmluL3NoCmVjaG8gIIVwZGF0aW5nIHBhY2thZ2VzIC4uLiIKYXB0IHVwZGF0ZQphcHQgdXBncmFkZSAteQo=	string
skipDos2Unix (e.g.)	false	boolean
timestamp (e.g.)	123456789	32-bit integer
storageAccountName (e.g.)	examplestorageacct	string
storageAccountKey (e.g.)	TmJK/1N3AbAZ3q/+hOXoi/I73zOqsaxX Dhqa9Y83/v5UpXQp2DQIBuv2Tifp60c E/OaHsJZmQZ7teQfczQj8hg==	string

Property value details

- `skipDos2Unix` : (optional, boolean) skip dos2unix conversion of script-based file URLs or script.
- `timestamp` (optional, 32-bit integer) use this field only to trigger a re-run of the script by changing value of this field. Any integer value is acceptable; it must only be different than the previous value.
 - `commandToExecute` : (**required** if script not set, string) the entry point script to execute. Use this field instead if your command contains secrets such as passwords.
- `script` : (**required** if commandToExecute not set, string)a base64 encoded (and optionally gzip'ed) script executed by /bin/sh.
- `fileUris` : (optional, string array) the URLs for file(s) to be downloaded.
- `storageAccountName` : (optional, string) the name of storage account. If you specify storage credentials, all `fileUris` must be URLs for Azure Blobs.
- `storageAccountKey` : (optional, string) the access key of storage account

The following values can be set in either public or protected settings, the extension will reject any configuration where the values below are set in both public and protected settings.

- `commandToExecute`
- `script`
- `fileUris`

Using public settings maybe useful for debugging, but it is strongly recommended that you use protected settings.

Public settings are sent in clear text to the VM where the script will be executed. Protected settings are encrypted

using a key known only to the Azure and the VM. The settings are saved to the VM as they were sent, i.e. if the settings were encrypted they are saved encrypted on the VM. The certificate used to decrypt the encrypted values is stored on the VM, and used to decrypt settings (if necessary) at runtime.

Property: skipDos2Unix

The default value is false, which means dos2unix conversion **is** executed.

The previous version of CustomScript, Microsoft.OSTCExtensions.CustomScriptForLinux, would automatically convert DOS files to UNIX files by translating `\r\n` to `\n`. This translation still exists, and is on by default. This conversion is applied to all files downloaded from fileUris or the script setting based on any of the following criteria.

- If the extension is one of `.sh`, `.txt`, `.py`, or `.pl` it will be converted. The script setting will always match this criteria because it is assumed to be a script executed with `/bin/sh`, and is saved as `script.sh` on the VM.
- If the file starts with `#!`.

The dos2unix conversion can be skipped by setting the skipDos2Unix to true.

```
{
  "fileUris": ["<url>"],
  "commandToExecute": "<command-to-execute>",
  "skipDos2Unix": true
}
```

Property: script

CustomScript supports execution of a user-defined script. The script settings to combine commandToExecute and fileUris into a single setting. Instead of having to setup a file for download from Azure storage or GitHub gist, you can simply encode the script as a setting. Script can be used to replaced commandToExecute and fileUris.

The script **must** be base64 encoded. The script can **optionally** be gzip'ed. The script setting can be used in public or protected settings. The maximum size of the script parameter's data is 256 KB. If the script exceeds this size it will not be executed.

For example, given the following script saved to the file `/script.sh/`.

```
#!/bin/sh
echo "Updating packages ..."
apt update
apt upgrade -y
```

The correct CustomScript script setting would be constructed by taking the output of the following command.

```
cat script.sh | base64 -w0
```

```
{
  "script": "IyEvYmluL3NoCmVjaG8gIlVwZGF0aW5nIHBhY2thZ2VzIC4uLiIKYXB0IHVwZGF0ZQphcHQgdXBncmFkZSAtE0o="
}
```

The script can optionally be gzip'ed to further reduce size (in most cases). (CustomScript auto-detects the use of gzip compression.)

```
cat script | gzip -9 | base64 -w 0
```

CustomScript uses the following algorithm to execute a script.

1. assert the length of the script's value does not exceed 256 KB.
2. base64 decode the script's value
3. attempt to gunzip the base64 decoded value
4. write the decoded (and optionally decompressed) value to disk (/var/lib/waagent/custom-script/#/script.sh)
5. execute the script using _/bin/sh -c /var/lib/waagent/custom-script/#/script.sh.

Template deployment

Azure VM extensions can be deployed with Azure Resource Manager templates. The JSON schema detailed in the previous section can be used in an Azure Resource Manager template to run the Custom Script Extension during an Azure Resource Manager template deployment. A sample template that includes the Custom Script Extension can be found here, [GitHub](#).

```
{
  "name": "config-app",
  "type": "extensions",
  "location": "[resourceGroup().location]",
  "apiVersion": "2015-06-15",
  "dependsOn": [
    "[concat('Microsoft.Compute/virtualMachines/', concat(variables('vmName'),copyindex()))]"
  ],
  "tags": {
    "displayName": "config-app"
  },
  "properties": {
    "publisher": "Microsoft.Azure.Extensions",
    "type": "CustomScript",
    "typeHandlerVersion": "2.0",
    "autoUpgradeMinorVersion": true,
    "settings": {
    },
    "protectedSettings": {
      "commandToExecute": "sh hello.sh <param2>",
      "fileUris": ["https://github.com/MyProject/Archive/hello.sh"]
    }
  }
}
```

NOTE

These property names are case-sensitive. To avoid deployment problems, use the names as shown here.

Azure CLI

When you're using Azure CLI to run the Custom Script Extension, create a configuration file or files. At a minimum, you must have 'commandToExecute'.

```
az vm extension set \
--resource-group myResourceGroup \
--vm-name myVM --name customScript \
--publisher Microsoft.Azure.Extensions \
--protected-settings ./script-config.json
```

Optionally, you can specify the settings in the command as a JSON formatted string. This allows the configuration to be specified during execution and without a separate configuration file.

```
az vm extension set \
--resource-group exttest \
--vm-name exttest \
--name customScript \
--publisher Microsoft.Azure.Extensions \
--protected-settings '{"fileUris": ["https://raw.githubusercontent.com/Microsoft/dotnet-core-sample-templates/master/dotnet-core-music-linux/scripts/config-music.sh"],"commandToExecute": "./config-music.sh"}'
```

Azure CLI examples

Public configuration with script file

```
{
  "fileUris": ["https://raw.githubusercontent.com/Microsoft/dotnet-core-sample-templates/master/dotnet-core-music-linux/scripts/config-music.sh"],
  "commandToExecute": "./config-music.sh"
}
```

Azure CLI command:

```
az vm extension set \
--resource-group myResourceGroup \
--vm-name myVM --name customScript \
--publisher Microsoft.Azure.Extensions \
--settings ./script-config.json
```

Public configuration with no script file

```
{
  "commandToExecute": "apt-get -y update && apt-get install -y apache2"
}
```

Azure CLI command:

```
az vm extension set \
--resource-group myResourceGroup \
--vm-name myVM --name customScript \
--publisher Microsoft.Azure.Extensions \
--settings ./script-config.json
```

Public and protected configuration files

You use a public configuration file to specify the script file URI. You use a protected configuration file to specify the command to be run.

Public configuration file:

```
{
  "fileUris": ["https://raw.githubusercontent.com/Microsoft/dotnet-core-sample-templates/master/dotnet-core-music-linux/scripts/config-music.sh"]
}
```

Protected configuration file:

```
{
  "commandToExecute": "./config-music.sh <param1>"
}
```

Azure CLI command:

```
az vm extension set \
--resource-group myResourceGroup \
--vm-name myVM \
--name customScript \
--publisher Microsoft.Azure.Extensions \
--settings ./script-config.json \
--protected-settings ./protected-config.json
```

Troubleshooting

When the Custom Script Extension runs, the script is created or downloaded into a directory that's similar to the following example. The command output is also saved into this directory in `stdout` and `stderr` files.

```
/var/lib/waagent/custom-script/download/0/
```

To troubleshoot, first check the Linux Agent Log, ensure the extension ran, check:

```
/var/log/waagent.log
```

You should look for the extension execution, it will look something like:

```
2018/04/26 17:47:22.110231 INFO [Microsoft.Azure.Extensions.customScript-2.0.6] [Enable] current handler state is: notinstalled
2018/04/26 17:47:22.306407 INFO Event: name=Microsoft.Azure.Extensions.customScript, op=Download, message=Download succeeded, duration=167
2018/04/26 17:47:22.339958 INFO [Microsoft.Azure.Extensions.customScript-2.0.6] Initialize extension directory
2018/04/26 17:47:22.368293 INFO [Microsoft.Azure.Extensions.customScript-2.0.6] Update settings file: 0.settings
2018/04/26 17:47:22.394482 INFO [Microsoft.Azure.Extensions.customScript-2.0.6] Install extension [bin/custom-script-shim install]
2018/04/26 17:47:23.432774 INFO Event: name=Microsoft.Azure.Extensions.customScript, op=Install, message=Launch command succeeded: bin/custom-script-shim install, duration=1007
2018/04/26 17:47:23.476151 INFO [Microsoft.Azure.Extensions.customScript-2.0.6] Enable extension [bin/custom-script-shim enable]
2018/04/26 17:47:24.516444 INFO Event: name=Microsoft.Azure.Extensions.customScript, op=Enable, message=Launch command succeeded: bin/custom-sc
```

Some points to note:

1. Enable is when the command starts running.
2. Download relates to the downloading of the CustomScript extension package from Azure, not the script files specified in fileUris.

The Azure Script Extension produces a log, which you can find here:

```
/var/log/azure/custom-script/handler.log
```

You should look for the individual execution, it will look something like:

```

time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event=start
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event=pre-check
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="comparing seqnum"
path=mrseq
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="seqnum saved"
path=mrseq
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="reading configuration"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="read configuration"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="validating json schema"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="json schema valid"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="parsing configuration json"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="parsed configuration json"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="validating configuration logically"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="validated configuration"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="creating output directory" path=/var/lib/waagent/custom-script/download/0
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="created output directory"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 files=1
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 file=0 event="download start"
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 file=0 event="download complete" output=/var/lib/waagent/custom-script/download/0
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="executing command" output=/var/lib/waagent/custom-script/download/0
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="executing protected commandToExecute" output=/var/lib/waagent/custom-script/download/0
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event="executed command" output=/var/lib/waagent/custom-script/download/0
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event=enabled
time=2018-04-26T17:47:23Z version=v2.0.6/git@1008306-clean operation=enable seq=0 event=end

```

Here you can see:

- The Enable command starting is this log
- The settings passed to the extension
- The extension downloading file and the result of that.
- The command being run and the result.

You can also retrieve the execution state of the Custom Script Extension by using Azure CLI:

```
az vm extension list -g myResourceGroup --vm-name myVM
```

The output looks like the following text:

```

info:  Executing command vm extension get
+ Looking up the VM "scripttst001"
data:  Publisher                           Name          Version  State
data:  -----                            -----        -----  -----
data:  Microsoft.Azure.Extensions  CustomScript      2.0     Succeeded
data:  Microsoft.OSTCExtensions    Microsoft.Insights.VMDiagnosticsSettings  2.3     Succeeded
info:  vm extension get command OK

```

Next steps

To see the code, current issues and versions, see [custom-script-extension-linux repo](#).

Run shell scripts in your Linux VM with Run Command

6/28/2019 • 3 minutes to read • [Edit Online](#)

Run Command uses the VM agent to run shell scripts within an Azure Linux VM. These scripts can be used for general machine or application management, and can be used to quickly diagnose and remediate VM access and network issues and get the VM back to a good state.

Benefits

There are multiple options that can be used to access your virtual machines. Run Command can run scripts on your virtual machines remotely using the VM agent. Run Command can be used through the Azure portal, [REST API](#), or [Azure CLI](#) for Linux VMs.

This capability is useful in all scenarios where you want to run a script within a virtual machines, and is one of the only ways to troubleshoot and remediate a virtual machine that doesn't have the RDP or SSH port open due to improper network or administrative user configuration.

Restrictions

The following are a list of restrictions that are present when using Run Command.

- Output is limited to the last 4096 bytes
- The minimum time to run a script about 20 seconds
- Scripts run by default as elevated user on Linux
- One script at a time may run
- Scripts that prompt for information (interactive mode) are not supported.
- You cannot cancel a running script
- The maximum time a script can run is 90 minutes, after which it will time out
- Outbound connectivity from the VM is required to return the results of the script.

NOTE

To function correctly, the Run Command requires connectivity (port 443) to Azure public IP addresses. If the extension doesn't have access to these endpoints, the scripts may run successfully but not return the results. If you are blocking traffic on the virtual machine, you can use [service tags](#) to allow traffic to Azure public IP addresses by using the `AzureCloud` tag.

Azure CLI

The following is an example using the [az vm run-command](#) command to run a shell script on an Azure Linux VM.

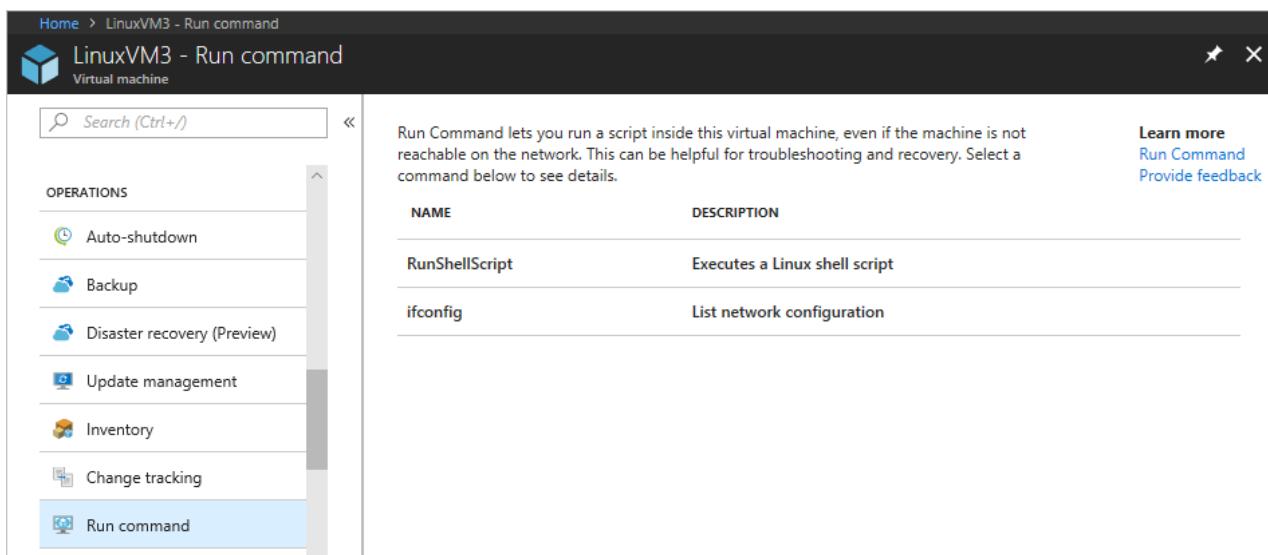
```
az vm run-command invoke -g myResourceGroup -n myVm --command-id RunShellScript --scripts "sudo apt-get update && sudo apt-get install -y nginx"
```

NOTE

To run commands as a different user, you can use `sudo -u` to specify a user account to use.

Azure portal

Navigate to a VM in [Azure](#) and select **Run command** under **OPERATIONS**. You are presented with a list of the available commands to run on the VM.



The screenshot shows the Azure portal interface for a virtual machine named "LinuxVM3". The left sidebar lists various operations, with "Run command" selected. The main pane displays a list of commands:

NAME	DESCRIPTION
RunShellScript	Executes a Linux shell script
ifconfig	List network configuration

At the top right, there are links to "Learn more", "Run Command", and "Provide feedback".

Choose a command to run. Some of the commands may have optional or required input parameters. For those commands the parameters are presented as text fields for you to provide the input values. For each command you can view the script that is being run by expanding **View script**. **RunShellScript** is different from the other commands as it allows you to provide your own custom script.

NOTE

The built-in commands are not editable.

Once the command is chosen, click **Run** to run the script. The script runs and when complete, returns the output and any errors in the output window. The following screenshot shows an example output from running the **ifconfig** command.

Run Command Script
ifconfig

Script execution complete

Details
Get the configuration of all network interfaces.

View script

Parameters

ARGUMENTS ⓘ
Default will be used

Run

Output

```
Enable succeeded:  
[stdout]  
eth0      Link encap:Ethernet  HWaddr 00:0d:3a:12:81:d1  
          inet  addr:10.0.0.7  Bcast:10.0.0.255  Mask:255.255.255.0  
          inet6 addr: fe80::20d:3aff:fe12:81d1/64 Scope:Link  
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
             RX packets:2134524 errors:0 dropped:0 overruns:0 frame:0  
             TX packets:1438287 errors:0 dropped:0 overruns:0 carrier:0  
             collisions:0 txqueuelen:1000  
             RX bytes:2208675468 (2.2 GB)  TX bytes:400069292 (400.0 MB)  
  
lo       Link encap:Local Loopback  
          inet  addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
             UP LOOPBACK RUNNING  MTU:65536  Metric:1  
             RX packets:160 errors:0 dropped:0 overruns:0 frame:0  
             TX packets:160 errors:0 dropped:0 overruns:0 carrier:0  
             collisions:0 txqueuelen:1000  
             RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)  
  
[stderr]
```

Available Commands

This table shows the list of commands available for Linux VMs. The **RunShellScript** command can be used to run any custom script you want.

NAME	DESCRIPTION
RunShellScript	Executes a Linux shell script.
ifconfig	Get the configuration of all network interfaces.

Limiting access to Run Command

Listing the run commands or showing the details of a command require the

`Microsoft.Compute/locations/runCommands/read` permission at the subscription level, which the built-in **Reader** role and higher have.

Running a command requires the `Microsoft.Compute/virtualMachines/runCommand/action` permission at the subscription level, which the **Virtual Machine Contributor** role and higher have.

You can use one of the **built-in** roles or create a **custom** role to use Run Command.

Next steps

See, [Run scripts in your Linux VM](#) to learn about other ways to run scripts and commands remotely in your VM.

Install and configure Remote Desktop to connect to a Linux VM in Azure

5/6/2019 • 4 minutes to read • [Edit Online](#)

Linux virtual machines (VMs) in Azure are usually managed from the command line using a secure shell (SSH) connection. When new to Linux, or for quick troubleshooting scenarios, the use of remote desktop may be easier. This article details how to install and configure a desktop environment ([xfce](#)) and remote desktop ([xrdp](#)) for your Linux VM using the Resource Manager deployment model.

Prerequisites

This article requires an existing Ubuntu 16.04 LTS VM in Azure. If you need to create a VM, use one of the following methods:

- The [Azure CLI](#)
- The [Azure portal](#)

Install a desktop environment on your Linux VM

Most Linux VMs in Azure do not have a desktop environment installed by default. Linux VMs are commonly managed using SSH connections rather than a desktop environment. There are various desktop environments in Linux that you can choose. Depending on your choice of desktop environment, it may consume one to 2 GB of disk space, and take 5 to 10 minutes to install and configure all the required packages.

The following example installs the lightweight [xfce4](#) desktop environment on an Ubuntu 16.04 LTS VM. Commands for other distributions vary slightly (use `yum` to install on Red Hat Enterprise Linux and configure appropriate `selinux` rules, or use `zypper` to install on SUSE, for example).

First, SSH to your VM. The following example connects to the VM named `myvm.westus.cloudapp.azure.com` with the username of `azureuser`. Use your own values:

```
ssh azureuser@myvm.westus.cloudapp.azure.com
```

If you are using Windows and need more information on using SSH, see [How to use SSH keys with Windows](#).

Next, install xfce using `apt` as follows:

```
sudo apt-get update  
sudo apt-get install xfce4
```

Install and configure a remote desktop server

Now that you have a desktop environment installed, configure a remote desktop service to listen for incoming connections. [xrdp](#) is an open source Remote Desktop Protocol (RDP) server that is available on most Linux distributions, and works well with xfce. Install xrdp on your Ubuntu VM as follows:

```
sudo apt-get install xrdp  
sudo systemctl enable xrdp
```

Tell xrdp what desktop environment to use when you start your session. Configure xrdp to use xfce as your desktop environment as follows:

```
echo xfce4-session >~/xsession
```

Restart the xrdp service for the changes to take effect as follows:

```
sudo service xrdp restart
```

Set a local user account password

If you created a password for your user account when you created your VM, skip this step. If you only use SSH key authentication and do not have a local account password set, specify a password before you use xrdp to log in to your VM. xrdp cannot accept SSH keys for authentication. The following example specifies a password for the user account *azureuser*:

```
sudo passwd azureuser
```

NOTE

Specifying a password does not update your SSHD configuration to permit password logins if it currently does not. From a security perspective, you may wish to connect to your VM with an SSH tunnel using key-based authentication and then connect to xrdp. If so, skip the following step on creating a network security group rule to allow remote desktop traffic.

Create a Network Security Group rule for Remote Desktop traffic

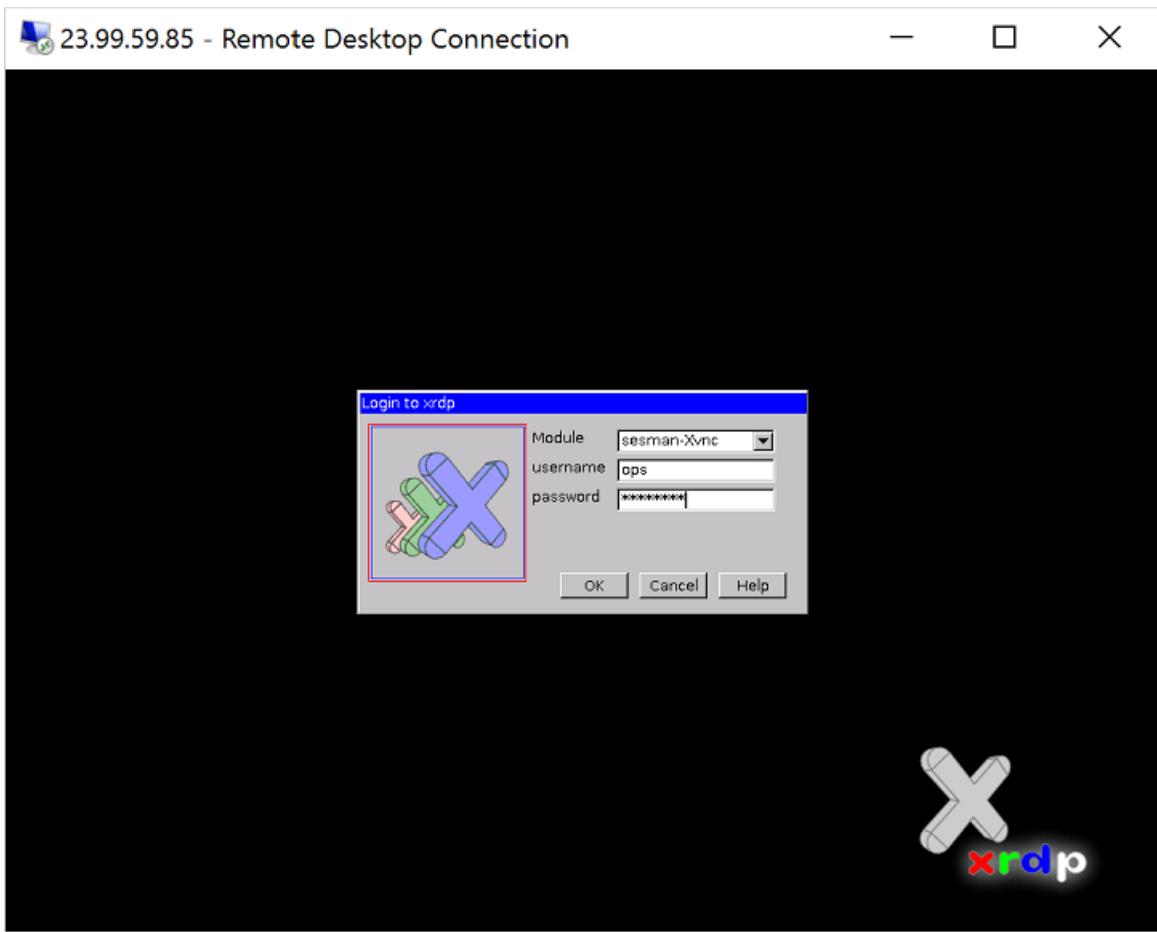
To allow Remote Desktop traffic to reach your Linux VM, a network security group rule needs to be created that allows TCP on port 3389 to reach your VM. For more information about network security group rules, see [What is a network security group?](#) You can also [use the Azure portal to create a network security group rule](#).

The following example creates a network security group rule with [az vm open-port](#) on port 3389. From the Azure CLI, not the SSH session to your VM, open the following network security group rule:

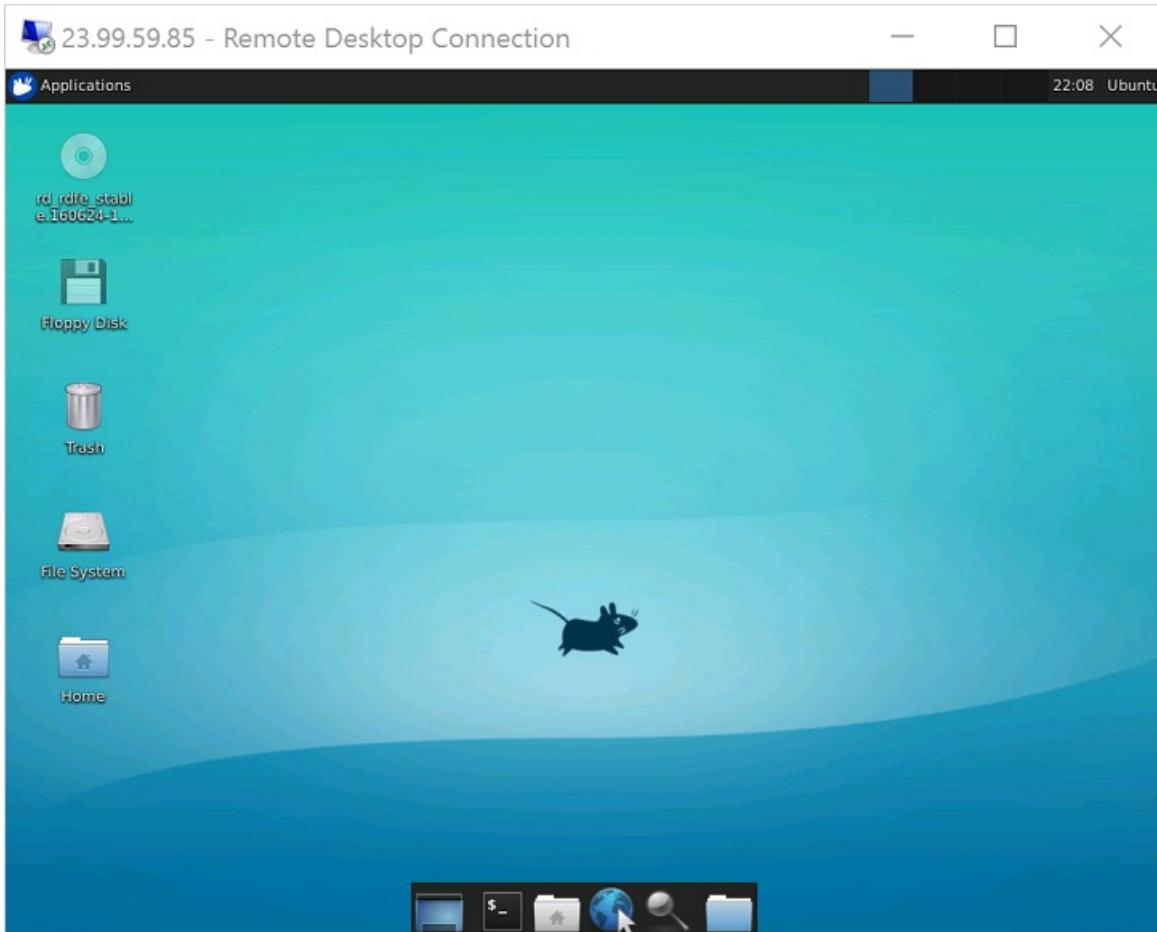
```
az vm open-port --resource-group myResourceGroup --name myVM --port 3389
```

Connect your Linux VM with a Remote Desktop client

Open your local remote desktop client and connect to the IP address or DNS name of your Linux VM. Enter the username and password for the user account on your VM as follows:



After authenticating, the xfce desktop environment will load and look similar to the following example:



If your local RDP client uses network level authentication (NLA), you may need to disable that connection setting. XRDp does not currently support NLA. You can also look at alternative RDP solutions that do support NLA, such

as [FreeRDP](#).

Troubleshoot

If you cannot connect to your Linux VM using a Remote Desktop client, use `netstat` on your Linux VM to verify that your VM is listening for RDP connections as follows:

```
sudo netstat -plnt | grep rdp
```

The following example shows the VM listening on TCP port 3389 as expected:

```
tcp      0      0      127.0.0.1:3350      0.0.0.0:*      LISTEN      53192/xrdp-sesman
tcp      0      0      0.0.0.0:3389      0.0.0.0:*      LISTEN      53188/xrdp
```

If the *xrdp-sesman* service is not listening, on an Ubuntu VM restart the service as follows:

```
sudo service xrdp restart
```

Review logs in `/var/log` on your Ubuntu VM for indications as to why the service may not be responding. You can also monitor the syslog during a remote desktop connection attempt to view any errors:

```
tail -f /var/log/syslog
```

Other Linux distributions such as Red Hat Enterprise Linux and SUSE may have different ways to restart services and alternate log file locations to review.

If you do not receive any response in your remote desktop client and do not see any events in the system log, this behavior indicates that remote desktop traffic cannot reach the VM. Review your network security group rules to ensure that you have a rule to permit TCP on port 3389. For more information, see [Troubleshoot application connectivity issues](#).

Next steps

For more information about creating and using SSH keys with Linux VMs, see [Create SSH keys for Linux VMs in Azure](#).

For information on using SSH from Windows, see [How to use SSH keys with Windows](#).

Join a Red Hat Enterprise Linux 7 virtual machine to a managed domain

6/28/2019 • 4 minutes to read • [Edit Online](#)

This article shows you how to join a Red Hat Enterprise Linux (RHEL) 7 virtual machine to an Azure AD Domain Services managed domain.

IMPORTANT

Enable password hash synchronization to Azure AD Domain Services, before you complete the tasks in this article.

Follow the instructions below, depending on the type of users in your Azure AD directory. Complete both sets of instructions if you have a mix of cloud-only and synced user accounts in your Azure AD directory. You may not be able to carry out the following operations in case you are trying to use a B2B Guest account (example , your gmail or MSA from a different Identity provider which we allow) because we do not have the password for these users synced to managed domain as these are guest accounts in the directory. The complete information about these accounts including their passwords would be outside of Azure AD and as this information is not in Azure AD hence it does not even get synced to the managed domain.

- [Instructions for cloud-only user accounts](#)
- [Instructions for user accounts synchronized from an on-premises directory](#)

Before you begin

To perform the tasks listed in this article, you need:

1. A valid **Azure subscription**.
2. An **Azure AD directory** - either synchronized with an on-premises directory or a cloud-only directory.
3. **Azure AD Domain Services** must be enabled for the Azure AD directory. If you haven't done so, follow all the tasks outlined in the [Getting Started guide](#).
4. Ensure that you have configured the IP addresses of the managed domain as the DNS servers for the virtual network. For more information, see [how to update DNS settings for the Azure virtual network](#)
5. Complete the steps required to [synchronize passwords to your Azure AD Domain Services managed domain](#).

Provision a Red Hat Enterprise Linux virtual machine

Provision a RHEL 7 virtual machine in Azure, using any of the following methods:

- [Azure portal](#)
- [Azure CLI](#)
- [Azure PowerShell](#)

IMPORTANT

- Deploy the virtual machine into the **same virtual network in which you have enabled Azure AD Domain Services**.
- Pick a **different subnet** than the one in which you have enabled Azure AD Domain Services.

Connect remotely to the newly provisioned Linux virtual machine

The RHEL 7.2 virtual machine has been provisioned in Azure. The next task is to connect remotely to the virtual

machine using the local administrator account created while provisioning the VM.

Follow the instructions in the article [How to log on to a virtual machine running Linux](#).

Configure the hosts file on the Linux virtual machine

In your SSH terminal, edit the /etc/hosts file and update your machine's IP address and hostname.

```
sudo vi /etc/hosts
```

In the hosts file, enter the following value:

```
127.0.0.1 contoso-rhel.contoso100.com contoso-rhel
```

Here, 'contoso100.com' is the DNS domain name of your managed domain. 'contoso-rhel' is the hostname of the RHEL virtual machine you are joining to the managed domain.

Install required packages on the Linux virtual machine

Next, install packages required for domain join on the virtual machine. In your SSH terminal, type the following command to install the required packages:

```
...
sudo yum install realmd sssd krb5-workstation krb5-libs samba-common-tools
...
```

Join the Linux virtual machine to the managed domain

Now that the required packages are installed on the Linux virtual machine, the next task is to join the virtual machine to the managed domain.

1. Discover the AAD Domain Services managed domain. In your SSH terminal, type the following command:

```
sudo realm discover CONTOSO100.COM
```

NOTE

Troubleshooting: If *realm discover* is unable to find your managed domain:

- Ensure that the domain is reachable from the virtual machine (try ping).
- Check that the virtual machine has indeed been deployed to the same virtual network in which the managed domain is available.
- Check to see if you have updated the DNS server settings for the virtual network to point to the domain controllers of the managed domain.

2. Initialize Kerberos. In your SSH terminal, type the following command:

TIP

- Ensure that you specify a user who belongs to the 'AAD DC Administrators' group.
- Specify the domain name in capital letters, else kinit fails.

```
kinit bob@CONTOSO100.COM
```

3. Join the machine to the domain. In your SSH terminal, type the following command:

TIP

Use the same user account you specified in the preceding step ('kinit').

```
sudo realm join --verbose CONTOSO100.COM -U 'bob@CONTOSO100.COM'
```

You should get a message ("Successfully enrolled machine in realm") when the machine is successfully joined to the managed domain.

Verify domain join

Verify whether the machine has been successfully joined to the managed domain. Connect to the domain joined RHEL VM using a different SSH connection. Use a domain user account and then check to see if the user account is resolved correctly.

1. In your SSH terminal, type the following command to connect to the domain joined RHEL virtual machine using SSH. Use a domain account that belongs to the managed domain (for example, 'bob@CONTOSO100.COM' in this case.)

```
ssh -l bob@CONTOSO100.COM contoso-rhel.contoso100.com
```

2. In your SSH terminal, type the following command to see if the home directory was initialized correctly.

```
pwd
```

3. In your SSH terminal, type the following command to see if the group memberships are being resolved correctly.

```
id
```

Troubleshooting domain join

Refer to the [Troubleshooting domain join](#) article.

Related Content

- [Azure AD Domain Services - Getting Started guide](#)
- [Join a Windows Server virtual machine to an Azure AD Domain Services managed domain](#)
- [How to log on to a virtual machine running Linux.](#)
- [Installing Kerberos](#)
- [Red Hat Enterprise Linux 7 - Windows Integration Guide](#)

Join a CentOS Linux virtual machine to a managed domain

6/28/2019 • 4 minutes to read • [Edit Online](#)

This article shows you how to join a CentOS Linux virtual machine in Azure to an Azure AD Domain Services managed domain.

IMPORTANT

Enable password hash synchronization to Azure AD Domain Services, before you complete the tasks in this article.

Follow the instructions below, depending on the type of users in your Azure AD directory. Complete both sets of instructions if you have a mix of cloud-only and synced user accounts in your Azure AD directory. You may not be able to carry out the following operations in case you are trying to use a B2B Guest account (example , your gmail or MSA from a different Identity provider which we allow) because we do not have the password for these users synced to managed domain as these are guest accounts in the directory. The complete information about these accounts including their passwords would be outside of Azure AD and as this information is not in Azure AD hence it does not even get synced to the managed domain.

- [Instructions for cloud-only user accounts](#)
- [Instructions for user accounts synchronized from an on-premises directory](#)

Before you begin

To perform the tasks listed in this article, you need:

1. A valid **Azure subscription**.
2. An **Azure AD directory** - either synchronized with an on-premises directory or a cloud-only directory.
3. **Azure AD Domain Services** must be enabled for the Azure AD directory. If you haven't done so, follow all the tasks outlined in the [Getting Started guide](#).
4. Ensure that you have configured the IP addresses of the managed domain as the DNS servers for the virtual network. For more information, see [how to update DNS settings for the Azure virtual network](#)
5. Complete the steps required to [synchronize passwords to your Azure AD Domain Services managed domain](#).

Provision a CentOS Linux virtual machine

Provision a CentOS virtual machine in Azure, using any of the following methods:

- [Azure portal](#)
- [Azure CLI](#)
- [Azure PowerShell](#)

IMPORTANT

- Deploy the virtual machine into the **same virtual network in which you have enabled Azure AD Domain Services**.
- Pick a **different subnet** than the one in which you have enabled Azure AD Domain Services.

Connect remotely to the newly provisioned Linux virtual machine

The CentOS virtual machine has been provisioned in Azure. The next task is to connect remotely to the virtual

machine using the local administrator account created while provisioning the VM.

Follow the instructions in the article [How to log on to a virtual machine running Linux](#).

Configure the hosts file on the Linux virtual machine

In your SSH terminal, edit the /etc/hosts file and update your machine's IP address and hostname.

```
sudo vi /etc/hosts
```

In the hosts file, enter the following value:

```
127.0.0.1 contoso-centos.contoso100.com contoso-centos
```

Here, 'contoso100.com' is the DNS domain name of your managed domain. 'contoso-centos' is the hostname of the CentOS virtual machine you are joining to the managed domain.

Install required packages on the Linux virtual machine

Next, install packages required for domain join on the virtual machine. In your SSH terminal, type the following command to install the required packages:

```
...
sudo yum install realmd sssd krb5-workstation krb5-libs oddjob oddjob-mkhomedir samba-common-tools
...
```

Join the Linux virtual machine to the managed domain

Now that the required packages are installed on the Linux virtual machine, the next task is to join the virtual machine to the managed domain.

1. Discover the AAD Domain Services managed domain. In your SSH terminal, type the following command:

```
sudo realm discover CONTOSO100.COM
```

NOTE

Troubleshooting: If *realm discover* is unable to find your managed domain:

- Ensure that the domain is reachable from the virtual machine (try ping).
- Check that the virtual machine has indeed been deployed to the same virtual network in which the managed domain is available.
- Check to see if you have updated the DNS server settings for the virtual network to point to the domain controllers of the managed domain.

2. Initialize Kerberos. In your SSH terminal, type the following command:

TIP

- Specify a user who belongs to the 'AAD DC Administrators' group.
- Specify the domain name in capital letters, else kinit fails.

```
kinit bob@CONTOSO100.COM
```

3. Join the machine to the domain. In your SSH terminal, type the following command:

TIP

Use the same user account you specified in the preceding step ('kinit').

```
sudo realm join --verbose CONTOSO100.COM -U 'bob@CONTOSO100.COM'
```

You should get a message ("Successfully enrolled machine in realm") when the machine is successfully joined to the managed domain.

Verify domain join

Verify whether the machine has been successfully joined to the managed domain. Connect to the domain joined CentOS VM using a different SSH connection. Use a domain user account and then check to see if the user account is resolved correctly.

1. In your SSH terminal, type the following command to connect to the domain joined CentOS virtual machine using SSH. Use a domain account that belongs to the managed domain (for example, 'bob@CONTOSO100.COM' in this case.)

```
ssh -l bob@CONTOSO100.COM contoso-centos.contoso100.com
```

2. In your SSH terminal, type the following command to see if the home directory was initialized correctly.

```
pwd
```

3. In your SSH terminal, type the following command to see if the group memberships are being resolved correctly.

```
id
```

Troubleshooting domain join

Refer to the [Troubleshooting domain join](#) article.

Related Content

- [Azure AD Domain Services - Getting Started guide](#)
- [Join a Windows Server virtual machine to an Azure AD Domain Services managed domain](#)
- [How to log on to a virtual machine running Linux.](#)
- [Installing Kerberos](#)
- [Red Hat Enterprise Linux 7 - Windows Integration Guide](#)

Join an Ubuntu virtual machine in Azure to a managed domain

6/28/2019 • 6 minutes to read • [Edit Online](#)

This article shows you how to join an Ubuntu Linux virtual machine to an Azure AD Domain Services managed domain.

IMPORTANT

Enable password hash synchronization to Azure AD Domain Services, before you complete the tasks in this article.

Follow the instructions below, depending on the type of users in your Azure AD directory. Complete both sets of instructions if you have a mix of cloud-only and synced user accounts in your Azure AD directory. You may not be able to carry out the following operations in case you are trying to use a B2B Guest account (example , your gmail or MSA from a different Identity provider which we allow) because we do not have the password for these users synced to managed domain as these are guest accounts in the directory. The complete information about these accounts including their passwords would be outside of Azure AD and as this information is not in Azure AD hence it does not even get synced to the managed domain.

- [Instructions for cloud-only user accounts](#)
- [Instructions for user accounts synchronized from an on-premises directory](#)

Before you begin

To perform the tasks listed in this article, you need:

1. A valid **Azure subscription**.
2. An **Azure AD directory** - either synchronized with an on-premises directory or a cloud-only directory.
3. **Azure AD Domain Services** must be enabled for the Azure AD directory. If you haven't done so, follow all the tasks outlined in the [Getting Started guide](#).
4. Ensure that you have configured the IP addresses of the managed domain as the DNS servers for the virtual network. For more information, see [how to update DNS settings for the Azure virtual network](#)
5. Complete the steps required to [synchronize passwords to your Azure AD Domain Services managed domain](#).

Provision an Ubuntu Linux virtual machine

Provision an Ubuntu Linux virtual machine in Azure, using any of the following methods:

- [Azure portal](#)
- [Azure CLI](#)
- [Azure PowerShell](#)

IMPORTANT

- Deploy the virtual machine into the **same virtual network in which you have enabled Azure AD Domain Services**.
- Pick a **different subnet** than the one in which you have enabled Azure AD Domain Services.

Connect remotely to the Ubuntu Linux virtual machine

The Ubuntu virtual machine has been provisioned in Azure. The next task is to connect remotely to the virtual

machine using the local administrator account created while provisioning the VM.

Follow the instructions in the article [How to log on to a virtual machine running Linux](#).

Configure the hosts file on the Linux virtual machine

In your SSH terminal, edit the /etc/hosts file and update your machine's IP address and hostname.

```
sudo vi /etc/hosts
```

In the hosts file, enter the following value:

```
127.0.0.1 contoso-ubuntu.contoso100.com contoso-ubuntu
```

Here, 'contoso100.com' is the DNS domain name of your managed domain. 'contoso-ubuntu' is the hostname of the Ubuntu virtual machine you are joining to the managed domain.

Install required packages on the Linux virtual machine

Next, install packages required for domain join on the virtual machine. Perform the following steps:

1. In your SSH terminal, type the following command to download the package lists from the repositories. This command updates the package lists to get information on the newest versions of packages and their dependencies.

```
sudo apt-get update
```

2. Type the following command to install the required packages.

```
sudo apt-get install krb5-user samba sssd sssd-tools libnss-sss libpam-sss ntp ntpdate realmd adcli
```

3. During the Kerberos installation, you see a pink screen. The installation of the 'krb5-user' package prompts for the realm name (in ALL UPPERCASE). The installation writes the [realm] and [domain_realm] sections in /etc/krb5.conf.

TIP

If the name of your managed domain is contoso100.com, enter CONTOSO100.COM as the realm. Remember, the realm name must be specified in UPPERCASE.

Configure the NTP (Network Time Protocol) settings on the Linux virtual machine

The date and time of your Ubuntu VM must synchronize with the managed domain. Add your managed domain's NTP hostname in the /etc/ntp.conf file.

```
sudo vi /etc/ntp.conf
```

In the ntp.conf file, enter the following value and save the file:

```
server contoso100.com
```

Here, 'contoso100.com' is the DNS domain name of your managed domain.

Now sync the Ubuntu VM's date and time with NTP server and then start the NTP service:

```
sudo systemctl stop ntp
sudo ntpdate contoso100.com
sudo systemctl start ntp
```

Join the Linux virtual machine to the managed domain

Now that the required packages are installed on the Linux virtual machine, the next task is to join the virtual machine to the managed domain.

1. Discover the AAD Domain Services managed domain. In your SSH terminal, type the following command:

```
sudo realm discover CONTOSO100.COM
```

NOTE

Troubleshooting: If *realm discover* is unable to find your managed domain:

- Ensure that the domain is reachable from the virtual machine (try ping).
- Check that the virtual machine has indeed been deployed to the same virtual network in which the managed domain is available.
- Check to see if you have updated the DNS server settings for the virtual network to point to the domain controllers of the managed domain.

2. Initialize Kerberos. In your SSH terminal, type the following command:

TIP

- Ensure that you specify a user who belongs to the 'AAD DC Administrators' group.
- Specify the domain name in capital letters, else kinit fails.

```
kinit bob@CONTOSO100.COM
```

3. Join the machine to the domain. In your SSH terminal, type the following command:

TIP

Use the same user account you specified in the preceding step ('kinit').

```
sudo realm join --verbose CONTOSO100.COM -U 'bob@CONTOSO100.COM' --install=/
```

You should get a message ("Successfully enrolled machine in realm") when the machine is successfully joined to the managed domain.

Update the SSSD configuration and restart the service

1. In your SSH terminal, type the following command. Open the sssd.conf file and make the following change

```
sudo vi /etc/sssd/sssd.conf
```

2. Comment out the line **use_fully_qualified_names = True** and save the file.

```
# use_fully_qualified_names = True
```

3. Restart the SSSD service.

```
sudo service sssd restart
```

Configure automatic home directory creation

To enable automatic creation of the home directory after logging in users, type the following commands in your PuTTY terminal:

```
sudo vi /etc/pam.d/common-session
```

Add the following line in this file below the line 'session optional pam_sss.so' and save it:

```
session required pam_mkhomedir.so skel=/etc/skel/ umask=0077
```

Verify domain join

Verify whether the machine has been successfully joined to the managed domain. Connect to the domain joined Ubuntu VM using a different SSH connection. Use a domain user account and then check to see if the user account is resolved correctly.

1. In your SSH terminal, type the following command to connect to the domain joined Ubuntu virtual machine using SSH. Use a domain account that belongs to the managed domain (for example, 'bob@CONTOSO100.COM' in this case.)

```
ssh -l bob@CONTOSO100.COM contoso-ubuntu.contoso100.com
```

2. In your SSH terminal, type the following command to see if the home directory was initialized correctly.

```
pwd
```

3. In your SSH terminal, type the following command to see if the group memberships are being resolved correctly.

```
id
```

Grant the 'AAD DC Administrators' group sudo privileges

You can grant members of the 'AAD DC Administrators' group administrative privileges on the Ubuntu VM. The sudo file is located at /etc/sudoers. The members of AD groups added in sudoers can perform sudo.

1. In your SSH terminal, ensure you are logged in with superuser privileges. You can use the local administrator account you specified while creating the VM. Execute the following command:

```
sudo vi /etc/sudoers
```

2. Add the following entry to the /etc/sudoers file and save it:

```
# Add 'AAD DC Administrators' group members as admins.  
%AAD\ DC\ Administrators ALL=(ALL) NOPASSWD:ALL
```

3. You can now log in as a member of the 'AAD DC Administrators' group and should have administrative privileges on the VM.

Troubleshooting domain join

Refer to the [Troubleshooting domain join](#) article.

Related Content

- [Azure AD Domain Services - Getting Started guide](#)
- [Join a Windows Server virtual machine to an Azure AD Domain Services managed domain](#)
- [How to log on to a virtual machine running Linux.](#)

Log in to a Linux virtual machine in Azure using Azure Active Directory authentication (Preview)

4/22/2019 • 9 minutes to read • [Edit Online](#)

To improve the security of Linux virtual machines (VMs) in Azure, you can integrate with Azure Active Directory (AD) authentication. When you use Azure AD authentication for Linux VMs, you centrally control and enforce policies that allow or deny access to the VMs. This article shows you how to create and configure a Linux VM to use Azure AD authentication.

NOTE

This feature is in preview and is not recommended for use with production virtual machines or workloads. Use this feature on a test virtual machine that you expect to discard after testing.

There are many benefits of using Azure AD authentication to log in to Linux VMs in Azure, including:

- **Improved security:**

- You can use your corporate AD credentials to log in to Azure Linux VMs. There is no need to create local administrator accounts and manage credential lifetime.
- By reducing your reliance on local administrator accounts, you do not need to worry about credential loss/theft, users configuring weak credentials etc.
- The password complexity and password lifetime policies configured for your Azure AD directory help secure Linux VMs as well.
- To further secure login to Azure virtual machines, you can configure multi-factor authentication.
- The ability to log in to Linux VMs with Azure Active Directory also works for customers that use [Federation Services](#).

- **Seamless collaboration:** With Role-Based Access Control (RBAC), you can specify who can sign in to a given VM as a regular user or with administrator privileges. When users join or leave your team, you can update the RBAC policy for the VM to grant access as appropriate. This experience is much simpler than having to scrub VMs to remove unnecessary SSH public keys. When employees leave your organization and their user account is disabled or removed from Azure AD, they no longer have access to your resources.

Supported Azure regions and Linux distributions

The following Linux distributions are currently supported during the preview of this feature:

DISTRIBUTION	VERSION
CentOS	CentOS 6, CentOS 7
Debian	Debian 9
openSUSE	openSUSE Leap 42.3
RedHat Enterprise Linux	RHEL 6, RHEL 7
SUSE Linux Enterprise Server	SLES 12

DISTRIBUTION	VERSION
Ubuntu Server	Ubuntu 14.04 LTS, Ubuntu Server 16.04, and Ubuntu Server 18.04

The following Azure regions are currently supported during the preview of this feature:

- All global Azure regions

IMPORTANT

To use this preview feature, only deploy a supported Linux distro and in a supported Azure region. The feature is not supported in Azure Government or sovereign clouds.

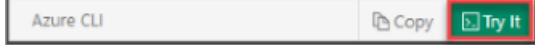
Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services.

Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

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 Linux virtual machine

Create a resource group with `az group create`, then create a VM with `az vm create` using a supported distro and in a supported region. The following example deploys a VM named *myVM* that uses *Ubuntu 16.04 LTS* into a resource group named *myResourceGroup* in the *southcentralus* region. In the following examples, you can provide your own resource group and VM names as needed.

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

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

It takes a few minutes to create the VM and supporting resources.

Install the Azure AD login VM extension

To log in to a Linux VM with Azure AD credentials, install the Azure Active Directory login VM extension. VM extensions are small applications that provide post-deployment configuration and automation tasks on Azure virtual machines. Use [az vm extension set](#) to install the `AADLoginForLinux` extension on the VM named `myVM` in the `myResourceGroup` resource group:

```
az vm extension set \
    --publisher Microsoft.Azure.ActiveDirectory.LinuxSSH \
    --name AADLoginForLinux \
    --resource-group myResourceGroup \
    --vm-name myVM
```

The `provisioningState` of `Succeeded` is shown once the extension is installed on the VM.

Configure role assignments for the VM

Azure Role-Based Access Control (RBAC) policy determines who can log in to the VM. Two RBAC roles are used to authorize VM login:

- **Virtual Machine Administrator Login:** Users with this role assigned can log in to an Azure virtual machine with Windows Administrator or Linux root user privileges.
- **Virtual Machine User Login:** Users with this role assigned can log in to an Azure virtual machine with regular user privileges.

NOTE

To allow a user to log in to the VM over SSH, you must assign either the `Virtual Machine Administrator Login` or `Virtual Machine User Login` role. An Azure user with the `Owner` or `Contributor` roles assigned for a VM do not automatically have privileges to log in to the VM over SSH.

The following example uses [az role assignment create](#) to assign the `Virtual Machine Administrator Login` role to the VM for your current Azure user. The username of your active Azure account is obtained with [az account show](#), and the `scope` is set to the VM created in a previous step with [az vm show](#). The scope could also be assigned at a resource group or subscription level, and normal RBAC inheritance permissions apply. For more information, see [Role-Based Access Controls](#)

```
username=$(az account show --query user.name --output tsv)
vm=$(az vm show --resource-group myResourceGroup --name myVM --query id -o tsv)

az role assignment create \
--role "Virtual Machine Administrator Login" \
--assignee $username \
--scope $vm
```

NOTE

If your AAD domain and logon username domain do not match, you must specify the object ID of your user account with the `--assignee-object-id`, not just the username for `--assignee`. You can obtain the object ID for your user account with [az ad user list](#).

For more information on how to use RBAC to manage access to your Azure subscription resources, see using the [Azure CLI](#), [Azure portal](#), or [Azure PowerShell](#).

You can also configure Azure AD to require multi-factor authentication for a specific user to sign in to the Linux virtual machine. For more information, see [Get started with Azure Multi-Factor Authentication in the cloud](#).

Log in to the Linux virtual machine

First, view the public IP address of your VM with `az vm show`:

```
az vm show --resource-group myResourceGroup --name myVM -d --query publicIps -o tsv
```

Log in to the Azure Linux virtual machine using your Azure AD credentials. The `-l` parameter lets you specify your own Azure AD account address. Account addresses should be entered in all lowercase. Use the public IP address of your VM from the previous command:

```
ssh -l azureuser@contoso.onmicrosoft.com publicIps
```

You are prompted to sign in to Azure AD with a one-time use code at <https://microsoft.com/devicelogin>. Copy and paste the one-time use code into the device login page, as shown in the following example:

```
~$ ssh -l azureuser@contoso.onmicrosoft.com 13.65.237.247
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code FJS3K6X4D
to authenticate. Press ENTER when ready.
```

When prompted, enter your Azure AD login credentials at the login page. The following message is shown in the web browser when you have successfully authenticated:

You have signed in to the Microsoft Azure Linux Virtual Machine Sign-In application on your device.

Close the browser window, return to the SSH prompt, and press the **Enter** key. You are now signed in to the Azure Linux virtual machine with the role permissions as assigned, such as *VM User* or *VM Administrator*. If your user account is assigned the *Virtual Machine Administrator Login* role, you can use the `sudo` to run commands that require root privileges.

Sudo and AAD login

The first time that you run `sudo`, you will be asked to authenticate a second time. If you don't want to have to

authenticate again to run sudo, you can edit your sudoers file `/etc/sudoers.d/aad_admins` and replace this line:

```
%aad_admins ALL=(ALL) ALL
```

With this line:

```
%aad_admins ALL=(ALL) NOPASSWD:ALL
```

Troubleshoot sign-in issues

Some common errors when you try to SSH with Azure AD credentials include no RBAC roles assigned, and repeated prompts to sign in. Use the following sections to correct these issues.

Access denied: RBAC role not assigned

If you see the following error on your SSH prompt, verify that you have configured RBAC policies for the VM that grants the user either the *Virtual Machine Administrator Login* or *Virtual Machine User Login* role:

```
login as: azureuser@contoso.onmicrosoft.com
Using keyboard-interactive authentication.
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code FJX327AXD
to authenticate. Press ENTER when ready.
Using keyboard-interactive authentication.
Access denied: to sign-in you be assigned a role with action 'Microsoft.Compute/virtualMachines/login/action',
for example 'Virtual Machine User Login'
Access denied
```

Continued SSH sign-in prompts

If you successfully complete the authentication step in a web browser, you may be immediately prompted to sign in again with a fresh code. This error is typically caused by a mismatch between the sign-in name you specified at the SSH prompt and the account you signed in to Azure AD with. To correct this issue:

- Verify that the sign-in name you specified at the SSH prompt is correct. A typo in the sign-in name could cause a mismatch between the sign-in name you specified at the SSH prompt and the account you signed in to Azure AD with. For example, you typed *azuresuer@contoso.onmicrosoft.com* instead of *azureuser@contoso.onmicrosoft.com*.
- If you have multiple user accounts, make sure you don't provide a different user account in the browser window when signing in to Azure AD.
- Linux is a case-sensitive operating system. There is a difference between '*Azureuser@contoso.onmicrosoft.com*' and '*azureuser@contoso.onmicrosoft.com*', which can cause a mismatch. Make sure that you specify the UPN with the correct case-sensitivity at the SSH prompt.

Preview feedback

Share your feedback about this preview feature or report issues using it on the [Azure AD feedback forum](#)

Next steps

For more information on Azure Active Directory, see [What is Azure Active Directory](#)

Red Hat Update Infrastructure for on-demand Red Hat Enterprise Linux VMs in Azure

6/10/2019 • 7 minutes to read • [Edit Online](#)

Red Hat Update Infrastructure (RHUI) allows cloud providers, such as Azure, to mirror Red Hat-hosted repository content, create custom repositories with Azure-specific content, and make it available to end-user VMs.

Red Hat Enterprise Linux (RHEL) Pay-As-You-Go (PAYG) images come preconfigured to access Azure RHUI. No additional configuration is needed. To get the latest updates, run `sudo yum update` after your RHEL instance is ready. This service is included as part of the RHEL PAYG software fees.

Additional information on RHEL images in Azure, including publishing and retention policies, is available [here](#).

Information on Red Hat support policies for all versions of RHEL can be found on the [Red Hat Enterprise Linux Life Cycle](#) page.

Important information about Azure RHUI

- Azure RHUI is the update infrastructure that supports all RHEL PAYG VMs created in Azure. This does not preclude you from registering your PAYG RHEL VMs with Subscription Manager or Satellite or other source of updates, but doing so with a PAYG VM will result in indirect double-billing. See the following point for details.
- Access to the Azure-hosted RHUI is included in the RHEL PAYG image price. If you unregister a PAYG RHEL VM from the Azure-hosted RHUI that does not convert the virtual machine into a bring-your-own-license (BYOL) type of VM. If you register the same VM with another source of updates, you might incur *indirect* double charges. You're charged the first time for the Azure RHEL software fee. You're charged the second time for Red Hat subscriptions that were purchased previously. If you consistently need to use an update infrastructure other than Azure-hosted RHUI, consider registering to use the [RHEL BYOS images](#).
- The default behavior of RHUI is to upgrade your RHEL VM to the latest minor version when you run `sudo yum update`.

For example, if you provision a VM from an RHEL 7.4 PAYG image and run `sudo yum update`, you end up with an RHEL 7.6 VM (the latest minor version in the RHEL7 family).

To avoid this behavior, you can switch to [Extended Update Support channels](#) or build your own image as described in the [Create and upload a Red Hat-based virtual machine for Azure](#) article. If you build your own image, you need to connect it to a different update infrastructure ([directly to Red Hat content delivery servers](#) or a [Red Hat Satellite server](#)).

- RHEL SAP PAYG images in Azure (RHEL for SAP, RHEL for SAP HANA, and RHEL for SAP Business Applications) are connected to dedicated RHUI channels that remain on the specific RHEL minor version as required for SAP certification.
- Access to Azure-hosted RHUI is limited to the VMs within the [Azure datacenter IP ranges](#). If you're proxying all VM traffic via an on-premises network infrastructure, you might need to set up user-defined routes for the RHEL PAYG VMs to access the Azure RHUI.

RHEL EUS and version-locking RHEL VMs

Some customers may want to lock their RHEL VMs to a certain RHEL minor release. You can version-lock your

RHEL VM to a specific minor version by updating the repositories to point to the Extended Update Support repositories. You can also undo the EUS version-locking operation.

NOTE

EUS is not supported on RHEL Extras. This means that if you are installing a package that is usually available from the RHEL Extras channel, you will not be able to do so while on EUS. The Red Hat Extras Product Life Cycle is detailed [here](#).

At the time of this writing, EUS support has ended for RHEL <= 7.3. See the "Red Hat Enterprise Linux Longer Support Add-Ons" section in the [Red Hat documentation](#) for more details.

- RHEL 7.4 EUS support ends August 31, 2019
- RHEL 7.5 EUS support ends April 30, 2020
- RHEL 7.6 EUS support ends October 31, 2020

Switch a RHEL VM to EUS (version-lock to a specific minor version)

Use the following instructions to lock a RHEL VM to a particular minor release (run as root):

NOTE

This only applies for RHEL versions for which EUS is available. At the time of this writing, this includes RHEL 7.2-7.6. More details are available at the [Red Hat Enterprise Linux Life Cycle](#) page.

1. Disable non-EUS repos:

```
yum --disablerepo='*' remove 'rhui-azure-rhel7'
```

2. Add EUS repos:

```
yum --config='https://rhelimage.blob.core.windows.net/repositories/rhui-microsoft-azure-rhel7-eus.config' install 'rhui-azure-rhel7-eus'
```

3. Lock the releasever variable (run as root):

```
echo $(. /etc/os-release && echo $VERSION_ID) > /etc/yum/vars/releasever
```

NOTE

The above instruction will lock the RHEL minor release to the current minor release. Enter a specific minor release if you are looking to upgrade and lock to a later minor release that is not the latest. For example,

`echo 7.5 > /etc/yum/vars/releasever` will lock your RHEL version to RHEL 7.5

4. Update your RHEL VM

```
sudo yum update
```

Switch a RHEL VM back to non-EUS (remove a version lock)

Run the following as root:

1. Remove the releasever file:

```
rm /etc/yum/vars/releasever
```

2. Disable EUS repos:

```
yum --disablerepo='*' remove 'rhui-azure-rhel7-eus'
```

3. Update your RHEL VM

```
sudo yum update
```

The IPs for the RHUI content delivery servers

RHUI is available in all regions where RHEL on-demand images are available. It currently includes all public regions listed on the [Azure status dashboard](#) page, Azure US Government, and Microsoft Azure Germany regions.

If you're using a network configuration to further restrict access from RHEL PAYG VMs, make sure the following IPs are allowed for `yum update` to work depending on the environment you're in:

```
# Azure Global
13.91.47.76
40.85.190.91
52.187.75.218
52.174.163.213
52.237.203.198

# Azure US Government
13.72.186.193
13.72.14.155
52.244.249.194

# Azure Germany
51.5.243.77
51.4.228.145
```

Azure RHUI Infrastructure

Update expired RHUI client certificate on a VM

If you are using an older RHEL VM image, for example, RHEL 7.4 (image URN: `RedHat:RHEL:7.4:7.4.2018010506`), you will experience connectivity issues to RHUI due to a now-expired SSL client certificate. The error you see may look like "SSL peer rejected your certificate as expired" or "Error: Cannot retrieve repository metadata (repomd.xml) for repository: ... Please verify its path and try again". To overcome this problem, please update the RHUI client package on the VM using the following command:

```
sudo yum update -y --disablerepo='*' --enablerepo='*microsoft*'
```

Alternatively, running `sudo yum update` may also update the client certificate package (depending on your RHEL version), despite "expired SSL certificate" errors you will see for other repositories. If this update is successful, normal connectivity to other RHUI repositories should be restored, so you will be able to run `sudo yum update` successfully.

If you run into a 404 error while running a `yum update`, try the following to refresh your yum cache:

```
sudo yum clean all;
sudo yum makecache
```

Troubleshoot connection problems to Azure RHUI

If you experience problems connecting to Azure RHUI from your Azure RHEL PAYG VM, follow these steps:

1. Inspect the VM configuration for the Azure RHUI endpoint:

- a. Check if the `/etc/yum.repos.d/rh-cloud.repo` file contains a reference to `rhui-[1-3].microsoft.com` in the `baseurl` of the `[rhui-microsoft-azure-rhel*]` section of the file. If it does, you're using the new Azure RHUI.
- b. If it points to a location with the following pattern, `mirrorlist.*cds[1-4].cloudapp.net`, a configuration update is required. You're using the old VM snapshot, and you need to update it to point to the new Azure RHUI.

2. Access to Azure-hosted RHUI is limited to VMs within the [Azure datacenter IP ranges](#).

3. If you're using the new configuration, have verified that the VM connects from the Azure IP range, and still can't connect to Azure RHUI, file a support case with Microsoft or Red Hat.

Infrastructure update

In September 2016, we deployed an updated Azure RHUI. In April 2017, we shut down the old Azure RHUI. If you have been using the RHEL PAYG images (or their snapshots) from September 2016 or later, you're automatically connecting to the new Azure RHUI. If, however, you have older snapshots on your VMs, you need to manually update their configuration to access the Azure RHUI as described in a following section.

The new Azure RHUI servers are deployed with [Azure Traffic Manager](#). In Traffic Manager, a single endpoint (`rhui-1.microsoft.com`) can be used by any VM, regardless of region.

Manual update procedure to use the Azure RHUI servers

This procedure is provided for reference only. RHEL PAYG images already have the correct configuration to connect to Azure RHUI. To manually update the configuration to use the Azure RHUI servers, complete the following steps:

- For RHEL 6:

```
yum --config='https://rhelimage.blob.core.windows.net/repositories/rhui-microsoft-azure-rhel6.config'
install 'rhui-azure-rhel6'
```

- For RHEL 7:

```
yum --config='https://rhelimage.blob.core.windows.net/repositories/rhui-microsoft-azure-rhel7.config'
install 'rhui-azure-rhel7'
```

Next steps

- To create a Red Hat Enterprise Linux VM from an Azure Marketplace PAYG image and to use Azure-hosted RHUI, go to the [Azure Marketplace](#).
- To learn more about the Red Hat images in Azure, go to the [documentation page](#).
- Information on Red Hat support policies for all versions of RHEL can be found on the [Red Hat Enterprise Linux Life Cycle](#) page.

Understanding and using the Azure Linux Agent

3/14/2019 • 7 minutes to read • [Edit Online](#)

The Microsoft Azure Linux Agent (waagent) manages Linux & FreeBSD provisioning, and VM interaction with the Azure Fabric Controller. In addition to the Linux Agent providing provisioning functionality, Azure also provides the option of using cloud-init for some Linux OSes. The Linux Agent provides the following functionality for Linux and FreeBSD IaaS deployments:

NOTE

For more information, see the [README](#).

- **Image Provisioning**

- Creation of a user account
- Configuring SSH authentication types
- Deployment of SSH public keys and key pairs
- Setting the host name
- Publishing the host name to the platform DNS
- Reporting SSH host key fingerprint to the platform
- Resource Disk Management
- Formatting and mounting the resource disk
- Configuring swap space

- **Networking**

- Manages routes to improve compatibility with platform DHCP servers
- Ensures the stability of the network interface name

- **Kernel**

- Configures virtual NUMA (disable for kernel < 2.6.37)
- Consumes Hyper-V entropy for /dev/random
- Configures SCSI timeouts for the root device (which could be remote)

- **Diagnostics**

- Console redirection to the serial port

- **SCVMM Deployments**

- Detects and bootstraps the VMM agent for Linux when running in a System Center Virtual Machine Manager 2012 R2 environment

- **VM Extension**

- Inject component authored by Microsoft and Partners into Linux VM (IaaS) to enable software and configuration automation
- VM Extension reference implementation on <https://github.com/Azure/azure-linux-extensions>

Communication

The information flow from the platform to the agent occurs via two channels:

- A boot-time attached DVD for IaaS deployments. This DVD includes an OVF-compliant configuration file that includes all provisioning information other than the actual SSH keypairs.
- A TCP endpoint exposing a REST API used to obtain deployment and topology configuration.

Requirements

The following systems have been tested and are known to work with the Azure Linux Agent:

NOTE

This list may differ from the official list of supported systems on the Microsoft Azure Platform, as described here:

<https://support.microsoft.com/kb/2805216>

- CoreOS
- CentOS 6.3+
- Red Hat Enterprise Linux 6.7+
- Debian 7.0+
- Ubuntu 12.04+
- openSUSE 12.3+
- SLES 11 SP3+
- Oracle Linux 6.4+

Other Supported Systems:

- FreeBSD 10+ (Azure Linux Agent v2.0.10+)

The Linux agent depends on some system packages in order to function properly:

- Python 2.6+
- OpenSSL 1.0+
- OpenSSH 5.3+
- Filesystem utilities: sfdisk, fdisk, mkfs, parted
- Password tools: chpasswd, sudo
- Text processing tools: sed, grep
- Network tools: ip-route
- Kernel support for mounting UDF filesystems.

Installation

Installation using an RPM or a DEB package from your distribution's package repository is the preferred method of installing and upgrading the Azure Linux Agent. All the [endorsed distribution providers](#) integrate the Azure Linux agent package into their images and repositories.

Refer to the documentation in the [Azure Linux Agent repo on GitHub](#) for advanced installation options, such as installing from source or to custom locations or prefixes.

Command-Line Options

Flags

- verbose: Increase verbosity of specified command
- force: Skip interactive confirmation for some commands

Commands

- help: Lists the supported commands and flags.
- deprovision: Attempt to clean the system and make it suitable for reprovisioning. The following operation deletes:
 - All SSH host keys (if Provisioning.RegenerateSshHostKeyPair is 'y' in the configuration file)
 - Nameserver configuration in /etc/resolv.conf
 - Root password from /etc/shadow (if Provisioning.DeleteRootPassword is 'y' in the configuration file)
 - Cached DHCP client leases
 - Resets host name to localhost.localdomain

WARNING

Deprovisioning does not guarantee that the image is cleared of all sensitive information and suitable for redistribution.

- deprovision+user: Performs everything in -deprovision (above) and also deletes the last provisioned user account (obtained from /var/lib/waagent) and associated data. This parameter is when de-provisioning an image that was previously provisioning on Azure so it may be captured and reused.
- version: Displays the version of waagent
- serialconsole: Configures GRUB to mark ttyS0 (the first serial port) as the boot console. This ensures that kernel bootup logs are sent to the serial port and made available for debugging.
- daemon: Run waagent as a daemon to manage interaction with the platform. This argument is specified to waagent in the waagent init script.
- start: Run waagent as a background process

Configuration

A configuration file (/etc/waagent.conf) controls the actions of waagent. The following shows a sample configuration file:

```
...
Provisioning.Enabled=y
Provisioning.DeleteRootPassword=
Provisioning.RegenerateSshHostKeyPair=y
Provisioning.SshHostKeyPairType=rsa
Provisioning.MonitorHostName=
Provisioning.DecodeCustomData=n
Provisioning.ExecuteCustomData=n
Provisioning.AllowResetSysUser=n
Provisioning.PasswordCryptId=6
Provisioning.PasswordCryptSaltLength=10
ResourceDisk.Format=y
ResourceDisk.Filesystem=ext4
ResourceDisk.MountPoint=/mnt/resource
ResourceDisk.MountOptions=None
ResourceDisk.EnableSwap=n
ResourceDisk.SwapSizeMB=0
LBProbeResponder=y
Logs.Verbose=
OS.RootDeviceScsiTimeout=300
OS.OpenSSLPath=None
HttpProxy.Host=None
HttpProxy.Port=None
AutoUpdate.Enabled=y
...
```

The following various configuration options are described. Configuration options are of three types; Boolean,

String, or Integer. The Boolean configuration options can be specified as "y" or "n". The special keyword "None" may be used for some string type configuration entries as the following details:

Provisioning.Enabled:

Type: Boolean
Default: y

This allows the user to enable or disable the provisioning functionality in the agent. Valid values are "y" or "n". If provisioning is disabled, SSH host and user keys in the image are preserved and any configuration specified in the Azure provisioning API is ignored.

NOTE

The `Provisioning.Enabled` parameter defaults to "n" on Ubuntu Cloud Images that use cloud-init for provisioning.

Provisioning.DeleteRootPassword:

Type: Boolean
Default: n

If set, the root password in the /etc/shadow file is erased during the provisioning process.

Provisioning.RegenerateSshHostKeyPair:

Type: Boolean
Default: y

If set, all SSH host key pairs (ecdsa, dsa, and rsa) are deleted during the provisioning process from /etc/ssh/. And a single fresh key pair is generated.

The encryption type for the fresh key pair is configurable by the `Provisioning.SshHostKeyPairType` entry. Some distributions re-create SSH key pairs for any missing encryption types when the SSH daemon is restarted (for example, upon a reboot).

Provisioning.SshHostKeyPairType:

Type: String
Default: rsa

This can be set to an encryption algorithm type that is supported by the SSH daemon on the virtual machine. The typically supported values are "rsa", "dsa" and "ecdsa". "putty.exe" on Windows does not support "ecdsa". So, if you intend to use putty.exe on Windows to connect to a Linux deployment, use "rsa" or "dsa".

Provisioning.MonitorHostName:

Type: Boolean
Default: y

If set, waagent monitors the Linux virtual machine for hostname changes (as returned by the "hostname" command) and automatically update the networking configuration in the image to reflect the change. In order to push the name change to the DNS servers, networking is restarted in the virtual machine. This results in brief loss of Internet connectivity.

Provisioning.DecodeCustomData

Type: Boolean
Default: n

If set, waagent decodes CustomData from Base64.

Provisioning.ExecuteCustomData

Type: Boolean
Default: n

If set, waagent executes CustomData after provisioning.

Provisioning.AllowResetSysUser

Type: Boolean
Default: n

This option allows the password for the sys user to be reset; default is disabled.

Provisioning.PasswordCryptId

Type: String
Default: 6

Algorithm used by crypt when generating password hash.

- 1 - MD5
- 2a - Blowfish
- 5 - SHA-256
- 6 - SHA-512

Provisioning.PasswordCryptSaltLength

Type: String
Default: 10

Length of random salt used when generating password hash.

ResourceDisk.Format:

Type: Boolean
Default: y

If set, the resource disk provided by the platform is formatted and mounted by waagent if the filesystem type requested by the user in "ResourceDisk.Filesystem" is anything other than "ntfs". A single partition of type Linux (83) is made available on the disk. This partition is not formatted if it can be successfully mounted.

ResourceDisk.Filesystem:

Type: String
Default: ext4

This specifies the filesystem type for the resource disk. Supported values vary by Linux distribution. If the string is X, then mkfs.X should be present on the Linux image. SLES 11 images should typically use 'ext3'. FreeBSD images should use 'ufs2' here.

ResourceDisk.MountPoint:

```
Type: String  
Default: /mnt/resource
```

This specifies the path at which the resource disk is mounted. The resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned.

ResourceDisk.MountOptions

```
Type: String  
Default: None
```

Specifies disk mount options to be passed to the mount -o command. This is a comma-separated list of values, ex. 'nodev,nosuid'. See mount(8) for details.

ResourceDisk.EnableSwap:

```
Type: Boolean  
Default: n
```

If set, a swap file (/swapfile) is created on the resource disk and added to the system swap space.

ResourceDisk.SwapSizeMB:

```
Type: Integer  
Default: 0
```

The size of the swap file in megabytes.

Logs.Verbose:

```
Type: Boolean  
Default: n
```

If set, log verbosity is boosted. Waagent logs to /var/log/waagent.log and utilizes the system logrotate functionality to rotate logs.

OS.EnableRDMA

```
Type: Boolean  
Default: n
```

If set, the agent attempts to install and then load an RDMA kernel driver that matches the version of the firmware on the underlying hardware.

OS.RootDeviceScsiTimeout:

Type: Integer
Default: 300

This setting configures the SCSI timeout in seconds on the OS disk and data drives. If not set, the system defaults are used.

OS.OpensslPath:

Type: String
Default: None

This setting can be used to specify an alternate path for the openssl binary to use for cryptographic operations.

HttpProxy.Host, HttpProxy.Port

Type: String
Default: None

If set, the agent uses this proxy server to access the internet.

AutoUpdate.Enabled

Type: Boolean
Default: y

Enable or disable auto-update for goal state processing; default is enabled.

Ubuntu Cloud Images

Ubuntu Cloud Images utilize [cloud-init](#) to perform many configuration tasks that would otherwise be managed by the Azure Linux Agent. The following differences apply:

- **Provisioning.Enabled** defaults to "n" on Ubuntu Cloud Images that use cloud-init to perform provisioning tasks.
- The following configuration parameters have no effect on Ubuntu Cloud Images that use cloud-init to manage the resource disk and swap space:
 - **ResourceDisk.Format**
 - **ResourceDisk.Filesystem**
 - **ResourceDisk.MountPoint**
 - **ResourceDisk.EnableSwap**
 - **ResourceDisk.SwapSizeMB**
- For more information, see the following resources to configure the resource disk mount point and swap space on Ubuntu Cloud Images during provisioning:
 - [Ubuntu Wiki: Configure Swap Partitions](#)
 - [Injecting Custom Data into an Azure Virtual Machine](#)

2 minutes to read

Handling planned maintenance notifications for Linux virtual machines

5/7/2019 • 11 minutes to read • [Edit Online](#)

Azure periodically performs updates to improve the reliability, performance, and security of the host infrastructure for virtual machines. Updates are changes like patching the hosting environment or upgrading and decommissioning hardware. A majority of these updates are performed without any impact to the hosted virtual machines. However, there are cases where updates do have an impact:

- If the maintenance does not require a reboot, Azure uses in-place migration to pause the VM while the host is updated. These non-rebootful maintenance operations are applied fault domain by fault domain, and 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 30 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 subscription owner and co-owners. You can add more recipients and messaging options like email, SMS, and webhooks, to the notifications using Azure [Activity Log Alerts](#).
- At the time of the notification, a *self-service window* is made available. During this window that is typically 30 days, you can find which of your virtual machines are included in this wave and 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 virtual machine.

The goal in having two windows is to give you enough time to start maintenance and reboot your virtual machine while knowing when Azure will automatically start maintenance.

You can use the Azure portal, PowerShell, REST API, and CLI to query for the maintenance windows for your VMs and start self-service maintenance.

Should you start maintenance using during the self-service window?

The following guidelines should help you decide whether to use this capability and start maintenance at your own time.

NOTE

Self-service maintenance might not be available for all of your VMs. To determine if proactive redeploy is available for your VM, look for the **Start now** in the maintenance status. Self-service maintenance is currently not available for Cloud Services (Web/Worker Role) and Service Fabric.

Self-service maintenance is not recommended for deployments using **availability sets** since these are highly available setups, where only one update domain is impacted at any given time.

- Let Azure trigger the maintenance. For maintenance that requires reboot, be aware that the maintenance will be done update domain by update domain, that the update domains do not necessarily receive the

maintenance sequentially, and that there is a 30-minute pause between update domains.

- If a temporary loss of some of your capacity (1/update domain count) is a concern, it can easily be compensated for by allocating additional instances during the maintenance period.
- For maintenance that does not require 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, using DevTest Labs, using auto-shutdown, or following a schedule, it could revert the maintenance status and therefore 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 is desired to be maintained upon update.
- For cases where you resize your VM often, as it could revert the maintenance status.
- If you have adopted scheduled events that enable proactive failover or graceful shutdown of your workload, 15 minutes before start of maintenance shutdown

Use self-service maintenance, if you are planning to run your VM uninterrupted during the scheduled maintenance phase and none of the counter-indications mentioned above are applicable.

It is best to use self-service maintenance in the following cases:

- You need to communicate an exact maintenance window to your management or end-customer.
- You need to complete the maintenance by a given date.
- You need to control the sequence of maintenance, for example, multi-tier application to guarantee safe recovery.
- You need more than 30 minutes of VM recovery time between two update domains (UDs). To control the time between update domains, you must trigger maintenance on your VMs one update domain (UD) at a time.

Find VMs scheduled for maintenance using CLI

Planned maintenance information can be seen using [az vm get-instance-view](#).

Maintenance information is returned only if there is maintenance planned. If there is no maintenance scheduled that impacts the VM, the command does not return any maintenance information.

```
az vm get-instance-view -g rgName -n vmName
```

The following values are returned under MaintenanceRedeployStatus:

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 window in which Azure initiates maintenance on your VM

VALUE	DESCRIPTION
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 using CLI

The following call will initiate maintenance on a VM if `IsCustomerInitiatedMaintenanceAllowed` is set to true.

```
az vm perform-maintenance -g rgName -n vmName
```

View VMs scheduled for maintenance in the portal

Once a planned maintenance wave is scheduled, you can observe the list of virtual machines that are impacted by the upcoming maintenance wave.

You can use the Azure portal and look for VMs scheduled for maintenance.

1. Sign in to the [Azure portal](#).
2. In the left navigation, click **Virtual Machines**.
3. In the Virtual Machines pane, click the **Columns** button to open the list of available columns.
4. Select and add the following columns:

Maintenance: Shows the maintenance status for the VM. The following are the potential values:

VALUE	DESCRIPTION
Start now	The VM is in the self-service maintenance window that lets you initiate the maintenance yourself. See below on how to start maintenance on your VM.
Scheduled	The VM is scheduled for maintenance with no option for you to initiate maintenance. You can learn of the maintenance window by selecting the Maintenance - Scheduled window in this view or by clicking on the VM.
Already updated	Your VM is already updated and no further action is required at this time.
Retry later	You have initiated maintenance with no success. You will be able to use the self-service maintenance option at a later time.
Retry now	You can retry a previously unsuccessful self-initiated maintenance.
-	Your VM is not part of a planned maintenance wave.

Maintenance - Self-service window: Shows the time window when you can self-start maintenance on your VMs.

Maintenance - Scheduled window: Shows the time window when Azure will maintain your VM in order to complete maintenance.

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 additional recipients and channels to this communication by creating Azure activity log alerts. For more information, see [Create activity log alerts on service notifications](#).

Make sure you set the **Event type** as **Planned maintenance** and **Services** as **Virtual Machine Scale Sets** and/or **Virtual Machines**

Start Maintenance on your VM from the portal

While looking at the VM details, you will be able to see more maintenance-related details.

At the top of the VM details view, a new notification ribbon will be added if your VM is included in a planned maintenance wave. In addition, a new option is added to start maintenance when possible.

Click on the maintenance notification to see the maintenance page with more details on the planned maintenance. From there, you will be able to **start maintenance** on your VM.

Once you start maintenance, your virtual machine will be maintained and the maintenance status will be updated to reflect the result within few minutes.

If you missed the self-service window, you will still be able to see the window when your VM will be maintained by Azure.

Classic deployments

If you still have legacy VMs that were deployed using the classic deployment model, you can use the Azure classic CLI to query for VMs and initiate maintenance.

Make sure you are in the correct mode to work with classic VM by typing:

```
azure config mode asm
```

To get the maintenance status of a VM named *myVM*, type:

```
azure vm show myVM
```

To start maintenance on your classic VM named *myVM* in the *myService* service and *myDeployment* deployment, type:

```
azure compute virtual-machine initiate-maintenance --service-name myService --name myDeployment --virtual-machine-name myVM
```

FAQ

Q: Why do you need to reboot my virtual machines now?

A: While the majority of updates and upgrades to the Azure platform do not impact virtual machine's availability, there are cases where we can't avoid rebooting virtual machines hosted in Azure. We have accumulated several changes that require us to restart our servers that will result in virtual machines 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 virtual machine scale sets have the notion of Update Domains (UD). When performing maintenance, Azure honors the UD constraint and will not reboot virtual machines from different UD (within the same availability set). Azure also waits for at least 30 minutes before moving to the next group of virtual machines.

For more information about high availability, see [Regions and availability for virtual machines in Azure](#).

Q: How do I get 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). Additional channels and recipients for this notification could be configured using Activity Log Alerts. In case you deploy a virtual machine to a region where planned maintenance is already scheduled, you will not receive the notification but rather need to check the maintenance state of the VM.

Q: I don't see any indication of planned maintenance in the portal, Powershell, or CLI. What is wrong?

A: Information related to planned maintenance is available during a planned maintenance wave only for the VMs that are going to be impacted by it. In other words, if you see no data, it could be that the maintenance wave has already completed (or not started) or that your virtual machine is already hosted in an updated server.

Q: Is there a way to know exactly when my virtual machine will be impacted?

A: When setting the schedule, we define a time window of several days. However, the exact sequencing of servers (and VMs) within this window is unknown. Customers who would like to know the exact time for their VMs can use [scheduled events](#) and query from within the virtual machine and receive a 15-minute notification before a VM reboot.

Q: How long will it take you to reboot my virtual machine?

A: Depending on the size of your VM, reboot may take up to several minutes during the self-service maintenance window. During the Azure initiated reboots in the scheduled maintenance window, the reboot will typically take about 25 minutes. Note that in case you use Cloud Services (Web/Worker Role), Virtual Machine Scale Sets, or availability sets, you will be given 30 minutes between each group of VMs (UD) during the scheduled maintenance window.

Q: What is the experience in the case of Virtual Machine Scale Sets?

A: Planned maintenance is now available for Virtual Machine Scale Sets. For instructions on how to initiate self-service maintenance refer [planned maintenance for VMSS](#) document.

Q: What is the experience in the case of Cloud Services (Web/Worker Role) and Service Fabric?

A: While these platforms are impacted by planned maintenance, customers using these platforms are considered safe given that only VMs in a single Upgrade Domain (UD) will be impacted at any given time. Self-service maintenance is currently not available for Cloud Services (Web/Worker Role) and Service Fabric.

Q: I don't see any maintenance information on my VMs. What went wrong?

A: There are several reasons why you're not seeing any maintenance information on your VMs:

1. You are using a subscription marked as Microsoft internal.
2. Your VMs are not scheduled for maintenance. It could be that the maintenance wave has ended, canceled, or modified so that your VMs are no longer impacted by it.
3. You don't have the **Maintenance** column added to your VM list view. While we have added this column to the default view, customers who configured to see non-default columns must manually add the **Maintenance** column to their VM list view.

Q: My VM is scheduled for maintenance for the second time. Why?

A: There are several use cases where you will see your VM scheduled for maintenance after you have already completed your maintenance-redeploy:

1. We have canceled the maintenance wave and restarted it with a different payload. It could be that we've detected faulted payload and we simply need to deploy an additional payload.
2. Your VM was *service healed* to another node due to a hardware fault.
3. You have selected to stop (deallocate) and restart the VM.
4. You have **auto shutdown** turned on for the VM.

Next steps

Learn how you can register for maintenance events from within the VM using [Scheduled Events](#).

Guidance for mitigating speculative execution side-channel vulnerabilities in Azure

6/6/2019 • 7 minutes to read • [Edit Online](#)

Last document update: 4 June 2019 3:00 PM PST.

The disclosure of a [new class of CPU vulnerabilities](#) known as speculative execution side-channel attacks has resulted in questions from customers seeking more clarity.

Microsoft has deployed mitigations across all our cloud services. The infrastructure that runs Azure and isolates customer workloads from each other is protected. This means that a potential attacker using the same infrastructure can't attack your application using these vulnerabilities.

Azure is using [memory preserving maintenance](#) whenever possible, to minimize customer impact and eliminate the need for reboots. Azure will continue utilizing these methods when making systemwide updates to the host and protect our customers.

More information about how security is integrated into every aspect of Azure is available on the [Azure Security Documentation](#) site.

NOTE

Since this document was first published, multiple variants of this vulnerability class have been disclosed. Microsoft continues to be heavily invested in protecting our customers and providing guidance. This page will be updated as we continue to release further fixes.

On May 14, 2019, [Intel disclosed](#) a new set of speculative execution side channel vulnerability known as Microarchitectural Data Sampling (MDS) see the Microsoft Security Guidance [ADV190013](#)), which has been assigned multiple CVEs:

- CVE-2019-11091 - Microarchitectural Data Sampling Uncacheable Memory (MDSUM)
- CVE-2018-12126 - Microarchitectural Store Buffer Data Sampling (MSBDS)
- CVE-2018-12127 - Microarchitectural Load Port Data Sampling (MLPDS)
- CVE-2018-12130 - Microarchitectural Fill Buffer Data Sampling (MFBDS)

This vulnerability affects Intel® Core® processors and Intel® Xeon® processors. Microsoft Azure has released operating system updates and is deploying new microcode, as it is made available by Intel, throughout our fleet to protect our customers against these new vulnerabilities. Azure is closely working with Intel to test and validate the new microcode prior to its official release on the platform.

Customers that are running untrusted code within their VM need to take action to protect against these vulnerabilities by reading below for additional guidance on all speculative execution side-channel vulnerabilities (Microsoft Advisories [ADV180002](#), [180018](#), and [190013](#)).

Other customers should evaluate these vulnerabilities from a Defense in Depth perspective and consider the security and performance implications of their chosen configuration.

Keeping your operating systems up-to-date

While an OS update is not required to isolate your applications running on Azure from other Azure customers, it is always a best practice to keep your software up-to-date. The latest Security Rollups for Windows contain mitigations for several speculative execution side channel vulnerabilities. Similarly, Linux distributions have released multiple updates to address these vulnerabilities. Here are our recommended actions to update your operating system:

OFFERING	RECOMMENDED ACTION
Azure Cloud Services	Enable auto update or ensure you are running the newest Guest OS.
Azure Linux Virtual Machines	Install updates from your operating system provider. For more information, see Linux later in this document.
Azure Windows Virtual Machines	Install the latest security rollup.
Other Azure PaaS Services	There is no action needed for customers using these services. Azure automatically keeps your OS versions up-to-date.

Additional guidance if you are running untrusted code

Customers who allow untrusted users to execute arbitrary code may wish to implement some additional security features inside their Azure Virtual Machines or Cloud Services. These features protect against the intra-process disclosure vectors that several speculative execution vulnerabilities describe.

Example scenarios where additional security features are recommended:

- You allow code that you do not trust to run inside your VM.
 - *For example, you allow one of your customers to upload a binary or script that you then execute within your application.*
- You allow users that you do not trust to log into your VM using low privileged accounts.
 - *For example, you allow a low-privileged user to log into one of your VMs using remote desktop or SSH.*
- You allow untrusted users access to virtual machines implemented via nested virtualization.
 - *For example, you control the Hyper-V host, but allocate the VMs to untrusted users.*

Customers who do not implement a scenario involving untrusted code do not need to enable these additional security features.

Enabling additional security

You can enable additional security features inside your VM or Cloud Service if you are running untrusted code. In parallel, ensure your operating system is up-to-date to enable security features inside your VM or Cloud Service

Windows

Your target operating system must be up-to-date to enable these additional security features. While numerous speculative execution side channel mitigations are enabled by default, the additional features described here must be enabled manually and may cause a performance impact.

Step 1: Disable hyper-threading on the VM - Customers running untrusted code on a hyper-threaded VM will need to disable hyper-threading or move to a non-hyper-threaded VM size. Reference [this doc](#) for a list of hyper-threaded VM sizes (where ratio of vCPU to Core is 2:1). To check if your VM has hyper-threading enabled, please refer to the below script using the Windows command line from within the VM.

Type `wmic` to enter the interactive interface. Then type the below to view the amount of physical and logical processors on the VM.

```
CPU Get NumberOfCores,NumberOfLogicalProcessors /Format:List
```

If the number of logical processors is greater than physical processors (cores), then hyper-threading is enabled. If you are running a hyper-threaded VM, please [contact Azure Support](#) to get hyper-threading disabled. Once hyper-

threading is disabled, **support will require a full VM reboot**. Please refer to [Core count](#) to understand why your VM core count decreased.

Step 2: In parallel to Step 1, follow the instructions in [KB4072698](#) to verify protections are enabled using the [SpeculationControl](#) PowerShell module.

NOTE

If you previously downloaded this module, you will need to install the newest version.

The output of the PowerShell script should have the below values to validate enabled protections against these vulnerabilities:

```
Windows OS support for branch target injection mitigation is enabled: True  
Windows OS support for kernel VA shadow is enabled: True  
Windows OS support for speculative store bypass disable is enabled system-wide: False  
Windows OS support for L1 terminal fault mitigation is enabled: True  
Windows OS support for MDS mitigation is enabled: True
```

If the output shows `MDS mitigation is enabled: False`, please [contact Azure Support](#) for available mitigation options.

Step 3: To enable Kernel Virtual Address Shadowing (KVAS) and Branch Target Injection (BTI) OS support, follow the instructions in [KB4072698](#) to enable protections using the [Session Manager](#) registry keys. A reboot is required.

Step 4: For deployments that are using [nested virtualization](#) (D3 and E3 only): These instructions apply inside the VM you are using as a Hyper-V host.

1. Follow the instructions in [KB4072698](#) to enable protections using the [MinVmVersionForCpuBasedMitigations](#) registry keys.
2. Set the hypervisor scheduler type to `Core` by following the instructions [here](#).

Linux

Enabling the set of additional security features inside requires that the target operating system be fully up-to-date. Some mitigations will be enabled by default. The following section describes the features which are off by default and/or reliant on hardware support (microcode). Enabling these features may cause a performance impact. Reference your operating system provider's documentation for further instructions

Step 1: Disable hyper-threading on the VM - Customers running untrusted code on a hyper-threaded VM will need to disable hyper-threading or move to a non-hyper-threaded VM. Reference [this doc](#) for a list of hyper-threaded VM sizes (where ratio of vCPU to Core is 2:1). To check if you are running a hyper-threaded VM, run the `lscpu` command in the Linux VM.

If `Thread(s) per core = 2`, then hyper-threading has been enabled.

If `Thread(s) per core = 1`, then hyper-threading has been disabled.

Sample output for a VM with hyper-threading enabled:

CPU Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
CPU(s):	8
On-line CPU(s) list:	0-7
Thread(s) per core:	2
Core(s) per socket:	4
Socket(s):	1
NUMA node(s):	1

If you are running a hyper-threaded VM, please [contact Azure Support](#) to get hyper-threading disabled. Once hyper-threading is disabled, **support will require a full VM reboot**. Please refer to [Core count](#) to understand why your VM core count decreased.

Step 2: To mitigate against any of the below speculative execution side-channel vulnerabilities, refer to your operating system provider's documentation:

- [Redhat and CentOS](#)
- [SUSE](#)
- [Ubuntu](#)

Core count

When a hyper-threaded VM is created, Azure allocates 2 threads per core - these are called vCPUs. When hyper-threading is disabled, Azure removes a thread and surfaces up single threaded cores (physical cores). The ratio of vCPU to CPU is 2:1, so once hyper-threading is disabled, the CPU count in the VM will appear to have decreased by half. For example, a D8_v3 VM is a hyper-threaded VM running on 8 vCPUs (2 threads per core x 4 cores).

When hyper-threading is disabled, CPUs will drop to 4 physical cores with 1 thread per core.

Next steps

This article provides guidance to the below speculative execution side-channel attacks that affect many modern processors:

Spectre Meltdown:

- CVE-2017-5715 - Branch Target Injection (BTI)
- CVE-2017-5754 - Kernel Page Table Isolation (KPTI)
- CVE-2018-3639 – Speculative Store Bypass (KPTI)

L1 Terminal Fault (L1TF):

- CVE-2018-3615 - Intel Software Guard Extensions (Intel SGX)
- CVE-2018-3620 - Operating Systems (OS) and System Management Mode (SMM)
- CVE-2018-3646 – impacts Virtual Machine Manager (VMM)

Microarchitectural Data Sampling:

- CVE-2019-11091 - Microarchitectural Data Sampling Uncacheable Memory (MDSUM)
- CVE-2018-12126 - Microarchitectural Store Buffer Data Sampling (MSBDS)
- CVE-2018-12127 - Microarchitectural Load Port Data Sampling (MLPDS)
- CVE-2018-12130 - Microarchitectural Fill Buffer Data Sampling (MFBDS)

Azure Metadata Service: Scheduled Events for Linux VMs

5/9/2019 • 6 minutes to read • [Edit Online](#)

Scheduled Events is an Azure Metadata Service that gives your application time to prepare for virtual machine (VM) maintenance. It provides information about upcoming maintenance events (for example, reboot) so that your application can prepare for them and limit disruption. It's available for all Azure Virtual Machines types, including PaaS and IaaS on both Windows and Linux.

For information about Scheduled Events on Windows, see [Scheduled Events for Windows VMs](#).

NOTE

Scheduled Events is generally available in all Azure Regions. See [Version and Region Availability](#) for latest release information.

Why use Scheduled Events?

Many applications can benefit from time to prepare for VM maintenance. The time can be used to perform application-specific tasks that improve availability, reliability, and serviceability, including:

- Checkpoint and restore.
- Connection draining.
- Primary replica failover.
- Removal from a load balancer pool.
- Event logging.
- Graceful shutdown.

With Scheduled Events, your application can discover when maintenance will occur and trigger tasks to limit its impact.

Scheduled Events provides events in the following use cases:

- [Platform initiated maintenance](#) (for example, VM reboot, live migration or memory preserving updates for host)
- Degraded hardware
- User-initiated maintenance (for example, a user restarts or redeploys a VM)
- [Low-Priority VM eviction](#) in scale sets

The Basics

Metadata Service exposes information about running VMs by using a REST endpoint that's accessible from within the VM. The information is available via a nonroutable IP so that it's not exposed outside the VM.

Scope

Scheduled events are delivered to:

- Standalone Virtual Machines.
- All the VMs in a cloud service.

- All the VMs in an availability set.
- All the VMs in a scale set placement group.

As a result, check the `Resources` field in the event to identify which VMs are affected.

Endpoint Discovery

For VNET enabled VMs, Metadata Service is available from a static nonroutable 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=2017-11-01
```

If the VM is not created within a Virtual Network, the default cases for cloud services and classic VMs, additional logic is required to discover the IP address to use. To learn how to [discover the host endpoint](#), see this sample.

Version and Region Availability

The Scheduled Events service is versioned. Versions are mandatory; the current version is `2017-11-01`.

VERSION	RELEASE TYPE	REGIONS	RELEASE NOTES
2017-11-01	General Availability	All	<ul style="list-style-type: none"> Added support for Low-priority VM eviction EventType 'Preempt'
2017-08-01	General Availability	All	<ul style="list-style-type: none"> Removed prepended underscore from resource names for IaaS VMs Metadata header requirement enforced for all requests
2017-03-01	Preview	All	<ul style="list-style-type: none"> Initial release

NOTE

Previous preview releases of Scheduled Events supported `{latest}` as the api-version. This format is no longer supported and will be deprecated in the future.

Enabling and Disabling Scheduled Events

Scheduled Events is enabled for your service the first time you make a request for events. You should expect a delayed response in your first call of up to two minutes.

Scheduled Events is disabled for your service if it does not make a request for 24 hours.

User-initiated Maintenance

User-initiated VM maintenance via the Azure portal, API, CLI, or PowerShell results in a scheduled event. You then can test the maintenance preparation logic in your application, and your application can prepare for user-initiated maintenance.

If you restart a VM, an event with the type `Reboot` is scheduled. If you redeploy a VM, an event with the type `Redeploy` is scheduled.

Use the API

Headers

When you query Metadata Service, you must provide the header `Metadata:true` to ensure the request wasn't unintentionally redirected. The `Metadata:true` header is required for all scheduled events requests. Failure to include the header in the request results in a "Bad Request" response from Metadata Service.

Query for events

You can query for scheduled events by making the following call:

Bash

```
curl -H Metadata:true http://169.254.169.254/metadata/scheduledevents?api-version=2017-08-01
```

A response contains an array of scheduled events. An empty array means that currently no events are scheduled. In the case where there are scheduled events, the response contains an array of events.

```
{
  "DocumentIncarnation": {IncarnationID},
  "Events": [
    {
      "EventId": {eventID},
      "EventType": "Reboot" | "Redeploy" | "Freeze" | "Preempt",
      "ResourceType": "VirtualMachine",
      "Resources": [{resourceName}],
      "EventStatus": "Scheduled" | "Started",
      "NotBefore": {timeInUTC},
    }
  ]
}
```

Event Properties

PROPERTY	DESCRIPTION
EventId	Globally unique identifier for this event. Example: <ul style="list-style-type: none">• 602d9444-d2cd-49c7-8624-8643e7171297
EventType	Impact this event causes. Values: <ul style="list-style-type: none">• <code>Freeze</code> : The Virtual Machine is scheduled to pause for a few seconds. CPU and network connectivity may be suspended, but there is no impact on memory or open files.• <code>Reboot</code> : The Virtual Machine is scheduled for reboot (non-persistent memory is lost).• <code>Redeploy</code> : The Virtual Machine is scheduled to move to another node (ephemeral disks are lost).• <code>Preempt</code> : The Low-priority Virtual Machine is being deleted (ephemeral disks are lost).
ResourceType	Type of resource this event affects. Values: <ul style="list-style-type: none">• <code>VirtualMachine</code>

PROPERTY	DESCRIPTION
Resources	<p>List of resources this event affects. The list is guaranteed to contain machines from at most one update domain, but it might not contain all machines in the UD.</p> <p>Example:</p> <ul style="list-style-type: none"> • ["FrontEnd_IN_0", "BackEnd_IN_0"]
EventStatus	<p>Status of this event.</p> <p>Values:</p> <ul style="list-style-type: none"> • <code>Scheduled</code> : This event is scheduled to start after the time specified in the <code>NotBefore</code> property. • <code>Started</code> : This event has started. <p>No <code>Completed</code> or similar status is ever provided. The event is no longer returned when the event is finished.</p>
NotBefore	<p>Time after which this event can start.</p> <p>Example:</p> <ul style="list-style-type: none"> • Mon, 19 Sep 2016 18:29:47 GMT

Event Scheduling

Each event is scheduled a minimum amount of time in the future based on the event type. This time is reflected in an event's `NotBefore` property.

EVENT TYPE	MINIMUM NOTICE
Freeze	15 minutes
Reboot	15 minutes
Redeploy	10 minutes
Preempt	30 seconds

Start an event

After you learn of an upcoming event and finish your logic for graceful shutdown, you can approve the outstanding event by making a `POST` call to Metadata Service with `EventId`. This call indicates to Azure that it can shorten the minimum notification time (when possible).

The following JSON sample is expected in the `POST` request body. The request should contain a list of `StartRequests`. Each `StartRequest` contains `EventId` for the event you want to expedite:

```
{
  "StartRequests" : [
    {
      "EventId": {EventId}
    }
  ]
}
```

Bash sample

```
curl -H Metadata:true -X POST -d '{"StartRequests": [{"EventId": "f020ba2e-3bc0-4c40-a10b-86575a9eabd5"}]}'  
http://169.254.169.254/metadata/scheduledevents?api-version=2017-11-01
```

NOTE

Acknowledging an event allows the event to proceed for all `Resources` in the event, not just the VM that acknowledges the event. Therefore, you can choose to elect a leader to coordinate the acknowledgement, which might be as simple as the first machine in the `Resources` field.

Python sample

The following sample queries Metadata Service for scheduled events and approves each outstanding event:

```
#!/usr/bin/python

import json
import urllib2
import socket
import sys

metadata_url = "http://169.254.169.254/metadata/scheduledevents?api-version=2017-11-01"
headers = "{Metadata:true}"
this_host = socket.gethostname()

def get_scheduled_events():
    req = urllib2.Request(metadata_url)
    req.add_header('Metadata', 'true')
    resp = urllib2.urlopen(req)
    data = json.loads(resp.read())
    return data

def handle_scheduled_events(data):
    for evt in data['Events']:
        eventid = evt['EventId']
        status = evt['EventStatus']
        resources = evt['Resources']
        eventtype = evt['EventType']
        resourcetype = evt['ResourceType']
        notbefore = evt['NotBefore'].replace(" ", "_")
        if this_host in resources:
            print "+ Scheduled Event. This host " + this_host + " is scheduled for " + eventtype + " not before " + notbefore
            # Add logic for handling events here

def main():
    data = get_scheduled_events()
    handle_scheduled_events(data)

if __name__ == '__main__':
    main()
    sys.exit(0)
```

Next steps

- Watch [Scheduled Events on Azure Friday](#) to see a demo.
- Review the Scheduled Events code samples in the [Azure Instance Metadata Scheduled Events GitHub repository](#).
- Read more about the APIs that are available in the [Instance Metadata Service](#).

- Learn about [planned maintenance for Linux virtual machines in Azure](#).

Azure Instance Metadata service

6/26/2019 • 16 minutes to read • [Edit Online](#)

The Azure Instance Metadata Service provides information about running virtual machine instances that can be used to manage and configure your virtual machines. This includes information such as SKU, network configuration, and upcoming maintenance events. For more information on what type of information is available, see [metadata APIs](#).

Azure's Instance Metadata Service is a REST Endpoint accessible to all IaaS VMs created via the [Azure Resource Manager](#). The endpoint is available at a well-known non-routable IP address (`169.254.169.254`) that can be accessed only from within the VM.

IMPORTANT

This service is **generally available** in all Azure Regions. It regularly receives updates to expose new information about virtual machine instances. This page reflects the up-to-date [metadata APIs](#) available.

Service availability

The service is available in generally available Azure regions. Not all API version may be available in all Azure Regions.

REGIONS	AVAILABILITY?	SUPPORTED VERSIONS
All Generally Available Global Azure Regions	Generally Available	2017-04-02, 2017-08-01, 2017-12-01, 2018-02-01, 2018-04-02, 2018-10-01
Azure Government	Generally Available	2017-04-02, 2017-08-01, 2017-12-01, 2018-02-01, 2018-04-02, 2018-10-01
Azure China	Generally Available	2017-04-02, 2017-08-01, 2017-12-01, 2018-02-01, 2018-04-02, 2018-10-01
Azure Germany	Generally Available	2017-04-02, 2017-08-01, 2017-12-01, 2018-02-01, 2018-04-02, 2018-10-01
Public West Central US	Generally Available	2017-04-02, 2017-08-01, 2017-12-01, 2018-02-01, 2018-04-02, 2018-10-01, 2019-02-01

This table is updated when there are service updates and or new supported versions are available.

NOTE

2019-02-01 is currently getting rolled out and will be available in other regions shortly.

To try out the Instance Metadata Service, create a VM from [Azure Resource Manager](#) or the [Azure portal](#) in the above regions and follow the examples below.

Usage

Versioning

The Instance Metadata Service is versioned. Versions are mandatory and the current version on Global Azure is `2018-10-01`. Current supported versions are (2017-04-02, 2017-08-01, 2017-12-01, 2018-02-01, 2018-04-02, 2018-10-01).

As newer versions are added, older versions can still be accessed for compatibility if your scripts have dependencies on specific data formats.

When no version is specified, an error is returned with a list of the newest supported versions.

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance"
```

Response

```
{
  "error": "Bad request. api-version was not specified in the request. For more information refer to aka.ms/azureimds",
  "newest-versions": [
    "2018-10-01",
    "2018-04-02",
    "2018-02-01"
  ]
}
```

Using headers

When you query the Instance Metadata Service, you must provide the header `Metadata: true` to ensure the request was not unintentionally redirected.

Retrieving metadata

Instance metadata is available for running VMs created/managed using [Azure Resource Manager](#). Access all data categories for a virtual machine instance using the following request:

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance?api-version=2017-08-01"
```

NOTE

All instance metadata queries are case-sensitive.

Data output

By default, the Instance Metadata Service returns data in JSON format (`Content-Type: application/json`).

However, different APIs return data in different formats if requested. The following table is a reference of other data formats APIs may support.

API	DEFAULT DATA FORMAT	OTHER FORMATS
/instance	json	text

API	DEFAULT DATA FORMAT	OTHER FORMATS
/scheduledevents	json	none
/attested	json	none

To access a non-default response format, specify the requested format as a query string parameter in the request.

For example:

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance?api-version=2017-08-01&format=text"
```

NOTE

For leaf nodes the `format=json` doesn't work. For these queries `format=text` needs to be explicitly specified if the default format is json.

Security

The Instance Metadata Service endpoint is accessible only from within the running virtual machine instance on a non-routable IP address. In addition, any request with a `X-Forwarded-For` header is rejected by the service.

Requests must also contain a `Metadata: true` header to ensure that the actual request was directly intended and not a part of unintentional redirection.

Error

If there is a data element not found or a malformed request, the Instance Metadata Service returns standard HTTP errors. For example:

HTTP STATUS CODE	REASON
200 OK	
400 Bad Request	Missing <code>Metadata: true</code> header or missing the format when querying a leaf node
404 Not Found	The requested element doesn't exist
405 Method Not Allowed	Only <code>GET</code> and <code>POST</code> requests are supported
429 Too Many Requests	The API currently supports a maximum of 5 queries per second
500 Service Error	Retry after some time

Examples

NOTE

All API responses are JSON strings. All following example responses are pretty-printed for readability.

Retrieving network information

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance/network?api-version=2017-08-01"
```

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{
  "interface": [
    {
      "ipv4": {
        "ipAddress": [
          {
            "privateIpAddress": "10.1.0.4",
            "publicIpAddress": "X.X.X.X"
          }
        ],
        "subnet": [
          {
            "address": "10.1.0.0",
            "prefix": "24"
          }
        ]
      },
      "ipv6": {
        "ipAddress": []
      },
      "macAddress": "000D3AF806EC"
    }
  ]
}
```

Retrieving public IP address

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance/network/interface/0/ipv4/ipAddress/0/publicIpAddress?api-version=2017-08-01&format=text"
```

Retrieving all metadata for an instance

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance?api-version=2018-10-01"
```

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{
  "compute": {
    "azEnvironment": "AzurePublicCloud",
    "location": "centralus",
    "name": "negasonic",
    "offer": "lampstack",
    "osType": "Linux",
    "placementGroupId": "",
    "plan": {
      "name": "5-6",
      "product": "lampstack",
      "publisher": "bitnami"
    },
    "platformFaultDomain": "0",
    "platformUpdateDomain": "0",
    "provider": "Microsoft.Compute",
    "publicKeys": [],
    "publisher": "bitnami",
    "resourceGroupName": "myrg",
    "sku": "5-6",
    "subscriptionId": "xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx",
    "tags": "Department:IT;Environment:Prod;Role:WorkerRole",
    "version": "7.1.1902271506",
    "vmId": "13f56399-bd52-4150-9748-7190aae1ff21",
    "vmScaleSetName": "",
    "vmSize": "Standard_A1_v2",
    "zone": "1"
  },
  "network": {
    "interface": [
      {
        "ipv4": {
          "ipAddress": [
            {
              "privateIpAddress": "10.1.2.5",
              "publicIpAddress": "X.X.X.X"
            }
          ],
          "subnet": [
            {
              "address": "10.1.2.0",
              "prefix": "24"
            }
          ]
        },
        "ipv6": {
          "ipAddress": []
        },
        "macAddress": "000D3A36DDDE"
      }
    ]
  }
}
```

Retrieving metadata in Windows Virtual Machine

Request

Instance metadata can be retrieved in Windows via the PowerShell utility `curl`:

```
curl -H @{'Metadata'='true'} http://169.254.169.254/metadata/instance?api-version=2018-10-01 | select -ExpandProperty Content
```

Or through the `Invoke-RestMethod` cmdlet:

```
Invoke-RestMethod -Headers @{"Metadata"="true"} -URI http://169.254.169.254/metadata/instance?api-version=2018-10-01 -Method get
```

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{
  "compute": {
    "azEnvironment": "AzurePublicCloud",
    "location": "centralus",
    "name": "negasonic",
    "offer": "lampstack",
    "osType": "Linux",
    "placementGroupId": "",
    "plan": {
      "name": "5-6",
      "product": "lampstack",
      "publisher": "bitnami"
    },
    "platformFaultDomain": "0",
    "platformUpdateDomain": "0",
    "provider": "Microsoft.Compute",
    "publicKeys": [],
    "publisher": "bitnami",
    "resourceGroupName": "myrg",
    "sku": "5-6",
    "subscriptionId": "xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx",
    "tags": "Department:IT;Environment:Test;Role:WebRole",
    "version": "7.1.1902271506",
    "vmId": "13f56399-bd52-4150-9748-7190aae1ff21",
    "vmScaleSetName": "",
    "vmSize": "Standard_A1_v2",
    "zone": "1"
  },
  "network": {
    "interface": [
      {
        "ipv4": {
          "ipAddress": [
            {
              "privateIpAddress": "10.0.1.4",
              "publicIpAddress": "X.X.X.X"
            }
          ],
          "subnet": [
            {
              "address": "10.0.1.0",
              "prefix": "24"
            }
          ]
        },
        "ipv6": {
          "ipAddress": []
        },
        "macAddress": "002248020E1E"
      }
    ]
  }
}
```

Metadata APIs

The following APIs are available through the metadata endpoint:

DATA	DESCRIPTION	VERSION INTRODUCED
attested	See Attested Data	2018-10-01
identity	Managed identities for Azure resources. See acquire an access token	2018-02-01

DATA	DESCRIPTION	VERSION INTRODUCED
instance	See Instance API	2017-04-02
scheduledevents	See Scheduled Events	2017-08-01

Instance API

The following Compute categories are available through the Instance API:

NOTE

Through the metadata endpoint, the following categories are accessed through instance/compute

DATA	DESCRIPTION	VERSION INTRODUCED
azEnvironment	Azure Environment where the VM is running in	2018-10-01
customData	See Custom Data	2019-02-01
location	Azure Region the VM is running in	2017-04-02
name	Name of the VM	2017-04-02
offer	Offer information for the VM image and is only present for images deployed from Azure image gallery	2017-04-02
osType	Linux or Windows	2017-04-02
placementGroupId	Placement Group of your virtual machine scale set	2017-08-01
plan	Plan containing name, product, and publisher for a VM if its an Azure Marketplace Image	2018-04-02
platformUpdateDomain	Update domain the VM is running in	2017-04-02
platformFaultDomain	Fault domain the VM is running in	2017-04-02
provider	Provider of the VM	2018-10-01
publicKeys	Collection of Public Keys assigned to the VM and paths	2018-04-02
publisher	Publisher of the VM image	2017-04-02
resourceGroupName	Resource group for your Virtual Machine	2017-08-01
sku	Specific SKU for the VM image	2017-04-02

DATA	DESCRIPTION	VERSION INTRODUCED
subscriptionId	Azure subscription for the Virtual Machine	2017-08-01
tags	Tags for your Virtual Machine	2017-08-01
version	Version of the VM image	2017-04-02
vmId	Unique identifier for the VM	2017-04-02
vmScaleSetName	Virtual Machine ScaleSet Name of your virtual machine scale set	2017-12-01
vmSize	VM size	2017-04-02
zone	Availability Zone of your virtual machine	2017-12-01

The following Network categories are available through the Instance API:

NOTE

Through the metadata endpoint, the following categories are accessed through instance/network/interface

DATA	DESCRIPTION	VERSION INTRODUCED
ipv4/privateIpAddress	Local IPv4 address of the VM	2017-04-02
ipv4/publicIpAddress	Public IPv4 address of the VM	2017-04-02
subnet/address	Subnet address of the VM	2017-04-02
subnet/prefix	Subnet prefix, example 24	2017-04-02
ipv6/ipAddress	Local IPv6 address of the VM	2017-04-02
macAddress	VM mac address	2017-04-02

Attested Data

Instance Metadata responds at http endpoint on 169.254.169.254. Part of the scenario served by Instance Metadata Service is to provide guarantees that the data responded is coming from Azure. We sign part of this information so that marketplace images can be sure that it's their image running on Azure.

Example Attested Data

NOTE

All API responses are JSON strings. The following example responses are pretty-printed for readability.

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/attested/document?api-version=2018-10-01&nonce=1234567890"
```

Api-version is a mandatory field and the version supported for attested data is 2018-10-01.Nonce is an optional 10-digit string provided.Nonce can be used to track the request and if not provided,in response encoded string the current UTC timestamp is returned.

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{  
  "encoding": "pkcs7", "signature": "MIIEGyJKoZIhvcNAQcCoIIEAzCCA/8CAQExDzANBgkqhkiG9w0BAQsFADCBugYJKoZIhvcNAQcBoIGsB1GpeyJub25jZSI6IjEyMzQ1NjY3NjYiLCJwbGFuIjp7Im5hbWUiOiiLCJwcm9kdWN0IjoiIiwicHVibGlzaGvIjoiIn0sInRpbwTdTfCtI6IejcmVhdGVkT24i0iIxMs8yMC8xOCAyMjowNzozOSAtMDAwMCIsImV4cGlyZXNPbiI6IjExLzIwLzE4IDIy0jA40jI0IC0wMDAwIn0sInZtSWQi0iIifaCCAjwggI7MIIBpKADAgECAhBnxW5Kh8ds1eBA0E2mIBj0MA0GCSqGSIB3DQEBAUAMCsxKTAnBgNVBAMTIHRLc3RzdWJkb21haw4ubW0YWRhdGEuYXp1cmUuY29tMB4XDTE4MTExMDIxNTc1N1oXDTE4MTIyMDIxNTc1N1lowKzEpMCCGA1UEAxMgdGVzdHN1YmRvbWFpbis5tZXRhZGF0YS5henVyZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAML/tBo86ENWPzmXZ0kPKx5dY5QZ150mA81ommszE71x2sCLonzv4/UWk4H+jMMWRwIea2CuQ5RhdWAhvKq6if4okKNT66fxm+YTVz9z0CTfCLmLT+nsdf0AsG1xZppEapC0Cd9vDGNCKyE8ayI1pliaeOnFjG0lwvMY04uWz2MdAgMBAAGjYDBeMFwGA1UdAQRMFOAENyKHLa04Ut4Mp7TkJFFyhLTArMSkwJwYDVQQDEyB0ZXN0c3ViZG9tYWluLm1ldGFkYXRhLmF6dXJlLmNvbYIQZ8VuSoFhbJRAQNBnpiAsdDANBgkqhkiG9w0BAQQAObgQCLSM6aX5Bs1KHCJp4VQtzXPzXF71rVKCocHy3N9PTJQ9Fpnd+bYw2SpQHg/Aig82WuDFpPreJvr7Pa938mZqW9HUOGjQKK2FYDTg6fXD8pkPdygh1X5boGwAMMr7bFkup+lsT+n2tRw2wbNkn01tQ0wICtqy2VqzWwL45RBwTGB6DCB5QIBATA/MCsxKTAnBgNVBAMTIHRLc3RzdWJkb21haw4ubW0YWRhdGEuYXp1cmUuY29tAhBnxW5Kh8ds1eBA0E2mIBj0MA0GCSqGSIB3DQEBCwUAMA0GCSqGSIB3DQEBAQUABIGA1d1BM/yYIqqv8SDE4kjQo3U1/IKAVR8ETKcve5BAdGSNkTUooUGVniTXeuvDj5Nkmaz0aKzp9fEtByqqPOyw/n1XaZg0044HDGiPUJ90xVYmfek6p9RpJBu6kiKhnnYTeluk5u75phe5ZbMzfBhuPhXmYAdjc7Nmw97nx8NnprQ="}
```

The signature blob is a [pkcs7](#) signed version of document. It contains the certificate used for signing along with the VM details like vmid, nonce, timeStamp for creation and expiry of the document and the plan information about the image. The plan information is only populated for Azure Market place images. The certificate can be extracted from the response and used to validate that the response is valid and is coming from Azure.

Retrieving attested metadata in Windows Virtual Machine

Request

Instance metadata can be retrieved in Windows via the PowerShell utility `curl`:

```
curl -H @{"Metadata"='true'} "http://169.254.169.254/metadata/attested/document?api-version=2018-10-01&nonce=1234567890" | select -ExpandProperty Content
```

Or through the `Invoke-RestMethod` cmdlet:

```
Invoke-RestMethod -Headers @{"Metadata"="true"} -URI "http://169.254.169.254/metadata/attested/document?api-version=2018-10-01&nonce=1234567890" -Method get
```

Api-version is a mandatory field and the version supported for attested data is 2018-10-01.Nonce is an optional 10-digit string provided.Nonce can be used to track the request and if not provided,in response encoded string the current UTC timestamp is returned.

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{  
  "encoding": "pkcs7", "signature": "MIIEegYJKoZIhvcNAQCoIIEAzCCA/8CAQExDzANBgkqhkiG9w0BAQsFADCBugYJKoZIhvcNAQcBoIGsB1GpeyJub25jZSI6IjEyMzQ1NjY3NjYiLCJwbGFuIjp7Im5hbWUiOiiLCJwcm9kdWN0IjoiIiwcHvibGlzaGVyIjoiIn0sInRpbtWTdGftcCI6eyJjcmVhdGVkT24i0iIxMS8yMC8xOCAYmjowNzozOSAtMDAwMCIsImV4cGlyZXNPbiI6IjExLzIwLzE4IDIy0ja40jI0IC0wMDAwIn0sInZtSWQiOiiifaCCAj8wggi7MIBpKADAgECAhBnxW5Kh8ds1EBA0E2mIBj0MA0GCSqGSiB3DQEBAUAMCsxKTAnBgNVBAMTIHRlc3RzdWJkb21haW4ubW0YWRhdGEuYXp1cmUuY29tMB4XDTE4MTExNTc1N1oXDE4MTIyMDIxNTc1N1lowKzEpMccGA1UEAxMgdGVzdHN1YmRvbWFpbis5tZXRhZGF0YS5henVyZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAML/tBo86ENWPzmXZ0kPKX5dY5QZ150mA81ommszE71x2sCLonz4/UWk4H+jMMWRwIea2CuQ5RhdWAhvKq6if4okKNt66fxm+YTVz9z0CTfCLmLT+nsdf0AsG1xZppEapC0Cd9vD6NCKyE8aYI1pliaeOnFjG0WvMY04uWz2MdAgMBAAGjYDBeMFwGA1UdAQRVMFOAENnYkHLa04Ut4Mpt7TkJFFyhLTArMSkwJwYDVQQDEyB0ZXN0c3ViZG9tYWluLm1ldGFkYXRhLmF6dXJ1LmNvbYIQZ8VuSoFhbJRAQNBnpiASdDANBgkqhkiG9w0BAQQFAOBgQCLSM6aX5Bs1KHCJp4VQtzXPzXF71rVKCocHy3N9PTJQ9Fpnd+bYw2vSpQHg/Aig82WuDFpPreJvr7Pa938mZq9HUOGjQKK2FYDTg6fXD8pkPdygh1X5boGWAMMrF7bFkup+lsT+n2tRw2wbNkn01tQ0WIctqy2VqzWwLi45RBwTGB6DCB5QIBATA/MCsxKTAnBgNVBAMTIHRlc3RzdWJkb21haW4ubW0YWRhdGEuYXp1cmUuY29tAhBnxW5Kh8ds1EBA0E2mIBj0MA0GCSqGSiB3DQEBCwUAMA0GCSqGSiB3DQEBAQUABIGA1d1BM/yYIqqv8SDE4kjQo3U1/IKAVR8ETKcve5BAdGSNkTUooUGVniTXeuvdj5NKmaz0aKZp9fEtByqqPOyw/n1XaZg0044HDGiPUJ90xVYmfefK6p9RpJBu6kiKhnnYTeluk5u75phe5zbMzfBhuPhXmYAdjc7Nmw97nx8NnprQ="}  
}
```

The signature blob is a [pkcs7](#) signed version of document. It contains the certificate used for signing along with the VM details like vmId, nonce, timeStamp for creation and expiry of the document and the plan information about the image. The plan information is only populated for Azure Market place images. The certificate can be extracted from the response and used to validate that the response is valid and is coming from Azure.

Example scenarios for usage

Tracking VM running on Azure

As a service provider, you may require to track the number of VMs running your software or have agents that need to track uniqueness of the VM. To be able to get a unique ID for a VM, use the `vmId` field from Instance Metadata Service.

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance/compute/vmId?api-version=2017-08-01&format=text"
```

Response

```
5c08b38e-4d57-4c23-ac45-aca61037f084
```

Placement of containers, data-partitions based fault/update domain

For certain scenarios, placement of different data replicas is of prime importance. For example, [HDFS replica placement](#) or container placement via an [orchestrator](#) may you require to know the `platformFaultDomain` and `platformUpdateDomain` the VM is running on. You can also use [Availability Zones](#) for the instances to make these decisions. You can query this data directly via the Instance Metadata Service.

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance/compute/platformFaultDomain?api-version=2017-08-01&format=text"
```

Response

0

Getting more information about the VM during support case

As a service provider, you may get a support call where you would like to know more information about the VM. Asking the customer to share the compute metadata can provide basic information for the support professional to know about the kind of VM on Azure.

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance/compute?api-version=2017-08-01"
```

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{
  "compute": {
    "location": "CentralUS",
    "name": "IMDSCanary",
    "offer": "RHEL",
    "osType": "Linux",
    "platformFaultDomain": "0",
    "platformUpdateDomain": "0",
    "publisher": "RedHat",
    "sku": "7.2",
    "version": "7.2.20161026",
    "vmId": "5c08b38e-4d57-4c23-ac45-aca61037f084",
    "vmSize": "Standard_DS2"
  }
}
```

Getting Azure Environment where the VM is running

Azure has various sovereign clouds like [Azure Government](#). Sometimes you need the Azure Environment to make some runtime decisions. The following sample shows you how you can achieve this behavior.

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance/compute/azEnvironment?api-version=2018-10-01&format=text"
```

Response

```
AzurePublicCloud
```

The cloud and the values of the Azure Environment are listed below.

CLOUD	AZURE ENVIRONMENT
All Generally Available Global Azure Regions	AzurePublicCloud

CLOUD	AZURE ENVIRONMENT
Azure Government	AzureUSGovernmentCloud
Azure China	AzureChinaCloud
Azure Germany	AzureGermanCloud

Getting the tags for the VM

Tags may have been applied to your Azure VM to logically organize them into a taxonomy. The tags assigned to a VM can be retrieved by using the request below.

Request

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance/compute/tags?api-version=2018-10-01&format=text"
```

Response

```
Department:IT;Environment:Test;Role:WebRole
```

NOTE

The tags are semicolon separated. If a parser is written to programmatically extract the tags, the tag names and values shouldn't contain semicolons in order for the parser to work correctly.

Validating that the VM is running in Azure

Marketplace vendors want to ensure that their software is licensed to run only in Azure. If someone copies the VHD out to on-premise, then they should have the ability to detect that. By calling into Instance Metadata Service, Marketplace vendors can get signed data that guarantees response only from Azure.

NOTE

Requires jq to be installed.

Request

```
# Get the signature
curl --silent -H Metadata:True http://169.254.169.254/metadata/attested/document?api-version=2018-10-01 |
jq -r '.["signature"]' > signature
# Decode the signature
base64 -d signature > decodedsignature
#Get PKCS7 format
openssl pkcs7 -in decodedsignature -inform DER -out sign.pk7
# Get Public key out of pkc7
openssl pkcs7 -in decodedsignature -inform DER -print_certs -out signer.pem
#Get the intermediate certificate
wget -q -O intermediate.cer "$(openssl x509 -in signer.pem -text -noout | grep " CA Issuers -" | awk -FURI:
'{print $2}')"
openssl x509 -inform der -in intermediate.cer -out intermediate.pem
#Verify the contents
openssl smime -verify -in sign.pk7 -inform pem -noverify
```

Response

```

Verification successful
{
  "nonce": "20181128-001617",
  "plan":
  {
    "name": "",
    "product": "",
    "publisher": ""
  },
  "timeStamp":
  {
    "createdOn": "11/28/18 00:16:17 -0000",
    "expiresOn": "11/28/18 06:16:17 -0000"
  },
  "vmId": "d3e0e374-fda6-4649-bbc9-7f20dc379f34"
}

```

DATA	DESCRIPTION
nonce	User supplied optional string with the request. If no nonce was supplied in the request, the current UTC timestamp is returned
plan	Plan for a VM in it's an Azure Marketplace Image, contains name, product, and publisher
timestamp/createdOn	The timestamp at which the first signed document was created
timestamp/expiresOn	The timestamp at which the signed document expires
vmId	Unique identifier for the VM

Verifying the signature

Once you get the signature above, you can verify that the signature is from Microsoft. Also you can verify the intermediate certificate and the certificate chain.

NOTE

The certificate for Public cloud and sovereign cloud will be different.

CLOUD	CERTIFICATE
All Generally Available Global Azure Regions	metadata.azure.com
Azure Government	metadata.azure.us
Azure China	metadata.azure.cn
Azure Germany	metadata.microsoftazure.de

```

# Verify the subject name for the main certificate
openssl x509 -noout -subject -in signer.pem
# Verify the issuer for the main certificate
openssl x509 -noout -issuer -in signer.pem
#Validate the subject name for intermediate certificate
openssl x509 -noout -subject -in intermediate.pem
# Verify the issuer for the intermediate certificate
openssl x509 -noout -issuer -in intermediate.pem
# Verify the certificate chain
openssl verify -verbose -CAfile /etc/ssl/certs/Baltimore_CyberTrust_Root.pem -untrusted intermediate.pem
signer.pem

```

In cases where the intermediate certificate cannot be downloaded due to network constraints during validation, the intermediate certificate can be pinned. However, Azure will roll over the certificates as per standard PKI practice. The pinned certificates would need to be updated when roll over happens. Whenever a change to update the intermediate certificate is planned, the Azure blog will be updated and Azure customers will be notified. The intermediate certificates can be found [here](#). The intermediate certificates for each of the regions can be different.

Failover Clustering in Windows Server

For certain scenarios, when querying Instance Metadata Service with Failover Clustering, it is necessary to add a route to the routing table.

1. Open command prompt with administrator privileges.
2. Run the following command and note the address of the Interface for Network Destination (`0.0.0.0`) in the IPv4 Route Table.

```
route print
```

NOTE

The following example output from a Windows Server VM with Failover Cluster enabled contains only the IPv4 Route Table for simplicity.

IPv4 Route Table

<hr/> <hr/> Active Routes:					
Network Destination	Netmask	Gateway	Interface	Metric	
0.0.0.0	0.0.0.0	10.0.1.1	10.0.1.10	266	
10.0.1.0	255.255.255.192	On-link	10.0.1.10	266	
10.0.1.10	255.255.255.255	On-link	10.0.1.10	266	
10.0.1.15	255.255.255.255	On-link	10.0.1.10	266	
10.0.1.63	255.255.255.255	On-link	10.0.1.10	266	
127.0.0.0	255.0.0.0	On-link	127.0.0.1	331	
127.0.0.1	255.255.255.255	On-link	127.0.0.1	331	
127.255.255.255	255.255.255.255	On-link	127.0.0.1	331	
169.254.0.0	255.255.0.0	On-link	169.254.1.156	271	
169.254.1.156	255.255.255.255	On-link	169.254.1.156	271	
169.254.255.255	255.255.255.255	On-link	169.254.1.156	271	
224.0.0.0	240.0.0.0	On-link	127.0.0.1	331	
224.0.0.0	240.0.0.0	On-link	169.254.1.156	271	
224.0.0.0	240.0.0.0	On-link	10.0.1.10	266	
255.255.255.255	255.255.255.255	On-link	127.0.0.1	331	
255.255.255.255	255.255.255.255	On-link	169.254.1.156	271	
255.255.255.255	255.255.255.255	On-link	10.0.1.10	266	

1. Run the following command and use the address of the Interface for Network Destination (`0.0.0.0`) which is (

10.0.1.10) in this example.

```
route add 169.254.169.254/32 10.0.1.10 metric 1 -p
```

Custom Data

Instance Metadata Service provides the ability for the VM to have access to its custom data. The binary data must be less than 64 KB and is provided to the VM in base64 encoded form.

Azure custom data can be inserted to the VM through REST APIs, PowerShell Cmdlets, Azure Command Line Interface (CLI), or an ARM template.

For an Azure Command Line Interface example, see [Custom Data and Cloud-Init on Microsoft Azure](#).

For an ARM template example, see [Deploy a Virtual Machine with CustomData](#).

Custom data is available to all processes running in the VM. It is suggested that customers do not insert secret information into custom data.

Currently, custom data is guaranteed to be available during bootstrap of a VM. If updates are made to the VM such as adding disks or resizing the VM, Instance Metadata Service will not provide custom data. Providing custom data persistently through Instance Metadata Service is currently in progress.

Retrieving custom data in Virtual Machine

Instance Metadata Service provides custom data to the VM in base64 encoded form. The following example decodes the base64 encoded string.

NOTE

The custom data in this example is interpreted as an ASCII string that reads, "My custom data.".

Request

```
curl -H "Metadata:true" "http://169.254.169.254/metadata/instance/compute/customData?api-version=2019-02-01&format=text" | base64 --decode
```

Response

```
My custom data.
```

Examples of calling metadata service using different languages inside the VM

LANGUAGE	EXAMPLE
Ruby	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.rb
Go	https://github.com/Microsoft/azureimds/blob/master/imdssample.go
Python	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.py
C++	https://github.com/Microsoft/azureimds/blob/master/IMDSSample-windows.cpp

LANGUAGE	EXAMPLE
C#	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.cs
JavaScript	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.js
PowerShell	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.ps1
Bash	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.sh
Perl	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.pl
Java	https://github.com/Microsoft/azureimds/blob/master/imdssample.java
Visual Basic	https://github.com/Microsoft/azureimds/blob/master/IMDSSample.vb
Puppet	https://github.com/keirans/azuremetadata

FAQ

- I am getting the error `400 Bad Request, Required metadata header not specified`. What does this mean?
 - The Instance Metadata Service requires the header `Metadata: true` to be passed in the request. Passing this header in the REST call allows access to the Instance Metadata Service.
- Why am I not getting compute information for my VM?
 - Currently the Instance Metadata Service only supports instances created with Azure Resource Manager. In the future, support for Cloud Service VMs might be added.
- I created my Virtual Machine through Azure Resource Manager a while back. Why am I not see compute metadata information?
 - For any VMs created after Sep 2016, add a [Tag](#) to start seeing compute metadata. For older VMs (created before Sep 2016), add/remove extensions or data disks to the VM to refresh metadata.
- I am not seeing all data populated for new version
 - For any VMs created after Sep 2016, add a [Tag](#) to start seeing compute metadata. For older VMs (created before Sep 2016), add/remove extensions or data disks to the VM to refresh metadata.
- Why am I getting the error `500 Internal Server Error`?
 - Retry your request based on exponential back off system. If the issue persists contact Azure support.
- Where do I share additional questions/comments?
 - Send your comments on <https://feedback.azure.com>.
- Would this work for Virtual Machine Scale Set Instance?
 - Yes Metadata service is available for Scale Set Instances.
- How do I get support for the service?

- To get support for the service, create a support issue in Azure portal for the VM where you are not able to get metadata response after long retries.

9. I get request timed out for my call to the service?

- Metadata calls must be made from the primary IP address assigned to the network card of the VM, in addition in case you have changed your routes there must be a route for 169.254.0.0/16 address out of your network card.

10. I updated my tags in virtual machine scale set but they don't appear in the instances unlike VMs?

- Currently for ScaleSets tags only show to the VM on a reboot/reimage/or a disk change to the instance.

Problem

NEW SUPPORT REQUEST



* Severity ⓘ

C - Minimal impact



* Problem type

Management



* Category

✓ Choose a category

Backup

Cannot stop, start, or restart a VM

Capacity issues that are related to SAP HANA large instances

Instance Metadata Service

Manage an Exchange Server

Manage an instance of SQL Server

Manage encrypted disks, keys or secrets, or permissions

Manage or use RDS in Azure

Manage or use a VPN

Manage or use a cluster in Azure

Manage or use a virtual network

Manage or use endpoints

Unable to delete a virtual machine

Virtual machine restarts

When did the problem start?

Choose a date



Enter a local time

File upload ⓘ

Select a file



Share diagnostic information ⓘ

[Learn more about the information we collect](#)

Next

Next steps

- Learn more about [Scheduled Events](#)

Get Virtual Machine usage metrics using the REST API

2/5/2019 • 2 minutes to read • [Edit Online](#)

This example shows how to retrieve the CPU usage for a [Linux Virtual Machine](#) using the [Azure REST API](#).

Complete reference documentation and additional samples for the REST API are available in the [Azure Monitor REST reference](#).

Build the request

Use the following GET request to collect the [Percentage CPU](#) metric from a Virtual Machine

```
GET  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmname}/providers/microsoft.insights/metrics?api-version=2018-01-01&metricnames=Percentage%20CPU&timespan=2018-06-05T03:00:00Z/2018-06-07T03:00:00Z
```

Request headers

The following headers are required:

REQUEST HEADER	DESCRIPTION
<i>Content-Type</i> :	Required. Set to <code>application/json</code> .
<i>Authorization</i> :	Required. Set to a valid <code>Bearer</code> access token .

URI parameters

NAME	DESCRIPTION
<code>subscriptionId</code>	The subscription ID that identifies an Azure subscription. If you have multiple subscriptions, see Working with multiple subscriptions .
<code>resourceGroupName</code>	The name of the Azure resource group associated with the resource. You can get this value from the Azure Resource Manager API, CLI, or the portal.
<code>vmname</code>	The name of the Azure Virtual Machine.
<code>metricnames</code>	Comma-separated list of valid Load Balancer metrics .
<code>api-version</code>	The API version to use for the request. This document covers api-version <code>2018-01-01</code> , included in the above URL.

NAME	DESCRIPTION
timespan	String with the following format <code>startDateTime_ISO/endDateTime_ISO</code> that defines the time range of the returned metrics. This optional parameter is set to return a day's worth of data in the example.

Request body

No request body is needed for this operation.

Handle the response

Status code 200 is returned when the list of metric values is returned successfully. A full list of error codes is available in the [reference documentation](#).

Example response

```
{
  "cost": 0,
  "timespan": "2018-06-08T23:48:10Z/2018-06-09T00:48:10Z",
  "interval": "PT1M",
  "value": [
    {
      "id": "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmname}/providers/microsoft.insights/metrics?api-version=2018-01-01&metricnames=Percentage%20CPU",
      "type": "Microsoft.Insights/metrics",
      "name": {
        "value": "Percentage CPU",
        "localizedValue": "Percentage CPU"
      },
      "unit": "Percent",
      "timeseries": [
        {
          "metadatavalues": [],
          "data": [
            {
              "timeStamp": "2018-06-08T23:48:00Z",
              "average": 0.44
            },
            {
              "timeStamp": "2018-06-08T23:49:00Z",
              "average": 0.31
            },
            {
              "timeStamp": "2018-06-08T23:50:00Z",
              "average": 0.29
            },
            {
              "timeStamp": "2018-06-08T23:51:00Z",
              "average": 0.29
            },
            {
              "timeStamp": "2018-06-08T23:52:00Z",
              "average": 0.285
            }
          ]
        }
      ]
    }
  ]
}
```


Prepay for Virtual Machines with Azure Reserved VM Instances (RI)

5/2/2018 • 6 minutes to read • [Edit Online](#)

Prepay for virtual machines and save money with Azure Reserved Virtual Machine (VM) Instances. For more information, see [Azure Reserved VM Instances offering](#).

You can buy a Reserved VM Instance in the [Azure portal](#). To buy an instance:

- You must be in an Owner role for at least one Enterprise or Pay-As-You-Go subscription.
- For Enterprise subscriptions, **Add Reserved Instances** must be enabled in the [EA portal](#). Or, if that setting is disabled, you must be an EA Admin on the subscription.
- For the Cloud Solution Provider (CSP) program, only the admin agents or sales agents can buy reservations.

The reservation discount is applied automatically to the number of running virtual machines that match the reservation scope and attributes. You can update the scope of the reservation through [Azure portal](#), PowerShell, CLI, or through the API.

Determine the right VM size before you buy

Before you buy a reservation, you should determine the size of the VM that you need. The following sections will help you determine the right VM size.

Use reservation recommendations

You can use reservation recommendations to help determine the reservations you should purchase.

- Purchase recommendations and recommended quantity are shown when you purchase a VM reserved instance in the Azure portal.
- Azure Advisor provides purchase recommendations for individual subscriptions.
- You can use the APIs to get purchase recommendations for both shared scope and single subscription scope. For more information, see [Reserved instance purchase recommendation APIs for enterprise customers](#).
- For EA customers, purchase recommendations for shared and single subscription scopes are available with the [Azure Consumption Insights Power BI content pack](#).

Classic VMs and cloud services

Virtual machine reserved instances automatically apply to both classic VMs and cloud services when instance size flexibility is enabled. There aren't any special SKUs for classic VMs or cloud services. The same VM SKUs apply to them.

For example, you might convert your classic VMs or cloud services to Azure Resource Manager-based VMs. In this example, the reservation discount automatically applies to matching VMs. There's no need to *exchange* an existing reserved instance - it automatically applies.

Analyze your usage information

You should analyze your usage information to help determine which reservations you should purchase.

Usage data is available in the usage file and APIs. Use them together to determine which reservation to purchase. You should check for VM instances that have high usage on daily basis to determine the quantity of reservations to purchase.

Avoid the `Meter` subcategory and `Product` fields in usage data. They don't distinguish between VM sizes that use

premium storage. If you use these fields to determine the VM size for reservation purchase, you may buy the wrong size. Then you won't get the reservation discount you expect. Instead, refer to the `AdditionalInfo` field in your usage file or usage API to determine the correct VM size.

Purchase restriction considerations

Reserved VM Instances are available for most VM sizes with some exceptions. Reservation discounts don't apply for the following VMs:

- **VM series** - A-series, Av2-series, or G-series.
- **VMs in preview** - Any VM-series or size that is in preview.
- **Clouds** - Reservations aren't available for purchase in Germany or China regions.
- **Insufficient quota** - A reservation that is scoped to a single subscription must have vCPU quota available in the subscription for the new RI. For example, if the target subscription has a quota limit of 10 vCPUs for D-Series, then you can't buy a reservation for 11 Standard_D1 instances. The quota check for reservations includes the VMs already deployed in the subscription. For example, if the subscription has a quota of 10 vCPUs for D-Series and has two standard_D1 instances deployed, then you can buy a reservation for 10 standard_D1 instances in this subscription. You can [create quote increase request](#) to resolve this issue.
- **Capacity restrictions** - In rare circumstances, Azure limits the purchase of new reservations for subset of VM sizes, due to low capacity in a region.

Buy a Reserved VM Instance

1. Sign in to the [Azure portal](#).
2. Select **All services > Reservations**.
3. Select **Add** to purchase a new reservation.
4. Fill in the required fields. Running VM instances that match the attributes you select qualify to get the reservation discount. The actual number of your VM instances that get the discount depend on the scope and quantity selected.

FIELD	DESCRIPTION
Name	The name of this reservation.
Subscription	The subscription used to pay for the reservation. The payment method on the subscription is charged the upfront costs for the reservation. The subscription type must be an enterprise agreement (offer numbers: MS-AZR-0017P or MS-AZR-0148P) or Pay-As-You-Go (offer numbers: MS-AZR-0003P or MS-AZR-0023P). For an enterprise subscription, the charges are deducted from the enrollment's monetary commitment balance or charged as overage. For Pay-As-You-Go subscription, the charges are billed to the credit card or invoice payment method on the subscription.

FIELD	DESCRIPTION
Scope	The reservation's scope can cover one subscription or multiple subscriptions (shared scope). If you select: <ul style="list-style-type: none"> Single subscription - The reservation discount is applied to VMs in this subscription. Shared - The reservation discount is applied to VMs running in any subscriptions within your billing context. For enterprise customers, the shared scope is the enrollment and includes all subscriptions within the enrollment. For Pay-As-You-Go customers, the shared scope is all Pay-As-You-Go subscriptions created by the account administrator.
Region	The Azure region that's covered by the reservation.
VM Size	The size of the VM instances.
Optimize for	VM instance size flexibility applies the reservation discount to other VMs in the same VM size group . Capacity priority prioritizes data center capacity for your deployments. This offers additional confidence in your ability to launch the VM instances when you need them. Capacity priority is only available when the reservation scope is single subscription.
Term	One year or three years.
Quantity	The number of instances being purchased within the reservation. The quantity is the number of running VM instances that can get the billing discount. For example, if you are running 10 Standard_D2 VMs in the East US, then you would specify quantity as 10 to maximize the benefit for all running machines.

Change a reservation after purchase

You can make the following types of changes to a reservation after purchase:

- Update reservation scope
- Instance size flexibility (if applicable)
- Ownership

You can also split a reservation into smaller chunks and merge already split reservations. None of the changes cause a new commercial transaction or change the end date of the reservation.

You cannot make the following types of changes after purchase, directly:

- An existing reservation's region
- SKU
- Quantity
- Duration

However, you can *exchange* a reservation if you want to make changes.

Cancellations and exchanges

If you need to cancel your reservation, there may be a 12% early termination fee. Refunds are based on the lowest price of either your purchase price or the current price of the reservation. Refunds are limited to \$50,000 per year. The refund you receive is the remaining pro-rated balance minus the 12% early termination fee. To request a cancellation, go to the reservation in the Azure portal and select **Refund** to create a support request.

If you need to change your Reserved VM Instances reservation to another region, VM size group, or term, you can exchange it for another reservation that's of equal or greater value. The term start date for the new reservation doesn't carry over from the exchanged reservation. The 1 or 3-year term starts from when you create the new reservation. To request an exchange, go to the reservation in the Azure portal, and select **Exchange** to create a support request.

For more information about how to exchange or refund reservations, see [Reservation exchanges and refunds](#).

Need help? Contact us.

If you have questions or need help, [create a support request](#).

Next steps

- To learn how to manage a reservation, see [Manage Azure Reservations](#).
- To learn more about Azure Reservations, see the following articles:
 - [What are Azure Reservations?](#)
 - [Manage Reservations in Azure](#)
 - [Understand how the reservation discount is applied](#)
 - [Understand reservation usage for your Pay-As-You-Go subscription](#)
 - [Understand reservation usage for your Enterprise enrollment](#)
 - [Windows software costs not included with reservations](#)
 - [Azure Reservations in Partner Center Cloud Solution Provider \(CSP\) program](#)

Prepay for Azure software plans

4/19/2019 • 3 minutes to read • [Edit Online](#)

When you prepay for your SUSE and RedHat software usage in Azure, you can save money over your pay-as-you-go costs. The discounts only apply to SUSE and RedHat meters and not on the virtual machine usage. You can buy reservations for virtual machines separately for additional savings.

You can buy SUSE and RedHat software plans in the Azure portal. To buy a plan:

- You must have the owner role for at least one Enterprise or Pay-As-You-Go subscription.
- For Enterprise subscriptions, the **Add Reserved Instances** option must be enabled in the [EA portal](#). If the setting is disabled, you must be an EA Admin for the subscription.
- For the Cloud Solution Provider (CSP) program, the admin agents or sales agents can buy the software plans.

Buy a software plan

1. Sign-in to the Azure portal and go to [Reservations](#).
2. Click **Add** and then select the software plan that you want to buy. Fill in the required fields. Any SUSE Linux VM or RedHat VM that matches the attributes of what you buy gets the discount. The actual number of deployments that get the discount depend on the scope and quantity selected.
3. Select a subscription. It's used to pay for the plan. The subscription payment method is charged the upfront costs for the reservation. The subscription type must be an Enterprise Agreement (offer numbers: MS-AZR-0017P or MS-AZR-0148P) or Pay-As-You-Go (offer numbers: MS-AZR-0003P or MS-AZR-0023P).
 - For an enterprise subscription, the charges are deducted from the enrollment's monetary commitment balance or charged as overage.
 - For a Pay-As-You-Go subscription, the charges are billed to the subscription's credit card or invoice payment method.
4. Select a scope. The scope can cover one subscription or multiple subscriptions (shared scope).
 - Single subscription - The plan discount is applied to matching usage in the subscription.
 - Shared - The plan discount is applied to matching instances in any subscription in your billing context. For enterprise customers, the billing context is the enrollment and includes all subscriptions in the enrollment. For Pay-As-You-Go customers, the billing context is all Pay-As-You-Go subscriptions created by the account administrator.
5. Select a product to choose the VM size and the image type. The discount applies to the selected VM size only.
6. Select a one-year or three-year term.
7. Choose a quantity, which is the number of prepaid VM instances that can get the billing discount.
8. Add the product to the cart, review and purchase.

The reservation discount is automatically applied to the software meter that you pre-pay for. VM compute charges aren't covered by the plan. You can purchase the VM reservations separately.

Discount applies to different SUSE VM sizes

Like reserved VM instances, SUSE Linux plans offer instance size flexibility. Your discount applies even when you deploy a VM that's a different size from the SUSE plan you bought. For more information, see [Understand how the software plan discount is applied](#).

RedHat plan discount

Plans are available only for Red Hat Enterprise Linux virtual machines. The discount doesn't apply to RedHat Enterprise Linux SAP HANA VMs or RedHat Enterprise Linux SAP Business Apps VMs.

RedHat plan discounts apply only to the VM size that you select at the time of purchase. RHEL plans can't be refunded or exchanged after purchase.

Cancellation and exchanges not allowed

You can't cancel or exchange a SUSE or RedHat plan that you bought. Check your usage to make sure you buy the right plan. For help to identify what to buy, see [Understand how the software plan discount is applied](#).

Need help? Contact us.

If you have questions or need help, [create a support request](#).

Next steps

To learn how to manage a reservation, see [Manage Azure reservations](#).

To learn more, see the following articles:

- [What are Azure Reservations?](#)
- [Manage Reservations in Azure](#)
- [Understand how the SUSE reservation discount is applied](#)
- [Understand reservation usage for your Pay-As-You-Go subscription](#)
- [Understand reservation usage for your Enterprise enrollment](#)

What are Azure Reservations?

4/22/2019 • 5 minutes to read • [Edit Online](#)

Azure Reservations help you save money by pre-paying for one-year or three-years of virtual machines, SQL Database compute capacity, Azure Cosmos DB throughput, or other Azure resources. Pre-paying allows you to get a discount on the resources you use. Reservations can significantly reduce your virtual machine, SQL database compute, Azure Cosmos DB, or other resource costs up to 72% on pay-as-you-go prices. Reservations provide a billing discount and don't affect the runtime state of your resources.

You can buy a reservation in the [Azure portal](#).

Why buy a reservation?

If you have virtual machines, Azure Cosmos DB, or SQL databases that run for long periods of time, buying a reservation gives you the most cost-effective option. For example, when you continuously run four instances of a service without a reservation, you're charged at pay-as-you-go rates. If you buy a reservation for those resources, you immediately get the reservation discount. The resources are no longer charged at the pay-as-you-go rates.

Charges covered by reservation

Service plans:

- Reserved Virtual Machine Instance: A reservation only covers the virtual machine compute costs. It doesn't cover additional software, networking, or storage charges.
- Azure Cosmos DB reserved capacity: A reservation covers throughput provisioned for your resources. It doesn't cover the storage and networking charges.
- SQL Database reserved vCore: Only the compute costs are included with a reservation. The license is billed separately.

For Windows virtual machines and SQL Database, you can cover the licensing costs with [Azure Hybrid Benefit](#).

Who's eligible to purchase a reservation?

To buy a plan, you must have a subscription owner role in an Enterprise (MS-AZR-0017P or MS-AZR-0148P) or Pay-As-You-Go subscription (MS-AZR-003P or MS-AZR-0023P). Cloud solution providers can use the Azure portal or [Partner Center](#) to purchase Azure Reservations.

EA customers can limit purchases to EA admins by disabling the **Add Reserved Instances** option in EA Portal. EA admins must be a subscription owner for at least one EA subscription to purchase a reservation. The option is useful for enterprises that want a centralized team to purchase reservations for different cost centers. After the purchase, centralized teams can add cost center owners to the reservations. Owners can then scope the reservation to their subscriptions. The central team doesn't need to have subscription owner access where the reservation is purchased.

A reservation discount only applies to resources associated with Enterprise, Pay-As-You-Go, or CSP subscription types.

Reservation scope

A reservation scope determines the resources to which the reservation discount applies. A reservation scope can have following values:

Shared scope - The reservation discount is applied to the matching resources in eligible subscriptions within the billing context.

- For Enterprise Agreement customers, the billing context is the enrollment. For Pay-as-you-go customers, the billing scope is all eligible subscriptions created by the account administrator.

Single subscription - The reservation discount is applied to the matching resources in the selected subscription.

You can [update the scope after you purchase a reservation](#).

Discounted subscription and offer types

Reservation discounts apply to the following eligible subscriptions and offer types.

- Enterprise agreement (offer numbers: MS-AZR-0017P or MS-AZR-0148P)
- Pay-As-You-Go (offer numbers: MS-AZR-0003P or MS-AZR-0023P)
- CSP subscriptions

Resources that run in a subscription with other offer types don't receive the reservation discount.

How is a reservation billed?

The reservation is charged to the payment method tied to the subscription. If you have an Enterprise subscription, the reservation cost is deducted from your monetary commitment balance. If your monetary commitment balance doesn't cover the cost of the reservation, you're billed the overage. If you have a Pay-As-You-Go subscription, the credit card you have on your account is billed immediately. If you're billed by invoice, you see the charges on your next invoice.

How reservation discount is applied

The reservation discount applies to the resource usage matching the attributes you select when you buy the reservation. The attributes include the scope where the matching VMs, SQL databases, Azure Cosmos DB, or other resources run. For example, if you want a reservation discount for four Standard D2 virtual machines in the West US region, then select the subscription where the VMs are running.

A reservation discount is "*use-it-or-lose-it*". If you don't have matching resources for any hour, then you lose a reservation quantity for that hour. You can't carry forward unused reserved hours.

When you shut down a resource, the reservation discount automatically applies to another matching resource in the specified scope. If no matching resources are found in the specified scope, then the reserved hours are *lost*.

For example, you might later create a resource and have a matching reservation that is underutilized. In this example, the reservation discount automatically applies to the new matching resource.

If the virtual machines are running in different subscriptions within your enrollment/account, then select the scope as shared. Shared scope allows the reservation discount to be applied across subscriptions. You can change the scope after you buy a reservation. For more information, see [Manage Azure Reservations](#).

A reservation discount only applies to resources associated with Enterprise, Pay-As-You-Go, or CSP subscription types. Resources that run in a subscription with other offer types don't receive the reservation discount.

When the reservation term expires

At the end of the reservation term, the billing discount expires, and the virtual machine, SQL database, Azure Cosmos DB, or other resource is billed at the pay-as-you go price. Azure Reservations don't auto-renew. To continue getting the billing discount, you must buy a new reservation for eligible services and software.

Discount applies to different sizes

When you buy a reservation, the discount can apply to other instances with attributes that are within the same size group. This feature is known as instance size flexibility. The flexibility of the discount coverage depends on the type of reservation and the attributes you pick when you buy the reservation.

Service plans:

- Reserved VM Instances: When you buy the reservation and select **Optimized for instance size flexibility**, the discount coverage depends on the VM size you select. The reservation can apply to the virtual machines (VMs) sizes in the same size series group. For more information, see [Virtual machine size flexibility with Reserved VM Instances](#).
- SQL Database reserved capacity: The discount coverage depends on the performance tier you pick. For more information, see [Understand how an Azure reservation discount is applied](#).
- Azure Cosmos DB reserved capacity: The discount coverage depends on the provisioned throughput. For more information, see [Understand how an Azure Cosmos DB reservation discount is applied](#).

Need help? Contact us.

If you have questions or need help, [create a support request](#).

Next steps

- Learn more about Azure Reservations with the following articles:
 - [Manage Azure Reservations](#)
 - [Understand reservation usage for your Pay-As-You-Go subscription](#)
 - [Understand reservation usage for your Enterprise enrollment](#)
 - [Windows software costs not included with reservations](#)
 - [Azure Reservations in Partner Center Cloud Solution Provider \(CSP\) program](#)
- Learn more about reservations for service plans:
 - [Virtual Machines with Azure Reserved VM Instances](#)
 - [Azure Cosmos DB resources with Azure Cosmos DB reserved capacity](#)
 - [SQL Database compute resources with Azure SQL Database reserved capacity](#) Learn more about reservations for software plans:
 - [Red Hat software plans from Azure Reservations](#)
 - [SUSE software plans from Azure Reservations](#)

Virtual machine size flexibility with Reserved VM Instances

8/3/2018 • 5 minutes to read • [Edit Online](#)

With a reserved virtual machine instance that's optimized for instance size flexibility, the reservation you buy can apply to the virtual machines (VMs) sizes in the same size series group. For example, if you buy a reservation for a VM size that's listed in the DSv2-series table, like Standard_DS5_v2, the reservation discount can apply to the other four sizes that are listed in that same table:

- Standard_DS1_v2
- Standard_DS2_v2
- Standard_DS3_v2
- Standard_DS4_v2

But that reservation discount doesn't apply to VMs sizes that are listed in different tables, like what's in the DSv2-series high memory table: Standard_DS11_v2, Standard_DS12_v2, and so on.

Within the size series group, the number of VMs the reservation discount applies to depends on the VM size you pick when you buy a reservation. It also depends on the sizes of the VMs that you have running. The ratio column that's listed in the following tables compares the relative footprint for each VM size in that group. Use the ratio value to calculate how the reservation discount applies to the VMs you have running.

Examples

The following examples use the sizes and ratios in the DSv2-series table.

You buy a reserved VM instance with the size Standard_DS4_v2 where the ratio or relative footprint compared to the other sizes in that series is 8.

- Scenario 1: Run eight Standard_DS1_v2 sized VMs with a ratio of 1. Your reservation discount applies to all eight of those VMs.
- Scenario 2: Run two Standard_DS2_v2 sized VMs with a ratio of 2 each. Also run a Standard_DS3_v2 sized VM with a ratio of 4. The total footprint is $2+2+4=8$. So your reservation discount applies to all three of those VMs.
- Scenario 3: Run one Standard_DS5_v2 with a ratio of 16. Your reservation discount applies to half that VM's compute cost.

The following sections show what sizes are in the same size series group when you buy a reserved VM instance optimized for instance size flexibility.

B-series

SIZE	RATIO
Standard_B1s	1
Standard_B2s	4

For more information, see [B-series burstable virtual machine sizes](#).

B-series high memory

SIZE	RATIO
Standard_B1ms	1
Standard_B2ms	4
Standard_B4ms	8
Standard_B8ms	16

For more information, see [B-series burstable virtual machine sizes](#).

D-series

SIZE	RATIO
Standard_D1	1
Standard_D2	2
Standard_D3	4
Standard_D4	8

For more information, see [Previous generations of virtual machine sizes](#).

D-series high memory

SIZE	RATIO
Standard_D11	1
Standard_D12	2
Standard_D13	4
Standard_D14	8

For more information, see [Previous generations of virtual machine sizes](#).

Ds-series

SIZE	RATIO
Standard_DS1	1
Standard_DS2	2
Standard_DS3	4

SIZE	RATIO
Standard_DS4	8

For more information, see [Previous generations of virtual machine sizes](#).

Ds-series high memory

SIZE	RATIO
Standard_DS11	1
Standard_DS12	2
Standard_DS13	4
Standard_DS14	8

For more information, see [Previous generations of virtual machine sizes](#).

DSv2-series

SIZE	RATIO
Standard_DS1_v2	1
Standard_DS2_v2	2
Standard_DS3_v2	4
Standard_DS4_v2	8
Standard_DS5_v2	16

For more information, see [Previous generations of virtual machine sizes](#).

DSv2-series high memory

SIZE	RATIO
Standard_DS11_v2	1
Standard_DS11-1_v2	1
Standard_DS12_v2	2
Standard_DS12-1_v2	2
Standard_DS12-2_v2	2
Standard_DS13_v2	4

SIZE	RATIO
Standard_DS13-2_v2	4
Standard_DS13-4_v2	4
Standard_DS14_v2	8
Standard_DS14-4_v2	8
Standard_DS14-8_v2	8
Standard_DS15_v2	10

For more information, see [Previous generations of virtual machine sizes](#).

DSv3-series

SIZE	RATIO
Standard_D2s_v3	1
Standard_D4s_v3	2
Standard_D8s_v3	4
Standard_D16s_v3	8
Standard_D32s_v3	16
Standard_D64s_v3	32

For more information, see [General purpose virtual machine sizes](#).

Dv2-series

SIZE	RATIO
Standard_D1_v2	1
Standard_D2_v2	2
Standard_D3_v2	4
Standard_D4_v2	8
Standard_D5_v2	16

For more information, see [Previous generations of virtual machine sizes](#).

Dv2-series high memory

SIZE	RATIO
Standard_D11_v2	1
Standard_D12_v2	2
Standard_D13_v2	4
Standard_D14_v2	8
Standard_D15_v2	10

For more information, see [Previous generations of virtual machine sizes](#).

Dv3-series

SIZE	RATIO
Standard_D2_v3	1
Standard_D4_v3	2
Standard_D8_v3	4
Standard_D16_v3	8
Standard_D32_v3	16
Standard_D64_v3	32

For more information, see [General purpose virtual machine sizes](#).

ESv3-series

SIZE	RATIO
Standard_E2s_v3	1
Standard_E4s_v3	2
Standard_E4-2s_v3	2
Standard_E8s_v3	4
Standard_E8-2s_v3	4
Standard_E8-4s_v3	4
Standard_E16s_v3	8
Standard_E16-4s_v3	8

SIZE	RATIO
Standard_E16-8s_v3	8
Standard_E20s_v3	10
Standard_E32s_v3	16
Standard_E32-8s_v3	16
Standard_E32-16s_v3	16
Standard_E64s_v3	28.8
Standard_E64-16s_v3	28.8
Standard_E64-32s_v3	28.8

For more information, see [Memory optimized virtual machine sizes](#).

Ev3-series

SIZE	RATIO
Standard_E2_v3	1
Standard_E4_v3	2
Standard_E8_v3	4
Standard_E16_v3	8
Standard_E20_v3	10
Standard_E32_v3	16
Standard_E64_v3	32

For more information, see [Memory optimized virtual machine sizes](#).

F-series

SIZE	RATIO
Standard_F1	1
Standard_F2	2
Standard_F4	4
Standard_F8	8

SIZE	RATIO
Standard_F16	16

For more information, see [Previous generations of virtual machine sizes](#).

FS-series

SIZE	RATIO
Standard_F1s	1
Standard_F2s	2
Standard_F4s	4
Standard_F8s	8
Standard_F16s	16

For more information, see [Previous generations of virtual machine sizes](#).

Fsv2-series

SIZE	RATIO
Standard_F2s_v2	1
Standard_F4s_v2	2
Standard_F8s_v2	4
Standard_F16s_v2	8
Standard_F32s_v2	16
Standard_F64s_v2	32
Standard_F72s_v2	36

For more information, see [Compute optimized virtual machine sizes](#).

H-series

SIZE	RATIO
Standard_H8	1
Standard_H16	2

For more information, see [High performance compute VM sizes](#).

H-series high memory

SIZE	RATIO
Standard_H8m	1
Standard_H16m	2

For more information, see [High performance compute VM sizes](#).

Ls-series

SIZE	RATIO
Standard_L4s	1
Standard_L8s	2
Standard_L16s	4
Standard_L32s	8

For more information, see [Storage optimized virtual machine sizes](#).

M-series

SIZE	RATIO
Standard_M64s	1
Standard_M128s	2

For more information, see [Memory optimized virtual machine sizes](#).

M-series fractional

SIZE	RATIO
Standard_M16s	1
Standard_M32s	2

For more information, see [Memory optimized virtual machine sizes](#).

M-series fractional high memory

SIZE	RATIO
Standard_M8ms	1
Standard_M8-2ms	1

SIZE	RATIO
Standard_M8-4ms	1
Standard_M16ms	2
Standard_M16-4ms	2
Standard_M16-8ms	2
Standard_M32ms	4
Standard_M32-8ms	4
Standard_M32-16ms	4

For more information, see [Memory optimized virtual machine sizes](#).

M-series fractional large

SIZE	RATIO
Standard_M32ls	1
Standard_M64ls	2

For more information, see [Memory optimized virtual machine sizes](#).

M-series high memory

SIZE	RATIO
Standard_M64ms	1
Standard_M64-16ms	1
Standard_M64-32ms	1
Standard_M128ms	2
Standard_M128-32ms	2
Standard_M128-64ms	2

For more information, see [Memory optimized virtual machine sizes](#).

NC-series

SIZE	RATIO
Standard_NC6	1

SIZE	RATIO
Standard_NC12	2
Standard_NC24	4

For more information, see [GPU optimized virtual machine sizes](#).

NCv2-series

SIZE	RATIO
Standard_NC6s_v2	1
Standard_NC12s_v2	2
Standard_NC24s_v2	4

For more information, see [GPU optimized virtual machine sizes](#).

NCv3-series

SIZE	RATIO
Standard_NC6s_v3	1
Standard_NC12s_v3	2
Standard_NC24s_v3	4

For more information, see [GPU optimized virtual machine sizes](#).

ND-series

SIZE	RATIO
Standard_ND6s	1
Standard_ND12s	2
Standard_ND24s	4

For more information, see [GPU optimized virtual machine sizes](#).

NV-series

SIZE	RATIO
Standard_NV6	1
Standard_NV12	2

SIZE	RATIO
Standard_NV24	4

For more information, see [GPU optimized virtual machine sizes](#).

Shared Image Galleries overview

5/6/2019 • 14 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.
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 Windows or Linux, minimum and maximum memory requirements, and release notes. 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.

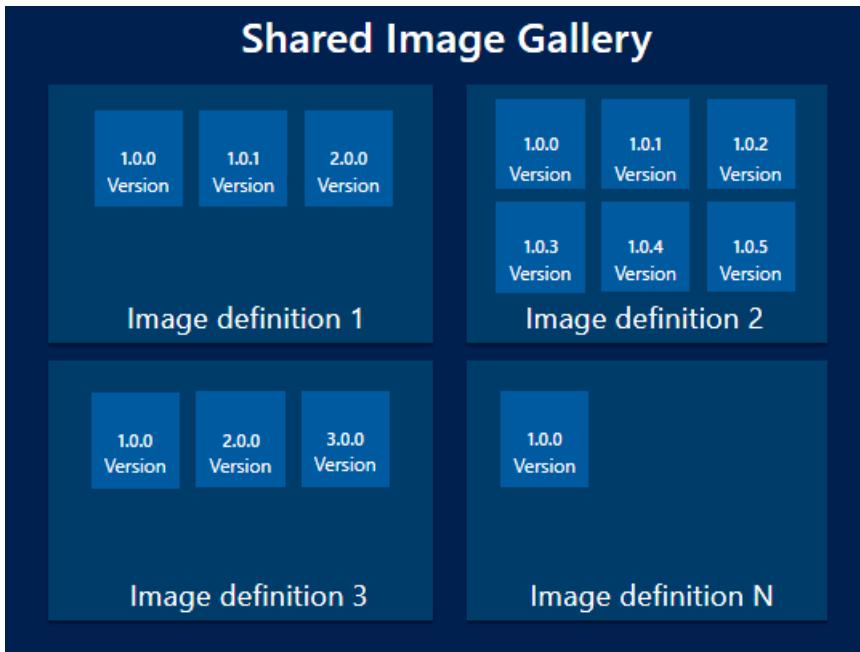


Image definitions

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, but only generalized is currently supported. Images must be created from VMs that have been generalized using Sysprep for Windows or `waagent -deprovision` for Linux.
- 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 in order 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.

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://www.microsoft.com/en-au/central-regions-eligibility/>

SOURCE REGIONS			
Australia Central	Central US EUAP	Korea Central	West Central US
Australia Central 2	East Asia	Korea South	West Europe
Australia East	East US	North Central US	West India
Australia Southeast	East US 2	North Europe	West US
Brazil South	East US 2 EUAP	South Central US	West US 2
Canada Central	France Central	South India	
Canada East	France South	Southeast Asia	
Central India	Japan East	UK South	
Central US	Japan West	UK West	

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

For more information, see [Check resource usage against limits](#) for examples on how to check your current usage.

Scaling

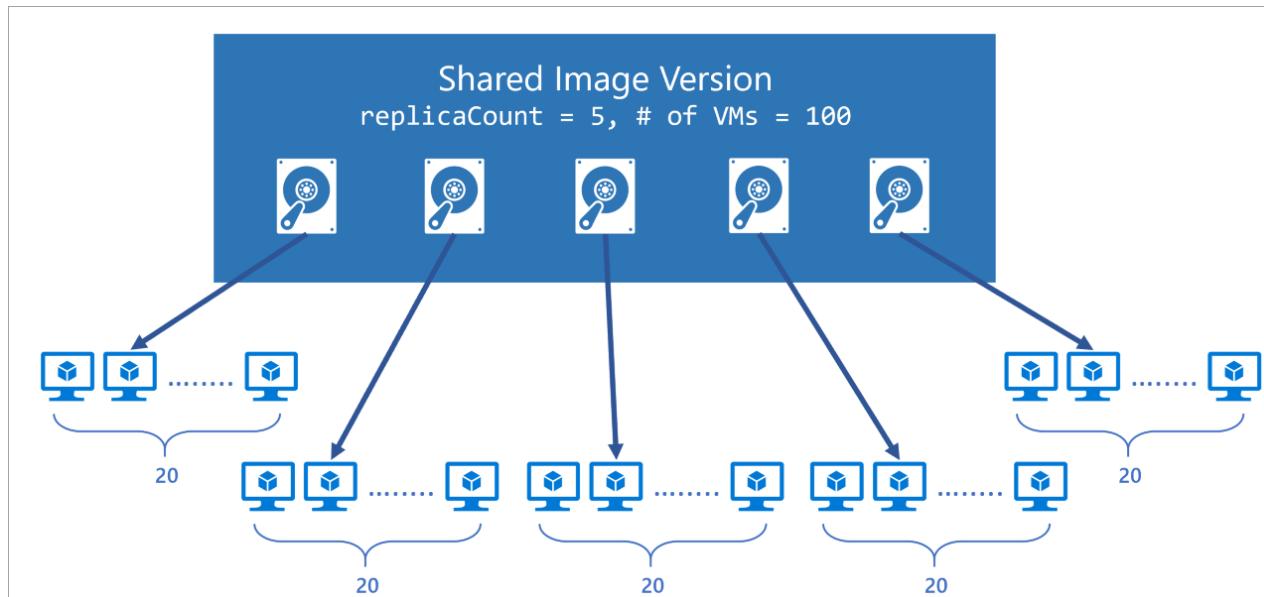
Shared Image Gallery allows you to specify the number of replicas you want Azure to keep of the images. This helps in multi-VM deployment scenarios as the VM deployments can be spread to different replicas reducing the chance of instance creation processing being throttled due to overloading of a single 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 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 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.

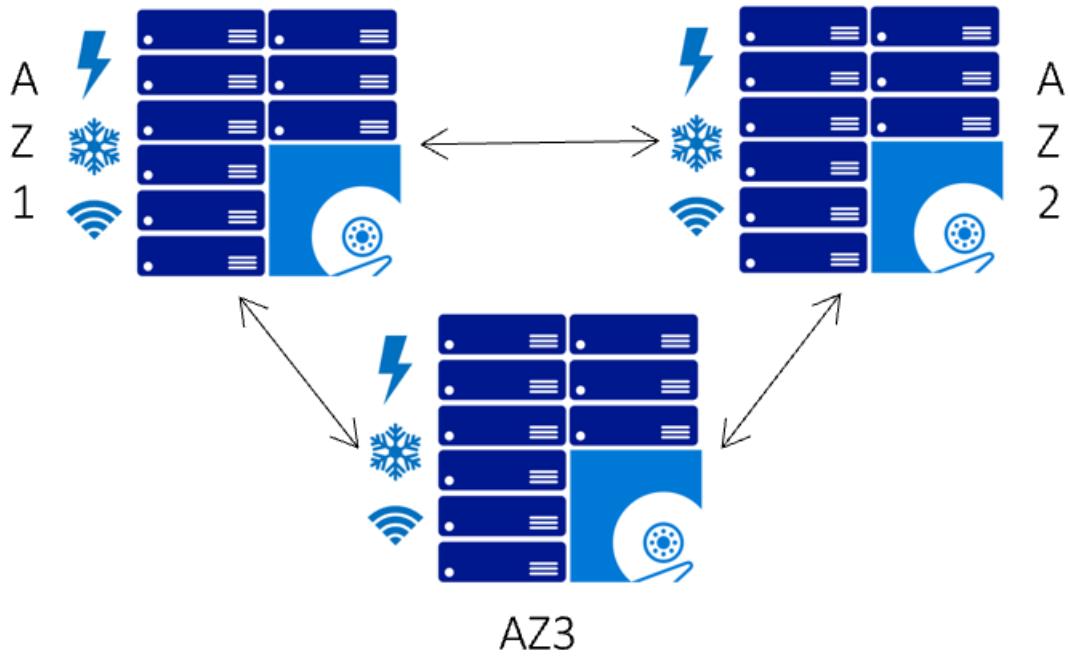
We always recommend you to overprovision the number of replicas due to factors like image size, content and OS type.



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](#).

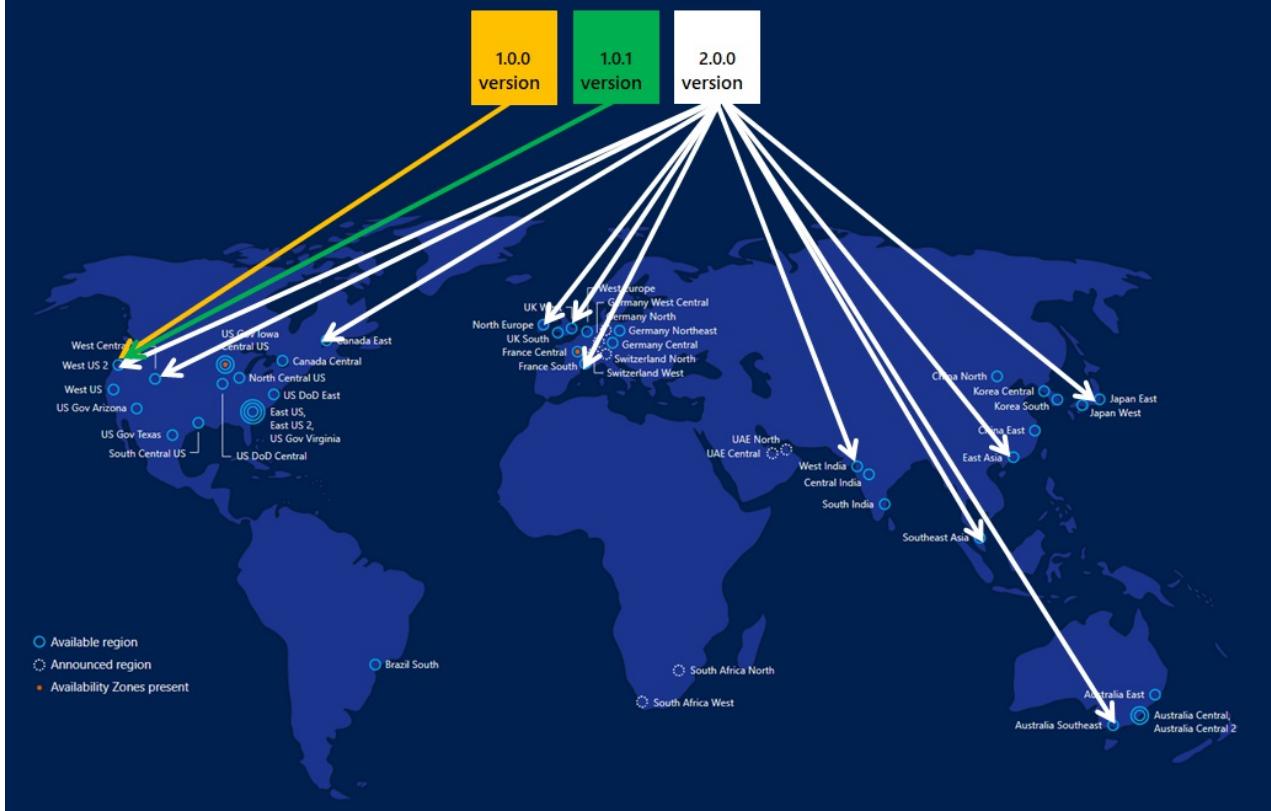


Replication

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.

Image version replication across regions



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

Q. How can I list all the Shared Image Gallery resources across subscriptions?

A. In order 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
```

Q. Can I move my existing image to the shared image gallery?

A. Yes. There are 3 scenarios based on the types of images you may have.

Scenario 1: If you have a managed image, then you can create an image definition and image version from it.

Scenario 2: If you have an unmanaged generalized image, 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, create 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 generalized VHD](#).
- If the VHD is for a Linux VM, see [Upload a VHD](#)

Q. Can I create an image version from a specialized disk?

A. No, we do not currently support specialized disks as images. If you have a specialized disk, you need to [create a VM from the VHD](#) by attaching the specialized disk to a new VM. Once you have a running VM, you need to follow the instructions to create a managed image from the [Windows VM](#) or [Linux VM](#). Once you have a generalized managed image, you can start the process to create a shared image description and image version.

Q. Once created, can I move the Shared Image Gallery resource to a different subscription?

A. 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.

Q. Can I replicate my image versions across clouds – Azure China 21Vianet, Azure Germany and Azure Government Cloud?

A. No, you cannot replicate image versions across clouds.

Q. Can I replicate my image versions across subscriptions?

A. No, you may replicate the image versions across regions in a subscription and use it in other subscriptions through RBAC.

Q. Can I share image versions across Azure AD tenants?

A. 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](#).

Q. How long does it take to replicate image versions across the target regions?

A. 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.

Q. What is the difference between source region and target region?

A. 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.

Q. How do I specify the source region while creating the image version?

A. 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.

Q. How do I specify the number of image version replicas to be created in each region?

A. 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.

Q. Can I create the shared image gallery in a different location than the one where I want to create the image definition and image version?

A. 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.

Q. What are the charges for using the Shared Image Gallery?

A. 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.

Q. What API version should I use to create Shared Image Gallery, Image Definition, Image Version, and VM/VMSS out of the Image Version?

A. For VM and Virtual Machine Scale Set deployments using an image version, we recommend you use API version 2018-04-01 or higher. 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.

Next steps

Learn how to [deploy shared images](#).

Create a shared image gallery with the Azure CLI

5/23/2019 • 7 minutes to read • [Edit Online](#)

A [Shared Image Gallery](#) 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.

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` when you create the image version.

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 VM

Create a VM from the latest image version using [az vm create](#).

```
az vm create\
--resource-group myGalleryRG \
--name myVM \
--image "/subscriptions/subscription ID where the gallery is
located/resourceGroups/myGalleryRG/providers/Microsoft.Compute/galleries/myGallery/images/myImageDefinition" \
--generate-ssh-keys
```

You can also use a specific version by using the image version ID for the `--image` parameter. For example, to use image version *1.0.0* type:

```
--image "/subscriptions/<subscription ID where the gallery is
located>/resourceGroups/myGalleryRG/providers/Microsoft.Compute/galleries/myGallery/images/myImageDefinition/versions/1.0.0"
```

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"
```

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.

Update the description of a gallery using [\(az sig update\)](#).

```
az sig update \
--gallery-name myGallery \
--resource-group myGalleryRG \
--set description="My updated description."
```

Update the description of an image definition using [az sig image-definition update](#).

```
az sig image-definition update \
--gallery-name myGallery\
--resource-group myGalleryRG \
--gallery-image-definition myImageDefinition \
--set description="My updated description."
```

Update an image version to add a region to replicate to using [az sig image-version update](#). This change will take a while as the image gets replicated to the new region.

```
az sig image-version update \
--resource-group myGalleryRG \
--gallery-name myGallery \
--gallery-image-definition myImageDefinition \
--gallery-image-version 1.0.0 \
--add publishingProfile.targetRegions name=eastus
```

Delete resources

You have to delete resources in reverse order, by deleting the image version first. After you delete all of the image versions, you can delete the image definition. After you delete all image definitions, you can delete the gallery.

Delete an image version using [az sig image-version delete](#).

```
az sig image-version delete \
--resource-group myGalleryRG \
--gallery-name myGallery \
--gallery-image-definition myImageDefinition \
--gallery-image-version 1.0.0
```

Delete an image definition using [az sig image-definition delete](#).

```
az sig image-definition delete \
--resource-group myGalleryRG \
--gallery-name myGallery \
--gallery-image-definition myImageDefinition
```

Delete an image gallery using [az sig delete](#).

```
az sig delete \
--resource-group myGalleryRG \
--gallery-name myGallery
```

Next steps

[Azure Image Builder \(preview\)](#) can help automate image version creation, you can even use it to update and [create a new image version from an existing image version](#).

You can also create Shared Image Gallery resources 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.](#)

Create a shared image gallery using the Azure portal

6/28/2019 • 8 minutes to read • [Edit Online](#)

A [Shared Image Gallery](#) 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 deployment tasks like 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.

Before you begin

To complete the example in this article, you must have an existing managed image. You can follow [Tutorial: Create a custom image of an Azure VM 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 this article, replace the resource group and VM names where needed.

Sign in to Azure

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

NOTE

If you registered to use Shared Image Galleries during the preview, you might need to re-register the `Microsoft.Compute` provider. Open [Cloud Shell](#) and type: `az provider register -n Microsoft.Compute`

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.

The following example creates a gallery named *myGallery* in the *myGalleryRG* resource group.

1. Select **Create a resource** in the upper left-hand corner of the Azure portal.
2. Use the type **Shared image gallery** in the search box and select **Shared image gallery** in the results.
3. In the **Shared image gallery** page, click **Create**.
4. Select the correct subscription.
5. In **Resource group**, select **Create new** and type *myGalleryRG* for the name.
6. In **Name**, type *myGallery* for the name of the gallery.
7. Leave the default for **Region**.
8. You can type a short description of the gallery, like *My image gallery for testing.* and then click **Review + create**.
9. After validation passes, select **Create**.
10. When the deployment is finished, select **Go to resource**.

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 gallery image definition inside of your gallery. In this example, the gallery image is named *myImageDefinition*.

1. On the page for your new image gallery, select **Add a new image definition** from the top of the page.
2. For **Image definition name**, type *myImageDefinition*.
3. For **Operating system**, select the correct option based on your source image.
4. For **Publisher**, type *myPublisher*.
5. For **Offer**, type *myOffer*.
6. For **SKU**, type *mySKU*.
7. When finished, select **Review + create**.
8. After the image definition passes validation, select **Create**.
9. When the deployment is finished, select **Go to resource**.

Create an image version

Create an image version from a managed image. In this example, the image version is *7.0.0* 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.

Allowed characters for image version are numbers and periods. Numbers must be within the range of a 32-bit integer. Format: *MajorVersion.MinorVersion.Patch*.

1. In the page for your image definition, select **Add version** from the top of the page.
2. In **Region**, select the region where your managed image is stored. Image versions need to be created in the same region as the managed image they are created from.
3. For **Name**, type *1.0.0*. The image version name should follow *major.minor.patch* format using integers.
4. In **Source image**, select your source managed image from the drop-down.
5. In **Exclude from latest**, leave the default value of *No*.
6. For **End of life date**, select a date from the calendar that is a couple of months in the future.
7. In **Replication**, leave the **Default replica count** as 1. You need to replicate to the source region, so leave the first replica as the default and then pick a second replica region to be *East US*.
8. When you are done, select **Review + create**. Azure will validate the configuration.
9. When image version passes validation, select **Create**.
10. When the deployment is finished, select **Go to resource**.

It can take a while to replicate the image to all of the target regions.

Share the gallery

We recommend that you share access at the image gallery level. The following walks you through sharing the gallery that you just created.

1. Open the [Azure portal](#).
2. In the menu at the left, select **Resource groups**.
3. In the list of resource groups, select **myGalleryRG**. The blade for your resource group will open.
4. In the menu on the left of the **myGalleryRG** page, select **Access control (IAM)**.
5. Under **Add a role assignment**, select **Add**. The **Add a role assignment** pane will open.
6. Under **Role**, select **Reader**.
7. Under **assign access to**, leave the default of **Azure AD user, group, or service principal**.
8. Under **Select**, type in the email address of the person that you would like to invite.
9. If the user is outside of your organization, you will see the message **This user will be sent an email that enables them to collaborate with Microsoft**. Select the user with the email address and then click **Save**.

If the user is outside of your organization, they will get an email invitation to join the organization. The user needs to accept the invitation, then they will be able to see the gallery and all of the image definitions and versions in their list of resources.

Create VMs from an image

Once the image version is complete, you can create one or more new VMs.

IMPORTANT

You cannot use the portal to deploy a VM from an image in another azure tenant. To create a VM from an image shared between tenants, you must use the [Azure CLI](#) or [Powershell](#).

This example creates a VM named *myVMfromImage*, in the *myResourceGroup* in the *East US* datacenter.

1. On the page for your image version, select **Create VM** from the menu at the top of the page.
2. For **Resource group**, select **Create new** and type *myResourceGroup* for the name.
3. In **Virtual machine name**, type *myVM*.

4. For **Region**, select *East US*.
5. For **Availability options**, leave the default of *No infrastructure redundancy required*.
6. The value for **Image** should be automatically filled in if you started from the page for the image version.
7. For **Size**, choose a VM size from the list of available sizes and then click "Select".
8. Under **Administrator account**, select **Password** or **SSH public key**, then enter your information.
9. If you want to allow remote access to the VM, under **Public inbound ports**, choose **Allow selected ports** and then select **SSH (22)** from the drop-down. If you don't want to allow remote access to the VM, leave **None** selected for **Public inbound ports**.
10. When you are finished, select the **Review + create** button at the bottom of the page.
11. After the VM passes validation, select **Create** at the bottom of the page to start the deployment.

Clean up resources

When no longer needed, you can delete the resource group, virtual machine, and all related resources. To do so, select the resource group for the virtual machine, select **Delete**, then confirm the name of the resource group to delete.

If you want to delete individual resources, you need to delete them in reverse order. For example, to delete an image definition, you need to delete all of the image versions created from that image.

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

6/28/2019 • 3 minutes to read • [Edit Online](#)

Shared Image Galleries let you share images using RBAC. You can use RBAC to share images within your tenant, and even to individuals outside of your tenant. 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 **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 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.

IMPORTANT

You cannot use the portal to deploy a VM from an image in another azure tenant. To create a VM from an image shared between tenants, you must use the Azure CLI or [Powershell](#).

Create a VM 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 VM. Replace the information in the example with your own.

```
az vm create \  
  --resource-group myResourceGroup \  
  --name myVM \  
  --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](#).

Troubleshooting shared image galleries

5/6/2019 • 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: Azure Image Builder overview

5/6/2019 • 4 minutes to read • [Edit Online](#)

Standardized virtual machine (VM) images allow organizations to migrate to the cloud and ensure consistency in the deployments. Images typically include predefined security and configuration settings and necessary software. Setting up your own imaging pipeline requires time, infrastructure and setup, but with Azure VM Image Builder, just provide a simple configuration describing your image, submit it to the service, and the image is built, and distributed.

The Azure VM Image Builder (Azure Image Builder) lets you start with a Windows or Linux-based Azure Marketplace image, existing custom images or Red Hat Enterprise Linux (RHEL) ISO and begin to add your own customizations. Because the Image Builder is built on [HashiCorp Packer](#), you can also import your existing Packer shell provisioner scripts. You can also specify where you would like your images hosted, in the Azure Shared Image Gallery (virtual-machines-common-shared-image-galleries.md), as a managed image or a VHD.

IMPORTANT

Azure Image Builder 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](#).

Preview features

For the preview, these features are supported:

- Creation of golden baseline images, that includes your minimum security and corporate configurations, and allow departments to customize it further for their needs.
- Patching of existing images, Image Builder will allow you to continually patch existing custom images.
- Integration with the Azure Shared Image Gallery, allows you to distribute, version, and scale images globally, and gives you an image management system.
- Integration with existing image build pipelines, just call Image Builder from your pipeline, or use the simple Preview Image Builder Azure DevOps Task.
- Migrate an existing image customization pipeline to Azure. Use your existing scripts, commands, and processes to customize images.
- Use Red Hat Bring Your Own Subscription support. Create Red Hat Enterprise images for use with your eligible, unused Red Hat subscriptions.
- Creation of images in VHD format.

Regions

The Azure Image Builder Service will be available for preview in these regions. Images can be distributed outside of these regions.

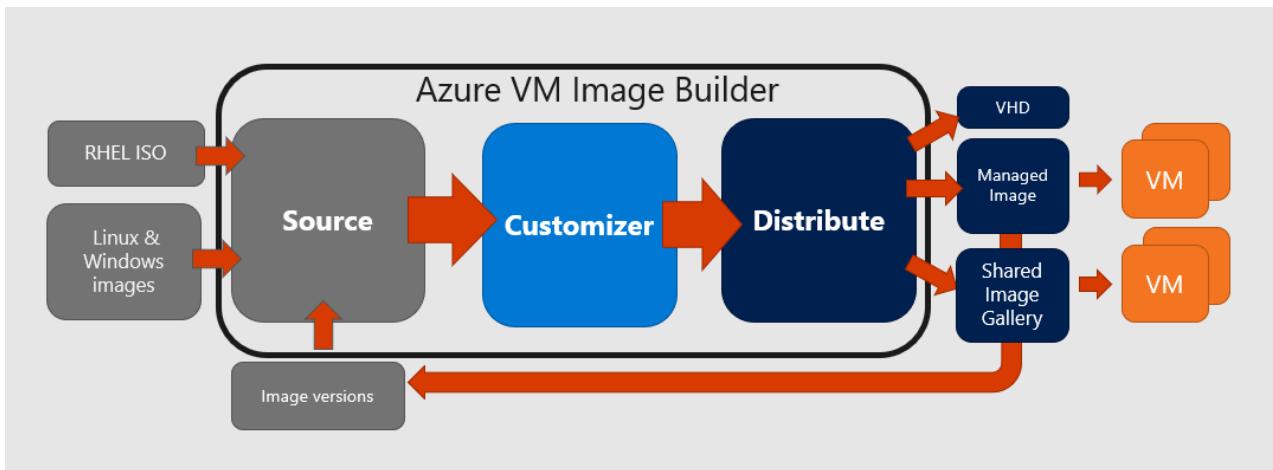
- East US
- East US 2
- West Central US
- West US
- West US 2

OS support

AIB will support Azure Marketplace base OS images:

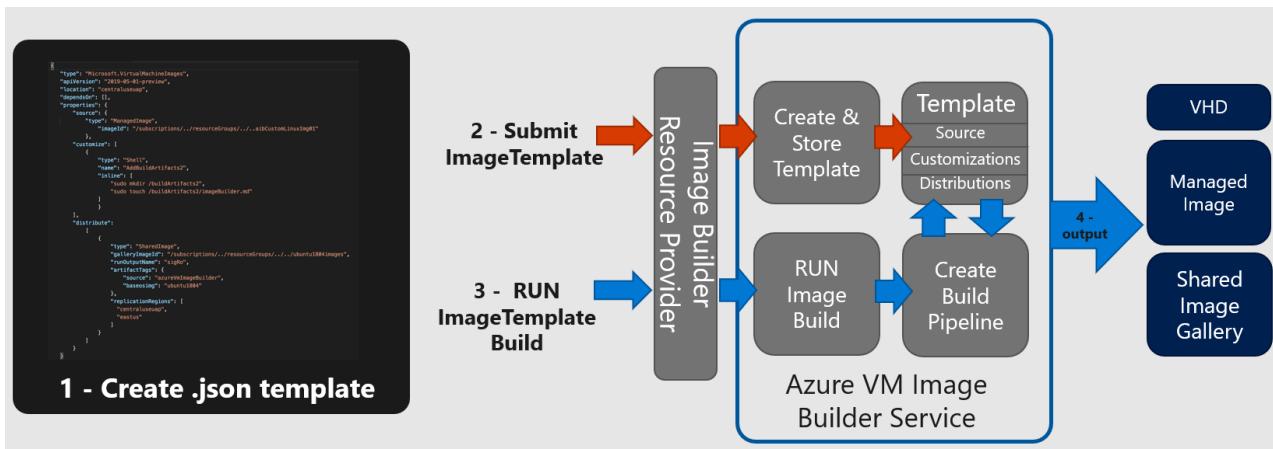
- Ubuntu 18.04
- Ubuntu 16.04
- RHEL 7.6
- CentOS 7.6
- Windows 2016
- Windows 2019

How it works



The Azure Image Builder is a fully managed Azure service that is accessible by an Azure resource provider. The Azure Image Builder process has three main parts: source, customize and distribute, these are represented in a template. The diagram below shows the components, with some of their properties.

Image Builder process



1. Create the Image Template as a json file. This json file contains information about the image source, customizations, and distribution. There are multiple examples in the [Azure Image Builder GitHub repository](#).
2. Submit it to the service, this will create an Image Template artifact in the resource group you specify. In the background, Image Builder will download the source image or ISO, and scripts as needed. These are stored in a separate resource group that is automatically created in your subscription, in the format:
IT_<DestinationResourceGroup>_<TemplateName>.
3. Once the Image Template is created, you can then build the image. In the background Image Builder uses the template and source files to create a VM, network, and storage in the IT_<DestinationResourceGroup>_<TemplateName> resource group.

4. As part of the image creation, Image builder distributes the image it according to the template, then deletes the additional resources in the `IT_<DestinationResourceGroup>_<TemplateName>` resource group that was created for the process.

Permissions

To allow Azure VM Image Builder to distribute images to either the managed images or to a Shared Image Gallery, you will need to provide 'Contributor' permissions for the service "Azure Virtual Machine Image Builder" (app ID: `cf32a0cc-373c-47c9-9156-0db11f6a6dfc`) on the resource groups.

If you are using an existing custom managed image or image version, then the Azure Image Builder will need a minimum of 'Reader' access to those resource groups.

You can assign access using the Azure CLI:

```
az role assignment create \
--assignee cf32a0cc-373c-47c9-9156-0db11f6a6dfc \
--role Contributor \
--scope /subscriptions/$subscriptionID/resourceGroups/<resourceGroupName>
```

If the service account is not found, that may mean that the subscription where you are adding the role assignment has not yet registered for the resource provider.

Costs

You will incur some compute, networking and storage costs when creating, building and storing images with Azure Image Builder. These costs are similar to the costs incurred in manually creating custom images. For the resources, you will be charged at your Azure rates.

During the image creation process, files are downloaded and stored in the `IT_<DestinationResourceGroup>_<TemplateName>` resource group, which will incur a small storage costs. If you do not want to keep these, delete the Image Template after the image build.

Image Builder creates a VM using a D1v2 VM size, and the storage, and networking needed for the VM. These resources will last for the duration of the build process, and will be deleted once Image Builder has finished creating the image.

Azure Image Builder will distribute the image to your chosen regions, which might incur network egress charges.

Next steps

To try out the Azure Image Builder, see the articles for building [Linux](#) or [Windows](#) images.

Preview: Create a Linux VM with Azure Image Builder

6/28/2019 • 4 minutes to read • [Edit Online](#)

This article shows you how you can create a customized Linux image using the Azure Image Builder and the Azure CLI. The example in this article uses three different [customizers](#) for customizing the image:

- Shell (ScriptUri) - downloads and runs a [shell script](#).
- Shell (inline) - runs specific commands. In this example, the inline commands include creating a directory and updating the OS.
- File - copies a [file from GitHub](#) into a directory on the VM.

We will be using a sample json template to configure the image. The json file we are using is here: [helloImageTemplateLinux.json](#).

IMPORTANT

Azure Image Builder 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](#).

Register the features

To use Azure Image Builder during the preview, you need to register the new feature.

```
az feature register --namespace Microsoft.VirtualMachineImages --name VirtualMachineTemplatePreview
```

Check the status of the feature registration.

```
az feature show --namespace Microsoft.VirtualMachineImages --name VirtualMachineTemplatePreview | grep state
```

Check your registration.

```
az provider show -n Microsoft.VirtualMachineImages | grep registrationState  
az provider show -n Microsoft.Storage | grep registrationState
```

If they do not say registered, run the following:

```
az provider register -n Microsoft.VirtualMachineImages  
az provider register -n Microsoft.Storage
```

Setup example variables

We will be using some pieces of information repeatedly, so we will create some variables to store that information.

```
# Resource group name - we are using myImageBuilderRG in this example
imageResourceGroup=myImageBuilderRGLinux
# Datacenter location - we are using West US 2 in this example
location=WestUS2
# Name for the image - we are using myBuilderImage in this example
imageName=myBuilderImage
# Run output name
runOutputName=aibLinux
```

Create a variable for your subscription ID. You can get this using `az account show | grep id`.

```
subscriptionID=<Your subscription ID>
```

Create the resource group.

This is used to store the image configuration template artifact and the image.

```
az group create -n $imageResourceGroup -l $location
```

Set permissions on the resource group

Give Image Builder 'contributor' permission to create the image in the resource group. Without the proper permissions, the image build will fail.

The `--assignee` value is the app registration ID for the Image Builder service.

```
az role assignment create \
  --assignee cf32a0cc-373c-47c9-9156-0db11f6a6dfc \
  --role Contributor \
  --scope /subscriptions/$subscriptionID/resourceGroups/$imageResourceGroup
```

Download the template example

A parameterized sample image configuration template has been created for you to use. Download the sample json file and configure it with the variables you set earlier.

```
curl
https://raw.githubusercontent.com/danielsollondon/azvmimagebuilder/master/quickstarts/0_Creating_a_Custom_Linux_Managed_Image/helloImageTemplateLinux.json -o helloImageTemplateLinux.json

sed -i -e "s/<subscriptionID>/$subscriptionID/g" helloImageTemplateLinux.json
sed -i -e "s/<rgName>/$imageResourceGroup/g" helloImageTemplateLinux.json
sed -i -e "s/<region>/$location/g" helloImageTemplateLinux.json
sed -i -e "s/<imageName>/$imageName/g" helloImageTemplateLinux.json
sed -i -e "s/<runOutputName>/$runOutputName/g" helloImageTemplateLinux.json
```

You can modify this example .json as needed. For example, you can increase the value of `buildTimeoutInMinutes` to allow for longer running builds. You can edit the file in Cloud Shell using `vi`.

```
vi helloImageTemplateLinux.json
```

NOTE

For source image, you must always [specify a version](#), you cannot use `latest`.

If you add or change the resource group where the image is being distributed, you need to make sure the [permissions are set for the resource group](#).

Submit the image configuration

Submit the image configuration to the VM Image Builder service

```
az resource create \
--resource-group $imageResourceGroup \
--properties @helloImageTemplateLinux.json \
--is-full-object \
--resource-type Microsoft.VirtualMachineImages/imageTemplates \
-n helloImageTemplateLinux01
```

If it completes successfully, it will return a success message, and create an image builder configuration template artifact in the \$imageResourceGroup. You can see the resource group in the portal if you enable 'Show hidden types'.

Also, in the background, Image Builder creates a staging resource group in your subscription. Image Builder uses the staging resource group for the image build. The name of the resource group will be in this format:

`IT_<DestinationResourceGroup>_<TemplateName>`.

IMPORTANT

Do not delete the staging resource group directly. If you delete the image template artifact, it will automatically delete the staging resource group. For more information, see the [Clean up](#) section at the end of this article.

If the service reports a failure during the image configuration template submission, see the [troubleshooting](#) steps. You will also need to delete the template before you retry submitting the build. To delete the template:

```
az resource delete \
--resource-group $imageResourceGroup \
--resource-type Microsoft.VirtualMachineImages/imageTemplates \
-n helloImageTemplateLinux01
```

Start the image build

Start the image build.

```
az resource invoke-action \
--resource-group $imageResourceGroup \
--resource-type Microsoft.VirtualMachineImages/imageTemplates \
-n helloImageTemplateLinux01 \
--action Run
```

Wait until the build is complete, for this example, it can take 10-15 minutes.

If you encounter any errors, please review these [troubleshooting](#) steps.

Create the VM

Create the VM using the image you built.

```
az vm create \
--resource-group $imageResourceGroup \
--name myVM \
--admin-username azureuser \
--image $imageName \
--location $location \
--generate-ssh-keys
```

Get the IP address from the output of creating the VM and use it to SSH to the VM.

```
ssh azureuser@<pubIp>
```

You should see the image was customized with a Message of the Day as soon as your SSH connection is established!

```
*****
**          This VM was built from the:      **
**      !! AZURE VM IMAGE BUILDER Custom Image !!  **
**          You have just been Customized :-)      **
*****
```

Type `exit` when you are done to close the SSH connection.

Check the source

In the Image Builder Template, in the 'Properties', you will see the source image, customization script it runs, and where it is distributed.

```
cat helloImageTemplateLinux.json
```

For more detailed information about this json file, see [Image builder template reference](#)

Clean up

When you are done, you can delete the resources.

Delete the image builder template.

```
az resource delete \
--resource-group $imageResourceGroup \
--resource-type Microsoft.VirtualMachineImages/imageTemplates \
-n helloImageTemplateLinux01
```

Delete the image resource group.

```
az group delete -n $imageResourceGroup
```

Next steps

To learn more about the components of the json file used in this article, see [Image Builder template reference](#).

Preview: Create an Azure Image Builder template

5/10/2019 • 12 minutes to read • [Edit Online](#)

Azure Image Builder uses a json file to pass information into the Image Builder service. In this article we will go over the sections of the json file, so you can build your own. To see examples of full .json files, see the [Azure Image Builder GitHub](#).

This is the basic template format:

```
{  
  "type": "Microsoft.VirtualMachineImages/imageTemplates",  
  "apiVersion": "2019-05-01-preview",  
  "location": "<region>",  
  "tags": {  
    "<name>": "<value>",  
    "<name>": "<value>"  
  },  
  "identity": {},  
  "dependsOn": [],  
  "properties": {  
    "buildTimeoutInMinutes": <minutes>,  
    "build": {},  
    "customize": {},  
    "distribute": {}  
  }  
}
```

Type and API version

The `type` is the resource type, which must be `"Microsoft.VirtualMachineImages/imageTemplates"`. The `apiVersion` will change over time as the API changes, but should be `"2019-05-01-preview"` for preview.

```
"type": "Microsoft.VirtualMachineImages/imageTemplates",  
"apiVersion": "2019-05-01-preview",
```

Location

The location is the region where the custom image will be created. For the Image Builder preview, the following regions are supported:

- East US
- East US 2
- West Central US
- West US
- West US 2

```
"location": "<region>",
```

Depends on (optional)

This optional section can be used to ensure that dependencies are completed before proceeding.

```
"dependsOn": [],
```

For more information, see [Define resource dependencies](#).

Identity

By default, Image Builder supports using scripts, or copying files from multiple locations, such as GitHub and Azure storage. To use these, they must be publicly accessible.

You can also use an Azure User-Assigned Managed Identity, defined by you, to allow Image Builder access Azure Storage, as long as the identity has been granted a minimum of 'Storage Blob Data Reader' on the Azure storage account. This means you do not need to make the storage blobs externally accessible, or setup SAS Tokens.

```
"identity": {  
    "type": "UserAssigned",  
    "userAssignedIdentities": {  
        "<imgBuilderId>": {}  
    }  
},
```

For a complete example, see [Use an Azure User-Assigned Managed Identity to access files in Azure Storage](#).

Image Builder support for a User-Assigned Identity:

- Supports a single identity only
- Does not support custom domain names

To learn more, see [What is managed identities for Azure resources?](#). For more information on deploying this feature, see [Configure managed identities for Azure resources on an Azure VM using Azure CLI](#).

Properties: source

The `source` section contains information about the source image that will be used by Image Builder.

The API requires a 'SourceType' that defines the source for the image build, currently there are three types:

- ISO - use this when the source is a RHEL ISO.
- PlatformImage - indicated the source image is a Marketplace image.
- ManagedImage - use this when starting from a regular managed image.
- SharedImageVersion - this is used when you are using an image version in a Shared Image Gallery as the source.

ISO source

Azure Image Builder only supports using published Red Hat Enterprise Linux 7.x Binary DVD ISOs, for preview. Image Builder supports:

- RHEL 7.3
- RHEL 7.4
- RHEL 7.5

```
"source": {  
    "type": "ISO",  
    "sourceURI": "<sourceURI from the download center>",  
    "sha256Checksum": "<checksum associated with ISO>"  
}
```

To get the `sourceURI` and `sha256Checksum` values, go to <https://access.redhat.com/downloads> then select the product **Red Hat Enterprise Linux**, and a supported version.

In the list of **Installers and Images for Red Hat Enterprise Linux Server**, you need to copy the link for Red Hat Enterprise Linux 7.x Binary DVD, and the checksum.

NOTE

The access tokens of the links are refreshed at frequent intervals, so every time you want to submit a template, you must check if the RH link address has changed.

PlatformImage source

Azure Image Builder supports the following Azure Marketplace images:

- Ubuntu 18.04
- Ubuntu 16.04
- RHEL 7.6
- CentOS 7.6
- Windows 2016
- Windows 2019

```
"source": {  
    "type": "PlatformImage",  
    "publisher": "Canonical",  
    "offer": "UbuntuServer",  
    "sku": "18.04-LTS",  
    "version": "18.04.201903060"  
},
```

The properties here are the same that are used to create VM's, using AZ CLI, run the below to get the properties:

```
az vm image list -l westus -f UbuntuServer -p Canonical --output table --all
```

NOTE

Version cannot be 'latest', you must use the command above to get a version number.

ManagedImage source

Sets the source image as an existing managed image of a generalized VHD or VM. The source managed image must be of a supported OS, and be in the same region as your Azure Image Builder template.

```
"source": {  
    "type": "ManagedImage",  
    "imageId":  
        "/subscriptions/<subscriptionId>/resourceGroups/{destinationResourceGroupName}/providers/Microsoft.Compute/images/<imageName>"  
}
```

The `imageId` should be the ResourceId of the managed image. Use `az image list` to list available images.

SharedImageVersion source

Sets the source image an existing image version in a Shared Image Gallery. The image version must be of a supported OS, and the image must be replicated to the same region as your Azure Image Builder template.

```
"source": {  
    "type": "SharedImageVersion",  
    "imageVersionID": "/subscriptions/<subscriptionId>/resourceGroups/<resourceGroup>/providers/Microsoft.Compute/galleries/<sharedImageGalleryName>/images/<imageDefinitionName>/versions/<imageVersion>"  
}
```

The `imageVersionId` should be the ResourceId of the image version. Use [az sig image-version list](#) to list image versions.

Properties: customize

Image Builder supports multiple 'customizers'. Customizers are functions that are used to customize your image, such as running scripts, or rebooting servers.

When using `customize`:

- You can use multiple customizers, but they must have a unique `name`.
- Customizers execute in the order specified in the template.
- If one customizer fails, then the whole customization component will fail and report back an error.
- Consider how much time your image build will require, and adjust the 'buildTimeoutInMinutes' property to allow image builder enough time to complete.
- It is strongly advised you test the script thoroughly before using it in a template. Debugging the script on your own VM will be easier.
- Do not put sensitive data in the scripts.
- The script locations need to be publicly accessible, unless you are using [MSI](#).

```
"customize": [  
    {  
        "type": "Shell",  
        "name": "<name>",  
        "scriptUri": "<path to script>"  
    },  
    {  
        "type": "Shell",  
        "name": "<name>",  
        "inline": [  
            "<command to run inline>"  
        ]  
    }  
,  
],
```

The customize section is an array. Azure Image Builder will run through the customizers in sequential order. Any failure in any customizer will fail the build process.

Shell customizer

The shell customizer supports running shell scripts, these must be publicly accessible for the IB to access them.

```

"customize": [
  {
    "type": "Shell",
    "name": "<name>",
    "scriptUri": "<link to script>"
  },
],
"customize": [
{
  "type": "Shell",
  "name": "<name>",
  "inline": "<commands to run>"
},
],

```

OS Support: Linux

Customize properties:

- **type** – Shell
- **name** - name for tracking the customization
- **scriptUri** - URI to the location of the file
- **inline** - array of shell commands, separated by commas.

NOTE

When running the shell customizer with RHEL ISO source, you need to ensure your first customization shell handles registering with a Red Hat entitlement server before any customization occurs. Once customization is complete, the script should unregister with the entitlement server.

Windows restart customizer

The Restart customizer allows you to restart a Windows VM and wait for it come back online, this allows you to install software that requires a reboot.

```

"customize": [
  "type": "WindowsRestart",
  "restartCommand": "shutdown /r /f /t 0 /c",
  "restartCheckCommand": "echo Azure-Image-Builder-Restarted-the-VM >
buildArtifacts/azureImageBuilderRestart.txt",
  "restartTimeout": "5m"
],

```

OS Support: Windows

Customize properties:

- **Type:** WindowsRestart
- **restartCommand** - Command to execute the restart (optional). The default is `'shutdown /r /f /t 0 /c \"packer restart\"'`.
- **restartCheckCommand** – Command to check if restart succeeded (optional).
- **restartTimeout** - Restart timeout specified as a string of magnitude and unit. For example, `5m` (5 minutes) or `2h` (2 hours). The default is: '5m'

PowerShell customizer

The shell customizer supports running PowerShell scripts and inline command, the scripts must be publicly accessible for the IB to access them.

```

"customize": [
  {
    "type": "PowerShell",
    "name": "<name>",
    "scriptUri": "<path to script>"
  },
  {
    "type": "PowerShell",
    "name": "<name>",
    "inline": "<PowerShell syntax to run>",
    "valid_exit_codes": "<exit code>"
  }
],

```

OS support: Windows and Linux

Customize properties:

- **type** – PowerShell.
- **scriptUri** - URI to the location of the PowerShell script file.
- **inline** – Inline commands to be run, separated by commas.
- **valid_exit_codes** – Optional, valid codes that can be returned from the script/inline command, this will avoid reported failure of the script/inline command.

File customizer

The File customizer lets image builder download a file from a GitHub or Azure storage. If you have an image build pipeline that relies on build artifacts, you can then set the file customizer to download from the build share, and move the artifacts into the image.

```

"customize": [
  {
    "type": "File",
    "name": "<name>",
    "sourceUri": "<source location>",
    "destination": "<destination>"
  }
]

```

OS support: Linux and Windows

File customizer properties:

- **sourceUri** - an accessible storage endpoint, this can be GitHub or Azure storage. You can only download one file, not an entire directory. If you need to download a directory, use a compressed file, then uncompress it using the Shell or PowerShell customizers.
- **destination** – this is the full destination path and file name. Any referenced path and subdirectories must exist, use the Shell or PowerShell customizers to set these up beforehand. You can use the script customizers to create the path.

This is supported by Windows directories and Linux paths, but there are some differences:

- Linux OS's – the only path Image builder can write to is /tmp.
- Windows – No path restriction, but the path must exist.

If there is an error trying to download the file, or put it in a specified directory, the customize step will fail, and this will be in the customization.log.

Files in the File customizer can be downloaded from Azure Storage using [MSI](#).

Generalize

By default, Azure Image Builder will also run 'deprovision' code at the end of each image customization phase, to 'generalize' the image. Generalizing is a process where the image is set up so it can be reused to create multiple VMs. For Windows VMs, Azure Image Builder uses Sysprep. For Linux, Azure Image Builder runs 'waagent - deprovision'.

The commands Image Builder users to generalize may not be suitable for every situation, so Azure Image Builder will allow you to customize this command, if needed.

If you are migrating existing customization, and you are using different Sysprep/waagent commands, you can use the Image Builder generic commands, and if the VM creation fails, use your own Sysprep or waagent commands.

If Azure Image Builder creates a Windows custom image successfully, and you create a VM from it, then find that the VM creation fails or does not complete successfully, you will need to review the Windows Server Sysprep documentation or raise a support request with the Windows Server Sysprep Customer Services Support team, who can troubleshoot and advise on the correct Sysprep usage.

Default Sysprep command

```
echo '>>> Waiting for GA to start ...'
while ((Get-Service RdAgent).Status -ne 'Running') { Start-Sleep -s 5 }
while ((Get-Service WindowsAzureTelemetryService).Status -ne 'Running') { Start-Sleep -s 5 }
while ((Get-Service WindowsAzureGuestAgent).Status -ne 'Running') { Start-Sleep -s 5 }
echo '>>> Sysprepping VM ...'
if( Test-Path $Env:SystemRoot\windows\system32\Sysprep\unattend.xml ){ rm
$Env:SystemRoot\windows\system32\Sysprep\unattend.xml -Force} &
$Env:SystemRoot\System32\Sysprep\Sysprep.exe /oobe /generalize /quiet /quit
while($true) { $imageState = Get-ItemProperty
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Setup\State | Select ImageState;
if($imageState.ImageState -ne 'IMAGE_STATE_GENERALIZE_RESEAL_TO_OOBE') { Write-Output $imageState.ImageState;
Start-Sleep -s 5 } else { break } }
```

Default Linux deprovision command

```
/usr/sbin/waagent -force -deprovision+user && export HISTSIZE=0 && sync
```

Overriding the Commands

To override the commands, use the PowerShell or Shell script provisioners to create the command files with the exact file name, and put them in the correct directories:

- Windows: c:\DeprovisioningScript.ps1
- Linux: /tmp/DeprovisioningScript.sh

Image Builder will read these commands, these are written out to the AIB logs, 'customization.log'. See [troubleshooting](#) on how to collect logs.

Properties: distribute

Azure Image Builder supports three distribution targets:

- **managedImage** - managed image.
- **sharedImage** - Shared Image Gallery.
- **VHD** - VHD in a storage account.

You can distribute an image to both of the target types in the same configuration, please see [examples](#).

Because you can have more than one target to distribute to, Image Builder maintains a state for every distribution

target that can be accessed by querying the `runOutputName`. The `runOutputName` is an object you can query post distribution for information about that distribution. For example, you can query the location of the VHD, or regions where the image version was replicated to. This is a property of every distribution target. The `runOutputName` must be unique to each distribution target.

Distribute: managedImage

The image output will be a managed image resource.

```
"distribute": [
    {
        "type": "managedImage",
        "imageId": "<resource ID>",
        "location": "<region>",
        "runOutputName": "<name>",
        "artifactTags": {
            "<name>": "<value>",
            "<name>": "<value>"
        }
    }
]
```

Distribute properties:

- **type** – managedImage
- **imageId** – Resource ID of the destination image, expected format:
`/subscriptions//resourceGroups//providers/Microsoft.Compute/images/`
- **location** - location of the managed image.
- **runOutputName** – unique name for identifying the distribution.
- **artifactTags** - Optional user specified key value pair tags.

NOTE

The destination resource group must exist. If you want the image distributed to a different region, it will increase the deployment time .

Distribute: sharedImage

The Azure Shared Image Gallery is a new Image Management service that allows managing of image region replication, versioning and sharing custom images. Azure Image Builder supports distributing with this service, so you can distribute images to regions supported by Shared Image Galleries.

A Shared Image Gallery is made up of:

- Gallery - Container for multiple shared images. A gallery is deployed in one region.
- Image definitions - a conceptual grouping for images.
- Image versions - this is an image type used for deploying a VM or scale set. Image versions can be replicated to other regions where VMs need to be deployed.

Before you can distribute to the Image Gallery, you must create a gallery and an image definition, see [Shared images](#).

```
{
    "type": "sharedImage",
    "galleryImageId": "<resource ID>",
    "runOutputName": "<name>",
    "artifactTags": {
        "<name>": "<value>",
        "<name>": "<value>"
    }
    "replicationRegions": [
        "<region where the gallery is deployed>",
        "<region>"
    ]
}
```

Distribute properties for shared image galleries:

- **type** - sharedImage
- **galleryImageId** – ID of the shared image gallery. The format is:
/subscriptions//resourceGroups//providers/Microsoft.Compute/galleries//images/.
- **runOutputName** – unique name for identifying the distribution.
- **artifactTags** - Optional user specified key value pair tags.
- **replicationRegions** - Array of regions for replication. One of the regions must be the region where the Gallery is deployed.

NOTE

You can use Azure Image Builder in a different region to the gallery, but the Azure Image Builder service will need to transfer the image between the datacenters and this will take longer. Image Builder will automatically version the image, based on a monotonic integer, you cannot specify it currently.

Distribute: VHD

You can output to a VHD. You can then copy the VHD, and use it to publish to Azure MarketPlace, or use with Azure Stack.

```
{
    "type": "VHD",
    "runOutputName": "<VHD name>",
    "tags": {
        "<name>": "<value>",
        "<name>": "<value>"
    }
}
```

OS Support: Windows and Linux

Distribute VHD parameters:

- **type** - VHD.
- **runOutputName** – unique name for identifying the distribution.
- **tags** - Optional user specified key value pair tags.

Azure Image Builder does not allow the user to specify a storage account location, but you can query the status of the `runOutputs` to get the location.

```
az resource show \
--ids
"/subscriptions/$subscriptionId/resourcegroups/<imageResourceGroup>/providers/Microsoft.VirtualMachineImages/i
mageTemplates/<imageTemplateName>/runOutputs/<runOutputName>" | grep artifactUri
```

NOTE

Once the VHD has been created, copy it to a different location, as soon as possible. The VHD is stored in a storage account in the temporary resource group created when the image template is submitted to the Azure Image Builder service. If you delete the image template, then you will lose the VHD.

Next steps

There are sample .json files for different scenarios in the [Azure Image Builder GitHub](#).

Preview: Create a Linux image and distribute it to a Shared Image Gallery

5/9/2019 • 4 minutes to read • [Edit Online](#)

This article shows you how you can use the Azure Image Builder to create an image version in a [Shared Image Gallery](#), then distribute the image globally.

We will be using a sample json template to configure the image. The json file we are using is here: [helloImageTemplateforSIG.json](#).

To distribute the image to a Shared Image Gallery, the template uses `sharedImage` as the value for the `distribute` section of the template.

IMPORTANT

Azure Image Builder 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](#).

Register the features

To use Azure Image Builder during the preview, you need to register the new feature.

```
az feature register --namespace Microsoft.VirtualMachineImages --name VirtualMachineTemplatePreview
```

Check the status of the feature registration.

```
az feature show --namespace Microsoft.VirtualMachineImages --name VirtualMachineTemplatePreview | grep state
```

Check your registration.

```
az provider show -n Microsoft.VirtualMachineImages | grep registrationState  
az provider show -n Microsoft.Storage | grep registrationState
```

If they do not say registered, run the following:

```
az provider register -n Microsoft.VirtualMachineImages  
az provider register -n Microsoft.Storage
```

Set variables and permissions

We will be using some pieces of information repeatedly, so we will create some variables to store that information.

For Preview, image builder will only support creating custom images in the same Resource Group as the source managed image. Update the resource group name in this example to be the same resource group as your source managed image.

```
# Resource group name - we are using ibLinuxGalleryRG in this example
sigResourceGroup=ibLinuxGalleryRG
# Datacenter location - we are using West US 2 in this example
location=westus2
# Additional region to replicate the image to - we are using East US in this example
additionalRegion=eastus
# name of the shared image gallery - in this example we are using myGallery
sigName=myIbGallery
# name of the image definition to be created - in this example we are using myImageDef
imageDefName=myIbImageDef
# image distribution metadata reference name
runOutputName=aibLinuxSIG
```

Create a variable for your subscription ID. You can get this using `az account show | grep id`.

```
subscriptionID=<Subscription ID>
```

Create the resource group.

```
az group create -n $sigResourceGroup -l $location
```

Give Azure Image Builder permission to create resources in that resource group. The `--assignee` value is the app registration ID for the Image Builder service.

```
az role assignment create \
--assignee cf32a0cc-373c-47c9-9156-0db11f6a6dfc \
--role Contributor \
--scope /subscriptions/$subscriptionID/resourceGroups/$sigResourceGroup
```

Create an image definition and gallery

Create an image gallery.

```
az sig create \
-g $sigResourceGroup \
--gallery-name $sigName
```

Create an image definition.

```
az sig image-definition create \
-g $sigResourceGroup \
--gallery-name $sigName \
--gallery-image-definition $imageDefName \
--publisher myIbPublisher \
--offer myOffer \
--sku 18.04-LTS \
--os-type Linux
```

Download and configure the .json

Download the .json template and configure it with your variables.

```
curl  
https://raw.githubusercontent.com/danielsollondon/azvmimagebuilder/master/quickquickstarts/1_Creating_a_Custom_  
Linux_Shared_Image_Gallery_Image/helloImageTemplateforSIG.json -o helloImageTemplateforSIG.json  
sed -i -e "s/<subscriptionID>/$subscriptionID/g" helloImageTemplateforSIG.json  
sed -i -e "s/<rgName>/$sigResourceGroup/g" helloImageTemplateforSIG.json  
sed -i -e "s/<imageDefName>/$imageDefName/g" helloImageTemplateforSIG.json  
sed -i -e "s/<sharedImageGalName>/$sigName/g" helloImageTemplateforSIG.json  
sed -i -e "s/<region1>/$location/g" helloImageTemplateforSIG.json  
sed -i -e "s/<region2>/$additionalRegion/g" helloImageTemplateforSIG.json  
sed -i -e "s/<runOutputName>/$runOutputName/g" helloImageTemplateforSIG.json
```

Create the image version

This next part will create the image version in the gallery.

Submit the image configuration to the Azure Image Builder service.

```
az resource create \  
    --resource-group $sigResourceGroup \  
    --properties @helloImageTemplateforSIG.json \  
    --is-full-object \  
    --resource-type Microsoft.VirtualMachineImages/imageTemplates \  
    -n helloImageTemplateforSIG01
```

Start the image build.

```
az resource invoke-action \  
    --resource-group $sigResourceGroup \  
    --resource-type Microsoft.VirtualMachineImages/imageTemplates \  
    -n helloImageTemplateforSIG01 \  
    --action Run
```

Creating the image and replicating it to both regions can take a while. Wait until this part is finished before moving on to creating a VM.

Create the VM

Create a VM from the image version that was created by Azure Image Builder.

```
az vm create \  
    --resource-group $sigResourceGroup \  
    --name myAibGalleryVM \  
    --admin-username aibuser \  
    --location $location \  
    --image  
    "/subscriptions/$subscriptionID/resourceGroups/$sigResourceGroup/providers/Microsoft.Compute/galleries/$sigName  
    /images/$imageDefName/versions/latest" \  
    --generate-ssh-keys
```

SSH into the VM.

```
ssh aibuser@<publicIpAddress>
```

You should see the image was customized with a *Message of the Day* as soon as your SSH connection is established!

```
*****
** This VM was built from the:      **
** !! AZURE VM IMAGE BUILDER Custom Image !!  **
** You have just been Customized :-)  **
*****
```

Clean up resources

If you want to now try re-customizing the image version to create a new version of the same image, skip the next steps and go on to [Use Azure Image Builder to create another image version](#).

This will delete the image that was created, along with all of the other resource files. Make sure you are finished with this deployment before deleting the resources.

When deleting image gallery resources, you need delete all of the image versions before you can delete the image definition used to create them. To delete a gallery, you first need to have deleted all of the image definitions in the gallery.

Delete the image builder template.

```
az resource delete \
--resource-group $sigResourceGroup \
--resource-type Microsoft.VirtualMachineImages/imageTemplates \
-n helloImageTemplateforSIG01
```

Get the image version created by image builder, this always starts with `0.`, and then delete the image version

```
sigDefImgVersion=$(az sig image-version list \
-g $sigResourceGroup \
--gallery-name $sigName \
--gallery-image-definition $imageDefName \
--subscription $subscriptionID --query [].'name' -o json | grep 0. | tr -d '')\n\naz sig image-version delete \
-g $sigResourceGroup \
--gallery-image-version $sigDefImgVersion \
--gallery-name $sigName \
--gallery-image-definition $imageDefName \
--subscription $subscriptionID
```

Delete the image definition.

```
az sig image-definition delete \
-g $sigResourceGroup \
--gallery-name $sigName \
--gallery-image-definition $imageDefName \
--subscription $subscriptionID
```

Delete the gallery.

```
az sig delete -r $sigName -g $sigResourceGroup
```

Delete the resource group.

```
az group delete -n $sigResourceGroup -y
```

Next steps

Learn more about [Azure Shared Image Galleries](#).

Preview: Create a new image version from an existing image version using Azure Image Builder

5/6/2019 • 3 minutes to read • [Edit Online](#)

This article shows you how to take an existing image version in a [Shared Image Gallery](#), update it, and publish it as a new image version to the gallery.

We will be using a sample json template to configure the image. The json file we are using is here: [helloImageTemplateforSIGfromSIG.json](#).

Register the features

To use Azure Image Builder during the preview, you need to register the new feature.

```
az feature register --namespace Microsoft.VirtualMachineImages --name VirtualMachineTemplatePreview
```

Check the status of the feature registration.

```
az feature show --namespace Microsoft.VirtualMachineImages --name VirtualMachineTemplatePreview | grep state
```

Check your registration.

```
az provider show -n Microsoft.VirtualMachineImages | grep registrationState  
az provider show -n Microsoft.Storage | grep registrationState
```

If they do not say registered, run the following:

```
az provider register -n Microsoft.VirtualMachineImages  
az provider register -n Microsoft.Storage
```

Set variables and permissions

If you used [Create an image and distribute to a Shared Image Gallery](#) to create your Shared Image Gallery, you've already created some of the variables we need. If not, please setup some variables to be used for this example.

For Preview, image builder will only support creating custom images in the same Resource Group as the source managed image. Update the resource group name in this example to be the same resource group as your source managed image.

```
# Resource group name
sigResourceGroup=ibLinuxGalleryRG
# Gallery location
location=westus2
# Additional region to replicate the image version to
additionalRegion=eastus
# Name of the shared image gallery
sigName=myIbGallery
# Name of the image definition to use
imageDefName=myIbImageDef
# image distribution metadata reference name
runOutputName=aibSIGLinuxUpdate
```

Create a variable for your subscription ID. You can get this using `az account show | grep id`.

```
subscriptionID=<Subscription ID>
```

Get the image version that you want to update.

```
sigDefImgVersionId=$(az sig image-version list \
    -g $sigResourceGroup \
    --gallery-name $sigName \
    --gallery-image-definition $imageDefName \
    --subscription $subscriptionID --query [].'id' -o json | grep 0. | tr -d '"' | tr -d '[:space:]')
```

If you already have your own Shared Image Gallery, and did not follow the previous example, you will need to assign permissions for Image Builder to access the Resource Group, so it can access the gallery.

```
az role assignment create \
    --assignee cf32a0cc-373c-47c9-9156-0db11f6a6dfc \
    --role Contributor \
    --scope /subscriptions/$subscriptionID/resourceGroups/$sigResourceGroup
```

Modify helloImage example

You can review the example we are about to use by opening the json file here: [helloImageTemplateforSIGfromSIG.json](#) along with the [Image Builder template reference](#).

Download the json example and configure it with your variables.

```
curl
https://raw.githubusercontent.com/danielsollondon/azvmimagebuilder/master/quickstarts/8_Creating_a_Custom_Linux_Shared_Image_Gallery_Image_from_SIG/helloImageTemplateforSIGfromSIG.json -o
helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<subscriptionID>/$subscriptionID/g" helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<rgName>/$sigResourceGroup/g" helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<imageDefName>/$imageDefName/g" helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<sharedImageGalName>/$sigName/g" helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<sigDefImgVersionId>/$sigDefImgVersionId/g" helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<region1>/$location/g" helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<region2>/$additionalRegion/g" helloImageTemplateforSIGfromSIG.json
sed -i -e "s/<runOutputName>/$runOutputName/g" helloImageTemplateforSIGfromSIG.json
```

Create the image

Submit the image configuration to the VM Image Builder Service.

```
az resource create \
--resource-group $sigResourceGroup \
--properties @helloImageTemplateforSIGfromSIG.json \
--is-full-object \
--resource-type Microsoft.VirtualMachineImages/imageTemplates \
-n helloImageTemplateforSIGfromSIG01
```

Start the image build.

```
az resource invoke-action \
--resource-group $sigResourceGroup \
--resource-type Microsoft.VirtualMachineImages/imageTemplates \
-n helloImageTemplateforSIGfromSIG01 \
--action Run
```

Wait until the image has been built and replication before moving on to the next step.

Create the VM

```
az vm create \
--resource-group $sigResourceGroup \
--name aibImgVm001 \
--admin-username azureuser \
--location $location \
--image
"/subscriptions/$subscriptionID/resourceGroups/$sigResourceGroup/providers/Microsoft.Compute/galleries/$imageName/images/$imageDefName/versions/latest" \
--generate-ssh-keys
```

Create an SSH connection to the VM using the public IP address of the VM.

```
ssh azureuser@<pubIp>
```

You should see the image was customized with a "Message of the Day" as soon as your SSH connection is established.

```
*****
**      This VM was built from the:          **
**      !! AZURE VM IMAGE BUILDER Custom Image !!  **
**      You have just been Customized :-)        **
*****
```

Type `exit` to close the SSH connection.

You can also list the image versions that are now available in your gallery.

```
az sig image-version list -g $sigResourceGroup -r $sigName -i $imageDefName -o table
```

Next steps

To learn more about the components of the json file used in this article, see [Image builder template reference](#).

Find Linux VM images in the Azure Marketplace with the Azure CLI

3/26/2019 • 9 minutes to read • [Edit Online](#)

This topic describes how to use the Azure CLI to find VM images in the Azure Marketplace. Use this information to specify a Marketplace image when you create a VM programmatically with the CLI, Resource Manager templates, or other tools.

Also browse available images and offers using the [Azure Marketplace](#) storefront, the [Azure portal](#), or [Azure PowerShell](#).

Make sure that you installed the latest [Azure CLI](#) and are logged in to an Azure account (`az login`).

Terminology

A Marketplace image in Azure has the following attributes:

- **Publisher:** The organization that created the image. Examples: Canonical, MicrosoftWindowsServer
- **Offer:** The name of a group of related images created by a publisher. Examples: UbuntuServer, WindowsServer
- **SKU:** An instance of an offer, such as a major release of a distribution. Examples: 18.04-LTS, 2019-Datacenter
- **Version:** The version number of an image SKU.

To identify a Marketplace image when you deploy a VM programmatically, supply these values individually as parameters. Some tools accept an image *URN*, which combines these values, separated by the colon (:) character: *Publisher:Offer:SKU:Version*. In a URN, you can replace the version number with "latest", which selects the latest version of the image.

If the image publisher provides additional license and purchase terms, then you must accept those terms and enable programmatic deployment. You'll also need to supply *purchase plan* parameters when deploying a VM programmatically. See [Deploy an image with Marketplace terms](#).

List popular images

Run the `az vm image list` command, without the `--all` option, to see a list of popular VM images in the Azure Marketplace. For example, run the following command to display a cached list of popular images in table format:

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

The output includes the image URN (the value in the *Urn* column). When creating a VM with one of these popular Marketplace images, you can alternatively specify the *UrnAlias*, a shortened form such as *UbuntuLTS*.

You are viewing an offline list of images, use --all to retrieve an up-to-date list			
Offer UrnAlias	Publisher Version	Sku	Urn
CentOS	OpenLogic	7.5	OpenLogic:CentOS:7.5:latest
Centos	latest		
CoreOS	CoreOS	Stable	CoreOS:CoreOS:Stable:latest
CoreOS	latest		
Debian	credativ	8	credativ:Debian:8:latest
Debian	latest		
openSUSE-Leap	SUSE	42.3	SUSE:openSUSE-Leap:42.3:latest
openSUSE-Leap	latest		
RHEL	RedHat	7-RAW	RedHat:RHEL:7-RAW: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		
...			

Find specific images

To find a specific VM image in the Marketplace, use the `az vm image list` command with the `--all` option. This version of the command takes some time to complete and can return lengthy output, so you usually filter the list by `--publisher` or another parameter.

For example, the following command displays all Debian offers (remember that without the `--all` switch, it only searches the local cache of common images):

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

Partial output:

Offer Version	Publisher	Sku	Urn
Debian 7.0.201602010	credativ	7	credativ:Debian:7:7.0.201602010
Debian 7.0.201603020	credativ	7	credativ:Debian:7:7.0.201603020
Debian 7.0.201604050	credativ	7	credativ:Debian:7:7.0.201604050
Debian 7.0.201604200	credativ	7	credativ:Debian:7:7.0.201604200
Debian 7.0.201606280	credativ	7	credativ:Debian:7:7.0.201606280
Debian 7.0.201609120	credativ	7	credativ:Debian:7:7.0.201609120
Debian 7.0.201611020	credativ	7	credativ:Debian:7:7.0.201611020
Debian 7.0.201701180	credativ	7	credativ:Debian:7:7.0.201701180
Debian 8.0.201602010	credativ	8	credativ:Debian:8:8.0.201602010
Debian 8.0.201603020	credativ	8	credativ:Debian:8:8.0.201603020
Debian 8.0.201604050	credativ	8	credativ:Debian:8:8.0.201604050
Debian	credativ	8	credativ:Debian:8:8.0.201604200

8.0.201604200			
Debian	credativ	8	credativ:Debian:8:8.0.201606280
8.0.201606280			
Debian	credativ	8	credativ:Debian:8:8.0.201609120
8.0.201609120			
Debian	credativ	8	credativ:Debian:8:8.0.201611020
8.0.201611020			
Debian	credativ	8	credativ:Debian:8:8.0.201701180
8.0.201701180			
Debian	credativ	8	credativ:Debian:8:8.0.201703150
8.0.201703150			
Debian	credativ	8	credativ:Debian:8:8.0.201704110
8.0.201704110			
Debian	credativ	8	credativ:Debian:8:8.0.201704180
8.0.201704180			
Debian	credativ	8	credativ:Debian:8:8.0.201706190
8.0.201706190			
Debian	credativ	8	credativ:Debian:8:8.0.201706210
8.0.201706210			
Debian	credativ	8	credativ:Debian:8:8.0.201708040
8.0.201708040			
Debian	credativ	8	credativ:Debian:8:8.0.201710090
8.0.201710090			
Debian	credativ	8	credativ:Debian:8:8.0.201712040
8.0.201712040			
Debian	credativ	8	credativ:Debian:8:8.0.201801170
8.0.201801170			
Debian	credativ	8	credativ:Debian:8:8.0.201803130
8.0.201803130			
Debian	credativ	8	credativ:Debian:8:8.0.201803260
8.0.201803260			
Debian	credativ	8	credativ:Debian:8:8.0.201804020
8.0.201804020			
Debian	credativ	8	credativ:Debian:8:8.0.201804150
8.0.201804150			
Debian	credativ	8	credativ:Debian:8:8.0.201805160
8.0.201805160			
Debian	credativ	8	credativ:Debian:8:8.0.201807160
8.0.201807160			
Debian	credativ	8	credativ:Debian:8:8.0.201901221
8.0.201901221			
...			

Apply similar filters with the `--location`, `--publisher`, and `--sku` options. You can perform partial matches on a filter, such as searching for `--offer Deb` to find all Debian images.

If you don't specify a particular location with the `--location` option, the values for the default location are returned. (Set a different default location by running `az configure --defaults location=<location>`.)

For example, the following command lists all Debian 8 SKUs in the West Europe location:

```
az vm image list --location westeurope --offer Deb --publisher credativ --sku 8 --all --output table
```

Partial output:

Offer	Publisher	Sku	Urn	Version
Debian	credativ	8	credativ:Debian:8:8.0.201602010	8.0.201602010
Debian	credativ	8	credativ:Debian:8:8.0.201603020	8.0.201603020
Debian	credativ	8	credativ:Debian:8:8.0.201604050	8.0.201604050
Debian	credativ	8	credativ:Debian:8:8.0.201604200	8.0.201604200
Debian	credativ	8	credativ:Debian:8:8.0.201606280	8.0.201606280
Debian	credativ	8	credativ:Debian:8:8.0.201609120	8.0.201609120
Debian	credativ	8	credativ:Debian:8:8.0.201611020	8.0.201611020
Debian	credativ	8	credativ:Debian:8:8.0.201701180	8.0.201701180
Debian	credativ	8	credativ:Debian:8:8.0.201703150	8.0.201703150
Debian	credativ	8	credativ:Debian:8:8.0.201704110	8.0.201704110
Debian	credativ	8	credativ:Debian:8:8.0.201704180	8.0.201704180
Debian	credativ	8	credativ:Debian:8:8.0.201706190	8.0.201706190
Debian	credativ	8	credativ:Debian:8:8.0.201706210	8.0.201706210
Debian	credativ	8	credativ:Debian:8:8.0.201708040	8.0.201708040
Debian	credativ	8	credativ:Debian:8:8.0.201710090	8.0.201710090
Debian	credativ	8	credativ:Debian:8:8.0.201712040	8.0.201712040
Debian	credativ	8	credativ:Debian:8:8.0.201801170	8.0.201801170
Debian	credativ	8	credativ:Debian:8:8.0.201803130	8.0.201803130
Debian	credativ	8	credativ:Debian:8:8.0.201803260	8.0.201803260
Debian	credativ	8	credativ:Debian:8:8.0.201804020	8.0.201804020
Debian	credativ	8	credativ:Debian:8:8.0.201804150	8.0.201804150
Debian	credativ	8	credativ:Debian:8:8.0.201805160	8.0.201805160
Debian	credativ	8	credativ:Debian:8:8.0.201807160	8.0.201807160
Debian	credativ	8	credativ:Debian:8:8.0.201901221	8.0.201901221
...				

Navigate the images

Another way to find an image in a location is to run the [az vm image list-publishers](#), [az vm image list-offers](#), and [az vm image list-skus](#) commands in sequence. With these commands, you determine these values:

1. List the image publishers.
2. For a given publisher, list their offers.
3. For a given offer, list their SKUs.

Then, for a selected SKU, you can choose a version to deploy.

For example, the following command lists the image publishers in the West US location:

```
az vm image list-publishers --location westus --output table
```

Partial output:

Location	Name
westus	128technology
westus	1e
westus	4psa
westus	5nine-software-inc
westus	7isolutions
westus	a10networks
westus	abiquo
westus	acellion
westus	accessdata-group
westus	accops
westus	Acronis
westus	Acronis.Backup
westus	actian-corp
westus	actian_matrix
westus	actifio
westus	activeeon
westus	advantech-webaccess
westus	aerospike
westus	affinio
westus	aiscaler-cache-control-ddos-and-url-rewriting-
westus	akamai-technologies
westus	akumina
...	

Use this information to find offers from a specific publisher. For example, for the *Canonical* publisher in the West US location, find offers by running `az vm image list-offers`. Pass the location and the publisher as in the following example:

```
az vm image list-offers --location westus --publisher Canonical --output table
```

Output:

Location	Name
westus	Ubuntu15.04Snappy
westus	Ubuntu15.04SnappyDocker
westus	UbunturollingSnappy
westus	UbuntuServer
westus	Ubuntu_Core

You see that in the West US region, Canonical publishes the *UbuntuServer* offer on Azure. But what SKUs? To get those values, run `az vm image list-skus` and set the location, publisher, and offer that you discovered:

```
az vm image list-skus --location westus --publisher Canonical --offer UbuntuServer --output table
```

Output:

Location	Name
westus	12.04.3-LTS
westus	12.04.4-LTS
westus	12.04.5-LTS
westus	14.04.0-LTS
westus	14.04.1-LTS
westus	14.04.2-LTS
westus	14.04.3-LTS
westus	14.04.4-LTS
westus	14.04.5-DAILY-LTS
westus	14.04.5-LTS
westus	16.04-DAILY-LTS
westus	16.04-LTS
westus	16.04.0-LTS
westus	18.04-DAILY-LTS
westus	18.04-LTS
westus	18.10
westus	18.10-DAILY
westus	19.04-DAILY

Finally, use the `az vm image list` command to find a specific version of the SKU you want, for example, `18.04-LTS`:

```
az vm image list --location westus --publisher Canonical --offer UbuntuServer --sku 18.04-LTS --all --output table
```

Partial output:

Offer	Publisher	Sku	Urn	Version
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201804262	18.04.201804262
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201805170	18.04.201805170
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201805220	18.04.201805220
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201806130	18.04.201806130
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201806170	18.04.201806170
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201807240	18.04.201807240
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201808060	18.04.201808060
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201808080	18.04.201808080
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201808140	18.04.201808140
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201808310	18.04.201808310
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201809110	18.04.201809110
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201810030	18.04.201810030
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201810240	18.04.201810240
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201810290	18.04.201810290
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201811010	18.04.201811010
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201812031	18.04.201812031
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201812040	18.04.201812040
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201812060	18.04.201812060
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201901140	18.04.201901140
UbuntuServer	Canonical	18.04-LTS	Canonical:UbuntuServer:18.04-LTS:18.04.201901220	18.04.201901220
...				

Now you can choose precisely the image you want to use by taking note of the URN value. Pass this value with the `--image` parameter when you create a VM with the `az vm create` command. Remember that you can optionally replace the version number in the URN with "latest". This version is always the latest version of the image.

If you deploy a VM with a Resource Manager template, you set the image parameters individually in the `imageReference` properties. See the [template reference](#).

Deploy an image with Marketplace terms

Some VM images in the Azure Marketplace have additional license and purchase terms that you must accept before you can deploy them programmatically.

To deploy a VM from such an image, you'll need to both accept the image's terms and enable programmatic deployment. You'll only need to do this once per subscription. Afterward, each time you deploy a VM programmatically from the image you'll also need to specify *purchase plan* parameters.

The following sections show how to:

- Find out whether a Marketplace image has additional license terms
- Accept the terms programmatically
- Provide purchase plan parameters when you deploy a VM programmatically

View plan properties

To view an image's purchase plan information, run the `az vm image show` command. If the `plan` property in the output is not `null`, the image has terms you need to accept before programmatic deployment.

For example, the Canonical Ubuntu Server 18.04 LTS image doesn't have additional terms, because the `plan` information is `null`:

```
az vm image show --location westus --urn Canonical:UbuntuServer:18.04-LTS:latest
```

Output:

```
{
  "dataDiskImages": [],
  "id": "/Subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/Providers/Microsoft.Compute/Locations/westus/Publishers/Canonical/ArtifactTypes/VMImage/Offers/Ub
untuServer/Skus/18.04-LTS/Versions/18.04.201901220",
  "location": "westus",
  "name": "18.04.201901220",
  "osDiskImage": {
    "operatingSystem": "Linux"
  },
  "plan": null,
  "tags": null
}
```

Running a similar command for the RabbitMQ Certified by Bitnami image shows the following `plan` properties: `name`, `product`, and `publisher`. (Some images also have a `promotion code` property.) To deploy this image, see the following sections to accept the terms and enable programmatic deployment.

```
az vm image show --location westus --urn bitnami:rabbitmq:rabbitmq:latest
```

Output:

```
{
  "dataDiskImages": [],
  "id": "/Subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/Providers/Microsoft.Compute/Locations/westus/Publishers/bitnami/ArtifactTypes/VMImage/Offers/rabb
itmq/Skus/rabbitmq/Versions/3.7.1901151016",
  "location": "westus",
  "name": "3.7.1901151016",
  "osDiskImage": {
    "operatingSystem": "Linux"
  },
  "plan": {
    "name": "rabbitmq",
    "product": "rabbitmq",
    "publisher": "bitnami"
  },
  "tags": null
}
```

Accept the terms

To view and accept the license terms, use the [az vm image accept-terms](#) command. When you accept the terms, you enable programmatic deployment in your subscription. You only need to accept terms once per subscription for the image. For example:

```
az vm image accept-terms --urn bitnami:rabbitmq:rabbitmq:latest
```

The output includes a `licenseTextLink` to the license terms, and indicates that the value of `accepted` is `true`:

```
{
  "accepted": true,
  "additionalProperties": {},
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/providers/Microsoft.MarketplaceOrdering/offertypes/bitnami/offers/rabbitmq/plans/rabbitmq",
  "licenseTextLink":
  "https://storelegalterms.blob.core.windows.net/legalterms/3E5ED_legalterms_BITNAMI%253a24RABBITMQ%253a24RABBIT
MQ%253a24IGRT7HHPIFOBV3IQYJHEN202FGUVXXZ3WUYIMEIVF3KCUNJ7GTVXNNM23I567GBMNDRFOY4WXJPN5PUYXNKB2QLAKCHP4IE5GO3B
2I.txt",
  "name": "rabbitmq",
  "plan": "rabbitmq",
  "privacyPolicyLink": "https://bitnami.com/privacy",
  "product": "rabbitmq",
  "publisher": "bitnami",
  "retrieveDatetime": "2019-01-25T20:37:49.937096Z",
  "signature":
  "XXXXXXLAZIK7ZL2YRV5JYQXONPV76NQJW3FKMDZYCRGXZYVDGX6BVY45J03BXVMNA2COBOEYG2N0760NORU7ITTRHGZDYNJXXXXXX",
  "type": "Microsoft.MarketplaceOrdering/offertypes"
}
```

Deploy using purchase plan parameters

After accepting the terms for the image, you can deploy a VM in the subscription. To deploy the image by using the [az vm create](#) command, provide parameters for the purchase plan in addition to a URN for the image. For example, to deploy a VM with the RabbitMQ Certified by Bitnami image:

```
az group create --name myResourceGroupVM --location westus

az vm create --resource-group myResourceGroupVM --name myVM --image bitnami:rabbitmq:rabbitmq:latest --plan-
name rabbitmq --plan-product rabbitmq --plan-publisher bitnami
```

Next steps

To create a virtual machine quickly by using the image information, see [Create and Manage Linux VMs with the Azure CLI](#).

Information for Non-endorsed Distributions

4/25/2019 • 9 minutes to read • [Edit Online](#)

NOTE

Azure has two different deployment models for creating and working with resources: [Resource Manager](#) and [classic](#). This article covers using both models, but Microsoft recommends that most new deployments use the Resource Manager model.

The Azure platform SLA applies to virtual machines running the Linux OS only when one of the [endorsed distributions](#) is used. For these endorsed distributions, pre-configured Linux images are provided in the Azure Marketplace.

- [Linux on Azure - Endorsed Distributions](#)
- [Support for Linux images in Microsoft Azure](#)

All distributions running on Azure have a number of prerequisites. This article can't be comprehensive, as every distribution is different. Even if you meet all the criteria below, you may need to significantly tweak your Linux system for it to run properly.

We recommend that you start with one of the [Linux on Azure Endorsed Distributions](#). The following articles show you how to prepare the various endorsed Linux distributions that are supported on Azure:

- [CentOS-based Distributions](#)
- [Debian Linux](#)
- [Oracle Linux](#)
- [Red Hat Enterprise Linux](#)
- [SLES & openSUSE](#)
- [Ubuntu](#)

This article focuses on general guidance for running your Linux distribution on Azure.

General Linux Installation Notes

- The Hyper-V virtual hard disk (VHDX) format isn't supported in Azure, only *fixed VHD*. You can convert the disk to VHD format using Hyper-V Manager or the [Convert-VHD](#) cmdlet. If you're using VirtualBox, select **Fixed size** rather than the default (dynamically allocated) when creating the disk.
- Azure only supports generation 1 virtual machines. You can convert a generation 1 virtual machine from VHDX to the VHD file format, and from dynamically expanding to a fixed sized disk. You can't change a virtual machine's generation. For more information, see [Should I create a generation 1 or 2 virtual machine in Hyper-V?](#)
- The maximum size allowed for the VHD is 1,023 GB.
- When installing the Linux system we recommend that you use standard partitions, rather than Logical Volume Manager (LVM) which is the default for many installations. Using standard partitions will avoid LVM name conflicts with cloned VMs, particularly if an OS disk is ever attached to another identical VM for troubleshooting. [LVM](#) or [RAID](#) may be used on data disks.
- Kernel support for mounting UDF file systems is necessary. At first boot on Azure the provisioning configuration is passed to the Linux VM by using UDF-formatted media that is attached to the guest. The Azure Linux agent must mount the UDF file system to read its configuration and provision the VM.
- Linux kernel versions earlier than 2.6.37 don't support NUMA on Hyper-V with larger VM sizes. This issue

primarily impacts older distributions using the upstream Red Hat 2.6.32 kernel, and was fixed in Red Hat Enterprise Linux (RHEL) 6.6 (kernel-2.6.32-504). Systems running custom kernels older than 2.6.37, or RHEL-based kernels older than 2.6.32-504 must set the boot parameter `numa=off` on the kernel command line in `grub.conf`. For more information, see [Red Hat KB 436883](#).

- Don't configure a swap partition on the OS disk. The Linux agent can be configured to create a swap file on the temporary resource disk, as described in the following steps.
- All VHDs on Azure must have a virtual size aligned to 1 MB. When converting from a raw disk to VHD you must ensure that the raw disk size is a multiple of 1 MB before conversion, as described in the following steps.

Installing kernel modules without Hyper-V

Azure runs on the Hyper-V hypervisor, so Linux requires certain kernel modules to run in Azure. If you have a VM that was created outside of Hyper-V, the Linux installers may not include the drivers for Hyper-V in the initial ramdisk (`initrd` or `initramfs`), unless the VM detects that it's running on a Hyper-V environment. When using a different virtualization system (such as Virtualbox, KVM, and so on) to prepare your Linux image, you may need to rebuild the `initrd` so that at least the `hv_vmbus` and `hv_storvsc` kernel modules are available on the initial ramdisk. This known issue is for systems based on the upstream Red Hat distribution, and possibly others.

The mechanism for rebuilding the `initrd` or `initramfs` image may vary depending on the distribution. Consult your distribution's documentation or support for the proper procedure. Here is one example for rebuilding the `initrd` by using the `mkinitrd` utility:

1. Back up the existing `initrd` image:

```
cd /boot  
sudo cp initrd-`uname -r`.img initrd-`uname -r`.img.bak
```

2. Rebuild the `initrd` with the `hv_vmbus` and `hv_storvsc` kernel modules:

```
sudo mkinitrd --preload=hv_storvsc --preload=hv_vmbus -v -f initrd-`uname -r`.img `uname -r`
```

Resizing VHDs

VHD images on Azure must have a virtual size aligned to 1 MB. Typically, VHDs created using Hyper-V are aligned correctly. If the VHD isn't aligned correctly, you may receive an error message similar to the following when you try to create an image from your VHD.

- The VHD `http://<mystorageaccount>.blob.core.windows.net/vhds/MyLinuxVM.vhd` has an unsupported virtual size of 21475270656 bytes. The size must be a whole number (in MBs).

In this case, resize the VM using either the Hyper-V Manager console or the [Resize-VHD](#) PowerShell cmdlet. If you aren't running in a Windows environment, we recommend using `qemu-img` to convert (if needed) and resize the VHD.

NOTE

There is a [known bug in qemu-img](#) versions $\geq 2.2.1$ that results in an improperly formatted VHD. The issue has been fixed in QEMU 2.6. We recommend using either `qemu-img` 2.2.0 or lower, or 2.6 or higher.

1. Resizing the VHD directly using tools such as `qemu-img` or `vbox-manage` may result in an unbootable VHD. We recommend first converting the VHD to a RAW disk image. If the VM image was created as a RAW disk image (the default for some hypervisors such as KVM), then you may skip this step.

```
qemu-img convert -f vpc -O raw MyLinuxVM.vhd MyLinuxVM.raw
```

2. Calculate the required size of the disk image so that the virtual size is aligned to 1 MB. The following bash shell script uses `qemu-img info` to determine the virtual size of the disk image, and then calculates the size to the next 1 MB.

```
rawdisk="MyLinuxVM.raw"
vhddisk="MyLinuxVM.vhd"

MB=$((1024*1024))
size=$(qemu-img info -f raw --output json "$rawdisk" | \
gawk 'match($0, /"virtual-size": ([0-9]+),/, val) {print val[1]}')

rounded_size=$(((size/$MB + 1)*$MB))

echo "Rounded Size = $rounded_size"
```

3. Resize the raw disk using `$rounded_size` as set above.

```
qemu-img resize MyLinuxVM.raw $rounded_size
```

4. Now, convert the RAW disk back to a fixed-size VHD.

```
qemu-img convert -f raw -o subformat=fixed -O vpc MyLinuxVM.raw MyLinuxVM.vhd
```

Or, with qemu version 2.6+, include the `force_size` option.

```
qemu-img convert -f raw -o subformat=fixed,force_size -O vpc MyLinuxVM.raw MyLinuxVM.vhd
```

Linux Kernel Requirements

The Linux Integration Services (LIS) drivers for Hyper-V and Azure are contributed directly to the upstream Linux kernel. Many distributions that include a recent Linux kernel version (such as 3.x) have these drivers available already, or otherwise provide backported versions of these drivers with their kernels. These drivers are constantly being updated in the upstream kernel with new fixes and features, so when possible we recommend running an [endorsed distribution](#) that includes these fixes and updates.

If you're running a variant of Red Hat Enterprise Linux versions 6.0 to 6.3, then you'll need to install the[latest LIS drivers for Hyper-V](#). Beginning with RHEL 6.4+ (and derivatives) the LIS drivers are already included with the kernel and so no additional installation packages are needed.

If a custom kernel is required, we recommend a recent kernel version (such as 3.8+). For distributions or vendors who maintain their own kernel, you'll need to regularly backport the LIS drivers from the upstream kernel to your custom kernel. Even if you're already running a relatively recent kernel version, we highly recommend keeping track of any upstream fixes in the LIS drivers and backport them as needed. The locations of the LIS driver source files are specified in the [MAINTAINERS](#) file in the Linux kernel source tree:

```
F: arch/x86/include/asm/mshyperv.h
F: arch/x86/include/uapi/asm/hyperv.h
F: arch/x86/kernel/cpu/mshyperv.c
F: drivers/hid/hid-hyperv.c
F: drivers/hv/
F: drivers/input/serio/hyperv-keyboard.c
F: drivers/net/hyperv/
F: drivers/scsi/storvsc_drv.c
F: drivers/video/fbdev/hyperv_fb.c
F: include/linux/hyperv.h
F: tools/hv/
```

The following patches must be included in the kernel. This list can't be complete for all distributions.

- [ata_piix: defer disks to the Hyper-V drivers by default](#)
- [storvsc: Account for in-transit packets in the RESET path](#)
- [storvsc: avoid usage of WRITE_SAME](#)
- [storvsc: Disable WRITE SAME for RAID and virtual host adapter drivers](#)
- [storvsc: NULL pointer dereference fix](#)
- [storvsc: ring buffer failures may result in I/O freeze](#)
- [scsi_sysfs: protect against double execution of __scsi_remove_device](#)

The Azure Linux Agent

The [Azure Linux Agent](#) `waagent` provisions a Linux virtual machine in Azure. You can get the latest version, file issues, or submit pull requests at the [Linux Agent GitHub repo](#).

- The Linux agent is released under the Apache 2.0 license. Many distributions already provide RPM or deb packages for the agent, and these packages can easily be installed and updated.
- The Azure Linux Agent requires Python v2.6+.
- The agent also requires the python-pyasn1 module. Most distributions provide this module as a separate package to be installed.
- In some cases, the Azure Linux Agent may not be compatible with NetworkManager. Many of the RPM/Deb packages provided by distributions configure NetworkManager as a conflict to the waagent package. In these cases, it will uninstall NetworkManager when you install the Linux agent package.
- The Azure Linux Agent must be at or above the [minimum supported version](#).

General Linux System Requirements

1. Modify the kernel boot line in GRUB or GRUB2 to include the following parameters, so that all console messages are sent to the first serial port. These messages can assist Azure support with debugging any issues.

```
console=ttyS0,115200n8 earlyprintk=ttyS0,115200 rootdelay=300
```

We also recommend *removing* the following parameters if they exist.

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot isn't useful in a cloud environment, where we want all logs sent to the serial port. The `crashkernel` option may be left configured if needed, but note that this parameter reduces the amount of available memory in the VM by at least 128 MB, which may be problematic for smaller VM sizes.

2. Install the Azure Linux Agent.

The Azure Linux Agent is required for provisioning a Linux image on Azure. Many distributions provide the agent as an RPM or Deb package (the package is typically called WALinuxAgent or walinuxagent). The agent can also be installed manually by following the steps in the [Linux Agent Guide](#).

3. Ensure that the SSH server is installed, and configured to start at boot time. This configuration is usually the default.
4. Don't create swap space on the OS disk.

The Azure Linux Agent can automatically configure swap space using the local resource disk that is attached to the VM after provisioning on Azure. The local resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned. After installing the Azure Linux Agent (step 2 above), modify the following parameters in /etc/waagent.conf as needed.

```
ResourceDisk.Format=y
ResourceDisk.Filesystem=ext4
ResourceDisk.MountPoint=/mnt/resource
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048    ## NOTE: Set this to your desired size.
```

5. Run the following commands to deprovision the virtual machine.

```
sudo waagent -force -deprovision
export HISTSIZE=0
logout
```

NOTE

On Virtualbox you may see the following error after running `waagent -force -deprovision` that says `[Errno 5] Input/output error`. This error message is not critical and can be ignored.

- Shut down the virtual machine and upload the VHD to Azure.

Prepare an Ubuntu virtual machine for Azure

6/25/2019 • 3 minutes to read • [Edit Online](#)

NOTE

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using both models, but Microsoft recommends that most new deployments use the Resource Manager model.

Official Ubuntu cloud images

Ubuntu now publishes official Azure VHDs for download at <https://cloud-images.ubuntu.com/>. If you need to build your own specialized Ubuntu image for Azure, rather than use the manual procedure below it is recommended to start with these known working VHDs and customize as needed. The latest image releases can always be found at the following locations:

- Ubuntu 12.04/Precise: [ubuntu-12.04-server-cloudimg-amd64-disk1.vhd.zip](#)
- Ubuntu 14.04/Trusty: [ubuntu-14.04-server-cloudimg-amd64-disk1.vhd.zip](#)
- Ubuntu 16.04/Xenial: [ubuntu-16.04-server-cloudimg-amd64-disk1.vhd.zip](#)
- Ubuntu 18.04/Bionic: [bionic-server-cloudimg-amd64.vhd.zip](#)
- Ubuntu 18.10/Cosmic: [cosmic-server-cloudimg-amd64.vhd.zip](#)

Prerequisites

This article assumes that you have already installed an Ubuntu Linux operating system to a virtual hard disk. Multiple tools exist to create .vhd files, for example a virtualization solution such as Hyper-V. For instructions, see [Install the Hyper-V Role and Configure a Virtual Machine](#).

Ubuntu installation notes

- Please see also [General Linux Installation Notes](#) for more tips on preparing Linux for Azure.
- The VHDX format is not supported in Azure, only **fixed VHD**. You can convert the disk to VHD format using Hyper-V Manager or the convert-vhd cmdlet.
- When installing the Linux system it is recommended that you use standard partitions rather than LVM (often the default for many installations). This will avoid LVM name conflicts with cloned VMs, particularly if an OS disk ever needs to be attached to another VM for troubleshooting. [LVM](#) or [RAID](#) may be used on data disks if preferred.
- Do not configure a swap partition on the OS disk. The Linux agent can be configured to create a swap file on the temporary resource disk. More information about this can be found in the steps below.
- All VHDs on Azure must have a virtual size aligned to 1MB. When converting from a raw disk to VHD you must ensure that the raw disk size is a multiple of 1MB before conversion. See [Linux Installation Notes](#) for more information.

Manual steps

NOTE

Before attempting to create your own custom Ubuntu image for Azure, please consider using the pre-built and tested images from <https://cloud-images.ubuntu.com/> instead.

1. In the center pane of Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open the window for the virtual machine.
3. Replace the current repositories in the image to use Ubuntu's Azure repos. The steps vary slightly depending on the Ubuntu version.

Before editing `/etc/apt/sources.list`, it is recommended to make a backup:

```
# sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
```

Ubuntu 12.04:

```
# sudo sed -i 's/[a-z][a-z].archive.ubuntu.com/azure.archive.ubuntu.com/g' /etc/apt/sources.list  
# sudo apt-get update
```

Ubuntu 14.04:

```
# sudo sed -i 's/[a-z][a-z].archive.ubuntu.com/azure.archive.ubuntu.com/g' /etc/apt/sources.list  
# sudo apt-get update
```

Ubuntu 16.04:

```
# sudo sed -i 's/[a-z][a-z].archive.ubuntu.com/azure.archive.ubuntu.com/g' /etc/apt/sources.list  
# sudo apt-get update
```

4. The Ubuntu Azure images are now following the *hardware enablement* (HWE) kernel. Update the operating system to the latest kernel by running the following commands:

Ubuntu 12.04:

```
# sudo apt-get update  
# sudo apt-get install linux-image-generic-lts-trusty linux-cloud-tools-generic-lts-trusty  
# sudo apt-get install hv-kvp-daemon-init  
(recommended) sudo apt-get dist-upgrade  
  
# sudo reboot
```

Ubuntu 14.04:

```
# sudo apt-get update  
# sudo apt-get install linux-image-virtual-lts-vivid linux-lts-vivid-tools-common  
# sudo apt-get install hv-kvp-daemon-init  
(recommended) sudo apt-get dist-upgrade  
  
# sudo reboot
```

Ubuntu 16.04:

```
# sudo apt-get update  
# sudo apt-get install linux-generic-hwe-16.04 linux-cloud-tools-generic-hwe-16.04  
(recommended) sudo apt-get dist-upgrade  
  
# sudo reboot
```

See also:

- <https://wiki.ubuntu.com/Kernel/LTSEnablementStack>
- <https://wiki.ubuntu.com/Kernel/RollingLTSEnablementStack>

5. Modify the kernel boot line for Grub to include additional kernel parameters for Azure. To do this open `/etc/default/grub` in a text editor, find the variable called `GRUB_CMDLINE_LINUX_DEFAULT` (or add it if needed) and edit it to include the following parameters:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty1 console=ttyS0,115200n8 earlyprintk=ttyS0,115200 rootdelay=300"
```

Save and close this file, and then run `sudo update-grub`. This will ensure all console messages are sent to the first serial port, which can assist Azure technical support with debugging issues.

6. Ensure that the SSH server is installed and configured to start at boot time. This is usually the default.

7. Install the Azure Linux Agent:

```
# sudo apt-get update  
# sudo apt-get install walinuxagent
```

NOTE

The `walinuxagent` package may remove the `NetworkManager` and `NetworkManager-gnome` packages, if they are installed.

8. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# sudo waagent -force -deprovision  
# export HISTSIZE=0  
# logout
```

9. Click **Action -> Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

References

[Ubuntu hardware enablement \(HWE\) kernel](#)

Next steps

You're now ready to use your Ubuntu Linux virtual hard disk to create new virtual machines in Azure. If this is the first time that you're uploading the .vhd file to Azure, see [Create a Linux VM from a custom disk](#).

Prepare a CentOS-based virtual machine for Azure

5/24/2019 • 9 minutes to read • [Edit Online](#)

Learn to create and upload an Azure virtual hard disk (VHD) that contains a CentOS-based Linux operating system.

- [Prepare a CentOS 6.x virtual machine for Azure](#)
- [Prepare a CentOS 7.0+ virtual machine for Azure](#)

Prerequisites

This article assumes that you have already installed a CentOS (or similar derivative) Linux operating system to a virtual hard disk. Multiple tools exist to create .vhd files, for example a virtualization solution such as Hyper-V. For instructions, see [Install the Hyper-V Role and Configure a Virtual Machine](#).

CentOS installation notes

- Please see also [General Linux Installation Notes](#) for more tips on preparing Linux for Azure.
- The VHDX format is not supported in Azure, only **fixed VHD**. You can convert the disk to VHD format using Hyper-V Manager or the convert-vhd cmdlet. If you are using VirtualBox this means selecting **Fixed size** as opposed to the default dynamically allocated when creating the disk.
- When installing the Linux system it is *recommended* that you use standard partitions rather than LVM (often the default for many installations). This will avoid LVM name conflicts with cloned VMs, particularly if an OS disk ever needs to be attached to another identical VM for troubleshooting. [LVM](#) or [RAID](#) may be used on data disks.
- Kernel support for mounting UDF file systems is required. At first boot on Azure the provisioning configuration is passed to the Linux VM via UDF-formatted media that is attached to the guest. The Azure Linux agent must be able to mount the UDF file system to read its configuration and provision the VM.
- Linux kernel versions below 2.6.37 do not support NUMA on Hyper-V with larger VM sizes. This issue primarily impacts older distributions using the upstream Red Hat 2.6.32 kernel, and was fixed in RHEL 6.6 (kernel-2.6.32-504). Systems running custom kernels older than 2.6.37, or RHEL-based kernels older than 2.6.32-504 must set the boot parameter `numa=off` on the kernel command-line in grub.conf. For more information see Red Hat [KB 436883](#).
- Do not configure a swap partition on the OS disk. The Linux agent can be configured to create a swap file on the temporary resource disk. More information about this can be found in the steps below.
- All VHDs on Azure must have a virtual size aligned to 1MB. When converting from a raw disk to VHD you must ensure that the raw disk size is a multiple of 1MB before conversion. See [Linux Installation Notes](#) for more information.

CentOS 6.x

1. In Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open a console window for the virtual machine.
3. In CentOS 6, NetworkManager can interfere with the Azure Linux agent. Uninstall this package by running the following command:

```
sudo rpm -e --nodeps NetworkManager
```

4. Create or edit the file `/etc/sysconfig/network` and add the following text:

```
NETWORKING=yes  
HOSTNAME=localhost.localdomain
```

5. Create or edit the file `/etc/sysconfig/network-scripts/ifcfg-eth0` and add the following text:

```
DEVICE=eth0  
ONBOOT=yes  
BOOTPROTO=dhcp  
TYPE=Ethernet  
USERCTL=no  
PEERDNS=yes  
IPV6INIT=no
```

6. Modify udev rules to avoid generating static rules for the Ethernet interface(s). These rules can cause problems when cloning a virtual machine in Microsoft Azure or Hyper-V:

```
sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules  
sudo rm -f /etc/udev/rules.d/70-persistent-net.rules
```

7. Ensure the network service will start at boot time by running the following command:

```
sudo chkconfig network on
```

8. If you would like to use the OpenLogic mirrors that are hosted within the Azure datacenters, then replace the `/etc/yum.repos.d/CentOS-Base.repo` file with the following repositories. This will also add the **[openlogic]** repository that includes additional packages such as the Azure Linux agent:

```

[openlogic]
name=CentOS-$releasever - openlogic packages for $basearch
baseurl=http://olcentgbl.trafficmanager.net/openlogic/$releasever/openlogic/$basearch/
enabled=1
gpgcheck=0

[base]
name=CentOS-$releasever - Base
#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

#released updates
[updates]
name=CentOS-$releasever - Updates
#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=updates&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

#additional packages that may be useful
[extras]
name=CentOS-$releasever - Extras
#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=extras&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/extras/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

#additional packages that extend functionality of existing packages
[centosplus]
name=CentOS-$releasever - Plus
#mirrorlist=http://mirrorlist.centos.org/?
release=$releasever&arch=$basearch&repo=centosplus&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/centosplus/$basearch/
gpgcheck=1
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

#contrib - packages by Centos Users
[contrib]
name=CentOS-$releasever - Contrib
#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=contrib&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/contrib/$basearch/
gpgcheck=1
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

```

NOTE

The rest of this guide will assume you are using at least the `[openlogic]` repo, which will be used to install the Azure Linux agent below.

- Add the following line to `/etc/yum.conf`:

```
http_caching=packages
```

- Run the following command to clear the current yum metadata and update the system with the latest packages:

```
yum clean all
```

Unless you are creating an image for an older version of CentOS, it is recommended to update all the packages to the latest:

```
sudo yum -y update
```

A reboot may be required after running this command.

11. (Optional) Install the drivers for the Linux Integration Services (LIS).

IMPORTANT

The step is **required** for CentOS 6.3 and earlier, and optional for later releases.

```
sudo rpm -e hypervkvpd ## (may return error if not installed, that's OK)
sudo yum install microsoft-hyper-v
```

Alternatively, you can follow the manual installation instructions on the [LIS download page](#) to install the RPM onto your VM.

12. Install the Azure Linux Agent and dependencies:

```
sudo yum install python-pyasn1 WALinuxAgent
```

The WALinuxAgent package will remove the NetworkManager and NetworkManager-gnome packages if they were not already removed as described in step 3.

13. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this, open `/boot/grub/menu.lst` in a text editor and ensure that the default kernel includes the following parameters:

```
console=ttyS0 earlyprintk=ttyS0 rootdelay=300
```

This will also ensure all console messages are sent to the first serial port, which can assist Azure support with debugging issues.

In addition to the above, it is recommended to *remove* the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. The `crashkernel` option may be left configured if desired, but note that this parameter will reduce the amount of available memory in the VM by 128MB or more, which may be problematic on the smaller VM sizes.

IMPORTANT

CentOS 6.5 and earlier must also set the kernel parameter `numa=off`. See Red Hat [KB 436883](#).

14. Ensure that the SSH server is installed and configured to start at boot time. This is usually the default.

15. Do not create swap space on the OS disk.

The Azure Linux Agent can automatically configure swap space using the local resource disk that is attached to the VM after provisioning on Azure. Note that the local resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned. After installing the Azure Linux Agent (see previous step), modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y
ResourceDisk.Filesystem=ext4
ResourceDisk.MountPoint=/mnt/resource
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048 ## NOTE: set this to whatever you need it to be.
```

16. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
sudo waagent -force -deprovision
export HISTSIZE=0
logout
```

17. Click **Action -> Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

CentOS 7.0+

Changes in CentOS 7 (and similar derivatives)

Preparing a CentOS 7 virtual machine for Azure is very similar to CentOS 6, however there are several important differences worth noting:

- The NetworkManager package no longer conflicts with the Azure Linux agent. This package is installed by default and we recommend that it is not removed.
- GRUB2 is now used as the default bootloader, so the procedure for editing kernel parameters has changed (see below).
- XFS is now the default file system. The ext4 file system can still be used if desired.

Configuration Steps

1. In Hyper-V Manager, select the virtual machine.

2. Click **Connect** to open a console window for the virtual machine.

3. Create or edit the file `/etc/sysconfig/network` and add the following text:

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
```

4. Create or edit the file `/etc/sysconfig/network-scripts/ifcfg-eth0` and add the following text:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
NM_CONTROLLED=no
```

5. Modify udev rules to avoid generating static rules for the Ethernet interface(s). These rules can cause problems when cloning a virtual machine in Microsoft Azure or Hyper-V:

```
sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules
```

6. If you would like to use the OpenLogic mirrors that are hosted within the Azure datacenters, then replace the `/etc/yum.repos.d/CentOS-Base.repo` file with the following repositories. This will also add the **[openlogic]** repository that includes packages for the Azure Linux agent:

```
[openlogic]
name=CentOS-$releasever - openlogic packages for $basearch
baseurl=http://olcentgbl.trafficmanager.net/openlogic/$releasever/openlogic/$basearch/
enabled=1
gpgcheck=0

[base]
name=CentOS-$releasever - Base
#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

#released updates
[updates]
name=CentOS-$releasever - Updates
#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=updates&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

#additional packages that may be useful
[extras]
name=CentOS-$releasever - Extras
#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=extras&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/extras/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

#additional packages that extend functionality of existing packages
[centosplus]
name=CentOS-$releasever - Plus
#mirrorlist=http://mirrorlist.centos.org/?
release=$releasever&arch=$basearch&repo=centosplus&infra=$infra
baseurl=http://olcentgbl.trafficmanager.net/centos/$releasever/centosplus/$basearch/
gpgcheck=1
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

NOTE

The rest of this guide will assume you are using at least the `[openlogic]` repo, which will be used to install the Azure Linux agent below.

7. Run the following command to clear the current yum metadata and install any updates:

```
sudo yum clean all
```

Unless you are creating an image for an older version of CentOS, it is recommended to update all the packages to the latest:

```
sudo yum -y update
```

A reboot maybe required after running this command.

8. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this, open `/etc/default/grub` in a text editor and edit the `GRUB_CMDLINE_LINUX` parameter, for example:

```
GRUB_CMDLINE_LINUX="rootdelay=300 console=ttyS0 earlyprintk=ttyS0 net.ifnames=0"
```

This will also ensure all console messages are sent to the first serial port, which can assist Azure support with debugging issues. It also turns off the new CentOS 7 naming conventions for NICs. In addition to the above, it is recommended to *remove* the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. The `crashkernel` option may be left configured if desired, but note that this parameter will reduce the amount of available memory in the VM by 128MB or more, which may be problematic on the smaller VM sizes.

9. Once you are done editing `/etc/default/grub` per above, run the following command to rebuild the grub configuration:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

10. If building the image from **VMware, VirtualBox or KVM**: Ensure the Hyper-V drivers are included in the initramfs:

Edit `/etc/dracut.conf`, add content:

```
add_drivers+="hv_vmbus hv_netvsc hv_storvsc"
```

Rebuild the initramfs:

```
sudo dracut -f -v
```

11. Install the Azure Linux Agent and dependencies:

```
sudo yum install python-pyasn1 WALinuxAgent  
sudo systemctl enable waagent
```

12. Do not create swap space on the OS disk.

The Azure Linux Agent can automatically configure swap space using the local resource disk that is attached to the VM after provisioning on Azure. Note that the local resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned. After installing the Azure Linux Agent (see previous step), modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y  
ResourceDisk.Filesystem=ext4  
ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=y  
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

13. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
sudo waagent -force -deprovision  
export HISTSIZE=0  
logout
```

14. Click **Action -> Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Next steps

You're now ready to use your CentOS Linux virtual hard disk to create new virtual machines in Azure. If this is the first time that you're uploading the .vhd file to Azure, see [Create a Linux VM from a custom disk](#).

Prepare a Red Hat-based virtual machine for Azure

5/21/2019 • 26 minutes to read • [Edit Online](#)

In this article, you will learn how to prepare a Red Hat Enterprise Linux (RHEL) virtual machine for use in Azure. The versions of RHEL that are covered in this article are 6.7+ and 7.1+. The hypervisors for preparation that are covered in this article are Hyper-V, kernel-based virtual machine (KVM), and VMware. For more information about eligibility requirements for participating in Red Hat's Cloud Access program, see [Red Hat's Cloud Access website](#) and [Running RHEL on Azure](#). For ways to automate building RHEL images see the [Azure Image Builder](#).

Prepare a Red Hat-based virtual machine from Hyper-V Manager

Prerequisites

This section assumes that you have already obtained an ISO file from the Red Hat website and installed the RHEL image to a virtual hard disk (VHD). For more details about how to use Hyper-V Manager to install an operating system image, see [Install the Hyper-V Role and Configure a Virtual Machine](#).

RHEL installation notes

- Azure does not support the VHDX format. Azure supports only fixed VHD. You can use Hyper-V Manager to convert the disk to VHD format, or you can use the convert-vhd cmdlet. If you use VirtualBox, select **Fixed size** as opposed to the default dynamically allocated option when you create the disk.
- Azure supports only generation 1 virtual machines. You can convert a generation 1 virtual machine from VHDX to the VHD file format and from dynamically expanding to a fixed-size disk. You can't change a virtual machine's generation. For more information, see [Should I create a generation 1 or 2 virtual machine in Hyper-V?](#).
- The maximum size that's allowed for the VHD is 1,023 GB.
- Logical Volume Manager (LVM) is supported and may be used on the OS disk or data disks in Azure virtual machines. However, in general it is recommended to use standard partitions on the OS disk rather than LVM. This practice will avoid LVM name conflicts with cloned virtual machines, particularly if you ever need to attach an operating system disk to another identical virtual machine for troubleshooting. See also [LVM](#) and [RAID](#) documentation.
- Kernel support for mounting Universal Disk Format (UDF) file systems is required. At first boot on Azure, the UDF-formatted media that is attached to the guest passes the provisioning configuration to the Linux virtual machine. The Azure Linux Agent must be able to mount the UDF file system to read its configuration and provision the virtual machine.
- Do not configure a swap partition on the operating system disk. The Linux Agent can be configured to create a swap file on the temporary resource disk. More information about this can be found in the following steps.
- All VHDs on Azure must have a virtual size aligned to 1MB. When converting from a raw disk to VHD you must ensure that the raw disk size is a multiple of 1MB before conversion. More details can be found in the steps below. See also [Linux Installation Notes](#) for more information.

Prepare a RHEL 6 virtual machine from Hyper-V Manager

1. In Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open a console window for the virtual machine.
3. In RHEL 6, NetworkManager can interfere with the Azure Linux agent. Uninstall this package by running the following command:

```
sudo yum remove NetworkManager
```

```
# sudo rpm -e --nodeps NetworkManager
```

4. Create or edit the `/etc/sysconfig/network` file, and add the following text:

```
NETWORKING=yes  
HOSTNAME=localhost.localdomain
```

5. Create or edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file, and add the following text:

```
DEVICE=eth0  
ONBOOT=yes  
BOOTPROTO=dhcp  
TYPE=Ethernet  
USERCTL=no  
PEERDNS=yes  
IPV6INIT=no
```

6. Move (or remove) the udev rules to avoid generating static rules for the Ethernet interface. These rules cause problems when you clone a virtual machine in Microsoft Azure or Hyper-V:

```
# sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules  
# sudo rm -f /etc/udev/rules.d/70-persistent-net.rules
```

7. Ensure that the network service will start at boot time by running the following command:

```
# sudo chkconfig network on
```

8. Register your Red Hat subscription to enable the installation of packages from the RHEL repository by running the following command:

```
# sudo subscription-manager register --auto-attach --username=XXX --password=XXX
```

9. The WALinuxAgent package, `WALinuxAgent-<version>`, has been pushed to the Red Hat extras repository. Enable the extras repository by running the following command:

```
# subscription-manager repos --enable=rhel-6-server-extras-rpms
```

10. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this modification, open `/boot/grub/menu.lst` in a text editor, and ensure that the default kernel includes the following parameters:

```
console=ttyS0 earlyprintk=ttyS0 rootdelay=300
```

This will also ensure that all console messages are sent to the first serial port, which can assist Azure support with debugging issues.

In addition, we recommended that you remove the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. You can leave the `crashkernel` option configured if desired. Note that this parameter reduces the amount of available memory in the virtual machine by 128 MB or more. This configuration might be problematic on smaller virtual machine sizes.

11. Ensure that the secure shell (SSH) server is installed and configured to start at boot time, which is usually the default. Modify `/etc/ssh/sshd_config` to include the following line:

```
ClientAliveInterval 180
```

12. Install the Azure Linux Agent by running the following command:

```
# sudo yum install WALinuxAgent  
  
# sudo chkconfig waagent on
```

Installing the WALinuxAgent package removes the NetworkManager and NetworkManager-gnome packages if they were not already removed in step 3.

13. Do not create swap space on the operating system disk.

The Azure Linux Agent can automatically configure swap space by using the local resource disk that is attached to the virtual machine after the virtual machine is provisioned on Azure. Note that the local resource disk is a temporary disk and that it might be emptied if the virtual machine is deprovisioned. After you install the Azure Linux Agent in the previous step, modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y  
ResourceDisk.Filesystem=ext4  
ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=y  
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

14. Unregister the subscription (if necessary) by running the following command:

```
# sudo subscription-manager unregister
```

15. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# Note: if you are migrating a specific virtual machine and do not wish to create a generalized image,  
# skip the deprovision step  
# sudo waagent -force -deprovision  
  
# export HISTSIZE=0  
  
# logout
```

16. Click **Action > Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Prepare a RHEL 7 virtual machine from Hyper-V Manager

1. In Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open a console window for the virtual machine.
3. Create or edit the `/etc/sysconfig/network` file, and add the following text:

```
NETWORKING=yes  
HOSTNAME=localhost.localdomain
```

4. Create or edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file, and add the following text:

```
DEVICE=eth0  
ONBOOT=yes  
BOOTPROTO=dhcp  
TYPE=Ethernet  
USERCTL=no  
PEERDNS=yes  
IPV6INIT=no  
NM_CONTROLLED=no
```

5. Ensure that the network service will start at boot time by running the following command:

```
# sudo systemctl enable network
```

6. Register your Red Hat subscription to enable the installation of packages from the RHEL repository by running the following command:

```
# sudo subscription-manager register --auto-attach --username=XXX --password=XXX
```

7. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this modification, open `/etc/default/grub` in a text editor, and edit the `GRUB_CMDLINE_LINUX` parameter. For example:

```
GRUB_CMDLINE_LINUX="rootdelay=300 console=ttyS0 earlyprintk=ttyS0 net.ifnames=0"
```

This will also ensure that all console messages are sent to the first serial port, which can assist Azure support with debugging issues. This configuration also turns off the new RHEL 7 naming conventions for NICs. In addition, we recommend that you remove the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. You can leave the `crashkernel` option configured if desired. Note that this parameter reduces the amount of available memory in the virtual machine by 128 MB or more, which might be problematic on smaller virtual machine sizes.

8. After you are done editing `/etc/default/grub`, run the following command to rebuild the grub configuration:

```
# sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

9. Ensure that the SSH server is installed and configured to start at boot time, which is usually the default. Modify `/etc/ssh/sshd_config` to include the following line:

```
ClientAliveInterval 180
```

10. The WALinuxAgent package, `WALinuxAgent-<version>`, has been pushed to the Red Hat extras repository.

Enable the extras repository by running the following command:

```
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

11. Install the Azure Linux Agent by running the following command:

```
# sudo yum install WALinuxAgent  
# sudo systemctl enable waagent.service
```

12. Do not create swap space on the operating system disk.

The Azure Linux Agent can automatically configure swap space by using the local resource disk that is attached to the virtual machine after the virtual machine is provisioned on Azure. Note that the local resource disk is a temporary disk, and it might be emptied if the virtual machine is deprovisioned. After you install the Azure Linux Agent in the previous step, modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y  
ResourceDisk.Filesystem=ext4  
ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=y  
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

13. If you want to unregister the subscription, run the following command:

```
# sudo subscription-manager unregister
```

14. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# Note: if you are migrating a specific virtual machine and do not wish to create a generalized image,  
# skip the deprovision step  
# sudo waagent -force -deprovision  
  
# export HISTSIZE=0  
  
# logout
```

15. Click **Action** > **Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Prepare a Red Hat-based virtual machine from KVM

Prepare a RHEL 6 virtual machine from KVM

1. Download the KVM image of RHEL 6 from the Red Hat website.

2. Set a root password.

Generate an encrypted password, and copy the output of the command:

```
# openssl passwd -1 changeme
```

Set a root password with guestfish:

```
# guestfish --rw -a <image-name>
> <fs> run
> <fs> list-filesystems
> <fs> mount /dev/sda1 /
> <fs> vi /etc/shadow
> <fs> exit
```

Change the second field of the root user from "!!" to the encrypted password.

3. Create a virtual machine in KVM from the qcow2 image. Set the disk type to **qcow2**, and set the virtual network interface device model to **virtio**. Then, start the virtual machine, and sign in as root.
4. Create or edit the `/etc/sysconfig/network` file, and add the following text:

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
```

5. Create or edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file, and add the following text:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
```

6. Move (or remove) the udev rules to avoid generating static rules for the Ethernet interface. These rules cause problems when you clone a virtual machine in Azure or Hyper-V:

```
# sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules
# sudo rm -f /etc/udev/rules.d/70-persistent-net.rules
```

7. Ensure that the network service will start at boot time by running the following command:

```
# chkconfig network on
```

8. Register your Red Hat subscription to enable the installation of packages from the RHEL repository by running the following command:

```
# subscription-manager register --auto-attach --username=XXX --password=XXX
```

9. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this configuration, open `/boot/grub/menu.lst` in a text editor, and ensure that the default kernel includes the following parameters:

```
console=ttyS0 earlyprintk=ttyS0 rootdelay=300
```

This will also ensure that all console messages are sent to the first serial port, which can assist Azure support with debugging issues.

In addition, we recommend that you remove the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. You can leave the `crashkernel` option configured if desired. Note that this parameter reduces the amount of available memory in the virtual machine by 128 MB or more, which might be problematic on smaller virtual machine sizes.

10. Add Hyper-V modules to initramfs:

Edit `/etc/dracut.conf`, and add the following content:

```
add_drivers+=" hv_vmbus hv_netvsc hv_storvsc "
```

Rebuild initramfs:

```
# dracut -f -v
```

11. Uninstall cloud-init:

```
# yum remove cloud-init
```

12. Ensure that the SSH server is installed and configured to start at boot time:

```
# chkconfig sshd on
```

Modify `/etc/ssh/sshd_config` to include the following lines:

```
PasswordAuthentication yes  
ClientAliveInterval 180
```

13. The WALinuxAgent package, `WALinuxAgent-<version>`, has been pushed to the Red Hat extras repository.

Enable the extras repository by running the following command:

```
# subscription-manager repos --enable=rhel-6-server-extras-rpms
```

14. Install the Azure Linux Agent by running the following command:

```
# yum install WALinuxAgent  
  
# chkconfig waagent on
```

15. The Azure Linux Agent can automatically configure swap space by using the local resource disk that is attached to the virtual machine after the virtual machine is provisioned on Azure. Note that the local resource disk is a temporary disk, and it might be emptied if the virtual machine is deprovisioned. After you install the Azure Linux Agent in the previous step, modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y
ResourceDisk.Filesystem=ext4
ResourceDisk.MountPoint=/mnt/resource
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

16. Unregister the subscription (if necessary) by running the following command:

```
# subscription-manager unregister
```

17. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# Note: if you are migrating a specific virtual machine and do not wish to create a generalized image,
# skip the deprovision step
# waagent -force -deprovision

# export HISTSIZE=0

# logout
```

18. Shut down the virtual machine in KVM.

19. Convert the qcow2 image to the VHD format.

NOTE

There is a known bug in qemu-img versions >=2.2.1 that results in an improperly formatted VHD. The issue has been fixed in QEMU 2.6. It is recommended to use either qemu-img 2.2.0 or lower, or update to 2.6 or higher. Reference:
<https://bugs.launchpad.net/qemu/+bug/1490611>.

First convert the image to raw format:

```
# qemu-img convert -f qcow2 -O raw rhel-6.9.qcow2 rhel-6.9.raw
```

Make sure that the size of the raw image is aligned with 1 MB. Otherwise, round up the size to align with 1 MB:

```
# MB=$((1024*1024))
# size=$(qemu-img info -f raw --output json "rhel-6.9.raw" | \
#       gawk 'match($0, /"virtual-size": ([0-9]+),/, val) {print val[1]}'')
#
# rounded_size=$(((size/$MB + 1)*$MB))
# qemu-img resize rhel-6.9.raw $rounded_size
```

Convert the raw disk to a fixed-sized VHD:

```
# qemu-img convert -f raw -o subformat=fixed -O vpc rhel-6.9.raw rhel-6.9.vhd
```

Or, with qemu version **2.6+** include the `force_size` option:

```
# qemu-img convert -f raw -o subformat=fixed,force_size -O vpc rhel-6.9.raw rhel-6.9.vhd
```

Prepare a RHEL 7 virtual machine from KVM

1. Download the KVM image of RHEL 7 from the Red Hat website. This procedure uses RHEL 7 as the example.

2. Set a root password.

Generate an encrypted password, and copy the output of the command:

```
# openssl passwd -1 changeme
```

Set a root password with guestfish:

```
# guestfish --rw -a <image-name>
> <fs> run
> <fs> list-filesystems
> <fs> mount /dev/sda1 /
> <fs> vi /etc/shadow
> <fs> exit
```

Change the second field of root user from "!!" to the encrypted password.

3. Create a virtual machine in KVM from the qcow2 image. Set the disk type to **qcow2**, and set the virtual network interface device model to **virtio**. Then, start the virtual machine, and sign in as root.
4. Create or edit the `/etc/sysconfig/network` file, and add the following text:

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
```

5. Create or edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file, and add the following text:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
NM_CONTROLLED=no
```

6. Ensure that the network service will start at boot time by running the following command:

```
# sudo systemctl enable network
```

7. Register your Red Hat subscription to enable installation of packages from the RHEL repository by running the following command:

```
# subscription-manager register --auto-attach --username=XXX --password=XXX
```

8. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this configuration, open `/etc/default/grub` in a text editor, and edit the `GRUB_CMDLINE_LINUX` parameter. For example:

```
GRUB_CMDLINE_LINUX="rootdelay=300 console=ttyS0 earlyprintk=ttyS0 net.ifnames=0"
```

This command also ensures that all console messages are sent to the first serial port, which can assist Azure support with debugging issues. The command also turns off the new RHEL 7 naming conventions for NICs.

In addition, we recommend that you remove the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. You can leave the `crashkernel` option configured if desired. Note that this parameter reduces the amount of available memory in the virtual machine by 128 MB or more, which might be problematic on smaller virtual machine sizes.

9. After you are done editing `/etc/default/grub`, run the following command to rebuild the grub configuration:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

10. Add Hyper-V modules into initramfs.

Edit `/etc/dracut.conf` and add content:

```
add_drivers+=" hv_vmbus hv_netvsc hv_storvsc "
```

Rebuild initramfs:

```
# dracut -f -v
```

11. Uninstall cloud-init:

```
# yum remove cloud-init
```

12. Ensure that the SSH server is installed and configured to start at boot time:

```
# systemctl enable sshd
```

Modify `/etc/ssh/sshd_config` to include the following lines:

```
PasswordAuthentication yes
ClientAliveInterval 180
```

13. The WALinuxAgent package, `WALinuxAgent-<version>`, has been pushed to the Red Hat extras repository. Enable the extras repository by running the following command:

```
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

14. Install the Azure Linux Agent by running the following command:

```
# yum install WALinuxAgent
```

Enable the waagent service:

```
# systemctl enable waagent.service
```

15. Do not create swap space on the operating system disk.

The Azure Linux Agent can automatically configure swap space by using the local resource disk that is attached to the virtual machine after the virtual machine is provisioned on Azure. Note that the local resource disk is a temporary disk, and it might be emptied if the virtual machine is deprovisioned. After you install the Azure Linux Agent in the previous step, modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y
ResourceDisk.Filesystem=ext4
ResourceDisk.MountPoint=/mnt/resource
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

16. Unregister the subscription (if necessary) by running the following command:

```
# subscription-manager unregister
```

17. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# Note: if you are migrating a specific virtual machine and do not wish to create a generalized image,
# skip the deprovision step
# sudo waagent -force -deprovision

# export HISTSIZE=0

# logout
```

18. Shut down the virtual machine in KVM.

19. Convert the qcow2 image to the VHD format.

NOTE

There is a known bug in qemu-img versions >=2.2.1 that results in an improperly formatted VHD. The issue has been fixed in QEMU 2.6. It is recommended to use either qemu-img 2.2.0 or lower, or update to 2.6 or higher. Reference: <https://bugs.launchpad.net/qemu/+bug/1490611>.

```
First convert the image to raw format:
```

```
# qemu-img convert -f qcow2 -O raw rhel-7.4.qcow2 rhel-7.4.raw
```

```
Make sure that the size of the raw image is aligned with 1 MB. Otherwise, round up the size to align with 1 MB:
```

```
# MB=$((1024*1024))
# size=$(qemu-img info -f raw --output json "rhel-7.4.raw" | \
#     gawk 'match($0, /"virtual-size": ([0-9]+),/, val) {print val[1]}')

# rounded_size=$(((size/$MB + 1)*$MB))
# qemu-img resize rhel-7.4.raw $rounded_size
```

```
Convert the raw disk to a fixed-sized VHD:
```

```
# qemu-img convert -f raw -o subformat=fixed -O vpc rhel-7.4.raw rhel-7.4.vhd
```

```
Or, with qemu version **2.6+** include the `force_size` option:
```

```
# qemu-img convert -f raw -o subformat=fixed,force_size -O vpc rhel-7.4.raw rhel-7.4.vhd
```

Prepare a Red Hat-based virtual machine from VMware

Prerequisites

This section assumes that you have already installed a RHEL virtual machine in VMware. For details about how to install an operating system in VMware, see [VMware Guest Operating System Installation Guide](#).

- When you install the Linux operating system, we recommend that you use standard partitions rather than LVM, which is often the default for many installations. This will avoid LVM name conflicts with cloned virtual machine, particularly if an operating system disk ever needs to be attached to another virtual machine for troubleshooting. LVM or RAID can be used on data disks if preferred.
- Do not configure a swap partition on the operating system disk. You can configure the Linux agent to create a swap file on the temporary resource disk. You can find more information about this in the steps that follow.
- When you create the virtual hard disk, select **Store virtual disk as a single file**.

Prepare a RHEL 6 virtual machine from VMware

1. In RHEL 6, NetworkManager can interfere with the Azure Linux agent. Uninstall this package by running the following command:

```
# sudo rpm -e --nodeps NetworkManager
```

2. Create a file named **network** in the `/etc/sysconfig/` directory that contains the following text:

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
```

3. Create or edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file, and add the following text:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
```

4. Move (or remove) the udev rules to avoid generating static rules for the Ethernet interface. These rules cause problems when you clone a virtual machine in Azure or Hyper-V:

```
# sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules  
# sudo rm -f /etc/udev/rules.d/70-persistent-net.rules
```

5. Ensure that the network service will start at boot time by running the following command:

```
# sudo chkconfig network on
```

6. Register your Red Hat subscription to enable the installation of packages from the RHEL repository by running the following command:

```
# sudo subscription-manager register --auto-attach --username=XXX --password=XXX
```

7. The WALinuxAgent package, `WALinuxAgent-<version>`, has been pushed to the Red Hat extras repository. Enable the extras repository by running the following command:

```
# subscription-manager repos --enable=rhel-6-server-extras-rpms
```

8. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this, open `/etc/default/grub` in a text editor, and edit the `GRUB_CMDLINE_LINUX` parameter. For example:

```
GRUB_CMDLINE_LINUX="rootdelay=300 console=ttyS0 earlyprintk=ttyS0"
```

This will also ensure that all console messages are sent to the first serial port, which can assist Azure support with debugging issues. In addition, we recommend that you remove the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. You can leave the `crashkernel` option configured if desired. Note that this parameter reduces the amount of available memory in the virtual machine by 128 MB or more, which might be problematic on smaller virtual machine sizes.

9. Add Hyper-V modules to initramfs:

Edit `/etc/dracut.conf`, and add the following content:

```
add_drivers+=" hv_vmbus hv_netvsc hv_storvsc "
```

Rebuild initramfs:

```
# dracut -f -v
```

10. Ensure that the SSH server is installed and configured to start at boot time, which is usually the default. Modify `/etc/ssh/sshd_config` to include the following line:

`ClientAliveInterval 180`

11. Install the Azure Linux Agent by running the following command:

```
# sudo yum install WALinuxAgent  
# sudo chkconfig waagent on
```

12. Do not create swap space on the operating system disk.

The Azure Linux Agent can automatically configure swap space by using the local resource disk that is attached to the virtual machine after the virtual machine is provisioned on Azure. Note that the local resource disk is a temporary disk, and it might be emptied if the virtual machine is deprovisioned. After you install the Azure Linux Agent in the previous step, modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y  
ResourceDisk.Filesystem=ext4  
ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=y  
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

13. Unregister the subscription (if necessary) by running the following command:

```
# sudo subscription-manager unregister
```

14. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# Note: if you are migrating a specific virtual machine and do not wish to create a generalized image,  
# skip the deprovision step  
# sudo waagent -force -deprovision  
  
# export HISTSIZE=0  
  
# logout
```

15. Shut down the virtual machine, and convert the VMDK file to a .vhdx file.

NOTE

There is a known bug in qemu-img versions $\geq 2.2.1$ that results in an improperly formatted VHD. The issue has been fixed in QEMU 2.6. It is recommended to use either qemu-img 2.2.0 or lower, or update to 2.6 or higher. Reference: <https://bugs.launchpad.net/qemu/+bug/1490611>.

```
First convert the image to raw format:
```

```
# qemu-img convert -f vmdk -O raw rhel-6.9.vmdk rhel-6.9.raw
```

Make sure that the size of the raw image is aligned with 1 MB. Otherwise, round up the size to align with 1 MB:

```
# MB=$((1024*1024))
# size=$(qemu-img info -f raw --output json "rhel-6.9.raw" | \
#     gawk 'match($0, /"virtual-size": ([0-9]+),/, val) {print val[1]}')

# rounded_size=$(((size/$MB + 1)*$MB))
# qemu-img resize rhel-6.9.raw $rounded_size
```

Convert the raw disk to a fixed-sized VHD:

```
# qemu-img convert -f raw -o subformat=fixed -O vpc rhel-6.9.raw rhel-6.9.vhd
```

Or, with qemu version **2.6+** include the `force_size` option:

```
# qemu-img convert -f raw -o subformat=fixed,force_size -O vpc rhel-6.9.raw rhel-6.9.vhd
```

Prepare a RHEL 7 virtual machine from VMware

1. Create or edit the `/etc/sysconfig/network` file, and add the following text:

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
```

2. Create or edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file, and add the following text:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
NM_CONTROLLED=no
```

3. Ensure that the network service will start at boot time by running the following command:

```
# sudo systemctl enable network
```

4. Register your Red Hat subscription to enable the installation of packages from the RHEL repository by running the following command:

```
# sudo subscription-manager register --auto-attach --username=XXX --password=XXX
```

5. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this modification, open `/etc/default/grub` in a text editor, and edit the `GRUB_CMDLINE_LINUX` parameter. For example:

```
GRUB_CMDLINE_LINUX="rootdelay=300 console=ttyS0 earlyprintk=ttyS0 net.ifnames=0"
```

This configuration also ensures that all console messages are sent to the first serial port, which can assist

Azure support with debugging issues. It also turns off the new RHEL 7 naming conventions for NICs. In addition, we recommend that you remove the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port. You can leave the `crashkernel` option configured if desired. Note that this parameter reduces the amount of available memory in the virtual machine by 128 MB or more, which might be problematic on smaller virtual machine sizes.

6. After you are done editing `/etc/default/grub`, run the following command to rebuild the grub configuration:

```
# sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

7. Add Hyper-V modules to initramfs.

Edit `/etc/dracut.conf`, add content:

```
add_drivers+=" hv_vmbus hv_netvsc hv_storvsc "
```

Rebuild initramfs:

```
# dracut -f -v
```

8. Ensure that the SSH server is installed and configured to start at boot time. This setting is usually the default. Modify `/etc/ssh/sshd_config` to include the following line:

```
ClientAliveInterval 180
```

9. The WALinuxAgent package, `WALinuxAgent-<version>`, has been pushed to the Red Hat extras repository. Enable the extras repository by running the following command:

```
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

10. Install the Azure Linux Agent by running the following command:

```
# sudo yum install WALinuxAgent  
# sudo systemctl enable waagent.service
```

11. Do not create swap space on the operating system disk.

The Azure Linux Agent can automatically configure swap space by using the local resource disk that is attached to the virtual machine after the virtual machine is provisioned on Azure. Note that the local resource disk is a temporary disk, and it might be emptied if the virtual machine is deprovisioned. After you install the Azure Linux Agent in the previous step, modify the following parameters in `/etc/waagent.conf` appropriately:

```
ResourceDisk.Format=y
ResourceDisk.Filesystem=ext4
ResourceDisk.MountPoint=/mnt/resource
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

12. If you want to unregister the subscription, run the following command:

```
# sudo subscription-manager unregister
```

13. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# Note: if you are migrating a specific virtual machine and do not wish to create a generalized image,
# skip the deprovision step
# sudo waagent -force -deprovision

# export HISTSIZE=0

# logout
```

14. Shut down the virtual machine, and convert the VMDK file to the VHD format.

NOTE

There is a known bug in qemu-img versions >=2.2.1 that results in an improperly formatted VHD. The issue has been fixed in QEMU 2.6. It is recommended to use either qemu-img 2.2.0 or lower, or update to 2.6 or higher. Reference: <https://bugs.launchpad.net/qemu/+bug/1490611>.

First convert the image to raw format:

```
# qemu-img convert -f vmdk -O raw rhel-7.4.vmdk rhel-7.4.raw
```

Make sure that the size of the raw image is aligned with 1 MB. Otherwise, round up the size to align with 1 MB:

```
# MB=$((1024*1024))
# size=$(qemu-img info -f raw --output json "rhel-7.4.raw" | \
#       gawk 'match($0, /"virtual-size": ([0-9]+),/, val) {print val[1]}')

# rounded_size=$(((size/$MB + 1)*$MB))
# qemu-img resize rhel-7.4.raw $rounded_size
```

Convert the raw disk to a fixed-sized VHD:

```
# qemu-img convert -f raw -o subformat=fixed -O vpc rhel-7.4.raw rhel-7.4.vhd
```

Or, with qemu version **2.6+** include the `force_size` option:

```
# qemu-img convert -f raw -o subformat=fixed,force_size -O vpc rhel-7.4.raw rhel-7.4.vhd
```

Prepare a Red Hat-based virtual machine from an ISO by using a kickstart file automatically

Prepare a RHEL 7 virtual machine from a kickstart file

1. Create a kickstart file that includes the following content, and save the file. For details about kickstart installation, see the [Kickstart Installation Guide](#).

```
# Kickstart for provisioning a RHEL 7 Azure VM

# System authorization information
auth --enablesshadow --passalgo=sha512

# Use graphical install
text

# Do not run the Setup Agent on first boot
firstboot --disable

# Keyboard layouts
keyboard --vckeymap=us --xlayouts='us'

# System language
lang en_US.UTF-8

# Network information
network --bootproto=dhcp

# Root password
rootpw --plaintext "to_be_disabled"

# System services
services --enabled="sshd,waagent,NetworkManager"

# System timezone
timezone Etc/UTC --isUtc --ntpservers
0.rhel.pool.ntp.org,1.rhel.pool.ntp.org,2.rhel.pool.ntp.org,3.rhel.pool.ntp.org

# Partition clearing information
clearpart --all --initlabel

# Clear the MBR
zerombr

# Disk partitioning information
part /boot --fstype="xfs" --size=500
part / --fstype="xfs" --size=1 --grow --asprimary

# System bootloader configuration
bootloader --location=mbr

# Firewall configuration
firewall --disabled

# Enable SELinux
selinux --enforcing

# Don't configure X
skipx

# Power down the machine after install
poweroff

%packages
@base
@console-internet
chrony
sudo
parted
-dracut-config-rescue

%end

%post --log=/var/log/anaconda/post-install.log

#!/bin/bash
```

```

# Register Red Hat Subscription
subscription-manager register --username=XXX --password=XXX --auto-attach --force

# Install latest repo update
yum update -y

# Enable extras repo
subscription-manager repos --enable=rhel-7-server-extras-rpms

# Install WALinuxAgent
yum install -y WALinuxAgent

# Unregister Red Hat subscription
subscription-manager unregister

# Enable waagent at boot-up
systemctl enable waagent

# Disable the root account
usermod root -p '!!!'

# Configure swap in WALinuxAgent
sed -i 's/^(\ResourceDisk\.EnableSwap\)=\[Nn]\$/\1=y/g' /etc/waagent.conf
sed -i 's/^(\ResourceDisk\.SwapSizeMB\)=[0-9]*$/\1=2048/g' /etc/waagent.conf

# Set the cmdline
sed -i 's/^(\GRUB_CMDLINE_LINUX\)=.*$/\1="console=tty1 console=ttyS0 earlyprintk=ttyS0
rootdelay=300"/g' /etc/default/grub

# Enable SSH keepalive
sed -i 's/^#\!(ClientAliveInterval\).*/\1 180/g' /etc/ssh/sshd_config

# Build the grub cfg
grub2-mkconfig -o /boot/grub2/grub.cfg

# Configure network
cat << EOF > /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
NM_CONTROLLED=no
EOF

# Deprovision and prepare for Azure if you are creating a generalized image
waagent -force -deprovision

%end

```

2. Place the kickstart file where the installation system can access it.
3. In Hyper-V Manager, create a new virtual machine. On the **Connect Virtual Hard Disk** page, select **Attach a virtual hard disk later**, and complete the New Virtual Machine Wizard.
4. Open the virtual machine settings:
 - a. Attach a new virtual hard disk to the virtual machine. Make sure to select **VHD Format** and **Fixed Size**.
 - b. Attach the installation ISO to the DVD drive.
 - c. Set the BIOS to boot from CD.
5. Start the virtual machine. When the installation guide appears, press **Tab** to configure the boot options.

6. Enter `inst.ks=<the location of the kickstart file>` at the end of the boot options, and press **Enter**.
7. Wait for the installation to finish. When it's finished, the virtual machine will be shut down automatically.
Your Linux VHD is now ready to be uploaded to Azure.

Known issues

The Hyper-V driver could not be included in the initial RAM disk when using a non-Hyper-V hypervisor

In some cases, Linux installers might not include the drivers for Hyper-V in the initial RAM disk (initrd or initramfs) unless Linux detects that it is running in a Hyper-V environment.

When you're using a different virtualization system (that is, Virtualbox, Xen, etc.) to prepare your Linux image, you might need to rebuild initrd to ensure that at least the hv_vmbus and hv_storvsc kernel modules are available on the initial RAM disk. This is a known issue at least on systems that are based on the upstream Red Hat distribution.

To resolve this issue, add Hyper-V modules to initramfs and rebuild it:

Edit `/etc/dracut.conf`, and add the following content:

```
add_drivers+=" hv_vmbus hv_netvsc hv_storvsc "
```

Rebuild initramfs:

```
# dracut -f -v
```

For more details, see the information about [rebuilding initramfs](#).

Next steps

You're now ready to use your Red Hat Enterprise Linux virtual hard disk to create new virtual machines in Azure. If this is the first time that you're uploading the .vhd file to Azure, see [Create a Linux VM from a custom disk](#).

For more details about the hypervisors that are certified to run Red Hat Enterprise Linux, see [the Red Hat website](#).

Prepare a Debian VHD for Azure

3/14/2019 • 3 minutes to read • [Edit Online](#)

Prerequisites

This section assumes that you have already installed a Debian Linux operating system from an .iso file downloaded from the [Debian website](#) to a virtual hard disk. Multiple tools exist to create .vhd files; Hyper-V is only one example. For instructions using Hyper-V, see [Install the Hyper-V Role and Configure a Virtual Machine](#).

Installation notes

- See also [General Linux Installation Notes](#) for more tips on preparing Linux for Azure.
- The newer VHDX format is not supported in Azure. You can convert the disk to VHD format using Hyper-V Manager or the **convert-vhd** cmdlet.
- When installing the Linux system, it is recommended that you use standard partitions rather than LVM (often the default for many installations). This will avoid LVM name conflicts with cloned VMs, particularly if an OS disk ever needs to be attached to another VM for troubleshooting. [LVM](#) or [RAID](#) may be used on data disks if preferred.
- Do not configure a swap partition on the OS disk. The Azure Linux agent can be configured to create a swap file on the temporary resource disk. More information can be found in the steps below.
- All VHDs on Azure must have a virtual size aligned to 1MB. When converting from a raw disk to VHD, you must ensure that the raw disk size is a multiple of 1MB before conversion. For more information, see [Linux Installation Notes](#).

Use Azure-Manage to create Debian VHDs

There are tools available for generating Debian VHDs for Azure, such as the `azure-manage` scripts from [Credativ](#). This is the recommended approach versus creating an image from scratch. For example, to create a Debian 8 VHD run the following commands to download the `azure-manage` utility (and dependencies) and run the `azure_build_image` script:

```
# sudo apt-get update
# sudo apt-get install git qemu-utils mbr kpartx debootstrap

# sudo apt-get install python3-pip python3-dateutil python3-cryptography
# sudo pip3 install azure-storage azure-servicemanagement-legacy azure-common pytest pyyaml
# git clone https://github.com/credativ/azure-manage.git
# cd azure-manage
# sudo pip3 install .

# sudo azure_build_image --option release=jessie --option image_size_gb=30 --option image_prefix=debian-jessie-azure section
```

Manually prepare a Debian VHD

1. In Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open a console window for the virtual machine.
3. If you installed the OS using an ISO, then comment out any line relating to "`deb cdrom`" in `/etc/apt/source.list`.

4. Edit the `/etc/default/grub` file and modify the **GRUB_CMDLINE_LINUX** parameter as follows to include additional kernel parameters for Azure.

```
GRUB_CMDLINE_LINUX="console=tty0 console=ttyS0,115200n8 earlyprintk=ttyS0,115200"
```

5. Rebuild the grub and run:

```
# sudo update-grub
```

6. Add Debian's Azure repositories to `/etc/apt/sources.list` for either Debian 8 or 9:

Debian 8.x "Jessie"

```
deb http://debian-archive.trafficmanager.net/debian jessie main
deb-src http://debian-archive.trafficmanager.net/debian jessie main
deb http://debian-archive.trafficmanager.net/debian-security jessie/updates main
deb-src http://debian-archive.trafficmanager.net/debian-security jessie/updates
deb http://debian-archive.trafficmanager.net/debian jessie-updates main
deb-src http://debian-archive.trafficmanager.net/debian jessie-updates main
deb http://debian-archive.trafficmanager.net/debian jessie-backports main
deb-src http://debian-archive.trafficmanager.net/debian jessie-backports main
```

Debian 9.x "Stretch"

```
deb http://debian-archive.trafficmanager.net/debian stretch main
deb-src http://debian-archive.trafficmanager.net/debian stretch main
deb http://debian-archive.trafficmanager.net/debian-security stretch/updates main
deb-src http://debian-archive.trafficmanager.net/debian-security stretch/updates main
deb http://debian-archive.trafficmanager.net/debian stretch-updates main
deb-src http://debian-archive.trafficmanager.net/debian stretch-updates main
deb http://debian-archive.trafficmanager.net/debian stretch-backports main
deb-src http://debian-archive.trafficmanager.net/debian stretch-backports main
```

7. Install the Azure Linux Agent:

```
# sudo apt-get update
# sudo apt-get install waagent
```

8. For Debian 9+, it is recommended to use the new Debian Cloud kernel for use with VMs in Azure. To install this new kernel, first create a file called `/etc/apt/preferences.d/linux.pref` with the following contents:

```
Package: linux-* initramfs-tools
Pin: release n=stretch-backports
Pin-Priority: 500
```

Then run `"sudo apt-get install linux-image-cloud-amd64"` to install the new Debian Cloud kernel.

9. Deprovision the virtual machine and prepare it for provisioning on Azure and run:

```
# sudo waagent -force -deprovision
# export HISTSIZE=0
# logout
```

10. Click **Action** -> Shut Down in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Next steps

You're now ready to use your Debian virtual hard disk to create new virtual machines in Azure. If this is the first time that you're uploading the .vhdx file to Azure, see [Create a Linux VM from a custom disk](#).

Prepare a SLES or openSUSE virtual machine for Azure

6/18/2019 • 6 minutes to read • [Edit Online](#)

NOTE

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using both models, but Microsoft recommends that most new deployments use the Resource Manager model.

Prerequisites

This article assumes that you have already installed a SUSE or openSUSE Linux operating system to a virtual hard disk. Multiple tools exist to create .vhdx files, for example a virtualization solution such as Hyper-V. For instructions, see [Install the Hyper-V Role and Configure a Virtual Machine](#).

SLES / openSUSE installation notes

- Please see also [General Linux Installation Notes](#) for more tips on preparing Linux for Azure.
- The VHDX format is not supported in Azure, only **fixed VHD**. You can convert the disk to VHD format using Hyper-V Manager or the convert-vhd cmdlet.
- When installing the Linux system it is recommended that you use standard partitions rather than LVM (often the default for many installations). This will avoid LVM name conflicts with cloned VMs, particularly if an OS disk ever needs to be attached to another VM for troubleshooting. [LVM](#) or [RAID](#) may be used on data disks if preferred.
- Do not configure a swap partition on the OS disk. The Linux agent can be configured to create a swap file on the temporary resource disk. More information about this can be found in the steps below.
- All VHDs on Azure must have a virtual size aligned to 1MB. When converting from a raw disk to VHD you must ensure that the raw disk size is a multiple of 1MB before conversion. See [Linux Installation Notes](#) for more information.

Use SUSE Studio

[SUSE Studio](#) can easily create and manage your SLES and openSUSE images for Azure and Hyper-V. This is the recommended approach for customizing your own SLES and openSUSE images.

As an alternative to building your own VHD, SUSE also publishes BYOS (Bring Your Own Subscription) images for SLES at [VMDepot](#).

Prepare SUSE Linux Enterprise Server 11 SP4

1. In the center pane of Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open the window for the virtual machine.
3. Register your SUSE Linux Enterprise system to allow it to download updates and install packages.
4. Update the system with the latest patches:

```
# sudo zypper update
```

5. Install the Azure Linux Agent from the SLES repository:

```
# sudo zypper install python-azure-agent
```

6. Check if waagent is set to "on" in chkconfig, and if not, enable it for autostart:

```
# sudo chkconfig waagent on
```

7. Check if waagent service is running, and if not, start it:

```
# sudo service waagent start
```

8. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this open "/boot/grub/menu.lst" in a text editor and ensure that the default kernel includes the following parameters:

```
console=ttyS0 earlyprintk=ttyS0 rootdelay=300
```

This will ensure all console messages are sent to the first serial port, which can assist Azure support with debugging issues.

9. Confirm that /boot/grub/menu.lst and /etc/fstab both reference the disk using its UUID (by-uuid) instead of the disk ID (by-id).

Get disk UUID

```
# ls /dev/disk/by-uuid/
```

If /dev/disk/by-id/ is used, update both /boot/grub/menu.lst and /etc/fstab with the proper by-uuid value

Before change

```
root=/dev/disk/by-id/SCSI-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx-part1
```

After change

```
root=/dev/disk/by-uuid/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

10. Modify udev rules to avoid generating static rules for the Ethernet interface(s). These rules can cause problems when cloning a virtual machine in Microsoft Azure or Hyper-V:

```
# sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules  
# sudo rm -f /etc/udev/rules.d/70-persistent-net.rules
```

11. It is recommended to edit the file "/etc/sysconfig/network/dhcp" and change the `DHCLIENT_SET_HOSTNAME` parameter to the following:

`DHCLIENT_SET_HOSTNAME="no"`

12. In "/etc/sudoers", comment out or remove the following lines if they exist:

`Defaults targetpw # ask for the password of the target user i.e. root ALL ALL=(ALL) ALL # WARNING!`

Only use this together with 'Defaults targetpw'!

13. Ensure that the SSH server is installed and configured to start at boot time. This is usually the default.

14. Do not create swap space on the OS disk.

The Azure Linux Agent can automatically configure swap space using the local resource disk that is attached to the VM after provisioning on Azure. Note that the local resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned. After installing the Azure Linux Agent (see previous step), modify the following parameters in /etc/waagent.conf appropriately:

```
ResourceDisk.Format=y ResourceDisk.Filesystem=ext4 ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=y ResourceDisk.SwapSizeMB=2048 ## NOTE: set this to whatever you need it to be.
```

15. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# sudo waagent -force -deprovision  
# export HISTSIZE=0  
# logout
```

16. Click **Action -> Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Prepare openSUSE 13.1+

1. In the center pane of Hyper-V Manager, select the virtual machine.

2. Click **Connect** to open the window for the virtual machine.

3. On the shell, run the command '`zypper lr`'. If this command returns output similar to the following, then the repositories are configured as expected--no adjustments are necessary (note that version numbers may vary):

#	Alias	Name	Enabled	Refresh
1	Cloud:Tools_13.1	Cloud:Tools_13.1	Yes	Yes
2	openSUSE_13.1_OSS	openSUSE_13.1_OSS	Yes	Yes
3	openSUSE_13.1_Updates	openSUSE_13.1_Updates	Yes	Yes

If the command returns "No repositories defined..." then use the following commands to add these repos:

```
# sudo zypper ar -f http://download.opensuse.org/repositories/Cloud:Tools/openSUSE_13.1  
Cloud:Tools_13.1  
# sudo zypper ar -f https://download.opensuse.org/distribution/13.1/repo/oss openSUSE_13.1_OSS  
# sudo zypper ar -f http://download.opensuse.org/update/13.1 openSUSE_13.1_Updates
```

You can then verify the repositories have been added by running the command '`zypper lr`' again. In case one of the relevant update repositories is not enabled, enable it with following command:

```
# sudo zypper mr -e [NUMBER OF REPOSITORY]
```

4. Update the kernel to the latest available version:

```
# sudo zypper up kernel-default
```

Or to update the system with all the latest patches:

```
# sudo zypper update
```

5. Install the Azure Linux Agent.

```
# sudo zypper install WALinuxAgent
```

6. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this, open "/boot/grub/menu.lst" in a text editor and ensure that the default kernel includes the following parameters:

```
console=ttyS0 earlyprintk=ttyS0 rootdelay=300
```

This will ensure all console messages are sent to the first serial port, which can assist Azure support with debugging issues. In addition, remove the following parameters from the kernel boot line if they exist:

```
libata.atapi_enabled=0 reserve=0x1f0,0x8
```

7. It is recommended to edit the file "/etc/sysconfig/network/dhcp" and change the **DHCLIENT_SET_HOSTNAME** parameter to the following:

```
DHCLIENT_SET_HOSTNAME="no"
```

8. **Important:** In "/etc/sudoers", comment out or remove the following lines if they exist:

```
Defaults targetpw # ask for the password of the target user i.e. root ALL ALL=(ALL) ALL # WARNING!  
Only use this together with 'Defaults targetpw'!
```

9. Ensure that the SSH server is installed and configured to start at boot time. This is usually the default.

10. Do not create swap space on the OS disk.

The Azure Linux Agent can automatically configure swap space using the local resource disk that is attached to the VM after provisioning on Azure. Note that the local resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned. After installing the Azure Linux Agent (see previous step), modify the following parameters in /etc/waagent.conf appropriately:

```
ResourceDisk.Format=y ResourceDisk.Filesystem=ext4 ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=y ResourceDisk.SwapSizeMB=2048 ## NOTE: set this to whatever you need it  
to be.
```

11. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# sudo waagent -force -deprovision  
# export HISTSIZE=0  
# logout
```

12. Ensure the Azure Linux Agent runs at startup:

```
# sudo systemctl enable waagent.service
```

13. Click **Action -> Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Next steps

You're now ready to use your SUSE Linux virtual hard disk to create new virtual machines in Azure. If this is the first time that you're uploading the .vhdx file to Azure, see [Create a Linux VM from a custom disk](#).

Prepare an Oracle Linux virtual machine for Azure

6/18/2019 • 7 minutes to read • [Edit Online](#)

NOTE

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using both models, but Microsoft recommends that most new deployments use the Resource Manager model.

Prerequisites

This article assumes that you have already installed an Oracle Linux operating system to a virtual hard disk. Multiple tools exist to create .vhd files, for example a virtualization solution such as Hyper-V. For instructions, see [Install the Hyper-V Role and Configure a Virtual Machine](#).

Oracle Linux installation notes

- Please see also [General Linux Installation Notes](#) for more tips on preparing Linux for Azure.
- Oracle's Red Hat compatible kernel and their UEK3 (Unbreakable Enterprise Kernel) are both supported on Hyper-V and Azure. For best results, please be sure to update to the latest kernel while preparing your Oracle Linux VHD.
- Oracle's UEK2 is not supported on Hyper-V and Azure as it does not include the required drivers.
- The VHDX format is not supported in Azure, only **fixed VHD**. You can convert the disk to VHD format using Hyper-V Manager or the convert-vhd cmdlet.
- When installing the Linux system it is recommended that you use standard partitions rather than LVM (often the default for many installations). This will avoid LVM name conflicts with cloned VMs, particularly if an OS disk ever needs to be attached to another VM for troubleshooting. [LVM](#) or [RAID](#) may be used on data disks if preferred.
- NUMA is not supported for larger VM sizes due to a bug in Linux kernel versions below 2.6.37. This issue primarily impacts distributions using the upstream Red Hat 2.6.32 kernel. Manual installation of the Azure Linux agent (waagent) will automatically disable NUMA in the GRUB configuration for the Linux kernel. More information about this can be found in the steps below.
- Do not configure a swap partition on the OS disk. The Linux agent can be configured to create a swap file on the temporary resource disk. More information about this can be found in the steps below.
- All VHDs on Azure must have a virtual size aligned to 1MB. When converting from a raw disk to VHD you must ensure that the raw disk size is a multiple of 1MB before conversion. See [Linux Installation Notes](#) for more information.
- Make sure that the `Addons` repository is enabled. Edit the file `/etc/yum.repos.d/public-yum-ol6.repo` (Oracle Linux 6) or `/etc/yum.repos.d/public-yum-ol7.repo` (Oracle Linux 7), and change the line `enabled=0` to `enabled=1` under `[ol6_addons]` or `[ol7_addons]` in this file.

Oracle Linux 6.4+

You must complete specific configuration steps in the operating system for the virtual machine to run in Azure.

1. In the center pane of Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open the window for the virtual machine.
3. Uninstall NetworkManager by running the following command:

```
# sudo rpm -e --nodeps NetworkManager
```

Note: If the package is not already installed, this command will fail with an error message. This is expected.

4. Create a file named **network** in the `/etc/sysconfig/` directory that contains the following text:

```
NETWORKING=yes  
HOSTNAME=localhost.localdomain
```

5. Create a file named **ifcfg-eth0** in the `/etc/sysconfig/network-scripts/` directory that contains the following text:

```
DEVICE=eth0  
ONBOOT=yes  
BOOTPROTO=dhcp  
TYPE=Ethernet  
USERCTL=no  
PEERDNS=yes  
IPV6INIT=no
```

6. Modify udev rules to avoid generating static rules for the Ethernet interface(s). These rules can cause problems when cloning a virtual machine in Microsoft Azure or Hyper-V:

```
# sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules  
# sudo rm -f /etc/udev/rules.d/70-persistent-net.rules
```

7. Ensure the network service will start at boot time by running the following command:

```
# chkconfig network on
```

8. Install `python-pyasn1` by running the following command:

```
# sudo yum install python-pyasn1
```

9. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this open `/boot/grub/menu.lst` in a text editor and ensure that the default kernel includes the following parameters:

```
console=ttyS0 earlyprintk=ttyS0 rootdelay=300 numa=off
```

This will also ensure all console messages are sent to the first serial port, which can assist Azure support with debugging issues. This will disable NUMA due to a bug in Oracle's Red Hat compatible kernel.

In addition to the above, it is recommended to *remove* the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port.

The `crashkernel` option may be left configured if desired, but note that this parameter will reduce the

amount of available memory in the VM by 128MB or more, which may be problematic on the smaller VM sizes.

10. Ensure that the SSH server is installed and configured to start at boot time. This is usually the default.
11. Install the Azure Linux Agent by running the following command. The latest version is 2.0.15.

```
# sudo yum install WALinuxAgent
```

Note that installing the WALinuxAgent package will remove the NetworkManager and NetworkManager-gnome packages if they were not already removed as described in step 2.

12. Do not create swap space on the OS disk.

The Azure Linux Agent can automatically configure swap space using the local resource disk that is attached to the VM after provisioning on Azure. Note that the local resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned. After installing the Azure Linux Agent (see previous step), modify the following parameters in /etc/waagent.conf appropriately:

```
ResourceDisk.Format=y
ResourceDisk.Filesystem=ext4
ResourceDisk.MountPoint=/mnt/resource
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

13. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# sudo waagent -force -deprovision
# export HISTSIZE=0
# logout
```

14. Click **Action -> Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Oracle Linux 7.0+

Changes in Oracle Linux 7

Preparing an Oracle Linux 7 virtual machine for Azure is very similar to Oracle Linux 6, however there are several important differences worth noting:

- Both the Red Hat compatible kernel and Oracle's UEK3 are supported in Azure. The UEK3 kernel is recommended.
- The NetworkManager package no longer conflicts with the Azure Linux agent. This package is installed by default and we recommend that it is not removed.
- GRUB2 is now used as the default bootloader, so the procedure for editing kernel parameters has changed (see below).
- XFS is now the default file system. The ext4 file system can still be used if desired.

Configuration steps

1. In Hyper-V Manager, select the virtual machine.
2. Click **Connect** to open a console window for the virtual machine.
3. Create a file named **network** in the `/etc/sysconfig/` directory that contains the following text:

```
NETWORKING=yes  
HOSTNAME=localhost.localdomain
```

4. Create a file named **ifcfg-eth0** in the `/etc/sysconfig/network-scripts/` directory that contains the following text:

```
DEVICE=eth0  
ONBOOT=yes  
BOOTPROTO=dhcp  
TYPE=Ethernet  
USERCTL=no  
PEERDNS=yes  
IPV6INIT=no
```

5. Modify udev rules to avoid generating static rules for the Ethernet interface(s). These rules can cause problems when cloning a virtual machine in Microsoft Azure or Hyper-V:

```
# sudo ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules
```

6. Ensure the network service will start at boot time by running the following command:

```
# sudo chkconfig network on
```

7. Install the `python-pyasn1` package by running the following command:

```
# sudo yum install python-pyasn1
```

8. Run the following command to clear the current yum metadata and install any updates:

```
# sudo yum clean all  
# sudo yum -y update
```

9. Modify the kernel boot line in your grub configuration to include additional kernel parameters for Azure. To do this open `/etc/default/grub` in a text editor and edit the `GRUB_CMDLINE_LINUX` parameter, for example:

```
GRUB_CMDLINE_LINUX="rootdelay=300 console=ttyS0 earlyprintk=ttyS0 net.ifnames=0"
```

This will also ensure all console messages are sent to the first serial port, which can assist Azure support with debugging issues. It also turns off the new OEL 7 naming conventions for NICs. In addition to the above, it is recommended to *remove* the following parameters:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot are not useful in a cloud environment where we want all the logs to be sent to the serial port.

The `crashkernel` option may be left configured if desired, but note that this parameter will reduce the amount of available memory in the VM by 128MB or more, which may be problematic on the smaller VM sizes.

10. Once you are done editing `/etc/default/grub` per above, run the following command to rebuild the grub

configuration:

```
# sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

11. Ensure that the SSH server is installed and configured to start at boot time. This is usually the default.
12. Install the Azure Linux Agent by running the following command:

```
# sudo yum install WALinuxAgent  
# sudo systemctl enable waagent
```

13. Do not create swap space on the OS disk.

The Azure Linux Agent can automatically configure swap space using the local resource disk that is attached to the VM after provisioning on Azure. Note that the local resource disk is a *temporary* disk, and might be emptied when the VM is deprovisioned. After installing the Azure Linux Agent (see the previous step), modify the following parameters in /etc/waagent.conf appropriately:

```
ResourceDisk.Format=y  
ResourceDisk.Filesystem=ext4  
ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=y  
ResourceDisk.SwapSizeMB=2048    ## NOTE: set this to whatever you need it to be.
```

14. Run the following commands to deprovision the virtual machine and prepare it for provisioning on Azure:

```
# sudo waagent -force -deprovision  
# export HISTSIZE=0  
# logout
```

15. Click **Action -> Shut Down** in Hyper-V Manager. Your Linux VHD is now ready to be uploaded to Azure.

Next steps

You're now ready to use your Oracle Linux .vhd to create new virtual machines in Azure. If this is the first time that you're uploading the .vhd file to Azure, see [Create a Linux VM from a custom disk](#).

Create and Upload an OpenBSD disk image to Azure

4/19/2019 • 3 minutes to read • [Edit Online](#)

This article shows you how to create and upload a virtual hard disk (VHD) that contains the OpenBSD operating system. After you upload it, you can use it as your own image to create a virtual machine (VM) in Azure through Azure CLI.

Prerequisites

This article assumes that you have the following items:

- **An Azure subscription** - If you don't have an account, you can create one in just a couple of minutes. If you have an MSDN subscription, see [Monthly Azure credit for Visual Studio subscribers](#). Otherwise, learn how to [create a free trial account](#).
- **Azure CLI** - Make sure you have the latest [Azure CLI](#) installed and logged in to your Azure account with [az login](#).
- **OpenBSD operating system installed in a .vhdx file** - A supported OpenBSD operating system ([6.2 version AMD64](#)) must be installed to a virtual hard disk. Multiple tools exist to create .vhdx files. For example, you can use a virtualization solution such as Hyper-V to create the .vhdx file and install the operating system. For instructions about how to install and use Hyper-V, see [Install Hyper-V and create a virtual machine](#).

Prepare OpenBSD image for Azure

On the VM where you installed the OpenBSD operating system 6.1, which added Hyper-V support, complete the following procedures:

1. If DHCP is not enabled during installation, enable the service as follows:

```
echo dhcp > /etc/hostname.hvn0
```

2. Set up a serial console as follows:

```
echo "stty com0 115200" >> /etc/boot.conf
echo "set tty com0" >> /etc/boot.conf
```

3. Configure Package installation as follows:

```
echo "https://ftp.openbsd.org/pub/OpenBSD" > /etc/installurl
```

4. By default, the `root` user is disabled on virtual machines in Azure. Users can run commands with elevated privileges by using the `doas` command on OpenBSD VM. Doas is enabled by default. For more information, see [doas.conf](#).

5. Install and configure prerequisites for the Azure Agent as follows:

```
pkg_add py-setuptools openssl git
ln -sf /usr/local/bin/python2.7 /usr/local/bin/python
ln -sf /usr/local/bin/python2.7-2to3 /usr/local/bin/2to3
ln -sf /usr/local/bin/python2.7-config /usr/local/bin/python-config
ln -sf /usr/local/bin/pydoc2.7 /usr/local/bin/pydoc
```

6. The latest release of the Azure agent can always be found on [GitHub](https://github.com/Azure/WALinuxAgent). Install the agent as follows:

```
git clone https://github.com/Azure/WALinuxAgent
cd WALinuxAgent
python setup.py install
waagent -register-service
```

IMPORTANT

After you install Azure Agent, it's a good idea to verify that it's running as follows:

```
ps auxw | grep waagent
root      79309  0.0  1.5  9184 15356 p1  S      4:11PM   0:00.46 python /usr/local/sbin/waagent -
daemon (python2.7)
cat /var/log/waagent.log
```

7. Deprovision the system to clean it and make it suitable for reprovisioning. The following command also deletes the last provisioned user account and the associated data:

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

Now you can shut down your VM.

Prepare the VHD

The VHDX format is not supported in Azure, only **fixed VHD**. You can convert the disk to fixed VHD format using Hyper-V Manager or the Powershell [convert-vhd](#) cmdlet. An example is as following.

```
Convert-VHD OpenBSD61.vhdx OpenBSD61.vhd -VHDTType Fixed
```

Create storage resources and upload

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
```

To upload your VHD, create a storage account with [az storage account create](#). Storage account names must be unique, so provide your own name. The following example creates a storage account named *mystorageaccount*:

```
az storage account create --resource-group myResourceGroup \
--name mystorageaccount \
--location eastus \
--sku Premium_LRS
```

To control access to the storage account, obtain the storage key with [az storage account keys list](#) as follows:

```
STORAGE_KEY=$(az storage account keys list \
--resource-group myResourceGroup \
--account-name mystorageaccount \
--query "[?keyName=='key1'] | [0].value" -o tsv)
```

To logically separate the VHDs you upload, create a container within the storage account with [az storage container create](#):

```
az storage container create \
--name vhds \
--account-name mystorageaccount \
--account-key ${STORAGE_KEY}
```

Finally, upload your VHD with [az storage blob upload](#) as follows:

```
az storage blob upload \
--container-name vhds \
--file ./OpenBSD61.vhd \
--name OpenBSD61.vhd \
--account-name mystorageaccount \
--account-key ${STORAGE_KEY}
```

Create VM from your VHD

You can create a VM with a [sample script](#) or directly with [az vm create](#). To specify the OpenBSD VHD you uploaded, use the `--image` parameter as follows:

```
az vm create \
--resource-group myResourceGroup \
--name myOpenBSD61 \
--image "https://mystorageaccount.blob.core.windows.net/vhds/OpenBSD61.vhd" \
--os-type linux \
--admin-username azureuser \
--ssh-key-value ~/.ssh/id_rsa.pub
```

Obtain the IP address for your OpenBSD VM with [az vm list-ip-addresses](#) as follows:

```
az vm list-ip-addresses --resource-group myResourceGroup --name myOpenBSD61
```

Now you can SSH to your OpenBSD VM as normal:

```
ssh azureuser@<ip address>
```

Next steps

If you want to know more about Hyper-V support on OpenBSD6.1, read [OpenBSD 6.1](#) and [hyperv.4](#).

If you want to create a VM from managed disk, read [az disk](#).

Introduction to FreeBSD on Azure

1/14/2019 • 3 minutes to read • [Edit Online](#)

This article provides an overview of running a FreeBSD virtual machine in Azure.

Overview

FreeBSD for Microsoft Azure is an advanced computer operating system used to power modern servers, desktops, and embedded platforms.

Microsoft Corporation is making images of FreeBSD available on Azure with the [Azure VM Guest Agent](#) pre-configured. Currently, the following FreeBSD versions are offered as images by Microsoft:

- FreeBSD 10.3-RELEASE
- FreeBSD 10.4-RELEASE
- FreeBSD 11.1-RELEASE

The agent is responsible for communication between the FreeBSD VM and the Azure fabric for operations such as provisioning the VM on first use (user name, password or SSH key, host name, etc.) and enabling functionality for selective VM extensions.

As for future versions of FreeBSD, the strategy is to stay current and make the latest releases available shortly after they are published by the FreeBSD release engineering team.

Deploying a FreeBSD virtual machine

Deploying a FreeBSD virtual machine is a straightforward process using an image from the Azure Marketplace from the Azure portal:

- [FreeBSD 10.4 on the Azure Marketplace](#)
- [FreeBSD 11.2 on the Azure Marketplace](#)

Create a FreeBSD VM through Azure CLI on FreeBSD

First you need to install [Azure CLI](#) though following command on a FreeBSD machine.

```
curl -L https://aka.ms/InstallAzureCli | bash
```

If bash is not installed on your FreeBSD machine, run following command before the installation.

```
sudo pkg install bash
```

If python is not installed on your FreeBSD machine, run following commands before the installation.

```
sudo pkg install python35
cd /usr/local/bin
sudo rm /usr/local/bin/python
sudo ln -s /usr/local/bin/python3.5 /usr/local/bin/python
```

During the installation, you are asked

Modify profile to update your \$PATH and enable shell/tab completion now? (Y/n) . If you answer **y** and enter

```
/etc/rc.conf as a path to an rc file to update , you may meet the problem  
ERROR: [Errno 13] Permission denied . To resolve this problem, you should grant the write right to current user  
against the file /etc/rc.conf .
```

Now you can sign in to Azure and create your FreeBSD VM. Below is an example to create a FreeBSD 11.0 VM. You can also add the parameter `--public-ip-address-dns-name` with a globally unique DNS name for a newly created Public IP.

```
az login  
az group create --name myResourceGroup --location eastus  
az vm create --name myFreeBSD11 \  
    --resource-group myResourceGroup \  
    --image MicrosoftOSTC:FreeBSD:11.0:latest \  
    --admin-username azureuser \  
    --generate-ssh-keys
```

Then you can sign in to your FreeBSD VM through the ip address that printed in the output of above deployment.

```
ssh azureuser@xx.xx.xx.xx -i /etc/ssh/ssh_host_rsa_key
```

VM extensions for FreeBSD

Following are supported VM extensions in FreeBSD.

VMAccess

The [VMAccess](#) extension can:

- Reset the password of the original sudo user.
- Create a new sudo user with the password specified.
- Set the public host key with the key given.
- Reset the public host key provided during VM provisioning if the host key is not provided.
- Open the SSH port (22) and restore the sshd_config if reset_ssh is set to true.
- Remove the existing user.
- Check disks.
- Repair an added disk.

CustomScript

The [CustomScript](#) extension can:

- If provided, download the customized scripts from Azure Storage or external public storage (for example, GitHub).
- Run the entry point script.
- Support inline commands.
- Convert Windows-style newline in shell and Python scripts automatically.
- Remove BOM in shell and Python scripts automatically.
- Protect sensitive data in CommandToExecute.

NOTE

FreeBSD VM only supports CustomScript version 1.x by now.

Authentication: user names, passwords, and SSH keys

When you're creating a FreeBSD virtual machine by using the Azure portal, you must provide a user name, password, or SSH public key. User names for deploying a FreeBSD virtual machine on Azure must not match names of system accounts (UID < 100) already present in the virtual machine ("root", for example). Currently, only the RSA SSH key is supported. A multiline SSH key must begin with `----- BEGIN SSH2 PUBLIC KEY -----` and end with `----- END SSH2 PUBLIC KEY -----`.

Obtaining superuser privileges

The user account that is specified during virtual machine instance deployment on Azure is a privileged account. The package of sudo was installed in the published FreeBSD image. After you're logged in through this user account, you can run commands as root by using the command syntax.

```
$ sudo <COMMAND>
```

You can optionally obtain a root shell by using `sudo -s`.

Known issues

The [Azure VM Guest Agent](#) version 2.2.2 has a [known issue](#) that causes the provision failure for FreeBSD VM on Azure. The fix was captured by [Azure VM Guest Agent](#) version 2.2.3 and later releases.

Next steps

- Go to [Azure Marketplace](#) to create a FreeBSD VM.

How to create an image of a virtual machine or VHD

2/4/2019 • 4 minutes to read • [Edit Online](#)

To create multiple copies of a virtual machine (VM) for use in Azure, capture an image of the VM or of the OS VHD. To create an image for deployment, you'll need to remove personal account information. In the following steps, you deprovision an existing VM, deallocate it and create an image. You can use this image to create VMs across any resource group within your subscription.

To create a copy of your existing Linux VM for backup or debugging, or to upload a specialized Linux VHD from an on-premises VM, see [Upload and create a Linux VM from custom disk image](#).

You can also use **Packer** to create your custom configuration. For more information, see [How to use Packer to create Linux virtual machine images in Azure](#).

You'll need the following items before creating an image:

- An Azure VM created in the Resource Manager deployment model that uses managed disks. If you haven't yet created a Linux VM, you can use the [portal](#), the [Azure CLI](#), or [Resource Manager templates](#). Configure the VM as needed. For example, [add data disks](#), apply updates, and install applications.
- The latest [Azure CLI](#) installed and be logged in to an Azure account with [az login](#).

Quick commands

For a simplified version of this article, and for testing, evaluating, or learning about VMs in Azure, see [Create a custom image of an Azure VM by using the CLI](#).

Step 1: Deprovision the VM

First you'll deprovision the VM by using the Azure VM agent to delete machine-specific files and data. Use the `waagent` command with the `-deprovision+user` parameter on your source Linux VM. For more information, see the [Azure Linux Agent user guide](#).

1. Connect to your Linux VM with an SSH client.
2. In the SSH window, enter the following command:

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

NOTE

Only run this command on a VM that you'll capture as an image. This command does not guarantee that the image is cleared of all sensitive information or is suitable for redistribution. The `+user` parameter also removes the last provisioned user account. To keep user account credentials in the VM, use only `-deprovision`.

3. Enter **y** to continue. You can add the `-force` parameter to avoid this confirmation step.

4. After the command completes, enter **exit** to close the SSH client.

Step 2: Create VM image

Use the Azure CLI to mark the VM as generalized and capture the image. In the following examples, replace

example parameter names with your own values. Example parameter names include *myResourceGroup*, *myVnet*, and *myVM*.

1. Deallocate the VM that you deprovisioned with [az vm deallocate](#). The following example deallocates the VM named *myVM* in the resource group named *myResourceGroup*.

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

2. Mark the VM as generalized with [az vm generalize](#). The following example marks the VM named *myVM* in the resource group named *myResourceGroup* as generalized.

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

3. Create an image of the VM resource with [az image create](#). The following example creates an image named *myImage* in the resource group named *myResourceGroup* using the VM resource named *myVM*.

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

NOTE

The image is created in the same resource group as your source VM. You can create VMs in any resource group within your subscription from this image. From a management perspective, you may wish to create a specific resource group for your VM resources and images.

If you would like to store your image in zone-resilient storage, you need to create it in a region that supports [availability zones](#) and include the `--zone-resilient true` parameter.

Step 3: Create a VM from the captured image

Create a VM by using the image you created with [az vm create](#). The following example creates a VM named *myVMDeployed* from the image named *myImage*.

```
az vm create \
--resource-group myResourceGroup \
--name myVMDeployed \
--image myImage \
--admin-username azureuser \
--ssh-key-value ~/.ssh/id_rsa.pub
```

Creating the VM in another resource group

You can create VMs from an image in any resource group within your subscription. To create a VM in a different resource group than the image, specify the full resource ID to your image. Use [az image list](#) to view a list of images. The output is similar to the following example.

```
"id": "/subscriptions/guid/resourceGroups/MYRESOURCEGROUP/providers/Microsoft.Compute/images/myImage",
"location": "westus",
"name": "myImage",
```

The following example uses `az vm create` to create a VM in a resource group other than the source image, by specifying the image resource ID.

```
az vm create \
--resource-group myOtherResourceGroup \
--name myOtherVMDeployed \
--image "/subscriptions/guid/resourceGroups/MYRESOURCEGROUP/providers/Microsoft.Compute/images/myImage" \
--admin-username azureuser \
--ssh-key-value ~/.ssh/id_rsa.pub
```

Step 4: Verify the deployment

SSH into the virtual machine you created to verify the deployment and start using the new VM. To connect via SSH, find the IP address or FQDN of your VM with `az vm show`.

```
az vm show \
--resource-group myResourceGroup \
--name myVMDeployed \
--show-details
```

Next steps

You can create multiple VMs from your source VM image. To make changes to your image:

- Create a VM from your image.
- Make any updates or configuration changes.
- Follow the steps again to deprovision, deallocate, generalize, and create an image.
- Use this new image for future deployments. You may delete the original image.

For more information on managing your VMs with the CLI, see [Azure CLI](#).

How to use Packer to create Linux virtual machine images in Azure

5/8/2019 • 6 minutes to read • [Edit Online](#)

Each virtual machine (VM) in Azure is created from an image that defines the Linux distribution and OS version. Images can include pre-installed applications and configurations. The Azure Marketplace provides many first and third-party images for most common distributions and application environments, or you can create your own custom images tailored to your needs. This article details how to use the open source tool [Packer](#) to define and build custom images in Azure.

NOTE

Azure now has a service, Azure Image Builder (preview), for defining and creating your own custom images. Azure Image Builder is built on Packer, so you can even use your existing Packer shell provisioner scripts with it. To get started with Azure Image Builder, see [Create a Linux VM with Azure Image Builder](#).

Create Azure resource group

During the build process, Packer creates temporary Azure resources as it builds the source VM. To capture that source VM for use as an image, you must define a resource group. The output from the Packer build process is stored in this resource group.

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

```
az group create -n myResourceGroup -l eastus
```

Create Azure credentials

Packer authenticates with Azure using a service principal. An Azure service principal is a security identity that you can use with apps, services, and automation tools like Packer. You control and define the permissions as to what operations the service principal can perform in Azure.

Create a service principal with [az ad sp create-for-rbac](#) and output the credentials that Packer needs:

```
az ad sp create-for-rbac --query "{ client_id: appId, client_secret: password, tenant_id: tenant }"
```

An example of the output from the preceding commands is as follows:

```
{
  "client_id": "f5b6a5cf-fbdf-4a9f-b3b8-3c2cd00225a4",
  "client_secret": "0e760437-bf34-4aad-9f8d-870be799c55d",
  "tenant_id": "72f988bf-86f1-41af-91ab-2d7cd011db47"
}
```

To authenticate to Azure, you also need to obtain your Azure subscription ID with [az account show](#):

```
az account show --query "{ subscription_id: id }"
```

You use the output from these two commands in the next step.

Define Packer template

To build images, you create a template as a JSON file. In the template, you define builders and provisioners that carry out the actual build process. Packer has a [provisioner for Azure](#) that allows you to define Azure resources, such as the service principal credentials created in the preceding step.

Create a file named *ubuntujson* and paste the following content. Enter your own values for the following:

PARAMETER	WHERE TO OBTAIN
<i>client_id</i>	First line of output from <code>az ad sp</code> create command - <i>appId</i>
<i>client_secret</i>	Second line of output from <code>az ad sp</code> create command - <i>password</i>
<i>tenant_id</i>	Third line of output from <code>az ad sp</code> create command - <i>tenant</i>
<i>subscription_id</i>	Output from <code>az account show</code> command
<i>managed_image_resource_group_name</i>	Name of resource group you created in the first step
<i>managed_image_name</i>	Name for the managed disk image that is created

```
{
  "builders": [
    {
      "type": "azure-arm",
      "client_id": "f5b6a5cf-fbdf-4a9f-b3b8-3c2cd00225a4",
      "client_secret": "0e760437-bf34-4aad-9f8d-870be799c55d",
      "tenant_id": "72f988bf-86f1-41af-91ab-2d7cd011db47",
      "subscription_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx",
      "managed_image_resource_group_name": "myResourceGroup",
      "managed_image_name": "myPackerImage",
      "os_type": "Linux",
      "image_publisher": "Canonical",
      "image_offer": "UbuntuServer",
      "image_sku": "16.04-LTS",
      "azure_tags": {
        "dept": "Engineering",
        "task": "Image deployment"
      },
      "location": "East US",
      "vm_size": "Standard_DS2_v2"
    }
  ],
  "provisioners": [
    {
      "execute_command": "chmod +x {{ .Path }}; {{ .Vars }} sudo -E sh '{{ .Path }}'",  

      "inline": [
        "apt-get update",
        "apt-get upgrade -y",
        "apt-get -y install nginx",
  

        "/usr/sbin/waagent -force -deprovision+user && export HISTSIZE=0 && sync"
      ],
      "inline_shebang": "/bin/sh -x",
      "type": "shell"
    }
  ]
}
```

This template builds an Ubuntu 16.04 LTS image, installs NGINX, then deprovisions the VM.

NOTE

If you expand on this template to provision user credentials, adjust the provisioner command that deprovisions the Azure agent to read `-deprovision` rather than `deprovision+user`. The `+user` flag removes all user accounts from the source VM.

Build Packer image

If you don't already have Packer installed on your local machine, [follow the Packer installation instructions](#).

Build the image by specifying your Packer template file as follows:

```
./packer build ubuntu.json
```

An example of the output from the preceding commands is as follows:

```

azure-arm output will be in this color.

==> azure-arm: Running builder ...
    azure-arm: Creating Azure Resource Manager (ARM) client ...
==> azure-arm: Creating resource group ...
==> azure-arm: -> ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: -> Location       : 'East US'
==> azure-arm: -> Tags          :
==> azure-arm: ->> dept : Engineering
==> azure-arm: ->> task : Image deployment
==> azure-arm: Validating deployment template ...
==> azure-arm: -> ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: -> DeploymentName   : 'pkrdpswtxmqm7ly'
==> azure-arm: Deploying deployment template ...
==> azure-arm: -> ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: -> DeploymentName   : 'pkrdpswtxmqm7ly'
==> azure-arm: Getting the VM's IP address ...
==> azure-arm: -> ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: -> PublicIPAddressName : 'packerPublicIP'
==> azure-arm: -> NicName        : 'packerNic'
==> azure-arm: -> Network Connection : 'PublicEndpoint'
==> azure-arm: -> IP Address      : '40.76.218.147'
==> azure-arm: Waiting for SSH to become available...
==> azure-arm: Connected to SSH!
==> azure-arm: Provisioning with shell script: /var/folders/h1/ymh5bdx15wgdn5hvgj1wc0zh0000gn/T/packer-
shell1868574263
    azure-arm: WARNING! The waagent service will be stopped.
    azure-arm: WARNING! Cached DHCP leases will be deleted.
    azure-arm: WARNING! root password will be disabled. You will not be able to login as root.
    azure-arm: WARNING! /etc/resolvconf/resolv.conf.d/tail and /etc/resolvconf/resolv.conf.d/original will be
deleted.
    azure-arm: WARNING! packer account and entire home directory will be deleted.
==> azure-arm: Querying the machine's properties ...
==> azure-arm: -> ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: -> ComputeName     : 'pkrvmswtxmqm7ly'
==> azure-arm: -> Managed OS Disk  : '/subscriptions/guid/resourceGroups/packer-Resource-Group-
swtxm7ly/providers/Microsoft.Compute/disks/osdisk'
==> azure-arm: Powering off machine ...
==> azure-arm: -> ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: -> ComputeName     : 'pkrvmswtxmqm7ly'
==> azure-arm: Capturing image ...
==> azure-arm: -> Compute ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: -> Compute Name       : 'pkrvmswtxmqm7ly'
==> azure-arm: -> Compute Location    : 'East US'
==> azure-arm: -> Image ResourceGroupName : 'myResourceGroup'
==> azure-arm: -> Image Name         : 'myPackerImage'
==> azure-arm: -> Image Location      : 'eastus'
==> azure-arm: Deleting resource group ...
==> azure-arm: -> ResourceGroupName : 'packer-Resource-Group-swtxmqm7ly'
==> azure-arm: Deleting the temporary OS disk ...
==> azure-arm: -> OS Disk : skipping, managed disk was used...
Build 'azure-arm' finished.

==> Builds finished. The artifacts of successful builds are:
--> azure-arm: Azure.ResourceManagement.VMImage:

ManagedImageResourceGroupName: myResourceGroup
ManagedImageName: myPackerImage
ManagedImageLocation: eastus

```

It takes a few minutes for Packer to build the VM, run the provisioners, and clean up the deployment.

Create VM from Azure Image

You can now create a VM from your Image with [az vm create](#). Specify the Image you created with the `--image`

parameter. The following example creates a VM named *myVM* from *myPackerImage* and generates SSH keys if they do not already exist:

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

If you wish to create VMs in a different resource group or region than your Packer image, specify the image ID rather than image name. You can obtain the image ID with [az image show](#).

It takes a few minutes to create the VM. Once the VM has been created, take note of the [publicIpAddress](#) displayed by the Azure CLI. This address is used to access the NGINX site via a web browser.

To allow web traffic to reach your VM, open port 80 from the Internet with [az vm open-port](#):

```
az vm open-port \
--resource-group myResourceGroup \
--name myVM \
--port 80
```

Test VM and NGINX

Now you can open a web browser and enter [http://publicIpAddress](#) in the address bar. Provide your own public IP address from the VM create process. The default NGINX page is displayed as in the following example:



Next steps

You can also use existing Packer provisioner scripts with [Azure Image Builder](#).

Red Hat Enterprise Linux images in Azure

6/10/2019 • 7 minutes to read • [Edit Online](#)

This article describes available Red Hat Enterprise Linux (RHEL) images in the Azure Marketplace along with policies around their naming and retention.

Information on Red Hat support policies for all versions of RHEL can be found on the [Red Hat Enterprise Linux Life Cycle](#) page.

IMPORTANT

RHEL images currently available in the Azure marketplace support either Bring-Your-Own-Subscription (BYOS) or Pay-As-You-Go (PAYG) licensing models. The [Azure Hybrid Use Benefit](#) and dynamic switching between BYOS and PAYG is not supported. Switching licensing mode requires redeploying the VM from the corresponding image.

NOTE

For any issue related to RHEL images in the Azure marketplace gallery please file a support ticket with Microsoft.

Images available in the UI

When you search for "Red Hat" in the Marketplace or when you create a resource in Azure portal UI, you'll see a subset of available RHEL images and related Red Hat products. You can always obtain the full set of available VM images using the Azure CLI/PowerShell/API.

To see the full set of available Red Hat images in Azure, run the following command

```
az vm image list --publisher RedHat --all
```

Naming convention

VM images in Azure are organized by Publisher, Offer, SKU, and Version. The combination of Publisher:Offer:SKU:Version is the image URN and uniquely identifies the image to be used.

For example, `RedHat:RHEL:7-RAW:7.6.2018103108` refers to a RHEL 7.6 raw-partitioned image built on October 31, 2018.

A sample of how to create a RHEL 7.6 VM is shown below.

```
az vm create --name RhelVM --resource-group TestRG --image RedHat:RHEL:7-RAW:7.6.2018103108 --no-wait
```

The "latest" moniker

Azure REST API allows use of moniker "latest" for version instead of the specific version. Using "latest" will provision the latest available image for the given Publisher, Offer, and SKU.

For example, `RedHat:RHEL:7-RAW:latest` refers to the latest RHEL 7 family raw-partitioned image available.

```
az vm create --name RhelVM --resource-group TestRG --image RedHat:RHEL:7-RAW:latest --no-wait
```

NOTE

In general, the comparison of versions to determine the latest follows the rules of the [CompareTo method](#).

Current naming convention

All currently published RHEL images use the Pay-As-You-Go model and are connected to [Red Hat Update Infrastructure \(RHUI\) in Azure](#). A new naming convention has been adopted for RHEL 7 family images in which the disk partitioning scheme (raw, LVM) is specified in the SKU instead of the version. The RHEL image version will contain either 7-RAW or 7-LVM. The RHEL 6 family naming hasn't been changed at this time.

There will be 2 types of RHEL 7 image SKUs in this naming convention: SKUs that list the minor version, and SKUs that don't. If you want to use a 7-RAW or 7-LVM SKU, you can specify the RHEL minor version you want to deploy in the version. If you choose the "latest" version, you will be provisioned the latest minor release of RHEL.

NOTE

In the RHEL for SAP set of images, the RHEL version remains fixed. As such, their naming convention includes a particular version in the SKU.

NOTE

RHEL 6 set of images were not moved to the new naming convention.

Extended Update Support (EUS)

As of April 2019, RHEL images are available that are attached to the Extended Update Support (EUS) repositories by default. More details on RHEL EUS are available in [Red Hat's documentation](#).

Instructions on how to switch your VM to EUS and more details about EUS support end-of-life dates are available [here](#).

NOTE

EUS is not supported on RHEL Extras. This means that if you are installing a package that is usually available from the RHEL Extras channel, you will not be able to do so while on EUS. The Red Hat Extras Product Life Cycle is detailed [here](#).

For customers that want to use EUS images:

Customers that want to use images that are attached to EUS repositories should use the RHEL image that contains a RHEL minor version number in the SKU. These images will be raw-partitioned (i.e. not LVM).

For example, you may see the following 2 RHEL 7.4 images available:

```
RedHat:RHEL:7-RAW:7.4.2018010506  
RedHat:RHEL:7.4:7.4.2019041718
```

In this case, `RedHat:RHEL:7.4:7.4.2019041718` will be attached to EUS repositories by default, and `RedHat:RHEL:7-RAW:7.4.2018010506` will be attached to non-EUS repositories by default.

For customers that don't want to use EUS images:

If you don't want to use an image that is connected to EUS by default, deploy using an image that does not contain a minor version number in the SKU.

RHEL images with EUS

The following table will apply for RHEL images that contain a minor version in the SKU.

NOTE		
At the time of writing, only RHEL 7.4 and later minor versions have EUS support. EUS is no longer supported for RHEL <= 7.3.		

MINOR VERSION	EUS IMAGE EXAMPLE	EUS STATUS
RHEL 7.4	RedHat:RHEL:7.4:7.4.2019041718	Images published April 2019 and later will be EUS by default
RHEL 7.5	RedHat:RHEL:7.5:7.5.2019060305	Images published June 2019 and later will be EUS by default
RHEL 7.6	RedHat:RHEL:7.6:7.6.2019052206	Images published May 2019 and later will be EUS by default
RHEL 8.0	N/A	No EUS currently images currently available

List of RHEL images available

The following offers are SKUs are currently available for general use:

OFFER	SKU	PARTITIONING	PROVISIONING	NOTES
RHEL	7-RAW	RAW	Linux Agent	RHEL 7 family of images. Not attached to EUS repositories by default.
	7-LVM	LVM	Linux Agent	RHEL 7 family of images. Not attached to EUS repositories by default.
	7-RAW-CI	RAW-CI	Cloud-init	RHEL 7 family of images. Not attached to EUS repositories by default.
	6.7	RAW	Linux Agent	RHEL 6.7 images, old naming convention
	6.8	RAW	Linux Agent	Same as above for RHEL 6.8

OFFER	SKU	PARTITIONING	PROVISIONING	NOTES
	6.9	RAW	Linux Agent	Same as above for RHEL 6.9
	6.10	RAW	Linux Agent	Same as above for RHEL 6.10
	7.2	RAW	Linux Agent	Same as above for RHEL 7.2
	7.3	RAW	Linux Agent	Same as above for RHEL 7.3
	7.4	RAW	Linux Agent	Same as above for RHEL 7.4. Attached to EUS repositories by default as of April 2019
	7.5	RAW	Linux Agent	Same as above for RHEL 7.5. Attached to EUS repositories by default as of June 2019
	7.6	RAW	Linux Agent	Same as above for RHEL 7.6. Attached to EUS repositories by default as of May 2019
RHEL-SAP	7.4	LVM	Linux Agent	RHEL 7.4 for SAP HANA and Business Apps
	7.5	LVM	Linux Agent	RHEL 7.5 for SAP HANA and Business Apps
RHEL-SAP-HANA	6.7	RAW	Linux Agent	RHEL 6.7 for SAP HANA
	7.2	LVM	Linux Agent	RHEL 7.2 for SAP HANA
	7.3	LVM	Linux Agent	RHEL 7.3 for SAP HANA
RHEL-SAP-APPS	6.8	RAW	Linux Agent	RHEL 6.8 for SAP Business Applications
	7.3	LVM	Linux Agent	RHEL 7.3 for SAP Business Applications

Old naming convention

The RHEL 7 family of images and the RHEL 6 family of images used specific versions in their SKUs up until the

naming convention change explained above.

You will find numeric SKUs in the full image list. Microsoft and Red Hat will create new numeric SKUs when a new minor release comes out.

Other available Offers and SKUs

The full list of available offers and SKUs may include additional images beyond what is listed in the above table, for example, `RedHat:rhel-ocp-marketplace:rhe174:7.4.1`. These offers may be used for providing support of specific marketplace solutions, or they could be published for previews and testing purposes. They may be changed or removed at any time without warning. Do not use them unless their presence has been publicly documented by either Microsoft or Red Hat.

Publishing policy

Microsoft and Red Hat update images as new minor versions are released, as required to address specific CVEs, or for occasional configuration changes/updates. We strive to provide updated images as soon as possible - within three business days following a release or availability of a CVE fix.

We only update the current minor release in a given image family. With the release of a newer minor version, we stop updating the older minor version. For example, with the release of RHEL 7.6, RHEL 7.5 images are no longer going to be updated.

NOTE

Active Azure VMs provisioned from RHEL Pay-As-You-Go images are connected to the Azure RHUI and can receive updates and fixes as soon as they are released by Red Hat and replicated to the Azure RHUI (usually in less than 24 hours following the official release by Red Hat). These VMs do not require a new published image for getting the updates and customers have full control over when to initiate the update.

Image retention policy

Our current policy is to keep all previously published images. We reserve the right to remove images that are known to cause problems of any kind. For example, images with incorrect configurations due to subsequent platform or component updates may be removed. Images that may be removed will follow the current Marketplace policy to provide notifications up to 30 days before image removal.

Next steps

- Learn more about the Azure Red Hat Update Infrastructure [here](#).
- Information on Red Hat support policies for all versions of RHEL can be found on the [Red Hat Enterprise Linux Life Cycle](#) page.

Download a Linux VHD from Azure

2/15/2019 • 2 minutes to read • [Edit Online](#)

In this article, you learn how to download a Linux virtual hard disk (VHD) file from Azure using the Azure CLI and Azure portal.

If you haven't already done so, install [Azure CLI](#).

Stop the VM

A VHD can't be downloaded from Azure if it's attached to a running VM. You need to stop the VM to download a VHD. If you want to use a VHD as an [image](#) to create other VMs with new disks, you need to deprovision and generalize the operating system contained in the file and stop the VM. To use the VHD as a disk for a new instance of an existing VM or data disk, you only need to stop and deallocate the VM.

To use the VHD as an image to create other VMs, complete these steps:

1. Use SSH, the account name, and the public IP address of the VM to connect to it and deprovision it. You can find the public IP address with [az network public-ip show](#). The `+user` parameter also removes the last provisioned user account. If you are baking account credentials in to the VM, leave out this `+user` parameter. The following example removes the last provisioned user account:

```
ssh azureuser@<publicIpAddress>
sudo waagent -deprovision+user -force
exit
```

2. Sign in to your Azure account with [az login](#).

3. Stop and deallocate the VM.

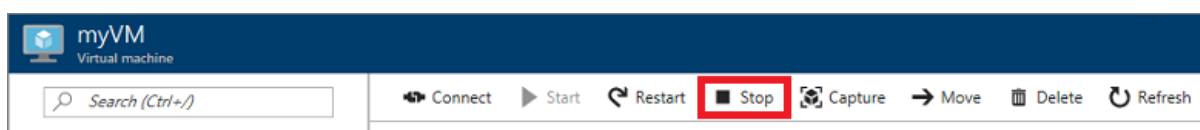
```
az vm deallocate --resource-group myResourceGroup --name myVM
```

4. Generalize the VM.

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

To use the VHD as a disk for a new instance of an existing VM or data disk, complete these steps:

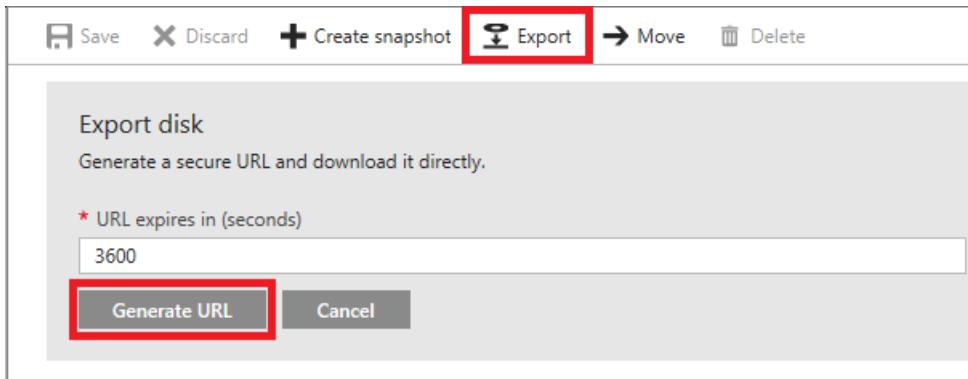
1. Sign in to the [Azure portal](#).
2. On the Hub menu, click **Virtual Machines**.
3. Select the VM from the list.
4. On the blade for the VM, click **Stop**.



Generate SAS URL

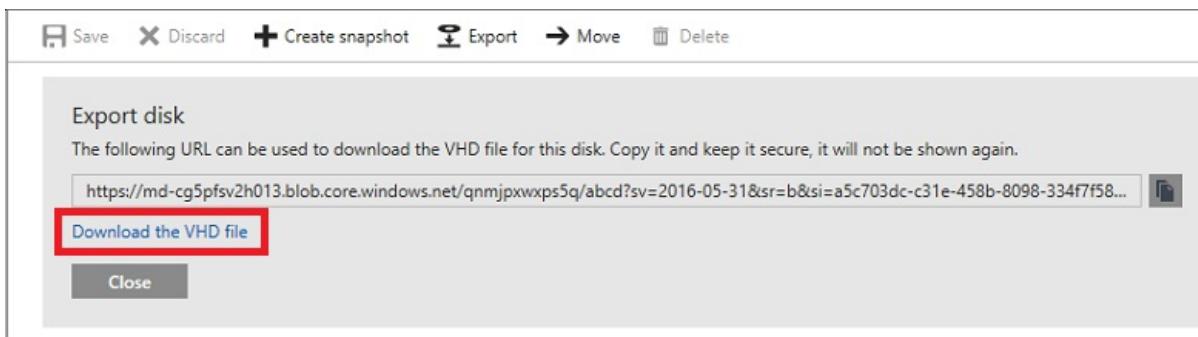
To download the VHD file, you need to generate a [shared access signature \(SAS\)](#) URL. When the URL is generated, an expiration time is assigned to the URL.

1. On the menu of the blade for the VM, click **Disks**.
2. Select the operating system disk for the VM, and then click **Export**.
3. Click **Generate URL**.



Download VHD

1. Under the URL that was generated, click Download the VHD file.



2. You may need to click **Save** in the browser to start the download. The default name for the VHD file is *abcd*.



Next steps

- Learn how to [upload and create a Linux VM from custom disk with the Azure CLI](#).
- [Manage Azure disks the Azure CLI](#).

2 minutes to read

Manage the availability of Linux virtual machines

1/24/2019 • 9 minutes to read • [Edit Online](#)

Learn ways to set up and manage multiple virtual machines to ensure high availability for your Linux application in Azure. You can also [manage the availability of Windows virtual machines](#).

For instructions on creating an availability set using CLI in the Resource Manager deployment model, see [azurce availset: commands to manage your availability sets](#).

Understand VM Reboots - maintenance vs. downtime

There are three scenarios that can lead to virtual machine in Azure being impacted: unplanned hardware maintenance, unexpected downtime, and planned maintenance.

- **Unplanned Hardware Maintenance Event** occurs when the Azure platform predicts that the hardware or any platform component associated to a physical machine, is about to fail. When the platform predicts a failure, it will issue an unplanned hardware maintenance event to reduce the impact to the virtual machines hosted on that hardware. Azure uses [Live Migration](#) technology to migrate the Virtual Machines from the failing hardware to a healthy physical machine. Live Migration is a VM preserving operation that only pauses the Virtual Machine for a short time. Memory, open files, and network connections are maintained, but performance might be reduced before and/or after the event. In cases where Live Migration cannot be used, the VM will experience Unexpected Downtime, as described below.
- **An Unexpected Downtime** is when the hardware or the physical infrastructure for the virtual machine fails unexpectedly. This can include local network failures, local disk failures, or other rack level failures. When detected, the Azure platform automatically migrates (heals) your virtual machine to a healthy physical machine in the same datacenter. During the healing procedure, virtual machines experience downtime (reboot) and in some cases loss of the temporary drive. The attached OS and data disks are always preserved.

Virtual machines can also experience downtime in the unlikely event of an outage or disaster that affects an entire datacenter, or even an entire region. For these scenarios, Azure provides protection options including [availability zones](#) and [paired regions](#).

- **Planned Maintenance events** are periodic updates made by Microsoft to the underlying Azure platform to improve overall reliability, performance, and security of the platform infrastructure that your virtual machines run on. Most of these updates are performed without any impact upon your Virtual Machines or Cloud Services (see [VM Preserving Maintenance](#)). While the Azure platform attempts to use VM Preserving Maintenance in all possible occasions, there are rare instances when these updates require a reboot of your virtual machine to apply the required updates to the underlying infrastructure. In this case, you can perform Azure Planned Maintenance with Maintenance-Redeploy operation by initiating the maintenance for their VMs in the suitable time window. For more information, see [Planned Maintenance for Virtual Machines](#).

To reduce the impact of downtime due to one or more of these events, we recommend the following high availability best practices for your virtual machines:

- [Configure multiple virtual machines in an availability set for redundancy](#)
- [Use managed disks for VMs in an availability set](#)
- [Use scheduled events to proactively response to VM impacting events](#)
- [Configure each application tier into separate availability sets](#)

- Combine a Load Balancer with availability sets
- Use availability zones to protect from datacenter level failures

Configure multiple virtual machines in an availability set for redundancy

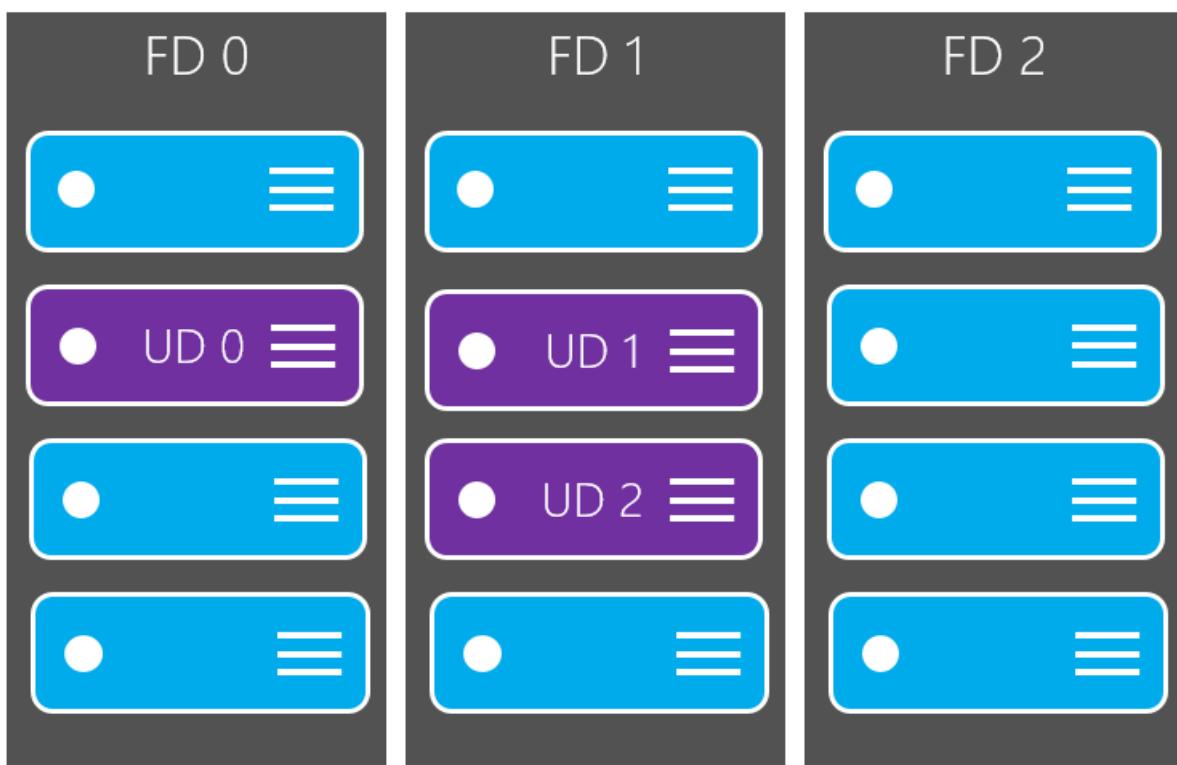
To provide redundancy to your application, we recommend that you group two or more virtual machines in an availability set. This configuration within a datacenter ensures that during either a planned or unplanned maintenance event, at least one virtual machine is available and meets the 99.95% Azure SLA. For more information, see the [SLA for Virtual Machines](#).

IMPORTANT

Avoid leaving a single instance virtual machine in an availability set by itself. VMs in this configuration do not qualify for a SLA guarantee and face downtime during Azure planned maintenance events, except when a single VM is using [Azure premium SSDs](#). For single VMs using premium SSDs, the Azure SLA applies.

Each virtual machine in your availability set is assigned an **update domain** and a **fault domain** by the underlying Azure platform. For a given availability set, five non-user-configurable update domains are assigned by default (Resource Manager deployments can then be increased to provide up to 20 update domains) to indicate groups of virtual machines and underlying physical hardware that can be rebooted at the same time. When more than five virtual machines are configured within a single availability set, the sixth virtual machine is placed into the same update domain as the first virtual machine, the seventh in the same update domain as the second virtual machine, and so on. The order of update domains being rebooted may not proceed sequentially during planned maintenance, but only one update domain is rebooted at a time. A rebooted update domain is given 30 minutes to recover before maintenance is initiated on a different update domain.

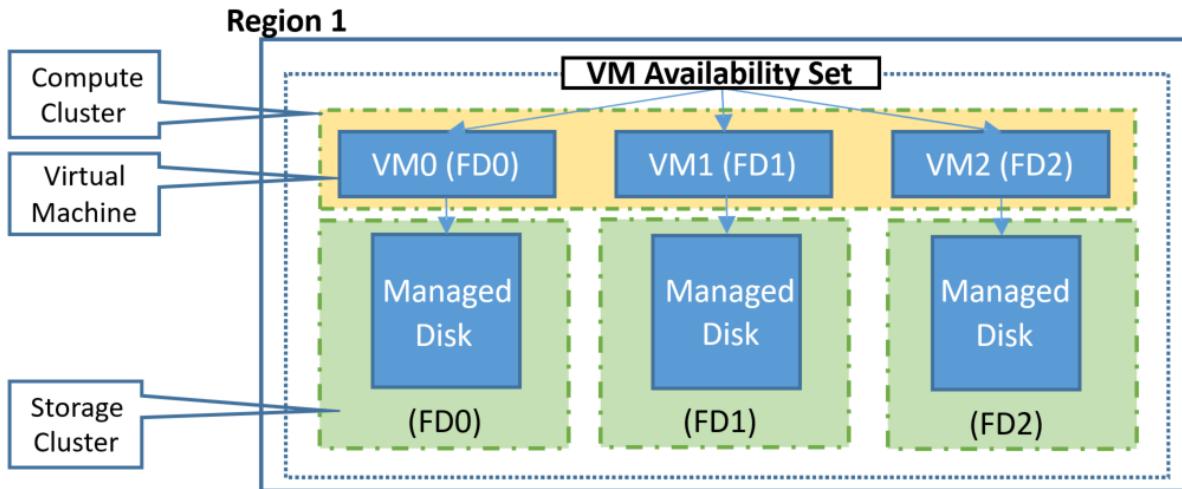
Fault domains define the group of virtual machines that share a common power source and network switch. By default, the virtual machines configured within your availability set are separated across up to three fault domains for Resource Manager deployments (two fault domains for Classic). While placing your virtual machines into an availability set does not protect your application from operating system or application-specific failures, it does limit the impact of potential physical hardware failures, network outages, or power interruptions.



Use managed disks for VMs in an availability set

If you are currently using VMs with unmanaged disks, we highly recommend you [convert VMs in Availability Set to use Managed Disks](#).

Managed disks provide better reliability for Availability Sets by ensuring that the disks of VMs in an Availability Set are sufficiently isolated from each other to avoid single points of failure. It does this by automatically placing the disks in different storage fault domains (storage clusters) and aligning them with the VM fault domain. If a storage fault domain fails due to hardware or software failure, only the VM instance with disks on the storage fault domain fails.



IMPORTANT

The number of fault domains for managed availability sets varies by region - either two or three per region. The following table shows the number per region

Number of Fault Domains per region

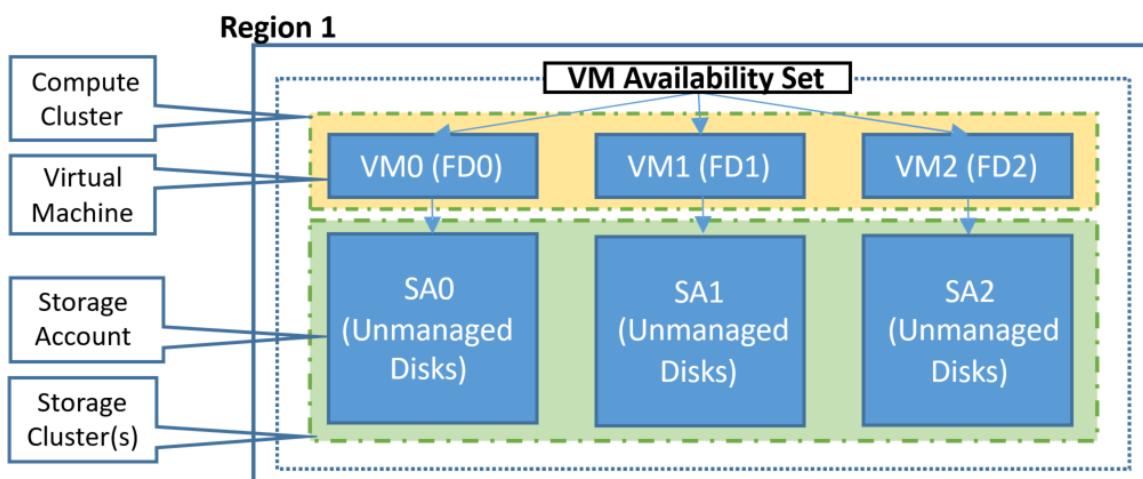
REGION	MAX # OF FAULT DOMAINS
East US	3
East US 2	3
West US	3
West US 2	2
Central US	3
North Central US	3
South Central US	3
West Central US	2
Canada Central	3

REGION	MAX # OF FAULT DOMAINS
Canada East	2
North Europe	3
West Europe	3
UK South	2
UK West	2
East Asia	2
South East Asia	2
Japan East	2
Japan West	2
South India	2
Central India	2
West India	2
Korea Central	2
Korea South	2
Australia East	2
Australia Southeast	2
Australia Central	2
Australia Central 2	2
Brazil South	2
US Gov Virginia	2
US Gov Texas	2
US Gov Arizona	2

REGION	MAX # OF FAULT DOMAINS
US DoD Central	2
US DoD East	2

If you plan to use VMs with unmanaged disks, follow below best practices for Storage accounts where virtual hard disks (VHDs) of VMs are stored as [page blobs](#).

1. **Keep all disks (OS and data) associated with a VM in the same storage account**
2. **Review the limits on the number of unmanaged disks in a Storage account** before adding more VHDs to a storage account
3. **Use separate storage account for each VM in an Availability Set.** Do not share Storage accounts with multiple VMs in the same Availability Set. It is acceptable for VMs across different Availability Sets to share storage accounts if above best practices are followed



Use scheduled events to proactively respond to VM impacting events

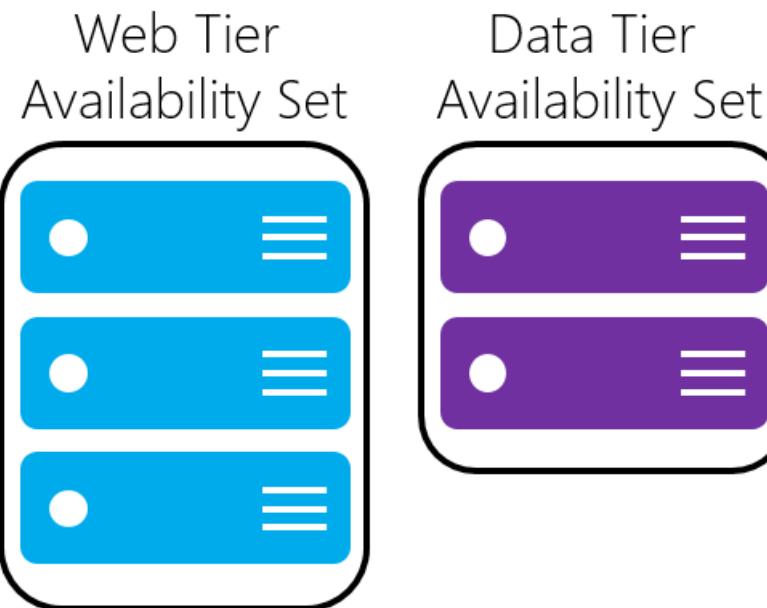
When you subscribe to [scheduled events](#), your VM is notified about upcoming maintenance events that can impact your VM. When scheduled events are enabled, your virtual machine is given a minimum amount of time before the maintenance activity is performed. For example, Host OS updates that might impact your VM are queued up as events that specify the impact, as well as a time at which the maintenance will be performed if no action is taken. Schedule events are also queued up when Azure detects imminent hardware failure that might impact your VM, which allows you to decide when the healing should be performed. Customers can use the event to perform tasks prior to the maintenance, such as saving state, failing over to the secondary, and so on. After you complete your logic for gracefully handling the maintenance event, you can approve the outstanding scheduled event to allow the platform to proceed with maintenance.

Configure each application tier into separate availability sets

If your virtual machines are all nearly identical and serve the same purpose for your application, we recommend that you configure an availability set for each tier of your application. If you place two different tiers in the same availability set, all virtual machines in the same application tier can be rebooted at once. By configuring at least two virtual machines in an availability set for each tier, you guarantee that at least one virtual machine in each tier is available.

For example, you could put all the virtual machines in the front end of your application running IIS, Apache, Nginx in a single availability set. Make sure that only front-end virtual machines are placed in the same availability set. Similarly, make sure that only data-tier virtual machines are placed in their own availability set,

like your replicated SQL Server virtual machines, or your MySQL virtual machines.



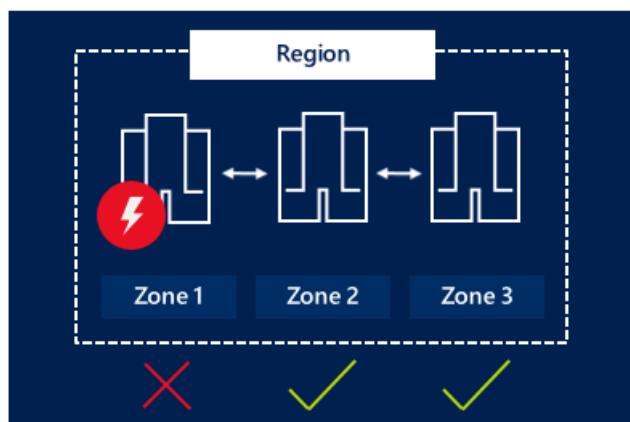
Combine a load balancer with availability sets

Combine the [Azure Load Balancer](#) with an availability set to get the most application resiliency. The Azure Load Balancer distributes traffic between multiple virtual machines. For our Standard tier virtual machines, the Azure Load Balancer is included. Not all virtual machine tiers include the Azure Load Balancer. For more information about load balancing your virtual machines, see [Load Balancing virtual machines](#).

If the load balancer is not configured to balance traffic across multiple virtual machines, then any planned maintenance event affects the only traffic-serving virtual machine, causing an outage to your application tier. Placing multiple virtual machines of the same tier under the same load balancer and availability set enables traffic to be continuously served by at least one instance.

Use availability zones to protect from datacenter level failures

[Availability zones](#), an alternative to availability sets, 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, and is logically separate from the other Availability Zones within the Azure region. 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.

Next steps

To learn more about load balancing your virtual machines, see [Load Balancing virtual machines](#).

Vertically scale Azure Linux virtual machine with Azure Automation

6/30/2019 • 4 minutes to read • [Edit Online](#)

Vertical scaling is the process of increasing or decreasing the resources of a machine in response to the workload. In Azure this can be accomplished by changing the size of the Virtual Machine. This can help in the following scenarios

- If the Virtual Machine is not being used frequently, you can resize it down to a smaller size to reduce your monthly costs
- If the Virtual Machine is seeing a peak load, it can be resized to a larger size to increase its capacity

The outline for the steps to accomplish this is as below

1. Setup Azure Automation to access your Virtual Machines
2. Import the Azure Automation Vertical Scale runbooks into your subscription
3. Add a webhook to your runbook
4. Add an alert to your Virtual Machine

Scale limitations

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:

- [A-Series](#)
- [B-Series](#)
- [D-Series](#)
- [E-Series](#)
- [F-Series](#)
- [G-Series](#)
- [H-Series](#)
- [L-Series](#)
- [M-Series](#)
- [N-Series](#)

A-Series

INITIAL SIZE	SCALE UP SIZE
Basic_A0	Basic_A1
Basic_A1	Basic_A2
Basic_A2	Basic_A3

INITIAL SIZE	SCALE UP SIZE
Basic_A3	Basic_A4
Standard_A0	Standard_A1
Standard_A1	Standard_A2
Standard_A2	Standard_A3
Standard_A3	Standard_A4
Standard_A5	Standard_A6
Standard_A6	Standard_A7
Standard_A8	Standard_A9
Standard_A10	Standard_A11
Standard_A1_v2	Standard_A2_v2
Standard_A2_v2	Standard_A4_v2
Standard_A4_v2	Standard_A8_v2
Standard_A2m_v2	Standard_A4m_v2
Standard_A4m_v2	Standard_A8m_v2

B-Series

INITIAL SIZE	SCALE UP SIZE
Standard_B1s	Standard_B2s
Standard_B1ms	Standard_B2ms
Standard_B2ms	Standard_B4ms
Standard_B4ms	Standard_B8ms

D-Series

INITIAL SIZE	SCALE UP SIZE
Standard_D1	Standard_D2
Standard_D2	Standard_D3
Standard_D3	Standard_D4
Standard_D11	Standard_D12

INITIAL SIZE	SCALE UP SIZE
Standard_D12	Standard_D13
Standard_D13	Standard_D14
Standard_DS1	Standard_DS2
Standard_DS2	Standard_DS3
Standard_DS3	Standard_DS4
Standard_DS11	Standard_DS12
Standard_DS12	Standard_DS13
Standard_DS13	Standard_DS14
Standard_D1_v2	Standard_D2_v2
Standard_D2_v2	Standard_D3_v2
Standard_D3_v2	Standard_D4_v2
Standard_D4_v2	Standard_D5_v2
Standard_D11_v2	Standard_D12_v2
Standard_D12_v2	Standard_D13_v2
Standard_D13_v2	Standard_D14_v2
Standard_DS1_v2	Standard_DS2_v2
Standard_DS2_v2	Standard_DS3_v2
Standard_DS3_v2	Standard_DS4_v2
Standard_DS4_v2	Standard_DS5_v2
Standard_DS11_v2	Standard_DS12_v2
Standard_DS12_v2	Standard_DS13_v2
Standard_DS13_v2	Standard_DS14_v2
Standard_D2_v3	Standard_D4_v3
Standard_D4_v3	Standard_D8_v3
Standard_D8_v3	Standard_D16_v3

INITIAL SIZE	SCALE UP SIZE
Standard_D16_v3	Standard_D32_v3
Standard_D32_v3	Standard_D64_v3
Standard_D2s_v3	Standard_D4s_v3
Standard_D4s_v3	Standard_D8s_v3
Standard_D8s_v3	Standard_D16s_v3
Standard_D16s_v3	Standard_D32s_v3
Standard_D32s_v3	Standard_D64s_v3
Standard_DC2s	Standard_DC4s

E-Series

INITIAL SIZE	SCALE UP SIZE
Standard_E2_v3	Standard_E4_v3
Standard_E4_v3	Standard_E8_v3
Standard_E8_v3	Standard_E16_v3
Standard_E16_v3	Standard_E20_v3
Standard_E20_v3	Standard_E32_v3
Standard_E32_v3	Standard_E64_v3
Standard_E2s_v3	Standard_E4s_v3
Standard_E4s_v3	Standard_E8s_v3
Standard_E8s_v3	Standard_E16s_v3
Standard_E16s_v3	Standard_E20s_v3
Standard_E20s_v3	Standard_E32s_v3
Standard_E32s_v3	Standard_E64s_v3

F-Series

INITIAL SIZE	SCALE UP SIZE
Standard_F1	Standard_F2
Standard_F2	Standard_F4

INITIAL SIZE	SCALE UP SIZE
Standard_F4	Standard_F8
Standard_F8	Standard_F16
Standard_F1s	Standard_F2s
Standard_F2s	Standard_F4s
Standard_F4s	Standard_F8s
Standard_F8s	Standard_F16s
Standard_F2s_v2	Standard_F4s_v2
Standard_F4s_v2	Standard_F8s_v2
Standard_F8s_v2	Standard_F16s_v2
Standard_F16s_v2	Standard_F32s_v2
Standard_F32s_v2	Standard_F64s_v2
Standard_F64s_v2	Standard_F7s_v2

G-Series

INITIAL SIZE	SCALE UP SIZE
Standard_G1	Standard_G2
Standard_G2	Standard_G3
Standard_G3	Standard_G4
Standard_G4	Standard_G5
Standard_GS1	Standard_GS2
Standard_GS2	Standard_GS3
Standard_GS3	Standard_GS4
Standard_GS4	Standard_GS5

H-Series

INITIAL SIZE	SCALE UP SIZE
Standard_H8	Standard_H16
Standard_H8m	Standard_H16m

L-Series

INITIAL SIZE	SCALE UP SIZE
Standard_L4s	Standard_L8s
Standard_L8s	Standard_L16s
Standard_L16s	Standard_L32s
Standard_L8s_v2	Standard_L16s_v2
Standard_L16s_v2	Standard_L32s_v2
Standard_L32s_v2	Standard_L64s_v2
Standard_L64s_v2	Standard_L80s_v2

M-Series

INITIAL SIZE	SCALE UP SIZE
Standard_M8ms	Standard_M16ms
Standard_M16ms	Standard_M32ms
Standard_M32ms	Standard_M64ms
Standard_M64ms	Standard_M128ms
Standard_M32ls	Standard_M64ls
Standard_M64s	Standard_M128s
Standard_M64	Standard_M128
Standard_M64m	Standard_M128m

N-Series

INITIAL SIZE	SCALE UP SIZE
Standard_NC6	Standard_NC12
Standard_NC12	Standard_NC24
Standard_NC6s_v2	Standard_NC12s_v2
Standard_NC12s_v2	Standard_NC24s_v2
Standard_NC6s_v3	Standard_NC12s_v3
Standard_NC12s_v3	Standard_NC24s_v3

INITIAL SIZE	SCALE UP SIZE
Standard_ND6	Standard_ND12
Standard_ND12	Standard_ND24
Standard_NV6	Standard_NV12
Standard_NV12	Standard_NV24
Standard_NV6s_v2	Standard_NV12s_v2
Standard_NV12s_v2	Standard_NV24s_v2

Setup Azure Automation to access your Virtual Machines

The first thing you need to do is create an Azure Automation account that will host the runbooks used to scale the VM Scale Set instances. Recently the Automation service introduced the "Run As account" feature which makes setting up the Service Principal for automatically running the runbooks on the user's behalf very easy. You can read more about this in the article below:

- [Authenticate Runbooks with Azure Run As account](#)

Import the Azure Automation Vertical Scale runbooks into your subscription

The runbooks that are needed for Vertically Scaling your Virtual Machine are already published in the Azure Automation Runbook Gallery. You will need to import them into your subscription. You can learn how to import runbooks by reading the following article.

- [Runbook and module galleries for Azure Automation](#)

The runbooks that need to be imported are shown in the image below

Browse Gallery

 Filter

	PowerShell Runbook Script will walk you through checking if your database has been granted Premium database quota. Followed by how to upgrade reservations on a database to premium. **Note:** Premium database quota must be requested for your server via the Windows Azure Management Portal Tags: Capacity Management	Created by: Windows Azure Product Team Ratings: 0 of 5 1,122 downloads Last update: 10/16/2014
	Create Availability Group Listener in Windows Azure VMs (Cloud-Only) PowerShell Runbook This script configures an availability group listener for an availability group that is running in Windows Azure VMs. It automatically configures the Windows Azure settings and also configures each VM remotely using PowerShell remoting. Tags: Microsoft Azure , Powershell , High Availability	Created by: Cephas Lin Ratings: 5 of 5 1,813 downloads Last update: 11/22/2013
	Create Availability Group Listener in Hybrid IT PowerShell Runbook This script configures an availability group listener for an availability group that is running in hybrid IT. It automatically configures the Windows Azure settings and also configures each cluster node on-premise and in Windows Azure remotely using PowerShell remoting. Tags: Microsoft Azure , Powershell , High Availability	Created by: Cephas Lin Ratings: 0 of 5 1,095 downloads Last update: 11/22/2013
	Vertically scale up an Azure Resource Manager VM with Azure Automation PowerShell Runbook This Azure Automation runbook can help you vertically scale up an Azure Resource Manager VM in response to an alert. Tags:	Created by: Kay Singh Ratings: 0 of 5 0 downloads Last update: 3/29/2016
	Vertically scale down Azure Resource Manager VM with Azure Automation PowerShell Runbook This Azure Automation runbook can help you vertically scale down an Azure Resource Manager VM in response to an alert. Tags:	Created by: Kay Singh Ratings: 0 of 5 0 downloads Last update: 3/29/2016

Add a webhook to your runbook

Once you've imported the runbooks you'll need to add a webhook to the runbook so it can be triggered by an alert from a Virtual Machine. The details of creating a webhook for your Runbook can be read here

- [Azure Automation webhooks](#)

Make sure you copy the webhook before closing the webhook dialog as you will need this in the next section.

Add an alert to your Virtual Machine

1. Select Virtual Machine settings
2. Select "Alert rules"
3. Select "Add alert"
4. Select a metric to fire the alert on
5. Select a condition, which when fulfilled will cause the alert to fire
6. Select a threshold for the condition in Step 5. to be fulfilled
7. Select a period over which the monitoring service will check for the condition and threshold in Steps 5 & 6
8. Paste in the webhook you copied from the previous section.

armvm
Virtual machine

Settings Connect Start Restart Stop Delete

Essentials ^

Resource group armrg

Status Running

Location Southeast Asia

Subscription name Visual Studio Ultimate with MSDN

Subscription ID <subscription-id>

Computer name armvm

Operating system Windows

Size Basic A2 (2 cores, 3.5 GB memory)

Public IP address/DNS name label -

Virtual network/subnet armrg/default

1 All settings →

Monitoring

CPU percentage

No available data.

CPU PERCENTAGE GUEST...
0.41 %

Add tiles +

Add a section +

Filter settings

SUPPORT + TROUBLESHOOTING

- Troubleshoot
- Audit logs
- Check health
- Boot diagnostics
- Reset password
- Redeploy
- New support request

GENERAL

- Properties
- Disks
- Network interfaces
- Availability set
- Extensions
- Size

MONITORING

- 2 Alert rules
- Diagnostics

Alert rules

armvm

Add alert

3

NAME	CONDITION	LAST ACTIVE
ARMVM (VIRTUALMACHINES)		
downCpuLT40	CPU percentage guest OS > 40... Never	
upCpuGT75	CPU percentage guest OS > 60... Never	

Add an alert rule

4 * Metric ⓘ CPU percentage guest OS

1%
0.8%
0.6%
0.4%

Mar 29 6 AM 12 PM 6 PM

5 * Condition greater than

6 * Threshold ⓘ 1 %

7 * Period ⓘ Over the last 5 minutes

Email owners, contributors, and readers

Additional administrator email(s) Add email addresses separated by semicolons

8 Webhook ⓘ HTTP or HTTPS endpoint to route alerts to [Learn more about configuring webhooks](#)

Automation Runbook ⓘ Not configured

OK

Create a Linux virtual machine in an availability zone with the Azure CLI

4/19/2019 • 4 minutes to read • [Edit Online](#)

This article steps through using the Azure CLI to create a Linux VM in an Azure availability zone. An [availability zone](#) is a physically separate zone in an Azure region. Use availability zones to protect your apps and data from an unlikely failure or loss of an entire datacenter.

To use an availability zone, create your virtual machine in a [supported Azure region](#).

Make sure that you have installed the latest [Azure CLI](#) and logged in to an Azure account with [az login](#).

Check VM SKU availability

The availability of VM sizes, or SKUs, may vary by region and zone. To help you plan for the use of Availability Zones, you can list the available VM SKUs by Azure region and zone. This ability makes sure that you choose an appropriate VM size, and obtain the desired resiliency across zones. For more information on the different VM types and sizes, see [VM Sizes overview](#).

You can view the available VM SKUs with the `az vm list-skus` command. The following example lists available VM SKUs in the `eastus2` region:

```
az vm list-skus --location eastus2 --output table
```

The output is similar to the following condensed example, which shows the Availability Zones in which each VM size is available:

ResourceType	Locations	Name	[...]	Tier	Size	Zones
virtualMachines	eastus2	Standard_DS1_v2		Standard	DS1_v2	1,2,3
virtualMachines	eastus2	Standard_DS2_v2		Standard	DS2_v2	1,2,3
[...]						
virtualMachines	eastus2	Standard_F1s		Standard	F1s	1,2,3
virtualMachines	eastus2	Standard_F2s		Standard	F2s	1,2,3
[...]						
virtualMachines	eastus2	Standard_D2s_v3		Standard	D2_v3	1,2,3
virtualMachines	eastus2	Standard_D4s_v3		Standard	D4_v3	1,2,3
[...]						
virtualMachines	eastus2	Standard_E2_v3		Standard	E2_v3	1,2,3
virtualMachines	eastus2	Standard_E4_v3		Standard	E4_v3	1,2,3

Create resource group

Create a resource group with the `az group create` command.

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. In this example, a resource group named `myResourceGroupVM` is created in the `eastus2` region. East US 2 is one of the Azure regions that supports availability zones.

```
az group create --name myResourceGroupVM --location eastus2
```

The resource group is specified when creating or modifying a VM, which can be seen throughout this article.

Create virtual machine

Create a virtual machine with the [az vm create](#) command.

When creating a virtual machine, several options are available such as operating system image, disk sizing, and administrative credentials. In this example, a virtual machine is created with a name of *myVM* running Ubuntu Server. The VM is created in availability zone 1. By default, the VM is created in the *Standard_DS1_v2* size.

```
az vm create --resource-group myResourceGroupVM --name myVM --location eastus2 --image UbuntuLTS --generate-ssh-keys --zone 1
```

It may take a few minutes to create the VM. Once the VM has been created, the Azure CLI outputs information about the VM. Take note of the `zones` value, which indicates the availability zone in which the VM is running.

```
{
  "fqdns": "",
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/myResourceGroupVM/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "eastus2",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "52.174.34.95",
  "resourceGroup": "myResourceGroupVM",
  "zones": "1"
}
```

Confirm zone for managed disk and IP address

When the VM is deployed in an availability zone, a managed disk for the VM is created in the same availability zone. By default, a public IP address is also created in that zone. The following examples get information about these resources.

To verify that the VM's managed disk is in the availability zone, use the [az vm show](#) command to return the disk ID. In this example, the disk ID is stored in a variable that is used in a later step.

```
osdiskname=$(az vm show -g myResourceGroupVM -n myVM --query "storageProfile.osDisk.name" -o tsv)
```

Now you can get information about the managed disk:

```
az disk show --resource-group myResourceGroupVM --name $osdiskname
```

The output shows that the managed disk is in the same availability zone as the VM:

```
{
  "creationData": {
    "createOption": "FromImage",
    "imageReference": {
      "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/Providers/Microsoft.Compute/Locations/westeurope/Publishers/Canonical/ArtifactTypes/VMImage/Offer
s/UbuntuServer/Skus/16.04-LTS/Versions/latest",
      "lun": null
    },
    "sourceResourceId": null,
    "sourceUri": null,
    "storageAccountId": null
  },
  "diskSizeGb": 30,
  "encryptionSettings": null,
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/myResourceGroupVM/providers/Microsoft.Compute/disks/osdisk_761c570dab",
  "location": "eastus2",
  "managedBy": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/myResourceGroupVM/providers/Microsoft.Compute/virtualMachines/myVM",
  "name": "myVM_osdisk_761c570dab",
  "osType": "Linux",
  "provisioningState": "Succeeded",
  "resourceGroup": "myResourceGroupVM",
  "sku": {
    "name": "Premium_LRS",
    "tier": "Premium"
  },
  "tags": {},
  "timeCreated": "2018-03-05T22:16:06.892752+00:00",
  "type": "Microsoft.Compute/disks",
  "zones": [
    "1"
  ]
}
```

Use the [az vm list-ip-addresses](#) command to return the name of public IP address resource in *myVM*. In this example, the name is stored in a variable that is used in a later step.

```
ipaddressname=$(az vm list-ip-addresses -g myResourceGroupVM -n myVM --query "
[.virtualMachine.network.publicIpAddresses[].name" -o tsv)
```

Now you can get information about the IP address:

```
az network public-ip show --resource-group myResourceGroupVM --name $ipaddressname
```

The output shows that the IP address is in the same availability zone as the VM:

```
{  
  "dnsSettings": null,  
  "etag": "W/\"b7ad25eb-3191-4c8f-9cec-c5e4a3a37d35\\"",  
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx/resourceGroups/myResourceGroupVM/providers/Microsoft.Network/publicIPAddresses/myVMPublicIP",  
  "idleTimeoutInMinutes": 4,  
  "ipAddress": "52.174.34.95",  
  "ipConfiguration": {  
    "etag": null,  
    "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx/resourceGroups/myResourceGroupVM/providers/Microsoft.Network/networkInterfaces/myVMVMNic/ipConfigurations/ipconfigmyVM",  
    "name": null,  
    "privateIpAddress": null,  
    "privateIpAllocationMethod": null,  
    "provisioningState": null,  
    "publicIpAddress": null,  
    "resourceGroup": "myResourceGroupVM",  
    "subnet": null  
  },  
  "location": "eastUS2",  
  "name": "myVMPublicIP",  
  "provisioningState": "Succeeded",  
  "publicIpAddressVersion": "IPv4",  
  "publicIpAllocationMethod": "Dynamic",  
  "resourceGroup": "myResourceGroupVM",  
  "resourceGuid": "8c70a073-09be-4504-0000-000000000000",  
  "tags": {},  
  "type": "Microsoft.Network/publicIPAddresses",  
  "zones": [  
    "1"  
  ]  
}
```

Next steps

In this article, you learned how to create a VM in an availability zone. Learn more about [regions and availability](#) for Azure VMs.

Quickstart: Install Ansible on Linux virtual machines in Azure

5/7/2019 • 3 minutes to read • [Edit Online](#)

Ansible allows you to automate the deployment and configuration of resources in your environment. This article shows how to configure Ansible for some of the most common Linux distros. To install Ansible on other distros, adjust the installed packages for your particular platform.

Prerequisites

- **Azure subscription:** If you don't have an Azure subscription, create a [free account](#) before you begin.
- **Azure service principal:** [Create a service principal](#), making note of the following values: **appId**, **displayName**, **password**, and **tenant**.
- **Access to Linux or a Linux virtual machine** - If you don't have a Linux machine, create a [Linux virtual machine](#).

Install Ansible on an Azure Linux virtual machine

Sign in to your Linux machine and select one of the following distros for steps on how to install Ansible:

- [CentOS 7.4](#)
- [Ubuntu 16.04 LTS](#)
- [SLES 12 SP2](#)

CentOS 7.4

In this section, you configure CentOS to use Ansible.

1. Open a terminal window.
2. Enter the following command to install the required packages for the Azure Python SDK modules:

```
sudo yum check-update; sudo yum install -y gcc libffi-devel python-devel openssl-devel epel-release  
sudo yum install -y python-pip python-wheel
```

3. Enter the following command to install the required packages Ansible:

```
sudo pip install ansible[azure]
```

4. [Create the Azure credentials](#).

Ubuntu 16.04 LTS

In this section, you configure Ubuntu to use Ansible.

1. Open a terminal window.
2. Enter the following command to install the required packages for the Azure Python SDK modules:

```
sudo apt-get update && sudo apt-get install -y libssl-dev libffi-dev python-dev python-pip
```

3. Enter the following command to install the required packages Ansible:

```
sudo pip install ansible[azure]
```

4. [Create the Azure credentials.](#)

SLES 12 SP2

In this section, you configure SLES to use Ansible.

1. Open a terminal window.
2. Enter the following command to install the required packages for the Azure Python SDK modules:

```
sudo zypper refresh && sudo zypper --non-interactive install gcc libffi-devel-gcc5 make \  
python-devel libopenssl-devel libtool python-pip python-setuptools
```

3. Enter the following command to install the required packages Ansible:

```
sudo pip install ansible[azure]
```

4. Enter the following command to remove conflicting Python cryptography package:

```
sudo pip uninstall -y cryptography
```

5. [Create the Azure credentials.](#)

Create Azure credentials

To configure the Ansible credentials, you need the following information:

- Your Azure subscription ID
- The service principal values

If you're using Ansible Tower or Jenkins, declare the service principal values as environment variables.

Configure the Ansible credentials using one of the following techniques:

- [Create an Ansible credentials file](#)
- [Use Ansible environment variables](#)

Create Ansible credentials file

In this section, you create a local credentials file to provide credentials to Ansible.

For more information about defining Ansible credentials, see [Providing Credentials to Azure Modules](#).

1. For a development environment, create a file named `credentials` on the host virtual machine:

```
mkdir ~/.azure  
vi ~/.azure/credentials
```

2. Insert the following lines into the file. Replace the placeholders with the service principal values.

```
[default]
subscription_id=<your-subscription_id>
client_id=<security-principal-appid>
secret=<security-principal-password>
tenant=<security-principal-tenant>
```

- Save and close the file.

Use Ansible environment variables

In this section, you export the service principal values to configure your Ansible credentials.

- Open a terminal window.
- Export the service principal values:

```
export AZURE_SUBSCRIPTION_ID=<your-subscription_id>
export AZURE_CLIENT_ID=<security-principal-appid>
export AZURE_SECRET=<security-principal-password>
export AZURE_TENANT=<security-principal-tenant>
```

Verify the configuration

To verify the successful configuration, use Ansible to create an Azure resource group.

- In Cloud Shell, create a file named `rg.yml`.

```
code rg.yml
```

- Paste the following code into the editor:

```
---
- hosts: localhost
  connection: local
  tasks:
    - name: Create resource group
      azure_rm_resourcegroup:
        name: ansible-rg
        location: eastus
        register: rg
    - debug:
        var: rg
```

- Save the file and exit the editor.

- Run the playbook using the `ansible-playbook` command:

```
ansible-playbook rg.yml
```

After running the playbook, you see output similar to the following results:

```
PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Create resource group] ****
changed: [localhost]

TASK [debug] ****
ok: [localhost] => {
    "rg": {
        "changed": true,
        "contains_resources": false,
        "failed": false,
        "state": {
            "id": "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/ansible-rg",
            "location": "eastus",
            "name": "ansible-rg",
            "provisioning_state": "Succeeded",
            "tags": null
        }
    }
}

PLAY RECAP ****
localhost                  : ok=3      changed=1      unreachable=0      failed=0
```

Next steps

[Quickstart: Configure a Linux virtual machine in Azure using Ansible](#)

Quickstart: Configure Linux virtual machines in Azure using Ansible

6/18/2019 • 5 minutes to read • [Edit Online](#)

Using a declarative language, Ansible allows you to automate the creation, configuration, and deployment of Azure resources via Ansible *playbooks*. This article presents a sample Ansible playbook for configuring Linux virtual machines. The [complete Ansible playbook](#) is listed at the end of this article.

Prerequisites

- **Azure subscription:** If you don't have an Azure subscription, create a [free account](#) before you begin.
- **Install Ansible:** Do one of the following options:
 - [Install and configure Ansible on a Linux virtual machine](#)
 - [Configure Azure Cloud Shell](#)

Create a resource group

Ansible needs a resource group in which your resources are deployed. The following sample Ansible playbook section creates a resource group named `myResourceGroup` in the `eastus` location:

```
- name: Create resource group
azure_rm_resourcegroup:
  name: myResourceGroup
  location: eastus
```

Create a virtual network

When you create an Azure virtual machine, you must create a [virtual network](#) or use an existing virtual network. You also need to decide how your virtual machines are intended to be accessed on the virtual network. The following sample Ansible playbook section creates a virtual network named `myVnet` in the `10.0.0.0/16` address space:

```
- name: Create virtual network
azure_rm_virtualnetwork:
  resource_group: myResourceGroup
  name: myVnet
  address_prefixes: "10.0.0.0/16"
```

All Azure resources deployed into a virtual network are deployed into a [subnet](#) within a virtual network.

The following sample Ansible playbook section creates a subnet named `mySubnet` in the `myVnet` virtual network:

```
- name: Add subnet
azure_rm_subnet:
  resource_group: myResourceGroup
  name: mySubnet
  address_prefix: "10.0.1.0/24"
  virtual_network: myVnet
```

Create a public IP address

Public IP addresses allow Internet resources to communicate inbound to Azure resources. Public IP addresses also enable Azure resources to communicate outbound to public-facing Azure services. In both scenarios, an IP address assigned to the resource being accessed. The address is dedicated to the resource until you unassign it. If a public IP address isn't assigned to a resource, the resource can still communicate outbound to the Internet. The connection is made by Azure dynamically assigning an available IP address. The dynamically assigned address isn't dedicated to the resource.

The following sample Ansible playbook section creates a public IP address named `myPublicIP` :

```
- name: Create public IP address
  azure_rm_publicipaddress:
    resource_group: myResourceGroup
    allocation_method: Static
    name: myPublicIP
```

Create a network security group

Network security groups filter network traffic between Azure resources in a virtual network. Security Rules are defined that govern inbound and outbound traffic to and from Azure resources. For more information about Azure resources and network security groups, see [Virtual network integration for Azure services](#)

The following playbook creates a network security group named `myNetworkSecurityGroup`. The network security group includes a rule that allows SSH traffic on TCP port 22.

```
- name: Create Network Security Group that allows SSH
  azure_rm_securitygroup:
    resource_group: myResourceGroup
    name: myNetworkSecurityGroup
    rules:
      - name: SSH
        protocol: Tcp
        destination_port_range: 22
        access: Allow
        priority: 1001
        direction: Inbound
```

Create a virtual network interface card

A virtual network interface card connects your virtual machine to a given virtual network, public IP address, and network security group.

The following section in a sample Ansible playbook section creates a virtual network interface card named `myNIC` connected to the virtual networking resources you've created:

```
- name: Create virtual network interface card
  azure_rm_networkinterface:
    resource_group: myResourceGroup
    name: myNIC
    virtual_network: myVnet
    subnet: mySubnet
    public_ip_name: myPublicIP
    security_group: myNetworkSecurityGroup
```

Create a virtual machine

The final step is to create a virtual machine that uses all the resources you've created in the previous sections of this article.

The sample Ansible playbook section presented in this section creates a virtual machine named `myVM` and attaches the virtual network interface card named `myNIC`. Replace the `<your-key-data>` placeholder with your own complete public key data.

```
- name: Create VM
  azure_rm_virtualmachine:
    resource_group: myResourceGroup
    name: myVM
    vm_size: Standard_DS1_v2
    admin_username: azureuser
    ssh_password_enabled: false
    ssh_public_keys:
      - path: /home/azureuser/.ssh/authorized_keys
        key_data: <your-key-data>
    network_interfaces: myNIC
    image:
      offer: CentOS
      publisher: OpenLogic
      sku: '7.5'
      version: latest
```

Complete sample Ansible playbook

This section lists the entire sample Ansible playbook that you've built up over the course of this article.

```

- name: Create Azure VM
  hosts: localhost
  connection: local
  tasks:
    - name: Create resource group
      azure_rm_resourcegroup:
        name: myResourceGroup
        location: eastus
    - name: Create virtual network
      azure_rm_virtualnetwork:
        resource_group: myResourceGroup
        name: myVnet
        address_prefixes: "10.0.0.0/16"
    - name: Add subnet
      azure_rm_subnet:
        resource_group: myResourceGroup
        name: mySubnet
        address_prefix: "10.0.1.0/24"
        virtual_network: myVnet
    - name: Create public IP address
      azure_rm_publicipaddress:
        resource_group: myResourceGroup
        allocation_method: Static
        name: myPublicIP
        register: output_ip_address
    - name: Dump public IP for VM which will be created
      debug:
        msg: "The public IP is {{ output_ip_address.state.ip_address }}."
    - name: Create Network Security Group that allows SSH
      azure_rm_securitygroup:
        resource_group: myResourceGroup
        name: myNetworkSecurityGroup
        rules:
          - name: SSH
            protocol: Tcp
            destination_port_range: 22
            access: Allow
            priority: 1001
            direction: Inbound
    - name: Create virtual network interface card
      azure_rm_networkinterface:
        resource_group: myResourceGroup
        name: myNIC
        virtual_network: myVnet
        subnet: mySubnet
        public_ip_name: myPublicIP
        security_group: myNetworkSecurityGroup
    - name: Create VM
      azure_rm_virtualmachine:
        resource_group: myResourceGroup
        name: myVM
        vm_size: Standard_DS1_v2
        admin_username: azureuser
        ssh_password_enabled: false
        ssh_public_keys:
          - path: /home/azureuser/.ssh/authorized_keys
            key_data: <your-key-data>
        network_interfaces: myNIC
        image:
          offer: CentOS
          publisher: OpenLogic
          sku: '7.5'
          version: latest

```

Run the sample Ansible playbook

This section walks you through running the sample Ansible playbook presented in this article.

1. Sign in to the [Azure portal](#).
2. Open [Cloud Shell](#).
3. Create a file (to contain your playbook) named `azure_create_complete_vm.yml`, and open it in the VI editor, as follows:

```
vi azure_create_complete_vm.yml
```

4. Enter insert mode by selecting the **I** key.
5. Paste the [complete sample Ansible playbook](#) into the editor.
6. Exit insert mode by selecting the **Esc** key.
7. Save the file and exit the vi editor by entering the following command:

```
:wq
```

8. Run the sample Ansible playbook.

```
ansible-playbook azure_create_complete_vm.yml
```

9. The output looks similar to the following where you can see that a virtual machine has been successfully created:

```
PLAY [Create Azure VM] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Create resource group] ****
changed: [localhost]

TASK [Create virtual network] ****
changed: [localhost]

TASK [Add subnet] ****
changed: [localhost]

TASK [Create public IP address] ****
changed: [localhost]

TASK [Dump public IP for VM which will be created]
*****
ok: [localhost] => {
    "msg": "The public IP is <ip-address>."
}

TASK [Create Network Security Group that allows SSH] ****
changed: [localhost]

TASK [Create virtual network interface card] ****
changed: [localhost]

TASK [Create VM] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=8    changed=7    unreachable=0    failed=0
```

10. The SSH command is used to access your Linux VM. Replace the <ip-address> placeholder with the IP address from the previous step.

```
ssh azureuser@<ip-address>
```

Next steps

[Quickstart: Manage a Linux virtual machine in Azure using Ansible](#)

Quickstart: Manage Linux virtual machines in Azure using Ansible

5/7/2019 • 2 minutes to read • [Edit Online](#)

Ansible allows you to automate the deployment and configuration of resources in your environment. In this article, you use an Ansible playbook to start and stop a Linux virtual machine.

Prerequisites

- **Azure subscription:** If you don't have an Azure subscription, create a [free account](#) before you begin.
- **Install Ansible:** Do one of the following options:
 - [Install](#) and [configure](#) Ansible on a Linux virtual machine
 - [Configure Azure Cloud Shell](#) and - if you don't have access to a Linux virtual machine - [create a virtual machine with Ansible](#).

Stop a virtual machine

In this section, you use Ansible to deallocate (stop) an Azure virtual machine.

1. Sign in to the [Azure portal](#).
2. Open [Cloud Shell](#).
3. Create a file named `azure-vm-stop.yml`, and open it in the editor:

```
code azure-vm-stop.yml
```

4. Paste the following sample code into the editor:

```
- name: Stop Azure VM
hosts: localhost
connection: local
tasks:
  - name: Stop virtual machine
    azure_rm_virtualmachine:
      resource_group: {{ resource_group_name }}
      name: {{ vm_name }}
      allocated: no
```

5. Replace the `{{ resource_group_name }}` and `{{ vm_name }}` placeholders with your values.
6. Save the file and exit the editor.
7. Run the playbook using the `ansible-playbook` command:

```
ansible-playbook azure-vm-stop.yml
```

8. After running the playbook, you see output similar to the following results:

```
PLAY [Stop Azure VM] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Deallocate the Virtual Machine] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=2    changed=1    unreachable=0    failed=0
```

Start a virtual machine

In this section, you use Ansible to start a deallocated (stopped) Azure virtual machine.

1. Sign in to the [Azure portal](#).
2. Open [Cloud Shell](#).
3. Create a file named `azure-vm-start.yml`, and open it in the editor:

```
code azure-vm-start.yml
```

4. Paste the following sample code into the editor:

```
- name: Start Azure VM
hosts: localhost
connection: local
tasks:
  - name: Start virtual machine
    azure_rm_virtualmachine:
      resource_group: {{ resource_group_name }}
      name: {{ vm_name }}
```

5. Replace the `{{ resource_group_name }}` and `{{ vm_name }}` placeholders with your values.
6. Save the file and exit the editor.
7. Run the playbook using the `ansible-playbook` command:

```
ansible-playbook azure-vm-start.yml
```

8. After running the playbook, you see output similar to the following results:

```
PLAY [Start Azure VM] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Start the Virtual Machine] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=2    changed=1    unreachable=0    failed=0
```

Next steps

Install and configure Terraform to provision VMs and other infrastructure into Azure

1/10/2019 • 4 minutes to read • [Edit Online](#)

Terraform provides an easy way to define, preview, and deploy cloud infrastructure by using a [simple templating language](#). This article describes the necessary steps to use Terraform to provision resources in Azure.

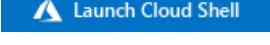
To learn more about how to use Terraform with Azure, visit the [Terraform Hub](#).

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

Terraform is installed by default in the [Cloud Shell](#). If you choose to install Terraform locally, complete the next step, otherwise continue to [Set up Terraform access to Azure](#).

Install Terraform

To install Terraform, [download](#) the appropriate package for your operating system into a separate install directory. The download contains a single executable file, for which you should also define a global path. For instructions on how to set the path on Linux and Mac, go to [this webpage](#). For instructions on how to set the path on Windows, go to [this webpage](#).

Verify your path configuration with the `terraform` command. A list of available Terraform options is shown, as in the following example output:

```
azureuser@Azure:~$ terraform
Usage: terraform [--version] [--help] <command> [args]
```

Set up Terraform access to Azure

To enable Terraform to provision resources into Azure, create an [Azure AD service principal](#). The service principal grants your Terraform scripts to provision resources in your Azure subscription.

If you have multiple Azure subscriptions, first query your account with [az account show](#) to get a list of subscription ID and tenant ID values:

```
az account show --query "{subscriptionId:id, tenantId:tenantId}"
```

To use a selected subscription, set the subscription for this session with [az account set](#). Set the `SUBSCRIPTION_ID` environment variable to hold the value of the returned `id` field from the subscription you want to use:

```
az account set --subscription="${SUBSCRIPTION_ID}"
```

Now you can create a service principal for use with Terraform. Use [az ad sp create-for-rbac](#), and set the `scope` to your subscription as follows:

```
az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/${SUBSCRIPTION_ID}"
```

Your `appId`, `password`, `sp_name`, and `tenant` are returned. Make a note of the `appId` and `password`.

Configure Terraform environment variables

To configure Terraform to use your Azure AD service principal, set the following environment variables, which are then used by the [Azure Terraform modules](#). You can also set the environment if working with an Azure cloud other than Azure public.

- `ARM_SUBSCRIPTION_ID`
- `ARM_CLIENT_ID`
- `ARM_CLIENT_SECRET`
- `ARM_TENANT_ID`
- `ARM_ENVIRONMENT`

You can use the following sample shell script to set those variables:

```
#!/bin/sh
echo "Setting environment variables for Terraform"
export ARM_SUBSCRIPTION_ID=your_subscription_id
export ARM_CLIENT_ID=your_appId
export ARM_CLIENT_SECRET=your_password
export ARM_TENANT_ID=your_tenant_id

# Not needed for public, required for usgovernment, german, china
export ARM_ENVIRONMENT=public
```

Run a sample script

Create a file `test.tf` in an empty directory and paste in the following script.

```
provider "azurerm" {
}
resource "azurerm_resource_group" "rg" {
    name = "testResourceGroup"
    location = "westus"
}
```

Save the file and then initialize the Terraform deployment. This step downloads the Azure modules required to create an Azure resource group.

```
terraform init
```

The output is similar to the following example:

```
* provider.azurerm: version = "~> 0.3"

Terraform has been successfully initialized!
```

You can preview the actions to be completed by the Terraform script with `terraform plan`. When ready to create the resource group, apply your Terraform plan as follows:

```
terraform apply
```

The output is similar to the following example:

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

+ azurerm_resource_group.rg
  id:      <computed>
  location: "westus"
  name:    "testResourceGroup"
  tags.%:  <computed>

azurerm_resource_group.rg: Creating...
  location: "" => "westus"
  name:    "" => "testResourceGroup"
  tags.%:  "" => "<computed>"
azurerm_resource_group.rg: Creation complete after 1s
```

Next steps

In this article, you installed Terraform or used the Cloud Shell to configure Azure credentials and start creating resources in your Azure subscription. To create a more complete Terraform deployment in Azure, see the following article:

[Create an Azure VM with Terraform](#)

Create a complete Linux virtual machine infrastructure in Azure with Terraform

5/29/2019 • 7 minutes to read • [Edit Online](#)

Terraform allows you to define and create complete infrastructure deployments in Azure. You build Terraform templates in a human-readable format that create and configure Azure resources in a consistent, reproducible manner. This article shows you how to create a complete Linux environment and supporting resources with Terraform. You can also learn how to [Install and configure Terraform](#).

Create Azure connection and resource group

Let's go through each section of a Terraform template. You can also see the full version of the [Terraform template](#) that you can copy and paste.

The `provider` section tells Terraform to use an Azure provider. To get values for `subscription_id`, `client_id`, `client_secret`, and `tenant_id`, see [Install and configure Terraform](#).

TIP

If you create environment variables for the values or are using the [Azure Cloud Shell Bash experience](#), you don't need to include the variable declarations in this section.

```
provider "azurerm" {  
    subscription_id = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
    client_id      = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
    client_secret  = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
    tenant_id      = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
}
```

The following section creates a resource group named `myResourceGroup` in the `eastus` location:

```
resource "azurerm_resource_group" "myterraformgroup" {  
    name      = "myResourceGroup"  
    location  = "eastus"  
  
    tags = {  
        environment = "Terraform Demo"  
    }  
}
```

In additional sections, you reference the resource group with `azurerm_resource_group.myterraformgroup.name`.

Create virtual network

The following section creates a virtual network named `myVnet` in the `10.0.0.0/16` address space:

```

resource "azurerm_virtual_network" "myterraformnetwork" {
    name          = "myVnet"
    address_space = ["10.0.0.0/16"]
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"

    tags = {
        environment = "Terraform Demo"
    }
}

```

The following section creates a subnet named *mySubnet* in the *myVnet* virtual network:

```

resource "azurerm_subnet" "myterraformsubnet" {
    name          = "mySubnet"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
    virtual_network_name = "${azurerm_virtual_network.myterraformnetwork.name}"
    address_prefix = "10.0.2.0/24"
}

```

Create public IP address

To access resources across the Internet, create and assign a public IP address to your VM. The following section creates a public IP address named *myPublicIP*:

```

resource "azurerm_public_ip" "myterraformpublicip" {
    name          = "myPublicIP"
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
    allocation_method = "Dynamic"

    tags = {
        environment = "Terraform Demo"
    }
}

```

Create Network Security Group

Network Security Groups control the flow of network traffic in and out of your VM. The following section creates a network security group named *myNetworkSecurityGroup* and defines a rule to allow SSH traffic on TCP port 22:

```

resource "azurerm_network_security_group" "myterraformnsg" {
    name          = "myNetworkSecurityGroup"
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"

    security_rule {
        name          = "SSH"
        priority      = 1001
        direction     = "Inbound"
        access        = "Allow"
        protocol      = "Tcp"
        source_port_range = "*"
        destination_port_range = "22"
        source_address_prefix = "*"
        destination_address_prefix = "*"
    }

    tags = {
        environment = "Terraform Demo"
    }
}

```

Create virtual network interface card

A virtual network interface card (NIC) connects your VM to a given virtual network, public IP address, and network security group. The following section in a Terraform template creates a virtual NIC named *myNIC* connected to the virtual networking resources you have created:

```

resource "azurerm_network_interface" "myterraformnic" {
    name          = "myNIC"
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
    network_security_group_id = "${azurerm_network_security_group.myterraformnsg.id}"

    ip_configuration {
        name          = "myNicConfiguration"
        subnet_id     = "${azurerm_subnet.myterraformsubnet.id}"
        private_ip_address_allocation = "Dynamic"
        public_ip_address_id         = "${azurerm_public_ip.myterraformpublicip.id}"
    }

    tags = {
        environment = "Terraform Demo"
    }
}

```

Create storage account for diagnostics

To store boot diagnostics for a VM, you need a storage account. These boot diagnostics can help you troubleshoot problems and monitor the status of your VM. The storage account you create is only to store the boot diagnostics data. As each storage account must have a unique name, the following section generates some random text:

```

resource "random_id" "randomId" {
  keepers = [
    # Generate a new ID only when a new resource group is defined
    resource_group = "${azurerm_resource_group.myterraformgroup.name}"
  ]

  byte_length = 8
}

```

Now you can create a storage account. The following section creates a storage account, with the name based on the random text generated in the preceding step:

```

resource "azurerm_storage_account" "mystorageaccount" {
  name          = "diag${random_id.randomId.hex}"
  resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
  location      = "eastus"
  account_replication_type = "LRS"
  account_tier   = "Standard"

  tags = {
    environment = "Terraform Demo"
  }
}

```

Create virtual machine

The final step is to create a VM and use all the resources created. The following section creates a VM named *myVM* and attaches the virtual NIC named *myNIC*. The latest *Ubuntu 16.04-LTS* image is used, and a user named *azureuser* is created with password authentication disabled.

SSH key data is provided in the *ssh_keys* section. Provide a valid public SSH key in the *key_data* field.

```

resource "azurerm_virtual_machine" "myterraformvm" {
    name          = "myVM"
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
    network_interface_ids = ["${azurerm_network_interface.myterraformnic.id}"]
    vm_size       = "Standard_DS1_v2"

    storage_os_disk {
        name          = "myOsDisk"
        caching       = "ReadWrite"
        create_option = "FromImage"
        managed_disk_type = "Premium_LRS"
    }

    storage_image_reference {
        publisher     = "Canonical"
        offer         = "UbuntuServer"
        sku           = "16.04.0-LTS"
        version       = "latest"
    }

    os_profile {
        computer_name = "myvm"
        admin_username = "azureuser"
    }

    os_profile_linux_config {
        disable_password_authentication = true
        ssh_keys {
            path      = "/home/azureuser/.ssh/authorized_keys"
            key_data = "ssh-rsa AAAAB3Nz{snip}hwhqT9h"
        }
    }

    boot_diagnostics {
        enabled      = "true"
        storage_uri = "${azurerm_storage_account.mystorageaccount.primary_blob_endpoint}"
    }

    tags = {
        environment = "Terraform Demo"
    }
}

```

Complete Terraform script

To bring all these sections together and see Terraform in action, create a file called *terraform_azure.tf* and paste the following content:

```

# Configure the Microsoft Azure Provider
provider "azurerm" {
    subscription_id = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    client_id       = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    client_secret   = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    tenant_id       = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}

# Create a resource group if it doesn't exist
resource "azurerm_resource_group" "myterraformgroup" {
    name      = "myResourceGroup"
    location  = "eastus"

    tags = {
        environment = "Terraform Demo"
    }
}

```

```

}

# Create virtual network
resource "azurerm_virtual_network" "myterraformnetwork" {
    name          = "myVnet"
    address_space = ["10.0.0.0/16"]
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"

    tags = {
        environment = "Terraform Demo"
    }
}

# Create subnet
resource "azurerm_subnet" "myterraformsubnet" {
    name          = "mySubnet"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
    virtual_network_name = "${azurerm_virtual_network.myterraformnetwork.name}"
    address_prefix = "10.0.1.0/24"
}

# Create public IPs
resource "azurerm_public_ip" "myterraformpublicip" {
    name          = "myPublicIP"
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
    allocation_method = "Dynamic"

    tags = {
        environment = "Terraform Demo"
    }
}

# Create Network Security Group and rule
resource "azurerm_network_security_group" "myterraformnsg" {
    name          = "myNetworkSecurityGroup"
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"

    security_rule {
        name          = "SSH"
        priority      = 1001
        direction     = "Inbound"
        access        = "Allow"
        protocol      = "Tcp"
        source_port_range = "*"
        destination_port_range = "22"
        source_address_prefix = "*"
        destination_address_prefix = "*"
    }

    tags = {
        environment = "Terraform Demo"
    }
}

# Create network interface
resource "azurerm_network_interface" "myterraformnic" {
    name          = "myNIC"
    location      = "eastus"
    resource_group_name = "${azurerm_resource_group.myterraformgroup.name}"
    network_security_group_id = "${azurerm_network_security_group.myterraformnsg.id}"

    ip_configuration {
        name          = "myNicConfiguration"
        subnet_id     = "${azurerm_subnet.myterraformsubnet.id}"
        private_ip_address_allocation = "Dynamic"
        public_ip_address_id         = "${azurerm_public_ip.myterraformpublicip.id}"
    }
}

```

```

        public_ip_address_id      = "${azurerm_public_ip.myterraformpublicip.id}"
    }

tags = {
    environment = "Terraform Demo"
}
}

# Generate random text for a unique storage account name
resource "random_id" "randomId" {
    keepers = [
        # Generate a new ID only when a new resource group is defined
        resource_group = "${azurerm_resource_group.myterraformgroup.name}"
    ]

    byte_length = 8
}

# Create storage account for boot diagnostics
resource "azurerm_storage_account" "mystorageaccount" {
    name                  = "diag${random_id.randomId.hex}"
    resource_group_name   = "${azurerm_resource_group.myterraformgroup.name}"
    location              = "eastus"
    account_tier          = "Standard"
    account_replication_type = "LRS"

    tags = {
        environment = "Terraform Demo"
    }
}

# Create virtual machine
resource "azurerm_virtual_machine" "myterraformvm" {
    name                  = "myVM"
    location              = "eastus"
    resource_group_name   = "${azurerm_resource_group.myterraformgroup.name}"
    network_interface_ids = ["${azurerm_network_interface.myterraformnic.id}"]
    vm_size               = "Standard_DS1_v2"

    storage_os_disk {
        name          = "myOsDisk"
        caching       = "ReadWrite"
        create_option = "FromImage"
        managed_disk_type = "Premium_LRS"
    }

    storage_image_reference {
        publisher = "Canonical"
        offer     = "UbuntuServer"
        sku       = "16.04.0-LTS"
        version   = "latest"
    }

    os_profile {
        computer_name  = "myvm"
        admin_username = "azureuser"
    }

    os_profile_linux_config {
        disable_password_authentication = true
        ssh_keys {
            path      = "/home/azureuser/.ssh/authorized_keys"
            key_data = "ssh-rsa AAAAB3Nz{snip}hwhqT9h"
        }
    }

    boot_diagnostics {
        enabled = "true"
        storage_uri = "${azurerm_storage_account.mystorageaccount.primary_blob_endpoint}"
    }
}

```

```
}

tags = {
    environment = "Terraform Demo"
}
}
```

Build and deploy the infrastructure

With your Terraform template created, the first step is to initialize Terraform. This step ensures that Terraform has all the prerequisites to build your template in Azure.

```
terraform init
```

The next step is to have Terraform review and validate the template. This step compares the requested resources to the state information saved by Terraform and then outputs the planned execution. Resources are *not* created in Azure.

```
terraform plan
```

After you execute the previous command, you should see something like the following screen:

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
```

```
...
```

```
Note: You didn't specify an "-out" parameter to save this plan, so when
"apply" is called, Terraform can't guarantee this is what will execute.
```

```
+ azurerm_resource_group.myterraform
  <snip>
+ azurerm_virtual_network.myterraformnetwork
  <snip>
+ azurerm_network_interface.myterraformnic
  <snip>
+ azurerm_network_security_group.myterraformnsg
  <snip>
+ azurerm_public_ip.myterraformpublicip
  <snip>
+ azurerm_subnet.myterraformsubnet
  <snip>
+ azurerm_virtual_machine.myterraformvm
  <snip>
```

```
Plan: 7 to add, 0 to change, 0 to destroy.
```

If everything looks correct and you are ready to build the infrastructure in Azure, apply the template in Terraform:

```
terraform apply
```

Once Terraform completes, your VM infrastructure is ready. Obtain the public IP address of your VM with [az vm show](#):

```
az vm show --resource-group myResourceGroup --name myVM -d --query [publicIps] --o tsv
```

You can then SSH to your VM:

```
ssh azureuser@<publicIps>
```

Next steps

You have created basic infrastructure in Azure by using Terraform. For more complex scenarios, including examples that use load balancers and virtual machine scale sets, see numerous [Terraform examples for Azure](#). For an up-to-date list of supported Azure providers, see the [Terraform documentation](#).

Cloud-init support for virtual machines in Azure

6/11/2019 • 5 minutes to read • [Edit Online](#)

This article explains the support that exists for [cloud-init](#) to configure a virtual machine (VM) or virtual machine scale sets (VMSS) at provisioning time in Azure. These cloud-init scripts run on first boot once the resources have been provisioned by Azure.

Cloud-init overview

[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 will make your cloud-init deployments and configurations work seamlessly with VMs and VM Scale Sets (VMSS). The following table outlines the current cloud-init enabled images availability on the Azure platform:

PUBLISHER	OFFER	SKU	VERSION	CLOUD-INIT READY
Canonical	UbuntuServer	18.04-LTS	latest	yes
Canonical	UbuntuServer	17.10	latest	yes
Canonical	UbuntuServer	16.04-LTS	latest	yes
Canonical	UbuntuServer	14.04.5-LTS	latest	yes
CoreOS	CoreOS	Stable	latest	yes
OpenLogic	CentOS	7-CI	latest	preview
RedHat	RHEL	7-RAW-CI	latest	preview

Currently Azure Stack does not support the provisioning of RHEL 7.4 and CentOS 7.4 using cloud-init.

What is the difference between cloud-init and the Linux Agent (WALA)?

WALA is an Azure platform-specific agent used to provision and configure VMs, and handle Azure extensions. We are enhancing the task of configuring VMs to use cloud-init instead of the Linux Agent in order to allow existing cloud-init customers to use their current cloud-init scripts. If you have existing investments in cloud-init scripts for configuring Linux systems, there are **no additional settings required** to enable them.

If you do not include the Azure CLI `--custom-data` switch at provisioning time, WALA takes the minimal VM provisioning parameters required to provision the VM and complete the deployment with the defaults. If you reference the cloud-init `--custom-data` switch, whatever is contained in your custom data (individual settings or full script) overrides the WALA defaults.

WALA configurations of VMs are time-constrained to work within the maximum VM provisioning time. Cloud-init configurations applied to VMs do not have time constraints and will not cause a deployment to fail by timing out.

Deploying a cloud-init enabled Virtual Machine

Deploying a cloud-init enabled virtual machine is as simple as referencing a cloud-init enabled distribution during deployment. Linux distribution maintainers have to choose to enable and integrate cloud-init into their base Azure published images. Once you have confirmed the image you want to deploy is cloud-init enabled, you can use the Azure CLI to deploy the image.

The first step in deploying this image is to create a resource group with the [az group create](#) command. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named *myResourceGroup* in the *eastus* location.

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

The next step is to create a file in your current shell, named *cloud-init.txt* and paste the following configuration. For this example, create the file in the Cloud Shell not on your local machine. You can use any editor you wish. Enter `sensible-editor cloud-init.txt` to create the file and see a list of available editors. Choose #1 to use the **nano** editor. Make sure that the whole cloud-init file is copied correctly, especially the first line:

```
#cloud-config
package_upgrade: true
packages:
- httpd
```

Press `ctrl-X` to exit the file, type `y` to save the file and press `enter` to confirm the file name on exit.

The final step is to create a VM with the [az vm create](#) command.

The following example creates a VM named *centos74* and creates SSH keys if they do not already exist in a default key location. To use a specific set of keys, use the `--ssh-key-value` option. Use the `--custom-data` parameter to pass in your cloud-init config file. Provide the full path to the *cloud-init.txt* config if you saved the file outside of your present working directory. The following example creates a VM named *centos74*:

```
az vm create \
--resource-group myResourceGroup \
--name centos74 \
--image OpenLogic:CentOS-CI:7-CI:latest \
--custom-data cloud-init.txt \
--generate-ssh-keys
```

When the VM has been created, the Azure CLI shows information specific to your deployment. Take note of the `publicIpAddress`. This address is used to access the VM. It takes some time for the VM to be created, the packages to install, and the app to start. There are background tasks that continue to run after the Azure CLI returns you to the prompt. You can SSH into the VM and use the steps outlined in the Troubleshooting section to view the cloud-init logs.

Troubleshooting cloud-init

Once the VM has been provisioned, cloud-init will run through all the modules and script defined in `--custom-data` in order to configure the VM. If you need to troubleshoot any errors or omissions from the configuration, you need to search for the module name (`disk_setup` or `runcmd` for example) in the cloud-init log - located in **/var/log/cloud-init.log**.

NOTE

Not every module failure results in a fatal cloud-init overall configuration failure. For example, using the `runcmd` module, if the script fails, cloud-init will still report provisioning succeeded because the runcmd module executed.

For more details of cloud-init logging, refer to the [cloud-init documentation](#)

Next steps

For cloud-init examples of configuration changes, see the following documents:

- [Add an additional Linux user to a VM](#)
- [Run a package manager to update existing packages on first boot](#)
- [Change VM local hostname](#)
- [Install an application package, update configuration files and inject keys](#)

Use cloud-init to set hostname for a Linux VM in Azure

1/28/2019 • 2 minutes to read • [Edit Online](#)

This article shows you how to use [cloud-init](#) to configure a specific hostname on a virtual machine (VM) or virtual machine scale sets (VMSS) at provisioning time in Azure. These cloud-init scripts run on first boot once the resources have been provisioned by Azure. For more information about how cloud-init works natively in Azure and the supported Linux distros, see [cloud-init overview](#)

Set the hostname with cloud-init

By default, the hostname is the same as the VM name when you create a new virtual machine in Azure. To run a cloud-init script to change this default hostname when you create a VM in Azure with [az vm create](#), specify the cloud-init file with the `--custom-data` switch.

To see upgrade process in action, create a file in your current shell named `cloud_init_hostname.txt` and paste the following configuration. For this example, create the file in the Cloud Shell not on your local machine. You can use any editor you wish. Enter `sensible-editor cloud_init_hostname.txt` to create the file and see a list of available editors. Choose #1 to use the **nano** editor. Make sure that the whole cloud-init file is copied correctly, especially the first line.

```
#cloud-config
hostname: myhostname
```

Before deploying this image, you need to create a resource group with the [az group create](#) command. An Azure resource group is a logical container into which Azure resources are deployed and managed. The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Now, create a VM with [az vm create](#) and specify the cloud-init file with `--custom-data cloud_init_hostname.txt` as follows:

```
az vm create \
--resource-group myResourceGroup \
--name centos74 \
--image OpenLogic:CentOS:7-CI:latest \
--custom-data cloud_init_hostname.txt \
--generate-ssh-keys
```

Once created, the Azure CLI shows information about the VM. Use the `publicIpAddress` to SSH to your VM. Enter your own address as follows:

```
ssh <publicIpAddress>
```

To see the VM name, use the `hostname` command as follows:

```
hostname
```

The VM should report the hostname as that value set in the cloud-init file, as shown in the following example output:

```
myhostname
```

Next steps

For additional cloud-init examples of configuration changes, see the following:

- [Add an additional Linux user to a VM](#)
- [Run a package manager to update existing packages on first boot](#)
- [Change VM local hostname](#)
- [Install an application package, update configuration files and inject keys](#)

Use cloud-init to update and install packages in a Linux VM in Azure

2/6/2019 • 2 minutes to read • [Edit Online](#)

This article shows you how to use [cloud-init](#) to update packages on a Linux virtual machine (VM) or virtual machine scale sets (VMSS) at provisioning time in Azure. These cloud-init scripts run on first boot once the resources have been provisioned by Azure. For more information about how cloud-init works natively in Azure and the supported Linux distros, see [cloud-init overview](#)

Update a VM with cloud-init

For security purposes, you may want to configure a VM to apply the latest updates on first boot. As cloud-init works across different Linux distros, there is no need to specify `apt` or `yum` for the package manager. Instead, you define `package_upgrade` and let the cloud-init process determine the appropriate mechanism for the distro in use. This workflow allows you to use the same cloud-init scripts across distros.

To see upgrade process in action, create a file in your current shell named `cloud_init_upgrade.txt` and paste the following configuration. For this example, create the file in the Cloud Shell not on your local machine. You can use any editor you wish. Enter `sensible-editor cloud_init_upgrade.txt` to create the file and see a list of available editors. Choose #1 to use the **nano** editor. Make sure that the whole cloud-init file is copied correctly, especially the first line.

```
#cloud-config
package_upgrade: true
packages:
- httpd
```

Before deploying this image, you need to create a resource group with the [az group create](#) command. An Azure resource group is a logical container into which Azure resources are deployed and managed. The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Now, create a VM with [az vm create](#) and specify the cloud-init file with `--custom-data cloud_init_upgrade.txt` as follows:

```
az vm create \
--resource-group myResourceGroup \
--name centos74 \
--image OpenLogic:CentOS:7-CI:latest \
--custom-data cloud_init_upgrade.txt \
--generate-ssh-keys
```

SSH to the public IP address of your VM shown in the output from the preceding command. Enter your own **publicIpAddress** as follows:

```
ssh <publicIpAddress>
```

Run the package management tool and check for updates.

```
sudo yum update
```

As cloud-init checked for and installed updates on boot, there should be no additional updates to apply. You see the update process, number of altered packages as well as the installation of `httpd` by running `yum history` and review the output similar to the one below.

```
Loaded plugins: fastestmirror, langpacks
ID      | Command line          | Date and time    | Action(s)       | Altered
-----+-----+-----+-----+-----+
 3 | -t -y install httpd   | 2018-04-20 22:42 | Install        | 5
 2 | -t -y upgrade         | 2018-04-20 22:38 | I, U           | 65
 1 |                         | 2017-12-12 20:32 | Install        | 522
```

Next steps

For additional cloud-init examples of configuration changes, see the following:

- [Add an additional Linux user to a VM](#)
- [Run a package manager to update existing packages on first boot](#)
- [Change VM local hostname](#)
- [Install an application package, update configuration files and inject keys](#)

Use cloud-init to add a user to a Linux VM in Azure

2/6/2019 • 2 minutes to read • [Edit Online](#)

This article shows you how to use [cloud-init](#) to add a user on a virtual machine (VM) or virtual machine scale sets (VMSS) at provisioning time in Azure. This cloud-init script runs on first boot once the resources have been provisioned by Azure. For more information about how cloud-init works natively in Azure and the supported Linux distros, see [cloud-init overview](#).

Add a user to a VM with cloud-init

One of the first tasks on any new Linux VM is to add an additional user for yourself to avoid the use of *root*. SSH keys are best practice for security and usability. Keys are added to the `~/.ssh/authorized_keys` file with this cloud-init script.

To add a user to a Linux VM, create a file in your current shell named `cloud_init_add_user.txt` and paste the following configuration. For this example, create the file in the Cloud Shell not on your local machine. You can use any editor you wish. Enter `sensible-editor cloud_init_add_user.txt` to create the file and see a list of available editors. Choose #1 to use the **nano** editor. Make sure that the whole cloud-init file is copied correctly, especially the first line. You need to provide your own public key (such as the contents of `~/.ssh/id_rsa.pub`) for the value of `ssh-authorized-keys:` - it has been shortened here to simplify the example.

```
#cloud-config
users:
  - default
  - name: myadminuser
    groups: sudo
    shell: /bin/bash
    sudo: [ 'ALL=(ALL) NOPASSWD:ALL' ]
    ssh-authorized-keys:
      - ssh-rsa AAAAB3
```

NOTE

The `#cloud-config` file includes the `- default` parameter included. This will append the user, to the existing admin user created during provisioning. If you create a user without the `- default` parameter - the auto generated admin user created by the Azure platform would be overwritten.

Before deploying this image, you need to create a resource group with the [az group create](#) command. An Azure resource group is a logical container into which Azure resources are deployed and managed. The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Now, create a VM with [az vm create](#) and specify the cloud-init file with `--custom-data cloud_init_add_user.txt` as follows:

```
az vm create \
--resource-group myResourceGroup \
--name centos74 \
--image OpenLogic:CentOS:7-CI:latest \
--custom-data cloud_init_add_user.txt \
--generate-ssh-keys
```

SSH to the public IP address of your VM shown in the output from the preceding command. Enter your own **publicIpAddress** as follows:

```
ssh <publicIpAddress>
```

To confirm your user was added to the VM and the specified groups, view the contents of the */etc/group* file as follows:

```
cat /etc/group
```

The following example output shows the user from the *cloud_init_add_user.txt* file has been added to the VM and the appropriate group:

```
root:x:0:
<snip />
sudo:x:27:myadminuser
<snip />
myadminuser:x:1000:
```

Next steps

For additional cloud-init examples of configuration changes, see the following:

- [Add an additional Linux user to a VM](#)
- [Run a package manager to update existing packages on first boot](#)
- [Change VM local hostname](#)
- [Install an application package, update configuration files and inject keys](#)

Use cloud-init to configure a swapfile on a Linux VM

2/6/2019 • 2 minutes to read • [Edit Online](#)

This article shows you how to use [cloud-init](#) to configure the swapfile on various Linux distributions. The swapfile was traditionally configured by the Linux Agent (WALA) based on which distributions required one. This document will outline the process for building the swapfile on demand during provisioning time using cloud-init. For more information about how cloud-init works natively in Azure and the supported Linux distros, see [cloud-init overview](#)

Create swapfile for Ubuntu based images

By default on Azure, Ubuntu gallery images do not create swap files. To enable swap file configuration during VM provisioning time using cloud-init - please see the [AzureSwapPartitions document](#) on the Ubuntu wiki.

Create swapfile for Red Hat and CentOS based images

Create a file in your current shell named `cloud_init_swapfile.txt` and paste the following configuration. For this example, create the file in the Cloud Shell not on your local machine. You can use any editor you wish. Enter `sensible-editor cloud_init_swapfile.txt` to create the file and see a list of available editors. Choose #1 to use the **nano** editor. Make sure that the whole cloud-init file is copied correctly, especially the first line.

```
#cloud-config
disk_setup:
  ephemeral0:
    table_type: gpt
    layout: [66, [33,82]]
    overwrite: true
fs_setup:
  - device: ephemeral0.1
    filesystem: ext4
  - device: ephemeral0.2
    filesystem: swap
mounts:
  - ["ephemeral0.1", "/mnt"]
  - ["ephemeral0.2", "none", "swap", "sw", "0", "0"]
```

Before deploying this image, you need to create a resource group with the `az group create` command. An Azure resource group is a logical container into which Azure resources are deployed and managed. The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Now, create a VM with `az vm create` and specify the cloud-init file with `--custom-data cloud_init_swapfile.txt` as follows:

```
az vm create \
  --resource-group myResourceGroup \
  --name centos74 \
  --image OpenLogic:CentOS:7-CI:latest \
  --custom-data cloud_init_swapfile.txt \
  --generate-ssh-keys
```

Verify swapfile was created

SSH to the public IP address of your VM shown in the output from the preceding command. Enter your own **publicIpAddress** as follows:

```
ssh <publicIpAddress>
```

Once you have SSH'ed into the vm, check if the swapfile was created

```
swapon -s
```

The output from this command should look like this:

Filename	Type	Size	Used	Priority
/dev/sdb2	partition	2494440	0	-1

NOTE

If you have an existing Azure image that has a swap file configured and you want to change the swap file configuration for new images, you should remove the existing swap file. Please see 'Customize Images to provision by cloud-init' document for more details.

Next steps

For additional cloud-init examples of configuration changes, see the following:

- [Add an additional Linux user to a VM](#)
- [Run a package manager to update existing packages on first boot](#)
- [Change VM local hostname](#)
- [Install an application package, update configuration files and inject keys](#)

Use cloud-init to run a bash script in a Linux VM in Azure

2/6/2019 • 2 minutes to read • [Edit Online](#)

This article shows you how to use [cloud-init](#) to run an existing bash script on a Linux virtual machine (VM) or virtual machine scale sets (VMSS) at provisioning time in Azure. These cloud-init scripts run on first boot once the resources have been provisioned by Azure. For more information about how cloud-init works natively in Azure and the supported Linux distros, see [cloud-init overview](#)

Run a bash script with cloud-init

With cloud-init you do not need to convert your existing scripts into a cloud-config, cloud-init accepts multiple input types, one of which is a bash script.

If you have been using the Linux Custom Script Azure Extension to run your scripts, you can migrate them to use cloud-init. However, Azure Extensions have integrated reporting to alert to script failures, a cloud-init image deployment will NOT fail if the script fails.

To see this functionality in action, create a simple bash script for testing. Like the cloud-init `#cloud-config` file, this script must be local to where you will be running the Azure CLI commands to provision your virtual machine. For this example, create the file in the Cloud Shell not on your local machine. You can use any editor you wish. Enter `sensible-editor simple_bash.sh` to create the file and see a list of available editors. Choose #1 to use the **nano** editor. Make sure that the whole cloud-init file is copied correctly, especially the first line.

```
#!/bin/sh
echo "this has been written via cloud-init" + $(date) >> /tmp/myScript.txt
```

Before deploying this image, you need to create a resource group with the [az group create](#) command. An Azure resource group is a logical container into which Azure resources are deployed and managed. The following example creates a resource group named *myResourceGroup* in the *eastus* location.

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

Now, create a VM with [az vm create](#) and specify the bash script file with `--custom-data simple_bash.sh` as follows:

```
az vm create \
--resource-group myResourceGroup \
--name centos74 \
--image OpenLogic:CentOS:7-CI:latest \
--custom-data simple_bash.sh \
--generate-ssh-keys
```

Verify bash script has run

SSH to the public IP address of your VM shown in the output from the preceding command. Enter your own **publicIpAddress** as follows:

```
ssh <publicIpAddress>
```

Change to the **/tmp** directory and verify that myScript.txt file exists and has the appropriate text inside of it. If it does not, you can check the **/var/log/cloud-init.log** for more details. Search for the following entry:

```
Running config-scripts-user using lock Running command ['/var/lib/cloud/instance/scripts/part-001']
```

Next steps

For additional cloud-init examples of configuration changes, see the following:

- [Add an additional Linux user to a VM](#)
- [Run a package manager to update existing packages on first boot](#)
- [Change VM local hostname](#)
- [Install an application package, update configuration files and inject keys](#)

Prepare an existing Linux Azure VM image for use with cloud-init

6/25/2019 • 3 minutes to read • [Edit Online](#)

This article shows you how to take an existing Azure virtual machine and prepare it to be redeployed and ready to use cloud-init. The resulting image can be used to deploy a new virtual machine or virtual machine scale sets - either of which could then be further customized by cloud-init at deployment time. These cloud-init scripts run on first boot once the resources have been provisioned by Azure. For more information about how cloud-init works natively in Azure and the supported Linux distros, see [cloud-init overview](#)

Prerequisites

This document assumes you already have a running Azure virtual machine running a supported version of the Linux operating system. You have already configured the machine to suit your needs, installed all the required modules, processed all the required updates and have tested it to ensure it meets your requirements.

Preparing RHEL 7.6 / CentOS 7.6

You need to SSH into your Linux VM and run the following commands in order to install cloud-init.

```
sudo yum makecache fast
sudo yum install -y gdisk cloud-utils-growpart
sudo yum install -y cloud-init
```

Update the `cloud_init_modules` section in `/etc/cloud/cloud.cfg` to include the following modules:

```
- disk_setup
- mounts
```

Here is a sample of what a general-purpose `cloud_init_modules` section looks like.

```
cloud_init_modules:
- migrator
- bootcmd
- write-files
- growpart
- resizefs
- disk_setup
- mounts
- set_hostname
- update_hostname
- update_etc_hosts
- rsyslog
- users-groups
- ssh
```

A number of tasks relating to provisioning and handling ephemeral disks need to be updated in `/etc/waagent.conf`. Run the following commands to update the appropriate settings.

```
sed -i 's/Provisioning.Enabled=y/Provisioning.Enabled=n/g' /etc/waagent.conf
sed -i 's/Provisioning.UseCloudInit=n/Provisioning.UseCloudInit=y/g' /etc/waagent.conf
sed -i 's/ResourceDisk.Format=y/ResourceDisk.Format=n/g' /etc/waagent.conf
sed -i 's/ResourceDisk.EnableSwap=y/ResourceDisk.EnableSwap=n/g' /etc/waagent.conf
cloud-init clean
```

Allow only Azure as a datasource for the Azure Linux Agent by creating a new file

/etc/cloud/cloud.cfg.d/91-azure_datasource.cfg using an editor of your choice with the following line:

```
# Azure Data Source config
datasource_list: [ Azure ]
```

If your existing Azure image has a swap file configured and you want to change the swap file configuration for new images using cloud-init, you need to remove the existing swap file.

For Red Hat based images - follow the instructions in the following Red Hat document explaining how to [remove the swap file](#).

For CentOS images with swapfile enabled, you can run the following command to turn off the swapfile:

```
sudo swapoff /mnt/resource/swapfile
```

Ensure the swapfile reference is removed from `/etc/fstab` - it should look something like the following output:

```
# /etc/fstab
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=99cf66df-2fef-4aad-b226-382883643a1c / xfs defaults 0 0
UUID=7c473048-a4e7-4908-bad3-a9be22e9d37d /boot xfs defaults 0 0
```

To save space and remove the swap file you can run the following command:

```
rm /mnt/resource/swapfile
```

Extra step for cloud-init prepared image

NOTE

If your image was previously a **cloud-init** prepared and configured image, you need to do the following steps.

The following three commands are only used if the VM you are customizing to be a new specialized source image was previously provisioned by cloud-init. You do NOT need to run these if your image was configured using the Azure Linux Agent.

```
sudo cloud-init clean --logs
sudo waagent -deprovision+user -force
```

Finalizing Linux Agent setting

All Azure platform images have the Azure Linux Agent installed, regardless if it was configured by cloud-init or not.

Run the following command to finish deprovisioning the user from the Linux machine.

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

For more information about the Azure Linux Agent deprovision commands, see the [Azure Linux Agent](#) for more details.

Exit the SSH session, then from your bash shell, run the following AzureCLI commands to deallocate, generalize and create a new Azure VM image. Replace `myResourceGroup` and `sourceVmName` with the appropriate information reflecting your sourceVM.

```
az vm deallocate --resource-group myResourceGroup --name sourceVmName
az vm generalize --resource-group myResourceGroup --name sourceVmName
az image create --resource-group myResourceGroup --name myCloudInitImage --source sourceVmName
```

Next steps

For additional cloud-init examples of configuration changes, see the following:

- [Add an additional Linux user to a VM](#)
- [Run a package manager to update existing packages on first boot](#)
- [Change VM local hostname](#)
- [Install an application package, update configuration files and inject keys](#)

Azure and Jenkins

2/28/2019 • 2 minutes to read • [Edit Online](#)

Jenkins is a popular open-source automation server used to set up continuous integration and delivery (CI/CD) for your software projects. You can host your Jenkins deployment in Azure or extend your existing Jenkins configuration using Azure resources. Jenkins plugins are also available to simplify CI/CD of your applications to Azure.

This article is an introduction to using Azure with Jenkins, detailing the core Azure features available to Jenkins users. For more information about getting started with your own Jenkins server in Azure, see [Create a Jenkins server on Azure](#).

Host your Jenkins servers in Azure

Host Jenkins in Azure to centralize your build automation and scale your deployment as the needs of your software projects grow. You can deploy Jenkins in Azure using:

- [The Jenkins solution template](#) in Azure Marketplace.
- [Azure virtual machines](#). See our [tutorial](#) to create a Jenkins instance on a VM.
- On a Kubernetes cluster running in [Azure Container Service](#), see our [how-to](#).

Monitor and manage your Azure Jenkins deployment using [Azure Monitor logs](#) and the [Azure CLI](#).

Scale your build automation on demand

Add build agents to your existing Jenkins deployment to scale your Jenkins build capacity as the number of builds and complexity of your jobs and pipelines increase. You can run these build agents on Azure virtual machines by using the [Azure VM Agents plugin](#). See our [tutorial](#) for more details.

Once configured with an [Azure service principal](#), Jenkins jobs and pipelines can use this credential to:

- Securely store and archive build artifacts in [Azure Storage](#) using the [Azure Storage plugin](#). Review the [Jenkins storage how-to](#) to learn more.
- Manage and configure Azure resources with the [Azure CLI](#).

Deploy your code into Azure services

Use Jenkins plugins to deploy your applications to Azure as part of your Jenkins CI/CD pipelines. Deploying into [Azure App Service](#) and [Azure Container Service](#) lets you stage, test, and release updates to your applications without managing the underlying infrastructure.

Plug-ins are available to deploy to the following services and environments:

- [Azure App Service on Linux](#). See the [tutorial](#) to get started.
- [Azure App Service](#). See the [how-to](#) to get started.

Create a Jenkins server on an Azure Linux VM from the Azure portal

3/14/2019 • 5 minutes to read • [Edit Online](#)

This quickstart shows how to install [Jenkins](#) on an Ubuntu Linux VM with the tools and plug-ins configured to work with Azure. When you're finished, you have a Jenkins server running in Azure building a sample Java app from [GitHub](#).

Prerequisites

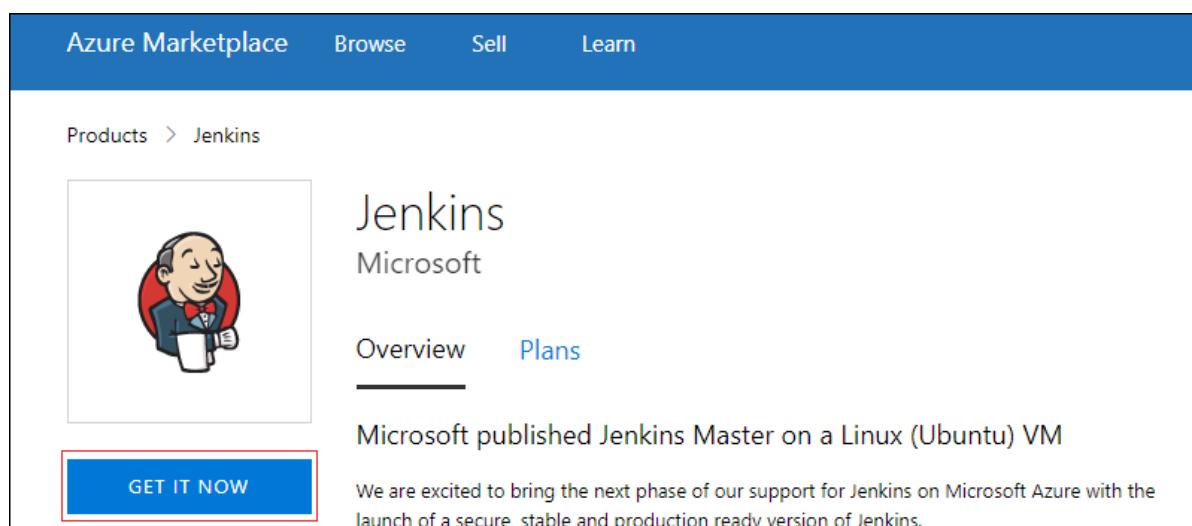
- An Azure subscription
- Access to SSH on your computer's command line (such as the Bash shell or [PuTTY](#))

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

Create the Jenkins VM from the solution template

Jenkins supports a model where the Jenkins server delegates work to one or more agents to allow a single Jenkins installation to host a large number of projects or to provide different environments needed for builds or tests. The steps in this section guide you through installing and configuring a Jenkins server on Azure.

1. In your browser, open the [Azure Marketplace image for Jenkins](#).
2. Select **GET IT NOW**.



3. After reviewing the pricing details and terms information, select **Continue**.

Create this app in Azure



Jenkins
By Microsoft

Software plan

Jenkins

Pricing: This solution template deploys software components and Azure infrastructure components. The price is the cost of those components.

Details: Microsoft published Jenkins Master on a Linux (Ubuntu) VM

I agree to the provider's [terms of use](#) and [privacy policy](#) and understand that the rights to use this product do not come from Microsoft, unless Microsoft is the provider. Use of Azure Marketplace is governed by separate [terms](#).

Continue

4. Select **Create** to configure the Jenkins server in the Azure portal.

 Jenkins
Microsoft

We are excited to bring the next phase of our support for Jenkins on Microsoft Azure with the launch of a secure, stable and production ready version of Jenkins.

Note: For instructions on connecting to this Jenkins instance once deployed, please browse to the URL or public IP of this instance. The URL is the Domain name label you enter in Settings and the suffix shown below this field.

This solution template will install the latest stable Jenkins version on a Linux (Ubuntu 14.04 LTS) VM along with tools and plugins configured to work with Azure. This includes –

- git for source control
- Azure Credentials plugin for connecting securely
- Azure VM Agents plugin for elastic build, test and continuous integration
- Azure Storage plugin for storing artifacts
- Azure CLI to deploy apps using scripts

For a detailed walkthrough of the steps this solution automates for you, please visit our [blog post](#).

This solution template is designed to configure a Jenkins instance following best practices with minimal Azure knowledge. With a handful of user inputs and a simple single-click deployment through the Azure portal, you can provision a fully configured Jenkins instance in minutes, which can use Azure services anywhere across the globe.

[!\[\]\(155ae6d8c2f6b72e622d11c12706902c_img.jpg\)](#) [!\[\]\(c25b81df0b4775e947ed1fac6c2bc6de_img.jpg\)](#) [!\[\]\(ca05a98498ba3790a7df993215e76bba_img.jpg\)](#) [!\[\]\(c91a869d4d4fcbce1ec34949af3394c4_img.jpg\)](#) [!\[\]\(81bebb5094ea8d0db14be83ca39fcae6_img.jpg\)](#) [!\[\]\(23f5ff5a99214a5616b27cee4fc6156c_img.jpg\)](#)

```

graph TD
    Jenkins((Jenkins)) -- "VM Agents" --> VM[Virtual Machines]
    Jenkins -- "Storage" --> Storage[Storage]
    Jenkins -- Docker --> CR[Container Registry]
    Jenkins -- Kubernetes --> CS[Container Service]
  
```

PUBLISHED

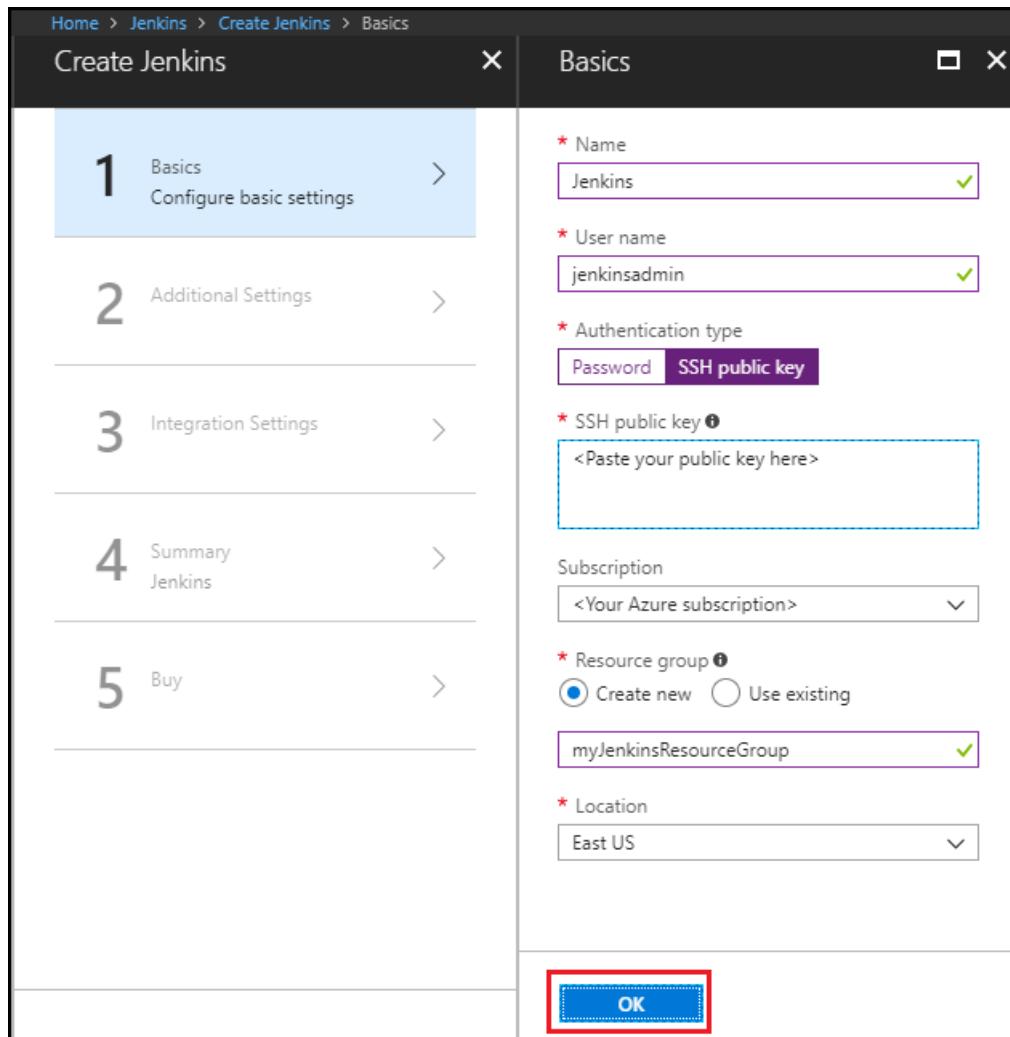
Select a deployment model [?](#)

5. In the **Basics** tab, specify the following values:

- **Name** - Enter `Jenkins`.
- **User name** - Enter the user name to use when signing in to the virtual machine on which Jenkins is running. The user name must meet [specific requirements](#).
- **Authentication type** - Select **SSH public key**.
- **SSH public key** - Copy and paste an RSA public key in single-line format (starting with `ssh-rsa`) or multi-line PEM format. You can generate SSH keys using ssh-keygen on Linux and macOS, or PuTTYGen on Windows. For more information about SSH keys and Azure, see the article, [How to](#)

Use SSH keys with Windows on Azure.

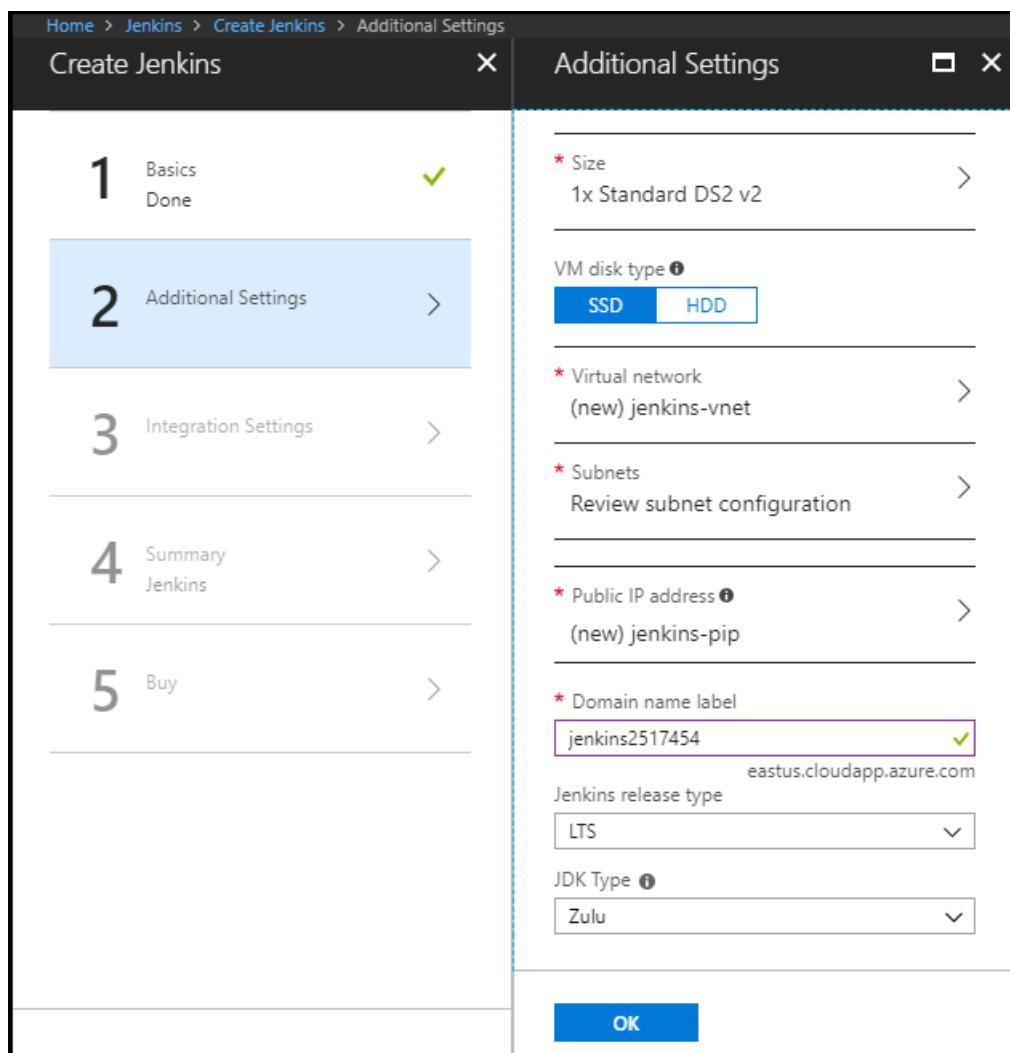
- **Subscription** - Select the Azure subscription into which you want to install Jenkins.
- **Resource group** - Select **Create new**, and enter a name for the resource group that serves as a logical container for the collection of resources that make up your Jenkins installation.
- **Location** - Select **East US**.



6. Select **OK** to proceed to the **Additional Settings** tab.
7. In the **Additional Settings** tab, specify the following values:
 - **Size** - Select the appropriate sizing option for your Jenkins virtual machine.
 - **VM disk type** - Specify either HDD (hard-disk drive) or SSD (solid-state drive) to indicate which storage disk type is allowed for the Jenkins virtual machine.
 - **Virtual network** - (Optional) Select **Virtual network** to modify the default settings.
 - **Subnets** - Select **Subnets**, verify the information, and select **OK**.
 - **Public IP address** - The IP address name defaults to the Jenkins name you specified in the previous page with a suffix of -IP. You can select the option to change that default.
 - **Domain name label** - Specify the value for the fully qualified URL to the Jenkins virtual machine.
 - **Jenkins release type** - Select the desired release type from the options: **LTS**, **Weekly build**, or **Azure Verified**. The **LTS** and **Weekly build** options are explained in the article, [Jenkins LTS Release Line](#). The **Azure Verified** option refers to a [Jenkins LTS version](#) that has been verified to run on

Azure.

- **JDK Type** - JDK to be installed. Default is Zulu tested, certified builds of OpenJDK.



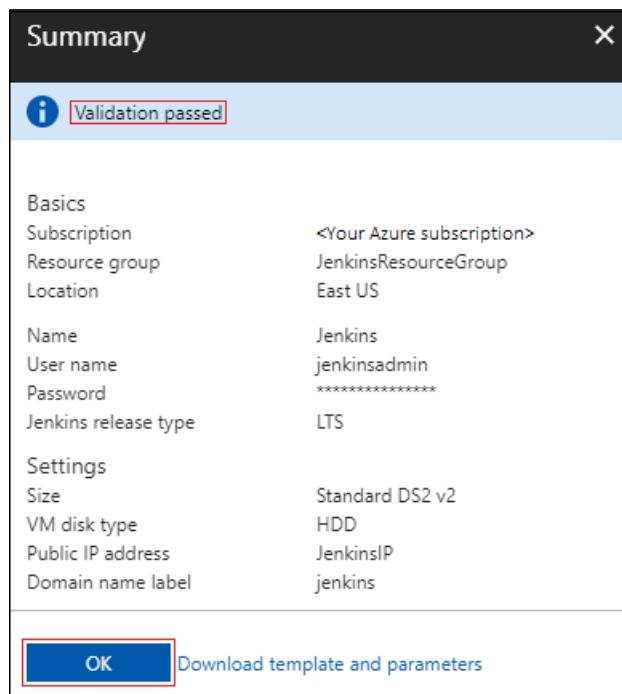
8. Select **OK** to proceed to the **Integration Settings** tab.

9. In the **Integration Settings** tab, specify the following values:

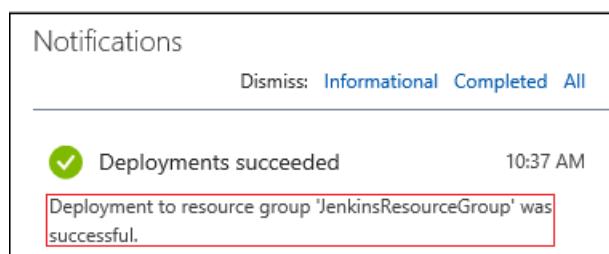
- **Service Principal** - The service principal is added into Jenkins as a credential for authentication with Azure. **Auto** means that the principal will be created by MSI (Managed Service Identity). **Manual** means that the principal should be created by you.
 - **Application ID** and **Secret** - If you select the **Manual** option for the **Service Principal** option, you'll need to specify the **Application ID** and **Secret** for your service principal. When [creating a service principal](#), note that the default role is **Contributor**, which is sufficient for working with Azure resources.
- **Enable Cloud Agents** - Specify the default cloud template for agents where **ACI** refers to Azure Container Instance, and **VM** refers to virtual machines. You can also specify **No** if you don't wish to enable a cloud agent.

10. Select **OK** to proceed to the **Summary** tab.

11. When the **Summary** tab displays, the information entered is validated. Once you see the **Validation passed** message (at the top of the tab), select **OK**.

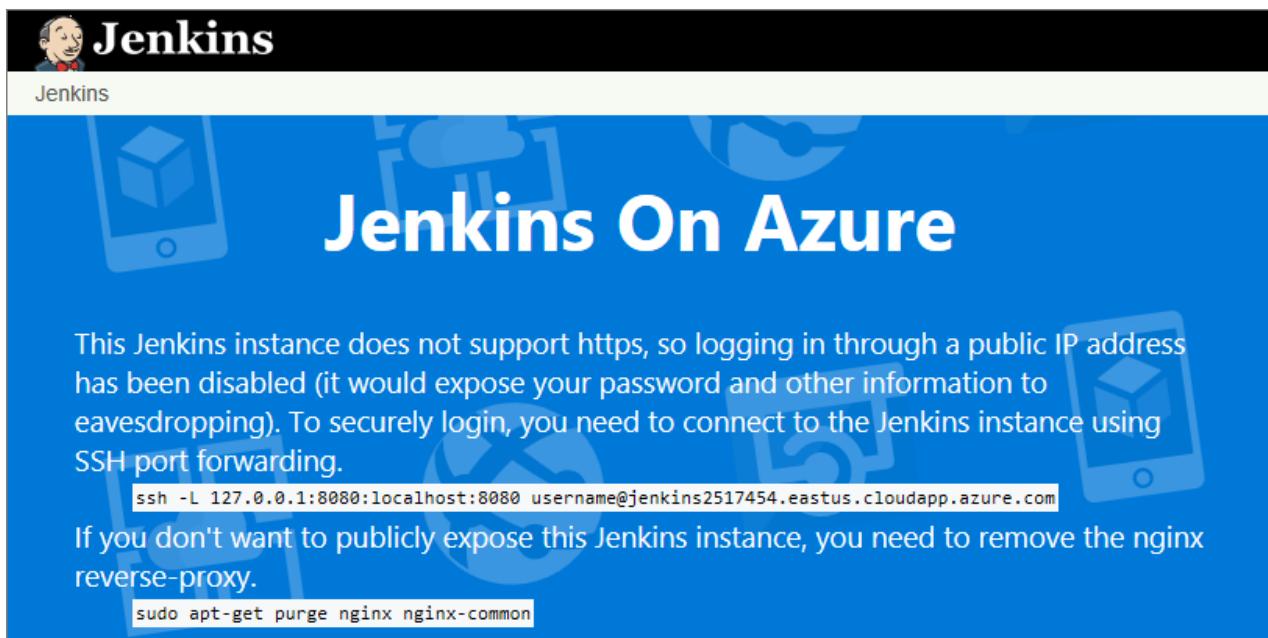


12. When the **Create** tab displays, select **Create** to create the Jenkins virtual machine. When your server is ready, a notification displays in the Azure portal.



Connect to Jenkins

Navigate to your virtual machine (for example, <http://jenkins2517454.eastus.cloudapp.azure.com/>) in your web browser. The Jenkins console is inaccessible through unsecured HTTP so instructions are provided on the page to access the Jenkins console securely from your computer using an SSH tunnel.



Set up the tunnel using the `ssh` command on the page from the command line, replacing `username` with the name of the virtual machine admin user chosen earlier when setting up the virtual machine from the solution

template.

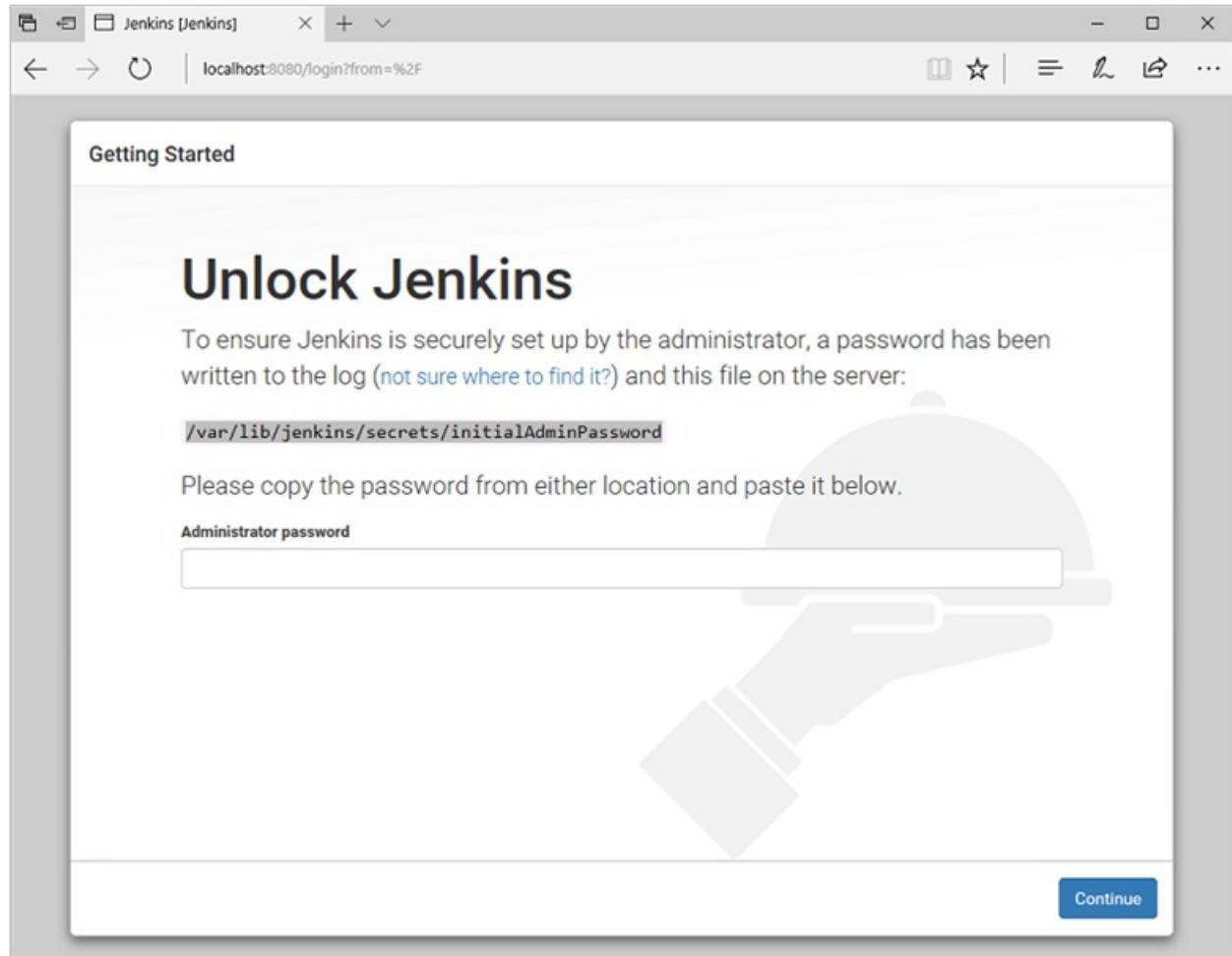
```
ssh -L 127.0.0.1:8080:localhost:8080 jenkinsadmin@jenkins2517454.eastus.cloudapp.azure.com
```

After you have started the tunnel, navigate to <http://localhost:8080/> on your local machine.

Get the initial password by running the following command in the command line while connected through SSH to the Jenkins VM.

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Unlock the Jenkins dashboard for the first time using this initial password.



Select **Install suggested plugins** on the next page and then create a Jenkins admin user used to access the Jenkins dashboard.

The screenshot shows the Jenkins dashboard at localhost:8080. The main header includes a Jenkins logo, the title "Dashboard [Jenkins]", and navigation icons. The top right features a search bar, a "Devops" link, and a "log out" button. A "ENABLE AUTO REFRESH" link is also present. On the left, a sidebar lists links: "New Item", "People", "Build History", "Manage Jenkins", "My Views", and "Credentials". The central area displays a large "Welcome to Jenkins!" message with a "Please [create new jobs](#) to get started." button. Below this are sections for "Build Queue" (empty) and "Build Executor Status" (1 Idle, 2 Idle). At the bottom, a footer bar shows the page was generated on June 19, 2017, at 6:02:58 PM UTC, with links to the REST API and Jenkins version 2.46.3.

The Jenkins server is now ready to build code.

Create your first job

Select **Create new jobs** from the Jenkins console, then name it **mySampleApp** and select **Freestyle project**, then select **OK**.

The screenshot shows the "Enter an item name" dialog. The input field contains "mySampleApp", which is highlighted as a "Required field". Below the input field, there are two options: "Freestyle project" (selected) and "Pipeline". The "Freestyle project" option is described as the central feature of Jenkins, mentioning software build. The "Pipeline" option is described as orchestrating long-running activities across multiple builds.

Select the **Source Code Management** tab, enable **Git**, and enter the following URL in **Repository URL** field:

```
https://github.com/spring-guides/gs-spring-boot.git
```

Source Code Management

The screenshot shows the Jenkins configuration interface for Source Code Management. The 'Git' option is selected. In the 'Repositories' section, there is one repository defined with the 'Repository URL' set to `https://github.com/spring-guides/gs-spring-boot.git`. A red box highlights this URL field.

Select the **Build** tab, then select **Add build step, Invoke Gradle script**. Select **Use Gradle Wrapper**, then enter `complete` in **Wrapper location** and `build` for **Tasks**.

The screenshot shows the Jenkins build step configuration for 'Invoke Gradle script'. The 'Use Gradle Wrapper' option is selected. The 'Wrapper location' field contains `complete` and the 'Tasks' field contains `build`. A red box highlights the 'Use Gradle Wrapper' radio button.

Select **Advanced** and then enter `complete` in the **Root Build script** field. Select **Save**.

The screenshot shows the Jenkins advanced configuration for the root build script. The 'Root Build script' field contains `complete`, which is highlighted with a red box.

Build the code

Select **Build Now** to compile the code and package the sample app. When your build completes, select the **Workspace** link for the project.

The screenshot shows the Jenkins workspace for the project 'mySampleApp'. It features a 'Workspace' link next to a folder icon, which is highlighted with a red box. Below it is a 'Recent Changes' link with a document icon. The 'Permalinks' section lists four recent builds.

Link	Date
Last build (#19)	13 sec ago
Last stable build (#19)	13 sec ago
Last successful build (#19)	13 sec ago
Last completed build (#19)	13 sec ago

Navigate to `complete/build/libs` and ensure the `gs-spring-boot-0.1.0.jar` is there to verify that your build was successful. Your Jenkins server is now ready to build your own projects in Azure.

Troubleshooting the Jenkins solution template

If you encounter any bugs with the Jenkins solution template, file an issue in the [Jenkins GitHub repo](#).

Next Steps

[Add Azure VMs as Jenkins agents](#)

Scale your Jenkins deployments to meet demand with Azure VM agents

1/7/2019 • 4 minutes to read • [Edit Online](#)

This tutorial shows how to use the Jenkins [Azure VM Agents plugin](#) to add on-demand capacity with Linux virtual machines running in Azure.

In this tutorial, you will:

- Install the Azure VM Agents plugin
- Configure the plugin to create resources in your Azure subscription
- Set the compute resources available to each agent
- Set the operating system and tools installed on each agent
- Create a new Jenkins freestyle job
- Run the job on an Azure VM agent

Prerequisites

- An Azure subscription
- A Jenkins master server. If you don't have one, view the [quickstart](#) to set up one in Azure.

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

Install Azure VM Agents plugin

TIP

If you deployed Jenkins on Azure using the [solution template](#), the Azure VM Agent plugin is already installed.

1. From the Jenkins dashboard, select **Manage Jenkins**, then select **Manage Plugins**.
2. Select the **Available** tab, then search for **Azure VM Agents**. Select the checkbox next to the entry for the plugin and select **Install without restart** from the bottom of the dashboard.

Configure the Azure VM Agents plugin

1. From the Jenkins dashboard, select **Manage Jenkins**, then **Configure System**.
2. Scroll to the bottom of the page and find the **Cloud** section with the **Add new cloud** dropdown and choose **Microsoft Azure VM Agents**.
3. Select an existing service principal from **Add** drop-down in the **Azure Credentials** section. If none is listed, perform the following steps to [create a service principal](#) for your Azure account and add it to your Jenkins configuration:
 - a. Select **Add** next to **Azure Credentials** and choose **Jenkins**.
 - b. In the **Add Credentials** dialog, select **Microsoft Azure Service Principal** from the **Kind** drop-down.
 - c. Create an Active Directory Service principal from the Azure CLI or [Cloud Shell](#).

```
az ad sp create-for-rbac --name jenkins_sp --password secure_password
```

```
{
  "appId": "BBBBBBBB-BBBB-BBBB-BBBB-BBBBBBBBBB",
  "displayName": "jenkins_sp",
  "name": "http://jenkins_sp",
  "password": "secure_password",
  "tenant": "CCCCCCCC-CCCC-CCCC-CCCCCCCCCCCC"
}
```

d. Enter the credentials from the service principal into the **Add credentials** dialog. If you don't know your Azure subscription ID, you can query it from the CLI:

```
az account list
```

```
{
  "cloudName": "AzureCloud",
  "id": "AAAAAAA-AAAA-AAAA-AAAA-AAAAAAA",
  "isDefault": true,
  "name": "Visual Studio Enterprise",
  "state": "Enabled",
  "tenantId": "CCCCCCCC-CCCC-CCCC-CCCCCCCCCCCC",
  "user": {
    "name": "raisa@fabrikam.com",
    "type": "user"
  }
}
```

The completed service principal should use the `id` field for **Subscription ID**, the `appId` value for **Client ID**, `password` for **Client Secret**, and `tenant` for **Tenant ID**. Select **Add** to add the service principal and then configure the plugin to use the newly created credential.

The screenshot shows the Jenkins 'Add Credentials' dialog with the following settings:

- Domain:** Global credentials (unrestricted)
- Kind:** Microsoft Azure Service Principal
- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- Subscription ID:** AAAAAAA-AAAA-AAAA-AAAA-AAAAAAA
- Client ID:** BBBB BBBB-BBBB-BBBB-BBBB-BBBBBBBB
- Client Secret:** (redacted)
- Or, Certificate:** ... Select a Certificate ... | Add
- Tenant ID:** CCCCCCCC-CCCC-CCCC-CCCC-CCCCCC
- Azure Environment:** Azure

- In the **Resource Group Name** section, leave **Create new** selected and enter `myJenkinsAgentGroup`.
- Select **Verify configuration** to connect to Azure to test the profile settings.

6. Select **Apply** to update the plugin configuration.

Configure agent resources

Configure a template for use to define an Azure VM agent. This template defines the compute resources each agent has when created.

1. Select **Add** next to **Add Azure Virtual Machine Template**.
2. Enter `defaulttemplate` for the **Name**
3. Enter `ubuntu` for the **Label**
4. Select the desired [Azure region](#) from the combo box.
5. Select a [VM size](#) from the drop-down under **Virtual Machine Size**. A general-purpose `Standard_DS1_v2` size is fine for this tutorial.
6. Leave the **Retention time** at `60`. This setting defines the number of minutes Jenkins can wait before it deallocated idle agents. Specify 0 if you do not want idle agents to be removed automatically.

General Configuration	
Name	<input type="text" value="defaulttemplate"/>
Description	<input type="text"/>
Labels	<input type="text" value="ubuntu"/>
Region	<input type="text" value="East US"/>
Virtual Machine Size	<input type="text" value="Standard_DS1_v2"/>
Storage Account Name	<input type="text" value="jnshwip6riim7wcip1n9pfa"/>
Retention Time (in minutes)	<input type="text" value="60"/>
Shutdown Only (Do Not Delete) After Retention Time	<input type="checkbox"/>
Usage	<input type="text" value="Use this node as much as possible"/>

Configure agent operating system and tools

In the **Image Configuration** section of the plugin configuration, select **Ubuntu 16.04 LTS**. Check the boxes next to **Install Git (Latest)**, **Install Maven (V3.5.0)**, and **Install Docker** to install these tools on newly created agents.

Image Configuration	
<input checked="" type="radio"/> Use Built-In Image	<input type="text" value="Ubuntu 16.04 LTS"/>
Pre-installed Tools	
Install Git (Latest)	<input checked="" type="checkbox"/>
Install Maven (V3.5.0)	<input checked="" type="checkbox"/>
Install Docker (Only for Linux)	<input checked="" type="checkbox"/>

Select **Add** next to **Admin Credentials**, then select **Jenkins**. Enter a username and password used to sign in to the agents, making sure they satisfy the [username and password policy](#) for administrative accounts on Azure VMs.

Select **Verify Template** to verify the configuration and then select **Save** to save your changes and return to the Jenkins dashboard.

Create a job in Jenkins

1. Within the Jenkins dashboard, click **New Item**.
2. Enter `demoproject1` for the name and select **Freestyle project**, then select **OK**.
3. In the **General** tab, choose **Restrict where project can be run** and type `ubuntu` in **Label Expression**. You see a message confirming that the label is served by the cloud configuration created in the previous step.

The screenshot shows the Jenkins 'General' configuration page for a new project. The 'Project name' is set to 'demoproject1'. Under 'Restrict where this project can be run', the 'Label Expression' is set to 'ubuntu', with a note below stating 'Label ubuntu is serviced by no nodes and 1 cloud'. There are several other optional checkboxes like 'Discard old builds' and 'Execute concurrent builds if necessary'.

4. In the **Source Code Management** tab, select **Git** and add the following URL into the **Repository URL** field:
`https://github.com/spring-projects/spring-petclinic.git`
5. In the **Build** tab, select **Add build step**, then **Invoke top-level Maven targets**. Enter `package` in the **Goals** field.
6. Select **Save** to save the job definition.

Build the new job on an Azure VM agent

1. Go back to the Jenkins dashboard.
2. Select the job you created in the previous step, then click **Build now**. A new build is queued, but does not start until an agent VM is created in your Azure subscription.
3. Once the build is complete, go to **Console output**. You see that the build was performed remotely on an Azure agent.



Console Output

```
Started by user Devops
Building remotely on defaulttemplate45db20 (ubuntu) in workspace /home/devops/workspace/demoproject1
Finished: SUCCESS
```

Troubleshooting the Jenkins plugin

If you encounter any bugs with the Jenkins plugins, file an issue in the [Jenkins JIRA](#) for the specific component.

Next steps

[CI/CD to Azure App Service](#)

Using Azure Storage with a Jenkins continuous integration solution

5/6/2019 • 9 minutes to read • [Edit Online](#)

This article illustrates how to use Blob storage as a repository of build artifacts created by a Jenkins continuous integration (CI) solution, or as a source of downloadable files to be used in a build process. One of the scenarios where you would find this solution useful is when you're coding in an agile development environment (using Java or other languages), builds are running based on continuous integration, and you need a repository for your build artifacts, so that you could, for example, share them with other organization members, your customers, or maintain an archive. Another scenario is when your build job itself requires other files, for example, dependencies to download as part of the build input.

In this tutorial, you will be using the Azure Storage Plugin for Jenkins CI made available by Microsoft.

Jenkins overview

Jenkins enables continuous integration of a software project by allowing developers to easily integrate their code changes and have builds produced automatically and frequently, thereby increasing the productivity of the developers. Builds are versioned, and build artifacts can be uploaded to various repositories. This article shows how to use Azure blob storage as the repository of the build artifacts. It will also show how to download dependencies from Azure blob storage.

More information about Jenkins can be found at [Meet Jenkins](#).

Benefits of using the Blob service

Benefits of using the Blob service to host your agile development build artifacts include:

- High availability of your build artifacts and/or downloadable dependencies.
- Performance when your Jenkins CI solution uploads your build artifacts.
- Performance when your customers and partners download your build artifacts.
- Control over user access policies, with a choice between anonymous access, expiration-based shared access signature access, private access, etc.

Prerequisites

- A Jenkins continuous integration solution.

If you currently don't have a Jenkins CI solution, you can run a Jenkins CI solution using the following technique:

1. On a Java-enabled machine, download jenkins.war from <https://jenkins-ci.org>.

2. At a command prompt that is opened to the folder that contains jenkins.war, run:

```
java -jar jenkins.war
```

3. In your browser, open <http://localhost:8080/> to open the Jenkins dashboard, which you will use to install and configure the Azure Storage plugin.

While a typical Jenkins CI solution would be set up to run as a service, running the Jenkins war at the command line will be sufficient for this tutorial.

- An Azure account. You can sign up for an Azure account at <https://www.azure.com>.
- An Azure storage account. If you don't already have a storage account, you can create one using the steps at [Create a Storage Account](#).
- Familiarity with the Jenkins CI solution is recommended but not required, as the following content will use a basic example to show you the steps needed when using the Blob service as a repository for Jenkins CI build artifacts.

How to use the Blob service with Jenkins CI

To use the Blob service with Jenkins, you'll need to install the Azure Storage plugin, configure the plugin to use your storage account, and then create a post-build action that uploads your build artifacts to your storage account. These steps are described in the following sections.

How to install the Azure Storage plugin

1. Within the Jenkins dashboard, select **Manage Jenkins**.
2. In the **Manage Jenkins** page, select **Manage Plugins**.
3. Select the **Available** tab.
4. In the **Artifact Uploaders** section, check **Microsoft Azure Storage plugin**.
5. Select either **Install without restart** or **Download now and install after restart**.
6. Restart Jenkins.

How to configure the Azure Storage plugin to use your storage account

1. Within the Jenkins dashboard, select **Manage Jenkins**.
2. In the **Manage Jenkins** page, select **Configure System**.
3. In the **Microsoft Azure Storage Account Configuration** section:
 - a. Enter your storage account name, which you can obtain from the [Azure Portal](#).
 - b. Enter your storage account key, also obtainable from the [Azure Portal](#).
 - c. Use the default value for **Blob Service Endpoint URL** if you are using the global Azure cloud. If you are using a different Azure cloud, use the endpoint as specified in the [Azure Portal](#) for your storage account.
 - d. Select **Validate storage credentials** to validate your storage account.
 - e. [Optional] If you have additional storage accounts that you want made available to your Jenkins CI, select **Add more Storage Accounts**.
 - f. Select **Save** to save your settings.

How to create a post-build action that uploads your build artifacts to your storage account

For instructional purposes, you first need to create a job that will create several files, and then add in the post-build action to upload the files to your storage account.

1. Within the Jenkins dashboard, select **New Item**.
2. Name the job **MyJob**, select **Build a free-style software project**, and then select **OK**.
3. In the **Build** section of the job configuration, select **Add build step** and select **Execute Windows batch command**.
4. In **Command**, use the following commands:

```
md text
cd text
echo Hello Azure Storage from Jenkins > hello.txt
date /t > date.txt
time /t >> date.txt
```

5. In the **Post-build Actions** section of the job configuration, select **Add post-build action** and select **Upload artifacts to Azure Blob storage**.
6. For **Storage account name**, select the storage account to use.
7. For **Container name**, specify the container name. (The container will be created if it does not already exist when the build artifacts are uploaded.) You can use environment variables, so for this example enter `${JOB_NAME}` as the container name.

Tip

Below the **Command** section where you entered a script for **Execute Windows batch command** is a link to the environment variables recognized by Jenkins. Select that link to learn the environment variable names and descriptions. Environment variables that contain special characters, such as the **BUILD_URL** environment variable, are not allowed as a container name or common virtual path.

8. Select **Make new container public by default** for this example. (If you want to use a private container, you'll need to create a shared access signature to allow access, which is beyond the scope of this article. You can learn more about shared access signatures at [Using Shared Access Signatures \(SAS\)](#).)
9. [Optional] Select **Clean container before uploading** if you want the container to be cleared of contents before build artifacts are uploaded (leave it unchecked if you do not want to clean the contents of the container).
10. For **List of Artifacts to upload**, enter `text/*.txt`.
11. For **Common virtual path for uploaded artifacts**, for purposes of this tutorial, enter `${BUILD_ID}/${BUILD_NUMBER}`.
12. Select **Save** to save your settings.
13. In the Jenkins dashboard, select **Build Now** to run **MyJob**. Examine the console output for status. Status messages for Azure storage will be included in the console output when the post-build action starts to upload build artifacts.
14. Upon successful completion of the job, you can examine the build artifacts by opening the public blob.
 - a. Sign in to the [Azure Portal](#).
 - b. Select **Storage**.
 - c. Select the storage account name that you used for Jenkins.
 - d. Select **Containers**.
 - e. Select the container named **myjob**, which is the lowercase version of the job name that you assigned when you created the Jenkins job. Container names and blob names are lowercase (and case-sensitive) in Azure storage. Within the list of blobs for the container named **myjob**, you should see **hello.txt** and **date.txt**. Copy the URL for either of these items and open it in your browser. You will see the text file that was uploaded as a build artifact.

Only one post-build action that uploads artifacts to Azure blob storage can be created per job. The single post-build action to upload artifacts to Azure blob storage can specify different files (including wildcards) and paths to files within **List of Artifacts to upload** using a semi-colon as a separator. For example, if your Jenkins build produces JAR files and TXT files in your workspace's **build** folder, and you want to upload both to Azure blob

storage, use the following value for the **List of Artifacts to upload** option: `build/*.jar;build/*.txt`. You can also use double-colon syntax to specify a path to use within the blob name. For example, if you want the JARs to get uploaded using **binaries** in the blob path and the TXT files to get uploaded using **notices** in the blob path, use the following value for the **List of Artifacts to upload** option: `build/*.jar::binaries;build/*.txt::notices`.

How to create a build step that downloads from Azure blob storage

The following steps illustrate to configure a build step to download items from Azure blob storage, which is useful if you want to include items in your build. An example of using this pattern is JARs that you might want to persist in Azure blob storage.

1. In the **Build** section of the job configuration, select **Add build step** and select **Download from Azure Blob storage**.
2. For **Storage account name**, select the storage account to use.
3. For **Container name**, specify the name of the container that has the blobs you want to download. You can use environment variables.
4. For **Blob name**, specify the blob name. You can use environment variables. Also, you can use an asterisk, as a wildcard after you specify the initial letter(s) of the blob name. For example, `project*` would specify all blobs whose names start with `project`.
5. [Optional] For **Download path**, specify the path on the Jenkins machine where you want to download files from Azure blob storage. Environment variables can also be used. (If you do not provide a value for **Download path**, the files from Azure blob storage will be downloaded to the job's workspace.)

If you have additional items you want to download from Azure blob storage, you can create additional build steps.

After you run a build, you can check the build history console output, or look at your download location, to see whether the blobs you expected were successfully downloaded.

Components used by the Blob service

This section provides an overview of the Blob service components.

- **Storage Account:** All access to Azure Storage is done through a storage account. A storage account is the highest level of the namespace for accessing blobs. An account can contain an unlimited number of containers, as long as their total size is under 100 TB.
- **Container:** A container provides a grouping of a set of blobs. All blobs must be in a container. An account can contain an unlimited number of containers. A container can store an unlimited number of blobs.
- **Blob:** A file of any type and size. There are two types of blobs that can be stored in Azure Storage: block and page blobs. Most files are block blobs. A single block blob can be up to 200 GB in size. This tutorial uses block blobs. Page blobs, another blob type, can be up to 1 TB in size, and are more efficient when ranges of bytes in a file are modified frequently. For more information about blobs, see [Understanding Block Blobs, Append Blobs, and Page Blobs](#).
- **URL format:** Blobs are addressable using the following URL format:

`http://storageaccount.blob.core.windows.net/container_name/blob_name`

(The format above applies to the global Azure cloud. If you are using a different Azure cloud, use the endpoint within the [Azure Portal](#) to determine your URL endpoint.)

In the format above, `storageaccount` represents the name of your storage account, `container_name` represents the name of your container, and `blob_name` represents the name of your blob, respectively. Within the container name, you can have multiple paths, separated by a forward slash, `/`. The example container name used for this tutorial was **MyJob**, and `${BUILD_ID}/${BUILD_NUMBER}` was used for

the common virtual path, resulting in the blob having a URL of the following form:

```
http://example.blob.core.windows.net/myjob/2014-04-14_23-57-00/1/hello.txt
```

Troubleshooting the Jenkins plugin

If you encounter any bugs with the Jenkins plugins, file an issue in the [Jenkins JIRA](#) for the specific component.

Next steps

- [Meet Jenkins](#)
- [Azure Storage SDK for Java](#)
- [Azure Storage Client SDK Reference](#)
- [Azure Storage Services REST API](#)
- [Azure Storage Team Blog](#)

For more information, visit [Azure for Java developers](#).

2 minutes to read

How to use Docker Machine to create hosts in Azure

2/15/2019 • 3 minutes to read • [Edit Online](#)

This article details how to use [Docker Machine](#) to create hosts in Azure. The `docker-machine` command creates a Linux virtual machine (VM) in Azure then installs Docker. You can then manage your Docker hosts in Azure using the same local tools and workflows. To use docker-machine in Windows 10, you must use Linux bash.

Create VMs with Docker Machine

First, obtain your Azure subscription ID with `az account show` as follows:

```
sub=$(az account show --query "id" -o tsv)
```

You create Docker host VMs in Azure with `docker-machine create` by specifying `azure` as the driver. For more information, see the [Docker Azure Driver documentation](#)

The following example creates a VM named `myVM`, based on "Standard D2 v2" plan, creates a user account named `azureuser`, and opens port `80` on the host VM. Follow any prompts to log in to your Azure account and grant Docker Machine permissions to create and manage resources.

```
docker-machine create -d azure \
--azure-subscription-id $sub \
--azure-ssh-user azureuser \
--azure-open-port 80 \
--azure-size "Standard_DS2_v2" \
myvm
```

The output looks similar to the following example:

```
Creating CA: /Users/user/.docker/machine/certs/ca.pem
Creating client certificate: /Users/user/.docker/machine/certs/cert.pem
Running pre-create checks...
(myvm) Completed machine pre-create checks.
Creating machine...
(myvm) Querying existing resource group. name="docker-machine"
(myvm) Creating resource group. name="docker-machine" location="westus"
(myvm) Configuring availability set. name="docker-machine"
(myvm) Configuring network security group. name="myvm-firewall" location="westus"
(myvm) Querying if virtual network already exists. rg="docker-machine" location="westus" name="docker-machine-vnet"
(myvm) Creating virtual network. name="docker-machine-vnet" rg="docker-machine" location="westus"
(myvm) Configuring subnet. name="docker-machine" vnet="docker-machine-vnet" cidr="192.168.0.0/16"
(myvm) Creating public IP address. name="myvm-ip" static=false
(myvm) Creating network interface. name="myvm-nic"
(myvm) Creating storage account. sku=Standard_LRS name="vhdski0hvfazyd8mn991cg50" location="westus"
(myvm) Creating virtual machine. location="westus" size="Standard_A2" username="azureuser"
osImage="canonical:UbuntuServer:16.04.0-LTS:latest" name="myvm"
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Installing Docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env myvm
```

Configure your Docker shell

To connect to your Docker host in Azure, define the appropriate connection settings. As noted at the end of the output, view the connection information for your Docker host as follows:

```
docker-machine env myvm
```

The output is similar to the following example:

```
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://40.68.254.142:2376"
export DOCKER_CERT_PATH="/Users/user/.docker/machine/machines/machine"
export DOCKER_MACHINE_NAME="machine"
# Run this command to configure your shell:
# eval $(docker-machine env myvm)
```

To define the connection settings, you can either run the suggested configuration command (`eval $(docker-machine env myvm)`), or you can set the environment variables manually.

Run a container

To see a container in action, let's run a basic NGINX webserver. Create a container with `docker run` and expose port 80 for web traffic as follows:

```
docker run -d -p 80:80 --restart=always nginx
```

The output is similar to the following example:

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
ff3d52d8f55f: Pull complete
226f4ec56ba3: Pull complete
53d7dd52b97d: Pull complete
Digest: sha256:41ad9967ea448d7c2b203c699b429abe1ed5af331cd92533900c6d77490e0268
Status: Downloaded newer image for nginx:latest
675e6056cb81167fe38ab98bf397164b01b998346d24e567f9eb7a7e94fba14a
```

View running containers with `docker ps`. The following example output shows the NGINX container running with port 80 exposed:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
d5b78f27b335	nginx	"nginx -g 'daemon off'"	5 minutes ago	Up 5 minutes	0.0.0.0:80->80/tcp, 443/tcp

Test the container

Obtain the public IP address of Docker host as follows:

```
docker-machine ip myvm
```

To see the container in action, open a web browser and enter the public IP address noted in the output of the preceding command:



Next steps

For examples on using Docker Compose, see [Get started with Docker and Compose in Azure](#).

Get started with Docker and Compose to define and run a multi-container application in Azure

3/14/2019 • 3 minutes to read • [Edit Online](#)

With [Compose](#), you use a simple text file to define an application consisting of multiple Docker containers. You then spin up your application in a single command that does everything to deploy your defined environment. As an example, this article shows you how to quickly set up a WordPress blog with a backend MariaDB SQL database on an Ubuntu VM. You can also use Compose to set up more complex applications.

This article was last tested on 2/14/2019 using the [Azure Cloud Shell](#) and the [Azure CLI](#) version 2.0.58.

Create Docker host with Azure CLI

Install the latest [Azure CLI](#) and log in to an Azure account using [az login](#).

First, create a resource group for your Docker environment with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

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

Create a file named *cloud-init.txt* and paste the following configuration. Enter `sensible-editor cloud-init.txt` to create the file and see a list of available editors.

```
#include https://get.docker.com
```

Now create a VM with [az vm create](#). Use the `--custom-data` parameter to pass in your cloud-init config file. Provide the full path to the *cloud-init.txt* config if you saved the file outside of your present working directory. The following example creates a VM named *myDockerVM* and opens port 80 to web traffic.

```
az vm create \
  --resource-group myDockerGroup \
  --name myDockerVM \
  --image UbuntuLTS \
  --admin-username azureuser \
  --generate-ssh-keys \
  --custom-data cloud-init.txt
az vm open-port --port 80 \
  --resource-group myDockerGroup \
  --name myDockerVM
```

It takes a few minutes for the VM to be created, the packages to install, and the app to start. There are background tasks that continue to run after the Azure CLI returns you to the prompt. When the VM has been created, take note of the `publicIpAddress` displayed by the Azure CLI.

Install Compose

SSH to your new Docker host VM. Provide your own IP address.

```
ssh azureuser@10.10.111.11
```

Install Compose on the VM.

```
sudo apt install docker-compose
```

Create a docker-compose.yml configuration file

Create a `docker-compose.yml` configuration file to define the Docker containers to run on the VM. The file specifies the image to run on each container, necessary environment variables and dependencies, ports, and the links between containers. For details on yml file syntax, see [Compose file reference](#).

Create a `docker-compose.yml` file. Use your favorite text editor to add some data to the file. The following example creates the file with a prompt for `sensible-editor` to pick an editor that you wish to use.

```
sensible-editor docker-compose.yml
```

Paste the following example into your Docker Compose file. This configuration uses images from the [DockerHub Registry](#) to install WordPress (the open source blogging and content management system) and a linked backend MariaDB SQL database. Enter your own `MYSQL_ROOT_PASSWORD`.

```
wordpress:
  image: wordpress
  links:
    - db:mysql
  ports:
    - 80:80

db:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: <your password>
```

Start the containers with Compose

In the same directory as your `docker-compose.yml` file, run the following command (depending on your environment, you might need to run `docker-compose` using `sudo`):

```
sudo docker-compose up -d
```

This command starts the Docker containers specified in `docker-compose.yml`. It takes a minute or two for this step to complete. You'll see output similar to the following:

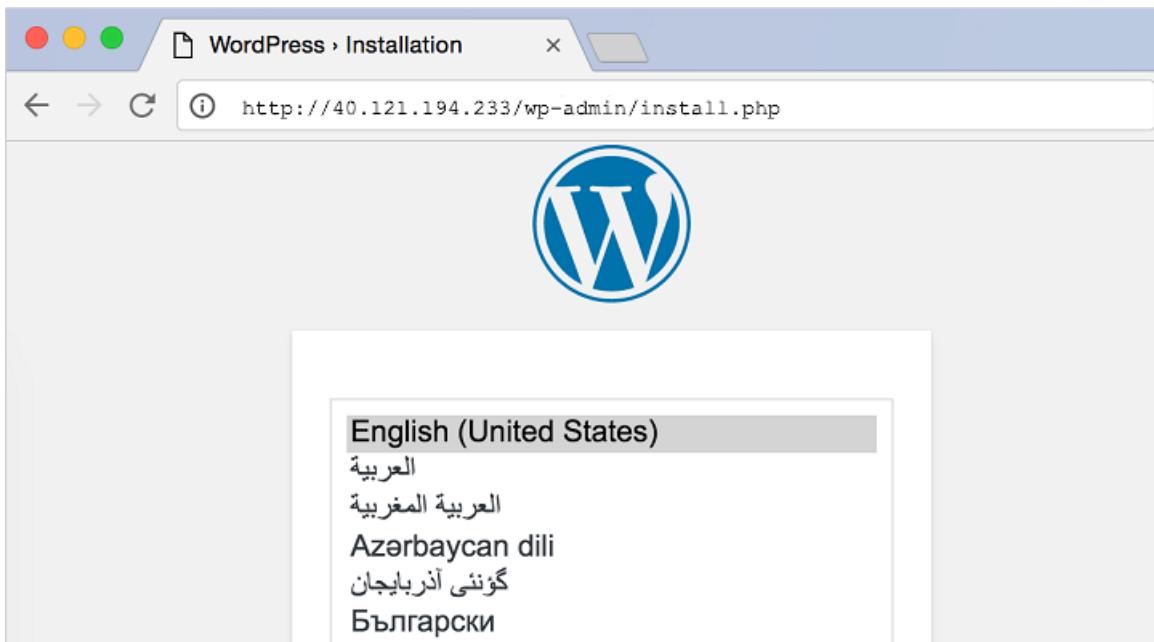
```
Creating wordpress_db_1...
Creating wordpress_wordpress_1...
...
```

To verify that the containers are up, type `sudo docker-compose ps`. You should see something like:

Name	Command	State	Ports
azureuser_db_1	docker-entrypoint.sh mysqld	Up	3306/tcp
azureuser_wordpress_1	docker-entrypoint.sh apach ...	Up	0.0.0.0:80->80/tcp

You can now connect to WordPress directly on the VM on port 80. Open a web browser and enter the IP address

name of your VM. You should now see the WordPress start screen, where you can complete the installation and get started with the application.



Next steps

- Check out the [Compose command-line reference](#) and [user guide](#) for more examples of building and deploying multi-container apps.
- Use an Azure Resource Manager template, either your own or one contributed from the [community](#), to deploy an Azure VM with Docker and an application set up with Compose. For example, the [Deploy a WordPress blog with Docker](#) template uses Docker and Compose to quickly deploy WordPress with a MySQL backend on an Ubuntu VM.
- Try integrating Docker Compose with a Docker Swarm cluster. See [Using Compose with Swarm](#) for scenarios.

Cloud Foundry on Azure

3/14/2019 • 2 minutes to read • [Edit Online](#)

Cloud Foundry is an open-source platform-as-a-service (PaaS) for building, deploying, and operating 12-factor applications developed in various languages and frameworks. This document describes the options you have for running Cloud Foundry on Azure and how you can get started.

Cloud Foundry offerings

There are two forms of Cloud Foundry available to run on Azure: open-source Cloud Foundry (OSS CF) and Pivotal Cloud Foundry (PCF). OSS CF is an entirely [open-source](#) version of Cloud Foundry managed by the Cloud Foundry Foundation. Pivotal Cloud Foundry is an enterprise distribution of Cloud Foundry from Pivotal Software Inc. We look at some of the differences between the two offerings.

Open-source Cloud Foundry

You can deploy OSS Cloud Foundry on Azure by first deploying a BOSH director and then deploying Cloud Foundry, using the [instructions provided on GitHub](#). To learn more about using OSS CF, see the [documentation](#) provided by the Cloud Foundry Foundation.

Microsoft provides best-effort support for OSS CF through the following community channels:

- #bosh-azure-cpi channel on [Cloud Foundry Slack](#)
- [cf-bosh mailing list](#)
- GitHub issues for the [CPI](#) and [service broker](#)

NOTE

The level of support for your Azure resources, such as the virtual machines where you run Cloud Foundry, is based on your Azure support agreement. Best-effort community support only applies to the Cloud Foundry-specific components.

Pivotal Cloud Foundry

Pivotal Cloud Foundry includes the same core platform as the OSS distribution, along with a set of proprietary management tools and enterprise support. To run PCF on Azure, you must acquire a license from Pivotal. The PCF offer from the Azure marketplace includes a 90-day trial license.

The tools include [Pivotal Operations Manager](#), a web application that simplifies deployment and management of a Cloud Foundry foundation, and [Pivotal Apps Manager](#), a web application for managing users and applications.

In addition to the support channels listed for OSS CF above, a PCF license entitles you to contact Pivotal for support. Microsoft and Pivotal have also enabled support workflows that allow you to contact either party for assistance and have your inquiry routed appropriately depending on where the issue lies.

Azure Service Broker

Cloud Foundry encourages the "[twelve-factor app](#)" methodology, which promotes a clean separation of stateless application processes and stateful backing services. [Service brokers](#) offer a consistent way to provision and bind backing services to applications. The [Azure service broker](#) provides some of the key Azure services through this channel, including Azure storage and Azure SQL.

If you are using Pivotal Cloud Foundry, the service broker is also [available as a tile](#) from the Pivotal Network.

Related resources

Azure DevOps Services plugin

Cloud Foundry is well suited to agile software development, including the use of continuous integration (CI) and continuous delivery (CD). If you use Azure DevOps Services to manage your projects and would like to set up a CI/CD pipeline targeting Cloud Foundry, you can use the [Azure DevOps Services Cloud Foundry build extension](#). The plugin makes it simple to configure and automate deployments to Cloud Foundry, whether running in Azure or another environment.

Next steps

- [Deploy Pivotal Cloud Foundry from the Azure Marketplace](#)
- [Deploy an app to Cloud Foundry in Azure](#)

Deploy your first app to Cloud Foundry on Microsoft Azure

3/14/2019 • 4 minutes to read • [Edit Online](#)

Cloud Foundry is a popular open-source application platform available on Microsoft Azure. In this article, we show how to deploy and manage an application on Cloud Foundry in an Azure environment.

Create a Cloud Foundry environment

There are several options for creating a Cloud Foundry environment on Azure:

- Use the [Pivotal Cloud Foundry offer](#) in the Azure Marketplace to create a standard environment that includes PCF Ops Manager and the Azure Service Broker. You can find [complete instructions](#) for deploying the marketplace offer in the Pivotal documentation.
- Create a customized environment by [deploying Pivotal Cloud Foundry manually](#).
- [Deploy the open-source Cloud Foundry packages directly](#) by setting up a [BOSH](#) director, a VM that coordinates the deployment of the Cloud Foundry environment.

IMPORTANT

If you are deploying PCF from the Azure Marketplace, make a note of the SYSTEMDOMAINURL and the admin credentials required to access the Pivotal Apps Manager, both of which are described in the marketplace deployment guide. They are needed to complete this tutorial. For marketplace deployments, the SYSTEMDOMAINURL is in the form <https://system.ip-address.cfpcfazure.com>.

Connect to the Cloud Controller

The Cloud Controller is the primary entry point to a Cloud Foundry environment for deploying and managing applications. The core Cloud Controller API (CCAPI) is a REST API, but it is accessible through various tools. In this case, we interact with it through the [Cloud Foundry CLI](#). You can install the CLI on Linux, MacOS, or Windows, but if you'd prefer not to install it at all, it is available pre-installed in the [Azure Cloud Shell](#).

To log in, prepend `api` to the SYSTEMDOMAINURL that you obtained from the marketplace deployment. Since the default deployment uses a self-signed certificate, you should also include the `--skip-ssl-validation` switch.

```
cf login -a https://api.SYSTEMDOMAINURL --skip-ssl-validation
```

You are prompted to log in to the Cloud Controller. Use the admin account credentials that you acquired from the marketplace deployment steps.

Cloud Foundry provides *orgs* and *spaces* as namespaces to isolate the teams and environments within a shared deployment. The PCF marketplace deployment includes the default *system* org and a set of spaces created to contain the base components, like the autoscaling service and the Azure service broker. For now, choose the *system* space.

Create an org and space

If you type `cf apps`, you see a set of system applications that have been deployed in the system space within the

system org.

You should keep the *system* org reserved for system applications, so create an org and space to house our sample application.

```
cf create-org myorg  
cf create-space dev -o myorg
```

Use the target command to switch to the new org and space:

```
cf target -o testorg -s dev
```

Now, when you deploy an application, it is automatically created in the new org and space. To confirm that there are currently no apps in the new org/space, type `cf apps` again.

NOTE

For more information about orgs and spaces and how they can be used for role-based access control (RBAC), see the [Cloud Foundry documentation](#).

Deploy an application

Let's use a sample Cloud Foundry application called Hello Spring Cloud, which is written in Java and based on the [Spring Framework](#) and [Spring Boot](#).

Clone the Hello Spring Cloud repository

The Hello Spring Cloud sample application is available on GitHub. Clone it to your environment and change into the new directory:

```
git clone https://github.com/cloudfoundry-samples/hello-spring-cloud  
cd hello-spring-cloud
```

Build the application

Build the app using [Apache Maven](#).

```
mvn clean package
```

Deploy the application with cf push

You can deploy most applications to Cloud Foundry using the `push` command:

```
cf push
```

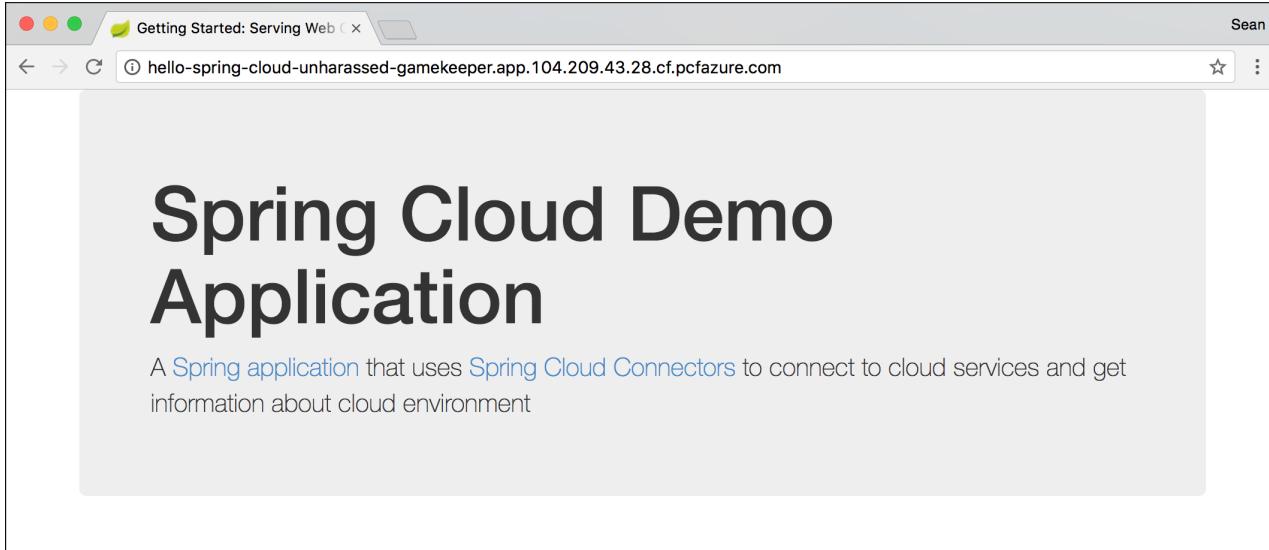
When you *push* an application, Cloud Foundry detects the type of application (in this case, a Java app) and identifies its dependencies (in this case, the Spring framework). It then packages everything required to run your code into a standalone container image, known as a *droplet*. Finally, Cloud Foundry schedules the application on one of the available machines in your environment and creates a URL where you can reach it, which is available in the output of the command.

```
Showing health and status for app hello-spring-cloud in org myorg / space dev as admin...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: hello-spring-cloud-unharassed-gamekeeper.app.104.209.43.28.cf.pcfazure.com
last uploaded: Tue Jun 6 06:36:43 UTC 2017
stack: cflinuxfs2
buildpack: container-certificate-trust-store=2.0.0_RELEASE java-buildpack=v3.13-offline-https://github.com/cloudfoundry/java-buildpack.git#03b493f java-main open-jdk-like-jre=1.8.0_121 open-jdk-like-memory-calculator=2.0.2_RELEASE spring-auto-reconfiguration=1.10...

      state      since      cpu      memory      disk      details
#0  running   2017-06-06 06:37:19 AM  0.0%  147M of 1G  157.8M of 1G
sean@Azure:~/hello-spring-cloud$
```

To see the hello-spring-cloud application, open the provided URL in your browser:



NOTE

To learn more about what happens during `cf push`, see [How Applications Are Staged](#) in the Cloud Foundry documentation.

View application logs

You can use the Cloud Foundry CLI to view logs for an application by its name:

```
cf logs hello-spring-cloud
```

By default, the logs command uses *tail*, which shows new logs as they are written. To see new logs appear, refresh the hello-spring-cloud app in the browser.

To view logs that have already been written, add the `recent` switch:

```
cf logs --recent hello-spring-cloud
```

Scale the application

By default, `cf push` only creates a single instance of your application. To ensure high availability and enable scale out for higher throughput, you generally want to run more than one instance of your applications. You can easily scale out already deployed applications using the `scale` command:

```
cf scale -i 2 hello-spring-cloud
```

Running the `cf app` command on the application shows that Cloud Foundry is creating another instance of the application. Once the application has started, Cloud Foundry automatically starts load balancing traffic to it.

Next steps

- [Read the Cloud Foundry documentation](#)
- [Set up the Azure DevOps Services plugin for Cloud Foundry](#)
- [Configure the Microsoft Log Analytics Nozzle for Cloud Foundry](#)

OpenShift in Azure

5/12/2019 • 2 minutes to read • [Edit Online](#)

OpenShift is an open and extensible container application platform that brings Docker and Kubernetes to the enterprise.

OpenShift includes Kubernetes for container orchestration and management. It adds developer-centric and operations-centric tools that enable:

- Rapid application development.
- Easy deployment and scaling.
- Long-term lifecycle maintenance for teams and applications.

There are multiple versions of OpenShift available. Of these versions, only two are available today for customers to deploy in Azure: OpenShift Container Platform and OKD (formerly OpenShift Origin).

Azure Red Hat OpenShift

Microsoft Azure Red Hat OpenShift is a fully managed offering of OpenShift running in Azure. This service is jointly managed and supported by Microsoft and Red Hat. For more details, see the [Azure Red Hat OpenShift Service](#) documentation.

OpenShift Container Platform

Container Platform is an enterprise-ready [commercial version](#) from and supported by Red Hat. With this version, customers purchase the necessary entitlements for OpenShift Container Platform and are responsible for installation and management of the entire infrastructure.

Because customers "own" the entire platform, they can install it in their on-premises datacenter, or in a public cloud (such as Azure).

OKD

OKD is an [open-source](#) upstream project of OpenShift that's community supported. OKD can be installed on CentOS or Red Hat Enterprise Linux (RHEL).

Next steps

- [Configure common prerequisites for OpenShift in Azure](#)
- [Deploy OpenShift Container Platform in Azure](#)
- [Deploy OpenShift Container Platform Self-Managed Marketplace Offer](#)
- [Deploy OpenShift in Azure Stack](#)
- [Post-deployment tasks](#)
- [Troubleshoot OpenShift deployment](#)

Common prerequisites for deploying OpenShift in Azure

6/16/2019 • 5 minutes to read • [Edit Online](#)

This article describes common prerequisites for deploying OpenShift Container Platform or OKD in Azure.

The installation of OpenShift uses Ansible playbooks. Ansible uses Secure Shell (SSH) to connect to all cluster hosts to complete installation steps.

When ansible makes the SSH connection to the remote hosts, it can't enter a password. For this reason, the private key can't have a password (passphrase) associated with it or deployment fails.

Because the virtual machines (VMs) deploy via Azure Resource Manager templates, the same public key is used for access to all VMs. The corresponding private key must be on the VM that executes all the playbooks as well. To perform this action securely, an Azure key vault is used to pass the private key into the VM.

If there's a need for persistent storage for containers, then persistent volumes are required. OpenShift supports Azure virtual hard disks (VHDs) for persistent volumes, but Azure must first be configured as the cloud provider.

In this model, OpenShift:

- Creates a VHD object in an Azure storage account or a managed disk.
- Mounts the VHD to a VM and formats the volume.
- Mounts the volume to the pod.

For this configuration to work, OpenShift needs permissions to perform these tasks in Azure. A service principal is used for this purpose. The service principal is a security account in Azure Active Directory that is granted permissions to resources.

The service principal needs to have access to the storage accounts and VMs that make up the cluster. If all OpenShift cluster resources deploy to a single resource group, the service principal can be granted permissions to that resource group.

This guide describes how to create the artifacts associated with the prerequisites.

- Create a key vault to manage SSH keys for the OpenShift cluster.
- Create a service principal for use by the Azure Cloud Provider.

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

Sign in to Azure

Sign in to your Azure subscription with the [az login](#) command and follow the on-screen directions, or click **Try it** to use Cloud Shell.

```
az login
```

Create a resource group

Create a resource group with the [az group create](#) command. An Azure resource group is a logical container into which Azure resources are deployed and managed. You should use a dedicated resource group to host the key

vault. This group is separate from the resource group into which the OpenShift cluster resources deploy.

The following example creates a resource group named *keyvaultrg* in the *eastus* location:

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

Create a key vault

Create a key vault to store the SSH keys for the cluster with the [az keyvault create](#) command. The key vault name must be globally unique and must be enabled for template deployment or the deployment will fail with "KeyVaultParameterReferenceSecretRetrieveFailed" error.

The following example creates a key vault named *keyvault* in the *keyvaultrg* resource group:

```
az keyvault create --resource-group keyvaultrg --name keyvault \
    --enabled-for-template-deployment true \
    --location eastus
```

Create an SSH key

An SSH key is needed to secure access to the OpenShift cluster. Create an SSH key pair by using the [ssh-keygen](#) command (on Linux or macOS):

```
ssh-keygen -f ~/.ssh/openshift_rsa -t rsa -N ''
```

NOTE

Your SSH key pair can't have a password / passphrase.

For more information on SSH keys on Windows, see [How to create SSH keys on Windows](#). Be sure to export the private key in OpenSSH format.

Store the SSH private key in Azure Key Vault

The OpenShift deployment uses the SSH key you created to secure access to the OpenShift master. To enable the deployment to securely retrieve the SSH key, store the key in Key Vault by using the following command:

```
az keyvault secret set --vault-name keyvault --name keysecret --file ~/.ssh/openshift_rsa
```

Create a service principal

OpenShift communicates with Azure by using a username and password or a service principal. An Azure service principal is a security identity that you can use with apps, services, and automation tools like OpenShift. You control and define the permissions as to which operations the service principal can perform in Azure. It's best to scope the permissions of the service principal to specific resource groups rather than the entire subscription.

Create a service principal with [az ad sp create-for-rbac](#) and output the credentials that OpenShift needs.

The following example creates a service principal and assigns it contributor permissions to a resource group named *openshiftrg*.

First, create the resource group named *openshiftrg*:

```
az group create -l eastus -n openshifttrg
```

Create service principal:

```
scope=`az group show --name openshifttrg --query id`  
az ad sp create-for-rbac --name openshiftsp \  
--role Contributor --password {Strong Password} \  
--scopes $scope
```

If you're using Windows, execute `az group show --name openshifttrg --query id` and use the output in place of \$scope.

Take note of the appId property returned from the command:

```
{  
  "appId": "11111111-abcd-1234-efgh-111111111111",  
  "displayName": "openshiftsp",  
  "name": "http://openshiftsp",  
  "password": {Strong Password},  
  "tenant": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"  
}
```

WARNING

Be sure to create a secure password. Follow the [Azure AD password rules and restrictions](#) guidance.

For more information on service principals, see [Create an Azure service principal with Azure CLI](#).

Prerequisites applicable only to Resource Manager template

Secrets will need to be created for the SSH private key (**sshPrivateKey**), Azure AD client secret (**aadClientSecret**), OpenShift admin password (**openshiftPassword**), and Red Hat Subscription Manager password or activation key (**rhsmPasswordOrActivationKey**). Additionally, if custom SSL certificates are used, then six additional secrets will need to be created - **routingcafile**, **routingcertfile**, **routingkeyfile**, **mastercafile**, **mastercertfile**, and **masterkeyfile**. These parameters will be explained in more detail.

The template references specific secret names so you **must** use the bolded names listed above (case sensitive).

Custom Certificates

By default, the template will deploy an OpenShift cluster using self-signed certificates for the OpenShift web console and the routing domain. If you want to use custom SSL certificates, set 'routingCertType' to 'custom' and 'masterCertType' to 'custom'. You'll need the CA, Cert, and Key files in .pem format for the certificates. It is possible to use custom certificates for one but not the other.

You'll need to store these files in Key Vault secrets. Use the same Key Vault as the one used for the private key. Rather than require 6 additional inputs for the secret names, the template is hard-coded to use specific secret names for each of the SSL certificate files. Store the certificate data using the information from the following table.

SECRET NAME	CERTIFICATE FILE
mastercafile	master CA file
mastercertfile	master CERT file

SECRET NAME	CERTIFICATE FILE
masterkeyfile	master Key file
routingcafile	routing CA file
routingcertfile	routing CERT file
routingkeyfile	routing Key file

Create the secrets using the Azure CLI. Below is an example.

```
az keyvault secret set --vault-name KeyVaultName -n mastercafile --file ~/certificates/masterca.pem
```

Next steps

This article covered the following topics:

- Create a key vault to manage SSH keys for the OpenShift cluster.
- Create a service principal for use by the Azure Cloud Solution Provider.

Next, deploy an OpenShift cluster:

- [Deploy OpenShift Container Platform](#)
- [Deploy OpenShift Container Platform Self-Managed Marketplace Offer](#)

Deploy OpenShift Container Platform in Azure

6/26/2019 • 11 minutes to read • [Edit Online](#)

You can use one of several methods to deploy OpenShift Container Platform in Azure:

- You can manually deploy the necessary Azure infrastructure components and then follow the [OpenShift Container Platform documentation](#).
- You can also use an existing [Resource Manager template](#) that simplifies the deployment of the OpenShift Container Platform cluster.
- Another option is to use the [Azure Marketplace offer](#).

For all options, a Red Hat subscription is required. During the deployment, the Red Hat Enterprise Linux instance is registered to the Red Hat subscription and attached to the Pool ID that contains the entitlements for OpenShift Container Platform. Make sure you have a valid Red Hat Subscription Manager (RHSM) username, password, and Pool ID. You can use an Activation Key, Org ID, and Pool ID. You can verify this information by signing in to <https://access.redhat.com>.

Deploy using the OpenShift Container Platform Resource Manager template

Private Clusters

Deploying private OpenShift clusters requires more than just not having a public IP associated to the master load balancer (web console) or to the infra load balancer (router). A private cluster generally uses a custom DNS server (not the default Azure DNS), a custom domain name (such as contoso.com), and pre-defined virtual network(s). For private clusters, you need to configure your virtual network with all the appropriate subnets and DNS server settings in advance. Then use **existingMasterSubnetReference**, **existingInfraSubnetReference**, **existingCnsSubnetReference**, and **existingNodeSubnetReference** to specify the existing subnet for use by the cluster.

If private master is selected (**masterClusterType**=private), a static private IP needs to be specified for **masterPrivateClusterIp**. This IP will be assigned to the front end of the master load balancer. The IP must be within the CIDR for the master subnet and not in use. **masterClusterDnsType** must be set to "custom" and the master DNS name must be provided for **masterClusterDns**. The DNS name must map to the static Private IP and will be used to access the console on the master nodes.

If private router is selected (**routerClusterType**=private), a static private IP needs to be specified for **routerPrivateClusterIp**. This IP will be assigned to the front end of the infra load balancer. The IP must be within the CIDR for the infra subnet and not in use. **routingSubDomainType** must be set to "custom" and the wildcard DNS name for routing must be provided for **routingSubDomain**.

If private masters and private router are selected, the custom domain name must also be entered for **domainName**

After successful deployment, the Bastion Node is the only node with a public IP that you can ssh into. Even if the master nodes are configured for public access, they aren't exposed for ssh access.

To deploy using the Resource Manager template, you use a parameters file to supply the input parameters. To further customize the deployment, fork the GitHub repo and change the appropriate items.

Some common customization options include, but aren't limited to:

- Bastion VM size (variable in azuredeploy.json)

- Naming conventions (variables in azuredeploy.json)
- OpenShift cluster specifics, modified via hosts file (deployOpenShift.sh)

Configure the parameters file

The [OpenShift Container Platform template](#) has multiple branches available for different versions of OpenShift Container Platform. Based on your needs, you can deploy directly from the repo or you can fork the repo and make custom changes to the templates or scripts before deploying.

Use the `appId` value from the service principal you created earlier for the `aadClientId` parameter.

The following example shows a parameters file named `azuredeploy.parameters.json` with all the required inputs.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "_artifactsLocation": {
      "value": "https://raw.githubusercontent.com/Microsoft/openshift-container-platform/master"
    },
    "location": {
      "value": "eastus"
    },
    "masterVmSize": {
      "value": "Standard_E2s_v3"
    },
    "infraVmSize": {
      "value": "Standard_D4s_v3"
    },
    "nodeVmSize": {
      "value": "Standard_D4s_v3"
    },
    "cnsVmSize": {
      "value": "Standard_E4s_v3"
    },
    "osImageType": {
      "value": "defaultgallery"
    },
    "marketplaceOsImage": {
      "value": {
        "publisher": "RedHat",
        "offer": "RHEL",
        "sku": "7-RAW",
        "version": "latest"
      }
    },
    "storageKind": {
      "value": "changeme"
    },
    "openshiftClusterPrefix": {
      "value": "changeme"
    },
    "minorVersion": {
      "value": "69"
    },
    "masterInstanceCount": {
      "value": 3
    },
    "infraInstanceCount": {
      "value": 3
    },
    "nodeInstanceCount": {
      "value": 3
    },
    "cnsInstanceCount": {
      "value": 3
    }
  }
}
```

```
"osDiskSize": {
    "value": 64
},
"dataDiskSize": {
    "value": 64
},
"cnsGlusterDiskSize": {
    "value": 128
},
"adminUsername": {
    "value": "changeme"
},
"enableMetrics": {
    "value": "false"
},
"enableLogging": {
    "value": "false"
},
"enableCNS": {
    "value": "false"
},
"rhsmUsernameOrOrgId": {
    "value": "changeme"
},
"rhsmPoolId": {
    "value": "changeme"
},
"rhsmBrokerPoolId": {
    "value": "changeme"
},
"sshPublicKey": {
    "value": "GEN-SSH-PUB-KEY"
},
"keyVaultSubscriptionId": {
    "value": "255a325e-8276-4ada-af8f-33af5658eb34"
},
"keyVaultResourceGroup": {
    "value": "changeme"
},
"keyVaultName": {
    "value": "changeme"
},
"enableAzure": {
    "value": "true"
},
"adClientId": {
    "value": "changeme"
},
"domainName": {
    "value": "contoso.com"
},
"masterClusterDnsType": {
    "value": "default"
},
"masterClusterDns": {
    "value": "console.contoso.com"
},
"routingSubDomainType": {
    "value": "nipio"
},
"routingSubDomain": {
    "value": "apps.contoso.com"
},
"virtualNetworkNewOrExisting": {
    "value": "new"
},
"virtualNetworkName": {
    "value": "changeme"
}.
```

```

  },
  "addressPrefixes": {
    "value": "10.0.0.0/14"
  },
  "masterSubnetName": {
    "value": "changeme"
  },
  "masterSubnetPrefix": {
    "value": "10.1.0.0/16"
  },
  "infraSubnetName": {
    "value": "changeme"
  },
  "infraSubnetPrefix": {
    "value": "10.2.0.0/16"
  },
  "nodeSubnetName": {
    "value": "changeme"
  },
  "nodeSubnetPrefix": {
    "value": "10.3.0.0/16"
  },
  "existingMasterSubnetReference": {
    "value": "/subscriptions/abc686f6-963b-4e64-bff4-
99dc369ab1cd/resourceGroups/vnetresourcegroup/providers/Microsoft.Network/virtualNetworks/openshiftvnet/subnet
s/mastersubnet"
  },
  "existingInfraSubnetReference": {
    "value": "/subscriptions/abc686f6-963b-4e64-bff4-
99dc369ab1cd/resourceGroups/vnetresourcegroup/providers/Microsoft.Network/virtualNetworks/openshiftvnet/subnet
s/infrasubnet"
  },
  "existingCnsSubnetReference": {
    "value": "/subscriptions/abc686f6-963b-4e64-bff4-
99dc369ab1cd/resourceGroups/vnetresourcegroup/providers/Microsoft.Network/virtualNetworks/openshiftvnet/subnet
s/cnssubnet"
  },
  "existingNodeSubnetReference": {
    "value": "/subscriptions/abc686f6-963b-4e64-bff4-
99dc369ab1cd/resourceGroups/vnetresourcegroup/providers/Microsoft.Network/virtualNetworks/openshiftvnet/subnet
s/nodesubnet"
  },
  "masterClusterType": {
    "value": "public"
  },
  "masterPrivateClusterIp": {
    "value": "10.1.0.200"
  },
  "routerClusterType": {
    "value": "public"
  },
  "routerPrivateClusterIp": {
    "value": "10.2.0.200"
  },
  "routingCertType": {
    "value": "selfsigned"
  },
  "masterCertType": {
    "value": "selfsigned"
  }
}
}

```

Replace the parameters with your specific information.

Different releases may have different parameters so verify the necessary parameters for the branch you use.

azuredeploy.Parameters.json file explained

PROPERTY	DESCRIPTION	VALID OPTIONS	DEFAULT VALUE
<code>_artifactsLocation</code>	URL for artifacts (json, scripts, etc.)		https://raw.githubusercontent.com/Microsoft/openshift-container-platform/master
<code>location</code>	Azure region to deploy resources to		
<code>masterVmSize</code>	Size of the Master VM. Select from one of the allowed VM sizes listed in the azuredeploy.json file		Standard_E2s_v3
<code>infraVmSize</code>	Size of the Infra VM. Select from one of the allowed VM sizes listed in the azuredeploy.json file		Standard_D4s_v3
<code>nodeVmSize</code>	Size of the App Node VM. Select from one of the allowed VM sizes listed in the azuredeploy.json file		Standard_D4s_v3
<code>cnsVmSize</code>	Size of the Container Native Storage (CNS) Node VM. Select from one of the allowed VM sizes listed in the azuredeploy.json file		Standard_E4s_v3
<code>osImageType</code>	The RHEL image to use. defaultgallery: On-Demand; marketplace: third-party image	defaultgallery marketplace	defaultgallery
<code>marketplaceOsImage</code>	If <code>osImageType</code> is marketplace, then enter the appropriate values for 'publisher', 'offer', 'sku', 'version' of the marketplace offer. This parameter is an object type		
<code>storageKind</code>	The type of storage to be used	managed unmanaged	managed
<code>openshiftClusterPrefix</code>	Cluster Prefix used to configure hostnames for all nodes. Between 1 and 20 characters		mycluster
<code>minoVersion</code>	The minor version of OpenShift Container Platform 3.11 to deploy		69
<code>masterInstanceCount</code>	Number of Masters nodes to deploy	1, 3, 5	3

PROPERTY	DESCRIPTION	VALID OPTIONS	DEFAULT VALUE
<code>infraInstanceCount</code>	Number of infra nodes to deploy	1, 2, 3	3
<code>nodeInstanceCount</code>	Number of Nodes to deploy	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30	2
<code>cnsInstanceCount</code>	Number of CNS nodes to deploy	3, 4	3
<code>osDiskSize</code>	Size of OS disk for the VM (in GB)	64, 128, 256, 512, 1024, 2048	64
<code>dataDiskSize</code>	Size of data disk to attach to nodes for Docker volume (in GB)	32, 64, 128, 256, 512, 1024, 2048	64
<code>cnsGlusterDiskSize</code>	Size of data disk to attach to CNS nodes for use by glusterfs (in GB)	32, 64, 128, 256, 512, 1024, 2048	128
<code>adminUsername</code>	Admin username for both OS (VM) login and initial OpenShift user		ocpadmin
<code>enableMetrics</code>	Enable Metrics. Metrics require more resources so select proper size for Infra VM	true false	false
<code>enableLogging</code>	Enable Logging. elasticsearch pod requires 8 GB RAM so select proper size for Infra VM	true false	false
<code>enableCNS</code>	Enable Container Native Storage	true false	false
<code>rhsmUsernameOrOrgId</code>	Red Hat Subscription Manager Username or Organization ID		
<code>rhsmPoolId</code>	The Red Hat Subscription Manager Pool ID that contains your OpenShift entitlements for compute nodes		

PROPERTY	DESCRIPTION	VALID OPTIONS	DEFAULT VALUE
<code>rhsmBrokerPoolId</code>	The Red Hat Subscription Manager Pool ID that contains your OpenShift entitlements for masters and infra nodes. If you don't have different pool IDs, enter same pool ID as 'rhsmPoolId'		
<code>sshPublicKey</code>	Copy your SSH Public Key here		
<code>keyVaultSubscriptionId</code>	The Subscription ID of the subscription that contains the Key Vault		
<code>keyVaultResourceGroup</code>	The name of the Resource Group that contains the Key Vault		
<code>keyVaultName</code>	The name of the Key Vault you created		
<code>enableAzure</code>	Enable Azure Cloud Provider	true false	true
<code>aadClientId</code>	Azure Active Directory Client ID also known as Application ID for Service Principal		
<code>domainName</code>	Name of the custom domain name to use (if applicable). Set to "none" if not deploying fully private cluster		none
<code>masterClusterDnsType</code>	Domain type for OpenShift web console. 'default' will use DNS label of master infra public IP. 'custom' allows you to define your own name	default custom	default
<code>masterClusterDns</code>	The custom DNS name to use to access the OpenShift web console if you selected 'custom' for <code>masterClusterDnsType</code>		console.contoso.com
<code>routingSubDomainType</code>	If set to 'nipio', <code>routingSubDomain</code> will use nip.io. Use 'custom' if you have your own domain that you want to use for routing	nipio custom	nipio

PROPERTY	DESCRIPTION	VALID OPTIONS	DEFAULT VALUE
<code>routingSubDomain</code>	The wildcard DNS name you want to use for routing if you selected 'custom' for <code>routingSubDomainType</code>		apps.contoso.com
<code>virtualNetworkNewOrExisting</code>	Select whether to use an existing Virtual Network or create a new Virtual Network	existing new	new
<code>virtualNetworkResourceGroupName</code>	Name of the Resource Group for the new Virtual Network if you selected 'new' for <code>virtualNetworkNewOrExisting</code>		resourceGroup().name
<code>virtualNetworkName</code>	The name of the new Virtual Network to create if you selected 'new' for <code>virtualNetworkNewOrExisting</code>		openshiftvnet
<code>addressPrefixes</code>	Address prefix of the new virtual network		10.0.0.0/14
<code>masterSubnetName</code>	The name of the master subnet		mastersubnet
<code>masterSubnetPrefix</code>	CIDR used for the master subnet - needs to be a subset of the addressPrefix		10.1.0.0/16
<code>infraSubnetName</code>	The name of the infra subnet		infrasubnet
<code>infraSubnetPrefix</code>	CIDR used for the infra subnet - needs to be a subset of the addressPrefix		10.2.0.0/16
<code>nodeSubnetName</code>	The name of the node subnet		nodesubnet
<code>nodeSubnetPrefix</code>	CIDR used for the node subnet - needs to be a subset of the addressPrefix		10.3.0.0/16
<code>existingMasterSubnetReference</code>	Full reference to existing subnet for master nodes. Not needed if creating new vNet / Subnet		
<code>existingInfraSubnetReference</code>	Full reference to existing subnet for infra nodes. Not needed if creating new vNet / Subnet		

PROPERTY	DESCRIPTION	VALID OPTIONS	DEFAULT VALUE
<code>existingCnsSubnetReference</code>	Full reference to existing subnet for CNS nodes. Not needed if creating new vNet / Subnet		
<code>existingNodeSubnetReference</code>	Full reference to existing subnet for compute nodes. Not needed if creating new vNet / Subnet		
<code>masterClusterType</code>	Specify whether the cluster uses private or public master nodes. If private is chosen, the master nodes won't be exposed to the Internet via a public IP. Instead, it will use the private IP specified in the <code>masterPrivateClusterIp</code>	public private	public
<code>masterPrivateClusterIp</code>	If private master nodes are selected, then a private IP address must be specified for use by the internal load balancer for master nodes. This static IP must be within the CIDR block for the master subnet and not already in use. If public master nodes are selected, this value won't be used but must still be specified		10.1.0.200
<code>routerClusterType</code>	Specify whether the cluster uses private or public infra nodes. If private is chosen, the infra nodes won't be exposed to the Internet via a public IP. Instead, it will use the private IP specified in the <code>routerPrivateClusterIp</code>	public private	public
<code>routerPrivateClusterIp</code>	If private infra nodes are selected, then a private IP address must be specified for use by the internal load balancer for infra nodes. This static IP must be within the CIDR block for the master subnet and not already in use. If public infra nodes are selected, this value won't be used but must still be specified		10.2.0.200

PROPERTY	DESCRIPTION	VALID OPTIONS	DEFAULT VALUE
<code>routingCertType</code>	Use custom certificate for routing domain or the default self-signed certificate - follow instructions in Custom Certificates section	selfsigned custom	selfsigned
<code>masterCertType</code>	Use custom certificate for master domain or the default self-signed certificate - follow instructions in Custom Certificates section	selfsigned custom	selfsigned

Deploy using Azure CLI

NOTE

The following command requires Azure CLI 2.0.8 or later. You can verify the CLI version with the `az --version` command. To update the CLI version, see [Install Azure CLI](#).

The following example deploys the OpenShift cluster and all related resources into a resource group named `openshiftrg`, with a deployment name of `myOpenShiftCluster`. The template is referenced directly from the GitHub repo, and a local parameters file named `azuredeploy.parameters.json` file is used.

```
az group deployment create -g openshiftrg --name myOpenShiftCluster \
    --template-uri https://raw.githubusercontent.com/Microsoft/openshift-container-
platform/master/azuredeploy.json \
    --parameters @./azuredeploy.parameters.json
```

The deployment takes at least 60 minutes to complete, based on the total number of nodes deployed and options configured. The Bastion DNS FQDN and URL of the OpenShift console prints to the terminal when the deployment finishes.

```
{
  "Bastion DNS FQDN": "bastiondns4hawllzaavu6g.eastus.cloudapp.azure.com",
  "OpenShift Console URL": "http://openshiftlb.eastus.cloudapp.azure.com/console"
}
```

If you don't want to tie up the command line waiting for the deployment to complete, add `--no-wait` as one of the options for the group deployment. The output from the deployment can be retrieved from the Azure portal in the deployment section for the resource group.

Connect to the OpenShift cluster

When the deployment finishes, retrieve the connection from the output section of the deployment. Connect to the OpenShift console with your browser by using the **OpenShift Console URL**. You can also SSH to the Bastion host. Following is an example where the admin username is `clusteradmin` and the bastion public IP DNS FQDN is `bastiondns4hawllzaavu6g.eastus.cloudapp.azure.com`:

```
$ ssh clusteradmin@bastiondns4hawllzaavu6g.eastus.cloudapp.azure.com
```

Clean up resources

Use the [az group delete](#) command to remove the resource group, OpenShift cluster, and all related resources when they're no longer needed.

```
az group delete --name openshiftrg
```

Next steps

- [Post-deployment tasks](#)
- [Troubleshoot OpenShift deployment in Azure](#)
- [Getting started with OpenShift Container Platform](#)

Configure prerequisites

5/7/2019 • 7 minutes to read • [Edit Online](#)

Before using the Marketplace offer to deploy a self-managed OpenShift Container Platform cluster in Azure, a few prerequisites must be configured. Read the [OpenShift prerequisites](#) article for instructions to create an ssh key (without a passphrase), Azure key vault, key vault secret, and a service principal.

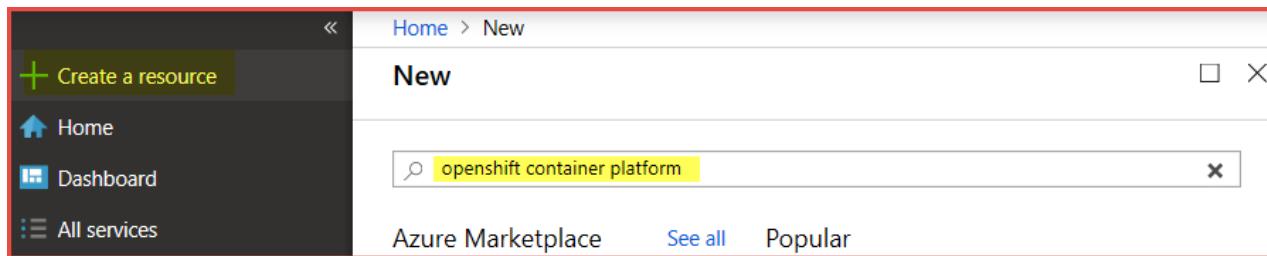
Deploy using the Marketplace offer

The simplest way to deploy a self-managed OpenShift Container Platform cluster into Azure is to use the [Azure Marketplace offer](#).

This option is the simplest, but it also has limited customization capabilities. The Marketplace offer deploys OpenShift Container Platform 3.11.82 and includes the following configuration options:

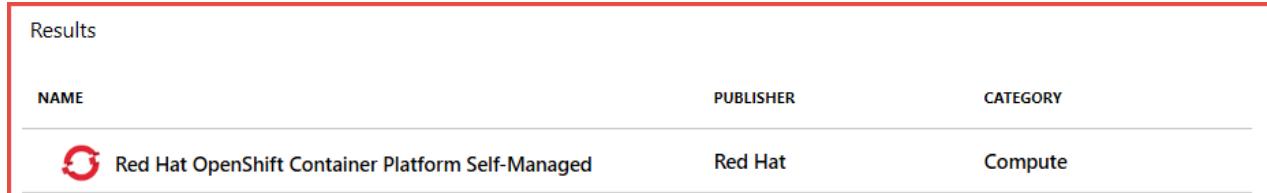
- **Master Nodes:** Three (3) Master Nodes with configurable instance type.
- **Infra Nodes:** Three (3) Infra Nodes with configurable instance type.
- **Nodes:** The number of Nodes (between 1 and 9) and the instance type are configurable.
- **Disk Type:** Managed Disks are used.
- **Networking:** Support for new or existing Network and custom CIDR range.
- **CNS:** CNS can be enabled.
- **Metrics:** Hawkular Metrics can be enabled.
- **Logging:** EFK Logging can be enabled.
- **Azure Cloud Provider:** Enabled by default, can be disabled.

In the upper left of the Azure portal, click **Create a resource**, enter 'openshift container platform' into the search box and hit Enter.



The screenshot shows the Azure portal's search interface. On the left, there's a sidebar with 'Create a resource' (highlighted with a red box), 'Home', 'Dashboard', and 'All services'. The main area has a 'New' tab selected. A search bar at the top contains the text 'openshift container platform'. Below the search bar, there are buttons for 'Azure Marketplace', 'See all', and 'Popular'.

The Results page will open with **Red Hat OpenShift Container Platform Self-Managed** in the list.



The screenshot shows the search results for 'Red Hat OpenShift Container Platform Self-Managed'. The results table has columns for NAME, PUBLISHER, and CATEGORY. One item is listed: 'Red Hat OpenShift Container Platform Self-Managed' by Red Hat, categorized under Compute.

NAME	PUBLISHER	CATEGORY
Red Hat OpenShift Container Platform Self-Managed	Red Hat	Compute

Click the offer to view details of the offer. To deploy this offer, click **Create**. The UI to enter necessary parameters will appear. The first screen is the **Basics** blade.

Red Hat OpenShift Container Platform Self-Managed



Red Hat OpenShift Container Platform Self-Managed

Red Hat

Create

Save for later

This is the Self-Managed offer

Read [Deployment Guide](#) for Azure Marketplace before deploying.

Red Hat OpenShift Container Platform helps organizations develop, deploy, and manage container-based applications seamlessly across physical, virtual, and public cloud infrastructures. OpenShift Container Platform brings application development and IT operations teams together in order to modernize applications, accelerate development processes, and deliver new services faster.

OpenShift Highlights:

- Simple to use with powerful tools for developers
- For traditional, stateful, and cloud-native applications
- Built on proven open source technologies including Red Hat Enterprise Linux, Kubernetes, and Docker
- Enterprise-grade security, compliance, and container management
- Expanded applications support with new and updated runtimes

This offering includes one bastion host, three master nodes, three infrastructure nodes, and a customizable number and size of application nodes.

- The number of application nodes is configurable between one and nine hosts with six options for the size of the virtual machines.
- The bastion host serves as a jump host for access to the OpenShift cluster nodes and system management.

Useful Links

[OpenShift Container Platform Documentation](#)

[OpenShift Container Platform Overview](#)

[OpenShift Container Platform Reference Architecture for Azure](#)

[Red Hat Cloud Access Program](#)

[Red Hat Solutions On Azure](#)

[Deploy OpenShift on Azure](#)

The screenshot shows the 'OPENSHIFT CONTAINER PLATFORM' interface. At the top, there's a navigation bar with 'Web Tier 1 > Add to Project'. Below it is a search bar labeled 'Filter by keyword' and a 'Browse' button. The main area is divided into two sections: 'Instant Apps' and 'xPaaS'. Under 'Instant Apps', there are four items: 'jenkins-ephemeral' (RED HAT JBoss), 'jenkins-persistent' (RED HAT JBoss), 'ruby-helloworld-sample' (RED HAT JBoss Ruby), and 'java-s2i-karaf2-camel-amq' (See all). Under 'xPaaS', there are four items: 'ts-java-openshift:1.0' (RED HAT JBoss Java), 'ts-karaf-openshift:1.0' (RED HAT JBoss Karaf), 'boss-decisionserver63-openshift:1.2' (RED HAT JBoss DecisionServer), and 'boss-decisionserver63-openshift:1.3' (RED HAT JBoss DecisionServer).

Basics

To get help on any of the input parameters, hover over the **i** next to the parameter name.

Enter values for the input parameters and click **OK**.

INPUT PARAMETER	PARAMETER DESCRIPTION
VM Admin User Name	The administrator user to be created on all VM instances
SSH Public Key for Admin User	SSH public key used to log into VM - must not have a passphrase
Subscription	Azure subscription to deploy cluster into
Resource Group	Create a new resource group or select an existing empty resource group for cluster resources
Location	Azure region to deploy cluster into

Create Red Hat OpenShift Co... X
Basics X

1 Basics >
Configure basic settings

2 Infrastructure Settings >
Configure Infrastructure Settings

3 OpenShift Container Plat... >
Configure OpenShift Container Pl...

4 Additional Settings >
Additional Settings

5 Summary >
Red Hat OpenShift Container Plat...

6 Buy >

* VM Admin User Name i

* SSH Public Key for VM Admin User i

Subscription

* Resource group i
 v
[Create new](#)

* Location
 v

Infrastructure Settings

Enter values for the input parameters and click **OK**.

INPUT PARAMETER	PARAMETER DESCRIPTION
OCP Cluster Name Prefix	Cluster Prefix used to configure hostnames for all nodes. Between 1 and 20 characters
Master Node Size	Accept the default VM size or click Change size to select a different VM size. Select appropriate VM size for your work load
Infrastructure Node Size	Accept the default VM size or click Change size to select a different VM size. Select appropriate VM size for your work load
Number of Application Nodes	Accept the default VM size or click Change size to select a different VM size. Select appropriate VM size for your work load
Application Node Size	Accept the default VM size or click Change size to select a different VM size. Select appropriate VM size for your work load
Bastion Host Size	Accept the default VM size or click Change size to select a different VM size. Select appropriate VM size for your work load

INPUT PARAMETER	PARAMETER DESCRIPTION
New or Existing Virtual Network	Create a new vNet (Default) or use an existing vNet
Choose Default CIDR Settings or customize IP Range (CIDR)	Accept default CIDR ranges or Select Custom IP Range and enter custom CIDR information. Default Settings will create vNet with CIDR of 10.0.0.0/14, master subnet with 10.1.0.0/16, infra subnet with 10.2.0.0/16, and compute and cns subnet with 10.3.0.0/16
Key Vault Resource Group Name	The name of the Resource Group that contains the Key Vault
Key Vault Name	The name of the Key Vault that contains the secret with the ssh private key. Only alphanumeric characters and dashes are allowed, and be between 3 and 24 characters
Secret Name	The name of the secret that contains the ssh private key. Only alphanumeric characters and dashes are allowed

Create Red Hat OpenShift Container Platform

Infrastructure Settings

1 Basics Done ✓

2 Infrastructure Settings > Configure Infrastructure Settings

3 OpenShift Container Platform... > Configure OpenShift Container Pl...

4 Additional Settings > Additional Settings

5 Summary > Red Hat OpenShift Container Plat...

6 Buy >

* OCP Cluster Name Prefix ⓘ ocpcluster

* Master Node Size ⓘ 3x Standard D4s v3
4 vcpus, 16 GB memory [Change size](#)

* Infrastructure Node Size ⓘ 3x Standard E2s v3
2 vcpus, 16 GB memory [Change size](#)

Number of Application Nodes ⓘ 3 ✓

* Application Node Size ⓘ 3x Standard D2s v3
2 vcpus, 8 GB memory [Change size](#)

* Bastion Host Size ⓘ 1x Standard DS2 v2
2 vcpus, 7 GB memory [Change size](#)

New or Existing Virtual Network ⓘ
 Default (New) Existing

Choosing the default or to customize the Virtual Network ⓘ
 Default Settings Custom IP Range

* Key Vault Resource Group Name ⓘ keyvaultrg ✓

* Key Vault Name ⓘ keyvault ✓

* Secret Name ⓘ sshprivatekey

Change size

To select a different VM size, click **Change size**. The VM selection window will open. Select the VM size you want and click **Select**.

Showing 4 VM sizes. | Subscription: [REDACTED] | Region: East US | Current size: Standard_D4s_v3

VM SIZE	OFFERING	FAMILY	VCPUS	RAM (GB)	DATA DISKS	MAX IOPS	TEMPORARY STORA...	PREMIUM DISK SUP...	COST/MONTH (ESTI...)
D2s_v3	Standard	General purpose	2	8	4	3200	16 GB	Yes	[REDACTED]
D4s_v3	Standard	General purpose	4	16	8	6400	32 GB	Yes	[REDACTED]
E2s_v3	Standard	Memory optimized	2	16	4	3200	32 GB	Yes	[REDACTED]
E4s_v3	Standard	Memory optimized	4	32	8	6400	64 GB	Yes	[REDACTED]

Existing Virtual Network

INPUT PARAMETER	PARAMETER DESCRIPTION
Existing Virtual Network Name	Name of the existing vNet
Subnet name for master nodes	Name of existing subnet for master nodes. Needs to contain at least 16 IP addresses and follow RFC 1918
Subnet name for infra nodes	Name of existing subnet for infra nodes. Needs to contain at least 32 IP addresses and follow RFC 1918
Subnet name for compute and cns nodes	Name of existing subnet for compute and cns nodes. Needs to contain at least 32 IP addresses and follow RFC 1918
Resource Group for the existing Virtual Network	Name of resource group that contains the existing vNet

New or Existing Virtual Network ⓘ

Default (New) Existing

* Existing Virtual Network Name
ocpvnet ✓

* Subnet name for master nodes ⓘ
mastersubnet ✓

* Subnet name for infra nodes ⓘ
infrasubnet ✓

* Subnet name for compute and cns nodes ⓘ
nodesubnet ✓

* Resource Group for the existing Virtual Network ⓘ
vnetresourcegroup ✓

Custom IP Range

INPUT PARAMETER	PARAMETER DESCRIPTION
Address Range for the Virtual Network	Custom CIDR for the vNet
Address Range for the subnet containing the master nodes	Custom CIDR for master subnet
Address Range for the subnet containing the infrastructure nodes	Custom CIDR for infrastructure subnet
Address Range for subnet containing the compute and cns nodes	Custom CIDR for the compute and cns nodes

Choosing the default or to customize the Virtual Network [?](#)

[Default Settings](#) [Custom IP Range](#)

* Address Range for the VirtualNetwork (default is 10.0.0.0/14) [?](#)
10.0.0.0/16 ✓

* Address Range for the subnet containing the master, infra, and cns nodes (default is 10.1.0.0/16) [?](#)
10.0.10.0/24 ✓

* Address Range for the subnet containing all the infrastructure nodes (default is 10.2.0.0/16) [?](#)
10.0.20.0/24 ✓

* Address Range for the subnet containing all the cns and compute nodes (default is 10.3.0.0/16) [?](#)
10.0.30.0/24 ✓

OpenShift Container Platform

Enter values for the Input Parameters and click **OK**

INPUT PARAMETER	PARAMETER DESCRIPTION
OpenShift Admin User Password	Password for the initial OpenShift user. This user will also be the cluster admin
Confirm OpenShift Admin User Password	Retype the OpenShift Admin User Password
Red Hat Subscription Manager User Name	User Name to access your Red Hat Subscription or Organization ID. This credential is used to register the RHEL instance to your subscription and will not be stored by Microsoft or Red Hat
Red Hat Subscription Manager User Password	Password to access your Red Hat Subscription or Activation Key. This credential is used to register the RHEL instance to your subscription and will not be stored by Microsoft or Red Hat
Red Hat Subscription Manager OpenShift Pool ID	Pool ID that contains OpenShift Container Platform entitlement. Ensure you have enough entitlements of OpenShift Container Platform for the installation of the cluster
Red Hat Subscription Manager OpenShift Pool ID for Broker / Master Nodes	Pool ID that contains OpenShift Container Platform entitlements for Broker / Master Nodes. Ensure you have enough entitlements of OpenShift Container Platform for the installation of the cluster. If not using broker / master pool ID, enter the pool ID for Application Nodes
Configure Azure Cloud Provider	Configure OpenShift to use Azure Cloud Provider. Necessary if using Azure disk attach for persistent volumes. Default is Yes

INPUT PARAMETER	PARAMETER DESCRIPTION
Azure AD Service Principal Client ID GUID	Azure AD Service Principal Client ID GUID - also known as AppID. Only needed if Configure Azure Cloud Provider set to Yes
Azure AD Service Principal Client ID Secret	Azure AD Service Principal Client ID Secret. Only needed if Configure Azure Cloud Provider set to Yes

Create Red Hat OpenShift Container Platform

1 Basics ✓

2 Infrastructure Settings ✓

3 OpenShift Container Platform > Configure OpenShift Container Platform

4 Additional Settings > Additional Settings

5 Summary > Red Hat OpenShift Container Platform

6 Buy >

OpenShift Container Platform

* OpenShift Admin User Password ✓
••••••••

* Confirm OpenShift Admin User Password ✓
••••••••

* Red Hat Subscription Manager User Name ✓
rhsmusername

* Red Hat Subscription Manager User Password ✓
••••••••

* Red Hat Subscription Manager OpenShift Pool ID ✓
8abcd12345e6f7890123abdbe01a000a

* Red Hat Subscription Manager OpenShift Pool ID for Broker / Master Nodes ✓
8abcd12345e6f7890123abdbe01a000b

Configure Azure Cloud Provider ✓

Yes No

* Azure AD Service Principal Client ID GUID ✓
abcd1234-5678-9ef0-89cb-d6da4f6cde12

* Azure AD Service Principal Client ID Secret ✓
••••••••••••••

Additional Settings

The Additional Settings blade allows the configuration of CNS for glusterfs storage, Logging, Metrics, and Router Sub domain. The default won't install any of these options and will use nip.io as the router sub domain for testing purposes. Enabling CNS will install three additional compute nodes with three additional attached disks that will host glusterfs pods.

Enter values for the Input Parameters and click **OK**

INPUT PARAMETER	PARAMETER DESCRIPTION

INPUT PARAMETER	PARAMETER DESCRIPTION
Configure Container Native Storage (CNS)	Installs CNS in the OpenShift cluster and enable it as storage. Will be default if Azure Provider is disabled
Configure Cluster Logging	Installs EFK logging functionality into the cluster. Size infra nodes appropriately to host EFK pods
Configure Metrics for the Cluster	Installs Hawkular metrics into the OpenShift cluster. Size infra nodes appropriately to host Hawkular metrics pods
Default Router Sub domain	Select nipio for testing or custom to enter your own sub domain for production

Create Red Hat OpenShift Co... X

Additional Settings X

1 Basics Done ✓

2 Infrastructure Settings Done ✓

3 OpenShift Container Platfor... Done ✓

4 Additional Settings > Additional Settings

5 Summary > Red Hat OpenShift Container Plat...

6 Buy >

Configure Container Native Storage (CNS) ⓘ

Configure Cluster Logging ⓘ

Configure Metrics for the Cluster ⓘ

Default Router Subdomain ⓘ
 ▾

Additional Settings - Extra Parameters

INPUT PARAMETER	PARAMETER DESCRIPTION
(CNS) Node Size	Accept the default node size or select Change size to select a new VM size
Enter your custom subdomain	The custom routing domain to be used for exposing applications via the router on the OpenShift cluster. Be sure to create the appropriate wildcard DNS entry]

Create Red Hat OpenShift Co... X Additional Settings □ X

1 Basics Done	Configure Container Native Storage (CNS) <small>i</small> <input type="button" value="Yes"/> <input type="button" value="No"/>
2 Infrastructure Settings Done	* CNS Node Size <small>i</small> 3x Standard E4s v3 4 vcpus, 32 GB memory Change size
3 OpenShift Container Plat... Done	Configure Cluster Logging <small>i</small> <input type="button" value="Yes"/> <input type="button" value="No"/>
4 Additional Settings > Additional Settings	Configure Metrics for the Cluster <small>i</small> <input type="button" value="Yes"/> <input type="button" value="No"/>
5 Summary > Red Hat OpenShift Container Plat...	Default Router Subdomain <small>i</small> <input style="width: 200px;" type="text" value="custom"/> <input type="button" value="▼"/>
6 Buy >	* Enter your custom subdomain <small>i</small> <input style="width: 200px;" type="text" value="apps.contoso.com"/> <input type="checkbox"/>

Summary

Validation occurs at this stage to check core quota is sufficient to deploy the total number of VMs selected for the cluster. Review all the parameters that were entered. If the inputs are acceptable, click **OK** to continue.

Create Red Hat OpenShift Co... X
Summary X

- 1** Basics ✓
Done
- 2** Infrastructure Settings ✓
Done
- 3** OpenShift Container Plat... ✓
Done
- 4** Additional Settings ✓
Done
- 5** Summary >
Red Hat OpenShift Container Plat...
- 6** Buy >

Validation passed

Basics	
Subscription	openshift-resourcegroup
Resource group	East US
Location	
VM Admin User Name	clusteradmin
SSH Public Key for VM Admi...	
Infrastructure Settings	
OCP Cluster Name Prefix	ocpcluster
Master Node Size	Standard D4s v3
Infrastructure Node Size	Standard D4s v3
Number of Application Nodes	3
Application Node Size	Standard D4s v3
Bastion Host Size	Standard DS2 v2
New or Existing Virtual Netw...	Default (New)
Choosing the default or to cus...	Default Settings
Key Vault Resource Group N...	
Key Vault Name	sshprivatekey
Secret Name	
OpenShift Container Platform Settings	
OpenShift Admin User Passw...	*****
Red Hat Subscription Manag...	rhmusername
Red Hat Subscription Manag...	*****
Red Hat Subscription Manag...	8abcd12345e6f7890123abdbe01a000a
Red Hat Subscription Manag...	8abcd12345e6f7890123abdbe01a000b
Configure Azure Cloud Provi...	Yes
Azure AD Service Principal Cl...	abcd1234-5678-9ef0-89cb-d6da4f6cde12
Azure AD Service Principal Cl...	*****
Additional Settings	
Configure Container Native S...	Yes
CNS Node Size	Standard E4s v3
Configure Cluster Logging	No
Configure Metrics for the Clu...	No
Default Router Subdomain	nipio

Buy

Confirm contact information on the Buy page and click **Purchase** to accept the terms of use and start deployment of the OpenShift Container Platform cluster.

Create Red Hat OpenShift Co... X

Create		
1	Basics	✓
	Done	
2	Infrastructure Settings	✓
	Done	
3	OpenShift Container Platfor...	✓
	Done	
4	Additional Settings	✓
	Done	
5	Summary	✓
	Red Hat OpenShift Container Plat...	
6	Buy	>

Red Hat OpenShift Container Platform Self-Managed by Red Hat
[Terms of use](#) | [privacy policy](#)

Deploying this template will result in various actions being performed, which may include the deployment of one or more Azure resources or Marketplace offerings and/or transmission of the information you provided as part of the deployment process to one or more parties, as specified in the template. You are responsible for reviewing the text of the template to determine which actions will be performed and which resources or offerings will be deployed, and for locating and reviewing the pricing and legal terms associated with those resources or offerings.

Current retail prices for Azure resources are set forth [here](#) and may not reflect discounts applicable to your Azure subscription.

Prices for Marketplace offerings are set forth [here](#), and the legal terms associated with any Marketplace offering may be found in the Azure portal; both are subject to change at any time prior to deployment.

Neither subscription credits nor monetary commitment funds may be used to purchase non-Microsoft offerings. These purchases are billed separately. If any Microsoft products are included in a Marketplace offering (e.g., Windows Server or SQL Server), such products are licensed by Microsoft and not by any third party.

Template deployment is intended for advanced users only. If you are uncertain which actions will be performed by this template, which resources or offerings will be deployed, or what prices or legal terms pertain to those resources or offerings, do not deploy this template.

Terms of use

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) provided above as well as the legal terms and privacy statement(s) associated with each Marketplace offering that will be deployed using this template, if any; (b) authorize Microsoft to charge or bill my current payment method for the fees associated with my use of the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s); (c) agree that Microsoft may share my contact information and transaction details with any third-party sellers of the offering(s); and (d) give Microsoft permission to share my contact information so that the provider of the template can contact me regarding this product and related products. Microsoft assumes no responsibility for any actions performed by third-party templates and does not provide rights for third-party products or services. See the [Azure Marketplace Terms](#) for additional terms.

By clicking Create, you give Microsoft permission to use or share your account information so that the provider or Microsoft can contact you regarding this product and related products.

Name:

* Preferred e-mail address:

* Preferred phone number:

Create

Connect to the OpenShift cluster

When the deployment finishes, retrieve the connection from the output section of the deployment. Connect to the OpenShift console with your browser by using the **OpenShift Console URL**. You can also SSH to the Bastion host. Following is an example where the admin username is clusteradmin and the bastion public IP DNS FQDN is bastiondns4hawllzaavu6g.eastus.cloudapp.azure.com:

```
$ ssh clusteradmin@bastiondns4hawllzaavu6g.eastus.cloudapp.azure.com
```

Clean up resources

Use the [az group delete](#) command to remove the resource group, OpenShift cluster, and all related resources when they're no longer needed.

```
az group delete --name openshifttrg
```

Next steps

- [Post-deployment tasks](#)
- [Troubleshoot OpenShift deployment in Azure](#)
- [Getting started with OpenShift Container Platform](#)

Deploy OpenShift Container Platform or OKD in Azure Stack

3/5/2019 • 2 minutes to read • [Edit Online](#)

OpenShift can be deployed in Azure Stack. There are some key differences between Azure and Azure Stack so deployment will differ slightly and capabilities will also differ slightly.

Currently, the Azure Cloud Provider doesn't work in Azure Stack. For this reason, you won't be able to use disk attach for persistent storage in Azure Stack. Instead, you can configure other storage options such as NFS, iSCSI, GlusterFS, etc. As an alternative, you can enable CNS and use GlusterFS for persistent storage. If CNS is enabled, three additional nodes will be deployed with additional storage for GlusterFS usage.

You can use one of several methods to deploy OpenShift Container Platform or OKD in Azure Stack:

- You can manually deploy the necessary Azure infrastructure components and then follow the [OpenShift Container Platform documentation](#) or [OKD documentation](#).
- You can also use an existing [Resource Manager template](#) that simplifies the deployment of the OpenShift Container Platform cluster.
- You can also use an existing [Resource Manager template](#) that simplifies the deployment of the OKD cluster.

If using the Resource Manager template, select the proper branch (azurestack-release-3.x). The templates for Azure won't work as the API versions are different between Azure and Azure Stack. The RHEL image reference is currently hard-coded as a variable in the azuredeploy.json file and will need to be changed to match your image.

```
"imageReference": {  
    "publisher": "Redhat",  
    "offer": "RHEL-OCP",  
    "sku": "7-4",  
    "version": "latest"  
}
```

For all options, a Red Hat subscription is required. During the deployment, the Red Hat Enterprise Linux instance is registered to the Red Hat subscription and attached to the Pool ID that contains the entitlements for OpenShift Container Platform. Make sure you have a valid Red Hat Subscription Manager (RHSM) username, password, and Pool ID. Alternatively, you can use an Activation Key, Org ID, and Pool ID. You can verify this information by signing in to <https://access.redhat.com>.

Azure Stack prerequisites

A RHEL image (OpenShift Container Platform) or CentOS image (OKD) needs to be added to your Azure Stack environment to deploy an OpenShift cluster. Contact your Azure Stack administrator to add these images. Instructions can be found here:

- <https://docs.microsoft.com/azure/azure-stack/azure-stack-add-vm-image>
- <https://docs.microsoft.com/azure/azure-stack/azure-stack-marketplace-azure-items>
- <https://docs.microsoft.com/azure/azure-stack/azure-stack-redhat-create-upload-vhd>

Deploy by using the OpenShift Container Platform or OKD Resource Manager template

To deploy by using the Resource Manager template, you use a parameters file to supply the input parameters. To further customize the deployment, fork the GitHub repo and change the appropriate items.

Some common customization options include, but aren't limited to:

- Bastion VM size (variable in azuredeploy.json)
- Naming conventions (variables in azuredeploy.json)
- OpenShift cluster specifics, modified via hosts file (deployOpenShift.sh)
- RHEL image reference (variable in azuredeploy.json)

For the steps to deploy using the Azure CLI, follow the appropriate section in the [OpenShift Container Platform](#) section or the [OKD](#) section.

Next steps

- [Post-deployment tasks](#)
- [Troubleshoot OpenShift deployment in Azure](#)

Post-deployment tasks

4/21/2019 • 3 minutes to read • [Edit Online](#)

After you deploy an OpenShift cluster, you can configure additional items. This article covers:

- How to configure single sign-on by using Azure Active Directory (Azure AD)
- How to configure Azure Monitor logs to monitor OpenShift
- How to configure metrics and logging
- How to install Open Service Broker for Azure (OSBA)

Configure single sign-on by using Azure Active Directory

To use Azure Active Directory for authentication, first you need to create an Azure AD app registration. This process involves two steps: creating the app registration, and configuring permissions.

Create an app registration

These steps use the Azure CLI to create the app registration, and the GUI (portal) to set the permissions. To create the app registration, you need the following five pieces of information:

- Display name: App registration name (for example, OCPAzureAD)
- Home page: OpenShift console URL (for example, <https://masterdns343khhde.westus.cloudapp.azure.com/console>)
- Identifier URI: OpenShift console URL (for example, <https://masterdns343khhde.westus.cloudapp.azure.com/console>)
- Reply URL: Master public URL and the app registration name (for example, <https://masterdns343khhde.westus.cloudapp.azure.com/oauth2callback/OCPAzureAD>)
- Password: Secure password (use a strong password)

The following example creates an app registration by using the preceding information:

```
az ad app create --display-name OCPAzureAD --homepage
https://masterdns343khhde.westus.cloudapp.azure.com/console --reply-urls
https://masterdns343khhde.westus.cloudapp.azure.com/oauth2callback/hwocpadint --identifier-uris
https://masterdns343khhde.westus.cloudapp.azure.com/console --password {Strong Password}
```

If the command is successful, you get a JSON output similar to:

```
{
  "appId": "12345678-ca3c-427b-9a04-ab12345cd678",
  "appPermissions": null,
  "availableToOtherTenants": false,
  "displayName": "OCPAzureAD",
  "homepage": "https://masterdns343khhde.westus.cloudapp.azure.com/console",
  "identifierUris": [
    "https://masterdns343khhde.westus.cloudapp.azure.com/console"
  ],
  "objectId": "62cd74c9-42bb-4b9f-b2b5-b6ee88991c80",
  "objectType": "Application",
  "replyUrls": [
    "https://masterdns343khhde.westus.cloudapp.azure.com/oauth2callback/OCPAzureAD"
  ]
}
```

Take note of the appId property returned from the command for a later step.

In the Azure portal:

1. Select **Azure Active Directory > App Registration**.
2. Search for your app registration (for example, OCPAzureAD).
3. In the results, click the app registration.
4. Under **Settings**, select **Required permissions**.
5. Under **Required Permissions**, select **Add**.

The screenshot shows the Azure portal interface for managing app registrations. On the left, the 'OCPAzureAD' app registration is selected. In the center, the 'Required permissions' section is displayed under the 'API ACCESS' tab. The 'Required permissions' link is highlighted with a red box. On the right, a modal window titled 'Add API access' is open, showing two steps: '1 Select an API' and '2 Select permissions'. Step 1 is also highlighted with a red box.

6. Click Step 1: Select API, and then click **Windows Azure Active Directory (Microsoft.Azure.ActiveDirectory)**. Click **Select** at the bottom.

The screenshot shows the 'Select an API' modal window. It contains a search bar with the placeholder text 'Search for other applications with Service Principal name'. Below the search bar, a list of applications is displayed, with 'Windows Azure Active Directory (Microsoft.Azure.ActiveDirectory)' highlighted by a yellow box. Other items in the list include 'Office 365 Exchange Online (Microsoft.Exchange)' and others which are partially visible.

7. On Step 2: Select Permissions, select **Sign in and read user profile** under **Delegated Permissions**, and then click **Select**.

Enable Access

APPLICATION PERMISSIONS	REQUIRES ADMIN
Read directory data	✓ Yes
Read and write domains	✓ Yes
Read and write directory data	✓ Yes
Read and write devices	✓ Yes
Read all hidden memberships	✓ Yes
Manage apps that this app creates or owns	✓ Yes
Read and write all applications	✓ Yes
Read and write domains	✓ Yes
DELEGATED PERMISSIONS	REQUIRES ADMIN
Access the directory as the signed-in user	✗ No
Read directory data	✓ Yes
Read and write directory data	✓ Yes
Read and write all groups	✓ Yes
Read all groups	✓ Yes
Read all users' full profiles	✓ Yes
Read all users' basic profiles	✗ No
Sign in and read user profile	✗ No
Read hidden memberships	✓ Yes

8. Select **Done**.

Configure OpenShift for Azure AD authentication

To configure OpenShift to use Azure AD as an authentication provider, the /etc/origin/master/master-config.yaml file must be edited on all master nodes.

Find the tenant ID by using the following CLI command:

```
az account show
```

In the yaml file, find the following lines:

```
oauthConfig:
  assetPublicURL: https://masterdns343khhde.westus.cloudapp.azure.com/console/
  grantConfig:
    method: auto
  identityProviders:
    - challenge: true
      login: true
      mappingMethod: claim
      name: htpasswd_auth
      provider:
        apiVersion: v1
        file: /etc/origin/master/htpasswd
        kind: HTPasswdPasswordIdentityProvider
```

Insert the following lines immediately after the preceding lines:

```
- name: <App Registration Name>
  challenge: false
  login: true
  mappingMethod: claim
  provider:
    apiVersion: v1
    kind: OpenIDIdentityProvider
    clientID: <appId>
    clientSecret: <Strong Password>
    claims:
      id:
      - sub
    preferredUsername:
      - unique_name
    name:
      - name
    email:
      - email
  urls:
    authorize: https://login.microsoftonline.com/<tenant Id>/oauth2/authorize
    token: https://login.microsoftonline.com/<tenant Id>/oauth2/token
```

Make sure the text aligns correctly under identityProviders. Find the tenant ID by using the following CLI command: `az account show`

Restart the OpenShift master services on all master nodes:

```
sudo /usr/local/bin/master-restart api
sudo /usr/local/bin/master-restart controllers
```

In the OpenShift console, you now see two options for authentication: htpasswd_auth and [App Registration].

Monitor OpenShift with Azure Monitor logs

There are three ways to add the Log Analytics agent to OpenShift.

- Install the Log Analytics agent for Linux directly on each OpenShift node
- Enable Azure Monitor VM Extension on each OpenShift node
- Install the Log Analytics agent as an OpenShift daemon-set

Read the full [instructions](#) for more details.

Configure metrics and logging

Based on the branch, the Azure Resource Manager templates for OpenShift Container Platform and OKD may provide input parameters for enabling metrics and logging as part of the installation.

The OpenShift Container Platform Marketplace offer also provides an option to enable metrics and logging during cluster installation.

If metrics / logging wasn't enabled during the installation of the cluster, they can easily be enabled after the fact.

Azure Cloud Provider in use

SSH to the bastion node or first master node (based on template and branch in use) using the credentials provided during deployment. Issue the following command:

```
ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml \
-e openshift_metrics_install_metrics=True \
-e openshift_metrics_cassandra_storage_type=dynamic

ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml \
-e openshift_logging_install_logging=True \
-e openshift_logging_es_pvc_dynamic=true
```

Azure Cloud Provider not in use

```
ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml \
-e openshift_metrics_install_metrics=True

ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml \
-e openshift_logging_install_logging=True
```

Install Open Service Broker for Azure (OSBA)

Open Service Broker for Azure, or OSBA, lets you provision Azure Cloud Services directly from OpenShift. OSBA is an Open Service Broker API implementation for Azure. The Open Service Broker API is a spec that defines a common language for cloud providers that cloud native applications can use to manage cloud services without lock-in.

To install OSBA on OpenShift, follow the instructions located here: <https://github.com/Azure/open-service-broker-azure#openshift-project-template>.

NOTE

Only complete the steps in the OpenShift Project Template section and not the entire Installing section.

Next steps

- [Getting started with OpenShift Container Platform](#)

Troubleshoot OpenShift deployment in Azure

4/21/2019 • 4 minutes to read • [Edit Online](#)

If the OpenShift cluster doesn't deploy successfully, the Azure portal will provide error output. The output may be difficult to read which makes it difficult to identify the problem. Quickly scan this output for exit code 3, 4 or 5. The following provides information on these three exit codes:

- Exit code 3: Your Red Hat Subscription User Name / Password or Organization ID / Activation Key is incorrect
- Exit code 4: Your Red Hat Pool ID is incorrect or there are no entitlements available
- Exit code 5: Unable to provision Docker Thin Pool Volume

For all other exit codes, connect to the host(s) via ssh to view the log files.

OpenShift Container Platform

SSH to the ansible playbook host. For the template or the Marketplace offer, use the bastion host. From the bastion, you can SSH to all other nodes in the cluster (master, infra, CNS, compute). You'll need to be root to view the log files. Root is disabled for SSH access by default so don't use root to SSH to other nodes.

OKD

SSH to the ansible playbook host. For the OKD template (version 3.9 and earlier), use the master-0 host. For the OKD template (version 3.10 and later), use the bastion host. From the ansible playbook host, you can SSH to all other nodes in the cluster (master, infra, CNS, compute). You'll need to be root (sudo su -) to view the log files. Root is disabled for SSH access by default so don't use root to SSH to other nodes.

Log files

The log files (stderr and stdout) for the host preparation scripts are located in

`/var/lib/waagent/custom-script/download/0` on all hosts. If an error occurred during the preparation of the host, view these log files to determine the error.

If the preparation scripts ran successfully, then the log files in the `/var/lib/waagent/custom-script/download/1` directory of the ansible playbook host will need to be examined. If the error occurred during the actual installation of OpenShift, the stdout file will display the error. Use this information to contact Support for further assistance.

Example output

```

TASK [openshift_storage_glusterfs : Load heketi topology] ****
fatal: [mycluster-master-0]: FAILED! => {"changed": true, "cmd": ["oc", "--config=/tmp/openshift-glusterfs-ansible-IbhnUM/admin.kubeconfig", "rsh", "--namespace=glusterfs", "deploy-heketi-storage-1-d9x15", "heketi-cli", "-s", "http://localhost:8080", "--user", "admin", "--secret", "VuojURT0/96E42Vv8+XHfsFpSS8R20rH10iMs30qARQ=", "topology", "load", "--json=/tmp/openshift-glusterfs-ansible-IbhnUM/topology.json", "2>&1"], "delta": "0:00:21.477831", "end": "2018-05-20 02:49:11.912899", "failed": true, "failed_when_result": true, "rc": 0, "start": "2018-05-20 02:48:50.435068", "stderr": "", "stderr_lines": [], "stdout": "Creating cluster ... ID: 794b285745b1c5d7089e1c5729ec7cd2\n\tAllowing file volumes on cluster.\n\tAllowing block volumes on cluster.\n\tCreating node mycluster-cns-0 ... ID: 45f1a3bfc20a4196e59ebb567e0e02b4\n\tAdding device /dev/sdd ... OK\n\tAdding device /dev/sde ...\nOK\n\tAdding device /dev/sdf ... OK\n\tCreating node mycluster-cns-1 ... ID: 596f80d7bbd78a1ea548930f23135131\n\tAdding device /dev/sdc ... Unable to add device: Unable to execute command on glusterfs-storage-4zc42: Device /dev/sdc excluded by a filter.\n\tAdding device /dev/sde ...\nOK\n\tAdding device /dev/sdd ... OK\n\tCreating node mycluster-cns-2 ... ID: 42c0170aa2799559747622acceba2e3f\n\tAdding device /dev/sde ... OK\n\tAdding device /dev/sdf ...\nOK\n\tAdding device /dev/sdd ... OK", "stdout_lines": ["Creating cluster ... ID: 794b285745b1c5d7089e1c5729ec7cd2", "\tAllowing file volumes on cluster.", "\tAllowing block volumes on cluster.", "\tCreating node mycluster-cns-0 ... ID: 45f1a3bfc20a4196e59ebb567e0e02b4", "\tAdding device /dev/sdd ... OK", "\tAdding device /dev/sde ... OK", "\tAdding device /dev/sdf ... OK", "\tCreating node mycluster-cns-1 ... ID: 596f80d7bbd78a1ea548930f23135131", "\tAdding device /dev/sdc ... Unable to add device: Unable to execute command on glusterfs-storage-4zc42: Device /dev/sdc excluded by a filter.", "\tAdding device /dev/sde ... OK", "\tAdding device /dev/sdd ... OK", "\tCreating node mycluster-cns-2 ... ID: 42c0170aa2799559747622acceba2e3f", "\tAdding device /dev/sde ... OK", "\tAdding device /dev/sdf ... OK", "\tAdding device /dev/sdd ... OK"]}

PLAY RECAP ****
mycluster-cns-0      : ok=146    changed=57    unreachable=0    failed=0
mycluster-cns-1      : ok=146    changed=57    unreachable=0    failed=0
mycluster-cns-2      : ok=146    changed=57    unreachable=0    failed=0
mycluster-infra-0    : ok=143    changed=55    unreachable=0    failed=0
mycluster-infra-1    : ok=143    changed=55    unreachable=0    failed=0
mycluster-infra-2    : ok=143    changed=55    unreachable=0    failed=0
mycluster-master-0   : ok=502    changed=198   unreachable=0    failed=1
mycluster-master-1   : ok=348    changed=140   unreachable=0    failed=0
mycluster-master-2   : ok=348    changed=140   unreachable=0    failed=0
mycluster-node-0     : ok=143    changed=55    unreachable=0    failed=0
mycluster-node-1     : ok=143    changed=55    unreachable=0    failed=0
localhost            : ok=13     changed=0     unreachable=0    failed=0

INSTALLER STATUS ****
Initialization          : Complete (0:00:39)
Health Check           : Complete (0:00:24)
etcd Install          : Complete (0:01:24)
Master Install         : Complete (0:14:59)
Master Additional Install : Complete (0:01:10)
Node Install           : Complete (0:10:58)
GlusterFS Install     : In Progress (0:03:33)
This phase can be restarted by running: playbooks/openshift-glusterfs/config.yml

Failure summary:

1. Hosts: mycluster-master-0
Play: Configure GlusterFS
Task: Load heketi topology
Message: Failed without returning a message.

```

The most common errors during installation are:

1. Private key has passphrase
2. Key vault secret with private key wasn't created correctly
3. Service principal credentials were entered incorrectly
4. Service principal doesn't have contributor access to the resource group

Private Key has a passphrase

You'll see an error that permission was denied for ssh. ssh to the ansible playbook host to check for a passphrase on the private key.

Key vault secret with private key wasn't created correctly

The private key is copied into the ansible playbook host - `~/.ssh/id_rsa`. Confirm this file is correct. Test by opening an SSH session to one of the cluster nodes from the ansible playbook host.

Service principal credentials were entered incorrectly

When providing the input to the template or Marketplace offer, the incorrect information was provided. Make sure you use the correct appId (clientId) and password (clientSecret) for the service principal. Verify by issuing the following azure cli command.

```
az login --service-principal -u <client id> -p <client secret> -t <tenant id>
```

Service principal doesn't have contributor access to the resource group

If the Azure cloud provider is enabled, then the service principal used must have contributor access to the resource group. Verify by issuing the following azure cli command.

```
az group update -g <openshift resource group> --set tags.sptest=test
```

Additional tools

For some errors, you can also use the following commands to get more information:

1. `systemctl status <service>`
2. `journalctl -xe`

Use Azure to host and run SAP workload scenarios

6/10/2019 • 4 minutes to read • [Edit Online](#)

When you use Microsoft Azure, you can reliably run your mission-critical SAP workloads and scenarios on a scalable, compliant, and enterprise-proven platform. You get the scalability, flexibility, and cost savings of Azure. With the expanded partnership between Microsoft and SAP, you can run SAP applications across development and test and production scenarios in Azure and be fully supported. From SAP NetWeaver to SAP S/4HANA, SAP BI on Linux to Windows, and SAP HANA to SQL, we've got you covered.

Besides hosting SAP NetWeaver scenarios with the different DBMS on Azure, you can host other SAP workload scenarios, like SAP BI on Azure.

The uniqueness of Azure for SAP HANA is an offer that sets Azure apart. To enable hosting more memory and CPU resource-demanding SAP scenarios that involve SAP HANA, Azure offers the use of customer-dedicated bare-metal hardware. Use this solution to run SAP HANA deployments that require up to 24 TB (120-TB scale-out) of memory for S/4HANA or other SAP HANA workload.

Hosting SAP workload scenarios in Azure also can create requirements of identity integration and single sign-on. This situation can occur when you use Azure Active Directory (Azure AD) to connect different SAP components and SAP software-as-a-service (SaaS) or platform-as-a-service (PaaS) offers. A list of such integration and single sign-on scenarios with Azure AD and SAP entities is described and documented in the section "AAD SAP identity integration and single sign-on."

Latest changes

- Introduction of ExpressRoute Fast Path and Global Reach for HANA Large Instances in [SAP HANA \(Large Instances\) network architecture](#) and related documents
- Release of [Azure HANA Large Instances control through the Azure portal](#)
- Release of [high availability for SAP NetWeaver on Azure VMs on SUSE Linux Enterprise Server with Azure NetApp Files for SAP Applications](#)
- Clarification on [Linux OS parameter net.ipv4.tcp_timestamps](#) settings in conjunction with an Azure load balancer

SAP HANA on Azure (Large Instances)

A series of documents leads you through SAP HANA on Azure (Large Instances), or for short, HANA Large Instances. For information on the following areas of HANA Large Instances, see:

- [Overview of SAP HANA on Azure \(Large Instances\)](#)
- [Architecture of SAP HANA on Azure \(Large Instances\)](#)
- [Infrastructure and connectivity to SAP HANA on Azure \(Large Instances\)](#)
- [Install SAP HANA on Azure \(Large Instances\)](#)
- [High availability and disaster recovery of SAP HANA on Azure \(Large Instances\)](#)
- [Troubleshoot and monitor SAP HANA on Azure \(Large Instances\)](#)

Next steps:

- Read [Overview and architecture of SAP HANA on Azure \(Large Instances\)](#)

SAP HANA on Azure virtual machines

This section of the documentation covers different aspects of SAP HANA. As a prerequisite, you should be familiar with the principal services of Azure that provide elementary services of Azure IaaS. So, you need knowledge of Azure compute, storage, and networking. Many of these subjects are handled in the SAP NetWeaver-related [Azure planning guide](#).

For information on HANA on Azure, see the following articles and their subarticles:

- [Quickstart: Manual installation of single-instance SAP HANA on Azure VMs](#)
- [Deploy SAP S/4HANA or BW/4HANA on Azure](#)
- [SAP HANA infrastructure configurations and operations on Azure](#)
- [SAP HANA high availability for Azure virtual machines](#)
- [SAP HANA availability within one Azure region](#)
- [SAP HANA availability across Azure regions](#)
- [High availability of SAP HANA on Azure virtual machines](#)
- [Backup guide for SAP HANA on Azure virtual machines](#)
- [SAP HANA Azure Backup on file level](#)
- [SAP HANA backup based on storage snapshots](#)

SAP NetWeaver deployed on Azure virtual machines

This section lists planning and deployment documentation for SAP NetWeaver and Business One on Azure. The documentation focuses on the basics and the use of non-HANA databases with an SAP workload on Azure. The documents and articles for high availability are also the foundation for HANA high availability in Azure, such as:

- [SAP Business One on Azure virtual machines](#)
- [Deploy SAP IDES EHP7 SP3 for SAP ERP 6.0 on Azure](#)
- [Run SAP NetWeaver on Microsoft Azure SUSE Linux VMs](#)
- [Azure Virtual Machines planning and implementation for SAP NetWeaver](#)
- [Azure Virtual Machines deployment for SAP NetWeaver](#)
- [Protect a multitier SAP NetWeaver application deployment by using Site Recovery](#)
- [SAP LaMa connector for Azure](#)

For information on non-HANA databases under an SAP workload on Azure, see:

- [Considerations for Azure Virtual Machines DBMS deployment for SAP workload](#)
- [SQL Server Azure Virtual Machines DBMS deployment for SAP NetWeaver](#)
- [Oracle Azure Virtual Machines DBMS deployment for SAP workload](#)
- [IBM DB2 Azure Virtual Machines DBMS deployment for SAP workload](#)
- [SAP ASE Azure Virtual Machines DBMS deployment for SAP workload](#)
- [SAP MaxDB, Live Cache, and Content Server deployment on Azure VMs](#)

For information on SAP HANA databases on Azure, see the section "SAP HANA on Azure virtual machines."

For information on high availability of an SAP workload on Azure, see:

- [Azure Virtual Machines high availability for SAP NetWeaver](#)

This document points to various other architecture and scenario documents. In later scenario documents, links to detailed technical documents that explain the deployment and configuration of the different high-availability methods are provided. The different documents that show how to establish and configure high availability for an SAP NetWeaver workload cover Linux and Windows operating systems.

For information on integration between Azure Active Directory (Azure AD) and SAP services and single sign-on, see:

- [Tutorial: Azure Active Directory integration with SAP Cloud for Customer](#)
- [Tutorial: Azure Active Directory integration with SAP Cloud Platform Identity Authentication](#)
- [Tutorial: Azure Active Directory integration with SAP Cloud Platform](#)
- [Tutorial: Azure Active Directory integration with SAP NetWeaver](#)
- [Tutorial: Azure Active Directory integration with SAP Business ByDesign](#)
- [Tutorial: Azure Active Directory integration with SAP HANA](#)
- [Your S/4HANA environment: Fiori Launchpad SAML single sign-on with Azure AD](#)

For information on integration of Azure services into SAP components, see:

- [Use SAP HANA in Power BI Desktop](#)
- [DirectQuery and SAP HANA](#)
- [Use the SAP BW Connector in Power BI Desktop](#)
- [Azure Data Factory offers SAP HANA and Business Warehouse data integration](#)

Create an Oracle Database in an Azure VM

2/4/2019 • 6 minutes to read • [Edit Online](#)

This guide details using the Azure CLI to deploy an Azure virtual machine from the [Oracle marketplace gallery image](#) in order to create an Oracle 12c database. Once the server is deployed, you will connect via SSH in order to configure the Oracle database.

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

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `Powershell` shells to run a variety of tools to work with Azure services.

Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this quickstart requires that you are running the Azure CLI version 2.0.4 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a resource group

Create a resource group with the `az group create` command. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Create virtual machine

To create a virtual machine (VM), use the `az vm create` command.

The following example creates a VM named `myVM`. It also creates SSH keys, if they do not already exist in a default key location. To use a specific set of keys, use the `--ssh-key-value` option.

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--image Oracle:Oracle-Database-Ee:12.1.0.2:latest \
--size Standard_DS2_v2 \
--admin-username azureuser \
--generate-ssh-keys
```

After you create the VM, Azure CLI displays information similar to the following example. Note the value for `publicIpAddress`. You use this address to access the VM.

```
{
  "fqdns": "",
  "id": "/subscriptions/{snip}/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "westus",
  "macAddress": "00-0D-3A-36-2F-56",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "13.64.104.241",
  "resourceGroup": "myResourceGroup"
}
```

Connect to the VM

To create an SSH session with the VM, use the following command. Replace the IP address with the `publicIpAddress` value for your VM.

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

Create the database

The Oracle software is already installed on the Marketplace image. Create a sample database as follows.

1. Switch to the *oracle* superuser, then initialize the listener for logging:

```
$ sudo su - oracle
$ lsnrctl start
```

The output is similar to the following:

```
Copyright (c) 1991, 2014, Oracle. All rights reserved.
```

```
Starting /u01/app/oracle/product/12.1.0/dbhome_1/bin/tnslsnr: please wait...
```

```
TNSLSNR for Linux: Version 12.1.0.2.0 - Production
Log messages written to /u01/app/oracle/diag/tnslsnr/myVM/listener/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=myVM.twltkue3xvsujaz1bvlrhfuiwf.dx.internal.cloudapp.net)(PORT=1521)))
```

```
Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
```

```
STATUS of the LISTENER
```

```
-----
Alias           LISTENER
Version        TNSLSNR for Linux: Version 12.1.0.2.0 - Production
Start Date     23-MAR-2017 15:32:08
Uptime         0 days 0 hr. 0 min. 0 sec
Trace Level    off
Security       ON: Local OS Authentication
SNMP           OFF
Listener Log File /u01/app/oracle/diag/tnslsnr/myVM/listener/alert/log.xml
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myVM.twltkue3xvsujaz1bvlrhfuiwf.dx.internal.cloudapp.net)
(PORT=1521)))
The listener supports no services
The command completed successfully
```

2. Create the database:

```
dbca -silent \
      -createDatabase \
      -templateName General_Purpose.dbc \
      -gdbname cdb1 \
      -sid cdb1 \
      -responseFile NO_VALUE \
      -characterSet AL32UTF8 \
      -sysPassword OraPasswd1 \
      -systemPassword OraPasswd1 \
      -createAsContainerDatabase true \
      -numberOfPDBs 1 \
      -pdbName pdb1 \
      -pdbAdminPassword OraPasswd1 \
      -databaseType MULTIPURPOSE \
      -automaticMemoryManagement false \
      -storageType FS \
      -ignorePreReqs
```

It takes a few minutes to create the database.

3. Set Oracle variables

Before you connect, you need to set two environment variables: *ORACLE_HOME* and *ORACLE_SID*.

```
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1; export ORACLE_HOME
ORACLE_SID=cdb1; export ORACLE_SID
```

You also can add *ORACLE_HOME* and *ORACLE_SID* variables to the .bashrc file. This would save the environment variables for future sign-ins. Confirm the following statements have been added to the `~/.bashrc` file using editor of your choice.

```
# Add ORACLE_HOME.  
export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1  
# Add ORACLE_SID.  
export ORACLE_SID=cdb1
```

Oracle EM Express connectivity

For a GUI management tool that you can use to explore the database, set up Oracle EM Express. To connect to Oracle EM Express, you must first set up the port in Oracle.

1. Connect to your database using sqlplus:

```
sqlplus / as sysdba
```

2. Once connected, set the port 5502 for EM Express

```
exec DBMS_XDB_CONFIG.SETHTTPSPORT(5502);
```

3. Open the container PDB1 if not already opened, but first check the status:

```
select con_id, name, open_mode from v$pdbs;
```

The output is similar to the following:

CON_ID	NAME	OPEN_MODE
2	PDB\$SEED	READ ONLY
3	PDB1	MOUNT

4. If the OPEN_MODE for **PDB1** is not READ WRITE, then run the followings commands to open PDB1:

```
alter session set container=pdb1;  
alter database open;
```

You need to type **quit** to end the sqlplus session and type **exit** to logout of the oracle user.

Automate database startup and shutdown

The Oracle database by default doesn't automatically start when you restart the VM. To set up the Oracle database to start automatically, first sign in as root. Then, create and update some system files.

1. Sign on as root

```
sudo su -
```

2. Using your favorite editor, edit the file **/etc/oratab** and change the default **N** to **Y**:

```
cdb1:/u01/app/oracle/product/12.1.0/dbhome_1:Y
```

3. Create a file named **/etc/init.d/dbora** and paste the following contents:

```

#!/bin/sh
# chkconfig: 345 99 10
# Description: Oracle auto start-stop script.
#
# Set ORA_HOME to be equivalent to $ORACLE_HOME.
ORA_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
ORA_OWNER=oracle

case "$1" in
'start')
    # Start the Oracle databases:
    # The following command assumes that the Oracle sign-in
    # will not prompt the user for any values.
    # Remove "&" if you don't want startup as a background process.
    su - $ORA_OWNER -c "$ORA_HOME/bin/dbstart $ORA_HOME" &
    touch /var/lock/subsys/dbora
    ;;
'stop')
    # Stop the Oracle databases:
    # The following command assumes that the Oracle sign-in
    # will not prompt the user for any values.
    su - $ORA_OWNER -c "$ORA_HOME/bin/dbshut $ORA_HOME" &
    rm -f /var/lock/subsys/dbora
    ;;
esac

```

4. Change permissions on files with *chmod* as follows:

```

chgrp dba /etc/init.d/dbora
chmod 750 /etc/init.d/dbora

```

5. Create symbolic links for startup and shutdown as follows:

```

ln -s /etc/init.d/dbora /etc/rc.d/rc0.d/K01dbora
ln -s /etc/init.d/dbora /etc/rc.d/rc3.d/S99dbora
ln -s /etc/init.d/dbora /etc/rc.d/rc5.d/S99dbora

```

6. To test your changes, restart the VM:

```

reboot

```

Open ports for connectivity

The final task is to configure some external endpoints. To set up the Azure Network Security Group that protects the VM, first exit your SSH session in the VM (should have been kicked out of SSH when rebooting in previous step).

- To open the endpoint that you use to access the Oracle database remotely, create a Network Security Group rule with [az network nsg rule create](#) as follows:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myVmNSG \
--name allow-oracle \
--protocol tcp \
--priority 1001 \
--destination-port-range 1521
```

2. To open the endpoint that you use to access Oracle EM Express remotely, create a Network Security Group rule with [az network nsg rule create](#) as follows:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myVmNSG \
--name allow-oracle-EM \
--protocol tcp \
--priority 1002 \
--destination-port-range 5502
```

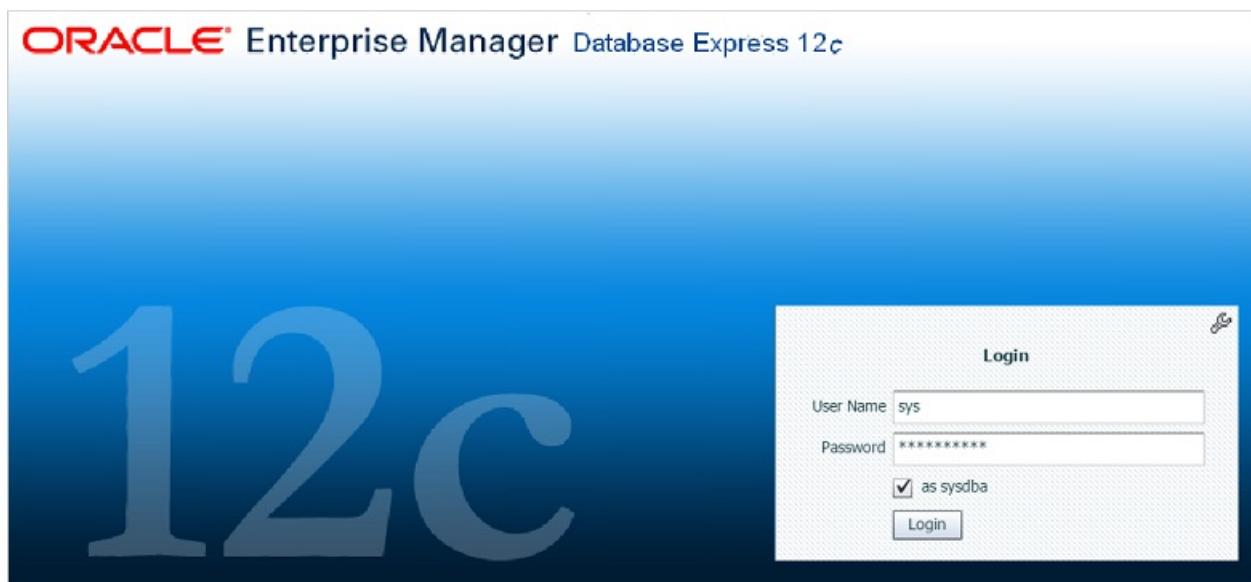
3. If needed, obtain the public IP address of your VM again with [az network public-ip show](#) as follows:

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

4. Connect EM Express from your browser. Make sure your browser is compatible with EM Express (Flash install is required):

```
https://<VM ip address or hostname>:5502/em
```

You can log in by using the **SYS** account, and check the **as sysdba** checkbox. Use the password **OraPasswd1** that you set during installation.



Clean up resources

Once you have finished exploring your first Oracle database on Azure and the VM is no longer needed, you can use the [az group delete](#) command to remove the resource group, VM, and all related resources.

```
az group delete --name myResourceGroup
```

Next steps

Learn about other Oracle solutions on Azure.

Try the [Installing and Configuring Oracle Automated Storage Management](#) tutorial.

Install the Elastic Stack on an Azure VM

2/4/2019 • 5 minutes to read • [Edit Online](#)

This article walks you through how to deploy [Elasticsearch](#), [Logstash](#), and [Kibana](#), on an Ubuntu VM in Azure. To see the Elastic Stack in action, you can optionally connect to Kibana and work with some sample logging data.

In this tutorial you learn how to:

- Create an Ubuntu VM in an Azure resource group
- Install Elasticsearch, Logstash, and Kibana on the VM
- Send sample data to Elasticsearch with Logstash
- Open ports and work with data in the Kibana console

This deployment is suitable for basic development with the Elastic Stack. For more on the Elastic Stack, including recommendations for a production environment, see the [Elastic documentation](#) and the [Azure Architecture Center](#).

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `PowerShell` shells to run a variety of tools to work with Azure services.

Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.4 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI](#).

Create a resource group

Create a resource group with the `az group create` command. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named `myResourceGroup` in the `eastus` location.

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

Create a virtual machine

Create a VM with the [az vm create](#) command.

The following example creates a VM named *myVM* and creates SSH keys if they do not already exist in a default key location. To use a specific set of keys, use the `--ssh-key-value` option.

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

When the VM has been created, the Azure CLI shows information similar to the following example. Take note of the `publicIpAddress`. This address is used to access the VM.

```
{
  "fqdns": "",
  "id": "/subscriptions/<subscription
ID>/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "40.68.254.142",
  "resourceGroup": "myResourceGroup"
}
```

SSH into your VM

If you don't already know the public IP address of your VM, run the [az network public-ip list](#) command:

```
az network public-ip list --resource-group myResourceGroup --query [].ipAddress
```

Use the following command to create an SSH session with the virtual machine. Substitute the correct public IP address of your virtual machine. In this example, the IP address is *40.68.254.142*.

```
ssh azureuser@40.68.254.142
```

Install the Elastic Stack

Import the Elasticsearch signing key and update your APT sources list to include the Elastic package repository:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a
/etc/apt/sources.list.d/elastic-5.x.list
```

Install the Java Virtual on the VM and configure the JAVA_HOME variable-this is necessary for the Elastic Stack components to run.

```
sudo apt update && sudo apt install openjdk-8-jre-headless
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Run the following commands to update Ubuntu package sources and install Elasticsearch, Kibana, and Logstash.

```
sudo apt update && sudo apt install elasticsearch kibana logstash
```

NOTE

Detailed installation instructions, including directory layouts and initial configuration, are maintained in [Elastic's documentation](#)

Start Elasticsearch

Start Elasticsearch on your VM with the following command:

```
sudo systemctl start elasticsearch.service
```

This command produces no output, so verify that Elasticsearch is running on the VM with this `curl` command:

```
sudo curl -XGET 'localhost:9200/'
```

If Elasticsearch is running, you see output like the following:

```
{
  "name" : "w6Z4NwR",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "SDzCajBoSK2EkXmHvJVaDQ",
  "version" : {
    "number" : "5.6.3",
    "build_hash" : "1a2f265",
    "build_date" : "2017-10-06T20:33:39.012Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.1"
  },
  "tagline" : "You Know, for Search"
}
```

Start Logstash and add data to Elasticsearch

Start Logstash with the following command:

```
sudo systemctl start logstash.service
```

Test Logstash in interactive mode to make sure it's working correctly:

```
sudo /usr/share/logstash/bin/logstash -e 'input { stdin { } } output { stdout {} }'
```

This is a basic logstash [pipeline](#) that echoes standard input to standard output.

```
The stdin plugin is now waiting for input:
hello azure
2017-10-11T20:01:08.904Z myVM hello azure
```

Set up Logstash to forward the kernel messages from this VM to Elasticsearch. Create a new file in an empty directory called `vm-syslog-logstash.conf` and paste in the following Logstash configuration:

```
input {
    stdin {
        type => "stdin-type"
    }

    file {
        type => "syslog"
        path => [ "/var/log/*.log", "/var/log/*/*.log", "/var/log/messages", "/var/log/syslog" ]
        start_position => "beginning"
    }
}

output {
    stdout {
        codec => rubydebug
    }
    elasticsearch {
        hosts => "localhost:9200"
    }
}
```

Test this configuration and send the syslog data to Elasticsearch:

```
sudo /usr/share/logstash/bin/logstash -f vm-syslog-logstash.conf
```

You see the syslog entries in your terminal echoed as they are sent to Elasticsearch. Use `CTRL+C` to exit out of Logstash once you've sent some data.

Start Kibana and visualize the data in Elasticsearch

Edit `/etc/kibana/kibana.yml` and change the IP address Kibana listens on so you can access it from your web browser.

```
server.host:"0.0.0.0"
```

Start Kibana with the following command:

```
sudo systemctl start kibana.service
```

Open port 5601 from the Azure CLI to allow remote access to the Kibana console:

```
az vm open-port --port 5601 --resource-group myResourceGroup --name myVM
```

Open up the Kibana console and select **Create** to generate a default index based on the syslog data you sent to Elasticsearch earlier.

Management / Kibana

Index Patterns Saved Objects Advanced Settings

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index pattern [advanced options](#)

logstash-*

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

Time Filter field name [refresh fields](#)

@timestamp

Expand index pattern when searching [DEPRECATED]

With this option selected, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern logstash-* will actually query Elasticsearch for the specific matching indices (e.g. logstash-2015.12.21) that fall within the current time range.

With recent changes to Elasticsearch, this option should no longer be necessary and will likely be removed in future versions of Kibana.

Use event times to create index names [DEPRECATED]

Create

Select **Discover** on the Kibana console to search, browse, and filter through the syslog events.

Kibana

12 hits

Search... (e.g. status:200 AND extension:PHP)

Uses lucene query syntax

Actions ▾

logstash-*

Selected Fields

⌚ @timestamp
t @version
t message
t type

Available Fields

t _id
t _index
_score
t _type
t host
t path

Time	@timestamp	@version	message	type
October 13th 2017, 14:05:25.686	October 13th 2017, 14:05:25.686	1	[2017-10-13T21:05:24,910] [WARN] syslog [o.e.d.i.m.TypeParsers] field [include_in_all] is deprecated, as [__all] is deprecated, and will be disallowed in 6.0, use [copy_to] instead.	syslog
October 13th 2017, 14:05:25.687	October 13th 2017, 14:05:25.687	1	[2017-10-13T21:05:24,907] [WARN] syslog [o.e.d.i.m.TypeParsers] field [include_in_all] is deprecated, as [__all] is deprecated, and will be disallowed in 6.0, use [copy_to] instead.	syslog
October 13th 2017, 14:05:25.686	October 13th 2017, 14:05:25.686	1	[2017-10-13T21:05:24,889] [WARN] syslog [o.e.d.i.m.TypeParsers] field [include_in_all] is	syslog

Next steps

In this tutorial, you deployed the Elastic Stack into a development VM in Azure. You learned how to:

- Create an Ubuntu VM in an Azure resource group
- Install Elasticsearch, Logstash, and Kibana on the VM
- Send sample data to Elasticsearch from Logstash
- Open ports and work with data in the Kibana console

How to use FreeBSD's Packet Filter to create a secure firewall in Azure

2/4/2019 • 2 minutes to read • [Edit Online](#)

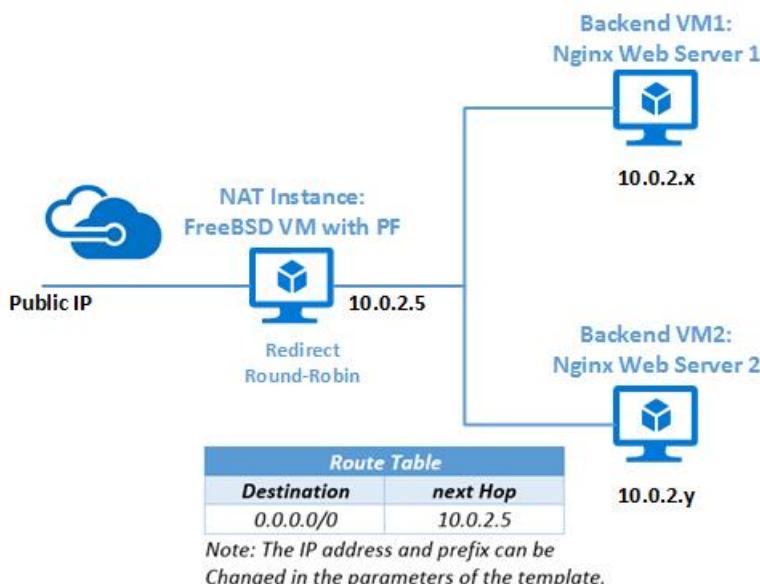
This article introduces how to deploy a NAT firewall using FreeBSD's Packer Filter through Azure Resource Manager template for common web server scenario.

What is PF?

PF (Packet Filter, also written pf) is a BSD licensed stateful packet filter, a central piece of software for firewalling. PF has since evolved quickly and now has several advantages over other available firewalls. Network Address Translation (NAT) is in PF since day one, then packet scheduler and active queue management have been integrated into PF, by integrating the ALTQ and making it configurable through PF's configuration. Features such as pfsync and CARP for failover and redundancy, authpf for session authentication, and ftp-proxy to ease firewalling the difficult FTP protocol, have also extended PF. In short, PF is a powerful and feature-rich firewall.

Get started

If you are interested in setting up a secure firewall in the cloud for your web servers, then let's get started. You can also apply the scripts used in this Azure Resource Manager template to set up your networking topology. The Azure Resource Manager template set up a FreeBSD virtual machine that performs NAT /redirection using PF and two FreeBSD virtual machines with the Nginx web server installed and configured. In addition to performing NAT for the two web servers egress traffic, the NAT/ redirection virtual machine intercepts HTTP requests and redirect them to the two web servers in round-robin fashion. The VNet uses the private non-routable IP address space 10.0.0.2/24 and you can modify the parameters of the template. The Azure Resource Manager template also defines a route table for the whole VNet, which is a collection of individual routes used to override Azure default routes based on the destination IP address.



Deploy through Azure CLI

You need the latest [Azure CLI](#) installed and logged in to an Azure account using [az login](#). Create a resource group with [az group create](#). The following example creates a resource group name `myResourceGroup` in the `West US` location.

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

Next, deploy the template [pf-freebsd-setup](#) with `az group deployment create`. Download `azuredeploy.parameters.json` under the same path and define your own resource values, such as `adminPassword`, `networkPrefix`, and `domainNamePrefix`.

```
az group deployment create --resource-group myResourceGroup --name myDeploymentName \
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/pf-freebsd-
setup/azuredeploy.json \
--parameters '@azuredeploy.parameters.json' --verbose
```

After about five minutes, you will get the information of `"provisioningState": "Succeeded"`. Then you can ssh to the frontend VM (NAT) or access Nginx web server in a browser using the public IP address or FQDN of the frontend VM (NAT). The following example lists FQDN and public IP address that assigned to the frontend VM (NAT) in the `myResourceGroup` resource group.

```
az network public-ip list --resource-group myResourceGroup
```

Next steps

Do you want to set up your own NAT in Azure? Open Source, free but powerful? Then PF is a good choice. By using the template [pf-freebsd-setup](#), you only need five minutes to set up a NAT firewall with round-robin load balancing using FreeBSD's PF in Azure for common web server scenario.

If you want to learn the offering of FreeBSD in Azure, refer to [introduction to FreeBSD on Azure](#).

If you want to know more about PF, refer to [FreeBSD handbook](#) or [PF-User's Guide](#).

Install MySQL on a virtual machine running OpenSUSE Linux in Azure

3/14/2019 • 3 minutes to read • [Edit Online](#)

MySQL is a popular, open-source SQL database. This tutorial shows you how to create a virtual machine running OpenSUSE Linux, then install MySQL.

Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either `bash` or `Powershell` shells to run a variety of tools to work with Azure services.

Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. This doesn't automatically copy text to Cloud Shell.	
Open Azure Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

If you choose to install and use the CLI locally, you need Azure CLI version 2.0 or later. To find the version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI](#).

Create a virtual machine running OpenSUSE Linux

First, create a resource group. In this example, the resource group is named `mySQLSUSEResourceGroup` and it is created in the *East US* region.

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

Create the VM. In this example, the VM is named `myVM` and the VM size is `Standard_D2s_v3`, but you should choose the [VM size](#) you think is most appropriate for your workload.

```
az vm create --resource-group mySQLSUSEResourceGroup \
  --name myVM \
  --image openSUSE-Leap \
  --size Standard_D2s_v3 \
  --generate-ssh-keys
```

You also need to add a rule to the network security group to allow traffic over port 3306 for MySQL.

```
az vm open-port --port 3306 --resource-group mySQLSUSEResourceGroup --name myVM
```

Connect to the VM

You'll use SSH to connect to the VM. In this example, the public IP address of the VM is 10.111.112.113. You can see the IP address in the output when you created the VM.

```
ssh 10.111.112.113
```

Update the VM

After you're connected to the VM, install system updates and patches.

```
sudo zypper update
```

Follow the prompts to update your VM.

Install MySQL

Install the MySQL in the VM over SSH. Reply to prompts as appropriate.

```
sudo zypper install mysql
```

Set MySQL to start when the system boots.

```
sudo systemctl enable mysql
```

Verify that MySQL is enabled.

```
systemctl is-enabled mysql
```

This should return: enabled.

Restart the server.

```
sudo reboot
```

MySQL password

After installation, the MySQL root password is empty by default. Run the **mysql_secure_installation** script to secure MySQL. The script prompts you to change the MySQL root password, remove anonymous user accounts, disable remote root sign in, remove test databases, and reload the privileges table.

Once the server reboots, ssh to the VM again.

```
ssh 10.111.112.113
```

```
mysql_secure_installation
```

Sign in to MySQL

You can now sign in and enter the MySQL prompt.

```
mysql -u root -p
```

This switches you to the MySQL prompt where you can issue SQL statements to interact with the database.

Now, create a new MySQL user.

```
CREATE USER 'mysqluser'@'localhost' IDENTIFIED BY 'password';
```

The semi-colon (;) at the end of the line is crucial for ending the command.

Create a database

Create a database and grant the `mysqluser` user permissions.

```
CREATE DATABASE testdatabase;
GRANT ALL ON testdatabase.* TO 'mysqluser'@'localhost' IDENTIFIED BY 'password';
```

Database user names and passwords are only used by scripts connecting to the database. Database user account names do not necessarily represent actual user accounts on the system.

Enable sign in from another computer. In this example, the IP address of the computer to allow sign in from is `10.112.113.114`.

```
GRANT ALL ON testdatabase.* TO 'mysqluser'@'10.112.113.114' IDENTIFIED BY 'password';
```

To exit the MySQL database administration utility, type:

```
quit
```

Next steps

For details about MySQL, see the [MySQL Documentation](#).

How to install and configure MongoDB on a Linux VM

3/14/2019 • 5 minutes to read • [Edit Online](#)

MongoDB is a popular open-source, high-performance NoSQL database. This article shows you how to install and configure MongoDB on a Linux VM with the Azure CLI. Examples are shown that detail how to:

- [Manually install and configure a basic MongoDB instance](#)
- [Create a basic MongoDB instance using a Resource Manager template](#)
- [Create a complex MongoDB sharded cluster with replica sets using a Resource Manager template](#)

Manually install and configure MongoDB on a VM

MongoDB [provide installation instructions](#) for Linux distros including Red Hat / CentOS, SUSE, Ubuntu, and Debian. The following example creates a *CentOS* VM. To create this environment, you need the latest [Azure CLI](#) installed and logged in to an Azure account using [az login](#).

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
```

Create a VM with [az vm create](#). The following example creates a VM named *myVM* with a user named *azureuser* using SSH public key authentication

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

SSH to the VM using your own username and the `publicIpAddress` listed in the output from the previous step:

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

To add the installation sources for MongoDB, create a **yum** repository file as follows:

```
sudo touch /etc/yum.repos.d/mongodb-org-3.6.repo
```

Open the MongoDB repo file for editing, such as with `vi` or `nano`. Add the following lines:

```
[mongodb-org-3.6]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.6/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.6.asc
```

Install MongoDB using **yum** as follows:

```
sudo yum install -y mongodb-org
```

By default, SELinux is enforced on CentOS images that prevents you from accessing MongoDB. Install policy management tools and configure SELinux to allow MongoDB to operate on its default TCP port 27017 as follows:

```
sudo yum install -y policycoreutils-python  
sudo semanage port -a -t mongod_port_t -p tcp 27017
```

Start the MongoDB service as follows:

```
sudo service mongod start
```

Verify the MongoDB installation by connecting using the local `mongo` client:

```
mongo
```

Now test the MongoDB instance by adding some data and then searching:

```
> db  
test  
> db.foo.insert( { a : 1 } )  
> db.foo.find()  
{ "_id" : ObjectId("57ec477cd639891710b90727"), "a" : 1 }  
> exit
```

If desired, configure MongoDB to start automatically during a system reboot:

```
sudo chkconfig mongod on
```

Create basic MongoDB instance on CentOS using a template

You can create a basic MongoDB instance on a single CentOS VM using the following Azure quickstart template from GitHub. This template uses the Custom Script extension for Linux to add a **yum** repository to your newly created CentOS VM and then install MongoDB.

- Basic MongoDB instance on CentOS - <https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/mongodb-on-centos/azuredploy.json>

To create this environment, you need the latest [Azure CLI](#) installed and logged in to an Azure account using `az login`. 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
```

Next, deploy the MongoDB template with `az group deployment create`. When prompted, enter your own unique values for `newStorageAccountName`, `dnsNameForPublicIP`, and admin username and password:

```
az group deployment create --resource-group myResourceGroup \
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/mongodb-on-
centos/azuredeploy.json
```

Log on to the VM using the public DNS address of your VM. You can view the public DNS address with [az vm show](#):

```
az vm show -g myResourceGroup -n myLinuxVM -d --query [fqdns] -o tsv
```

SSH to your VM using your own username and public DNS address:

```
ssh azureuser@mypublicdns.eastus.cloudapp.azure.com
```

Verify the MongoDB installation by connecting using the local `mongo` client as follows:

```
mongo
```

Now test the instance by adding some data and searching as follows:

```
> db
test
> db.foo.insert( { a : 1 } )
> db.foo.find()
{ "_id" : ObjectId("57ec477cd639891710b90727"), "a" : 1 }
> exit
```

Create a complex MongoDB Sharded Cluster on CentOS using a template

You can create a complex MongoDB sharded cluster using the following Azure quickstart template from GitHub. This template follows the [MongoDB sharded cluster best practices](#) to provide redundancy and high availability. The template creates two shards, with three nodes in each replica set. One config server replica set with three nodes is also created, plus two **mongos** router servers to provide consistency to applications from across the shards.

- [MongoDB Sharding Cluster on CentOS](#) - <https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/mongodb-sharding-centos/azuredeploy.json>

WARNING

Deploying this complex MongoDB sharded cluster requires more than 20 cores, which is typically the default core count per region for a subscription. Open an Azure support request to increase your core count.

To create this environment, you need the latest [Azure CLI](#) installed and logged in to an Azure account using [az login](#). 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
```

Next, deploy the MongoDB template with [az group deployment create](#). Define your own resource names and sizes where needed such as for *mongoAdminUsername*, *sizeOfDataDiskInGB*, and *configNodeVmSize*:

```
az group deployment create --resource-group myResourceGroup \
--parameters '{
    "adminUsername": {"value": "azureuser"},  

    "adminPassword": {"value": "P@ssw0rd!"},  

    "mongoAdminUsername": {"value": "mongoadmin"},  

    "mongoAdminPassword": {"value": "P@ssw0rd!"},  

    "dnsNamePrefix": {"value": "mypublicdns"},  

    "environment": {"value": "AzureCloud"},  

    "numDataDisks": {"value": "4"},  

    "sizeOfDataDiskInGB": {"value": 20},  

    "centOsVersion": {"value": "7.0"},  

    "routerNodeVmSize": {"value": "Standard_DS3_v2"},  

    "configNodeVmSize": {"value": "Standard_DS3_v2"},  

    "replicaNodeVmSize": {"value": "Standard_DS3_v2"},  

    "zabbixServerIPAddress": {"value": "Null"} }' \
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/mongodb-sharding-
centos/azuredeploy.json \
--name myMongoDBCluster \
--no-wait
```

This deployment can take over an hour to deploy and configure all the VM instances. The `--no-wait` flag is used at the end of the preceding command to return control to the command prompt once the template deployment has been accepted by the Azure platform. You can then view the deployment status with [az group deployment show](#). The following example views the status for the *myMongoDBCluster* deployment in the *myResourceGroup* resource group:

```
az group deployment show \
--resource-group myResourceGroup \
--name myMongoDBCluster \
--query [properties.provisioningState] \
--output tsv
```

Next steps

In these examples, you connect to the MongoDB instance locally from the VM. If you want to connect to the MongoDB instance from another VM or network, ensure the appropriate [Network Security Group rules are created](#).

These examples deploy the core MongoDB environment for development purposes. Apply the required security configuration options for your environment. For more information, see the [MongoDB security docs](#).

For more information about creating using templates, see the [Azure Resource Manager overview](#).

The Azure Resource Manager templates use the Custom Script Extension to download and execute scripts on your VMs. For more information, see [Using the Azure Custom Script Extension with Linux Virtual Machines](#).

Install and configure PostgreSQL on Azure

5/20/2019 • 5 minutes to read • [Edit Online](#)

PostgreSQL is an advanced open-source database similar to Oracle and DB2. It includes enterprise-ready features such as full ACID compliance, reliable transactional processing, and multi-version concurrency control. It also supports standards such as ANSI SQL and SQL/MED (including foreign data wrappers for Oracle, MySQL, MongoDB, and many others). It is highly extensible with support for over 12 procedural languages, GIN and GiST indexes, spatial data support, and multiple NoSQL-like features for JSON or key-value-based applications.

In this article, you will learn how to install and configure PostgreSQL on an Azure virtual machine running Linux.

NOTE

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using both models, but Microsoft recommends that most new deployments use the Resource Manager model.

Install PostgreSQL

NOTE

You must already have an Azure virtual machine running Linux in order to complete this tutorial. To create and set up a Linux VM before proceeding, see the [Azure Linux VM tutorial](#).

In this case, use port 1999 as the PostgreSQL port.

Connect to the Linux VM you created via PuTTY. If this is the first time you're using an Azure Linux VM, see [How to Use SSH with Linux on Azure](#) to learn how to use PuTTY to connect to a Linux VM.

1. Run the following command to switch to the root (admin):

```
# sudo su -
```

2. Some distributions have dependencies that you must install before installing PostgreSQL. Check for your distro in this list and run the appropriate command:

- Red Hat base Linux:

```
# yum install readline-devel gcc make zlib-devel openssl openssl-devel libxml2-devel pam-devel  
pam libxslt-devel tcl-devel python-devel -y
```

- Debian base Linux:

```
# apt-get install readline-devel gcc make zlib-devel openssl openssl-devel libxml2-devel pam-devel  
pam libxslt-devel tcl-devel python-devel -y
```

- SUSE Linux:

```
# zypper install readline-devel gcc make zlib-devel openssl openssl-devel libxml2-devel pam-devel pam libxslt-devel tcl-devel python-devel -y
```

3. Download PostgreSQL into the root directory, and then unzip the package:

```
# wget https://ftp.postgresql.org/pub/source/v9.3.5/postgresql-9.3.5.tar.bz2 -P /root/  
# tar jxvf postgresql-9.3.5.tar.bz2
```

The above is an example. You can find the more detailed download address in the [Index of /pub/source/](#).

4. To start the build, run these commands:

```
# cd postgresql-9.3.5  
# ./configure --prefix=/opt/postgresql-9.3.5
```

5. If you want to build everything that can be built, including the documentation (HTML and man pages) and additional modules (contrib), run the following command instead:

```
# gmake install-world
```

You should receive the following confirmation message:

```
PostgreSQL, contrib, and documentation successfully made. Ready to install.
```

Configure PostgreSQL

1. (Optional) Create a symbolic link to shorten the PostgreSQL reference to not include the version number:

```
# ln -s /opt/postgresql-9.3.5 /opt/pgsql
```

2. Create a directory for the database:

```
# mkdir -p /opt/pgsql_data
```

3. Create a non-root user and modify that user's profile. Then, switch to this new user (called *postgres* in our example):

```
# useradd postgres  
# chown -R postgres.postgres /opt/pgsql_data  
# su - postgres
```

NOTE

For security reasons, PostgreSQL uses a non-root user to initialize, start, or shut down the database.

4. Edit the *bash_profile* file by entering the commands below. These lines will be added to the end of the

bash_profile file:

```
cat >> ~/.bash_profile <<EOF
export PGPORT=1999
export PGDATA=/opt/pgsql_data
export LANG=en_US.utf8
export PGHOME=/opt/pgsql
export PATH=\$PATH:\$PGHOME/bin
export MANPATH=\$MANPATH:\$PGHOME/share/man
export DATA=`date +"%Y%m%d%H%M"`
export PGUSER=postgres
alias rm='rm -i'
alias ll='ls -lh'
EOF
```

5. Execute the *bash_profile* file:

```
$ source .bash_profile
```

6. Validate your installation by using the following command:

```
$ which psql
```

If your installation is successful, you will see the following response:

```
/opt/pgsql/bin/psql
```

7. You can also check the PostgreSQL version:

```
$ psql -V
```

8. Initialize the database:

```
$ initdb -D $PGDATA -E UTF8 --locale=C -U postgres -W
```

You should receive the following output:

```
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A
--auth-local and --auth-host, the next time you run initdb.
```

```
Success. You can now start the database server using:
```

```
postgres -D /opt/pgsql_data
or
pg_ctl -D /opt/pgsql_data -l logfile start
```

Set up PostgreSQL

Run the following commands:

```
# cd /root/postgresql-9.3.5/contrib/start-scripts  
# cp linux /etc/init.d/postgresql
```

Modify two variables in the /etc/init.d/postgresql file. The prefix is set to the installation path of PostgreSQL: **/opt/pgsql**. PGDATA is set to the data storage path of PostgreSQL: **/opt/pgsql_data**.

```
# sed -i '32s#usr/local#opt#' /etc/init.d/postgresql  
# sed -i '35s#usr/local/pgsql/data#opt/pgsql_data#' /etc/init.d/postgresql
```

```
root@test:~  
27 # contrib/start-scripts/linux  
28  
29 ## EDIT FROM HERE  
30  
31 # Installation prefix  
32 prefix=/opt/pgsql  
33  
34 # Data directory  
35 PGDATA="/opt/pgsql_data"  
36
```

Change the file to make it executable:

```
# chmod +x /etc/init.d/postgresql
```

Start PostgreSQL:

```
# /etc/init.d/postgresql start
```

Check if the endpoint of PostgreSQL is on:

```
# netstat -tunlp|grep 1999
```

You should see the following output:

```
[root@test start-scripts]# netstat -tunlp|grep 1999  
tcp        0      0 127.0.0.1:1999          0.0.0.0:*  
tcp        0      0 ::1:1999                ::* LISTEN
```

Connect to the Postgres database

Switch to the postgres user once again:

```
# su - postgres
```

Create a Postgres database:

```
$ createdb events
```

Connect to the events database that you just created:

```
$ psql -d events
```

Create and delete a Postgres table

Now that you have connected to the database, you can create tables in it.

For example, create a new example Postgres table by using the following command:

```
CREATE TABLE potluck (name VARCHAR(20), food VARCHAR(30), confirmed CHAR(1), signup_date DATE);
```

You have now set up a four-column table with the following column names and restrictions:

1. The "name" column has been limited by the VARCHAR command to be under 20 characters long.
2. The "food" column indicates the food item that each person will bring. VARCHAR limits this text to be under 30 characters.
3. The "confirmed" column records whether the person has RSVP'd to the potluck. The acceptable values are "Y" and "N".
4. The "date" column shows when they signed up for the event. Postgres requires that dates be written as yyyy-mm-dd.

You should see the following if your table has been successfully created:

```
[postgres@test ~] $ psql -d events
psql (9.3.5)
Type "help" for help.

events=# CREATE TABLE potluck (name VARCHAR(20),
events(# food VARCHAR(30),
events(# confirmed CHAR(1),
events(# signup_date DATE);
CREATE TABLE
```

You can also check the table structure by using the following command:

```
events=# \dt
          List of relations
 Schema |   Name    | Type  | Owner
-----+-----+-----+
 public | potluck | table | postgres
(1 row)
```

Add data to a table

First, insert information into a row:

```
INSERT INTO potluck (name, food, confirmed, signup_date) VALUES('John', 'Casserole', 'Y', '2012-04-11');
```

You should see this output:

```
events=# INSERT INTO potluck (name, food, confirmed, signup_date) VALUES('John', 'Casserole', 'Y', '2012-04-11')
INSERT 0 1
```

You can add a couple more people to the table as well. Here are some options, or you can create your own:

```
INSERT INTO potluck (name, food, confirmed, signup_date) VALUES('Sandy', 'Key Lime Tarts', 'N', '2012-04-14');

INSERT INTO potluck (name, food, confirmed, signup_date) VALUES ('Tom', 'BBQ','Y', '2012-04-18');

INSERT INTO potluck (name, food, confirmed, signup_date) VALUES('Tina', 'Salad', 'Y', '2012-04-18');
```

Show tables

Use the following command to show a table:

```
select * from potluck;
```

The output is:

```
events=# select * from potluck;
 name |      food      | confirmed | signup_date
-----+-----+-----+-----+
 John | Casserole    | Y          | 2012-04-11
 Sandy | Key Lime Tarts | N          | 2012-04-14
 Tom  | BBQ           | Y          | 2012-04-18
 Tina | Salad         | Y          | 2012-04-18
(4 rows)
```

Delete data in a table

Use the following command to delete data in a table:

```
delete from potluck where name='John';
```

This deletes all the information in the "John" row. The output is:

```
events=# DELETE FROM potluck WHERE name = 'John' ;
DELETE 1
```

Update data in a table

Use the following command to update data in a table. For this one, Sandy has confirmed that they are attending, so we will change the RSVP from "N" to "Y":

```
UPDATE potluck set confirmed = 'Y' WHERE name = 'Sandy';
```

Get more information about PostgreSQL

Now that you have completed the installation of PostgreSQL in an Azure Linux VM, you can enjoy using it in Azure. To learn more about PostgreSQL, visit the [PostgreSQL website](#).

IBM DB2 pureScale on Azure

3/14/2019 • 5 minutes to read • [Edit Online](#)

The IBM DB2 pureScale environment provides a database cluster for Azure with high availability and scalability on Linux operating systems. This article shows an architecture for running DB2 pureScale on Azure.

Overview

Enterprises have long used relational database management system (RDBMS) platforms for their online transaction processing (OLTP) needs. These days, many are migrating their mainframe-based database environments to Azure as a way to expand capacity, reduce costs, and maintain a steady operational cost structure.

Migration is often the first step in modernizing an older platform. For example, one enterprise customer recently rehosted its IBM DB2 environment running on z/OS to IBM DB2 pureScale on Azure. Though not identical to the original environment, IBM DB2 pureScale on Linux delivers similar high-availability and scalability features as IBM DB2 for z/OS running in a Parallel Sysplex configuration on the mainframe.

This article describes the architecture used for this Azure migration. The customer used Red Hat Linux 7.4 to test the configuration. This version is available from the Azure Marketplace. Before you choose a Linux distribution, make sure to verify the currently supported versions. For details, see the documentation for [IBM DB2 pureScale](#) and [GlusterFS](#).

This article is a starting point for your DB2 implementation plan. Your business requirements will differ, but the same basic pattern applies. You can also use this architectural pattern for online analytical processing (OLAP) applications on Azure.

This article doesn't cover differences and possible migration tasks for moving an IBM DB2 for z/OS database to IBM DB2 pureScale running on Linux. And it doesn't provide sizing estimations and workload analyses for moving from DB2 z/OS to DB2 pureScale.

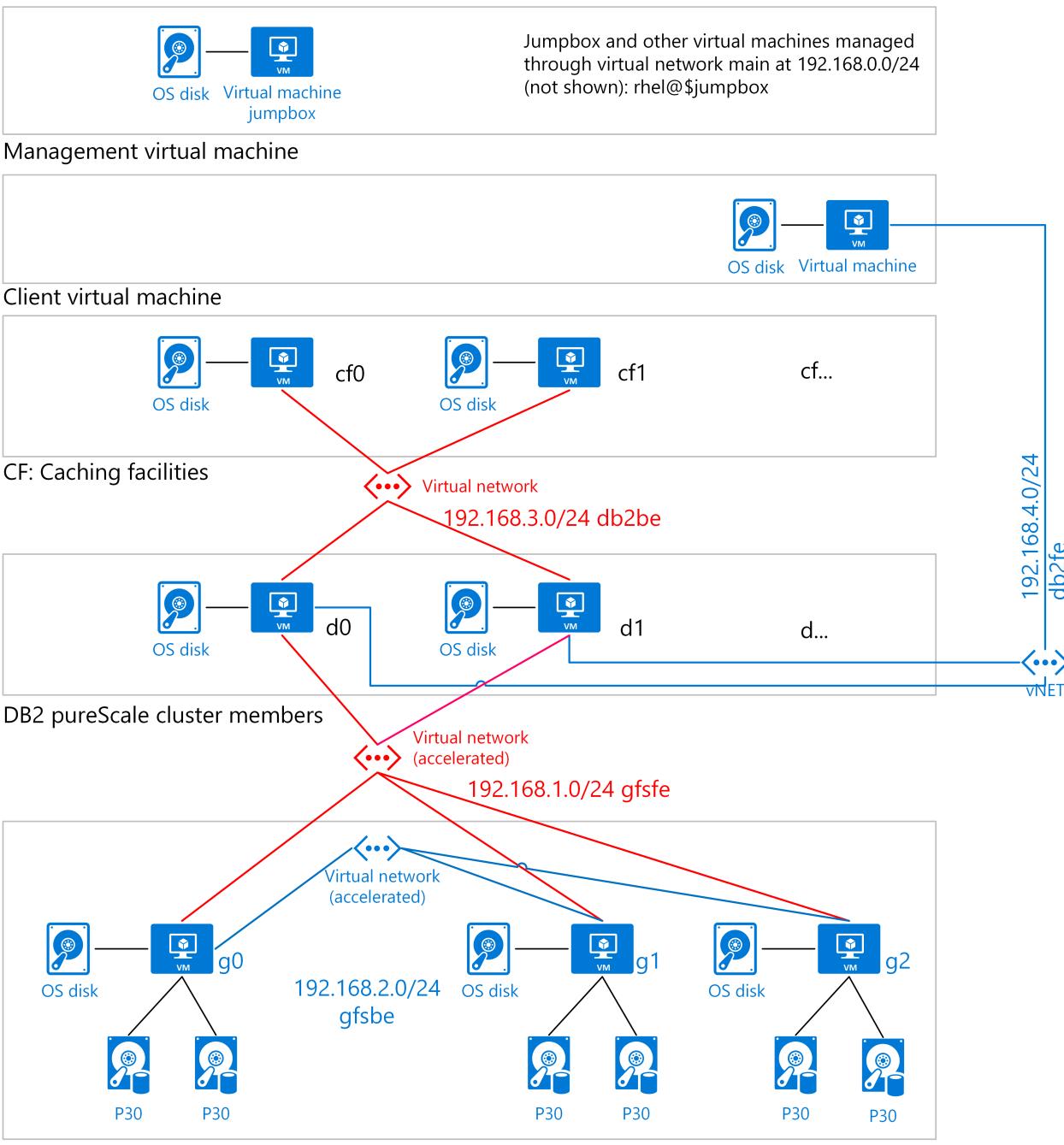
To help you decide on the best DB2 pureScale architecture for your environment, we recommend that you fully estimate sizing and make a hypothesis. On the source system, make sure to consider DB2 z/OS Parallel Sysplex with data-sharing architecture, Coupling Facility configuration, and distributed data facility (DDF) usage statistics.

NOTE

This article describes one approach to DB2 migration, but there are others. For example, DB2 pureScale can also run in virtualized on-premises environments. IBM supports DB2 on Microsoft Hyper-V in various configurations. For more information, see [DB2 pureScale virtualization architecture](#) in the IBM Knowledge Center.

Architecture

To support high availability and scalability on Azure, you can use a scale-out, shared data architecture for DB2 pureScale. The customer migration used the following example architecture.



The diagram shows the logical layers needed for a DB2 pureScale cluster. These include virtual machines for a client, for management, for caching, for the database engine, and for shared storage.

In addition to the database engine nodes, the diagram includes two nodes used for cluster caching facilities (CFs). At least two nodes are used for the database engine itself. A DB2 server that belongs to a pureScale cluster is called a member.

The cluster is connected via iSCSI to a three-node GlusterFS shared storage cluster to provide scale-out storage and high availability. DB2 pureScale is installed on Azure virtual machines running Linux.

This approach is a template that you can modify for the size and scale of your organization. It's based on the following:

- Two or more database members are combined with at least two CF nodes. The nodes manage a global buffer pool (GBP) for shared memory and global lock manager (GLM) services to control shared access and lock contention from active members. One CF node acts as the primary and the other as the secondary, failover CF node. To avoid a single point of failure in the environment, a DB2 pureScale cluster requires at least four nodes.

- High-performance shared storage (shown in P30 size in the diagram). Each of the Gluster FS nodes uses this storage.
- High-performance networking for the data members and shared storage.

Compute considerations

This architecture runs the application, storage, and data tiers on Azure virtual machines. The [deployment setup scripts](#) create the following:

- A DB2 pureScale cluster. The type of compute resources you need on Azure depends on your setup. In general, you can use two approaches:
 - Use a multi-node, high-performance computing (HPC)-style network where small to medium-sized instances access shared storage. For this HPC type of configuration, Azure memory-optimized E-series or storage-optimized L-series [virtual machines](#) provide the needed compute power.
 - Use fewer large virtual machine instances for the data engines. For large instances, the largest memory-optimized [M-series](#) virtual machines are ideal for heavy in-memory workloads. You might need a dedicated instance, depending on the size of the logical partition (LPAR) that's used to run DB2.
- The DB2 CF uses memory-optimized virtual machines, such as E-series or L-series.
- GlusterFS storage uses Standard_DS4_v2 virtual machines running Linux.
- A GlusterFS jumpbox is a Standard_DS2_v2 virtual machine running Linux.
- The client is a Standard_DS3_v2 virtual machine running Windows (used for testing).
- A witness server is a Standard_DS3_v2 virtual machine running Linux (used for DB2 pureScale).

NOTE

A DB2 pureScale cluster requires at least two DB2 instances. It also requires a cache instance and a lock manager instance.

Storage considerations

Like Oracle RAC, DB2 pureScale is a high-performance block I/O, scale-out database. We recommend using the largest [Azure premium SSD](#) option that suits your needs. Smaller storage options might be suitable for development and test environments, while production environments often need more storage capacity. The example architecture uses [P30](#) because of its ratio of IOPS to size and price. Regardless of size, use Premium Storage for best performance.

DB2 pureScale uses a shared-everything architecture, where all data is accessible from all cluster nodes. Premium storage must be shared across instances, whether on demand or on dedicated instances.

A large DB2 pureScale cluster can require 200 terabytes (TB) or more of premium shared storage, with IOPS of 100,000. DB2 pureScale supports an iSCSI block interface that you can use on Azure. The iSCSI interface requires a shared storage cluster that you can implement with GlusterFS, S2D, or another tool. This type of solution creates a virtual storage area network (vSAN) device in Azure. DB2 pureScale uses the vSAN to install the clustered file system that's used to share data among virtual machines.

The example architecture uses GlusterFS, a free, scalable, open-source distributed file system that's optimized for cloud storage.

Networking considerations

IBM recommends InfiniBand networking for all members in a DB2 pureScale cluster. DB2 pureScale also uses remote direct memory access (RDMA), where available, for the CFs.

During setup, you create an Azure [resource group](#) to contain all the virtual machines. In general, you group resources based on their lifetime and who will manage them. The virtual machines in this architecture require [accelerated networking](#). It's an Azure feature that provides consistent, ultra-low network latency via single-root I/O virtualization (SR-IOV) to a virtual machine.

Every Azure virtual machine is deployed into a virtual network that has subnets: main, Gluster FS front end (gfsfe), Gluster FS back end (bfsbe), DB2 pureScale (db2be), and DB2 pureScale front end (db2fe). The installation script also creates the primary [NICs](#) on the virtual machines in the main subnet.

Use [network security groups](#) to restrict network traffic within the virtual network and to isolate the subnets.

On Azure, DB2 pureScale needs to use TCP/IP as the network connection for storage.

Next steps

- [Deploy this architecture on Azure](#)

Deploy IBM DB2 pureScale on Azure

3/14/2019 • 5 minutes to read • [Edit Online](#)

This article describes how to deploy an [example architecture](#) that an enterprise customer recently used to migrate from its IBM DB2 environment running on z/OS to IBM DB2 pureScale on Azure.

To follow the steps used for the migration, see the installation scripts in the [DB2onAzure](#) repository on GitHub. These scripts are based on the architecture for a typical, medium-sized online transaction processing (OLTP) workload.

Get started

To deploy this architecture, download and run the deploy.sh script found in the [DB2onAzure](#) repository on GitHub.

The repository also has scripts for setting up a Grafana dashboard. You can use the dashboard to query Prometheus, the open-source monitoring and alerting system included with DB2.

NOTE

The deploy.sh script on the client creates private SSH keys and passes them to the deployment template over HTTPS. For greater security, we recommend using [Azure Key Vault](#) to store secrets, keys, and passwords.

How the deployment script works

The deploy.sh script creates and configures the Azure resources for this architecture. The script prompts you for the Azure subscription and virtual machines used in the target environment, and then performs the following operations:

- Sets up the resource group, virtual network, and subnets on Azure for the installation
- Sets up the network security groups and SSH for the environment
- Sets up NICs on both the GlusterFS and the DB2 pureScale virtual machines
- Creates the GlusterFS storage virtual machines
- Creates the jumpbox virtual machine
- Creates the DB2 pureScale virtual machines
- Creates the witness virtual machine that DB2 pureScale pings
- Creates a Windows virtual machine to use for testing but doesn't install anything on it

Next, the deployment scripts set up an iSCSI virtual storage area network (vSAN) for shared storage on Azure. In this example, iSCSI connects to GlusterFS. This solution also gives you the option to install the iSCSI targets as a single Windows node. iSCSI provides a shared block storage interface over TCP/IP that allows the DB2 pureScale setup procedure to use a device interface to connect to shared storage. For GlusterFS basics, see the [Architecture: Types of volumes](#) topic in Gluster Docs.

The deployment scripts run these general steps:

1. Use GlusterFS to set up a shared storage cluster on Azure. This step involves at least two Linux nodes. For setup details, see [Setting up Red Hat Gluster Storage in Microsoft Azure](#) in the Red Hat Gluster

documentation.

2. Set up an iSCSI Direct interface on target Linux servers for GlusterFS. For setup details, see [GlusterFS iSCSI](#) in the GlusterFS Administration Guide.
3. Set up the iSCSI initiator on the Linux virtual machines. The initiator will access the GlusterFS cluster by using an iSCSI target. For setup details, see [How To Configure An iSCSI Target And Initiator In Linux](#) in the RootUsers documentation.
4. Install GlusterFS as the storage layer for the iSCSI interface.

After the scripts create the iSCSI device, the final step is to install DB2 pureScale. As part of the DB2 pureScale setup, [IBM Spectrum Scale](#) (formerly known as GPFS) is compiled and installed on the GlusterFS cluster. This clustered file system enables DB2 pureScale to share data among the virtual machines that run the DB2 pureScale engine. For more information, see the [IBM Spectrum Scale](#) documentation on the IBM website.

DB2 pureScale response file

The GitHub repository includes DB2server.rsp, a response (.rsp) file that enables you to generate an automated script for the DB2 pureScale installation. The following table lists the DB2 pureScale options that the response file uses for setup. You can customize the response file as needed for your environment.

NOTE

A sample response file, DB2server.rsp, is included in the [DB2onAzure](#) repository on GitHub. If you use this file, you must edit it before it can work in your environment.

SCREEN NAME	FIELD	VALUE
Welcome		New Install
Choose a Product		DB2 Version 11.1.3.3. Server Editions with DB2 pureScale
Configuration	Directory	/data1/opt/ibm/db2/V11.1
	Select the installation type	Typical
	I agree to the IBM terms	Checked
Instance Owner	Existing User For Instance, User name	DB2sdin1
Fenced User	Existing User, User name	DB2sdf1
Cluster File System	Shared disk partition device path	/dev/dm-2
	Mount point	/DB2sd_1804a
	Shared disk for data	/dev/dm-1
	Mount point (Data)	/DB2fs/datafs1
	Shared disk for log	/dev/dm-0

SCREEN NAME	FIELD	VALUE
	Mount point (Log)	/DB2fs/logfs1
	DB2 Cluster Services Tiebreaker. Device path	/dev/dm-3
Host List	d1 [eth1], d2 [eth1], cf1 [eth1], cf2 [eth1]	
	Preferred primary CF	cf1
	Preferred secondary CF	cf2
Response File and Summary	first option	Install DB2 Server Edition with the IBM DB2 pureScale feature and save my settings in a response file
	Response file name	/root/DB2server.rsp

Notes about this deployment

- The values for /dev-dm0, /dev-dm1, /dev-dm2, and /dev-dm3 can change after a restart on the virtual machine where the setup takes place (d0 in the automated script). To find the right values, you can issue the following command before completing the response file on the server where the setup will run:

```
[root@d0 rhel]# ls -als /dev/mapper
total 0
0 drwxr-xr-x 2 root root 140 May 30 11:07 .
0 drwxr-xr-x 19 root root 4060 May 30 11:31 ..
0 crw----- 1 root root 10, 236 May 30 11:04 control
0 lrwxrwxrwx 1 root root 7 May 30 11:07 db2data1 -> ../dm-1
0 lrwxrwxrwx 1 root root 7 May 30 11:07 db2log1 -> ../dm-0
0 lrwxrwxrwx 1 root root 7 May 30 11:26 db2shared -> ../dm-2
0 lrwxrwxrwx 1 root root 7 May 30 11:08 db2tieb -> ../dm-3
```

- The setup scripts use aliases for the iSCSI disks so that the actual names can be found easily.
- When the setup script is run on d0, the **/dev/dm-*** values might be different on d1, cf0, and cf1. The difference in values doesn't affect the DB2 pureScale setup.

Troubleshooting and known issues

The GitHub repo includes a knowledge base that the authors maintain. It lists potential problems you might have and resolutions you can try. For example, known problems can happen when:

- You're trying to reach the gateway IP address.
- You're compiling General Public License (GPL).
- The security handshake between hosts fails.
- The DB2 installer detects an existing file system.
- You're manually installing IBM Spectrum Scale.
- You're installing DB2 pureScale when IBM Spectrum Scale is already created.
- You're removing DB2 pureScale and IBM Spectrum Scale.

For more information about these and other known problems, see the kb.md file in the [DB2onAzure](#) repo.

Next steps

- [GlusterFS iSCSI](#)
- [Creating required users for a DB2 pureScale Feature installation](#)
- [DB2icrt - Create instance command](#)
- [DB2 pureScale Clusters Data Solution](#)
- [IBM Data Studio](#)
- [Platform Modernization Alliance: IBM DB2 on Azure](#)
- [Azure Virtual Data Center Lift and Shift Guide](#)

Add a disk to a Linux VM

6/28/2019 • 8 minutes to read • [Edit Online](#)

This article shows you how to attach a persistent disk to your VM so that you can preserve your data - even if your VM is reprovisioned due to maintenance or resizing.

Attach a new disk to a VM

If you want to add a new, empty data disk on your VM, use the `az vm disk attach` command with the `--new` parameter. If your VM is in an Availability Zone, the disk is automatically created in the same zone as the VM. For more information, see [Overview of Availability Zones](#). The following example creates a disk named *myDataDisk* that is 50 Gb in size:

```
az vm disk attach \
    -g myResourceGroup \
    --vm-name myVM \
    --name myDataDisk \
    --new \
    --size-gb 50
```

Attach an existing disk

To attach an existing disk, find the disk ID and pass the ID to the `az vm disk attach` command. The following example queries for a disk named *myDataDisk* in *myResourceGroup*, then attaches it to the VM named *myVM*:

```
diskId=$(az disk show -g myResourceGroup -n myDataDisk --query 'id' -o tsv)

az vm disk attach -g myResourceGroup --vm-name myVM --name $diskId
```

Connect to the Linux VM to mount the new disk

To partition, format, and mount your new disk so your Linux VM can use it, SSH into your VM. For more information, see [How to use SSH with Linux on Azure](#). The following example connects to a VM with the public DNS entry of *mypublicdns.westus.cloudapp.azure.com* with the username *azureuser*:

```
ssh azureuser@mypublicdns.westus.cloudapp.azure.com
```

Once connected to your VM, you're ready to attach a disk. First, find the disk using `dmesg` (the method you use to discover your new disk may vary). The following example uses `dmesg` to filter on *SCSI* disks:

```
dmesg | grep SCSI
```

The output is similar to the following example:

```
[ 0.294784] SCSI subsystem initialized
[ 0.573458] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 252)
[ 7.110271] sd 2:0:0:0: [sda] Attached SCSI disk
[ 8.079653] sd 3:0:1:0: [sdb] Attached SCSI disk
[ 1828.162306] sd 5:0:0:0: [sdc] Attached SCSI disk
```

NOTE

It is recommended that you use the latest versions of fdisk or parted that are available for your distro.

Here, *sdc* is the disk that we want. Partition the disk with `parted`, if the disk size is 2 tebibytes (TiB) or larger then you must use GPT partitioning, if it is under 2TiB, then you can use either MBR or GPT partitioning. If you're using MBR partitioning, you can use `fdisk`. Make it a primary disk on partition 1, and accept the other defaults. The following example starts the `fdisk` process on `/dev/sdc`:

```
sudo fdisk /dev/sdc
```

Use the `n` command to add a new partition. In this example, we also choose `p` for a primary partition and accept the rest of the default values. The output will be similar to the following example:

```
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x2a59b123.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
  p   primary (0 primary, 0 extended, 4 free)
  e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-10485759, default 10485759):
Using default value 10485759
```

Print the partition table by typing `p` and then use `w` to write the table to disk and exit. The output should look similar to the following example:

```
Command (m for help): p

Disk /dev/sdc: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders, total 10485760 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
Disk identifier: 0x2a59b123

      Device Boot      Start        End      Blocks   Id  System
/dev/sdc1          2048    10485759    5241856   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Use the below command to update the kernel:

```
partprobe
```

Now, write a file system to the partition with the `mkfs` command. Specify your filesystem type and the device name. The following example creates an `ext4` filesystem on the `/dev/sdc1` partition that was created in the preceding steps:

```
sudo mkfs -t ext4 /dev/sdc1
```

The output is similar to the following example:

```
mke2fs 1.42.9 (4-Feb-2014)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
327680 inodes, 1310464 blocks
65523 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1342177280
40 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736
Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Now, create a directory to mount the file system using `mkdir`. The following example creates a directory at `/datadrive`:

```
sudo mkdir /datadrive
```

Use `mount` to then mount the filesystem. The following example mounts the `/dev/sdc1` partition to the `/datadrive` mount point:

```
sudo mount /dev/sdc1 /datadrive
```

To ensure that the drive is remounted automatically after a reboot, it must be added to the `/etc/fstab` file. It is also highly recommended that the UUID (Universally Unique Identifier) is used in `/etc/fstab` to refer to the drive rather than just the device name (such as, `/dev/sdc1`). If the OS detects a disk error during boot, using the UUID avoids the incorrect disk being mounted to a given location. Remaining data disks would then be assigned those same device IDs. To find the UUID of the new drive, use the `blkid` utility:

```
sudo blkid
```

The output looks similar to the following example:

```
/dev/sda1: UUID="11111111-1b1b-1c1c-1d1d-1e1e1e1e1e" TYPE="ext4"
/dev/sdb1: UUID="22222222-2b2b-2c2c-2d2d-2e2e2e2e2e" TYPE="ext4"
/dev/sdc1: UUID="33333333-3b3b-3c3c-3d3d-3e3e3e3e3e" TYPE="ext4"
```

NOTE

Improperly editing the `/etc/fstab` file could result in an unbootable system. If unsure, refer to the distribution's documentation for information on how to properly edit this file. It is also recommended that a backup of the `/etc/fstab` file is created before editing.

Next, open the `/etc/fstab` file in a text editor as follows:

```
sudo vi /etc/fstab
```

In this example, use the UUID value for the `/dev/sdc1` device that was created in the previous steps, and the mountpoint of `/datadrive`. Add the following line to the end of the `/etc/fstab` file:

```
UUID=33333333-3b3b-3c3c-3d3d-3e3e3e3e3e   /datadrive   ext4   defaults,nofail   1   2
```

NOTE

Later removing a data disk without editing fstab could cause the VM to fail to boot. Most distributions provide either the `nofail` and/or `nobootwait` fstab options. These options allow a system to boot even if the disk fails to mount at boot time. Consult your distribution's documentation for more information on these parameters.

The `nofail` option ensures that the VM starts even if the filesystem is corrupt or the disk does not exist at boot time. Without this option, you may encounter behavior as described in [Cannot SSH to Linux VM due to FSTAB errors](#)

The Azure VM Serial Console can be used for console access to your VM if modifying fstab has resulted in a boot failure. More details are available in the [Serial Console documentation](#).

TRIM/UNMAP support for Linux in Azure

Some Linux kernels support TRIM/UNMAP operations to discard unused blocks on the disk. This feature is primarily useful in standard storage to inform Azure that deleted pages are no longer valid and can be discarded, and can save money if you create large files and then delete them.

There are two ways to enable TRIM support in your Linux VM. As usual, consult your distribution for the recommended approach:

- Use the `discard` mount option in `/etc/fstab`, for example:

```
UUID=33333333-3b3b-3c3c-3d3d-3e3e3e3e3e /datadrive ext4 defaults,discard 1 2
```

- In some cases, the `discard` option may have performance implications. Alternatively, you can run the `fstrim` command manually from the command line, or add it to your crontab to run regularly:

Ubuntu

```
sudo apt-get install util-linux
sudo fstrim /datadrive
```

RHEL/CentOS

```
sudo yum install util-linux
sudo fstrim /datadrive
```

Troubleshooting

When adding data disks to a Linux VM, you may encounter errors if a disk does not exist at LUN 0. If you are adding a disk manually using the `azure vm disk attach-new` command and you specify a LUN (`--lun`) rather than allowing the Azure platform to determine the appropriate LUN, take care that a disk already exists / will exist at LUN 0.

Consider the following example showing a snippet of the output from `lsscsi`:

```
[5:0:0:0]    disk    Msft      Virtual Disk    1.0   /dev/sdc
[5:0:0:1]    disk    Msft      Virtual Disk    1.0   /dev/sdd
```

The two data disks exist at LUN 0 and LUN 1 (the first column in the `lsscsi` output details `[host:channel:target:lun]`). Both disks should be accessible from within the VM. If you had manually specified the first disk to be added at LUN 1 and the second disk at LUN 2, you may not see the disks correctly from within your VM.

NOTE

The Azure `host` value is 5 in these examples, but this may vary depending on the type of storage you select.

This disk behavior is not an Azure problem, but the way in which the Linux kernel follows the SCSI specifications. When the Linux kernel scans the SCSI bus for attached devices, a device must be found at LUN 0 in order for the system to continue scanning for additional devices. As such:

- Review the output of `lsscsi` after adding a data disk to verify that you have a disk at LUN 0.
- If your disk does not show up correctly within your VM, verify a disk exists at LUN 0.

Next steps

- To ensure your Linux VM is configured correctly, review the [Optimize your Linux machine performance recommendations](#).
- Expand your storage capacity by adding additional disks and [configure RAID](#) for additional performance.

Use the portal to attach a data disk to a Linux VM

2/15/2019 • 7 minutes to read • [Edit Online](#)

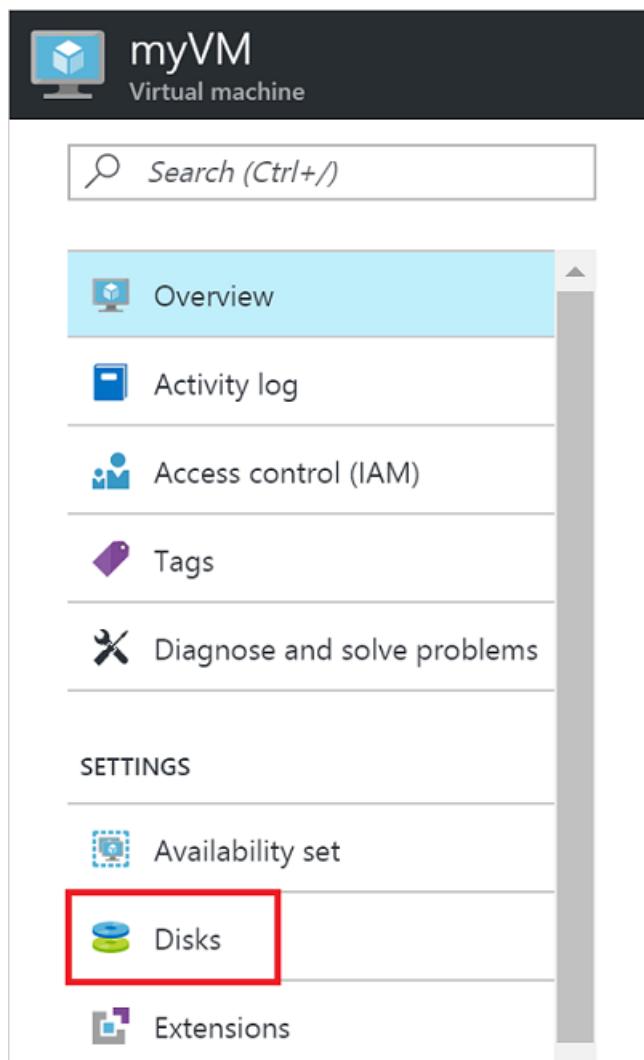
This article shows you how to attach both new and existing disks to a Linux virtual machine through the Azure portal. You can also [attach a data disk to a Windows VM in the Azure portal](#).

Before you attach disks to your VM, review these tips:

- The size of the virtual machine controls how many data disks you can attach. For details, see [Sizes for virtual machines](#).
- Disks attached to virtual machines are actually .vhd files stored in Azure. For details, see our [Introduction to managed disks](#).
- After attaching the disk, you need to [connect to the Linux VM to mount the new disk](#).

Find the virtual machine

1. Sign in to the [Azure portal](#).
2. On the left menu, click **Virtual Machines**.
3. Select the virtual machine from the list.
4. To the Virtual machines page, in **Essentials**, click **Disks**.



Attach a new disk

1. On the **Disks** pane, click **+ Add data disk**.
2. Click the drop-down menu for **Name** and select **Create disk**:

The screenshot shows the 'Disks' pane in the Azure portal. At the top, there are 'Save' and 'Discard' buttons. Below is a table for the 'OS disk':

NAME	SIZE	ACCOUNT TYPE
myVM		Premium_LRS

Below the OS disk section is a 'Data disks' section:

LUN	NAME	SIZE	ACCOUNT TYPE
0	<input type="text"/> Create disk		

A red box highlights the 'NAME' dropdown, which has a dropdown menu open showing 'Create disk' and 'All disks'. A 'Create disk' button is also visible in the background.

3. Enter a name for your managed disk. Review the default settings, update as necessary, and then click **Create**.

The screenshot shows the 'Create managed disk' dialog box. It contains the following fields:

- Name**: myDataDisk (highlighted with a red box)
- Resource group**: myResourceGroup (radio buttons for 'Create new' and 'Use existing')
- Account type**: Premium_LRS
- Source type**: None (empty disk)
- Size (GiB)**: 1023
- Estimated performance** (details: IOPS limit 5000, Throughput limit 200 MB/s)

A red box highlights the 'Create' button at the bottom left of the dialog.

4. Click **Save** to create the managed disk and update the VM configuration:

The screenshot shows the 'Disk' configuration page in the Azure portal. At the top, there are 'Save' and 'Discard' buttons. Below them, the 'OS disk' section lists a single disk named 'myVM' with a size of 1023 GiB, account type 'Premium_LRS', and caching set to 'Read/write'. In the 'Data disks' section, a new disk named 'myDataDisk' is being added, with a size of 1023 GiB, account type 'Premium_LRS', and caching set to 'None'. A '+ Add data disk' button is visible at the bottom.

- After Azure creates the disk and attaches it to the virtual machine, the new disk is listed in the virtual machine's disk settings under **Data Disks**. As managed disks are a top-level resource, the disk appears at the root of the resource group:

The screenshot shows the 'myResourceGroup' resource group details in the Azure portal. On the left, a navigation menu includes 'Overview' (which is selected and highlighted in blue), 'Activity log', 'Access control (IAM)', 'Tags', 'SETTINGS' (with 'Quickstart', 'Resource costs', and 'Deployments' options), and a 'Disk' icon. The main pane displays the 'Essentials' section with subscription information and deployment status. Below this is a table listing resources, with 'myDataDisk' highlighted by a red box. The table columns are NAME, TYPE, and LOCATION.

NAME	TYPE	LOCATION
myAvailabilitySet	Availability set	West US
myDataDisk	Disk	West US
myNetworkSecurityGroup	Network security gro...	West US

Attach an existing disk

- On the **Disk** pane, click **+ Add data disk**.
- Click the drop-down menu for **Name** to view a list of existing managed disks accessible to your Azure subscription. Select the managed disk to attach:

Save Discard

OS disk

NAME	SIZE	ACCOUNT TYPE
myVM		Premium_LRS

Data disks

LUN	NAME	SIZE	ACCOUNT TYPE
0	myDataDisk	1023 GiB	Premium_LRS
1	<input type="button" value="▼"/>		

Create disk

Disks in resource group 'myResourceGroup'

myExistingDisk
size: 1023 GiB, account type: Premium_LRS

All disks

myExistingDisk
size: 1023 GiB, account type: Premium_LRS, resource group: MYRESOURCEGROUP

- Click **Save** to attach the existing managed disk and update the VM configuration:

Save Discard

OS disk

NAME	SIZE	ACCOUNT TYPE	ENCRYPTION	CACHING
myVM		Premium_LRS		Read/write

Data disks

LUN	NAME	SIZE	ACCOUNT TYPE	ENCRYPTION	CACHING
0	myDataDisk	1023 GiB	Premium_LRS		None
1	<input type="button" value="▼"/> myExistingDisk	1023 GiB	Premium_LRS		None
+ Add data disk					

- After Azure attaches the disk to the virtual machine, it's listed in the virtual machine's disk settings under **Data Disks**.

Connect to the Linux VM to mount the new disk

To partition, format, and mount your new disk so your Linux VM can use it, SSH into your VM. For more information, see [How to use SSH with Linux on Azure](#). The following example connects to a VM with the public DNS entry of *mypublicdns.westus.cloudapp.azure.com* with the username *azureuser*:

```
ssh azureuser@mypublicdns.westus.cloudapp.azure.com
```

Once connected to your VM, you're ready to attach a disk. First, find the disk using `dmesg` (the method you use to discover your new disk may vary). The following example uses `dmesg` to filter on `SCSI` disks:

```
dmesg | grep SCSI
```

The output is similar to the following example:

```
[ 0.294784] SCSI subsystem initialized
[ 0.573458] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 252)
[ 7.110271] sd 2:0:0:0: [sda] Attached SCSI disk
[ 8.079653] sd 3:0:1:0: [sdb] Attached SCSI disk
[ 1828.162306] sd 5:0:0:0: [sdc] Attached SCSI disk
```

Here, `sdc` is the disk that we want.

Partition a new disk

If you are using an existing disk that contains data, skip to mounting the disk. If you are attaching a new disk, you need to partition the disk.

Partition the disk with `fdisk`. If the disk size is 2 tebibytes (TiB) or larger then you must use GPT partitioning, you can use `parted` to perform GPT partitioning. If disk size is under 2TiB, then you can use either MBR or GPT partitioning. Make it a primary disk on partition 1, and accept the other defaults. The following example starts the `fdisk` process on `/dev/sdc`:

```
sudo fdisk /dev/sdc
```

Use the `n` command to add a new partition. In this example, we also choose `p` for a primary partition and accept the rest of the default values. The output will be similar to the following example:

```
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x2a59b123.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
  p  primary (0 primary, 0 extended, 4 free)
  e  extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-10485759, default 10485759):
Using default value 10485759
```

Print the partition table by typing `p` and then use `w` to write the table to disk and exit. The output should look similar to the following example:

```

Command (m for help): p

Disk /dev/sdc: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders, total 10485760 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
Disk identifier: 0x2a59b123

      Device Boot      Start        End    Blocks   Id  System
/dev/sdc1            2048    10485759   5241856   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

```

Now, write a file system to the partition with the `mkfs` command. Specify your filesystem type and the device name. The following example creates an `ext4` filesystem on the `/dev/sdc1` partition that was created in the preceding steps:

```
sudo mkfs -t ext4 /dev/sdc1
```

The output is similar to the following example:

```

mke2fs 1.42.9 (4-Feb-2014)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
327680 inodes, 1310464 blocks
65523 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1342177280
40 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736
Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

```

Mount the disk

Create a directory to mount the file system using `mkdir`. The following example creates a directory at `/datadrive`:

```
sudo mkdir /datadrive
```

Use `mount` to then mount the filesystem. The following example mounts the `/dev/sdc1` partition to the `/datadrive` mount point:

```
sudo mount /dev/sdc1 /datadrive
```

To ensure that the drive is remounted automatically after a reboot, it must be added to the `/etc/fstab` file. It is also highly recommended that the UUID (Universally Unique Identifier) is used in `/etc/fstab` to refer to the drive rather than just the device name (such as, `/dev/sdc1`). If the OS detects a disk error during boot, using the UUID avoids the incorrect disk being mounted to a given location. Remaining data disks would then be assigned those same device IDs. To find the UUID of the new drive, use the `blkid` utility:

```
sudo -i blkid
```

The output looks similar to the following example:

```
/dev/sda1: UUID="11111111-1b1b-1c1c-1d1d-1e1e1e1e1e" TYPE="ext4"  
/dev/sdb1: UUID="22222222-2b2b-2c2c-2d2d-2e2e2e2e2e" TYPE="ext4"  
/dev/sdc1: UUID="33333333-3b3b-3c3c-3d3d-3e3e3e3e3e" TYPE="ext4"
```

NOTE

Improperly editing the `/etc/fstab` file could result in an unbootable system. If unsure, refer to the distribution's documentation for information on how to properly edit this file. It is also recommended that a backup of the `/etc/fstab` file is created before editing.

Next, open the `/etc/fstab` file in a text editor as follows:

```
sudo vi /etc/fstab
```

In this example, use the UUID value for the `/dev/sdc1` device that was created in the previous steps, and the mountpoint of `/datadrive`. Add the following line to the end of the `/etc/fstab` file:

```
UUID=33333333-3b3b-3c3c-3d3d-3e3e3e3e3e /datadrive ext4 defaults,nofail 1 2
```

NOTE

Later removing a data disk without editing fstab could cause the VM to fail to boot. Most distributions provide either the `nofail` and/or `nobootwait` fstab options. These options allow a system to boot even if the disk fails to mount at boot time. Consult your distribution's documentation for more information on these parameters.

The `nofail` option ensures that the VM starts even if the filesystem is corrupt or the disk does not exist at boot time. Without this option, you may encounter behavior as described in [Cannot SSH to Linux VM due to FSTAB errors](#)

TRIM/UNMAP support for Linux in Azure

Some Linux kernels support TRIM/UNMAP operations to discard unused blocks on the disk. This feature is primarily useful in standard storage to inform Azure that deleted pages are no longer valid and can be discarded, and can save money if you create large files and then delete them.

There are two ways to enable TRIM support in your Linux VM. As usual, consult your distribution for the recommended approach:

- Use the `discard` mount option in `/etc/fstab`, for example:

```
UUID=33333333-3b3b-3c3c-3d3d-3e3e3e3e3e /datadrive ext4 defaults,discard 1 2
```

- In some cases, the `discard` option may have performance implications. Alternatively, you can run the

`fstrim` command manually from the command line, or add it to your crontab to run regularly:

Ubuntu

```
sudo apt-get install util-linux  
sudo fstrim /datadrive
```

RHEL/CentOS

```
sudo yum install util-linux  
sudo fstrim /datadrive
```

Next steps

You can also [attach a data disk](#) using the Azure CLI.

How to detach a data disk from a Linux virtual machine

4/23/2019 • 2 minutes to read • [Edit Online](#)

When you no longer need a data disk that's attached to a virtual machine, you can easily detach it. This removes the disk from the virtual machine, but doesn't remove it from storage. In this article, we are working with an Ubuntu LTS 16.04 distribution. If you are using a different distribution, the instructions for unmounting the disk might be different.

WARNING

If you detach a disk it is not automatically deleted. If you have subscribed to Premium storage, you will continue to incur storage charges for the disk. For more information, see [Pricing and Billing when using Premium Storage](#).

If you want to use the existing data on the disk again, you can reattach it to the same virtual machine, or another one.

Connect to the VM to unmount the disk

Before you can detach the disk using either CLI or the portal, you need to unmount the disk and removed references to it from your fstab file.

Connect to the VM. In this example, the public IP address of the VM is `10.0.1.4` with the username `azureuser`:

```
ssh azureuser@10.0.1.4
```

First, find the data disk that you want to detach. The following example uses dmesg to filter on SCSI disks:

```
dmesg | grep SCSI
```

The output is similar to the following example:

```
[ 0.294784] SCSI subsystem initialized
[ 0.573458] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 252)
[ 7.110271] sd 2:0:0:0: [sda] Attached SCSI disk
[ 8.079653] sd 3:0:1:0: [sdb] Attached SCSI disk
[ 1828.162306] sd 5:0:0:0: [sdc] Attached SCSI disk
```

Here, `sdc` is the disk that we want to detach. You also should grab the UUID of the disk.

```
sudo -i blkid
```

The output looks similar to the following example:

```
/dev/sda1: UUID="11111111-1b1b-1c1c-1d1d-1e1e1e1e1e" TYPE="ext4"
/dev/sdb1: UUID="22222222-2b2b-2c2c-2d2d-2e2e2e2e2e" TYPE="ext4"
/dev/sdc1: UUID="33333333-3b3b-3c3c-3d3d-3e3e3e3e3e" TYPE="ext4"
```

Edit the `/etc/fstab` file to remove references to the disk.

NOTE

Improperly editing the `/etc/fstab` file could result in an unbootable system. If unsure, refer to the distribution's documentation for information on how to properly edit this file. It is also recommended that a backup of the `/etc/fstab` file is created before editing.

Open the `/etc/fstab` file in a text editor as follows:

```
sudo vi /etc/fstab
```

In this example, the following line needs to be deleted from the `/etc/fstab` file:

```
UUID=33333333-3b3b-3c3c-3d3d-3e3e3e3e3e /datadrive ext4 defaults,nofail 1 2
```

Use `umount` to unmount the disk. The following example unmounts the `/dev/sdc1` partition from the `/datadrive` mount point:

```
sudo umount /dev/sdc1 /datadrive
```

Detach a data disk using Azure CLI

This example detaches the `myDataDisk` disk from VM named `myVM` in `myResourceGroup`.

```
az vm disk detach \
    -g myResourceGroup \
    --vm-name myVm \
    -n myDataDisk
```

The disk stays in storage but is no longer attached to a virtual machine.

Detach a data disk using the portal

1. In the left menu, select **Virtual Machines**.
2. Select the virtual machine that has the data disk you want to detach and click **Stop** to deallocate the VM.
3. In the virtual machine pane, select **Disks**.
4. At the top of the **Disk** pane, select **Edit**.
5. In the **Disk** pane, to the far right of the data disk that you would like to detach, click the detach button.
6. After the disk has been removed, click **Save** on the top of the pane.
7. In the virtual machine pane, click **Overview** and then click the **Start** button at the top of the pane to restart the VM.

The disk stays in storage but is no longer attached to a virtual machine.

Next steps

If you want to reuse the data disk, you can just [attach it to another VM](#).

Using Managed Disks in Azure Resource Manager Templates

2/4/2019 • 5 minutes to read • [Edit Online](#)

This document walks through the differences between managed and unmanaged disks when using Azure Resource Manager templates to provision virtual machines. The examples help you to update existing templates that are using unmanaged Disks to managed disks. For reference, we are using the [101-vm-simple-windows](#) template as a guide. You can see the template using both [managed Disks](#) and a prior version using [unmanaged disks](#) if you'd like to directly compare them.

Unmanaged Disks template formatting

To begin, let's take a look at how unmanaged disks are deployed. When creating unmanaged disks, you need a storage account to hold the VHD files. You can create a new storage account or use one that already exists. This article shows you how to create a new storage account. Create a storage account resource in the resources block as shown below.

```
{  
  "type": "Microsoft.Storage/storageAccounts",  
  "apiVersion": "2018-07-01",  
  "name": "[variables('storageAccountName')]",  
  "location": "[resourceGroup().location]",  
  "sku": {  
    "name": "Standard_LRS"  
  },  
  "kind": "Storage",  
  "properties": {}  
}
```

Within the virtual machine object, add a dependency on the storage account to ensure that it's created before the virtual machine. Within the `storageProfile` section, specify the full URI of the VHD location, which references the storage account and is needed for the OS disk and any data disks.

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
  "properties": {
    "hardwareProfile": {...},
    "osProfile": {...},
    "storageProfile": {
      "imageReference": {
        "publisher": "MicrosoftWindowsServer",
        "offer": "WindowsServer",
        "sku": "[parameters('windowsOSVersion')]",
        "version": "latest"
      },
      "osDisk": {
        "name": "osdisk",
        "vhd": {
          "uri": "[concat(reference(resourceId('Microsoft.Storage/storageAccounts/',
variables('storageAccountName'))).primaryEndpoints.blob, 'vhds/osdisk.vhd')]"
        },
        "caching": "ReadWrite",
        "createOption": "FromImage"
      },
      "dataDisks": [
        {
          "name": "datadisk1",
          "diskSizeGB": 1023,
          "lun": 0,
          "vhd": {
            "uri": "[concat(reference(resourceId('Microsoft.Storage/storageAccounts/',
variables('storageAccountName'))).primaryEndpoints.blob, 'vhds/datadisk1.vhd')]"
          },
          "createOption": "Empty"
        }
      ]
    },
    "networkProfile": {...},
    "diagnosticsProfile": {...}
  }
}
```

Managed disks template formatting

With Azure Managed Disks, the disk becomes a top-level resource and no longer requires a storage account to be created by the user. Managed disks were first exposed in the [2016-04-30-preview](#) API version, they are available in all subsequent API versions and are now the default disk type. The following sections walk through the default settings and detail how to further customize your disks.

NOTE

It is recommended to use an API version later than [2016-04-30-preview](#) as there were breaking changes between [2016-04-30-preview](#) and [2017-03-30](#).

Default managed disk settings

To create a VM with managed disks, you no longer need to create the storage account resource and can update your virtual machine resource as follows. Specifically note that the `apiVersion` reflects [2017-03-30](#) and the

`osDisk` and `dataDisks` no longer refer to a specific URI for the VHD. When deploying without specifying additional properties, the disk will use a Storage type based on the size of the VM. For example, if you are using a Premium capable VM Size (sizes with "s" in their name such as Standard_D2s_v3) then system will use Premium_LRS storage. Use the sku setting of the disk to specify a Storage type. If no name is specified, it takes the format of `<VMName>_OsDisk_1_<randomstring>` for the OS disk and `<VMName>_disk<#>_<randomstring>` for each data disk. By default, Azure disk encryption is disabled; caching is Read/Write for the OS disk and None for data disks. You may notice in the example below there is still a storage account dependency, though this is only for storage of diagnostics and is not needed for disk storage.

```
{  
    "type": "Microsoft.Compute/virtualMachines",  
    "apiVersion": "2018-10-01",  
    "name": "[variables('vmName')]",  
    "location": "[resourceGroup().location]",  
    "dependsOn": [  
        "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",  
        "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"  
    ],  
    "properties": {  
        "hardwareProfile": {...},  
        "osProfile": {...},  
        "storageProfile": {  
            "imageReference": {  
                "publisher": "MicrosoftWindowsServer",  
                "offer": "WindowsServer",  
                "sku": "[parameters('windowsOSVersion')]",  
                "version": "latest"  
            },  
            "osDisk": {  
                "createOption": "FromImage"  
            },  
            "dataDisks": [  
                {  
                    "diskSizeGB": 1023,  
                    "lun": 0,  
                    "createOption": "Empty"  
                }  
            ]  
        },  
        "networkProfile": {...},  
        "diagnosticsProfile": {...}  
    }  
}
```

Using a top-level managed disk resource

As an alternative to specifying the disk configuration in the virtual machine object, you can create a top-level disk resource and attach it as part of the virtual machine creation. For example, you can create a disk resource as follows to use as a data disk.

```
{
  "type": "Microsoft.Compute/disks",
  "apiVersion": "2018-06-01",
  "name": "[concat(variables('vmName'),'-datadisk1')]",
  "location": "[resourceGroup().location]",
  "sku": {
    "name": "Standard_LRS"
  },
  "properties": {
    "creationData": {
      "createOption": "Empty"
    },
    "diskSizeGB": 1023
  }
}
```

Within the VM object, reference the disk object to be attached. Specifying the resource ID of the managed disk created in the `managedDisk` property allows the attachment of the disk as the VM is created. The `apiVersion` for the VM resource is set to `2017-03-30`. A dependency on the disk resource is added to ensure it's successfully created before VM creation.

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]",
    "[resourceId('Microsoft.Compute/disks/', concat(variables('vmName'),'-datadisk1'))]"
  ],
  "properties": {
    "hardwareProfile": {...},
    "osProfile": {...},
    "storageProfile": {
      "imageReference": {
        "publisher": "MicrosoftWindowsServer",
        "offer": "WindowsServer",
        "sku": "[parameters('windowsOSVersion')]",
        "version": "latest"
      },
      "osDisk": {
        "createOption": "FromImage"
      },
      "dataDisks": [
        {
          "lun": 0,
          "name": "[concat(variables('vmName'),'-datadisk1')]",
          "createOption": "attach",
          "managedDisk": {
            "id": "[resourceId('Microsoft.Compute/disks/', concat(variables('vmName'),'-datadisk1'))]"
          }
        }
      ]
    },
    "networkProfile": {...},
    "diagnosticsProfile": {...}
  }
}
```

Create managed availability sets with VMs using managed disks

To create managed availability sets with VMs using managed disks, add the `sku` object to the availability set

resource and set the `name` property to `Aligned`. This property ensures that the disks for each VM are sufficiently isolated from each other to avoid single points of failure. Also note that the `apiVersion` for the availability set resource is set to `2018-10-01`.

```
{  
    "type": "Microsoft.Compute/availabilitySets",  
    "apiVersion": "2018-10-01",  
    "location": "[resourceGroup().location]",  
    "name": "[variables('avSetName')]",  
    "properties": {  
        "PlatformUpdateDomainCount": 3,  
        "PlatformFaultDomainCount": 2  
    },  
    "sku": {  
        "name": "Aligned"  
    }  
}
```

Standard SSD disks

Below are the parameters needed in the Resource Manager template to create Standard SSD Disks:

- `apiVersion` for Microsoft.Compute must be set as `2018-04-01` (or later)
- Specify `managedDisk.storageAccountType` as `StandardSSD_LRS`

The following example shows the `properties.storageProfile.osDisk` section for a VM that uses Standard SSD Disks:

```
"osDisk": {  
    "osType": "Windows",  
    "name": "myOsDisk",  
    "caching": "ReadWrite",  
    "createOption": "FromImage",  
    "managedDisk": {  
        "storageAccountType": "StandardSSD_LRS"  
    }  
}
```

For a complete template example of how to create a Standard SSD disk with a template, see [Create a VM from a Windows Image with Standard SSD Data Disks](#).

Additional scenarios and customizations

To find full information on the REST API specifications, please review the [create a managed disk REST API documentation](#). You will find additional scenarios, as well as default and acceptable values that can be submitted to the API through template deployments.

Next steps

- For full templates that use managed disks visit the following Azure Quickstart Repo links.
 - [Windows VM with managed disk](#)
 - [Linux VM with managed disk](#)
 - [Full list of managed disk templates](#)
- Visit the [Azure Managed Disks Overview](#) document to learn more about managed disks.
- Review the template reference documentation for virtual machine resources by visiting the [Microsoft.Compute/virtualMachines template reference](#) document.
- Review the template reference documentation for disk resources by visiting the [Microsoft.Compute/disks template reference](#) document.
- For information on how to use managed disks in Azure virtual machine scale sets, visit the [Use data disks with](#)

[scale sets](#) document.

Expand virtual hard disks on a Linux VM with the Azure CLI

5/9/2019 • 3 minutes to read • [Edit Online](#)

This article describes how to expand managed disks for a Linux virtual machine (VM) with the Azure CLI. You can [add data disks](#) to provide for additional storage space, and you can also expand an existing data disk. The default virtual hard disk size for the operating system (OS) is typically 30 GB on a Linux VM in Azure.

WARNING

Always make sure that your filesystem is in a healthy state, your disk partition table type will support the new size, and ensure your data is backed up before you perform disk resize operations. For more information, see [Back up Linux VMs in Azure](#).

Expand an Azure Managed Disk

Make sure that you have the latest [Azure CLI](#) installed and are signed in to an Azure account by using [az login](#).

This article requires an existing VM in Azure with at least one data disk attached and prepared. If you do not already have a VM that you can use, see [Create and prepare a VM with data disks](#).

In the following samples, replace example parameter names such as *myResourceGroup* and *myVM* with your own values.

- Operations on virtual hard disks can't be performed with the VM running. Deallocate your VM with [az vm deallocate](#). The following example deallocates the VM named *myVM* in the resource group named *myResourceGroup*:

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

NOTE

The VM must be deallocated to expand the virtual hard disk. Stopping the VM with `az vm stop` does not release the compute resources. To release compute resources, use `az vm deallocate`.

- View a list of managed disks in a resource group with [az disk list](#). The following example displays a list of managed disks in the resource group named *myResourceGroup*:

```
az disk list \
    --resource-group myResourceGroup \
    --query '[*].{Name:name,Gb:diskSizeGb,Tier:accountType}' \
    --output table
```

Expand the required disk with [az disk update](#). The following example expands the managed disk named *myDataDisk* to 200 GB:

```
az disk update \
--resource-group myResourceGroup \
--name myDataDisk \
--size-gb 200
```

NOTE

When you expand a managed disk, the updated size is rounded up to the nearest managed disk size. For a table of the available managed disk sizes and tiers, see [Azure Managed Disks Overview - Pricing and Billing](#).

3. Start your VM with `az vm start`. The following example starts the VM named *myVM* in the resource group named *myResourceGroup*:

```
az vm start --resource-group myResourceGroup --name myVM
```

Expand a disk partition and filesystem

To use an expanded disk, expand the underlying partition and filesystem.

1. SSH to your VM with the appropriate credentials. You can see the public IP address of your VM with `az vm show`:

```
az vm show --resource-group myResourceGroup --name myVM -d --query [publicIps] --o tsv
```

2. Expand the underlying partition and filesystem.

- a. If the disk is already mounted, unmount it:

```
sudo umount /dev/sdc1
```

- b. Use `parted` to view disk information and resize the partition:

```
sudo parted /dev/sdc
```

View information about the existing partition layout with `print`. The output is similar to the following example, which shows the underlying disk is 215 GB:

```
GNU Parted 3.2
Using /dev/sdc1
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Unknown Msft Virtual Disk (scsi)
Disk /dev/sdc1: 215GB
Sector size (logical/physical): 512B/4096B
Partition Table: loop
Disk Flags:

Number  Start   End     Size    File system  Flags
          0.00B  107GB  107GB   ext4
```

- c. Expand the partition with `resizepart`. Enter the partition number, *1*, and a size for the new partition:

```
(parted) resizepart  
Partition number? 1  
End? [107GB]? 215GB
```

d. To exit, enter `quit`.

3. With the partition resized, verify the partition consistency with `e2fsck`:

```
sudo e2fsck -f /dev/sdc1
```

4. Resize the filesystem with `resize2fs`:

```
sudo resize2fs /dev/sdc1
```

5. Mount the partition to the desired location, such as `/datadrive`:

```
sudo mount /dev/sdc1 /datadrive
```

6. To verify the data disk has been resized, use `df -h`. The following example output shows the data drive `/dev/sdc1` is now 200 GB:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdc1	197G	60M	187G	1%	/datadrive

Next steps

- If you need additional storage, you can also [add data disks to a Linux VM](#).
- For more information about disk encryption, see [Encrypt disks on a Linux VM using the Azure CLI](#).

Create a snapshot

4/23/2019 • 2 minutes to read • [Edit Online](#)

Take a snapshot of an OS or data disk for backup or to troubleshoot VM issues. A snapshot is a full, read-only copy of a VHD.

Use Azure CLI

The following example requires that you use [Cloud Shell](#) or have the Azure CLI installed.

The following steps show how to take a snapshot using the **az snapshot create** command with the **--source-disk** parameter. The following example assumes that there is a VM called *myVM* in the *myResourceGroup* resource group.

Get the disk ID using [az vm show](#).

```
osDiskId=$(az vm show \
    -g myResourceGroup \
    -n myVM \
    --query "storageProfile.osDisk.managedDisk.id" \
    -o tsv)
```

Take a snapshot named *osDisk-backup* using [az snapshot create](#).

```
az snapshot create \
    -g myResourceGroup \
    --source "$osDiskId" \
    --name osDisk-backup
```

NOTE

If you would like to store your snapshot in zone-resilient storage, you need to create it in a region that supports [availability zones](#) and include the **--sku Standard_ZRS** parameter.

You can see a list of the snapshots using [az snapshot list](#).

```
az snapshot list \
    -g myResourceGroup \
    -o table
```

Use Azure portal

1. Sign in to the [Azure portal](#).
2. Starting in the upper-left, click **Create a resource** and search for **snapshot**. Select **Snapshot** from the search results.
3. In the **Snapshot** blade, click **Create**.
4. Enter a **Name** for the snapshot.
5. Select an existing resource group or type the name for a new one.
6. For **Source disk**, select the managed disk to snapshot.

7. Select the **Account type** to use to store the snapshot. Use **Standard HDD** unless you need it stored on a high performing SSD.
8. Click **Create**.

Next steps

Create a virtual machine from a snapshot by creating a managed disk from the snapshot and then attaching the new managed disk as the OS disk. For more information, see the [Create a VM from a snapshot](#) script.

Back up Azure unmanaged VM disks with incremental snapshots

1/30/2019 • 7 minutes to read • [Edit Online](#)

Overview

Azure Storage provides the capability to take snapshots of blobs. Snapshots capture the blob state at that point in time. In this article, we describe a scenario in which you can maintain backups of virtual machine disks using snapshots. You can use this methodology when you choose not to use Azure Backup and Recovery Service, and wish to create a custom backup strategy for your virtual machine disks.

Azure virtual machine disks are stored as page blobs in Azure Storage. Since we are describing a backup strategy for virtual machine disks in this article, we refer to snapshots in the context of page blobs. To learn more about snapshots, refer to [Creating a Snapshot of a Blob](#).

What is a snapshot?

A blob snapshot is a read-only version of a blob that is captured at a point in time. Once a snapshot has been created, it can be read, copied, or deleted, but not modified. Snapshots provide a way to back up a blob as it appears at a moment in time. Until REST version 2015-04-05, you had the ability to copy full snapshots. With the REST version 2015-07-08 and above, you can also copy incremental snapshots.

Full snapshot copy

Snapshots can be copied to another storage account as a blob to keep backups of the base blob. You can also copy a snapshot over its base blob, which is like restoring the blob to an earlier version. When a snapshot is copied from one storage account to another, it occupies the same space as the base page blob. Therefore, copying whole snapshots from one storage account to another is slow and consumes much space in the target storage account.

NOTE

If you copy the base blob to another destination, the snapshots of the blob are not copied along with it. Similarly, if you overwrite a base blob with a copy, snapshots associated with the base blob are not affected and stay intact under the base blob name.

Back up disks using snapshots

As a backup strategy for your virtual machine disks, you can take periodic snapshots of the disk or page blob, and copy them to another storage account using tools like [Copy Blob](#) operation or [AzCopy](#). You can copy a snapshot to a destination page blob with a different name. The resulting destination page blob is a writeable page blob and not a snapshot. Later in this article, we describe steps to take backups of virtual machine disks using snapshots.

Restore disks using snapshots

When it is time to restore your disk to a stable version that was previously captured in one of the backup snapshots, you can copy a snapshot over the base page blob. After the snapshot is promoted to the base page blob, the snapshot remains, but its source is overwritten with a copy that can be both read and written. Later in this article we describe steps to restore a previous version of your disk from its snapshot.

Implementing full snapshot copy

You can implement a full snapshot copy by doing the following,

- First, take a snapshot of the base blob using the [Snapshot Blob](#) operation.
- Then, copy the snapshot to a target storage account using [Copy Blob](#).
- Repeat this process to maintain backup copies of your base blob.

Incremental snapshot copy

The new feature in the [GetPageRanges](#) API provides a much better way to back up the snapshots of your page blobs or disks. The API returns the list of changes between the base blob and the snapshots, which reduces the amount of storage space used on the backup account. The API supports page blobs on Premium Storage as well as Standard Storage. Using this API, you can build faster and more efficient backup solutions for Azure VMs. This API will be available with the REST version 2015-07-08 and higher.

Incremental Snapshot Copy allows you to copy from one storage account to another the difference between,

- Base blob and its Snapshot OR
- Any two snapshots of the base blob

Provided the following conditions are met,

- The blob was created on Jan-1-2016 or later.
- The blob was not overwritten with [PutPage](#) or [Copy Blob](#) between two snapshots.

Note: This feature is available for Premium and Standard Azure Page Blobs.

When you have a custom backup strategy using snapshots, copying the snapshots from one storage account to another can be slow and can consume much storage space. Instead of copying the entire snapshot to a backup storage account, you can write the difference between consecutive snapshots to a backup page blob. This way, the time to copy and the space to store backups is substantially reduced.

Implementing Incremental Snapshot Copy

You can implement incremental snapshot copy by doing the following,

- Take a snapshot of the base blob using [Snapshot Blob](#).
- Copy the snapshot to the target backup storage account in same or any other Azure region using [Copy Blob](#).
This is the backup page blob. Take a snapshot of the backup page blob and store it in the backup account.
- Take another snapshot of the base blob using [Snapshot Blob](#).
- Get the difference between the first and second snapshots of the base blob using [GetPageRanges](#). Use the new parameter **prevsnapshot**, to specify the DateTime value of the snapshot you want to get the difference with.
When this parameter is present, the REST response includes only the pages that were changed between target snapshot and previous snapshot including clear pages.
- Use [PutPage](#) to apply these changes to the backup page blob.
- Finally, take a snapshot of the backup page blob and store it in the backup storage account.

In the next section, we will describe in more detail how you can maintain backups of disks using Incremental Snapshot Copy

Scenario

In this section, we describe a scenario that involves a custom backup strategy for virtual machine disks using snapshots.

Consider a DS-series Azure VM with a premium storage P30 disk attached. The P30 disk called *mypremiumdisk* is stored in a premium storage account called *mypremiumaccount*. A standard storage account called *mybackupstdaccount* is used for storing the backup of *mypremiumdisk*. We would like to keep a snapshot of *mypremiumdisk* every 12 hours.

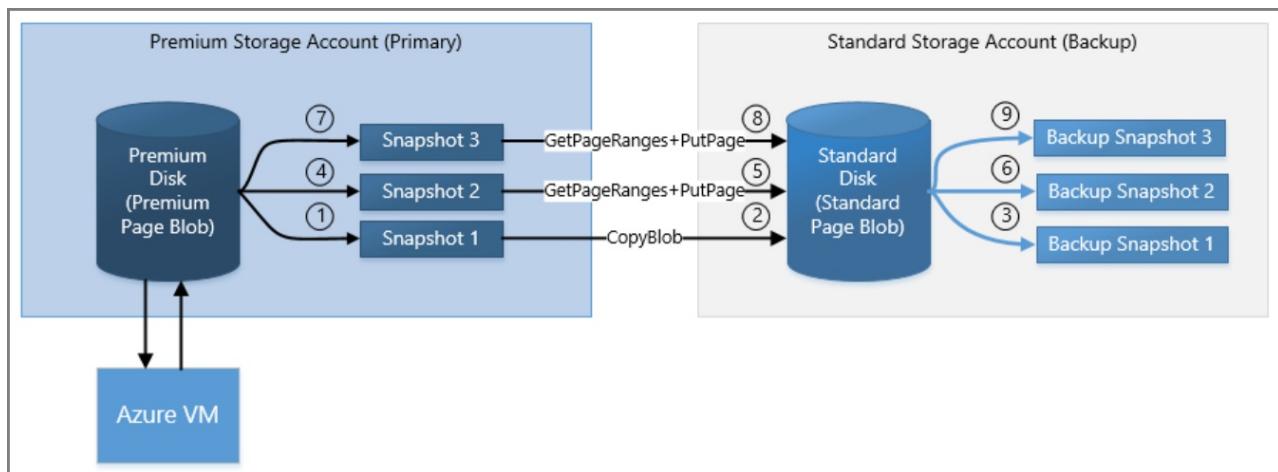
To learn about creating a storage account, see [Create a storage account](#).

To learn about backing up Azure VMs, refer to [Plan Azure VM backups](#).

Steps to maintain backups of a disk using incremental snapshots

The following steps describe how to take snapshots of *mypremiumdisk* and maintain the backups in *mybackupsdaccount*. The backup is a standard page blob called *mybackupsdpageblob*. The backup page blob always reflects the same state as the last snapshot of *mypremiumdisk*.

1. Create the backup page blob for your premium storage disk, by taking a snapshot of *mypremiumdisk* called *mypremiumdisk_ss1*.
2. Copy this snapshot to *mybackupsdaccount* as a page blob called *mybackupsdpageblob*.
3. Take a snapshot of *mybackupsdpageblob* called *mybackupsdpageblob_ss1*, using [Snapshot Blob](#) and store it in *mybackupsdaccount*.
4. During the backup window, create another snapshot of *mypremiumdisk*, say *mypremiumdisk_ss2*, and store it in *mypremiumaccount*.
5. Get the incremental changes between the two snapshots, *mypremiumdisk_ss2* and *mypremiumdisk_ss1*, using [GetPageRanges](#) on *mypremiumdisk_ss2* with the **prevsnapshot** parameter set to the timestamp of *mypremiumdisk_ss1*. Write these incremental changes to the backup page blob *mybackupsdpageblob* in *mybackupsdaccount*. If there are deleted ranges in the incremental changes, they must be cleared from the backup page blob. Use [PutPage](#) to write incremental changes to the backup page blob.
6. Take a snapshot of the backup page blob *mybackupsdpageblob*, called *mybackupsdpageblob_ss2*. Delete the previous snapshot *mypremiumdisk_ss1* from premium storage account.
7. Repeat steps 4–6 every backup window. In this way, you can maintain backups of *mypremiumdisk* in a standard storage account.



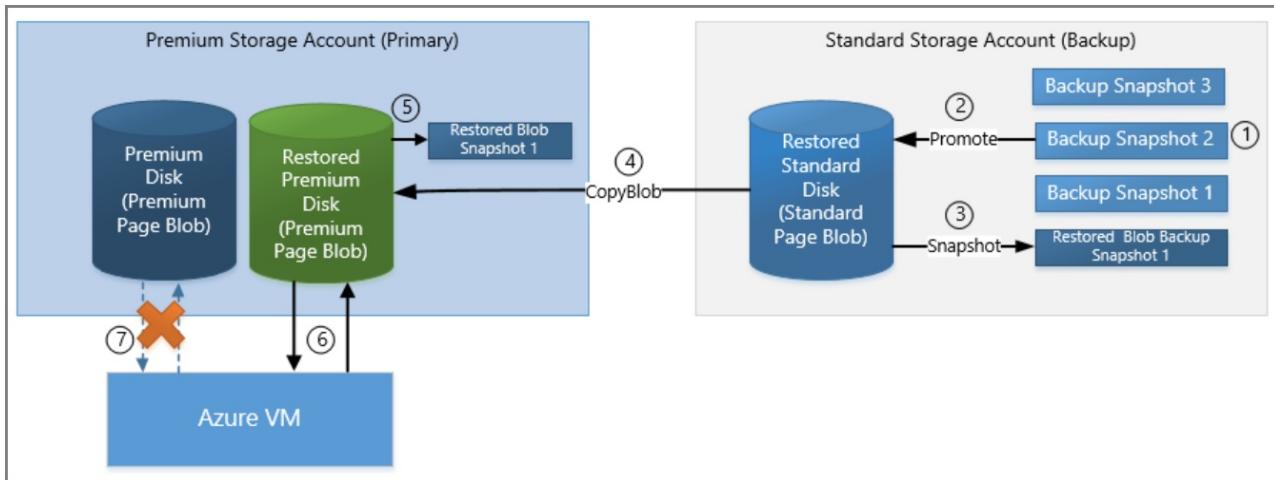
Steps to restore a disk from snapshots

The following steps, describe how to restore the premium disk, *mypremiumdisk* to an earlier snapshot from the backup storage account *mybackupsdaccount*.

1. Identify the point in time that you wish to restore the premium disk to. Let's say that it is snapshot *mybackupsdpageblob_ss2*, which is stored in the backup storage account *mybackupsdaccount*.
2. In *mybackupsdaccount*, promote the snapshot *mybackupsdpageblob_ss2* as the new backup base page blob *mybackupsdpageblobrestored*.
3. Take a snapshot of this restored backup page blob, called *mybackupsdpageblobrestored_ss1*.
4. Copy the restored page blob *mybackupsdpageblobrestored* from *mybackupsdaccount* to *mypremiumaccount* as the new premium disk *mypremiumdiskrestored*.
5. Take a snapshot of *mypremiumdiskrestored*, called *mypremiumdiskrestored_ss1* for making future incremental

backups.

6. Point the DS series VM to the restored disk *mypremiumdiskrestored* and detach the old *mypremiumdisk* from the VM.
7. Begin the Backup process described in previous section for the restored disk *mypremiumdiskrestored*, using the *mybackupstdpageblobrestored* as the backup page blob.



Next Steps

Use the following links to learn more about creating snapshots of a blob and planning your VM backup infrastructure.

- [Creating a Snapshot of a Blob](#)
- [Plan your VM Backup Infrastructure](#)

Migrate to Premium Storage by using Azure Site Recovery

3/14/2019 • 11 minutes to read • [Edit Online](#)

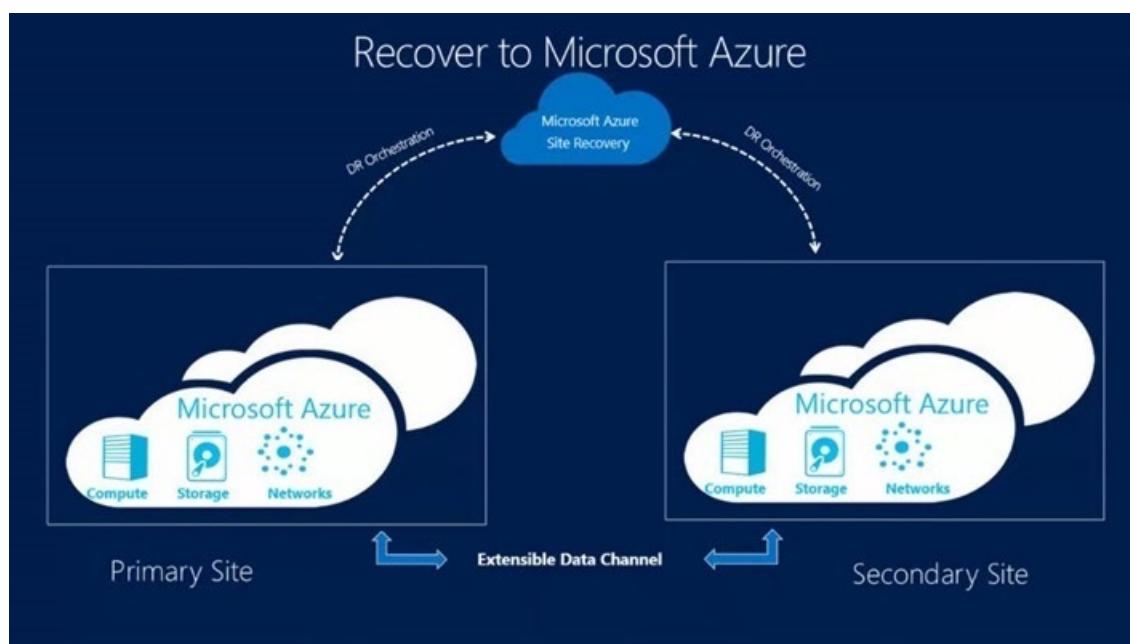
Azure premium SSDs delivers high-performance, low-latency disk support for virtual machines (VMs) that are running I/O-intensive workloads. This guide helps you migrate your VM disks from a standard storage account to a premium storage account by using [Azure Site Recovery](#).

Site Recovery is an Azure service that contributes to your strategy for business continuity and disaster recovery by orchestrating the replication of on-premises physical servers and VMs to the cloud (Azure) or to a secondary datacenter. When outages occur in your primary location, you fail over to the secondary location to keep applications and workloads available. You fail back to your primary location when it returns to normal operation.

Site Recovery provides test failovers to support disaster recovery drills without affecting production environments. You can run failovers with minimal data loss (depending on replication frequency) for unexpected disasters. In the scenario of migrating to Premium Storage, you can use the [failover in Site Recovery](#) to migrate target disks to a premium storage account.

We recommend migrating to Premium Storage by using Site Recovery because this option provides minimal downtime. This option also avoids the manual execution of copying disks and creating new VMs. Site Recovery will systematically copy your disks and create new VMs during failover.

Site Recovery supports a number of types of failover with minimal or no downtime. To plan your downtime and estimate data loss, see the [types of failover in Site Recovery](#). If you [prepare to connect to Azure VMs after failover](#), you should be able to connect to the Azure VM by using RDP after failover.



Azure Site Recovery components

These Site Recovery components are relevant to this migration scenario:

- **Configuration server** is an Azure VM that coordinates communication and manages data replication and recovery processes. On this VM, you run a single setup file to install the configuration server and an additional component, called a process server, as a replication gateway. Read about [configuration server](#)

[prerequisites](#). You set up the configuration server only once, and you can use it for all migrations to the same region.

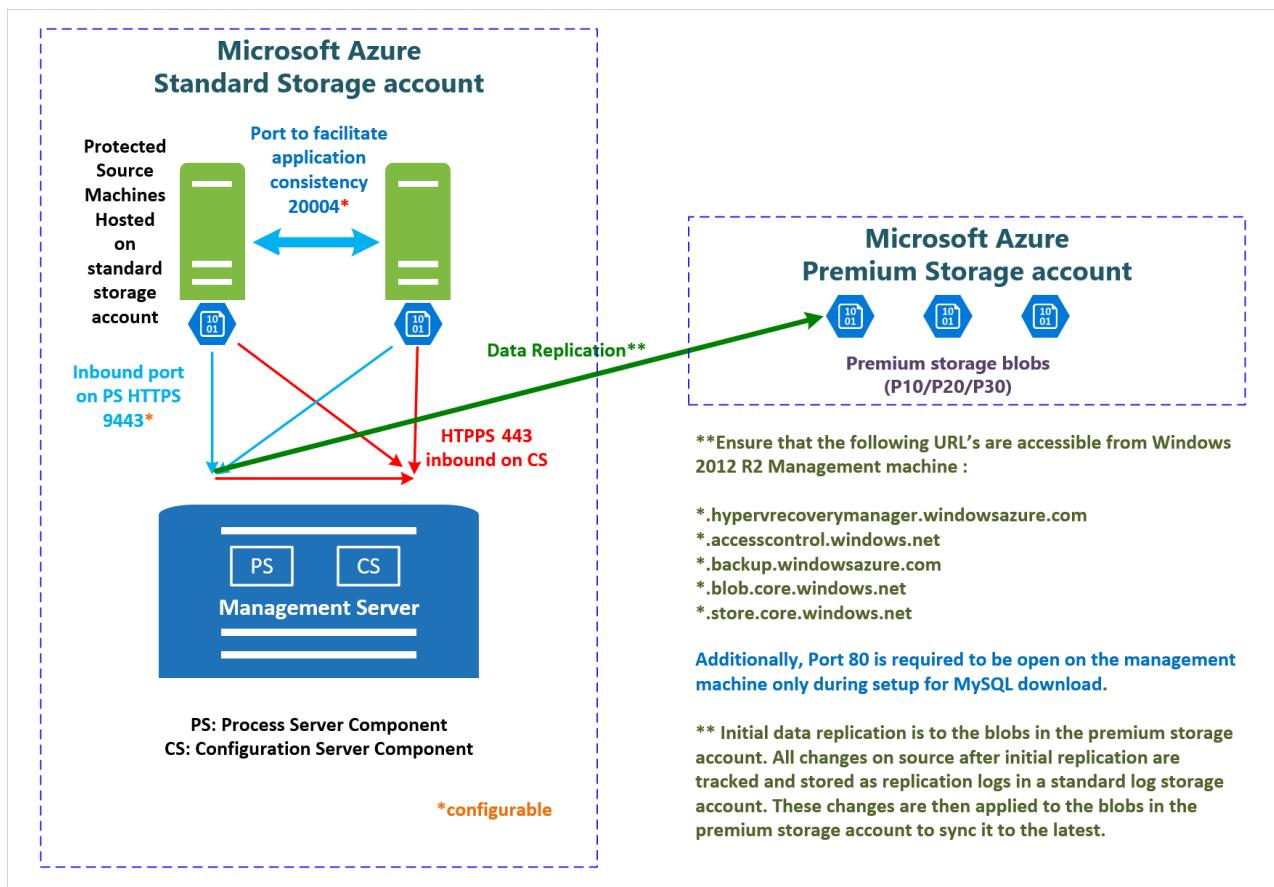
- **Process server** is a replication gateway that:

1. Receives replication data from source VMs.
2. Optimizes the data with caching, compression, and encryption.
3. Sends the data to a storage account.

It also handles push installation of the mobility service to source VMs and performs automatic discovery of source VMs. The default process server is installed on the configuration server. You can deploy additional standalone process servers to scale your deployment. Read about [best practices for process server deployment](#) and [deploying additional process servers](#). You set up the process server only once, and you can use it for all migrations to the same region.

- **Mobility service** is a component that is deployed on every standard VM that you want to replicate. It captures data writes on the standard VM and forwards them to the process server. Read about [replicated machine prerequisites](#).

This graphic shows how these components interact:



NOTE

Site Recovery does not support the migration of Storage Spaces disks.

For additional components for other scenarios, see [Scenario architecture](#).

Azure essentials

These are the Azure requirements for this migration scenario:

- An Azure subscription.

- An Azure premium storage account to store replicated data.
- An Azure virtual network to which VMs will connect when they're created at failover. The Azure virtual network must be in the same region as the one in which Site Recovery runs.
- An Azure standard storage account to store replication logs. This can be the same storage account for the VM disks that are being migrated.

Prerequisites

- Understand the relevant migration scenario components in the preceding section.
- Plan your downtime by learning about [failover in Site Recovery](#).

Setup and migration steps

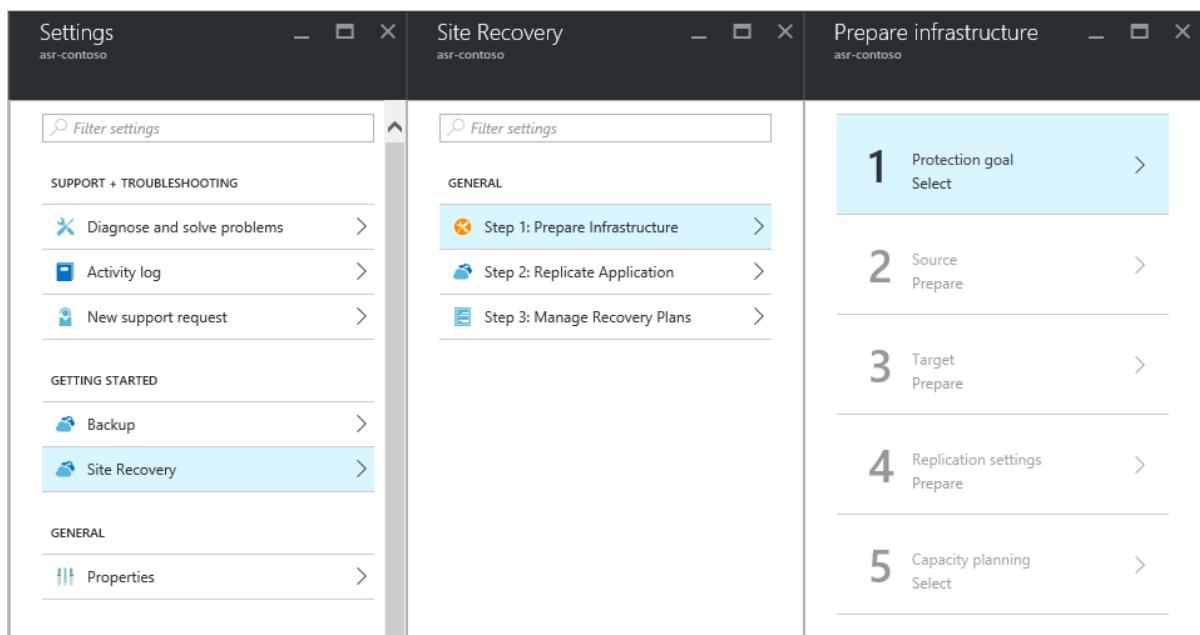
You can use Site Recovery to migrate Azure IaaS VMs between regions or within same region. The following instructions are tailored for this migration scenario from the article [Replicate VMware VMs or physical servers to Azure](#). Please follow the links for detailed steps in addition to the instructions in this article.

Step 1: Create a Recovery Services vault

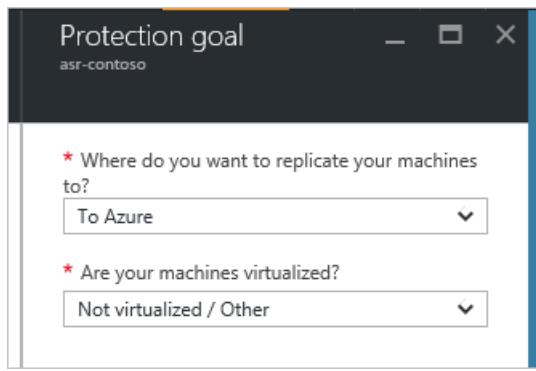
1. Open the [Azure portal](#).
2. Select **Create a resource > Management > Backup and Site Recovery (OMS)**. Alternatively, you can select **Browse > Recovery Services Vault > Add**.
3. Specify a region that VMs will be replicated to. For the purpose of migration in the same region, select the region where your source VMs and source storage accounts are.

Step 2: Choose your protection goals

1. On the VM where you want to install the configuration server, open the [Azure portal](#).
2. Go to **Recovery Services vaults > Settings > Site Recovery > Step 1: Prepare Infrastructure > Protection goal**.



3. Under **Protection goal**, in the first drop-down list, select **To Azure**. In the second drop-down list, select **Not virtualized / Other**, and then select **OK**.

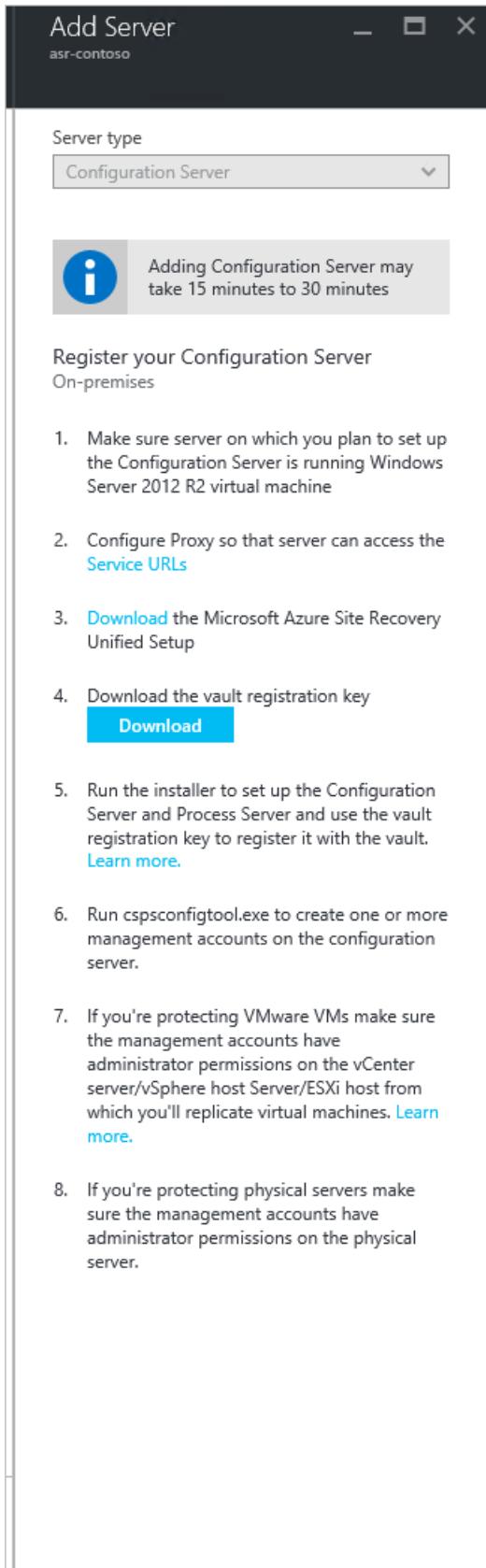


Step 3: Set up the source environment (configuration server)

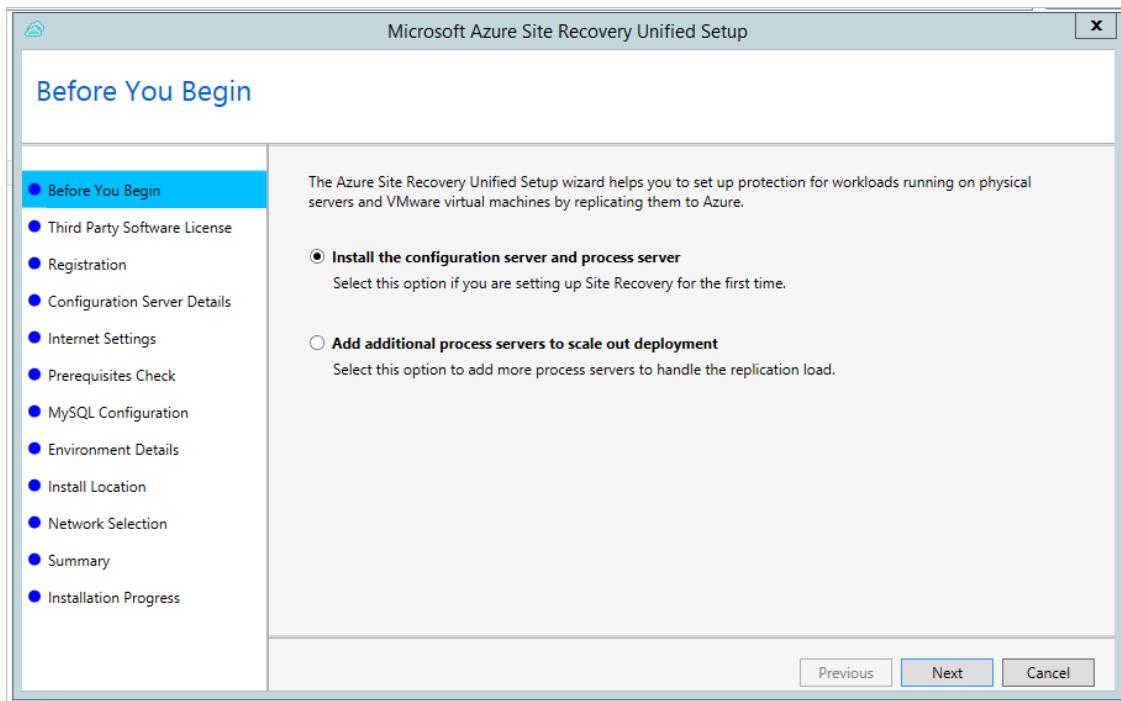
1. Download **Azure Site Recovery Unified Setup** and the vault registration key by going to the **Prepare infrastructure > Prepare source > Add Server** panes.

You will need the vault registration key to run the unified setup. The key is valid for five days after you generate it.

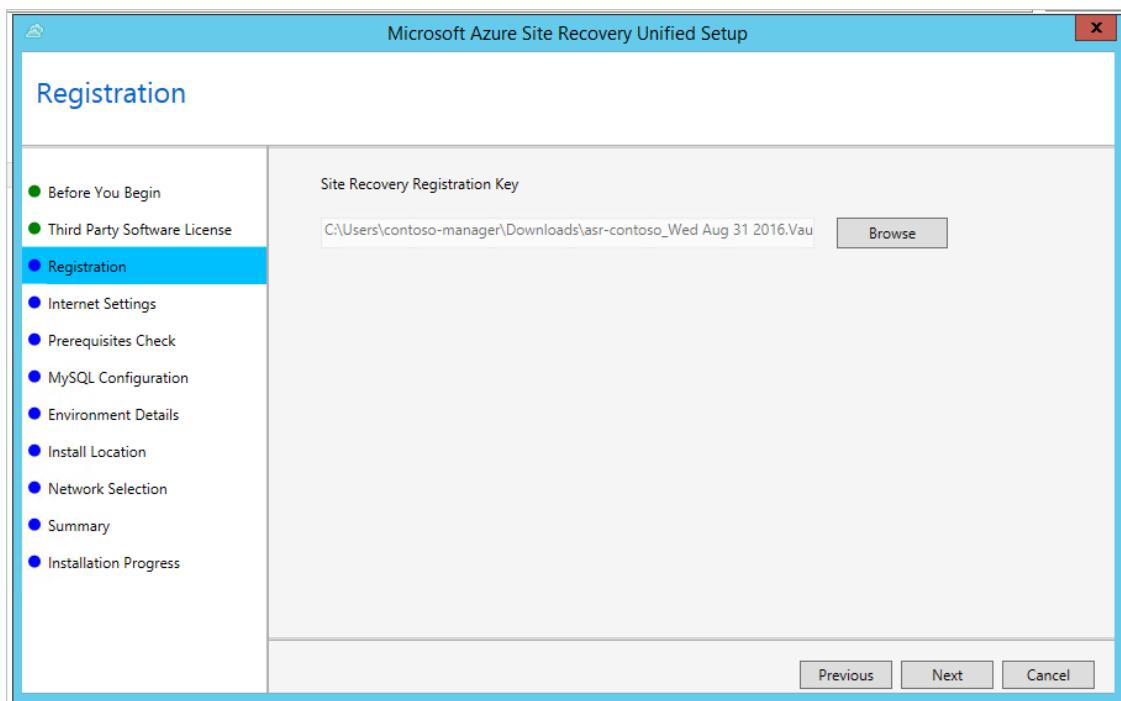
2. In the **Add Server** pane, add a configuration server.



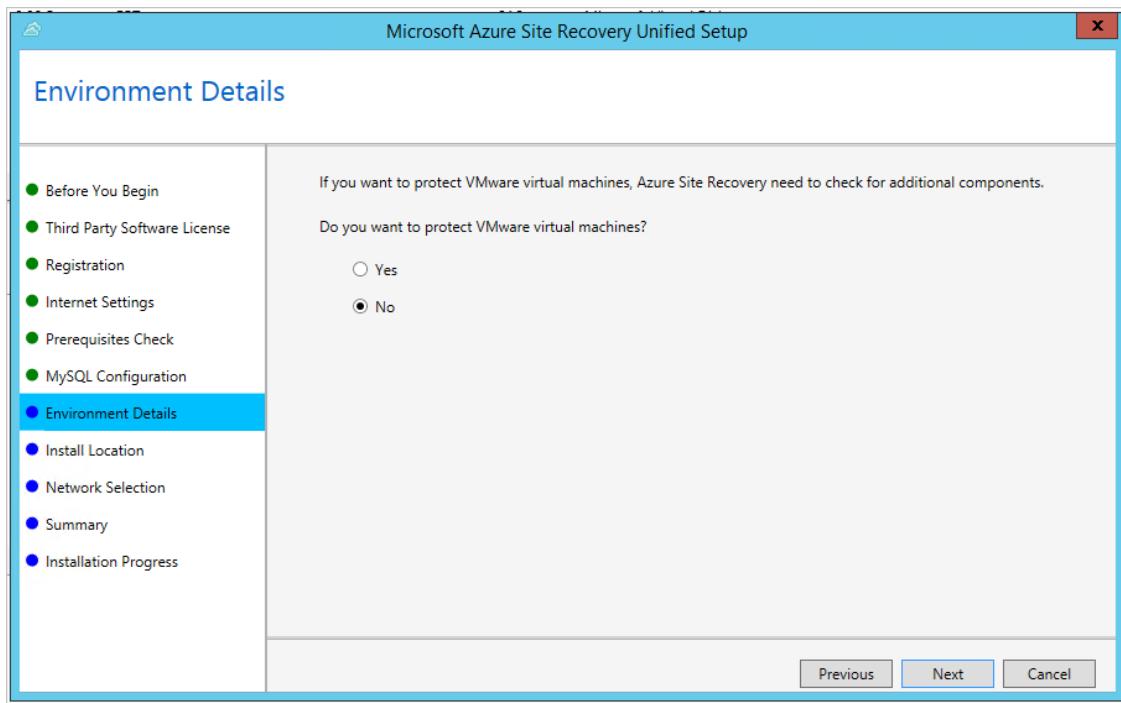
3. On the VM that you're using as the configuration server, run Unified Setup to install the configuration server and the process server. You can [walk through the screenshots](#) to complete the installation. You can refer to the following screenshots for steps specified for this migration scenario.
 - a. In **Before You Begin**, select **Install the configuration server and process server**.



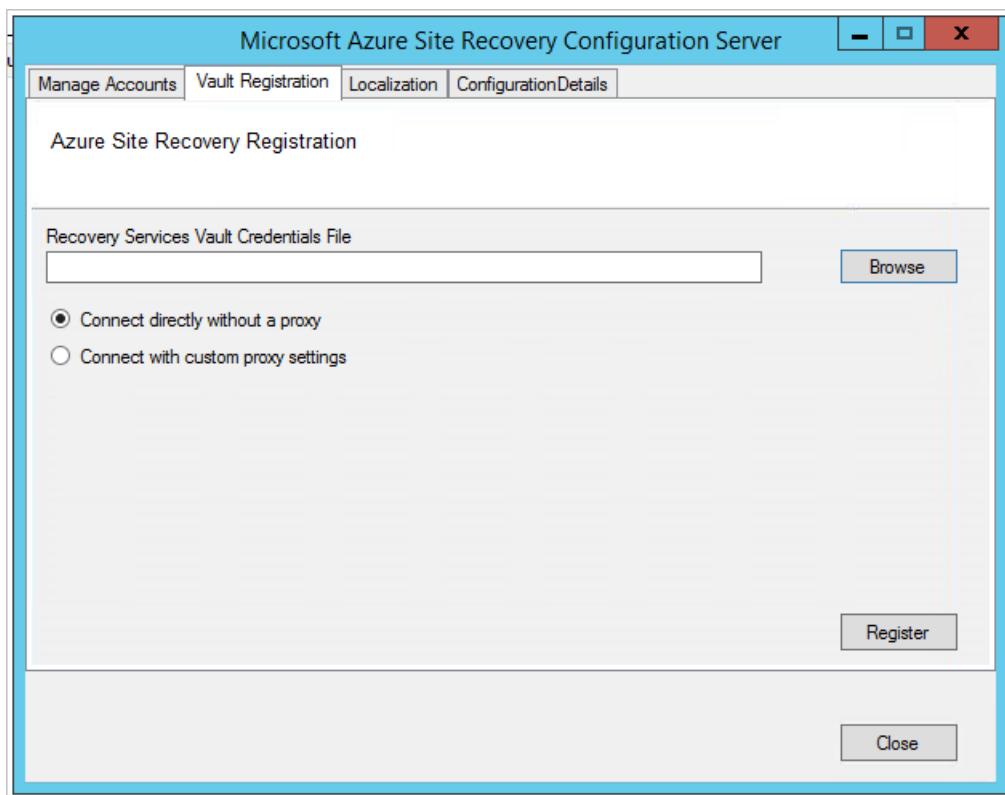
- b. In **Registration**, browse and select the registration key that you downloaded from the vault.



- c. In **Environment Details**, select whether you're going to replicate VMware VMs. For this migration scenario, choose **No**.



4. After the installation is complete, do the following in the **Microsoft Azure Site Recovery Configuration Server** window:
 - a. Use the **Manage Accounts** tab to create the account that Site Recovery can use for automatic discovery. (In the scenario about protecting physical machines, setting up the account isn't relevant, but you need at least one account to enable one of the following steps. In this case, you can name the account and password as any.)
 - b. Use the **Vault Registration** tab to upload the vault credential file.



Step 4: Set up the target environment

Select **Prepare infrastructure > Target**, and specify the deployment model that you want to use for VMs after failover. You can choose **Classic** or **Resource Manager**, depending on your scenario.

The screenshot shows two windows side-by-side. The left window is titled 'Prepare infrastructure' and lists five steps: 1. Protection goal (VMware VMs/physical servers t...), 2. Source (CONTOSO-CONFIG), 3. Target (Prepare, highlighted in blue), 4. Replication settings (Prepare), and 5. Capacity planning (Select). The right window is titled 'Target' and shows the configuration for step 3. It includes sections for 'Step 1 : Select Azure subscription' (Subscription: Visual Studio Enterprise, Deployment model: Classic selected), 'Step 2 : Ensure that at least one compatible Azure storage account exist' (Storage account(s): Found 5 compatible Azure storage accounts out of 6 available in the subscription), and 'Step 3 : Ensure that at least one compatible Azure virtual network exist' (Network(s): Found 2 compatible Azure virtual networks out of 2 available in the subscription).

Site Recovery checks that you have one or more compatible Azure storage accounts and networks.

NOTE

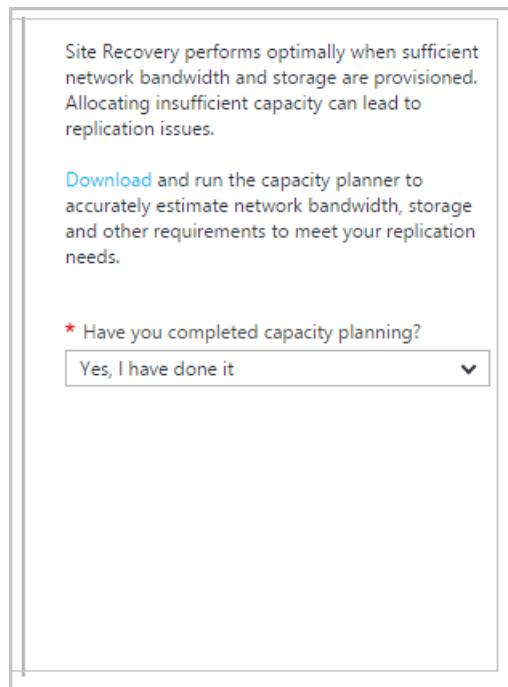
If you're using a premium storage account for replicated data, you need to set up an additional standard storage account to store replication logs.

Step 5: Set up replication settings

To verify that your configuration server is successfully associated with the replication policy that you create, follow [Set up replication settings](#).

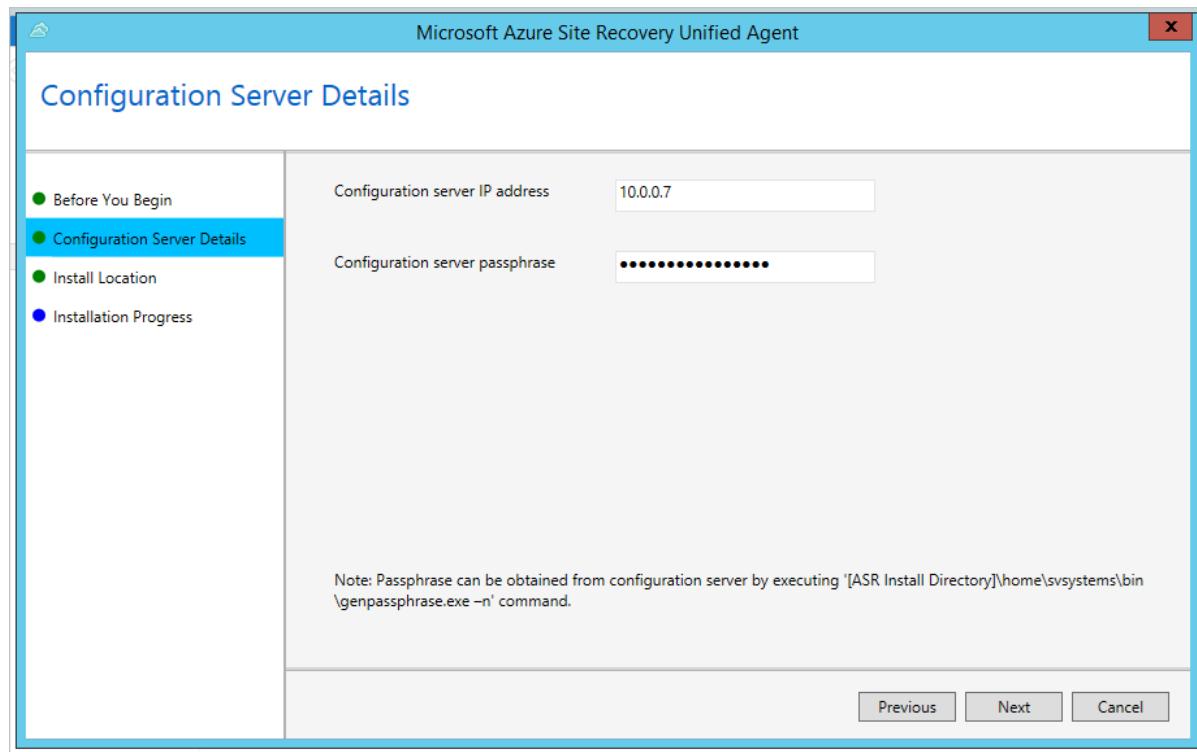
Step 6: Plan capacity

1. Use the [capacity planner](#) to accurately estimate network bandwidth, storage, and other requirements to meet your replication needs.
2. When you're done, select **Yes, I have done it** in **Have you completed capacity planning?**



Step 7: Install the mobility service and enable replication

1. You can choose to [push installation](#) to your source VMs or to [manually install the mobility service](#) on your source VMs. You can find the requirement of pushing installation and the path of the manual installer in the provided link. If you're doing a manual installation, you might need to use an internal IP address to find the configuration server.



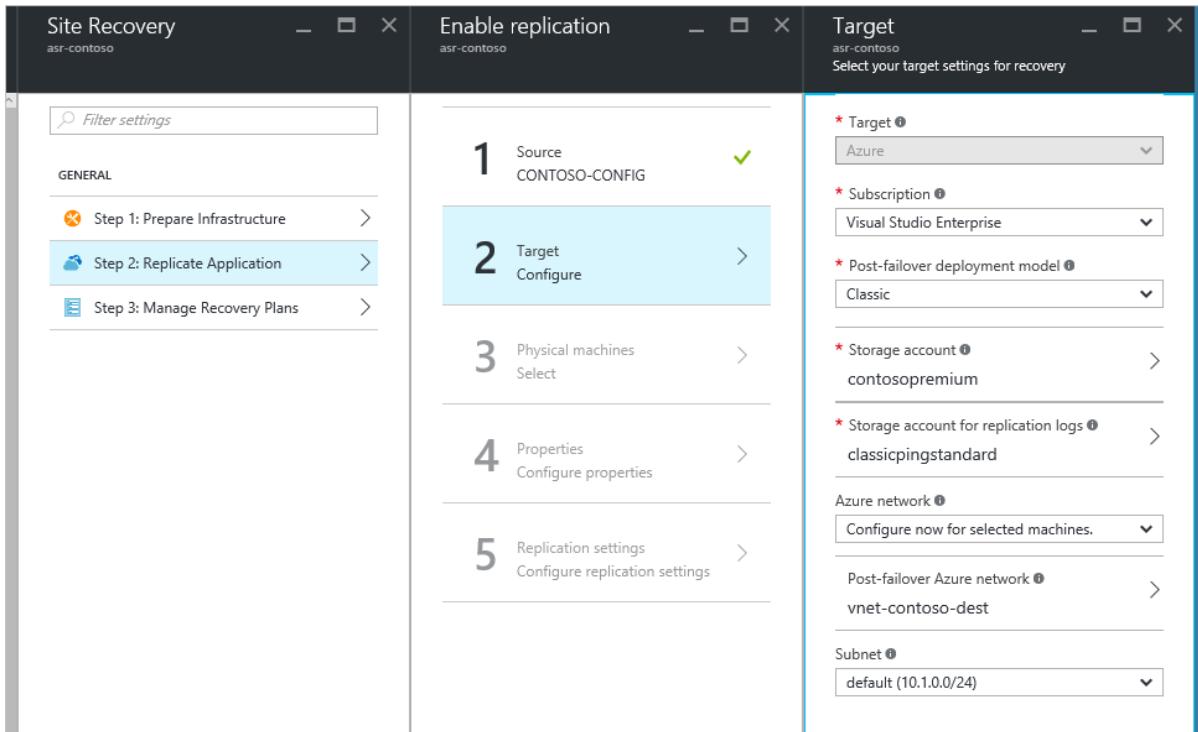
The failed-over VM will have two temporary disks: one from the primary VM and the other created during the provisioning of the VM in the recovery region. To exclude the temporary disk before replication, install the mobility service before you enable replication. To learn more about how to exclude the temporary disk, see [Exclude disks from replication](#).

2. Enable replication as follows:
 - a. Select **Replicate Application > Source**. After you've enabled replication for the first time, select **+Replicate** in the vault to enable replication for additional machines.

- b. In step 1, set up **Source** as your process server.
- c. In step 2, specify the post-failover deployment model, a premium storage account to migrate to, a standard storage account to save logs, and a virtual network to fail to.
- d. In step 3, add protected VMs by IP address. (You might need an internal IP address to find them.)
- e. In step 4, configure the properties by selecting the accounts that you set up previously on the process server.
- f. In step 5, choose the replication policy that you created previously in "Step 5: Set up replication settings."
- g. Select **OK**.

NOTE

When an Azure VM is deallocated and started again, there is no guarantee that it will get the same IP address. If the IP address of the configuration server/process server or the protected Azure VMs changes, the replication in this scenario might not work correctly.



When you design your Azure Storage environment, we recommend that you use separate storage accounts for each VM in an availability set. We recommend that you follow the best practice in the storage layer to [use multiple storage accounts for each availability set](#). Distributing VM disks to multiple storage accounts helps to improve storage availability and distributes the I/O across the Azure storage infrastructure.

If your VMs are in an availability set, instead of replicating disks of all VMs into one storage account, we highly recommend migrating multiple VMs multiple times. That way, the VMs in the same availability set do not share a single storage account. Use the **Enable Replication** pane to set up a destination storage account for each VM, one at a time.

You can choose a post-failover deployment model according to your need. If you choose Azure Resource Manager as your post-failover deployment model, you can fail over a VM (Resource Manager) to a VM (Resource Manager), or you can fail over a VM (classic) to a VM (Resource Manager).

Step 8: Run a test failover

To check whether your replication is complete, select your Site Recovery instance and then select **Settings > Replicated Items**. You will see the status and percentage of your replication process.

After initial replication is complete, run a test failover to validate your replication strategy. For detailed steps of a

test failover, see [Run a test failover in Site Recovery](#).

NOTE

Before you run any failover, make sure that your VMs and replication strategy meet the requirements. For more information about running a test failover, see [Test failover to Azure in Site Recovery](#).

You can see the status of your test failover in **Settings > Jobs > YOUR_FAILOVER_PLAN_NAME**. In the pane, you can see a breakdown of the steps and success/failure results. If the test failover fails at any step, select the step to check the error message.

Step 9: Run a failover

After the test failover is completed, run a failover to migrate your disks to Premium Storage and replicate the VM instances. Follow the detailed steps in [Run a failover](#).

Be sure to select **Shut down VMs and synchronize the latest data**. This option specifies that Site Recovery should try to shut down the protected VMs and synchronize the data so that the latest version of the data will be failed over. If you don't select this option or the attempt doesn't succeed, the failover will be from the latest available recovery point for the VM.

Site Recovery will create a VM instance whose type is the same as or similar to a Premium Storage-capable VM. You can check the performance and price of various VM instances by going to [Windows Virtual Machines Pricing](#) or [Linux Virtual Machines Pricing](#).

Post-migration steps

1. **Configure replicated VMs to the availability set if applicable.** Site Recovery does not support migrating VMs along with the availability set. Depending on the deployment of your replicated VM, do one of the following:
 - For a VM created through the classic deployment model: Add the VM to the availability set in the Azure portal. For detailed steps, go to [Add an existing virtual machine to an availability set](#).
 - For a VM created through the Resource Manager deployment model: Save your configuration of the VM and then delete and re-create the VMs in the availability set. To do so, use the script at [Set Azure Resource Manager VM Availability Set](#). Before you run this script, check its limitations and plan your downtime.
2. **Delete old VMs and disks.** Make sure that the Premium disks are consistent with source disks and that the new VMs perform the same function as the source VMs. Delete the VM and delete the disks from your source storage accounts in the Azure portal. If there's a problem in which the disk is not deleted even though you deleted the VM, see [Troubleshoot storage resource deletion errors](#).
3. **Clean the Azure Site Recovery infrastructure.** If Site Recovery is no longer needed, you can clean its infrastructure. Delete replicated items, the configuration server, and the recovery policy, and then delete the Azure Site Recovery vault.

Troubleshooting

- [Monitor and troubleshoot protection for virtual machines and physical servers](#)
- [Microsoft Azure Site Recovery forum](#)

Next steps

For specific scenarios for migrating virtual machines, see the following resources:

- [Migrate Azure Virtual Machines between Storage Accounts](#)
- [Upload a Linux virtual hard disk](#)
- [Migrating Virtual Machines from Amazon AWS to Microsoft Azure](#)

Also, see the following resources to learn more about Azure Storage and Azure Virtual Machines:

- [Azure Storage](#)
- [Azure Virtual Machines](#)
- [Select a disk type for IaaS VMs](#)

Convert a Linux virtual machine from unmanaged disks to managed disks

5/30/2019 • 4 minutes to read • [Edit Online](#)

If you have existing Linux virtual machines (VMs) that use unmanaged disks, you can convert the VMs to use [Azure Managed Disks](#). This process converts both the OS disk and any attached data disks.

This article shows you how to convert VMs by using the Azure CLI. If you need to install or upgrade it, see [Install Azure CLI](#).

Before you begin

- Review [the FAQ about migration to Managed Disks](#).
- The conversion requires a restart of the VM, so schedule the migration of your VMs during a pre-existing maintenance window.
- The conversion is not reversible.
- Be aware that any users with the [Virtual Machine Contributor](#) role will not be able to change the VM size (as they could pre-conversion). This is because VMs with managed disks require the user to have the Microsoft.Compute/disks/write permission on the OS disks.
- Be sure to test the conversion. Migrate a test virtual machine before you perform the migration in production.
- During the conversion, you deallocate the VM. The VM receives a new IP address when it is started after the conversion. If needed, you can [assign a static IP address](#) to the VM.
- Review the minimum version of the Azure VM agent required to support the conversion process. For information on how to check and update your agent version, see [Minimum version support for VM agents in Azure](#)
- The original VHDs and the storage account used by the VM before conversion are not deleted. They continue to incur charges. To avoid being billed for these artifacts, delete the original VHD blobs after you verify that the conversion is complete. If you need to find these unattached disks in order to delete them, see our article [Find and delete unattached Azure managed and unmanaged disks](#).

Convert single-instance VMs

This section covers how to convert single-instance Azure VMs from unmanaged disks to managed disks. (If your VMs are in an availability set, see the next section.) You can use this process to convert the VMs from premium (SSD) unmanaged disks to premium managed disks, or from standard (HDD) unmanaged disks to standard managed disks.

1. Deallocate the VM by using `az vm deallocate`. The following example deallocates the VM named `myVM` in the resource group named `myResourceGroup` :

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

2. Convert the VM to managed disks by using `az vm convert`. The following process converts the VM named

`myVM`, including the OS disk and any data disks:

```
az vm convert --resource-group myResourceGroup --name myVM
```

3. Start the VM after the conversion to managed disks by using `az vm start`. The following example starts the VM named `myVM` in the resource group named `myResourceGroup`.

```
az vm start --resource-group myResourceGroup --name myVM
```

Convert VMs in an availability set

If the VMs that you want to convert to managed disks are in an availability set, you first need to convert the availability set to a managed availability set.

All VMs in the availability set must be deallocated before you convert the availability set. Plan to convert all VMs to managed disks after the availability set itself has been converted to a managed availability set. Then, start all the VMs and continue operating as normal.

1. List all VMs in an availability set by using `az vm availability-set list`. The following example lists all VMs in the availability set named `myAvailabilitySet` in the resource group named `myResourceGroup`:

```
az vm availability-set show \  
    --resource-group myResourceGroup \  
    --name myAvailabilitySet \  
    --query [virtualMachines[*].id] \  
    --output table
```

2. Deallocate all the VMs by using `az vm deallocate`. The following example deallocates the VM named `myVM` in the resource group named `myResourceGroup`:

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

3. Convert the availability set by using `az vm availability-set convert`. The following example converts the availability set named `myAvailabilitySet` in the resource group named `myResourceGroup`:

```
az vm availability-set convert \  
    --resource-group myResourceGroup \  
    --name myAvailabilitySet
```

4. Convert all the VMs to managed disks by using `az vm convert`. The following process converts the VM named `myVM`, including the OS disk and any data disks:

```
az vm convert --resource-group myResourceGroup --name myVM
```

5. Start all the VMs after the conversion to managed disks by using `az vm start`. The following example starts the VM named `myVM` in the resource group named `myResourceGroup`:

```
az vm start --resource-group myResourceGroup --name myVM
```

Convert using the Azure portal

You can also convert unmanaged disks to managed disks using the Azure portal.

1. Sign in to the [Azure portal](#).
2. Select the VM from the list of VMs in the portal.
3. In the blade for the VM, select **Disks** from the menu.
4. At the top of the **Disks** blade, select **Migrate to managed disks**.
5. If your VM is in an availability set, there will be a warning on the **Migrate to managed disks** blade that you need to convert the availability set first. The warning should have a link you can click to convert the availability set. Once the availability set is converted or if your VM is not in an availability set, click **Migrate** to start the process of migrating your disks to managed disks.

The VM will be stopped and restarted after migration is complete.

Next steps

For more information about storage options, see [Azure Managed Disks overview](#).

Convert Azure managed disks storage from Standard to Premium or Premium to Standard

5/30/2019 • 4 minutes to read • [Edit Online](#)

There are four disk types of Azure managed disks: Azure ultra SSDs (preview), premium SSD, standard SSD, and standard HDD. You can switch between the three GA disk types (premium SSD, standard SSD, and standard HDD) based on your performance needs. You are not yet able to switch from or to an ultra SSD, you must deploy a new one.

This functionality is not supported for unmanaged disks. But you can easily [convert an unmanaged disk to a managed disk](#) to be able to switch between disk types.

This article shows how to convert managed disks from Standard to Premium or Premium to Standard by using the Azure CLI. To install or upgrade the tool, see [Install Azure CLI](#).

Before you begin

- Disk conversion requires a restart of the virtual machine (VM), so schedule the migration of your disk storage during a pre-existing maintenance window.
- For unmanaged disks, first [convert to managed disks](#) so you can switch between storage options.

Switch all managed disks of a VM between Premium and Standard

This example shows how to convert all of a VM's disks from Standard to Premium storage or from Premium to Standard storage. To use Premium managed disks, your VM must use a [VM size](#) that supports Premium storage. This example also switches to a size that supports Premium storage.

```

#resource group that contains the virtual machine
rgName='yourResourceGroup'

#Name of the virtual machine
vmName='yourVM'

#Premium capable size
#Required only if converting from Standard to Premium
size='Standard_DS2_v2'

#Choose between Standard_LRS and Premium_LRS based on your scenario
sku='Premium_LRS'

#Deallocate the VM before changing the size of the VM
az vm deallocate --name $vmName --resource-group $rgName

#Change the VM size to a size that supports Premium storage
#Skip this step if converting storage from Premium to Standard
az vm resize --resource-group $rgName --name $vmName --size $size

#Update the SKU of all the data disks
az vm show -n $vmName -g $rgName --query storageProfile.dataDisks[*].managedDisk -o tsv \
| awk -v sku=$sku '{system("az disk update --sku \"sku\" --ids \"$1\"")}' 

#Update the SKU of the OS disk
az vm show -n $vmName -g $rgName --query storageProfile.osDisk.managedDisk -o tsv \
| awk -v sku=$sku '{system("az disk update --sku \"sku\" --ids \"$1\"")}' 

az vm start --name $vmName --resource-group $rgName

```

Switch individual managed disks between Standard and Premium

For your dev/test workload, you might want to have a mix of Standard and Premium disks to reduce your costs. You can choose to upgrade only those disks that need better performance. This example shows how to convert a single VM disk from Standard to Premium storage or from Premium to Standard storage. To use Premium managed disks, your VM must use a [VM size](#) that supports Premium storage. This example also switches to a size that supports Premium storage.

```

#resource group that contains the managed disk
rgName='yourResourceGroup'

#Name of your managed disk
diskName='yourManagedDiskName'

#Premium capable size
#Required only if converting from Standard to Premium
size='Standard_DS2_v2'

#Choose between Standard_LRS and Premium_LRS based on your scenario
sku='Premium_LRS'

#Get the parent VM Id
vmId=$(az disk show --name $diskName --resource-group $rgName --query managedBy --output tsv)

#Deallocate the VM before changing the size of the VM
az vm deallocate --ids $vmId

#Change the VM size to a size that supports Premium storage
#Skip this step if converting storage from Premium to Standard
az vm resize --ids $vmId --size $size

# Update the SKU
az disk update --sku $sku --name $diskName --resource-group $rgName

az vm start --ids $vmId

```

Switch managed disks between Standard HDD and Standard SSD

This example shows how to convert a single VM disk from Standard HDD to Standard SSD or from Standard SSD to Standard HDD.

```

#resource group that contains the managed disk
rgName='yourResourceGroup'

#Name of your managed disk
diskName='yourManagedDiskName'

#Choose between Standard_LRS and StandardSSD_LRS based on your scenario
sku='StandardSSD_LRS'

#Get the parent VM ID
vmId=$(az disk show --name $diskName --resource-group $rgName --query managedBy --output tsv)

#Deallocate the VM before changing the disk type
az vm deallocate --ids $vmId

# Update the SKU
az disk update --sku $sku --name $diskName --resource-group $rgName

az vm start --ids $vmId

```

Switch managed disks between Standard and Premium in Azure portal

Follow these steps:

1. Sign in to the [Azure portal](#).
2. Select the VM from the list of **Virtual machines**.

3. If the VM isn't stopped, select **Stop** at the top of the VM **Overview** pane, and wait for the VM to stop.
4. In the pane for the VM, select **Disks** from the menu.
5. Select the disk that you want to convert.
6. Select **Configuration** from the menu.
7. Change the **Account type** from **Standard HDD** to **Premium SSD** or from **Premium SSD** to **Standard HDD**.
8. Select **Save**, and close the disk pane.

The update of the disk type is instantaneous. You can restart your VM after the conversion.

Next steps

Make a read-only copy of a VM by using [snapshots](#).

Move files to and from a Linux VM using SCP

1/30/2019 • 2 minutes to read • [Edit Online](#)

This article shows how to move files from your workstation up to an Azure Linux VM, or from an Azure Linux VM down to your workstation, using Secure Copy (SCP). Moving files between your workstation and a Linux VM, quickly and securely, is critical for managing your Azure infrastructure.

For this article, you need a Linux VM deployed in Azure using [SSH public and private key files](#). You also need an SCP client for your local computer. It is built on top of SSH and included in the default Bash shell of most Linux and Mac computers and some Windows shells.

Quick commands

Copy a file up to the Linux VM

```
scp file azureuser@azurehost:directory/targetfile
```

Copy a file down from the Linux VM

```
scp azureuser@azurehost:directory/file targetfile
```

Detailed walkthrough

As examples, we move an Azure configuration file up to a Linux VM and pull down a log file directory, both using SCP and SSH keys.

SSH key pair authentication

SCP uses SSH for the transport layer. SSH handles the authentication on the destination host, and it moves the file in an encrypted tunnel provided by default with SSH. For SSH authentication, usernames and passwords can be used. However, SSH public and private key authentication are recommended as a security best practice. Once SSH has authenticated the connection, SCP then begins copying the file. Using a properly configured `~/.ssh/config` and SSH public and private keys, the SCP connection can be established by just using a server name (or IP address). If you only have one SSH key, SCP looks for it in the `~/.ssh/` directory, and uses it by default to log in to the VM.

For more information on configuring your `~/.ssh/config` and SSH public and private keys, see [Create SSH keys](#).

SCP a file to a Linux VM

For the first example, we copy an Azure configuration file up to a Linux VM that is used to deploy automation. Because this file contains Azure API credentials, which include secrets, security is important. The encrypted tunnel provided by SSH protects the contents of the file.

The following command copies the local `.azure/config` file to an Azure VM with FQDN `myservereastus.cloudapp.azure.com`. The admin user name on the Azure VM is `azureuser`. The file is targeted to the `/home/azureuser/` directory. Substitute your own values in this command.

```
scp ~/.azure/config azureuser@myserver.eastus.cloudapp.com:/home/azureuser/config
```

SCP a directory from a Linux VM

For this example, we copy a directory of log files from the Linux VM down to your workstation. A log file may or may not contain sensitive or secret data. However, using SCP ensures the contents of the log files are encrypted. Using SCP to transfer the files is the easiest way to get the log directory and files down to your workstation while also being secure.

The following command copies files in the `/home/azureuser/logs/` directory on the Azure VM to the local `/tmp` directory:

```
scp -r azureuser@myserver.eastus.cloudapp.com:/home/azureuser/logs/. /tmp/
```

The `-r` cli flag instructs SCP to recursively copy the files and directories from the point of the directory listed in the command. Also notice that the command-line syntax is similar to a `cp` copy command.

Next steps

- [Manage users, SSH, and check or repair disks on Azure Linux VMs using the VMAccess Extension](#)

Enable Write Accelerator

2/21/2019 • 8 minutes to read • [Edit Online](#)

Write Accelerator is a disk capability for M-Series Virtual Machines (VMs) on Premium Storage with Azure Managed Disks exclusively. As the name states, the purpose of the functionality is to improve the I/O latency of writes against Azure Premium Storage. Write Accelerator is ideally suited where log file updates are required to persist to disk in a highly performant manner for modern databases.

Write Accelerator is generally available for M-series VMs in the Public Cloud.

Planning for using Write Accelerator

Write Accelerator should be used for the volumes that contain the transaction log or redo logs of a DBMS. It is not recommended to use Write Accelerator for the data volumes of a DBMS as the feature has been optimized to be used against log disks.

Write Accelerator only works in conjunction with [Azure managed disks](#).

IMPORTANT

Enabling Write Accelerator for the operating system disk of the VM will reboot the VM.

To enable Write Accelerator to an existing Azure disk that is NOT part of a volume build out of multiple disks with Windows disk or volume managers, Windows Storage Spaces, Windows Scale-out file server (SOFS), Linux LVM, or MDADM, the workload accessing the Azure disk needs to be shut down. Database applications using the Azure disk MUST be shut down.

If you want to enable or disable Write Accelerator for an existing volume that is built out of multiple Azure Premium Storage disks and striped using Windows disk or volume managers, Windows Storage Spaces, Windows Scale-out file server (SOFS), Linux LVM or MDADM, all disks building the volume must be enabled or disabled for Write Accelerator in separate steps.

Before enabling or disabling Write Accelerator in such a configuration, shut down the Azure VM.

Enabling Write Accelerator for OS disks should not be necessary for SAP-related VM configurations.

Restrictions when using Write Accelerator

When using Write Accelerator for an Azure disk/VHD, these restrictions apply:

- The Premium disk caching must be set to 'None' or 'Read Only'. All other caching modes are not supported.
- Snapshot are not currently supported for Write Accelerator-enabled disks. During backup, the Azure Backup service automatically excludes Write Accelerator-enabled disks attached to the VM.
- Only smaller I/O sizes (<=512 KiB) are taking the accelerated path. In workload situations where data is getting bulk loaded or where the transaction log buffers of the different DBMS are filled to a larger degree before getting persisted to the storage, chances are that the I/O written to disk is not taking the accelerated path.

There are limits of Azure Premium Storage VHDs per VM that can be supported by Write Accelerator. The current limits are:

VM SKU	NUMBER OF WRITE ACCELERATOR DISKS	WRITE ACCELERATOR DISK IOPS PER VM
M208ms_v2, M208s_v2	8	10000
M128ms, 128s	16	20000

VM SKU	NUMBER OF WRITE ACCELERATOR DISKS	WRITE ACCELERATOR DISK IOPS PER VM
M64ms, M64ls, M64s	8	10000
M32ms, M32ls, M32ts, M32s	4	5000
M16ms, M16s	2	2500
M8ms, M8s	1	1250

The IOPS limits are per VM and *not* per disk. All Write Accelerator disks share the same IOPS limit per VM.

Enabling Write Accelerator on a specific disk

The next few sections will describe how Write Accelerator can be enabled on Azure Premium Storage VHDS.

Prerequisites

The following prerequisites apply to the usage of Write Accelerator at this point in time:

- The disks you want to apply Azure Write Accelerator against need to be [Azure managed disks](#) on Premium Storage.
- You must be using an M-series VM

Enabling Azure Write Accelerator using Azure PowerShell

The Azure Power Shell module from version 5.5.0 include the changes to the relevant cmdlets to enable or disable Write Accelerator for specific Azure Premium Storage disks. In order to enable or deploy disks supported by Write Accelerator, the following Power Shell commands got changed, and extended to accept a parameter for Write Accelerator.

A new switch parameter, **-WriteAccelerator** has been added to the following cmdlets:

- [Set-AzVMOsDisk](#)
- [Add-AzVMDataDisk](#)
- [Set-AzVMDataDisk](#)
- [Add-AzVmssDataDisk](#)

Not giving the parameter sets the property to false and will deploy disks that have no support by Write Accelerator.

A new switch parameter, **-OsDiskWriteAccelerator** was added to the following cmdlets:

- [Set-AzVmssStorageProfile](#)

Not specifying the parameter sets the property to false by default, returning disks that don't leverage Write Accelerator.

A new optional Boolean (non-nullable) parameter, **-OsDiskWriteAccelerator** was added to the following cmdlets:

- [Update-AzVM](#)
- [Update-AzVmss](#)

Specify either \$true or \$false to control support of Azure Write Accelerator with the disks.

Examples of commands could look like:

```

New-AzVMConfig | Set-AzVMOSDisk | Add-AzVMDataDisk -Name "datadisk1" | Add-AzVMDataDisk -Name "logdisk1" -WriteAccelerator | New-AzVM

Get-AzVM | Update-AzVM -OsDiskWriteAccelerator $true

New-AzVmssConfig | Set-AzVmssStorageProfile -OsDiskWriteAccelerator | Add-AzVmssDataDisk -Name "datadisk1" -WriteAccelerator:$false | Add-AzVmssDataDisk -Name "logdisk1" -WriteAccelerator | New-AzVmss

Get-AzVmss | Update-AzVmss -OsDiskWriteAccelerator:$false

```

Two main scenarios can be scripted as shown in the following sections.

Adding a new disk supported by Write Accelerator using PowerShell

You can use this script to add a new disk to your VM. The disk created with this script uses Write Accelerator.

Replace `myVM`, `myWAVMs`, `log001`, size of the disk, and LunID of the disk with values appropriate for your specific deployment.

```

# Specify your VM Name
$vmName="myVM"
#Specify your Resource Group
$rgName = "myWAVMs"
#data disk name
$datadiskname = "log001"
#LUN Id
$lunid=8
#size
$size=1023
#Pulls the VM info for later
$vm=Get-AzVM -ResourceGroupName $rgname -Name $vmname
#add a new VM data disk
Add-AzVMDataDisk -CreateOption empty -DiskSizeInGB $size -Name $vmname-$datadiskname -VM $vm -Caching None -WriteAccelerator:$true -lun $lunid
#Updates the VM with the disk config - does not require a reboot
Update-AzVM -ResourceGroupName $rgname -VM $vm

```

Enabling Write Accelerator on an existing Azure disk using PowerShell

You can use this script to enable Write Accelerator on an existing disk. Replace `myVM`, `myWAVMs`, and `test-log001` with values appropriate for your specific deployment. The script adds Write Accelerator to an existing disk where the value for `$newstatus` is set to '\$true'. Using the value '\$false' will disable Write Accelerator on a given disk.

```

#Specify your VM Name
$vmName="myVM"
#Specify your Resource Group
$rgName = "myWAVMs"
#data disk name
$datadiskname = "test-log001"
#new Write Accelerator status ($true for enabled, $false for disabled)
$newstatus = $true
#Pulls the VM info for later
$vm=Get-AzVM -ResourceGroupName $rgname -Name $vmname
#add a new VM data disk
Set-AzVMDataDisk -VM $vm -Name $datadiskname -Caching None -WriteAccelerator:$newstatus
#Updates the VM with the disk config - does not require a reboot
Update-AzVM -ResourceGroupName $rgname -VM $vm

```

NOTE

Executing the script above will detach the disk specified, enable Write Accelerator against the disk, and then attach the disk again

Enabling Write Accelerator using the Azure portal

You can enable Write Accelerator via the portal where you specify your disk caching settings:

LUN	NAME	SIZE	STORAGE ACCOUNT TYPE	ENCRYPTION	HOST CACHING
0	WADisk1	1023 GiB	Premium_LRS	Not enabled	Read-only + Write Accelerator
1	WADisk2	1023 GiB	Premium_LRS	Not enabled	None + Write Accelerator

Enabling Write Accelerator using the Azure CLI

You can use the [Azure CLI](#) to enable Write Accelerator.

To enable Write Accelerator on an existing disk, use [az vm update](#), you may use the following examples if you replace the diskName, VMName, and ResourceGroup with your own values:

```
az vm update -g group1 -n vm1 -write-accelerator 1=true
```

To attach a disk with Write Accelerator enabled use [az vm disk attach](#), you may use the following example if you substitute in your own values: `az vm disk attach -g group1 -vm-name vm1 -disk d1 --enable-write-accelerator`

To disable Write Accelerator, use [az vm update](#), setting the properties to false:

```
az vm update -g group1 -n vm1 -write-accelerator 0=false 1=false
```

Enabling Write Accelerator using Rest APIs

To deploy through Azure Rest API, you need to install the Azure armclient.

Install armclient

To run armclient, you need to install it through Chocolatey. You can install it through cmd.exe or powershell. Use elevated rights for these commands ("Run as Administrator").

Using cmd.exe, run the following command:

```
@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))"  
& SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
```

Using Power Shell, run the following command:

```
Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

Now you can install the armclient by using the following command in either cmd.exe or PowerShell

```
choco install armclient
```

Getting your current VM configuration

To change the attributes of your disk configuration, you first need to get the current configuration in a JSON file.

You can get the current configuration by executing the following command:

```
armclient GET /subscriptions/<<subscription-ID>>/resourceGroups/<<ResourceGroup>>/providers/Microsoft.Compute/virtualMachines/<<virtualmachinename>>?api-version=2017-12-01 <<filename.json>>
```

Replace the terms within '<< >>' with your data, including the file name the JSON file should have.

The output could look like:

```
{
  "properties": {
    "vmId": "2444c93e-f8bb-4a20-af2d-1658d9dbbbc",
    "hardwareProfile": {
      "vmSize": "Standard_M64s"
    },
    "storageProfile": {
      "imageReference": {
        "publisher": "SUSE",
        "offer": "SLES-SAP",
        "sku": "12-SP3",
        "version": "latest"
      },
      "osDisk": {
        "osType": "Linux",
        "name": "mylittlesap_OsDisk_1_754a1b8bb390468e9b4c429b81cc5f5a",
        "createOption": "FromImage",
        "caching": "ReadWrite",
        "managedDisk": {
          "storageAccountType": "Premium_LRS",
          "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/disks/mylittlesap_OsDisk_1_754a1b8bb390468e9b4c429b81cc5f5a"
        },
        "diskSizeGB": 30
      },
      "dataDisks": [
        {
          "lun": 0,
          "name": "data1",
          "createOption": "Attach",
          "caching": "None",
          "managedDisk": {
            "storageAccountType": "Premium_LRS",
            "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/disks/data1"
          },
          "diskSizeGB": 1023
        },
        {
          "lun": 1,
          "name": "log1",
          "createOption": "Attach",
          "caching": "None",
          "managedDisk": {
            "storageAccountType": "Premium_LRS",
            "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/disks/data2"
          }
        }
      ]
    }
  }
}
```

```

        },
        "diskSizeGB": 1023
    }
]
},
"osProfile": {
    "computerName": "mylittlesapVM",
    "adminUsername": "pl",
    "linuxConfiguration": {
        "disablePasswordAuthentication": false
    },
    "secrets": []
},
"networkProfile": {
    "networkInterfaces": [
        {
            "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Network/networkInterfaces/mylittlesap518"
        }
    ]
},
"diagnosticsProfile": {
    "bootDiagnostics": {
        "enabled": true,
        "storageUri": "https://mylittlesapdiag895.blob.core.windows.net/"
    }
},
"provisioningState": "Succeeded"
},
"type": "Microsoft.Compute/virtualMachines",
"location": "westeurope",
"id": "/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/virtualMachines/mylittlesapVM",
"name": "mylittlesapVM"

```

Next, update the JSON file and to enable Write Accelerator on the disk called 'log1'. This can be accomplished by adding this attribute into the JSON file after the cache entry of the disk.

```

{
    "lun": 1,
    "name": "log1",
    "createOption": "Attach",
    "caching": "None",
    "writeAcceleratorEnabled": true,
    "managedDisk": {
        "storageAccountType": "Premium_LRS",
        "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/disks/data2"
    },
    "diskSizeGB": 1023
}

```

Then update the existing deployment with this command:

```
armclient PUT /subscriptions/<<subscription-ID</>>/resourceGroups/<<ResourceGroup>>/providers/Microsoft.Compute/virtualMachines/<<virtualmachinename>>?api-version=2017-12-01 @<<filename.json>>
```

The output should look like the one below. You can see that Write Accelerator enabled for one disk.

```
{
    "properties": {
        "osProfile": {
            "computerName": "mylittlesapVM",
            "adminUsername": "pl",
            "linuxConfiguration": {
                "disablePasswordAuthentication": false
            },
            "secrets": []
        },
        "networkProfile": {
            "networkInterfaces": [
                {
                    "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Network/networkInterfaces/mylittlesap518"
                }
            ]
        },
        "diagnosticsProfile": {
            "bootDiagnostics": {
                "enabled": true,
                "storageUri": "https://mylittlesapdiag895.blob.core.windows.net/"
            }
        },
        "provisioningState": "Succeeded"
    },
    "type": "Microsoft.Compute/virtualMachines",
    "location": "westeurope",
    "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/virtualMachines/mylittlesapVM",
    "name": "mylittlesapVM"
}
```

```
    "vmSize": "Standard_M64s"
  },
  "storageProfile": {
    "imageReference": {
      "publisher": "SUSE",
      "offer": "SLES-SAP",
      "sku": "12-SP3",
      "version": "latest"
    },
    "osDisk": {
      "osType": "Linux",
      "name": "mylittlesap_OsDisk_1_754a1b8bb390468e9b4c429b81cc5f5a",
      "createOption": "FromImage",
      "caching": "ReadWrite",
      "managedDisk": {
        "storageAccountType": "Premium_LRS",
        "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/disks/mylittlesap_OsDisk_1_754a1b8bb390468e9b4c429b81cc5f5a"
      },
      "diskSizeGB": 30
    },
    "dataDisks": [
      {
        "lun": 0,
        "name": "data1",
        "createOption": "Attach",
        "caching": "None",
        "managedDisk": {
          "storageAccountType": "Premium_LRS",
          "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/disks/data1"
        },
        "diskSizeGB": 1023
      },
      {
        "lun": 1,
        "name": "log1",
        "createOption": "Attach",
        "caching": "None",
        "writeAcceleratorEnabled": true,
        "managedDisk": {
          "storageAccountType": "Premium_LRS",
          "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/disks/data2"
        },
        "diskSizeGB": 1023
      }
    ]
  },
  "osProfile": {
    "computerName": "mylittlesapVM",
    "adminUsername": "pl",
    "linuxConfiguration": {
      "disablePasswordAuthentication": false
    },
    "secrets": []
  },
  "networkProfile": {
    "networkInterfaces": [
      {
        "id": "/subscriptions/XXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Network/networkInterfaces/mylittlesap518"
      }
    ]
  }
}
```

```
        ],
    },
    "diagnosticsProfile": {
        "bootDiagnostics": {
            "enabled": true,
            "storageUri": "https://mylittlesapdiag895.blob.core.windows.net/"
        }
    },
    "provisioningState": "Succeeded"
},
"type": "Microsoft.Compute/virtualMachines",
"location": "westeurope",
"id":
"/subscriptions/XXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/vir
tualMachines/mylittlesapVM",
"name": "mylittlesapVM"
```

Once you've made this change, the drive should be supported by Write Accelerator.

Enable and deploy Azure ultra SSDs (preview)

5/12/2019 • 4 minutes to read • [Edit Online](#)

Azure ultra solid state drives (SSD) (preview) offer high throughput, high IOPS, and consistent low latency disk storage for Azure IaaS virtual machines (VMs). This new offering provides top of the line performance at the same availability levels as our existing disks offerings. One major benefit of ultra SSDs is the ability to dynamically change the performance of the SSD along with your workloads without the need to restart your VMs. Ultra SSDs are suited for data-intensive workloads such as SAP HANA, top tier databases, and transaction-heavy workloads.

Currently, ultra SSDs are in preview and you must [enroll](#) in the preview in order to access them.

Determine your availability zone

Once approved, you need to determine which availability zone you are in, in order to use ultra SSDs. Run either of the following commands to determine which zone in East US 2 to deploy your ultra disk to:

PowerShell: `Get-AzComputeResourceSku | where {$_.ResourceType -eq "disks" -and $_.Name -eq "UltraSSD_LRS" }`

CLI: `az vm list-skus --resource-type disks --query "[?name=='UltraSSD_LRS'].locationInfo"`

The response will be similar to the form below, where X is the zone to use for deploying in East US 2. X could be either 1, 2, or 3.

Preserve the **Zones** value, it represents your availability zone and you will need it in order to deploy an ultra SSD.

RESOURCETYPE	NAME	LOCATION	ZONES	RESTRICTION	CAPABILITY	VALUE
disks	UltraSSD_LRS	eastus2	X			

NOTE

If there was no response from the command, then your registration to the feature is either still pending, or not approved yet.

Now that you know which zone to deploy to, follow the deployment steps in this article to get your first VMs deployed with ultra SSD.

Deploy an ultra SSD using Azure Resource Manager

First, determine the VM size to deploy. As part of this preview, only DsV3 and EsV3 VM families are supported. Refer to the second table on this [blog](#) for additional details about these VM sizes.

If you would like to create a VM with multiple ultra SSDs, refer to the sample [Create a VM with multiple ultra SSD](#).

If you intend to use your own template, make sure that **apiVersion** for `Microsoft.Compute/virtualMachines` and `Microsoft.Compute/Disks` is set as `2018-06-01` (or later).

Set the disk sku to **UltraSSD_LRS**, then set the disk capacity, IOPS, availability zone, and throughput in MBps to create an ultra disk.

Once the VM is provisioned, you can partition and format the data disks and configure them for your workloads.

Deploy an ultra SSD using CLI

First, determine the VM size to deploy. As part of this preview, only DsV3 and EsV3 VM families are supported. Refer to the second table on this [blog](#) for additional details about these VM sizes.

To use ultra SSDs, you must create a VM that is capable of using ultra SSDs.

Replace or set the **\$vmname**, **\$rgname**, **\$diskname**, **\$location**, **\$password**, **\$user** variables with your own values. Set **\$zone** to the value of your availability zone that you got from the [start of this article](#). Then run the following CLI command to create an ultra enabled VM:

```
az vm create --subscription $subscription -n $vmname -g $rgname --image Win2016Datacenter --ultra-ssd-enabled --zone $zone --authentication-type password --admin-password $password --admin-username $user --attach-data-disks $diskname --size Standard_D4s_v3 --location $location
```

Create an ultra SSD using CLI

Now that you have a VM that is capable of using ultra SSDs, you can create and attach an ultra SSD to it.

```
location="eastus2"
subscription="xxx"
rgname="ultraRG"
diskname="ssd1"
vmname="ultravm1"
zone=123

#create an Ultra SSD disk
az disk create \
--subscription $subscription \
-n $diskname \
-g $rgname \
--size-gb 4 \
--location $location \
--zone $zone \
--sku UltraSSD_LRS \
--disk-iops-read-write 1000 \
--disk-mbps-read-write 50
```

Adjust the performance of an ultra SSD using CLI

Ultra SSDs offer a unique capability that allows you to adjust their performance, the following command depicts how to use this feature:

```
az disk update \
--subscription $subscription \
--resource-group $rgname \
--name $diskName \
--set diskIopsReadWrite=80000 \
--set diskMbpsReadWrite=800
```

Deploy an ultra SSD using PowerShell

First, determine the VM size to deploy. As part of this preview, only DsV3 and EsV3 VM families are supported. Refer to the second table on this [blog](#) for additional details about these VM sizes.

To use ultra SSDs, you must create a VM that is capable of using ultra SSDs. Replace or set the **\$resourcegroup** and **\$vmName** variables with your own values. Set **\$zone** to the value of your availability zone that you got from the [start of this article](#). Then run the following **New-AzVm** command to create an ultra enabled VM:

```
New-AzVm ` 
    -ResourceGroupName $resourcegroup ` 
    -Name $vmName ` 
    -Location "eastus2" ` 
    -Image "Win2016Datacenter" ` 
    -EnableUltraSSD ` 
    -size "Standard_D4s_v3" ` 
    -zone $zone
```

Create an ultra SSD using PowerShell

Now that you have a VM that is capable of using ultra SSDs, you can create and attach an ultra SSD to it:

```
$diskconfig = New-AzDiskConfig ` 
    -Location 'EastUS2' ` 
    -DiskSizeGB 8 ` 
    -DiskIOPSReadWrite 1000 ` 
    -DiskMBpsReadWrite 100 ` 
    -AccountType UltraSSD_LRS ` 
    -CreateOption Empty ` 
    -zone $zone; 

New-AzDisk ` 
    -ResourceGroupName $resourceGroup ` 
    -DiskName 'Disk02' ` 
    -Disk $diskconfig;
```

Adjust the performance of an ultra SSD using PowerShell

Ultra SSDs have a unique capability that allows you to adjust their performance, the following command is an example that adjusts the performance without having to detach the disk:

```
$diskupdateconfig = New-AzDiskUpdateConfig -DiskMBpsReadWrite 2000 
Update-AzDisk -ResourceGroupName $resourceGroup -DiskName $diskName -DiskUpdate $diskupdateconfig
```

Next steps

If you would like to try the new disk type [request access to the preview with this survey](#).

Benchmarking a disk

2/15/2019 • 8 minutes to read • [Edit Online](#)

Benchmarking is the process of simulating different workloads on your application and measuring the application performance for each workload. Using the steps described in the [designing for high performance article](#). By running benchmarking tools on the VMs hosting the application, you can determine the performance levels that your application can achieve with Premium Storage. In this article, we provide you examples of benchmarking a Standard DS14 VM provisioned with Azure Premium Storage disks.

We have used common benchmarking tools lometer and FIO, for Windows and Linux respectively. These tools spawn multiple threads simulating a production like workload, and measure the system performance. Using the tools you can also configure parameters like block size and queue depth, which you normally cannot change for an application. This gives you more flexibility to drive the maximum performance on a high scale VM provisioned with premium disks for different types of application workloads. To learn more about each benchmarking tool visit [lometer](#) and [FIO](#).

To follow the examples below, create a Standard DS14 VM and attach 11 Premium Storage disks to the VM. Of the 11 disks, configure 10 disks with host caching as "None" and stripe them into a volume called NoCacheWrites. Configure host caching as "ReadOnly" on the remaining disk and create a volume called CacheReads with this disk. Using this setup, you are able to see the maximum Read and Write performance from a Standard DS14 VM. For detailed steps about creating a DS14 VM with premium disks, go to [Designing for high performance](#).

Warming up the Cache

The disk with ReadOnly host caching are able to give higher IOPS than the disk limit. To get this maximum read performance from the host cache, first you must warm up the cache of this disk. This ensures that the Read IOs that the benchmarking tool will drive on CacheReads volume, actually hits the cache, and not the disk directly. The cache hits result in additional IOPS from the single cache enabled disk.

IMPORTANT

You must warm up the cache before running benchmarking, every time VM is rebooted.

Tools

lometer

[Download the lometer tool](#) on the VM.

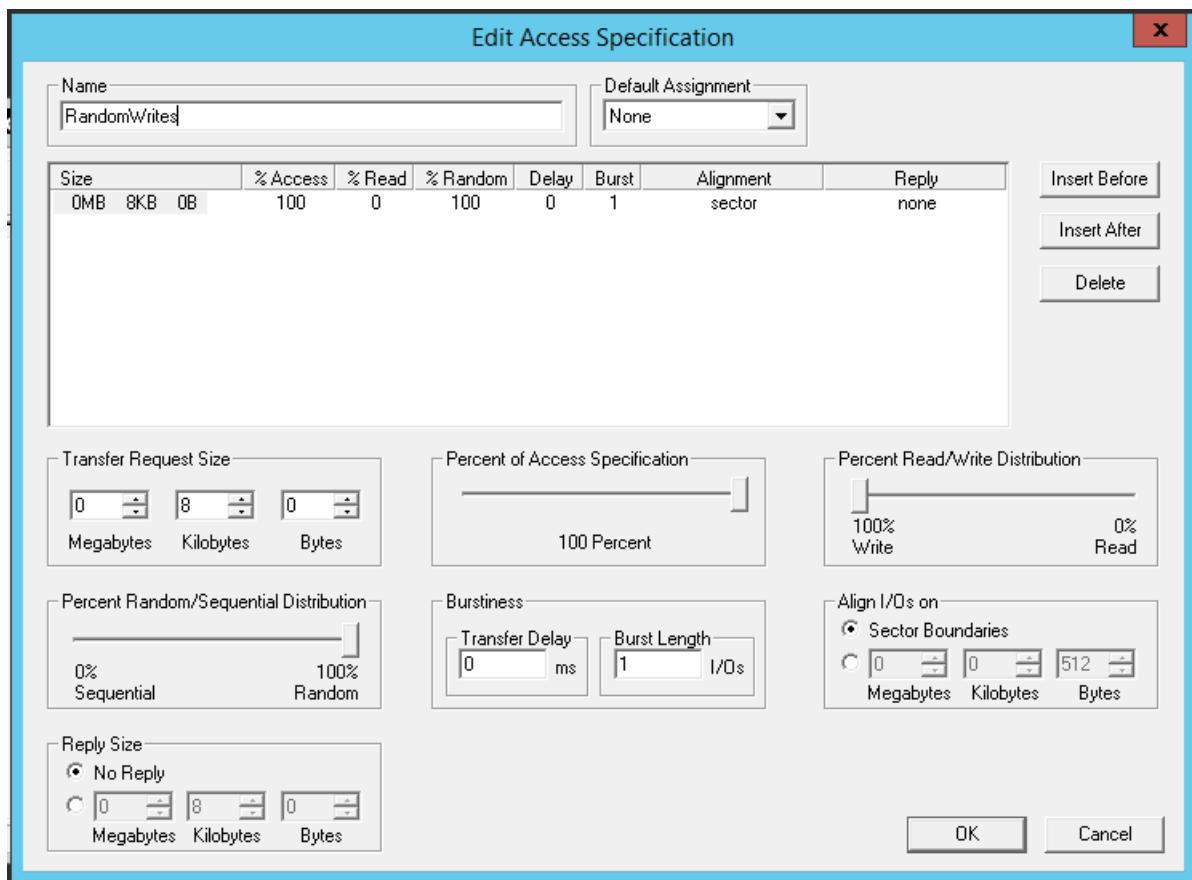
Test file

lometer uses a test file that is stored on the volume on which you run the benchmarking test. It drives Reads and Writes on this test file to measure the disk IOPS and Throughput. lometer creates this test file if you have not provided one. Create a 200 GB test file called iobw.tst on the CacheReads and NoCacheWrites volumes.

Access specifications

The specifications, request IO size, % read/write, % random/sequential are configured using the "Access Specifications" tab in lometer. Create an access specification for each of the scenarios described below. Create the access specifications and "Save" with an appropriate name like – RandomWrites_8K, RandomReads_8K. Select the corresponding specification when running the test scenario.

An example of access specifications for maximum Write IOPS scenario is shown below,



Maximum IOPS test specifications

To demonstrate maximum IOPs, use smaller request size. Use 8K request size and create specifications for Random Writes and Reads.

ACCESS SPECIFICATION	REQUEST SIZE	RANDOM %	READ %
RandomWrites_8K	8K	100	0
RandomReads_8K	8K	100	100

Maximum throughput test specifications

To demonstrate maximum Throughput, use larger request size. Use 64 K request size and create specifications for Random Writes and Reads.

ACCESS SPECIFICATION	REQUEST SIZE	RANDOM %	READ %
RandomWrites_64K	64 K	100	0
RandomReads_64K	64 K	100	100

Run the Iometer test

Perform the steps below to warm up cache

1. Create two access specifications with values shown below,

NAME	REQUEST SIZE	RANDOM %	READ %
RandomWrites_1MB	1 MB	100	0
RandomReads_1MB	1 MB	100	100

2. Run the lometer test for initializing cache disk with following parameters. Use three worker threads for the target volume and a queue depth of 128. Set the "Run time" duration of the test to 2 hrs on the "Test Setup" tab.

SCENARIO	TARGET VOLUME	NAME	DURATION
Initialize Cache Disk	CacheReads	RandomWrites_1MB	2 hrs

3. Run the lometer test for warming up cache disk with following parameters. Use three worker threads for the target volume and a queue depth of 128. Set the "Run time" duration of the test to 2 hrs on the "Test Setup" tab.

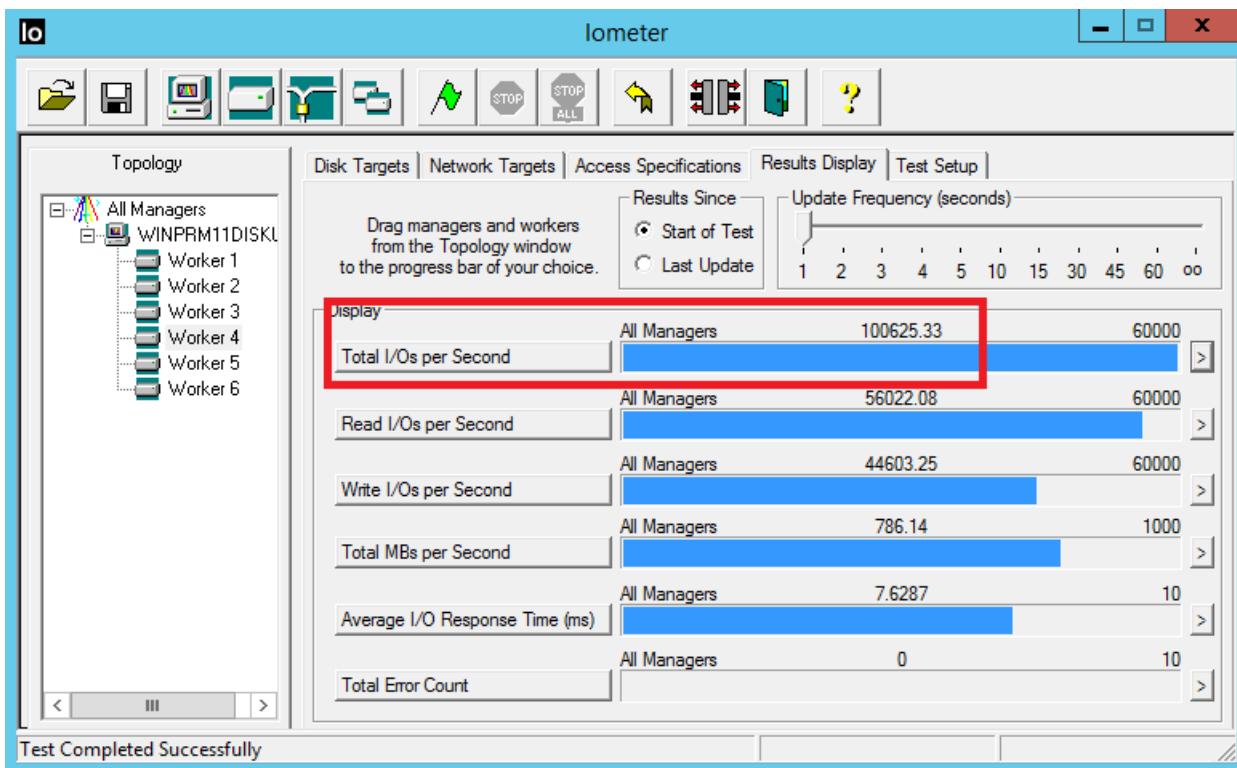
SCENARIO	TARGET VOLUME	NAME	DURATION
Warm up Cache Disk	CacheReads	RandomReads_1MB	2 hrs

After cache disk is warmed up, proceed with the test scenarios listed below. To run the lometer test, use at least three worker threads for **each** target volume. For each worker thread, select the target volume, set queue depth and select one of the saved test specifications, as shown in the table below, to run the corresponding test scenario. The table also shows expected results for IOPS and Throughput when running these tests. For all scenarios, a small IO size of 8 KB and a high queue depth of 128 is used.

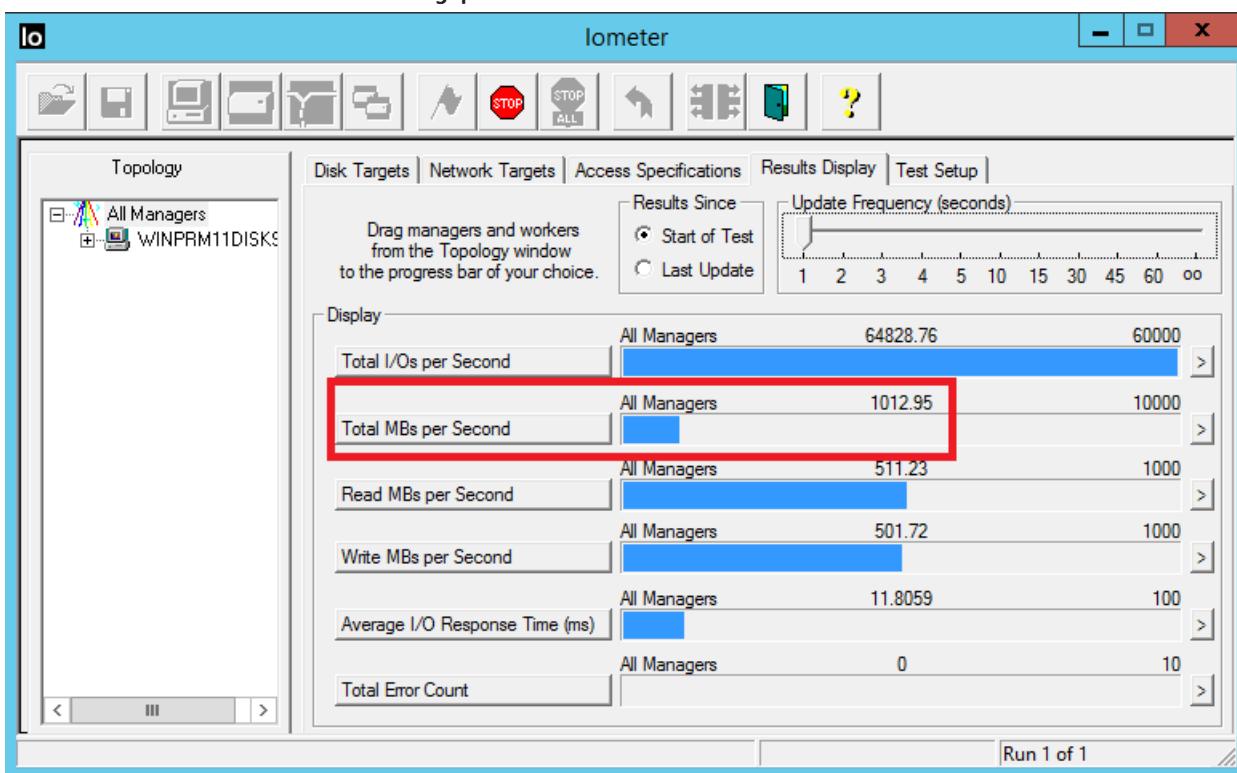
TEST SCENARIO	TARGET VOLUME	NAME	RESULT
Max. Read IOPS	CacheReads	RandomWrites_8K	50,000 IOPS
Max. Write IOPS	NoCacheWrites	RandomReads_8K	64,000 IOPS
Max. Combined IOPS	CacheReads	RandomWrites_8K	100,000 IOPS
NoCacheWrites	RandomReads_8K		
Max. Read MB/sec	CacheReads	RandomWrites_64K	524 MB/sec
Max. Write MB/sec	NoCacheWrites	RandomReads_64K	524 MB/sec
Combined MB/sec	CacheReads	RandomWrites_64K	1000 MB/sec
NoCacheWrites	RandomReads_64K		

Below are screenshots of the lometer test results for combined IOPS and Throughput scenarios.

Combined reads and writes maximum IOPS



Combined reads and writes maximum throughput



FIO

FIO is a popular tool to benchmark storage on the Linux VMs. It has the flexibility to select different IO sizes, sequential or random reads and writes. It spawns worker threads or processes to perform the specified I/O operations. You can specify the type of I/O operations each worker thread must perform using job files. We created one job file per scenario illustrated in the examples below. You can change the specifications in these job files to benchmark different workloads running on Premium Storage. In the examples, we are using a Standard DS 14 VM running **Ubuntu**. Use the same setup described in the beginning of the Benchmarking section and warm up the cache before running the benchmarking tests.

Before you begin, [download FIO](#) and install it on your virtual machine.

Run the following command for Ubuntu,

```
apt-get install fio
```

We use four worker threads for driving Write operations and four worker threads for driving Read operations on the disks. The Write workers are driving traffic on the "nocache" volume, which has 10 disks with cache set to "None". The Read workers are driving traffic on the "readcache" volume, which has one disk with cache set to "ReadOnly".

Maximum write IOPS

Create the job file with following specifications to get maximum Write IOPS. Name it "fiowrite.ini".

```
[global]
size=30g
direct=1
iodepth=256
ioengine=libaio
bs=8k

[writer1]
rw=randwrite
directory=/mnt/nocache
[writer2]
rw=randwrite
directory=/mnt/nocache
[writer3]
rw=randwrite
directory=/mnt/nocache
[writer4]
rw=randwrite
directory=/mnt/nocache
```

Note the follow key things that are in line with the design guidelines discussed in previous sections. These specifications are essential to drive maximum IOPS,

- A high queue depth of 256.
- A small block size of 8 KB.
- Multiple threads performing random writes.

Run the following command to kick off the FIO test for 30 seconds,

```
sudo fio --runtime 30 fiowrite.ini
```

While the test runs, you are able to see the number of write IOPS the VM and Premium disks are delivering. As shown in the sample below, the DS14 VM is delivering its maximum write IOPS limit of 50,000 IOPS.

```
demo@DS-VM-Linux-Demo:~$ sudo fio --runtime 30 fiowrite.ini
[sudo] password for demo:
writer1: (g=0): rw=randwrite, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
writer2: (g=0): rw=randwrite, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
writer3: (g=0): rw=randwrite, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
writer4: (g=0): rw=randwrite, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
fio-2.1.11
Starting 4 processes
Jobs: 4 (f=4): [w(4)] [63.3% done] [0KB/396.4MB/0KB /s] [0/50.8K/0 iops] [eta 00m:11s]
```

Maximum read IOPS

Create the job file with following specifications to get maximum Read IOPS. Name it "ioread.ini".

```

[global]
size=30g
direct=1
iodepth=256
ioengine=libaio
bs=8k

[reader1]
rw=randread
directory=/mnt/readcache
[reader2]
rw=randread
directory=/mnt/readcache
[reader3]
rw=randread
directory=/mnt/readcache
[reader4]
rw=randread
directory=/mnt/readcache

```

Note the follow key things that are in line with the design guidelines discussed in previous sections. These specifications are essential to drive maximum IOPS,

- A high queue depth of 256.
- A small block size of 8 KB.
- Multiple threads performing random writes.

Run the following command to kick off the FIO test for 30 seconds,

```
sudo fio --runtime 30 fioread.ini
```

While the test runs, you are able to see the number of read IOPS the VM and Premium disks are delivering. As shown in the sample below, the DS 14 VM is delivering more than 64,000 Read IOPS. This is a combination of the disk and the cache performance.

```

demo@DS-VM-Linux-Demo:~$ sudo fio --runtime 30 fioread.ini
[sudo] password for demo:
reader1: (g=0): rw=randread, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
reader2: (g=0): rw=randread, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
reader3: (g=0): rw=randread, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
reader4: (g=0): rw=randread, bs=8K-8K/8K-8K/8K-8K, ioengine=libaio, iodepth=256
fio-2.1.11
Starting 4 processes
Jobs: 4 (f=4): [r(4)] [70.0% done] [514.8MB/0KB/0KB /s] [65.9K/0/0 iops] [eta 00m:09s]
```

Maximum read and write IOPS

Create the job file with following specifications to get maximum combined Read and Write IOPS. Name it "fioreadwrite.ini".

```

[global]
size=30g
direct=1
iodepth=128
ioengine=libaio
bs=4k

[reader1]
rw=randread
directory=/mnt/readcache
[reader2]
rw=randread
directory=/mnt/readcache
[reader3]
rw=randread
directory=/mnt/readcache
[reader4]
rw=randread
directory=/mnt/readcache

[writer1]
rw=randwrite
directory=/mnt/nocache
rate_iops=12500
[writer2]
rw=randwrite
directory=/mnt/nocache
rate_iops=12500
[writer3]
rw=randwrite
directory=/mnt/nocache
rate_iops=12500
[writer4]
rw=randwrite
directory=/mnt/nocache
rate_iops=12500

```

Note the follow key things that are in line with the design guidelines discussed in previous sections. These specifications are essential to drive maximum IOPS,

- A high queue depth of 128.
- A small block size of 4 KB.
- Multiple threads performing random reads and writes.

Run the following command to kick off the FIO test for 30 seconds,

```
sudo fio --runtime 30 fioreadwrite.ini
```

While the test runs, you are able to see the number of combined read and write IOPS the VM and Premium disks are delivering. As shown in the sample below, the DS14 VM is delivering more than 100,000 combined Read and Write IOPS. This is a combination of the disk and the cache performance.

```

demo@DS-VM-Linux-Demo:~$ sudo fio --runtime 30 fioreadwrite.ini
reader1: (g=0): rw=randread, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
reader2: (g=0): rw=randread, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
reader3: (g=0): rw=randread, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
reader4: (g=0): rw=randread, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
writer1: (g=0): rw=randwrite, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
writer2: (g=0): rw=randwrite, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
writer3: (g=0): rw=randwrite, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
writer4: (g=0): rw=randwrite, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
fio-2.1.11
Starting 8 processes
Jobs: 8 (f=8), CR=50000/0 IOPS: [r(4),w(4)] [22.6% done] [251.2MB/183.3MB/0KB /s] [64.3K/46.1K/0 iops] [eta 00m:24s]
```

Maximum combined throughput

To get the maximum combined Read and Write Throughput, use a larger block size and large queue depth with

multiple threads performing reads and writes. You can use a block size of 64 KB and queue depth of 128.

Next steps

Proceed through our design for high performance article. In it, you create a checklist similar to your existing application for the prototype. Using Benchmarking tools you can simulate the workloads and measure performance on the prototype application. By doing so, you can determine which disk offering can match or surpass your application performance requirements. Then you can implement the same guidelines for your production application.

See the article on [designing for high performance](#) begin.

Optimize your Linux VM on Azure

2/15/2019 • 6 minutes to read • [Edit Online](#)

Creating a Linux virtual machine (VM) is easy to do from the command line or from the portal. This tutorial shows you how to ensure you have set it up to optimize its performance on the Microsoft Azure platform. This topic uses an Ubuntu Server VM, but you can also create Linux virtual machine using [your own images as templates](#).

Prerequisites

This topic assumes you already have a working Azure subscription ([free trial signup](#)) and have already provisioned a VM into your Azure subscription. Make sure that you have the latest [Azure CLI](#) installed and logged in to your Azure subscription with [az login](#) before you [create a VM](#).

Azure OS Disk

Once you create a Linux VM in Azure, it has two disks associated with it. **/dev/sda** is your OS disk, **/dev/sdb** is your temporary disk. Do not use the main OS disk (**/dev/sda**) for anything except the operating system as it is optimized for fast VM boot time and does not provide good performance for your workloads. You want to attach one or more disks to your VM to get persistent and optimized storage for your data.

Adding Disks for Size and Performance targets

Based on the VM size, you can attach up to 16 additional disks on an A-Series, 32 disks on a D-Series and 64 disks on a G-Series machine - each up to 1 TB in size. You add extra disks as needed per your space and IOps requirements. Each disk has a performance target of 500 IOps for Standard Storage and up to 5000 IOps per disk for Premium Storage.

To achieve the highest IOps on Premium Storage disks where their cache settings have been set to either **ReadOnly** or **None**, you must disable **barriers** while mounting the file system in Linux. You do not need barriers because the writes to Premium Storage backed disks are durable for these cache settings.

- If you use **reiserFS**, disable barriers using the mount option `barrier=none` (For enabling barriers, use `barrier=flush`)
- If you use **ext3/ext4**, disable barriers using the mount option `barrier=0` (For enabling barriers, use `barrier=1`)
- If you use **XFS**, disable barriers using the mount option `nobarrier` (For enabling barriers, use the option `barrier`)

Unmanaged storage account considerations

The default action when you create a VM with the Azure CLI is to use Azure Managed Disks. These disks are handled by the Azure platform and do not require any preparation or location to store them. Unmanaged disks require a storage account and have some additional performance considerations. For more information about managed disks, see [Azure Managed Disks overview](#). The following section outlines performance considerations only when you use unmanaged disks. Again, the default and recommended storage solution is to use managed disks.

If you create a VM with unmanaged disks, make sure that you attach disks from storage accounts residing in the same region as your VM to ensure close proximity and minimize network latency. Each Standard storage account has a maximum of 20k IOps and a 500 TB size capacity. This limit works out to approximately 40 heavily used

disks including both the OS disk and any data disks you create. For Premium Storage accounts, there is no Maximum IOps limit but there is a 32 TB size limit.

When dealing with high IOps workloads and you have chosen Standard Storage for your disks, you might need to split the disks across multiple storage accounts to make sure you have not hit the 20,000 IOps limit for Standard Storage accounts. Your VM can contain a mix of disks from across different storage accounts and storage account types to achieve your optimal configuration.

Your VM Temporary drive

By default when you create a VM, Azure provides you with an OS disk (**/dev/sda**) and a temporary disk (**/dev/sdb**). All additional disks you add show up as **/dev/sdc**, **/dev/sdd**, **/dev/sde** and so on. All data on your temporary disk (**/dev/sdb**) is not durable, and can be lost if specific events like VM Resizing, redeployment, or maintenance forces a restart of your VM. The size and type of your temporary disk is related to the VM size you chose at deployment time. All of the premium size VMs (DS, G, and DS_V2 series) the temporary drive are backed by a local SSD for additional performance of up to 48k IOps.

Linux Swap File

If your Azure VM is from an Ubuntu or CoreOS image, then you can use CustomData to send a cloud-config to cloud-init. If you [uploaded a custom Linux image](#) that uses cloud-init, you also configure swap partitions using cloud-init.

On Ubuntu Cloud Images, you must use cloud-init to configure the swap partition. For more information, see [AzureSwapPartitions](#).

For images without cloud-init support, VM images deployed from the Azure Marketplace have a VM Linux Agent integrated with the OS. This agent allows the VM to interact with various Azure services. Assuming you have deployed a standard image from the Azure Marketplace, you would need to do the following to correctly configure your Linux swap file settings:

Locate and modify two entries in the **/etc/waagent.conf** file. They control the existence of a dedicated swap file and size of the swap file. The parameters you are looking to modify are **ResourceDisk.EnableSwap=N** and **ResourceDisk.SwapSizeMB=0**

Change the parameters to the following settings:

- **ResourceDisk.EnableSwap=Y**
- **ResourceDisk.SwapSizeMB={size in MB to meet your needs}**

Once you have made the change, you need to restart the waagent or restart your Linux VM to reflect those changes. You know the changes have been implemented and a swap file has been created when you use the **free** command to view free space. The following example has a 512MB swap file created as a result of modifying the **waagent.conf** file:

```
azuseruser@myVM:~$ free
              total        used        free      shared  buffers   cached
Mem:       3525156     804168    2720988        408     8428   633192
 -/+ buffers/cache:    162548    3362608
Swap:      524284          0    524284
```

I/O scheduling algorithm for Premium Storage

With the 2.6.18 Linux kernel, the default I/O scheduling algorithm was changed from Deadline to CFQ (Completely fair queuing algorithm). For random access I/O patterns, there is negligible difference in performance

differences between CFQ and Deadline. For SSD-based disks where the disk I/O pattern is predominantly sequential, switching back to the NOOP or Deadline algorithm can achieve better I/O performance.

View the current I/O scheduler

Use the following command:

```
cat /sys/block/sda/queue/scheduler
```

You see following output, which indicates the current scheduler.

```
noop [deadline] cfq
```

Change the current device (`/dev/sda`) of I/O scheduling algorithm

Use the following commands:

```
azureuser@myVM:~$ sudo su -
root@myVM:~# echo "noop" >/sys/block/sda/queue/scheduler
root@myVM:~# sed -i 's/GRUB_CMDLINE_LINUX=""/GRUB_CMDLINE_LINUX_DEFAULT="quiet splash elevator=noop"/g' /etc/default/grub
root@myVM:~# update-grub
```

NOTE

Applying this setting for `/dev/sda` alone is not useful. Set on all data disks where sequential I/O dominates the I/O pattern.

You should see the following output, indicating that **grub.cfg** has been rebuilt successfully and that the default scheduler has been updated to NOOP.

```
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.13.0-34-generic
Found initrd image: /boot/initrd.img-3.13.0-34-generic
Found linux image: /boot/vmlinuz-3.13.0-32-generic
Found initrd image: /boot/initrd.img-3.13.0-32-generic
Found memtest86+ image: /memtest86+.elf
Found memtest86+ image: /memtest86+.bin
done
```

For the Red Hat distribution family, you only need the following command:

```
echo 'echo noop >/sys/block/sda/queue/scheduler' >> /etc/rc.local
```

Using Software RAID to achieve higher I/Ops

If your workloads require more IOps than a single disk can provide, you need to use a software RAID configuration of multiple disks. Because Azure already performs disk resiliency at the local fabric layer, you achieve the highest level of performance from a RAID-0 striping configuration. Provision and create disks in the Azure environment and attach them to your Linux VM before partitioning, formatting and mounting the drives. More details on configuring a software RAID setup on your Linux VM in azure can be found in the [Configuring Software RAID on Linux](#) document.

Next Steps

Remember, as with all optimization discussions, you need to perform tests before and after each change to measure the impact the change has. Optimization is a step by step process that has different results across different machines in your environment. What works for one configuration may not work for others.

Some useful links to additional resources:

- [Azure Linux Agent User Guide](#)
- [Optimizing MySQL Performance on Azure Linux VMs](#)
- [Configure Software RAID on Linux](#)

Configure Software RAID on Linux

3/15/2019 • 5 minutes to read • [Edit Online](#)

It's a common scenario to use software RAID on Linux virtual machines in Azure to present multiple attached data disks as a single RAID device. Typically this can be used to improve performance and allow for improved throughput compared to using just a single disk.

Attaching data disks

Two or more empty data disks are needed to configure a RAID device. The primary reason for creating a RAID device is to improve performance of your disk IO. Based on your IO needs, you can choose to attach disks that are stored in our Standard Storage, with up to 500 IO/ps per disk or our Premium storage with up to 5000 IO/ps per disk. This article does not go into detail on how to provision and attach data disks to a Linux virtual machine. See the Microsoft Azure article [attach a disk](#) for detailed instructions on how to attach an empty data disk to a Linux virtual machine on Azure.

Install the mdadm utility

- **Ubuntu**

```
sudo apt-get update  
sudo apt-get install mdadm
```

- **CentOS & Oracle Linux**

```
sudo yum install mdadm
```

- **SLES and openSUSE**

```
zypper install mdadm
```

Create the disk partitions

In this example, we create a single disk partition on /dev/sdc. The new disk partition will be called /dev/sdc1.

1. Start `fdisk` to begin creating partitions

```
sudo fdisk /dev/sdc  
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel  
Building a new DOS disklabel with disk identifier 0xa34cb70c.  
Changes will remain in memory only, until you decide to write them.  
After that, of course, the previous content won't be recoverable.  
  
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to  
switch off the mode (command 'c') and change display units to  
sectors (command 'u').
```

2. Press 'n' at the prompt to create a new partition:

```
Command (m for help): n
```

3. Next, press 'p' to create a primary partition:

```
Command action
e   extended
p   primary partition (1-4)
```

4. Press '1' to select partition number 1:

```
Partition number (1-4): 1
```

5. Select the starting point of the new partition, or press <enter> to accept the default to place the partition at the beginning of the free space on the drive:

```
First cylinder (1-1305, default 1):
Using default value 1
```

6. Select the size of the partition, for example type '+10G' to create a 10 gigabyte partition. Or, press <enter> to create a single partition that spans the entire drive:

```
Last cylinder, +cylinders or +size{K,M,G} (1-1305, default 1305):
Using default value 1305
```

7. Next, change the ID and type of the partition from the default ID '83' (Linux) to ID 'fd' (Linux raid auto):

```
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
```

8. Finally, write the partition table to the drive and exit fdisk:

```
Command (m for help): w
The partition table has been altered!
```

Create the RAID array

1. The following example will "stripe" (RAID level 0) three partitions located on three separate data disks (sdc1, sdd1, sde1). After running this command a new RAID device called /dev/md127 is created. Also note that if these data disks were previously part of another defunct RAID array it may be necessary to add the --force parameter to the mdadm command:

```
sudo mdadm --create /dev/md127 --level 0 --raid-devices 3 \
/dev/sdc1 /dev/sdd1 /dev/sde1
```

2. Create the file system on the new RAID device

- a. CentOS, Oracle Linux, SLES 12, openSUSE, and Ubuntu

```
sudo mkfs -t ext4 /dev/md127
```

b. SLES 11

```
sudo mkfs -t ext3 /dev/md127
```

c. SLES 11 - enable boot.md and create mdadm.conf

```
sudo -i chkconfig --add boot.md  
sudo echo 'DEVICE /dev/sd*[0-9]' >> /etc/mdadm.conf
```

NOTE

A reboot may be required after making these changes on SUSE systems. This step is *not* required on SLES 12.

Add the new file system to /etc/fstab

IMPORTANT

Improperly editing the /etc/fstab file could result in an unbootable system. If unsure, refer to the distribution's documentation for information on how to properly edit this file. It is also recommended that a backup of the /etc/fstab file is created before editing.

1. Create the desired mount point for your new file system, for example:

```
sudo mkdir /data
```

2. When editing /etc/fstab, the **UUID** should be used to reference the file system rather than the device name. Use the `blkid` utility to determine the UUID for the new file system:

```
sudo /sbin/blkid  
.....  
/dev/md127: UUID="aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee" TYPE="ext4"
```

3. Open /etc/fstab in a text editor and add an entry for the new file system, for example:

```
UUID=aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee /data ext4 defaults 0 2
```

Or on **SLES 11**:

```
/dev/disk/by-uuid/aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee /data ext3 defaults 0 2
```

Then, save and close /etc/fstab.

4. Test that the /etc/fstab entry is correct:

```
sudo mount -a
```

If this command results in an error message, please check the syntax in the /etc/fstab file.

Next run the `mount` command to ensure the file system is mounted:

```
mount  
.....  
/dev/md127 on /data type ext4 (rw)
```

5. (Optional) Failsafe Boot Parameters

fstab configuration

Many distributions include either the `nobootwait` or `nofail` mount parameters that may be added to the /etc/fstab file. These parameters allow for failures when mounting a particular file system and allow the Linux system to continue to boot even if it is unable to properly mount the RAID file system. Refer to your distribution's documentation for more information on these parameters.

Example (Ubuntu):

```
UUID=aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee /data ext4 defaults,nobootwait 0 2
```

Linux boot parameters

In addition to the above parameters, the kernel parameter "`bootdegraded=true`" can allow the system to boot even if the RAID is perceived as damaged or degraded, for example if a data drive is inadvertently removed from the virtual machine. By default this could also result in a non-bootable system.

Please refer to your distribution's documentation on how to properly edit kernel parameters. For example, in many distributions (CentOS, Oracle Linux, SLES 11) these parameters may be added manually to the "`/boot/grub/menu.lst`" file. On Ubuntu this parameter can be added to the `GRUB_CMDLINE_LINUX_DEFAULT` variable on "/etc/default/grub".

TRIM/UNMAP support

Some Linux kernels support TRIM/UNMAP operations to discard unused blocks on the disk. These operations are primarily useful in standard storage to inform Azure that deleted pages are no longer valid and can be discarded. Discarding pages can save cost if you create large files and then delete them.

NOTE

RAID may not issue discard commands if the chunk size for the array is set to less than the default (512KB). This is because the unmap granularity on the Host is also 512KB. If you modified the array's chunk size via mdadm's `--chunk=` parameter, then TRIM/unmap requests may be ignored by the kernel.

There are two ways to enable TRIM support in your Linux VM. As usual, consult your distribution for the recommended approach:

- Use the `discard` mount option in `/etc/fstab`, for example:

```
UUID=aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee /data ext4 defaults,discard 0 2
```

- In some cases the `discard` option may have performance implications. Alternatively, you can run the `fstrim` command manually from the command line, or add it to your crontab to run regularly:

Ubuntu

```
# sudo apt-get install util-linux  
# sudo fstrim /data
```

RHEL/CentOS

```
# sudo yum install util-linux  
# sudo fstrim /data
```

Configure LVM on a Linux VM in Azure

1/30/2019 • 4 minutes to read • [Edit Online](#)

This document will discuss how to configure Logical Volume Manager (LVM) in your Azure virtual machine. LVM may be used on the OS disk or data disks in Azure VMs, however, by default most cloud images will not have LVM configured on the OS disk. The steps below will focus on configuring LVM for your data disks.

Linear vs. striped logical volumes

LVM can be used to combine a number of physical disks into a single storage volume. By default LVM will usually create linear logical volumes, which means that the physical storage is concatenated together. In this case read/write operations will typically only be sent to a single disk. In contrast, we can also create striped logical volumes where reads and writes are distributed to multiple disks contained in the volume group (similar to RAID0). For performance reasons, it is likely you will want to stripe your logical volumes so that reads and writes utilize all your attached data disks.

This document will describe how to combine several data disks into a single volume group, and then create a striped logical volume. The steps below are generalized to work with most distributions. In most cases the utilities and workflows for managing LVM on Azure are not fundamentally different than other environments. As usual, also consult your Linux vendor for documentation and best practices for using LVM with your particular distribution.

Attaching data disks

One will usually want to start with two or more empty data disks when using LVM. Based on your IO needs, you can choose to attach disks that are stored in our Standard Storage, with up to 500 IO/ps per disk or our Premium storage with up to 5000 IO/ps per disk. This article will not go into detail on how to provision and attach data disks to a Linux virtual machine. See the Microsoft Azure article [attach a disk](#) for detailed instructions on how to attach an empty data disk to a Linux virtual machine on Azure.

Install the LVM utilities

- **Ubuntu**

```
sudo apt-get update  
sudo apt-get install lvm2
```

- **RHEL, CentOS & Oracle Linux**

```
sudo yum install lvm2
```

- **SLES 12 and openSUSE**

```
sudo zypper install lvm2
```

- **SLES 11**

```
sudo zypper install lvm
```

On SLES11, you must also edit `/etc/sysconfig/lvm` and set `LVM_ACTIVATED_ON_DISCOVERED` to "enable":

```
LVM_ACTIVATED_ON_DISCOVERED="enable"
```

Configure LVM

In this guide we will assume you have attached three data disks, which we'll refer to as `/dev/sdc`, `/dev/sdd` and `/dev/sde`. These paths may not match the disk path names in your VM. You can run '`sudo fdisk -l`' or similar command to list your available disks.

1. Prepare the physical volumes:

```
sudo pvcreate /dev/sd[cde]
Physical volume "/dev/sdc" successfully created
Physical volume "/dev/sdd" successfully created
Physical volume "/dev/sde" successfully created
```

2. Create a volume group. In this example we are calling the volume group `data-vg01`:

```
sudo vgcreate data-vg01 /dev/sd[cde]
Volume group "data-vg01" successfully created
```

3. Create the logical volume(s). The command below we will create a single logical volume called `data-lv01` to span the entire volume group, but note that it is also feasible to create multiple logical volumes in the volume group.

```
sudo lvcreate --extents 100%FREE --stripes 3 --name data-lv01 data-vg01
Logical volume "data-lv01" created.
```

4. Format the logical volume

```
sudo mkfs -t ext4 /dev/data-vg01/data-lv01
```

NOTE

With SLES11 use `-t ext3` instead of ext4. SLES11 only supports read-only access to ext4 filesystems.

Add the new file system to `/etc/fstab`

IMPORTANT

Improperly editing the `/etc/fstab` file could result in an unbootable system. If unsure, refer to the distribution's documentation for information on how to properly edit this file. It is also recommended that a backup of the `/etc/fstab` file is created before editing.

1. Create the desired mount point for your new file system, for example:

```
sudo mkdir /data
```

2. Locate the logical volume path

```
lvdisplay
--- Logical volume ---
LV Path          /dev/data-vg01/data-lv01
....
```

3. Open `/etc/fstab` in a text editor and add an entry for the new file system, for example:

```
/dev/data-vg01/data-lv01  /data  ext4  defaults  0  2
```

Then, save and close `/etc/fstab`.

4. Test that the `/etc/fstab` entry is correct:

```
sudo mount -a
```

If this command results in an error message check the syntax in the `/etc/fstab` file.

Next run the `mount` command to ensure the file system is mounted:

```
mount
.....
/dev/mapper/data--vg01-data--lv01 on /data type ext4 (rw)
```

5. (Optional) Failsafe boot parameters in `/etc/fstab`

Many distributions include either the `nobootwait` or `nofail` mount parameters that may be added to the `/etc/fstab` file. These parameters allow for failures when mounting a particular file system and allow the Linux system to continue to boot even if it is unable to properly mount the RAID file system. Refer to your distribution's documentation for more information on these parameters.

Example (Ubuntu):

```
/dev/data-vg01/data-lv01  /data  ext4  defaults,nobootwait  0  2
```

TRIM/UNMAP support

Some Linux kernels support TRIM/UNMAP operations to discard unused blocks on the disk. These operations are primarily useful in standard storage to inform Azure that deleted pages are no longer valid and can be discarded. Discarding pages can save cost if you create large files and then delete them.

There are two ways to enable TRIM support in your Linux VM. As usual, consult your distribution for the recommended approach:

- Use the `discard` mount option in `/etc/fstab`, for example:

```
/dev/data-vg01/data-lv01  /data  ext4  defaults,discard  0  2
```

- In some cases the `discard` option may have performance implications. Alternatively, you can run the

`fstrim` command manually from the command line, or add it to your crontab to run regularly:

Ubuntu

```
# sudo apt-get install util-linux  
# sudo fstrim /datadrive
```

RHEL/CentOS

```
# sudo yum install util-linux  
# sudo fstrim /datadrive
```

Find and delete unattached Azure managed and unmanaged disks

4/24/2019 • 2 minutes to read • [Edit Online](#)

When you delete a virtual machine (VM) in Azure, by default, any disks that are attached to the VM aren't deleted. This feature helps to prevent data loss due to the unintentional deletion of VMs. After a VM is deleted, you will continue to pay for unattached disks. This article shows you how to find and delete any unattached disks and reduce unnecessary costs.

Managed disks: Find and delete unattached disks

The following script looks for unattached [managed disks](#) by examining the value of the **ManagedBy** property. When a managed disk is attached to a VM, the **ManagedBy** property contains the resource ID of the VM. When a managed disk is unattached, the **ManagedBy** property is null. The script examines all the managed disks in an Azure subscription. When the script locates a managed disk with the **ManagedBy** property set to null, the script determines that the disk is unattached.

IMPORTANT

First, run the script by setting the **deleteUnattachedDisks** variable to 0. This action lets you find and view all the unattached managed disks.

After you review all the unattached disks, run the script again and set the **deleteUnattachedDisks** variable to 1. This action lets you delete all the unattached managed disks.

```
# Set deleteUnattachedDisks=1 if you want to delete unattached Managed Disks
# Set deleteUnattachedDisks=0 if you want to see the Id of the unattached Managed Disks
deleteUnattachedDisks=0

unattachedDiskIds=$(az disk list --query '[?managedBy==`null`].[id]' -o tsv)
for id in ${unattachedDiskIds[@]}
do
    if (( $deleteUnattachedDisks == 1 ))
    then
        echo "Deleting unattached Managed Disk with Id: \"$id"
        az disk delete --ids $id --yes
        echo "Deleted unattached Managed Disk with Id: \"$id"
    else
        echo $id
    fi
done
```

Unmanaged disks: Find and delete unattached disks

Unmanaged disks are VHD files that are stored as [page blobs](#) in [Azure storage accounts](#). The following script looks for unattached unmanaged disks (page blobs) by examining the value of the **LeaseStatus** property. When an unmanaged disk is attached to a VM, the **LeaseStatus** property is set to **Locked**. When an unmanaged disk is unattached, the **LeaseStatus** property is set to **Unlocked**. The script examines all the unmanaged disks in all the Azure storage accounts in an Azure subscription. When the script locates an unmanaged disk with a **LeaseStatus**

property set to **Unlocked**, the script determines that the disk is unattached.

IMPORTANT

First, run the script by setting the **deleteUnattachedVHDs** variable to 0. This action lets you find and view all the unattached unmanaged VHDs.

After you review all the unattached disks, run the script again and set the **deleteUnattachedVHDs** variable to 1. This action lets you delete all the unattached unmanaged VHDs.

```
# Set deleteUnattachedVHDs=1 if you want to delete unattached VHDs
# Set deleteUnattachedVHDs=0 if you want to see the details of the unattached VHDs
deleteUnattachedVHDs=0

storageAccountIds=$(az storage account list --query [].[id] -o tsv)

for id in ${storageAccountIds[@]}
do
    connectionString=$(az storage account show-connection-string --ids $id --query connectionString -o tsv)
    containers=$(az storage container list --connection-string $connectionString --query [].[name] -o tsv)

    for container in ${containers[@]}
    do

        blobs=$(az storage blob list -c $container --connection-string $connectionString --query "[?properties.blobType=='PageBlob' && ends_with(name,'.vhd')].[name]" -o tsv)

        for blob in ${blobs[@]}
        do
            leaseStatus=$(az storage blob show -n $blob -c $container --connection-string $connectionString --query "properties.lease.status" -o tsv)

            if [ "$leaseStatus" == "unlocked" ]
            then

                if (( $deleteUnattachedVHDs == 1 ))
                then

                    echo "Deleting VHD: \"$blob\" in container: \"$container\" in storage account: \"$id"
                    az storage blob delete --delete-snapshots include -n $blob -c $container --connection-string $connectionString

                    echo "Deleted VHD: \"$blob\" in container: \"$container\" in storage account: \"$id"
                else
                    echo "StorageAccountId: \"$id\" container: \"$container\" VHD: \"$blob"
                fi

            fi
        done
    done
done
```

Next steps

[Delete storage account](#)

Mount Azure File storage on Linux VMs using SMB

3/27/2019 • 3 minutes to read • [Edit Online](#)

This article shows you how to use the Azure File storage service on a Linux VM using an SMB mount with the Azure CLI. Azure File storage offers file shares in the cloud using the standard SMB protocol.

File storage offers file shares in the cloud that use the standard SMB protocol. You can mount a file share from any OS that supports SMB 3.0. When you use an SMB mount on Linux, you get easy backups to a robust, permanent archiving storage location that is supported by an SLA.

Moving files from a VM to an SMB mount that's hosted on File storage is a great way to debug logs. The same SMB share can be mounted locally to your Mac, Linux, or Windows workstation. SMB isn't the best solution for streaming Linux or application logs in real time, because the SMB protocol is not built to handle such heavy logging duties. A dedicated, unified logging layer tool such as Fluentd would be a better choice than SMB for collecting Linux and application logging output.

This guide requires that you're running the Azure CLI version 2.0.4 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install the Azure CLI](#).

Create a resource group

Create a resource group named *myResourceGroup* in the *East US* location.

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

Create a storage account

Create a new storage account, within the resource group that you created, using [az storage account create](#). This example creates a storage account named *mySTORAGEACCT<random number>* and puts the name of that storage account in the variable **STORAGEACCT**. Storage account names must be unique, using `$RANDOM` appends a number to the end to make it unique.

```
STORAGEACCT=$(az storage account create \
    --resource-group "myResourceGroup" \
    --name "mystorageacct$RANDOM" \
    --location eastus \
    --sku Standard_LRS \
    --query "name" | tr -d '')
```

Get the storage key

When you create a storage account, the account keys are created in pairs so that they can be rotated without any service interruption. When you switch to the second key in the pair, you create a new key pair. New storage account keys are always created in pairs, so you always have at least one unused storage account key ready to switch to.

View the storage account keys using [az storage account keys list](#). This example stores the value of key 1 in the **STORAGEKEY** variable.

```
STORAGEKEY=$(az storage account keys list \
--resource-group "myResourceGroup" \
--account-name $STORAGEACCT \
--query "[0].value" | tr -d '')
```

Create a file share

Create the File storage share using [az storage share create](#).

Share names need to be all lower case letters, numbers, and single hyphens but can't start with a hyphen. For complete details about naming file shares and files, see [Naming and Referencing Shares, Directories, Files, and Metadata](#).

This example creates a share named *myshare* with a 10-GiB quota.

```
az storage share create --name myshare \
--quota 10 \
--account-name $STORAGEACCT \
--account-key $STORAGEKEY
```

Create a mount point

To mount the Azure file share on your Linux computer, you need to make sure you have the **cifs-utils** package installed. For installation instructions, see [Install the cifs-utils package for your Linux distribution](#).

Azure Files uses SMB protocol, which communicates over TCP port 445. If you're having trouble mounting your Azure file share, make sure your firewall is not blocking TCP port 445.

```
mkdir -p /mnt/MyAzureFileShare
```

Mount the share

Mount the Azure file share to the local directory.

```
sudo mount -t cifs //${STORAGEACCT}.file.core.windows.net/myshare /mnt/MyAzureFileShare -o
vers=3.0,username=${STORAGEACCT},password=${STORAGEKEY},dir_mode=0777,file_mode=0777,serverino
```

The above command uses the [mount](#) command to mount the Azure file share and options specific to [cifs](#). Specifically, the `file_mode` and `dir_mode` options set files and directories to permission `0777`. The `0777` permission gives read, write, and execute permissions to all users. You can change these permissions by replacing the values with other [chmod permissions](#). You can also use other [cifs](#) options such as `gid` or `uid`.

Persist the mount

When you reboot the Linux VM, the mounted SMB share is unmounted during shutdown. To remount the SMB share on boot, add a line to the Linux `/etc/fstab`. Linux uses the `fstab` file to list the file systems that it needs to mount during the boot process. Adding the SMB share ensures that the File storage share is a permanently mounted file system for the Linux VM. Adding the File storage SMB share to a new VM is possible when you use `cloud-init`.

```
//myaccountname.file.core.windows.net/mystorageshare /mnt/mymountpoint cifs  
vers=3.0,username=mystorageaccount,password=myStorageAccountKeyEndingIn==,dir_mode=0777,file_mode=0777
```

For increased security in production environments, you should store your credentials outside of fstab.

Next steps

- [Using cloud-init to customize a Linux VM during creation](#)
- [Add a disk to a Linux VM](#)
- [Encrypt disks on a Linux VM by using the Azure CLI](#)

Frequently asked questions about Azure IaaS VM disks and managed and unmanaged premium disks

1/30/2019 • 15 minutes to read • [Edit Online](#)

This article answers some frequently asked questions about Azure Managed Disks and Azure Premium SSD disks.

Managed Disks

What is Azure Managed Disks?

Managed Disks is a feature that simplifies disk management for Azure IaaS VMs by handling storage account management for you. For more information, see the [Managed Disks overview](#).

If I create a standard managed disk from an existing VHD that's 80 GB, how much will that cost me?

A standard managed disk created from an 80-GB VHD is treated as the next available standard disk size, which is an S10 disk. You're charged according to the S10 disk pricing. For more information, see the [pricing page](#).

Are there any transaction costs for standard managed disks?

Yes. You're charged for each transaction. For more information, see the [pricing page](#).

For a standard managed disk, will I be charged for the actual size of the data on the disk or for the provisioned capacity of the disk?

You're charged based on the provisioned capacity of the disk. For more information, see the [pricing page](#).

How is pricing of premium managed disks different from unmanaged disks?

The pricing of premium managed disks is the same as unmanaged premium disks.

Can I change the storage account type (Standard or Premium) of my managed disks?

Yes. You can change the storage account type of your managed disks by using the Azure portal, PowerShell, or the Azure CLI.

Can I use a VHD file in an Azure storage account to create a managed disk with a different subscription?

Yes.

Can I use a VHD file in an Azure storage account to create a managed disk in a different region?

No.

Are there any scale limitations for customers that use managed disks?

Managed Disks eliminates the limits associated with storage accounts. However, the maximum limit is 50,000 managed disks per region and per disk type for a subscription.

Can I take an incremental snapshot of a managed disk?

No. The current snapshot capability makes a full copy of a managed disk.

Can VMs in an availability set consist of a combination of managed and unmanaged disks?

No. The VMs in an availability set must use either all managed disks or all unmanaged disks. When you create an availability set, you can choose which type of disks you want to use.

Is Managed Disks the default option in the Azure portal?

Yes.

Can I create an empty managed disk?

Yes. You can create an empty disk. A managed disk can be created independently of a VM, for example, without attaching it to a VM.

What is the supported fault domain count for an availability set that uses Managed Disks?

Depending on the region where the availability set that uses Managed Disks is located, the supported fault domain count is 2 or 3.

How is the standard storage account for diagnostics set up?

You set up a private storage account for VM diagnostics.

What kind of Role-Based Access Control support is available for Managed Disks?

Managed Disks supports three key default roles:

- Owner: Can manage everything, including access
- Contributor: Can manage everything except access
- Reader: Can view everything, but can't make changes

Is there a way that I can copy or export a managed disk to a private storage account?

You can generate a read-only shared access signature (SAS) URI for the managed disk and use it to copy the contents to a private storage account or on-premises storage. You can use the SAS URI using the Azure portal, Azure PowerShell, the Azure CLI, or [AzCopy](#)

Can I create a copy of my managed disk?

Customers can take a snapshot of their managed disks and then use the snapshot to create another managed disk.

Are unmanaged disks still supported?

Yes, both unmanaged and managed disks are supported. We recommend that you use managed disks for new workloads and migrate your current workloads to managed disks.

Can I co-locate unmanaged and managed disks on the same VM?

No.

If I create a 128-GB disk and then increase the size to 130 gibibytes (GiB), will I be charged for the next disk size (256 GiB)?

Yes.

Can I create locally redundant storage, geo-redundant storage, and zone-redundant storage managed disks?

Azure Managed Disks currently supports only locally redundant storage managed disks.

Can I shrink or downsize my managed disks?

No. This feature is not supported currently.

Can I break a lease on my disk?

No. This is not supported currently as a lease is present to prevent accidental deletion when the disk is being used.

Can I change the computer name property when a specialized (not created by using the System Preparation tool or generalized) operating system disk is used to provision a VM?

No. You can't update the computer name property. The new VM inherits it from the parent VM, which was used to create the operating system disk.

Where can I find sample Azure Resource Manager templates to create VMs with managed disks?

- [List of templates using Managed Disks](#)
- <https://github.com/chagarw/MDPP>

When creating a disk from a blob, is there any continually existing relationship with that source blob?

No, when the new disk is created it is a full standalone copy of that blob at that time and there is no connection between the two. If you like, once you've created the disk, the source blob may be deleted without affecting the newly created disk in any way.

Can I rename a managed or unmanaged disk after it has been created?

For managed disks you cannot rename them. However, you may rename an unmanaged disk as long as it is not currently attached to a VHD or VM.

Can I use GPT partitioning on an Azure Disk?

GPT partitioning can be used only on data disks, not OS disks. OS disks must use the MBR partition style.

What disk types support snapshots?

Premium SSD, standard SSD, and standard HDD support snapshots. For these three disk types, snapshots are supported for all disk sizes (including disks up to 32 TiB in size). Ultra SSDs do not support snapshots.

Standard SSD disks

What are Azure Standard SSD disks? Standard SSD disks are standard disks backed by solid-state media, optimized as cost effective storage for workloads that need consistent performance at lower IOPS levels.

What are the regions currently supported for Standard SSD disks? All Azure regions now support Standard SSD disks.

Is Azure Backup available when using Standard SSDs? Yes, Azure Backup is now available.

How do I create Standard SSD disks? You can create Standard SSD disks using Azure Resource Manager templates, SDK, PowerShell, or CLI. Below are the parameters needed in the Resource Manager template to create Standard SSD Disks:

- *apiVersion* for Microsoft.Compute must be set as `2018-04-01` (or later)
- Specify *managedDisk.storageAccountType* as `StandardSSD_LRS`

The following example shows the *properties.storageProfile.osDisk* section for a VM that uses Standard SSD Disks:

```
"osDisk": {  
    "osType": "Windows",  
    "name": "myOsDisk",  
    "caching": "ReadWrite",  
    "createOption": "FromImage",  
    "managedDisk": {  
        "storageAccountType": "StandardSSD_LRS"  
    }  
}
```

For a complete template example of how to create a Standard SSD disk with a template, see [Create a VM from a Windows Image with Standard SSD Data Disks](#).

Can I convert my existing disks to Standard SSD? Yes, you can. Refer to [Convert Azure managed disks storage from standard to premium, and vice versa](#) for the general guidelines for converting Managed Disks. And, use the following value to update the disk type to Standard SSD. -AccountType StandardSSD_LRS

What is the benefit of using Standard SSD disks instead of HDD? Standard SSD disks deliver better latency, consistency, availability, and reliability compared to HDD disks. Application workloads run a lot more smoothly on Standard SSD because of that. Note, Premium SSD disks are the recommended solution for most IO-intensive production workloads.

Can I use Standard SSDs as Unmanaged Disks? No, Standard SSDs disks are only available as Managed Disks.

Do Standard SSD Disks support "single instance VM SLA"? No, Standard SSDs do not have single instance VM SLA. Use Premium SSD disks for single instance VM SLA.

Migrate to Managed Disks

Is there any impact of migration on the Managed Disks performance?

Migration involves movement of the Disk from one Storage location to another. This is orchestrated via background copy of data, which can take several hours to complete, typically less than 24 Hrs 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.

What changes are required in a pre-existing Azure Backup service configuration prior/after migration to Managed Disks?

No changes are required.

Will my VM backups created via Azure Backup service before the migration continue to work?

Yes, backups work seamlessly.

What changes are required in a pre-existing Azure Disks Encryption configuration prior/after migration to Managed Disks?

No changes are required.

Is automated migration of an existing virtual machine scale set from unmanaged disks to Managed Disks supported?

No. You can create a new scale set with Managed Disks using the image from your old scale set with unmanaged disks.

Can I create a Managed Disk from a page blob snapshot taken before migrating to Managed Disks?

No. You can export a page blob snapshot as a page blob and then create a Managed Disk from the exported page blob.

Can I fail over my on-premises machines protected by Azure Site Recovery to a VM with Managed Disks?

Yes, you can choose to failover to a VM with Managed Disks.

Is there any impact of migration on Azure VMs protected by Azure Site Recovery via Azure to Azure replication?

Yes. Currently, Azure Site Recovery Azure to Azure protection for VMs with Managed Disks is available as a GA service.

Can I migrate VMs with unmanaged disks that are located on storage accounts that are or were previously encrypted to managed disks?

Yes

Managed Disks and Storage Service Encryption

Is Azure Storage Service Encryption enabled by default when I create a managed disk?

Yes.

Who manages the encryption keys?

Microsoft manages the encryption keys.

Can I disable Storage Service Encryption for my managed disks?

No.

Is Storage Service Encryption only available in specific regions?

No. It's available in all the regions where Managed Disks are available. Managed Disks is available in all public regions and Germany. It is also available in China, however, only for Microsoft managed keys, not customer managed keys.

How can I find out if my managed disk is encrypted?

You can find out the time when a managed disk was created from the Azure portal, the Azure CLI, and PowerShell. If the time is after June 9, 2017, then your disk is encrypted.

How can I encrypt my existing disks that were created before June 10, 2017?

As of June 10, 2017, new data written to existing managed disks is automatically encrypted. We are also planning to encrypt existing data, and the encryption will happen asynchronously in the background. If you must encrypt existing data now, create a copy of your disk. New disks will be encrypted.

- [Copy managed disks by using the Azure CLI](#)
- [Copy managed disks by using PowerShell](#)

Are managed snapshots and images encrypted?

Yes. All managed snapshots and images created after June 9, 2017, are automatically encrypted.

Can I convert VMs with unmanaged disks that are located on storage accounts that are or were previously encrypted to managed disks?

Yes

Will an exported VHD from a managed disk or a snapshot also be encrypted?

No. But if you export a VHD to an encrypted storage account from an encrypted managed disk or snapshot, then it's encrypted.

Premium disks: Managed and unmanaged

If a VM uses a size series that supports Premium SSD disks, such as a DSv2, can I attach both premium and standard data disks?

Yes.

Can I attach both premium and standard data disks to a size series that doesn't support Premium SSD disks, such as D, Dv2, G, or F series?

No. You can attach only standard data disks to VMs that don't use a size series that supports Premium SSD disks.

If I create a premium data disk from an existing VHD that was 80 GB, how much will that cost?

A premium data disk created from an 80-GB VHD is treated as the next-available premium disk size, which is a P10 disk. You're charged according to the P10 disk pricing.

Are there transaction costs to use Premium SSD disks?

There is a fixed cost for each disk size, which comes provisioned with specific limits on IOPS and throughput. The other costs are outbound bandwidth and snapshot capacity, if applicable. For more information, see the [pricing page](#).

What are the limits for IOPS and throughput that I can get from the disk cache?

The combined limits for cache and local SSD for a DS series are 4,000 IOPS per core and 33 MiB per second per core. The GS series offers 5,000 IOPS per core and 50 MiB per second per core.

Is the local SSD supported for a Managed Disks VM?

The local SSD is temporary storage that is included with a Managed Disks VM. There is no extra cost for this temporary storage. We recommend that you do not use this local SSD to store your application data because it isn't persisted in Azure Blob storage.

Are there any repercussions for the use of TRIM on premium disks?

There is no downside to the use of TRIM on Azure disks on either premium or standard disks.

New disk sizes: Managed and unmanaged

What is the largest Managed disk size supported for operating system and data disks?

The partition type that Azure supports for an operating system disk is the master boot record (MBR). The MBR format supports a disk size up to 2 TiB. The largest size that Azure supports for an operating system disk is 2 TiB. Azure supports up to 32 TiB for managed data disks in global Azure, 4 TiB in Azure sovereign clouds.

What is the largest Unmanaged Disk size supported for operating system and data disks?

The partition type that Azure supports for an operating system disk is the master boot record (MBR). The MBR format supports a disk size up to 2 TiB. The largest size that Azure supports for an operating system Unmanaged disk is 2 TiB. Azure supports up to 4 TiB for data Unmanaged disks.

What is the largest page blob size that's supported?

The largest page blob size that Azure supports is 8 TiB (8,191 GiB). The maximum page blob size when attached to a VM as data or operating system disks is 4 TiB (4,095 GiB).

Do I need to use a new version of Azure tools to create, attach, resize, and upload disks larger than 1 TiB?

You don't need to upgrade your existing Azure tools to create, attach, or resize disks larger than 1 TiB. To upload your VHD file from on-premises directly to Azure as a page blob or unmanaged disk, you need to use the latest tool sets listed below. We only support VHD uploads of up to 8 TiB.

AZURE TOOLS	SUPPORTED VERSIONS
Azure PowerShell	Version number 4.1.0: June 2017 release or later
Azure CLI v1	Version number 0.10.13: May 2017 release or later
Azure CLI v2	Version number 2.0.12: July 2017 release or later
AzCopy	Version number 6.1.0: June 2017 release or later

Are P4 and P6 disk sizes supported for unmanaged disks or page blobs?

P4 (32 GiB) and P6 (64 GiB) disk sizes are not supported as the default disk tiers for unmanaged disks and page blobs. You need to explicitly [set the Blob Tier](#) to P4 and P6 to have your disk mapped to these tiers. If you deploy a unmanaged disk or page blob with the disk size or content length less than 32 GiB or between 32 GiB to 64 GiB without setting the Blob Tier, you will continue to land on P10 with 500 IOPS and 100 MiB/s and the mapped pricing tier.

If my existing premium managed disk less than 64 GiB was created before the small disk was enabled (around June 15, 2017), how is it billed?

Existing small premium disks less than 64 GiB continue to be billed according to the P10 pricing tier.

How can I switch the disk tier of small premium disks less than 64 GiB from P10 to P4 or P6?

You can take a snapshot of your small disks and then create a disk to automatically switch the pricing tier to P4 or P6 based on the provisioned size.

Can you resize existing Managed Disks from sizes fewer than 4 tebibytes (TiB) to new newly introduced disk sizes up to 32 TiB?

Yes.

What are the largest disk sizes supported by Azure Backup and Azure Site Recovery service?

The largest disk size supported by Azure Backup and Azure Site Recovery service is 4 TiB. Support for the larger disks up to 32 TiB is not yet available.

What are the recommended VM sizes for larger disk sizes (>4 TiB) for Standard SSD and Standard HDD disks to achieve optimized disk IOPS and Bandwidth?

To achieve the disk throughput of Standard SSD and Standard HDD large disk sizes (>4 TiB) beyond 500 IOPS and 60 MiB/s, we recommend you deploy a new VM from one of the following VM sizes to optimize your performance: B-series, DSv2-series, Dsv3-Series, ESv3-Series, Fs-series, Fsv2-series, M-series, GS-series, NCv2-series, NCv3-series, or Ls-series VMs. Attaching large disks to existing VMs or VMs that are not using the recommended sizes above may experience lower performance.

How can I upgrade my disks (>4 TiB) which were deployed during the larger disk sizes preview in order to get the higher IOPS & bandwidth at GA?

You can either stop and start the VM that the disk is attached to or, detach and re-attach your disk. The performance targets of larger disk sizes have been increased for both premium SSDs and standard SSDs at GA.

What regions are the managed disk sizes of 8 TiB, 16 TiB, and 32 TiB supported in?

The 8 TiB, 16 TiB, and 32 TiB disk SKUs are supported in all regions under global Azure, Microsoft Azure Government, and Azure China 21Vianet.

Do we support enabling Host Caching on all disk sizes?

We support Host Caching of ReadOnly and Read/Write on disk sizes less than 4 TiB. For disk sizes more than 4 TiB, we don't support setting caching option other than None. We recommend leveraging caching for smaller disk sizes where you can expect to observe better performance boost with data cached to the VM.

What if my question isn't answered here?

If your question isn't listed here, let us know and we'll help you find an answer. You can post a question at the end of this article in the comments. To engage with the Azure Storage team and other community members about this article, use the MSDN [Azure Storage forum](#).

To request features, submit your requests and ideas to the [Azure Storage feedback forum](#).

2 minutes to read

Open ports and endpoints to a Linux VM with the Azure CLI

2/4/2019 • 2 minutes to read • [Edit Online](#)

You open a port, or create an endpoint, to a virtual machine (VM) in Azure by creating a network filter on a subnet or VM network interface. You place these filters, which control both inbound and outbound traffic, on a Network Security Group attached to the resource that receives the traffic. Let's use a common example of web traffic on port 80. This article shows you how to open a port to a VM with the Azure CLI.

To create a Network Security Group and rules you need the latest [Azure CLI](#) installed and logged in to an Azure account using [az login](#).

In the following examples, replace example parameter names with your own values. Example parameter names include *myResourceGroup*, *myNetworkSecurityGroup*, and *myVnet*.

Quickly open a port for a VM

If you need to quickly open a port for a VM in a dev/test scenario, you can use the [az vm open-port](#) command. This command creates a Network Security Group, adds a rule, and applies it to a VM or subnet. The following example opens port 80 on the VM named *myVM* in the resource group named *myResourceGroup*.

```
az vm open-port --resource-group myResourceGroup --name myVM --port 80
```

For more control over the rules, such as defining a source IP address range, continue with the additional steps in this article.

Create a Network Security Group and rules

Create the network security group with [az network nsg create](#). The following example creates a network security group named *myNetworkSecurityGroup* in the *eastus* location:

```
az network nsg create \
--resource-group myResourceGroup \
--location eastus \
--name myNetworkSecurityGroup
```

Add a rule with [az network nsg rule create](#) to allow HTTP traffic to your webserver (or adjust for your own scenario, such as SSH access or database connectivity). The following example creates a rule named *myNetworkSecurityGroupRule* to allow TCP traffic on port 80:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNetworkSecurityGroup \
--name myNetworkSecurityGroupRule \
--protocol tcp \
--priority 1000 \
--destination-port-range 80
```

Apply Network Security Group to VM

Associate the Network Security Group with your VM's network interface (NIC) with [az network nic update](#). The following example associates an existing NIC named *myNic* with the Network Security Group named *myNetworkSecurityGroup*:

```
az network nic update \
--resource-group myResourceGroup \
--name myNic \
--network-security-group myNetworkSecurityGroup
```

Alternatively, you can associate your Network Security Group with a virtual network subnet with [az network vnet subnet update](#) rather than just to the network interface on a single VM. The following example associates an existing subnet named *mySubnet* in the *myVnet* virtual network with the Network Security Group named *myNetworkSecurityGroup*:

```
az network vnet subnet update \
--resource-group myResourceGroup \
--vnet-name myVnet \
--name mySubnet \
--network-security-group myNetworkSecurityGroup
```

More information on Network Security Groups

The quick commands here allow you to get up and running with traffic flowing to your VM. Network Security Groups provide many great features and granularity for controlling access to your resources. You can read more about [creating a Network Security Group and ACL rules here](#).

For highly available web applications, you should place your VMs behind an Azure Load Balancer. The load balancer distributes traffic to VMs, with a Network Security Group that provides traffic filtering. For more information, see [How to load balance Linux virtual machines in Azure to create a highly available application](#).

Next steps

In this example, you created a simple rule to allow HTTP traffic. You can find information on creating more detailed environments in the following articles:

- [Azure Resource Manager overview](#)
- [What is a Network Security Group \(NSG\)?](#)

Create a virtual machine with a static public IP address using the Azure CLI

4/25/2019 • 2 minutes to read • [Edit Online](#)

You can create a virtual machine with a static public IP address. A public IP address enables you to communicate to a virtual machine from the internet. Assign a static public IP address, rather than a dynamic address, to ensure that the address never changes. Learn more about [static public IP addresses](#). To change a public IP address assigned to an existing virtual machine from dynamic to static, or to work with private IP addresses, see [Add, change, or remove IP addresses](#). Public IP addresses have a [nominal charge](#), and there is a [limit](#) to the number of public IP addresses that you can use per subscription.

Create a virtual machine

You can complete the following steps from your local computer or by using the Azure Cloud Shell. To use your local computer, ensure you have the [Azure CLI installed](#). To use the Azure Cloud Shell, select **Try It** in the top right corner of any command box that follows. The Cloud Shell signs you into Azure.

1. If using the Cloud Shell, skip to step 2. Open a command session and sign into Azure with `az login`.
2. Create a resource group with the `az group create` command. The following example creates a resource group in the East US Azure region:

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

3. Create a virtual machine with the `az vm create` command. The `--public-ip-address-allocation=static` option assigns a static public IP address to the virtual machine. The following example creates an Ubuntu virtual machine with a static, basic SKU public IP address named *myPublicIpAddress*:

```
az vm create \
  --resource-group myResourceGroup \
  --name myVM \
  --image UbuntuLTS \
  --admin-username azureuser \
  --generate-ssh-keys \
  --public-ip-address myPublicIpAddress \
  --public-ip-address-allocation static
```

If the public IP address must be a standard SKU, add `--public-ip-sku Standard` to the previous command. Learn more about [Public IP address SKUs](#). If the virtual machine will be added to the back-end pool of a public Azure Load Balancer, the SKU of the virtual machine's public IP address must match the SKU of the load balancer's public IP address. For details, see [Azure Load Balancer](#).

4. View the public IP address assigned and confirm that it was created as a static, basic SKU address, with `az network public-ip show`:

```
az network public-ip show \
  --resource-group myResourceGroup \
  --name myPublicIpAddress \
  --query [ipAddress,publicIpAllocationMethod,sku] \
  --output table
```

Azure assigned a public IP address from addresses used in the region you created the virtual machine in. You can download the list of ranges (prefixes) for the Azure [Public](#), [US government](#), [China](#), and [Germany](#) clouds.

WARNING

Do not modify the IP address settings within the virtual machine's operating system. The operating system is unaware of Azure public IP addresses. Though you can add private IP address settings to the operating system, we recommend not doing so unless necessary, and not until after reading [Add a private IP address to an operating system](#).

Clean up resources

When no longer needed, you can use `az group delete` to remove the resource group and all of the resources it contains:

```
az group delete --name myResourceGroup --yes
```

Next steps

- Learn more about [public IP addresses](#) in Azure
- Learn more about all [public IP address settings](#)
- Learn more about [private IP addresses](#) and assigning a [static private IP address](#) to an Azure virtual machine
- Learn more about creating [Linux](#) and [Windows](#) virtual machines

How to create a Linux virtual machine in Azure with multiple network interface cards

5/23/2019 • 6 minutes to read • [Edit Online](#)

This article details how to create a VM with multiple NICs with the Azure CLI.

Create supporting resources

Install the latest [Azure CLI](#) and log in to an Azure account using [az login](#).

In the following examples, replace example parameter names with your own values. Example parameter names included *myResourceGroup*, *mystorageaccount*, and *myVM*.

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
```

Create the virtual network with [az network vnet create](#). The following example creates a virtual network named *myVnet* and subnet named *mySubnetFrontEnd*:

```
az network vnet create \
  --resource-group myResourceGroup \
  --name myVnet \
  --address-prefix 10.0.0.0/16 \
  --subnet-name mySubnetFrontEnd \
  --subnet-prefix 10.0.1.0/24
```

Create a subnet for the back-end traffic with [az network vnet subnet create](#). The following example creates a subnet named *mySubnetBackEnd*:

```
az network vnet subnet create \
  --resource-group myResourceGroup \
  --vnet-name myVnet \
  --name mySubnetBackEnd \
  --address-prefix 10.0.2.0/24
```

Create a network security group with [az network nsg create](#). The following example creates a network security group named *myNetworkSecurityGroup*:

```
az network nsg create \
  --resource-group myResourceGroup \
  --name myNetworkSecurityGroup
```

Create and configure multiple NICs

Create two NICs with [az network nic create](#). The following example creates two NICs, named *myNic1* and *myNic2*, connected the network security group, with one NIC connecting to each subnet:

```
az network nic create \
--resource-group myResourceGroup \
--name myNic1 \
--vnet-name myVnet \
--subnet mySubnetFrontEnd \
--network-security-group myNetworkSecurityGroup
az network nic create \
--resource-group myResourceGroup \
--name myNic2 \
--vnet-name myVnet \
--subnet mySubnetBackEnd \
--network-security-group myNetworkSecurityGroup
```

Create a VM and attach the NICs

When you create the VM, specify the NICs you created with `--nics`. You also need to take care when you select the VM size. There are limits for the total number of NICs that you can add to a VM. Read more about [Linux VM sizes](#).

Create a VM with [az vm create](#). The following example creates a VM named *myVM*:

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--image UbuntuLTS \
--size Standard_DS3_v2 \
--admin-username azureuser \
--generate-ssh-keys \
--nics myNic1 myNic2
```

Add routing tables to the guest OS by completing the steps in [Configure the guest OS for multiple NICs](#).

Add a NIC to a VM

The previous steps created a VM with multiple NICs. You can also add NICs to an existing VM with the Azure CLI. Different [VM sizes](#) support a varying number of NICs, so size your VM accordingly. If needed, you can [resize a VM](#).

Create another NIC with [az network nic create](#). The following example creates a NIC named *myNic3* connected to the back-end subnet and network security group created in the previous steps:

```
az network nic create \
--resource-group myResourceGroup \
--name myNic3 \
--vnet-name myVnet \
--subnet mySubnetBackEnd \
--network-security-group myNetworkSecurityGroup
```

To add a NIC to an existing VM, first deallocate the VM with [az vm deallocate](#). The following example deallocates the VM named *myVM*:

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

Add the NIC with [az vm nic add](#). The following example adds *myNic3* to *myVM*:

```
az vm nic add \
--resource-group myResourceGroup \
--vm-name myVM \
--nics myNic3
```

Start the VM with [az vm start](#):

```
az vm start --resource-group myResourceGroup --name myVM
```

Add routing tables to the guest OS by completing the steps in [Configure the guest OS for multiple NICs](#).

Remove a NIC from a VM

To remove a NIC from an existing VM, first deallocate the VM with [az vm deallocate](#). The following example deallocates the VM named *myVM*:

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

Remove the NIC with [az vm nic remove](#). The following example removes *myNic3* from *myVM*:

```
az vm nic remove \
--resource-group myResourceGroup \
--vm-name myVM \
--nics myNic3
```

Start the VM with [az vm start](#):

```
az vm start --resource-group myResourceGroup --name myVM
```

Create multiple NICs using Resource Manager templates

Azure Resource Manager templates use declarative JSON files to define your environment. You can read an [overview of Azure Resource Manager](#). Resource Manager templates provide a way to create multiple instances of a resource during deployment, such as creating multiple NICs. You use *copy* to specify the number of instances to create:

```
"copy": {
  "name": "multiplenics"
  "count": "[parameters('count')]"
}
```

Read more about [creating multiple instances using copy](#).

You can also use a `copyIndex()` to then append a number to a resource name, which allows you to create `myNic1`, `myNic2`, etc. The following shows an example of appending the index value:

```
"name": "[concat('myNic', copyIndex())]",
```

You can read a complete example of [creating multiple NICs using Resource Manager templates](#).

Add routing tables to the guest OS by completing the steps in [Configure the guest OS for multiple NICs](#).

Configure guest OS for multiple NICs

The previous steps created a virtual network and subnet, attached NICs, then created a VM. A public IP address and network security group rules that allow SSH traffic were not created. To configure the guest OS for multiple NICs, you need to allow remote connections and run commands locally on the VM.

To allow SSH traffic, create a network security group rule with [az network nsg rule create](#) as follows:

```
az network nsg rule create \
    --resource-group myResourceGroup \
    --nsg-name myNetworkSecurityGroup \
    --name allow_ssh \
    --priority 101 \
    --destination-port-ranges 22
```

Create a public IP address with [az network public-ip create](#) and assign it to the first NIC with [az network nic ip-config update](#):

```
az network public-ip create --resource-group myResourceGroup --name myPublicIP

az network nic ip-config update \
    --resource-group myResourceGroup \
    --nic-name myNic1 \
    --name ipconfig1 \
    --public-ip myPublicIP
```

To view the public IP address of the VM, use [az vm show](#) as follows::

```
az vm show --resource-group myResourceGroup --name myVM -d --query publicIps -o tsv
```

Now SSH to the public IP address of your VM. The default username provided in a previous step was *azureuser*. Provide your own username and public IP address:

```
ssh azureuser@137.117.58.232
```

To send to or from a secondary network interface, you have to manually add persistent routes to the operating system for each secondary network interface. In this article, *eth1* is the secondary interface. Instructions for adding persistent routes to the operating system vary by distro. See documentation for your distro for instructions.

When adding the route to the operating system, the gateway address is *.1* for whichever subnet the network interface is in. For example, if the network interface is assigned the address *10.0.2.4*, the gateway you specify for the route is *10.0.2.1*. You can define a specific network for the route's destination, or specify a destination of *0.0.0.0*, if you want all traffic for the interface to go through the specified gateway. The gateway for each subnet is managed by the virtual network.

Once you've added the route for a secondary interface, verify that the route is in your route table with [route -n](#). The following example output is for the route table that has the two network interfaces added to the VM in this article:

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	10.0.1.1	0.0.0.0	UG	0	0	0	eth0
0.0.0.0	10.0.2.1	0.0.0.0	UG	0	0	0	eth1
10.0.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.2.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1
168.63.129.16	10.0.1.1	255.255.255.255	UGH	0	0	0	eth0
169.254.169.254	10.0.1.1	255.255.255.255	UGH	0	0	0	eth0

Confirm that the route you added persists across reboots by checking your route table again after a reboot. To test connectivity, you can enter the following command, for example, where *eth1* is the name of a secondary network interface:

```
ping bing.com -c 4 -I eth1
```

Next steps

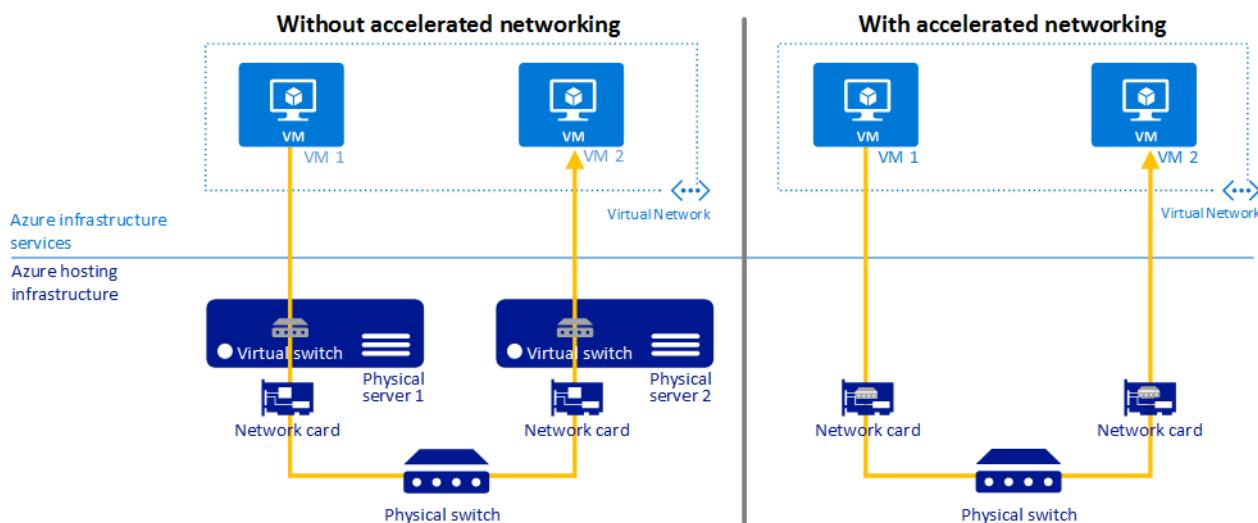
Review [Linux VM sizes](#) when trying to creating a VM with multiple NICs. Pay attention to the maximum number of NICs each VM size supports.

To further secure your VMs, use just in time VM access. This feature opens network security group rules for SSH traffic when needed, and for a defined period of time. For more information, see [Manage virtual machine access using just in time](#).

Create a Linux virtual machine with Accelerated Networking

5/17/2019 • 10 minutes to read • [Edit Online](#)

In this tutorial, you learn how to create a Linux virtual machine (VM) with Accelerated Networking. To create a Windows VM with Accelerated Networking, see [Create a Windows VM with Accelerated Networking](#). Accelerated networking enables single root I/O virtualization (SR-IOV) to a VM, greatly improving its networking performance. This high-performance path bypasses the host from the datapath, reducing latency, jitter, and CPU utilization, for use with the most demanding network workloads on supported VM types. The following picture shows communication between two VMs with and without accelerated networking:



Without accelerated networking, all networking traffic in and out of the VM must traverse the host and the virtual switch. The virtual switch provides all policy enforcement, such as network security groups, access control lists, isolation, and other network virtualized services to network traffic. To learn more about virtual switches, read the [Hyper-V network virtualization and virtual switch](#) article.

With accelerated networking, network traffic arrives at the virtual machine's network interface (NIC), and is then forwarded to the VM. All network policies that the virtual switch applies are now offloaded and applied in hardware. Applying policy in hardware enables the NIC to forward network traffic directly to the VM, bypassing the host and the virtual switch, while maintaining all the policy it applied in the host.

The benefits of accelerated networking only apply to the VM that it is enabled on. For the best results, it is ideal to enable this feature on at least two VMs connected to the same Azure virtual network (VNet). When communicating across VNets or connecting on-premises, this feature has minimal impact to overall latency.

Benefits

- **Lower Latency / Higher packets per second (pps):** Removing the virtual switch from the datapath removes the time packets spend in the host for policy processing and increases the number of packets that can be processed inside the VM.
- **Reduced jitter:** Virtual switch processing depends on the amount of policy that needs to be applied and the workload of the CPU that is doing the processing. Offloading the policy enforcement to the hardware removes that variability by delivering packets directly to the VM, removing the host to VM communication and all software interrupts and context switches.
- **Decreased CPU utilization:** Bypassing the virtual switch in the host leads to less CPU utilization for

processing network traffic.

Supported operating systems

The following distributions are supported out of the box from the Azure Gallery:

- **Ubuntu 14.04 with the linux-azure kernel**
- **Ubuntu 16.04 or later**
- **SLES12 SP3 or later**
- **RHEL 7.4 or later**
- **CentOS 7.4 or later**
- **CoreOS Linux**
- **Debian "Stretch" with backports kernel**
- **Oracle Linux 7.4 and later with Red Hat Compatible Kernel (RHCK)**
- **Oracle Linux 7.5 and later with UEK version 5**
- **FreeBSD 10.4, 11.1 & 12.0**

Limitations and Constraints

Supported VM instances

Accelerated Networking is supported on most general purpose and compute-optimized instance sizes with 2 or more vCPUs. These supported series are: D/DSv2 and F/Fs

On instances that support hyperthreading, Accelerated Networking is supported on VM instances with 4 or more vCPUs. Supported series are: D/Dsv3, E/Esv3, Fsv2, Lsv2, Ms/Mms and Ms/Mmsv2.

For more information on VM instances, see [Linux VM sizes](#).

Regions

Available in all public Azure regions as well as Azure Government Clouds.

Enabling Accelerated Networking on a running VM

A supported VM size without accelerated networking enabled can only have the feature enabled when it is stopped and deallocated.

Deployment through Azure Resource Manager

Virtual machines (classic) cannot be deployed with Accelerated Networking.

Create a Linux VM with Azure Accelerated Networking

Portal creation

Though this article provides steps to create a virtual machine with accelerated networking using the Azure CLI, you can also [create a virtual machine with accelerated networking using the Azure portal](#). When creating a virtual machine in the portal, in the **Create a virtual machine** blade, choose the **Networking** tab. In this tab, there is an option for **Accelerated networking**. If you have chosen a [supported operating system](#) and [VM size](#), this option will automatically populate to "On." If not, it will populate the "Off" option for Accelerated Networking and give the user a reason why it is not be enabled.

- *Note:* Only supported operating systems can be enabled through the portal. If you are using a custom image, and your image supports Accelerated Networking, please create your VM using CLI or Powershell.

After the virtual machine is created, you can confirm Accelerated Networking is enabled by following the instructions in the [Confirm that accelerated networking is enabled](#).

CLI creation

Create a virtual network

Install the latest [Azure CLI](#) and log in to an Azure account using [az login](#). In the following examples, replace example parameter names with your own values. Example parameter names included *myResourceGroup*, *myNic*, and *myVm*.

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

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

Select a supported Linux region listed in [Linux accelerated networking](#).

Create a virtual network with [az network vnet create](#). The following example creates a virtual network named *myVnet* with one subnet:

```
az network vnet create \
--resource-group myResourceGroup \
--name myVnet \
--address-prefix 192.168.0.0/16 \
--subnet-name mySubnet \
--subnet-prefix 192.168.1.0/24
```

Create a network security group

Create a network security group with [az network nsg create](#). The following example creates a network security group named *myNetworkSecurityGroup*:

```
az network nsg create \
--resource-group myResourceGroup \
--name myNetworkSecurityGroup
```

The network security group contains several default rules, one of which disables all inbound access from the Internet. Open a port to allow SSH access to the virtual machine with [az network nsg rule create](#):

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNetworkSecurityGroup \
--name Allow-SSH-Internet \
--access Allow \
--protocol Tcp \
--direction Inbound \
--priority 100 \
--source-address-prefix Internet \
--source-port-range "*" \
--destination-address-prefix "*" \
--destination-port-range 22
```

Create a network interface with accelerated networking

Create a public IP address with [az network public-ip create](#). A public IP address isn't required if you don't plan to access the virtual machine from the Internet, but to complete the steps in this article, it is required.

```
az network public-ip create \
--name myPublicIp \
--resource-group myResourceGroup
```

Create a network interface with [az network nic create](#) with accelerated networking enabled. The following example creates a network interface named *myNic* in the *mySubnet* subnet of the *myVnet* virtual network and associates the *myNetworkSecurityGroup* network security group to the network interface:

```
az network nic create \
--resource-group myResourceGroup \
--name myNic \
--vnet-name myVnet \
--subnet mySubnet \
--accelerated-networking true \
--public-ip-address myPublicIp \
--network-security-group myNetworkSecurityGroup
```

Create a VM and attach the NIC

When you create the VM, specify the NIC you created with `--nics`. Select a size and distribution listed in [Linux accelerated networking](#).

Create a VM with [az vm create](#). The following example creates a VM named *myVM* with the UbuntuLTS image and a size that supports Accelerated Networking (*Standard_DS4_v2*):

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--image UbuntuLTS \
--size Standard_DS4_v2 \
--admin-username azureuser \
--generate-ssh-keys \
--nics myNic
```

For a list of all VM sizes and characteristics, see [Linux VM sizes](#).

Once the VM is created, output similar to the following example output is returned. Take note of the **publicIpAddress**. This address is used to access the VM in subsequent steps.

```
{
  "fqdns": "",
  "id": "/subscriptions/<ID>/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "centralus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "192.168.0.4",
  "publicIpAddress": "40.68.254.142",
  "resourceGroup": "myResourceGroup"
}
```

Confirm that accelerated networking is enabled

Use the following command to create an SSH session with the VM. Replace `<your-public-ip-address>` with the public IP address assigned to the virtual machine you created, and replace `azureuser` if you used a different value for `--admin-username` when you created the VM.

```
ssh azureuser@<your-public-ip-address>
```

From the Bash shell, enter `uname -r` and confirm that the kernel version is one of the following versions, or greater:

- **Ubuntu 16.04:** 4.11.0-1013
- **SLES SP3:** 4.4.92-6.18
- **RHEL:** 7.4.2017120423
- **CentOS:** 7.4.20171206

Confirm the Mellanox VF device is exposed to the VM with the `lspci` command. The returned output is similar to the following output:

```
0000:00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (AGP disabled) (rev 03)
0000:00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 01)
0000:00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
0000:00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 02)
0000:00:08.0 VGA compatible controller: Microsoft Corporation Hyper-V virtual VGA
0001:00:02.0 Ethernet controller: Mellanox Technologies MT27500/MT27520 Family [ConnectX-3/ConnectX-3 Pro
Virtual Function]
```

Check for activity on the VF (virtual function) with the `ethtool -S eth0 | grep vf_` command. If you receive output similar to the following sample output, accelerated networking is enabled and working.

```
vf_rx_packets: 992956
vf_rx_bytes: 2749784180
vf_tx_packets: 2656684
vf_tx_bytes: 1099443970
vf_tx_dropped: 0
```

Accelerated Networking is now enabled for your VM.

Handle dynamic binding and revocation of virtual function

Applications must run over the synthetic NIC that is exposed in VM. If the application runs directly over the VF NIC, it doesn't receive **all** packets that are destined to the VM, since some packets show up over the synthetic interface. If you run an application over the synthetic NIC, it guarantees that the application receives **all** packets that are destined to it. It also makes sure that the application keeps running, even if the VF is revoked when the host is being serviced. Applications binding to the synthetic NIC is a **mandatory** requirement for all applications taking advantage of **Accelerated Networking**.

Enable Accelerated Networking on existing VMs

If you have created a VM without Accelerated Networking, it is possible to enable this feature on an existing VM. The VM must support Accelerated Networking by meeting the following prerequisites that are also outlined above:

- The VM must be a supported size for Accelerated Networking
- The VM must be a supported Azure Gallery image (and kernel version for Linux)
- All VMs in an availability set or VMSS must be stopped/deallocated before enabling Accelerated Networking on any NIC

Individual VMs & VMs in an availability set

First stop/deallocate the VM or, if an Availability Set, all the VMs in the Set:

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

Important, please note, if your VM was created individually, without an availability set, you only need to stop/deallocate the individual VM to enable Accelerated Networking. If your VM was created with an availability set, all VMs contained in the availability set will need to be stopped/deallocated before enabling Accelerated Networking on any of the NICs.

Once stopped, enable Accelerated Networking on the NIC of your VM:

```
az network nic update \
--name myNic \
--resource-group myResourceGroup \
--accelerated-networking true
```

Restart your VM or, if in an Availability Set, all the VMs in the Set and confirm that Accelerated Networking is enabled:

```
az vm start --resource-group myResourceGroup \
--name myVM
```

VMSS

VMSS is slightly different but follows the same workflow. First, stop the VMs:

```
az vmss deallocate \
--name myvmss \
--resource-group myrg
```

Once the VMs are stopped, update the Accelerated Networking property under the network interface:

```
az vmss update --name myvmss \
--resource-group myrg \
--set
virtualMachineProfile.networkProfile.networkInterfaceConfigurations[0].enableAcceleratedNetworking=true
```

Please note, a VMSS has VM upgrades that apply updates using three different settings, automatic, rolling and manual. In these instructions the policy is set to automatic so that the VMSS will pick up the changes immediately after restarting. To set it to automatic so that the changes are immediately picked up:

```
az vmss update \
--name myvmss \
--resource-group myrg \
--set upgradePolicy.mode="automatic"
```

Finally, restart the VMSS:

```
az vmss start \
--name myvmss \
--resource-group myrg
```

Once you restart, wait for the upgrades to finish but once completed, the VF will appear inside the VM. (Please make sure you are using a supported OS and VM size.)

Resizing existing VMs with Accelerated Networking

VMs with Accelerated Networking enabled can only be resized to VMs that support Accelerated Networking.

A VM with Accelerated Networking enabled cannot be resized to a VM instance that does not support Accelerated Networking using the resize operation. Instead, to resize one of these VMs:

- Stop/Deallocate the VM or if in an availability set/VMSS, stop/deallocate all the VMs in the set/VMSS.
- Accelerated Networking must be disabled on the NIC of the VM or if in an availability set/VMSS, all VMs in the set/VMSS.
- Once Accelerated Networking is disabled, the VM/availability set/VMSS can be moved to a new size that does not support Accelerated Networking and restarted.

Create a fully qualified domain name in the Azure portal for a Linux VM

8/16/2018 • 2 minutes to read • [Edit Online](#)

When you create a virtual machine (VM) in the [Azure portal](#), a public IP resource for the virtual machine is automatically created. You use this IP address to remotely access the VM. Although the portal does not create a [fully qualified domain name](#), or FQDN, you can add one once the VM is created. This article demonstrates the steps to create a DNS name or FQDN.

Create a FQDN

This article assumes that you have already created a VM. If needed, you can [create a VM in the portal](#) or [with the Azure CLI](#). Follow these steps once your VM is up and running:

1. Select your VM in the portal. Under **DNS name**, click **Configure**.

The screenshot shows the Azure portal interface for a virtual machine named 'myVM'. The left sidebar lists navigation options like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main content area displays VM details under 'Resource group (change)' 'myResourceGroup'. It includes fields for Computer name ('myVM'), Operating system ('Linux'), Size ('Standard DS1 v2 (1 vcpus, 3.5 GB memory)'), Public IP address ('40.76.54.250'), and Virtual network/subnet ('myVMNET/myVMSubnet'). The 'DNS name' field is labeled 'Configure' and is highlighted with a red box. Below it, there's a 'Tags (change)' section with a link to 'Click here to add tags'.

2. Enter the desired DNS name and then select **Save**.

The screenshot shows the 'Configuration' dialog for a public IP named 'myVMPublicIP'. It has two tabs: 'Save' (selected) and 'Discard'. Under 'Assignment', 'Dynamic' is selected. The 'IP address' is listed as '40.76.54.250'. The 'Idle timeout (minutes)' slider is set to '4'. At the bottom, there's a 'DNS name label (optional)' input field containing '.eastus.cloudapp.azure.com', which is also highlighted with a red box. A note at the bottom says 'Prefer to use your own domain name? Try Azure DNS now'.

3. To return to the VM overview blade, close the *Public IP address* blade. Verify that the *DNS name* is now shown.

You can now connect remotely to the VM using this DNS name such as with

```
ssh azureuser@mydns.westus.cloudapp.azure.com .
```

Next steps

Now that your VM has a public IP and DNS name, you can deploy common application frameworks or services such as nginx, MongoDB, Docker, etc.

You can also read more about [using Resource Manager](#) for tips on building your Azure deployments.

How to find and delete unattached network interface cards (NICs) for Azure VMs

9/24/2018 • 2 minutes to read • [Edit Online](#)

When you delete a virtual machine (VM) in Azure, the network interface cards (NICs) are not deleted by default. If you create and delete multiple VMs, the unused NICs continue to use the internal IP address leases. As you create other VM NICs, they may be unable to obtain an IP lease in the address space of the subnet. This article shows you how to find and delete unattached NICs.

Find and delete unattached NICs

The *virtualMachine* property for a NIC stores the ID and resource group of the VM the NIC is attached to. The following script loops through all the NICs in a subscription and checks if the *virtualMachine* property is null. If this property is null, the NIC is not attached to a VM.

To view all the unattached NICs, it's highly recommend to first run the script with the *deleteUnattachedNics* variable to 0. To delete all the unattached NICs after you review the list output, run the script with *deleteUnattachedNics* to 1.

```
# Set deleteUnattachedNics=1 if you want to delete unattached NICs
# Set deleteUnattachedNics=0 if you want to see the Id(s) of the unattached NICs
deleteUnattachedNics=0

unattachedNicsIds=$(az network nic list --query '[?virtualMachine==`null`].[id]' -o tsv)
for id in ${unattachedNicsIds[@]}
do
    if (( $deleteUnattachedNics == 1 ))
    then
        echo "Deleting unattached NIC with Id: \"$id"
        az network nic delete --ids $id
        echo "Deleted unattached NIC with Id: \"$id"
    else
        echo $id
    fi
done
```

Next steps

For more information on how to create and manage virtual networks in Azure, see [create and manage VM networks](#).

DNS Name Resolution options for Linux virtual machines in Azure

2/6/2019 • 7 minutes to read • [Edit Online](#)

Azure provides DNS name resolution by default for all virtual machines that are in a single virtual network. You can implement your own DNS name resolution solution by configuring your own DNS services on your virtual machines that Azure hosts. The following scenarios should help you choose the one that works for your situation.

- [Name resolution that Azure provides](#)
- [Name resolution using your own DNS server](#)

The type of name resolution that you use depends on how your virtual machines and role instances need to communicate with each other.

The following table illustrates scenarios and corresponding name resolution solutions:

SCENARIO	SOLUTION	SUFFIX
Name resolution between role instances or virtual machines in the same virtual network	Name resolution that Azure provides	hostname or fully-qualified domain name (FQDN)
Name resolution between role instances or virtual machines in different virtual networks	Customer-managed DNS servers that forward queries between virtual networks for resolution by Azure (DNS proxy). See Name resolution using your own DNS server .	FQDN only
Resolution of on-premises computers and service names from role instances or virtual machines in Azure	Customer-managed DNS servers (for example, on-premises domain controller, local read-only domain controller, or a DNS secondary synced by using zone transfers). See Name resolution using your own DNS server .	FQDN only
Resolution of Azure hostnames from on-premises computers	Forward queries to a customer-managed DNS proxy server in the corresponding virtual network. The proxy server forwards queries to Azure for resolution. See Name resolution using your own DNS server .	FQDN only
Reverse DNS for internal IPs	Name resolution using your own DNS server	n/a

Name resolution that Azure provides

Along with resolution of public DNS names, Azure provides internal name resolution for virtual machines and role instances that are in the same virtual network. In virtual networks that are based on Azure Resource Manager, the DNS suffix is consistent across the virtual network; the FQDN is not needed. DNS names can be assigned to both network interface cards (NICs) and virtual machines. Although the name resolution that Azure provides does not require any configuration, it is not the appropriate choice for all deployment scenarios, as seen on the preceding

table.

Features and considerations

Features:

- No configuration is required to use name resolution that Azure provides.
- The name resolution service that Azure provides is highly available. You don't need to create and manage clusters of your own DNS servers.
- The name resolution service that Azure provides can be used along with your own DNS servers to resolve both on-premises and Azure hostnames.
- Name resolution is provided between virtual machines in virtual networks without need for the FQDN.
- You can use hostnames that best describe your deployments rather than working with auto-generated names.

Considerations:

- The DNS suffix that Azure creates cannot be modified.
- You cannot manually register your own records.
- WINS and NetBIOS are not supported.
- Hostnames must be DNS-compatible. Names must use only 0-9, a-z, and '-', and they cannot start or end with a '-'. See RFC 3696 Section 2.
- DNS query traffic is throttled for each virtual machine. Throttling shouldn't impact most applications. If request throttling is observed, ensure that client-side caching is enabled. For more information, see [Getting the most from name resolution that Azure provides](#).

Getting the most from name resolution that Azure provides

Client-side caching:

Some DNS queries are not sent across the network. Client-side caching helps reduce latency and improve resilience to network inconsistencies by resolving recurring DNS queries from a local cache. DNS records contain a Time-To-Live (TTL), which enables the cache to store the record for as long as possible without impacting record freshness. As a result, client-side caching is suitable for most situations.

Some Linux distributions do not include caching by default. We recommend that you add a cache to each Linux virtual machine after you check that there isn't a local cache already.

Several different DNS caching packages, such as dnsmasq, are available. Here are the steps to install dnsmasq on the most common distributions:

Ubuntu (uses resolvconf)

- Install the dnsmasq package ("sudo apt-get install dnsmasq").

SUSE (uses netconf):

1. Install the dnsmasq package ("sudo zypper install dnsmasq").
2. Enable the dnsmasq service ("systemctl enable dnsmasq.service").
3. Start the dnsmasq service ("systemctl start dnsmasq.service").
4. Edit "/etc/sysconfig/network/config", and change NETCONFIG_DNS_FORWARDER="" to "dnsmasq".
5. Update resolv.conf ("netconfig update") to set the cache as the local DNS resolver.

CentOS by Rogue Wave Software (formerly OpenLogic; uses NetworkManager)

1. Install the dnsmasq package ("sudo yum install dnsmasq").
2. Enable the dnsmasq service ("systemctl enable dnsmasq.service").
3. Start the dnsmasq service ("systemctl start dnsmasq.service").
4. Add "prepend domain-name-servers 127.0.0.1;" to "/etc/dhclient-eth0.conf".

5. Restart the network service ("service network restart") to set the cache as the local DNS resolver

NOTE

: The 'dnsmasq' package is only one of the many DNS caches that are available for Linux. Before you use it, check its suitability for your needs and that no other cache is installed.

Client-side retries

DNS is primarily a UDP protocol. Because the UDP protocol doesn't guarantee message delivery, the DNS protocol itself handles retry logic. Each DNS client (operating system) can exhibit different retry logic depending on the creator's preference:

- Windows operating systems retry after one second and then again after another two, four, and another four seconds.
- The default Linux setup retries after five seconds. You should change this to retry five times at one-second intervals.

To check the current settings on a Linux virtual machine, 'cat /etc/resolv.conf', and look at the 'options' line, for example:

```
options timeout:1 attempts:5
```

The resolv.conf file is auto-generated and should not be edited. The specific steps that add the 'options' line vary by distribution:

Ubuntu (uses resolvconf)

1. Add the options line to '/etc/resolvconf/resolv.conf.d/head'.
2. Run 'resolvconf -u' to update.

SUSE (uses netconfig)

1. Add 'timeout:1 attempts:5' to the NETCONFIG_DNS_RESOLVER_OPTIONS="" parameter in '/etc/sysconfig/network/config'.
2. Run 'netconfig update' to update.

CentOS by Rogue Wave Software (formerly OpenLogic) (uses NetworkManager)

1. Add 'RES_OPTIONS="timeout:1 attempts:5"' to '/etc/sysconfig/network'.
2. Run 'service network restart' to update.

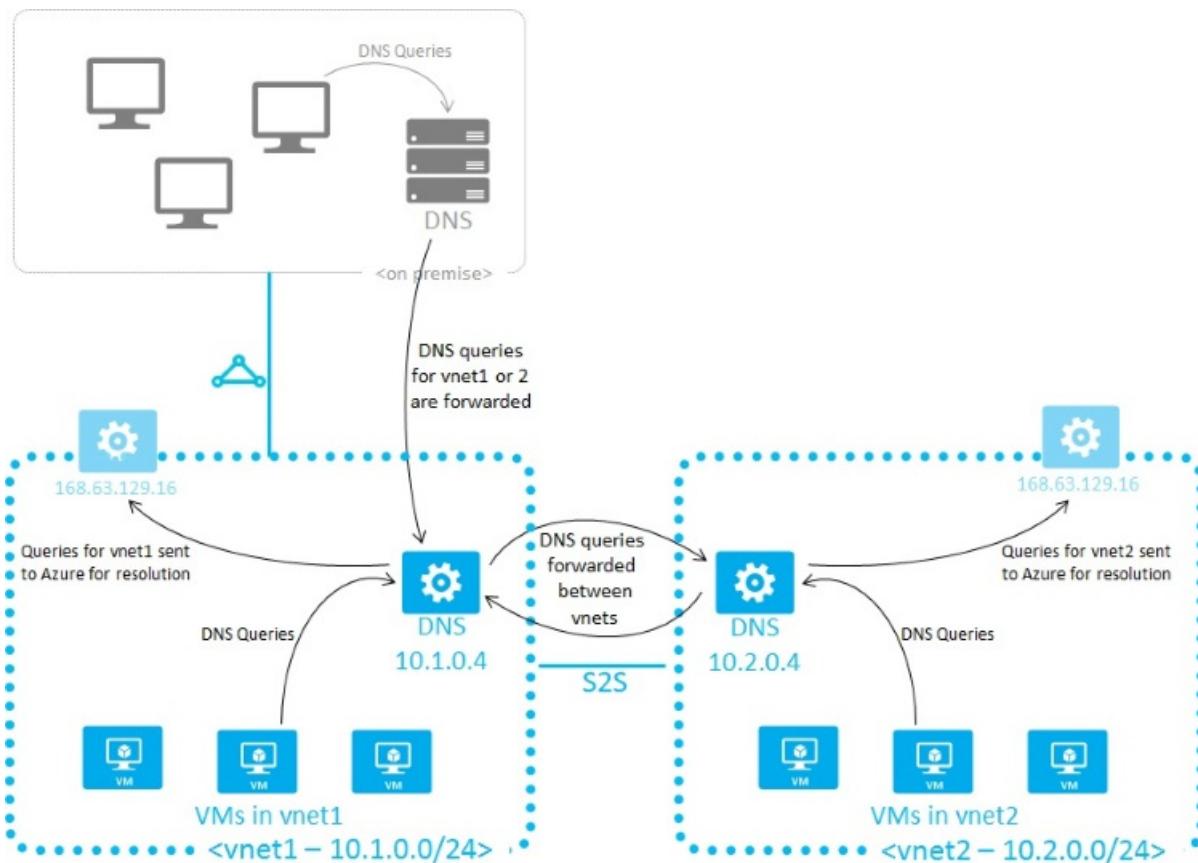
Name resolution using your own DNS server

Your name resolution needs may go beyond the features that Azure provides. For example, you might require DNS resolution between virtual networks. To cover this scenario, you can use your own DNS servers.

DNS servers within a virtual network can forward DNS queries to recursive resolvers of Azure to resolve hostnames that are in the same virtual network. For example, a DNS server that runs in Azure can respond to DNS queries for its own DNS zone files and forward all other queries to Azure. This functionality enables virtual machines to see both your entries in your zone files and hostnames that Azure provides (via the forwarder). Access to the recursive resolvers of Azure is provided via the virtual IP 168.63.129.16.

DNS forwarding also enables DNS resolution between virtual networks and enables your on-premises machines to resolve hostnames that Azure provides. To resolve a virtual machine's hostname, the DNS server virtual machine must reside in the same virtual network and be configured to forward hostname queries to Azure.

Because the DNS suffix is different in each virtual network, you can use conditional forwarding rules to send DNS queries to the correct virtual network for resolution. The following image shows two virtual networks and an on-premises network doing DNS resolution between virtual networks by using this method:



When you use name resolution that Azure provides, the internal DNS suffix is provided to each virtual machine by using DHCP. When you use your own name resolution solution, this suffix is not supplied to virtual machines because the suffix interferes with other DNS architectures. To refer to machines by FQDN or to configure the suffix on your virtual machines, you can use PowerShell or the API to determine the suffix:

- For virtual networks that are managed by Azure Resource Manager, the suffix is available via the [network interface card](#) resource. You can also run the `azurerm network public-ip show <resource group> <ip name>` command to display the details of your public IP, which includes the FQDN of the NIC.

If forwarding queries to Azure doesn't suit your needs, you need to provide your own DNS solution. Your DNS solution needs to:

- Provide appropriate hostname resolution, for example via [DDNS](#). If you use DDNS, you might need to disable DNS record scavenging. DHCP leases of Azure are very long and scavenging may remove DNS records prematurely.
- Provide appropriate recursive resolution to allow resolution of external domain names.
- Be accessible (TCP and UDP on port 53) from the clients it serves and be able to access the Internet.
- Be secured against access from the Internet to mitigate threats posed by external agents.

NOTE

For best performance, when you use virtual machines in Azure DNS servers, disable IPv6 and assign an [Instance-Level Public IP](#) to each DNS server virtual machine.

Create virtual network interface cards and use internal DNS for VM name resolution on Azure

2/4/2019 • 5 minutes to read • [Edit Online](#)

This article shows you how to set static internal DNS names for Linux VMs using virtual network interface cards (vNics) and DNS label names with the Azure CLI. Static DNS names are used for permanent infrastructure services like a Jenkins build server, which is used for this document, or a Git server.

The requirements are:

- [an Azure account](#)
- [SSH public and private key files](#)

Quick commands

If you need to quickly accomplish the task, the following section details the commands needed. More detailed information and context for each step can be found in the rest of the document, [starting here](#). To perform these steps, you need the latest [Azure CLI](#) installed and logged in to an Azure account using [az login](#).

Pre-Requirements: Resource Group, virtual network and subnet, Network Security Group with SSH inbound.

Create a virtual network interface card with a static internal DNS name

Create the vNic with [az network nic create](#). The `--internal-dns-name` CLI flag is for setting the DNS label, which provides the static DNS name for the virtual network interface card (vNic). The following example creates a vNic named `myNic`, connects it to the `myVnet` virtual network, and creates an internal DNS name record called `jenkins`:

```
az network nic create \
    --resource-group myResourceGroup \
    --name myNic \
    --vnet-name myVnet \
    --subnet mySubnet \
    --internal-dns-name jenkins
```

Deploy a VM and connect the vNic

Create a VM with [az vm create](#). The `--nics` flag connects the vNic to the VM during the deployment to Azure. The following example creates a VM named `myVM` with Azure Managed Disks and attaches the vNic named `myNic` from the preceding step:

```
az vm create \
    --resource-group myResourceGroup \
    --name myVM \
    --nics myNic \
    --image UbuntuLTS \
    --admin-username azureuser \
    --ssh-key-value ~/.ssh/id_rsa.pub
```

Detailed walkthrough

A full continuous integration and continuous deployment (CiCd) infrastructure on Azure requires certain servers to

be static or long-lived servers. It is recommended that Azure assets like the virtual networks and Network Security Groups are static and long lived resources that are rarely deployed. Once a virtual network has been deployed, it can be reused by new deployments without any adverse affects to the infrastructure. You can later add a Git repository server or a Jenkins automation server delivers CiCd to this virtual network for your development or test environments.

Internal DNS names are only resolvable inside an Azure virtual network. Because the DNS names are internal, they are not resolvable to the outside internet, providing additional security to the infrastructure.

In the following examples, replace example parameter names with your own values. Example parameter names include `myResourceGroup`, `myNic`, and `myVM`.

Create the resource group

First, create the resource group with [az group create](#). The following example creates a resource group named `myResourceGroup` in the `westus` location:

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

Create the virtual network

The next step is to build a virtual network to launch the VMs into. The virtual network contains one subnet for this walkthrough. For more information on Azure virtual networks, see [Create a virtual network](#).

Create the virtual network with [az network vnet create](#). The following example creates a virtual network named `myVnet` and subnet named `mySubnet`:

```
az network vnet create \
    --resource-group myResourceGroup \
    --name myVnet \
    --address-prefix 192.168.0.0/16 \
    --subnet-name mySubnet \
    --subnet-prefix 192.168.1.0/24
```

Create the Network Security Group

Azure Network Security Groups are equivalent to a firewall at the network layer. For more information about Network Security Groups, see [How to create NSGs in the Azure CLI](#).

Create the network security group with [az network nsg create](#). The following example creates a network security group named `myNetworkSecurityGroup`:

```
az network nsg create \
    --resource-group myResourceGroup \
    --name myNetworkSecurityGroup
```

Add an inbound rule to allow SSH

Add an inbound rule for the network security group with [az network nsg rule create](#). The following example creates a rule named `myRuleAllowSSH`:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNetworkSecurityGroup \
--name myRuleAllowSSH \
--protocol tcp \
--direction inbound \
--priority 1000 \
--source-address-prefix '*' \
--source-port-range '*' \
--destination-address-prefix '*' \
--destination-port-range 22 \
--access allow
```

Associate the subnet with the Network Security Group

To associate the subnet with the Network Security Group, use [az network vnet subnet update](#). The following example associates the subnet name `mySubnet` with the Network Security Group named `myNetworkSecurityGroup`:

```
az network vnet subnet update \
--resource-group myResourceGroup \
--vnet-name myVnet \
--name mySubnet \
--network-security-group myNetworkSecurityGroup
```

Create the virtual network interface card and static DNS names

Azure is very flexible, but to use DNS names for VM name resolution, you need to create virtual network interface cards (vNics) that include a DNS label. vNics are important as you can reuse them by connecting them to different VMs over the infrastructure lifecycle. This approach keeps the vNic as a static resource while the VMs can be temporary. By using DNS labeling on the vNic, we are able to enable simple name resolution from other VMs in the VNet. Using resolvable names enables other VMs to access the automation server by the DNS name `Jenkins` or the Git server as `gitrepo`.

Create the vNic with [az network nic create](#). The following example creates a vNic named `myNic`, connects it to the `myVnet` virtual network named `myVnet`, and creates an internal DNS name record called `jenkins`:

```
az network nic create \
--resource-group myResourceGroup \
--name myNic \
--vnet-name myVnet \
--subnet mySubnet \
--internal-dns-name jenkins
```

Deploy the VM into the virtual network infrastructure

We now have a virtual network and subnet, a Network Security Group acting as a firewall to protect our subnet by blocking all inbound traffic except port 22 for SSH, and a vNic. You can now deploy a VM inside this existing network infrastructure.

Create a VM with [az vm create](#). The following example creates a VM named `myVM` with Azure Managed Disks and attaches the vNic named `myNic` from the preceding step:

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--nics myNic \
--image UbuntuLTS \
--admin-username azureuser \
--ssh-key-value ~/.ssh/id_rsa.pub
```

By using the CLI flags to call out existing resources, we instruct Azure to deploy the VM inside the existing network. To reiterate, once a VNet and subnet have been deployed, they can be left as static or permanent resources inside your Azure region.

Next steps

- [Create your own custom environment for a Linux VM using Azure CLI commands directly](#)
- [Create a Linux VM on Azure using templates](#)

Azure virtual machine extensions and features

2/26/2019 • 2 minutes to read • [Edit Online](#)

Azure virtual machine (VM) extensions are small applications that provide post-deployment configuration and automation tasks on Azure VMs, you can use existing images and then customize them as part of your deployments, getting you out of the business of custom image building.

The Azure platform hosts many extensions that range from VM configuration, monitoring, security, and utility applications. Publishers take an application, then wrap it into an extension, and simplify the installation, so all you need to do is provide mandatory parameters.

There is a large choice of first and third party extensions, if the application in the extension repository does not exist, then you can use the Custom Script extension and configure your VM with your own scripts and commands.

Examples of key scenarios that extensions are used for:

- VM configuration, you can use Powershell DSC (Desired State Configuration), Chef, Puppet and Custom Script Extensions to install VM configuration agents and configure your VM.
- AV products, such as Symantec, ESET.
- VM vulnerability tool, such as Qualys, Rapid7, HPE.
- VM and App monitoring tooling, such as DynaTrace, Azure Network Watcher, Site24x7, and Stackify.

Extensions can be bundled with a new VM deployment. For example, they can be part of a larger deployment, configuring applications on VM provision, or run against any supported extension operated systems post deployment.

How can I find What extensions are available?

You can view available extensions in the VM blade in the Portal, under extensions, this represents just a small amount, for the full list, you can use the CLI tools, see [Discovering VM Extensions for Linux](#) and [Discovering VM Extensions for Windows](#).

How can I install an extension?

Azure VM extensions can be managed using either the Azure CLI, Azure PowerShell, Azure Resource Manager templates, and the Azure portal. To try an extension, you can go to the Azure portal, select the Custom Script Extension, then pass in a command / script and run the extensions.

If you want to same extension you added in the portal by CLI or Resource Manager template, see different extension documentation, such as [Windows Custom Script Extension](#) and [Linux Custom Script Extension](#).

How do I manage extension application lifecycle?

You do not need to connect to a VM directly to install or delete the extension. As the Azure extension application lifecycle is managed outside of the VM and integrated into the Azure platform, you also get integrated status of the extension.

Anything else I should be thinking about for extensions?

Extensions install applications, like any applications there are some requirements, for extensions there is a list of supported Windows and Linux OSes, and you need to have the Azure VM agents installed. Some individual VM

extension applications may have their own environmental prerequisites, such as access to an endpoint.

Next steps

- For more information about how the Linux Agent and Extensions work, see [Azure VM extensions and features for Linux](#).
- For more information about how the Windows Guest Agent and Extensions work, see [Azure VM extensions and features for Windows](#).
- To install the Windows Guest Agent, see [Azure Windows Virtual Machine Agent Overview](#).
- To install the Linux Agent, see [Azure Linux Virtual Machine Agent Overview](#).

Move a Linux VM to another subscription or resource group

2/4/2019 • 3 minutes to read • [Edit Online](#)

This article walks you through how to move a Linux virtual machine (VM) between resource groups or subscriptions. Moving a VM between subscriptions can be handy if you created a VM in a personal subscription and now want to move it to your company's subscription.

IMPORTANT

You cannot move Azure Managed Disks at this time.

New resource IDs are created as part of the move. After the VM has been moved, you will need to update your tools and scripts to use the new resource IDs.

Use the Azure CLI to move a VM

Before you can move your VM by using the Azure CLI, you need to make sure the source and destination subscriptions exist within the same tenant. To check that both subscriptions have the same tenant ID, use [az account show](#).

```
az account show --subscription mySourceSubscription --query tenantId  
az account show --subscription myDestinationSubscription --query tenantId
```

If the tenant IDs for the source and destination subscriptions are not the same, you must contact [support](#) to move the resources to a new tenant.

To successfully move a VM, you need to move the VM and all its supporting resources. Use the [az resource list](#) command to list all the resources in a resource group and their IDs. It helps to pipe the output of this command to a file so you can copy and paste the IDs into later commands.

```
az resource list --resource-group "mySourceResourceGroup" --query "[].{Id:id}" --output table
```

To move a VM and its resources to another resource group, use [az resource move](#). The following example shows how to move a VM and the most common resources it requires. Use the **-ids** parameter and pass in a comma-separated list (without spaces) of IDs for the resources to move.

```

vm=/subscriptions/mySourceSubscriptionID/resourceGroups/mySourceResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM
nic=/subscriptions/mySourceSubscriptionID/resourceGroups/mySourceResourceGroup/providers/Microsoft.Network/networkInterfaces/myNIC
nsg=/subscriptions/mySourceSubscriptionID/resourceGroups/mySourceResourceGroup/providers/Microsoft.Network/networkSecurityGroups/myNSG
pip=/subscriptions/mySourceSubscriptionID/resourceGroups/mySourceResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIPAddress
vnet=/subscriptions/mySourceSubscriptionID/resourceGroups/mySourceResourceGroup/providers/Microsoft.Network/virtualNetworks/myVNet
diag=/subscriptions/mySourceSubscriptionID/resourceGroups/mySourceResourceGroup/providers/Microsoft.Storage/storageAccounts/mydiagnosticstorageaccount
storage=/subscriptions/mySourceSubscriptionID/resourceGroups/mySourceResourceGroup/providers/Microsoft.Storage/storageAccounts/mystorageaccountname

az resource move \
--ids $vm,$nic,$nsg,$pip,$vnet,$storage,$diag \
--destination-group "myDestinationResourceGroup"

```

If you want to move the VM and its resources to a different subscription, add the **--destination-subscriptionId** parameter to specify the destination subscription.

When you are asked to confirm that you want to move the specified resources, enter **Y** to confirm.

Use the Azure portal to move a VM to a different subscription

You can move a VM and its associated resources to a different subscription by using the Azure portal.

1. Open the [Azure portal](#).
2. Click **Browse > Resource groups** and select the resource group containing the VM that you would like to move.
3. At the top of the page for the resource group, select **Move** and then select **Move to another subscription**. The **Move resources** page opens.
4. Select each of the resources to move. In most cases, you should move all of the related resources that are listed.
5. Select the **Subscription** where you want the VM to be moved.
6. Select an existing **Resource group**, or enter a name to have a new resource group created.
7. When you are done, select that you understand that new resource IDs will be created and that the new IDs will need to be used with the VM after it is moved, and then select **OK**.

Use the Azure portal to move a VM to another resource group

You can move a VM and its associated resources to another resource group by using the Azure portal.

1. Open the [Azure portal](#).
2. Click **Browse > Virtual machines** and select the VM you would like to move from the list.
3. In the page for the VM, next to the label for resource group, select **Change**. The **Move resources** page opens.
4. Select each of the resources to move. In most cases, you should move all of the related resources that are listed.
5. Select an existing **Resource group**, or enter a name to have a new resource group created.
6. When you are done, select that you understand that new resource IDs will be created and that the new IDs will need to be used with the VM after it is moved, and then select **OK**.

Next steps

You can move many different types of resources between resource groups and subscriptions. For more information, see [Move resources to a new resource group or subscription](#).

Move Azure VMs to another region

5/14/2019 • 7 minutes to read • [Edit Online](#)

There are various scenarios in which you'd want to move your existing Azure IaaS virtual machines (VMs) from one region to another. For example, you want to improve reliability and availability of your existing VMs, to improve manageability, or to move for governance reasons. For more information, see the [Azure VM move overview](#).

You can use the [Azure Site Recovery](#) service to manage and orchestrate disaster recovery of on-premises machines and Azure VMs for business continuity and disaster recovery (BCDR). You can also use Site Recovery to manage the move of Azure VMs to a secondary region.

In this tutorial, you will:

- Verify prerequisites for the move
- Prepare the source VMs and the target region
- Copy the data and enable replication
- Test the configuration and perform the move
- Delete the resources in the source region

NOTE

This tutorial shows you how to move Azure VMs from one region to another as is. If you need to improve availability by moving VMs in an availability set to zone pinned VMs in a different region, see the [Move Azure VMs into Availability Zones tutorial](#).

Prerequisites

- Make sure that the Azure VMs are in the Azure region from which you want to move.
- Verify that your choice of [source region - target region combination is supported](#), and make an informed decision about the target region.
- Make sure that you understand the [scenario architecture and components](#).
- Review the [support limitations and requirements](#).
- Verify account permissions. If you created your free Azure account, you're the administrator of your subscription. If you're not the subscription administrator, work with the administrator to assign the permissions that you need. To enable replication for a VM and essentially copy data by using Azure Site Recovery, you must have:
 - Permissions to create a VM in Azure resources. The Virtual Machine Contributor built-in role has these permissions, which include:
 - Permission to create a VM in the selected resource group
 - Permission to create a VM in the selected virtual network
 - Permission to write to the selected storage account
 - Permissions to manage Azure Site Recovery operations. The Site Recovery Contributor role has all the permissions that are required to manage Site Recovery operations in a Recovery Services vault.

Prepare the source VMs

1. Make sure that all the latest root certificates are on the Azure VMs that you want to move. If the latest root certificates aren't on the VM, security constraints will prevent the data copy to the target region.
 - For Windows VMs, install all the latest Windows updates on the VM, so that all the trusted root certificates are on the machine. In a disconnected environment, follow the standard Windows Update and certificate update processes for your organization.
 - For Linux VMs, follow the guidance provided by your Linux distributor to get the latest trusted root certificates and certificate revocation list on the VM.
2. Make sure that you're not using an authentication proxy to control network connectivity for VMs that you want to move.
3. If the VM that you're trying to move doesn't have access to the internet, or it's using a firewall proxy to control outbound access, [check the requirements](#).
4. Identify the source networking layout and all the resources that you're currently using. This includes but isn't limited to load balancers, network security groups (NSGs), and public IPs.

Prepare the target region

1. Verify that your Azure subscription allows you to create VMs in the target region that's used for disaster recovery. Contact support to enable the required quota.
2. Make sure that your subscription has enough resources to support VMs with sizes that match your source VMs. If you're using Site Recovery to copy data to the target, Site Recovery chooses the same size or the closest possible size for the target VM.
3. Make sure that you create a target resource for every component that's identified in the source networking layout. This step is important to ensure that your VMs have all the functionality and features in the target region that you had in the source region.

NOTE

Azure Site Recovery automatically discovers and creates a virtual network when you enable replication for the source VM. You can also pre-create a network and assign it to the VM in the user flow for enable replication. As mentioned later, you need to manually create any other resources in the target region.

To create the most commonly used network resources that are relevant for you based on the source VM configuration, see the following documentation:

- [Network security groups](#)
- [Load balancers](#)
- [Public IP](#)

For any other networking components, see the [networking documentation](#).

4. Manually [create a non-production network](#) in the target region if you want to test the configuration before you perform the final move to the target region. We recommend this step because it ensures minimal interference with the production network.

Copy data to the target region

The following steps show you how to use Azure Site Recovery to copy data to the target region.

Create the vault in any region, except the source region

1. Sign in to the [Azure portal](#) > **Recovery Services**.
2. Select **Create a resource** > **Management Tools** > **Backup and Site Recovery**.
3. In **Name**, specify the friendly name **ContosoVMVault**. If you have more than one subscription, select the appropriate one.
4. Create the resource group **ContosoRG**.
5. Specify an Azure region. To check supported regions, see geographic availability in [Azure Site Recovery pricing details](#).
6. In **Recovery Services vaults**, select **Overview** > **ContosoVMVault** > **+Replicate**.
7. In **Source**, select **Azure**.
8. In **Source location**, select the source Azure region where your VMs are currently running.
9. Select the Resource Manager deployment model. Then select the **Source subscription** and **Source resource group**.
10. Select **OK** to save the settings.

Enable replication for Azure VMs and start copying the data

Site Recovery retrieves a list of the VMs that are associated with the subscription and resource group.

1. In the next step, select the VM that you want to move, then select **OK**.
2. In **Settings**, select **Disaster recovery**.
3. In **Configure disaster recovery** > **Target region**, select the target region to which you'll replicate.
4. For this tutorial, accept the other default settings.
5. Select **Enable replication**. This step starts a job to enable replication for the VM.

Configure settings

Resource group, Network, Storage and Availability sets [Customize](#)

By default Azure Site Recovery(ASR) will mirror the source site configuration to target site by creating/using the required resource groups, storage accounts, virtual network and availability sets as below. Click 'Customize' above to change the configuration. The resources created by ASR are appended with "asr" suffix.

Target resource group 	Target virtual network
ContosoRG	A2ATest2-vnet-asr(new)
Cache storage accounts 	Target storage accounts
a2atest2disks86cacheasr(new)	a2atest2disks864asr(new)
Target availability sets 	
Replication Policy 	
Name: 24-hour-retention-policy	
Recovery point retention: 24 hour(s)	
App consistent snapshot frequency: 4 hour(s)	

Test the configuration

1. Go to the vault. In **Settings > Replicated items**, select the VM that you want to move to the target region, and then select **+Test Failover**.
2. In **Test Failover**, select a recovery point to use for the failover:
 - **Latest processed**: Fails the VM over to the latest recovery point that was processed by the Site Recovery service. The time stamp is shown. With this option, no time is spent processing data, so it provides a low Recovery Time Objective (RTO).
 - **Latest app-consistent**: This option fails over all VMs to the latest app-consistent recovery point. The time stamp is shown.
 - **Custom**: Select any recovery point.
3. Select the target Azure virtual network to which you want to move the Azure VMs to test the configuration.

IMPORTANT

We recommend that you use a separate Azure VM network for the test failover. Don't use the production network that was set up when you enabled replication and that you want to move your VMs into eventually.

4. To start testing the move, click **OK**. To track progress, click the VM to open its properties. Or, you can click the **Test Failover** job in the vault name > **Settings > Jobs > Site Recovery jobs**.
5. After the failover finishes, the replica Azure VM appears in the Azure portal > **Virtual Machines**. Make sure that the VM is running, is sized appropriately, and is connected to the appropriate network.
6. If you want to delete the VM that was created as part of testing the move, click **Cleanup test failover** on the replicated item. In **Notes**, record and save any observations that are associated with the test.

Perform the move to the target region and confirm

1. Go to the vault. In **Settings > Replicated items**, select the VM, and then select **Failover**.
2. In **Failover**, select **Latest**.
3. Select **Shut down machine before beginning failover**. Site Recovery attempts to shut down the source VM before triggering the failover. Failover continues even if shutdown fails. You can follow the failover progress on the **Jobs** page.
4. After the job is finished, check that the VM appears in the target Azure region as expected.
5. In **Replicated items**, right-select the VM > **Commit**. This step finishes the move process to the target region. Wait until the commit job finishes.

Delete the resource in the source region

Go to the VM. Select **Disable Replication**. This step stops the process from copying the data for the VM.

IMPORTANT

It's important to perform this step to avoid being charged for Azure Site Recovery replication.

If you have no plans to reuse any of the source resources, follow these steps:

1. Delete all the relevant network resources in the source region that you listed out as part of step 4 in [Prepare the source VMs](#).
2. Delete the corresponding storage account in the source region.

Next steps

In this tutorial, you moved an Azure VM to a different Azure region. Now you can configure disaster recovery for the VM that you moved.

[Set up disaster recovery after migration](#)

Move Azure VMs into Availability Zones

6/6/2019 • 7 minutes to read • [Edit Online](#)

Availability Zones in Azure help protect your applications and data from datacenter failures. Each Availability 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. The physical separation of Availability Zones within a region helps protect applications and data from datacenter failures. With Availability Zones, Azure offers a service-level agreement (SLA) of 99.99% for uptime of virtual machines (VMs). Availability Zones are supported in select regions, as mentioned in [What are Availability Zones in Azure?](#).

In a scenario where your VMs are deployed as *single instance* into a specific region, and you want to improve your availability by moving these VMs into an Availability Zone, you can do so by using Azure Site Recovery. This action can further be categorized into:

- Move single-instance VMs into Availability Zones in a target region
- Move VMs in an availability set into Availability Zones in a target region

IMPORTANT

Currently, Azure Site Recovery supports moving VMs from one region to another but doesn't support moving within a region.

Check prerequisites

- Check whether the target region has [support for Availability Zones](#). Check that your choice of [source region/target region combination is supported](#). Make an informed decision on the target region.
- Make sure that you understand the [scenario architecture and components](#).
- Review the [support limitations and requirements](#).
- Check account permissions. If you just created your free Azure account, you're the admin of your subscription. If you aren't the subscription admin, work with the admin to assign the permissions you need. To enable replication for a VM and eventually copy data to the target by using Azure Site Recovery, you must have:
 1. Permission to create a VM in Azure resources. The *Virtual Machine Contributor* built-in role has these permissions, which include:
 - Permission to create a VM in the selected resource group
 - Permission to create a VM in the selected virtual network
 - Permission to write to the selected storage account
 2. Permission to manage Azure Site Recovery tasks. The *Site Recovery Contributor* role has all permissions required to manage Site Recovery actions in a Recovery Services vault.

Prepare the source VMs

1. Your VMs should use managed disks if you want to move them to an Availability Zone by using Site Recovery. You can convert existing Windows VMs that use unmanaged disks to use managed disks. Follow the steps at [Convert a Windows virtual machine from unmanaged disks to managed disks](#). Ensure that the availability set is configured as *managed*.

2. Check that all the latest root certificates are present on the Azure VMs you want to move. If the latest root certificates aren't present, the data copy to the target region can't be enabled because of security constraints.
3. For Windows VMs, install all the latest Windows updates on the VM, so that all the trusted root certificates are on the machine. In a disconnected environment, follow the standard Windows update and certificate update processes for your organization.
4. For Linux VMs, follow the guidance provided by your Linux distributor to get the latest trusted root certificates and certificate revocation list on the VM.
5. Make sure you don't use an authentication proxy to control network connectivity for VMs that you want to move.
6. If the VM you're trying to move doesn't have access to the internet and uses a firewall proxy to control outbound access, check the requirements at [Configure outbound network connectivity](#).
7. Identify the source networking layout and the resources you currently use for verification, including load balancers, NSGs, and public IP.

Prepare the target region

1. Check that your Azure subscription lets you create VMs in the target region used for disaster recovery. If necessary, contact support to enable the required quota.
2. Make sure your subscription has enough resources to support VMs with sizes that match your source VMs. If you use Site Recovery to copy data to the target, it picks the same size or the closest possible size for the target VM.
3. Create a target resource for every component identified in the source networking layout. This action ensures that after you cut over to the target region, your VMs have all the functionality and features that you had in the source.

NOTE

Azure Site Recovery automatically discovers and creates a virtual network and storage account when you enable replication for the source VM. You can also pre-create these resources and assign to the VM as part of the enable replication step. But for any other resources, as mentioned later, you need to manually create them in the target region.

The following documents tell how to create the most commonly used network resources that are relevant to you, based on the source VM configuration.

- [Network security groups](#)
- [Load balancers](#)
- [Public IP](#)

For any other networking components, refer to the networking [documentation](#).

IMPORTANT

Ensure that you use a zone-redundant load balancer in the target. You can read more at [Standard Load Balancer and Availability Zones](#).

4. Manually [create a non-production network](#) in the target region if you want to test the configuration before you cut over to the target region. We recommend this approach because it causes minimal interference with the production environment.

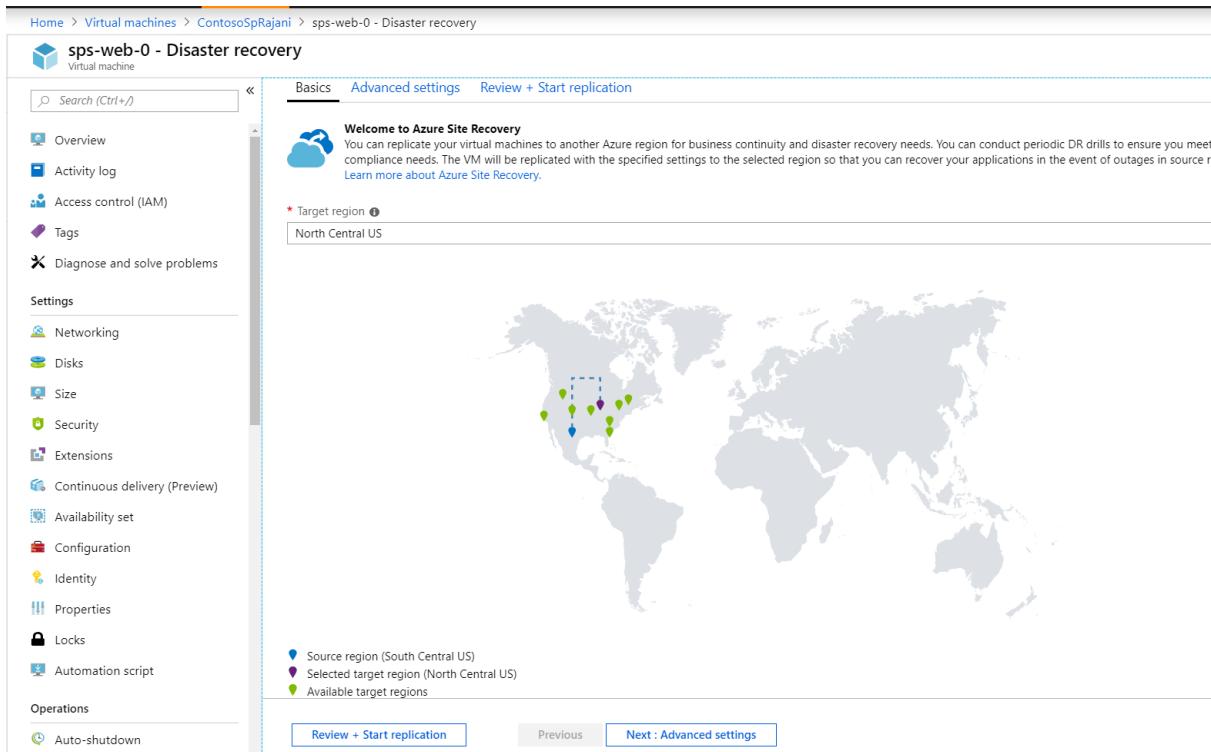
Enable replication

The following steps will guide you when using Azure Site Recovery to enable replication of data to the target region, before you eventually move them into Availability Zones.

NOTE

These steps are for a single VM. You can extend the same to multiple VMs. Go to the Recovery Services vault, select **+ Replicate**, and select the relevant VMs together.

1. In the Azure portal, select **Virtual machines**, and select the VM you want to move into Availability Zones.
2. In **Operations**, select **Disaster recovery**.
3. In **Configure disaster recovery > Target region**, select the target region to which you'll replicate. Ensure this region [supports](#) Availability Zones.



4. Select **Next: Advanced settings**.
5. Choose the appropriate values for the target subscription, target VM resource group, and virtual network.
6. In the **Availability** section, choose the Availability Zone into which you want to move the VM.

NOTE

If you don't see the option for availability set or Availability Zone, ensure that the [prerequisites](#) are met and the [preparation](#) of source VMs is complete.

The screenshot shows the 'Review + Start replication' step of the Site Recovery wizard. Under 'Target settings', the 'Availability' section is selected. It shows the source subscription ('ASR PM team subscription') and target subscription ('(new) CONTOSOPRAJANI-asr'). The 'Availability' dropdown is expanded, showing 'Single instance', 'Availability set', and 'Availability zone'. 'Availability zone' is selected, and a list below shows three zones: 1, 2, and 3.

7. Select **Enable Replication**. This action starts a job to enable replication for the VM.

Check settings

After the replication job has finished, you can check the replication status, modify replication settings, and test the deployment.

1. In the VM menu, select **Disaster recovery**.
2. You can check replication health, the recovery points that have been created and the source, and target regions on the map.

The screenshot shows the Azure portal's 'Disaster recovery' blade for the 'BCDR-Demo-VM1' virtual machine. The 'Health' section indicates 'Healthy' status. The 'Events' section shows 0 events. The 'Latest Recovery Points' table lists 'Crash-consistent' and 'App-consistent' points. On the right, a world map displays the replication path from the source location in 'East US' to the target location in 'South Central US'.

Test the configuration

1. In the virtual machine menu, select **Disaster recovery**.

2. Select the **Test Failover** icon.
3. In **Test Failover**, select a recovery point to use for the failover:
 - **Latest processed**: Fails the VM over to the latest recovery point that was processed by the Site Recovery service. The time stamp is shown. With this option, no time is spent processing data, so it provides a low recovery time objective (RTO).
 - **Latest app-consistent**: This option fails over all VMs to the latest app-consistent recovery point. The time stamp is shown.
 - **Custom**: Select any recovery point.
4. Select the test target Azure virtual network to which you want to move the Azure VMs to test the configuration.

IMPORTANT

We recommend that you use a separate Azure VM network for the test failure, and not the production network in the target region into which you want to move your VMs.

5. To start testing the move, select **OK**. To track progress, select the VM to open its properties. Or, you can select the **Test Failover** job in the vault name > **Settings** > **Jobs** > **Site Recovery jobs**.
6. After the failover finishes, the replica Azure VM appears in the Azure portal > **Virtual Machines**. Make sure that the VM is running, sized appropriately, and connected to the appropriate network.
7. If you want to delete the VM created as part of testing the move, select **Cleanup test failover** on the replicated item. In **Notes**, record and save any observations associated with the test.

Move to the target region and confirm

1. In the virtual machine menu, select **Disaster recovery**.
2. Select the **Failover** icon.
3. In **Failover**, select **Latest**.
4. Select **Shut down machine before beginning failover**. Site Recovery attempts to shut down the source VM before triggering the failover. Failover continues even if shutdown fails. You can follow the failover progress on the **Jobs** page.
5. After the job is finished, check that the VM appears in the target Azure region as expected.
6. In **Replicated items**, right-click the VM > **Commit**. This finishes the move process to the target region. Wait until the commit job is finished.

Discard the resource in the source region

Go to the VM. Select **Disable Replication**. This action stops the process of copying the data for the VM.

IMPORTANT

Do the preceding step to avoid getting charged for Site Recovery replication after the move. The source replication settings are cleaned up automatically. Note that the Site Recovery extension that is installed as part of the replication isn't removed and needs to be removed manually.

Next steps

In this tutorial, you increased the availability of an Azure VM by moving into an availability set or Availability Zone. Now you can set disaster recovery for the moved VM.

[Set up disaster recovery after migration](#)

Migrate from Amazon Web Services (AWS) and other platforms to Managed Disks in Azure

2/15/2019 • 4 minutes to read • [Edit Online](#)

You can upload VHD files from AWS or on-premises virtualization solutions to Azure to create VMs that take advantage of Managed Disks. Azure Managed Disks removes the need to manage storage accounts for Azure IaaS VMs. You have to only specify the type (Premium or Standard) and size of disk you need, and Azure creates and manages the disk for you.

You can upload either generalized and specialized VHDs.

- **Generalized VHD** - has had all of your personal account information removed using Sysprep.
- **Specialized VHD** - maintains the user accounts, applications, and other state data from your original VM.

IMPORTANT

Before uploading any VHD to Azure, you should follow [Prepare a Windows VHD or VHDX to upload to Azure](#)

SCENARIO	DOCUMENTATION
You have existing AWS EC2 instances that you would like to migrate to Azure VMs using managed disks	Move a VM from Amazon Web Services (AWS) to Azure
You have a VM from another virtualization platform that you would like to use as an image to create multiple Azure VMs.	Upload a generalized VHD and use it to create a new VM in Azure
You have a uniquely customized VM that you would like to recreate in Azure.	Upload a specialized VHD to Azure and create a new VM

Overview of Managed Disks

Azure Managed Disks simplifies VM management by removing the need to manage storage accounts. Managed Disks also benefit from better reliability of VMs in an Availability Set. It ensures that the disks of different VMs in an Availability Set are sufficiently isolated from each other to avoid a single point of failure. It automatically places disks of different VMs in an Availability Set in different Storage scale units (stamps) which limits the impact of single Storage scale unit failures caused due to hardware and software failures. Based on your needs, you can choose from four types of storage options. To learn about the available disk types, see our article [Select a disk type](#).

Plan for the migration to Managed Disks

This section helps you to make the best decision on VM and disk types.

If you are planning on migrating from unmanaged disks to managed disks, you should be aware that users with the [Virtual Machine Contributor](#) role will not be able to change the VM size (as they could pre-conversion). This is because VMs with managed disks require the user to have the Microsoft.Compute/disks/write permission on the OS disks.

Location

Pick a location where Azure Managed Disks are available. If you are migrating to Premium Managed Disks, also

ensure that Premium storage is available in the region where you are planning to migrate to. See [Azure Services by Region](#) for up-to-date information on available locations.

VM sizes

If you are migrating to Premium Managed Disks, you have to update the size of the VM to Premium Storage capable size available in the region where VM is located. Review the VM sizes that are Premium Storage capable. The Azure VM size specifications are listed in [Sizes for virtual machines](#). Review the performance characteristics of virtual machines that work with Premium Storage and choose the most appropriate VM size that best suits your workload. Make sure that there is sufficient bandwidth available on your VM to drive the disk traffic.

Disk sizes

Premium Managed Disks

There are seven types of premium managed disks that can be used with your VM and each has specific IOPs and throughput limits. Take into consideration these limits when choosing the Premium disk type for your VM based on the needs of your application in terms of capacity, performance, scalability, and peak loads.

PREMIUM DISKS TYPE	P4	P6	P10	P15	P20	P30	P40	P50
Disk size	32 GB	64 GB	128 GB	256 GB	512 GB	1024 GB (1 TB)	2048 GB (2 TB)	4095 GB (4 TB)
IOPS per disk	120	240	500	1100	2300	5000	7500	7500
Throughput per disk	25 MB per second	50 MB per second	100 MB per second	125 MB per second	150 MB per second	200 MB per second	250 MB per second	250 MB per second

Standard Managed Disks

There are seven types of standard managed disks that can be used with your VM. Each of them have different capacity but have same IOPS and throughput limits. Choose the type of Standard Managed disks based on the capacity needs of your application.

STANDARD DISK TYPE	S4	S6	S10	S15	S20	S30	S40	S50
Disk size	30 GB	64 GB	128 GB	256 GB	512 GB	1024 GB (1 TB)	2048 GB (2TB)	4095 GB (4 TB)
IOPS per disk	500	500	500	500	500	500	500	500
Throughput per disk	60 MB per second							

Disk caching policy

Premium Managed Disks

By default, disk caching policy is *Read-Only* for all the Premium data disks, and *Read-Write* for the Premium operating system disk attached to the VM. This configuration setting is recommended to achieve the optimal performance for your application's IOs. For write-heavy or write-only data disks (such as SQL Server log files),

disable disk caching so that you can achieve better application performance.

Pricing

Review the [pricing for Managed Disks](#). Pricing of Premium Managed Disks is same as the Premium Unmanaged Disks. But pricing for Standard Managed Disks is different than Standard Unmanaged Disks.

Next Steps

- Before uploading any VHD to Azure, you should follow [Prepare a Windows VHD or VHDX to upload to Azure](#)

Move a Windows VM from Amazon Web Services (AWS) to an Azure virtual machine

3/27/2019 • 2 minutes to read • [Edit Online](#)

If you are evaluating Azure virtual machines for hosting your workloads, you can export an existing Amazon Web Services (AWS) EC2 Windows VM instance then upload the virtual hard disk (VHD) to Azure. Once the VHD is uploaded, you can create a new VM in Azure from the VHD.

This article covers moving a single VM from AWS to Azure. If you want to move VMs from AWS to Azure at scale, see [Migrate virtual machines in Amazon Web Services \(AWS\) to Azure with Azure Site Recovery](#).

Prepare the VM

You can upload both generalized and specialized VHDs to Azure. Each type requires that you prepare the VM before exporting from AWS.

- **Generalized VHD** - a generalized VHD has had all of your personal account information removed using Sysprep. If you intend to use the VHD as an image to create new VMs from, you should:
 - [Prepare a Windows VM](#).
 - Generalize the virtual machine using Sysprep.
- **Specialized VHD** - a specialized VHD maintains the user accounts, applications and other state data from your original VM. If you intend to use the VHD as-is to create a new VM, ensure the following steps are completed.
 - [Prepare a Windows VHD to upload to Azure](#). **Do not** generalize the VM using Sysprep.
 - Remove any guest virtualization tools and agents that are installed on the VM (i.e. VMware tools).
 - Ensure the VM is configured to pull its IP address and DNS settings via DHCP. This ensures that the server obtains an IP address within the VNet when it starts up.

Export and download the VHD

Export the EC2 instance to a VHD in an Amazon S3 bucket. Follow the steps in the Amazon documentation article [Exporting an Instance as a VM Using VM Import/Export](#) and run the `create-instance-export-task` command to export the EC2 instance to a VHD file.

The exported VHD file is saved in the Amazon S3 bucket you specify. The basic syntax for exporting the VHD is below, just replace the placeholder text in <brackets> with your information.

```
aws ec2 create-instance-export-task --instance-id <instanceID> --target-environment Microsoft \
--export-to-s3-task DiskImageFormat=VHD,ContainerFormat=ova,S3Bucket=<bucket>,S3Prefix=<prefix>
```

Once the VHD has been exported, follow the instructions in [How Do I Download an Object from an S3 Bucket?](#) to download the VHD file from the S3 bucket.

IMPORTANT

AWS charges data transfer fees for downloading the VHD. See [Amazon S3 Pricing](#) for more information.

Next steps

Now you can upload the VHD to Azure and create a new VM.

- If you ran Sysprep on your source to **generalize** it before exporting, see [Upload a generalized VHD and use it to create a new VMs in Azure](#)
- If you did not run Sysprep before exporting, the VHD is considered **specialized**, see [Upload a specialized VHD to Azure and create a new VM](#)

Create a Linux VM from a custom disk with the Azure CLI

3/14/2019 • 6 minutes to read • [Edit Online](#)

This article shows you how to upload a customized virtual hard disk (VHD), and how to copy an existing VHD in Azure. The newly created VHD is then used to create new Linux virtual machines (VMs). You can install and configure a Linux distro to your requirements and then use that VHD to create a new Azure virtual machine.

To create multiple VMs from your customized disk, first create an image from your VM or VHD. For more information, see [Create a custom image of an Azure VM by using the CLI](#).

You have two options to create a custom disk:

- Upload a VHD
- Copy an existing Azure VM

Quick commands

When creating a new VM with `az vm create` from a customized or specialized disk, you **attach** the disk (`--attach-os-disk`) rather than specifying a custom or marketplace image (`--image`). The following example creates a VM named *myVM* using the managed disk named *myManagedDisk* created from your customized VHD:

```
az vm create --resource-group myResourceGroup --location eastus --name myVM \
--os-type linux --attach-os-disk myManagedDisk
```

Requirements

To complete the following steps, you'll need:

- A Linux virtual machine that has been prepared for use in Azure. The [Prepare the VM](#) section of this article covers how to find distro-specific information on installing the Azure Linux Agent (waagent), which is needed for you to connect to a VM with SSH.
- The VHD file from an existing [Azure-endorsed Linux distribution](#) (or see [information for non-endorsed distributions](#)) to a virtual disk in the VHD format. Multiple tools exist to create a VM and VHD:
 - Install and configure [QEMU](#) or [KVM](#), taking care to use VHD as your image format. If needed, you can [convert an image](#) with `qemu-img convert`.
 - You can also use Hyper-V [on Windows 10](#) or [on Windows Server 2012/2012 R2](#).

NOTE

The newer VHDX format is not supported in Azure. When you create a VM, specify VHD as the format. If needed, you can convert VHDX disks to VHD with `qemu-img convert` or the `Convert-VHD` PowerShell cmdlet. Azure does not support uploading dynamic VHDs, so you'll need to convert such disks to static VHDs before uploading. You can use tools such as [Azure VHD Utilities for GO](#) to convert dynamic disks during the process of uploading them to Azure.

- Make sure that you have the latest [Azure CLI](#) installed and you are signed in to an Azure account with `az login`.

In the following examples, replace example parameter names with your own values, such as *myResourceGroup*, *mystorageaccount*, and *mydisks*.

Prepare the VM

Azure supports various Linux distributions (see [Endorsed Distributions](#)). The following articles describe how to prepare the various Linux distributions that are supported on Azure:

- [CentOS-based Distributions](#)
- [Debian Linux](#)
- [Oracle Linux](#)
- [Red Hat Enterprise Linux](#)
- [SLES & openSUSE](#)
- [Ubuntu](#)
- [Others: Non-Endorsed Distributions](#)

Also see the [Linux Installation Notes](#) for more general tips on preparing Linux images for Azure.

NOTE

The [Azure platform SLA](#) applies to VMs running Linux only when one of the endorsed distributions is used with the configuration details as specified under "Supported Versions" in [Linux on Azure-Endorsed Distributions](#).

Option 1: Upload a VHD

You can upload a customized VHD that you have running on a local machine or that you exported from another cloud. To use a VHD to create a new Azure VM, you'll need to upload the VHD to a storage account and create a managed disk from the VHD. For more information, see [Azure Managed Disks overview](#).

Create a resource group

Before uploading your custom disk and creating VMs, you'll need to 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
```

Create a storage account

Create a storage account for your custom disk and VMs with [az storage account create](#). The following example creates a storage account named *mystorageaccount* in the resource group previously created:

```
az storage account create \
    --resource-group myResourceGroup \
    --location eastus \
    --name mystorageaccount \
    --kind Storage \
    --sku Standard_LRS
```

List storage account keys

Azure generates two 512-bit access keys for each storage account. These access keys are used when authenticating to the storage account, such as when carrying out write operations. For more information, see [managing access to storage](#).

You view the access keys with [az storage account keys list](#). For example, to view the access keys for the storage account you created:

```
az storage account keys list \
    --resource-group myResourceGroup \
    --account-name mystorageaccount
```

The output is similar to:

```
info: Executing command storage account keys list
+ Getting storage account keys
data: Name Key
Permissions
data: ---- -----
data: key1 d4XAvZz1GAgWdvh1WfkZ9q4k9bYZkXkuPCJ15NTsQ0eDeowCDAdB80r9zA/tUINApdSGQ94H9zkszYyxpe8erw== Full
data: key2 Ww0T7g4UyYLaBnLYCxIOTVziGAAHvU+wpwuPvK4ZG0CDFwu/mAxS/YYvAQGHocq1w7/3HcalbnfxtFdqoX0w8g== Full
info: storage account keys list command OK
```

Make a note of **key1** as you will use it to interact with your storage account in the next steps.

Create a storage container

In the same way that you create different directories to logically organize your local file system, you'll create containers within a storage account to organize your disks. A storage account can contain many containers. Create a container with [az storage container create](#).

The following example creates a container named *mydisks*:

```
az storage container create \
    --account-name mystorageaccount \
    --name mydisks
```

Upload the VHD

Upload your custom disk with [az storage blob upload](#). You'll upload and store your custom disk as a page blob.

Specify your access key, the container you created in the previous step, and then the path to the custom disk on your local computer:

```
az storage blob upload --account-name mystorageaccount \
    --account-key key1 \
    --container-name mydisks \
    --type page \
    --file /path/to/disk/mydisk.vhd \
    --name myDisk.vhd
```

Uploading the VHD may take a while.

Create a managed disk

Create a managed disk from the VHD with [az disk create](#). The following example creates a managed disk named *myManagedDisk* from the VHD you uploaded to your named storage account and container:

```
az disk create \
    --resource-group myResourceGroup \
    --name myManagedDisk \
    --source https://mystorageaccount.blob.core.windows.net/mydisks/myDisk.vhd
```

Option 2: Copy an existing VM

You can also create a customized VM in Azure and then copy the OS disk and attach it to a new VM to create another copy. This is fine for testing, but if you want to use an existing Azure VM as the model for multiple new VMs, create an *image* instead. For more information about creating an image from an existing Azure VM, see [Create a custom image of an Azure VM by using the CLI](#).

Create a snapshot

This example creates a snapshot of a VM named *myVM* in resource group *myResourceGroup* and creates a snapshot named *osDiskSnapshot*.

```
osDiskId=$(az vm show -g myResourceGroup -n myVM --query "storageProfile.osDisk.managedDisk.id" -o tsv)
az snapshot create \
    -g myResourceGroup \
    --source "$osDiskId" \
    --name osDiskSnapshot
```

Create the managed disk

Create a new managed disk from the snapshot.

Get the ID of the snapshot. In this example, the snapshot is named *osDiskSnapshot* and it is in the *myResourceGroup* resource group.

```
snapshotId=$(az snapshot show --name osDiskSnapshot --resource-group myResourceGroup --query [id] -o tsv)
```

Create the managed disk. In this example, we will create a managed disk named *myManagedDisk* from our snapshot, where the disk is in standard storage and sized at 128 GB.

```
az disk create \
    --resource-group myResourceGroup \
    --name myManagedDisk \
    --sku Standard_LRS \
    --size-gb 128 \
    --source $snapshotId
```

Create the VM

Create your VM with [az vm create](#) and attach (--attach-os-disk) the managed disk as the OS disk. The following example creates a VM named *myNewVM* using the managed disk you created from your uploaded VHD:

```
az vm create \
    --resource-group myResourceGroup \
    --location eastus \
    --name myNewVM \
    --os-type linux \
    --attach-os-disk myManagedDisk
```

You should be able to SSH into the VM with the credentials from the source VM.

Next steps

After you have prepared and uploaded your custom virtual disk, you can read more about [using Resource Manager and templates](#). You may also want to [add a data disk](#) to your new VMs. If you have applications running on your VMs that you need to access, be sure to [open ports and endpoints](#).

2 minutes to read

Platform-supported migration of IaaS resources from classic to Azure Resource Manager

4/9/2018 • 8 minutes to read • [Edit Online](#)

This article describes how to migrate infrastructure as a service (IaaS) resources from the Classic to Resource Manager deployment models and details how to connect resources from the two deployment models that coexist in your subscription by using virtual network site-to-site gateways. You can read more about [Azure Resource Manager features and benefits](#).

Goal for migration

Resource Manager enables deploying complex applications through templates, configures virtual machines by using VM extensions, and incorporates access management and tagging. Azure Resource Manager includes scalable, parallel deployment for virtual machines into availability sets. The new deployment model also provides lifecycle management of compute, network, and storage independently. Finally, there's a focus on enabling security by default with the enforcement of virtual machines in a virtual network.

Almost all the features from the classic deployment model are supported for compute, network, and storage under Azure Resource Manager. To benefit from the new capabilities in Azure Resource Manager, you can migrate existing deployments from the Classic deployment model.

Supported resources for migration

These classic IaaS resources are supported during migration

- Virtual Machines
- Availability Sets
- Cloud Services with Virtual Machines
- Storage Accounts
- Virtual Networks
- VPN Gateways
- Express Route Gateways (*in the same subscription as Virtual Network only*)
- Network Security Groups
- Route Tables
- Reserved IPs

Supported scopes of migration

There are four different ways to complete migration of compute, network, and storage resources:

- [Migration of virtual machines \(NOT in a virtual network\)](#)
- [Migration of virtual machines \(in a virtual network\)](#)
- [Migration of storage accounts](#)
- [Migration of unattached resources](#)

Migration of virtual machines (NOT in a virtual network)

In the Resource Manager deployment model, security is enforced for your applications by default. All VMs need to be in a virtual network in the Resource Manager model. The Azure platform restarts (`Stop`, `Deallocate`, and

Start) the VMs as part of the migration. You have two options for the virtual networks that the Virtual Machines will be migrated to:

- You can request the platform to create a new virtual network and migrate the virtual machine into the new virtual network.
- You can migrate the virtual machine into an existing virtual network in Resource Manager.

NOTE

In this migration scope, both the management-plane operations and the data-plane operations may not be allowed for a period of time during the migration.

Migration of virtual machines (in a virtual network)

For most VM configurations, only the metadata is migrating between the Classic and Resource Manager deployment models. The underlying VMs are running on the same hardware, in the same network, and with the same storage. The management-plane operations may not be allowed for a certain period of time during the migration. However, the data plane continues to work. That is, your applications running on top of VMs (classic) do not incur downtime during the migration.

The following configurations are not currently supported. If support is added in the future, some VMs in this configuration might incur downtime (go through stop, deallocate, and restart VM operations).

- You have more than one availability set in a single cloud service.
- You have one or more availability sets and VMs that are not in an availability set in a single cloud service.

NOTE

In this migration scope, the management plane may not be allowed for a period of time during the migration. For certain configurations as described earlier, data-plane downtime occurs.

Migration of storage accounts

To allow seamless migration, you can deploy Resource Manager VMs in a classic storage account. With this capability, compute and network resources can and should be migrated independently of storage accounts. Once you migrate over your Virtual Machines and Virtual Network, you need to migrate over your storage accounts to complete the migration process.

If your storage account does not have any associated disks or Virtual Machines data and only has blobs, files, tables, and queues then the migration to Azure Resource Manager can be done as a standalone migration without dependencies.

NOTE

The Resource Manager deployment model doesn't have the concept of Classic images and disks. When the storage account is migrated, Classic images and disks are not visible in the Resource Manager stack but the backing VHDs remain in the storage account.

The following screenshots show how to upgrade a Classic storage account to an Azure Resource Manager storage account using Azure portal:

1. Sign in to the [Azure portal](#).
2. Navigate to your storage account.
3. In the **Settings** section, click **Migrate to ARM**.

4. Click on **Validate** to determine migration feasibility.
5. If validation passes, click on **Prepare** to create a migrated storage account.
6. Type **yes** to confirm migration and click **Commit** to finish the migration.

Migrate to ARM

testclassicaccount2 - Migrate to ARM

Storage account (classic)

Search (Ctrl+ /)

- Overview
- Activity log
- Access control (IAM)
- Diagnose and solve problem...
- Storage Explorer (preview)
- Settings
 - Access keys
 - CORS
 - Configuration
 - Shared access signature
 - Migrate to ARM
- Properties
- Locks

Blob service

- Blobs

To benefit from new capabilities in Azure Resource Manager, you can migrate existing deployments from the Classic deployment model. extensions, incorporates role-based access management, and tagging. [Learn more](#)

Take the following steps to complete migration:

1. Validate if the resource is capable of migration
- Validate
2. Prepare
3. Commit or abort

Migrate to ARM

testclassicaccount2 - Migrate to ARM

Storage account (classic)

Search (Ctrl+ /)

- Overview
- Activity log
- Access control (IAM)
- Diagnose and solve problem...
- Storage Explorer (preview)
- Settings
 - Access keys
 - CORS
 - Configuration
 - Shared access signature
 - Migrate to ARM
- Properties
- Locks

Blob service

- Blobs

✓ Validation passed.

To benefit from new capabilities in Azure Resource Manager, you can migrate existing deployments from the Classic deployment model. extensions, incorporates role-based access management, and tagging. [Learn more](#)

Take the following steps to complete migration:

1. Validate if the resource is capable of migration
- ✓ Validation passed.
- 1 storage account will be migrated.
- View details
2. Prepare
- Simulate the transformation of classic resources into Resource Manager resources.
If you see any issues with the results of 'Prepare', you will be able to abort migration in the next step.
- Prepare
3. Commit or abort

Migration of unattached resources

Storage Accounts with no associated disks or Virtual Machines data may be migrated independently.

Network Security Groups, Route Tables & Reserved IPs that are not attached to any Virtual Machines and Virtual Networks can also be migrated independently.

Unsupported features and configurations

Some features and configurations are not currently supported; the following sections describe our recommendations around them.

Unsupported features

The following features are not currently supported. You can optionally remove these settings, migrate the VMs, and then re-enable the settings in the Resource Manager deployment model.

RESOURCE PROVIDER	FEATURE	RECOMMENDATION
Compute	Unassociated virtual machine disks.	The VHD blobs behind these disks will get migrated when the Storage Account is migrated
Compute	Virtual machine images.	The VHD blobs behind these disks will get migrated when the Storage Account is migrated
Network	Endpoint ACLs.	Remove Endpoint ACLs and retry migration.
Network	Application Gateway	Remove the Application Gateway before beginning migration and then recreate the Application Gateway once migration is complete.
Network	Virtual networks using VNet Peering.	Migrate Virtual Network to Resource Manager, then peer. Learn more about VNet Peering .

Unsupported configurations

The following configurations are not currently supported.

Service	Configuration	Recommendation
Resource Manager	Role-Based Access Control (RBAC) for classic resources	Because the URI of the resources is modified after migration, it is recommended that you plan the RBAC policy updates that need to happen after migration.
Compute	Multiple subnets associated with a VM	Update the subnet configuration to reference only one subnet. This may require you to remove a secondary NIC (that is referring to another subnet) from the VM and reattach it after migration has completed.
Compute	Virtual machines that belong to a virtual network but don't have an explicit subnet assigned	You can optionally delete the VM.
Compute	Virtual machines that have alerts, Autoscale policies	The migration goes through and these settings are dropped. It is highly recommended that you evaluate your environment before you do the migration. Alternatively, you can reconfigure the alert settings after migration is complete.
Compute	XML VM extensions (BGInfo 1.* , Visual Studio Debugger, Web Deploy, and Remote Debugging)	This is not supported. It is recommended that you remove these extensions from the virtual machine to continue migration or they will be dropped automatically during the migration process.
Compute	Boot diagnostics with Premium storage	Disable Boot Diagnostics feature for the VMs before continuing with migration. You can re-enable boot diagnostics in the Resource Manager stack after the migration is complete. Additionally, blobs that are being used for screenshot and serial logs should be deleted so you are no longer charged for those blobs.
Compute	Cloud services that contain web/worker roles	This is currently not supported.
Compute	Cloud services that contain more than one availability set or multiple availability sets.	This is currently not supported. Please move the Virtual Machines to the same availability set before migrating.

Service	Configuration	Recommendation
Compute	VM with Azure Security Center extension	Azure Security Center automatically installs extensions on your Virtual Machines to monitor their security and raise alerts. These extensions usually get installed automatically if the Azure Security Center policy is enabled on the subscription. To migrate the Virtual Machines, disable the security center policy on the subscription, which will remove the Security Center monitoring extension from the Virtual Machines.
Compute	VM with backup or snapshot extension	These extensions are installed on a Virtual Machine configured with the Azure Backup service. While the migration of these VMs is not supported, follow the guidance here to keep backups that were taken prior to migration.
Network	Virtual networks that contain virtual machines and web/worker roles	This is currently not supported. Please move the Web/Worker roles to their own Virtual Network before migrating. Once the classic Virtual Network is migrated, the migrated Azure Resource Manager Virtual Network can be peered with the classic Virtual Network to achieve similar configuration as before.
Network	Classic Express Route circuits	This is currently not supported. These circuits need to be migrated to Azure Resource Manager before beginning IaaS migration. To learn more, see Moving ExpressRoute circuits from the classic to the Resource Manager deployment model .
Azure App Service	Virtual networks that contain App Service environments	This is currently not supported.
Azure HDInsight	Virtual networks that contain HDInsight services	This is currently not supported.
Microsoft Dynamics Lifecycle Services	Virtual networks that contain virtual machines that are managed by Dynamics Lifecycle Services	This is currently not supported.
Azure AD Domain Services	Virtual networks that contain Azure AD Domain services	This is currently not supported.
Azure API Management	Virtual networks that contain Azure API Management deployments	This is currently not supported. To migrate the IaaS VNET, change the VNET of the API Management deployment, which is a no downtime operation.

Next steps

- Technical deep dive on platform-supported migration from classic to Azure Resource Manager
- Planning for migration of IaaS resources from classic to Azure Resource Manager
- Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager
- Use CLI to migrate IaaS resources from classic to Azure Resource Manager
- Community tools for assisting with migration of IaaS resources from classic to Azure Resource Manager
- Review most common migration errors
- Review the most frequently asked questions about migrating IaaS resources from classic to Azure Resource Manager

Technical deep dive on platform-supported migration from classic to Azure Resource Manager

4/9/2018 • 13 minutes to read • [Edit Online](#)

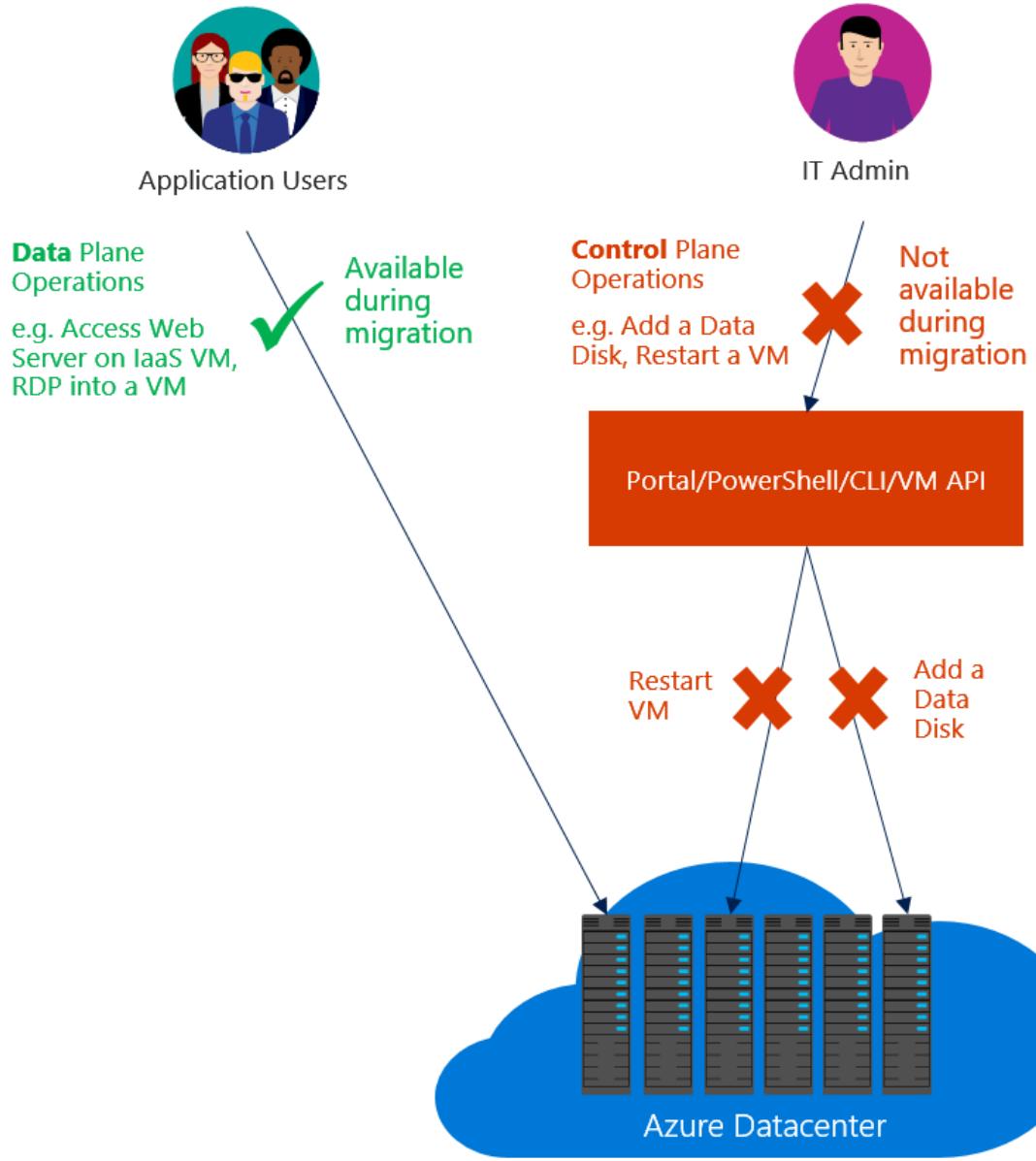
Let's take a deep-dive on migrating from the Azure classic deployment model to the Azure Resource Manager deployment model. We look at resources at a resource and feature level to help you understand how the Azure platform migrates resources between the two deployment models. For more information, please read the service announcement article: [Platform-supported migration of IaaS resources from classic to Azure Resource Manager](#).

Migrate IaaS resources from the classic deployment model to Azure Resource Manager

First, it's important to understand the difference between data-plane and management-plane operations on the infrastructure as a service (IaaS) resources.

- *Management/control plane* describes the calls that come into the management/control plane or the API for modifying resources. For example, operations like creating a VM, restarting a VM, and updating a virtual network with a new subnet manage the running resources. They don't directly affect connecting to the VMs.
- *Data plane* (application) describes the runtime of the application itself, and involves interaction with instances that don't go through the Azure API. For example, accessing your website, or pulling data from a running SQL Server instance or a MongoDB server, are data plane or application interactions. Other examples include copying a blob from a storage account, and accessing a public IP address to use Remote Desktop Protocol (RDP) or Secure Shell (SSH) into the virtual machine. These operations keep the application running across compute, networking, and storage.

The data plane is the same between the classic deployment model and Resource Manager stacks. The difference is that during the migration process, Microsoft translates the representation of the resources from the classic deployment model to that in the Resource Manager stack. As a result, you need to use new tools, APIs, and SDKs to manage your resources in the Resource Manager stack.



NOTE

In some migration scenarios, the Azure platform stops, deallocates, and restarts your virtual machines. This causes a brief data-plane downtime.

The migration experience

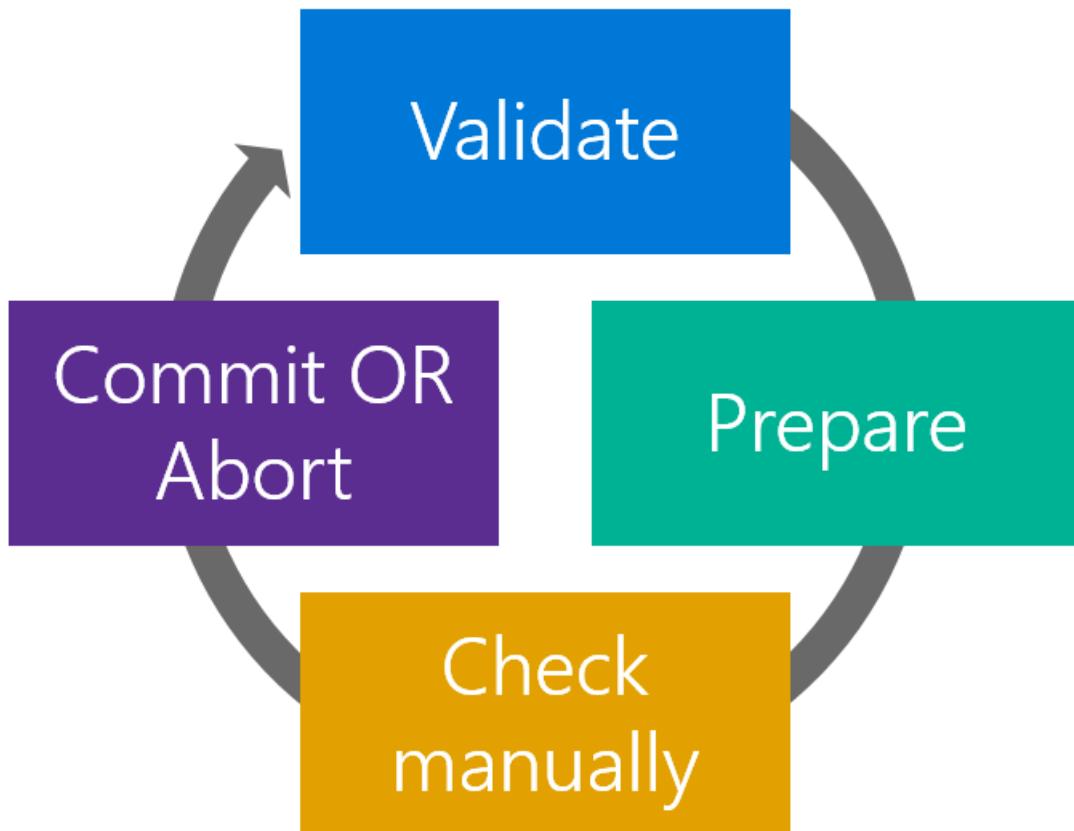
Before you start the migration:

- Ensure that the resources that you want to migrate don't use any unsupported features or configurations. Usually the platform detects these issues and generates an error.
- If you have VMs that are not in a virtual network, they are stopped and deallocated as part of the prepare operation. If you don't want to lose the public IP address, consider reserving the IP address before triggering the prepare operation. If the VMs are in a virtual network, they are not stopped and deallocated.
- Plan your migration during non-business hours to accommodate for any unexpected failures that might happen during migration.
- Download the current configuration of your VMs by using PowerShell, command-line interface (CLI) commands, or REST APIs to make it easier for validation after the prepare step is complete.
- Update your automation and operationalization scripts to handle the Resource Manager deployment model,

before you start the migration. You can optionally do GET operations when the resources are in the prepared state.

- Evaluate the Role-Based Access Control (RBAC) policies that are configured on the IaaS resources in the classic deployment model, and plan for after the migration is complete.

The migration workflow is as follows:



NOTE

The operations described in the following sections are all idempotent. If you have a problem other than an unsupported feature or a configuration error, retry the prepare, abort, or commit operation. Azure tries the action again.

Validate

The validate operation is the first step in the migration process. The goal of this step is to analyze the state of the resources you want to migrate in the classic deployment model. The operation evaluates whether the resources are capable of migration (success or failure).

You select the virtual network or a cloud service (if it's not in a virtual network) that you want to validate for migration. If the resource is not capable of migration, Azure lists the reasons why.

Checks not done in the validate operation

The validate operation only analyzes the state of the resources in the classic deployment model. It can check for all failures and unsupported scenarios due to various configurations in the classic deployment model. It is not possible to check for all issues that the Azure Resource Manager stack might impose on the resources during migration. These issues are only checked when the resources undergo transformation in the next step of migration (the prepare operation). The following table lists all the issues not checked in the validate operation:

NETWORKING CHECKS NOT IN THE VALIDATE OPERATION

A virtual network having both ER and VPN gateways.

A virtual network gateway connection in a disconnected state.

All ER circuits are pre-migrated to Azure Resource Manager stack.

Azure Resource Manager quota checks for networking resources. For example: static public IP, dynamic public IPs, load balancer, network security groups, route tables, and network interfaces.

All load balancer rules are valid across deployment and the virtual network.

Conflicting private IPs between stop-deallocated VMs in the same virtual network.

Prepare

The prepare operation is the second step in the migration process. The goal of this step is to simulate the transformation of the IaaS resources from the classic deployment model to Resource Manager resources. Further, the prepare operation presents this side-by-side for you to visualize.

NOTE

Your resources in the classic deployment model are not modified during this step. It's a safe step to run if you're trying out migration.

You select the virtual network or the cloud service (if it's not a virtual network) that you want to prepare for migration.

- If the resource is not capable of migration, Azure stops the migration process and lists the reason why the prepare operation failed.
- If the resource is capable of migration, Azure locks down the management-plane operations for the resources under migration. For example, you are not able to add a data disk to a VM under migration.

Azure then starts the migration of metadata from the classic deployment model to Resource Manager for the migrating resources.

After the prepare operation is complete, you have the option of visualizing the resources in both the classic deployment model and Resource Manager. For every cloud service in the classic deployment model, the Azure platform creates a resource group name that has the pattern `cloud-service-name>-Migrated`.

NOTE

It is not possible to select the name of a resource group created for migrated resources (that is, "-Migrated"). After migration is complete, however, you can use the move feature of Azure Resource Manager to move resources to any resource group you want. For more information, see [Move resources to new resource group or subscription](#).

The following two screenshots show the result after a successful prepare operation. The first one shows a resource group that contains the original cloud service. The second one shows the new "-Migrated" resource group that contains the equivalent Azure Resource Manager resources.

portalmigrate
Resource group

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

SETTINGS

Quickstart

Resource costs

Deployments

Properties

Locks

Automation script

Essentials

Subscription name (change) Subscription ID

Deployments Location
No deployments East US

Filter by name...

2 items

NAME	TYPE	LOCATION
portalmigrate	Cloud service (class...)	East US
portalmigrate	Virtual machine (cl...)	East US

portalmigrate-Migrated
Resource group

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

SETTINGS

Quickstart

Resource costs

Deployments

Properties

Locks

Automation script

Essentials

Subscription name (change) Subscription ID

Deployments Location
2 Succeeded East US

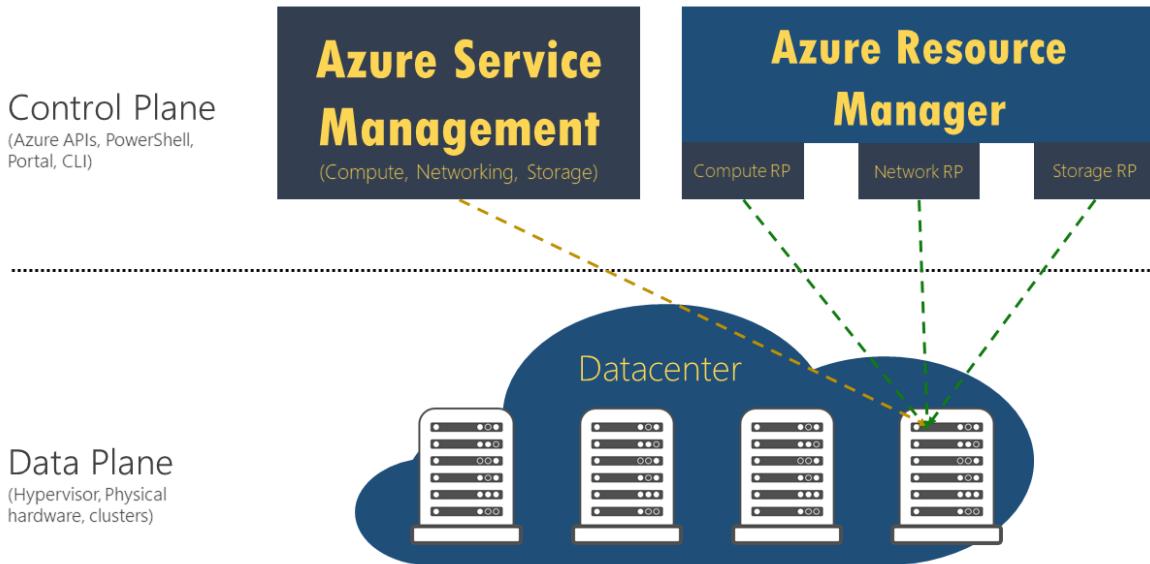
Filter by name...

5 items

NAME	TYPE	LOCATION
portalmigrate	Virtual machine	East US
portalmigrate-PrimaryNic	Network interface	East US
portalmigrate-PrimaryVirtualIP	Public IP address	East US
portalmigrate-PublicLoadBalancer	Load balancer	East US
portalmigrate-VirtualNetwork	Virtual network	East US

Here is a behind-the-scenes look at your resources after the completion of the prepare phase. Note that the resource in the data plane is the same. It's represented in both the management plane (classic deployment model) and the control plane (Resource Manager).

Prepare



NOTE

VMs that are not in a virtual network in the classic deployment model are stopped and deallocated in this phase of migration.

Check (manual or scripted)

In the check step, you have the option to use the configuration that you downloaded earlier to validate that the migration looks correct. Alternatively, you can sign in to the portal, and spot check the properties and resources to validate that metadata migration looks good.

If you are migrating a virtual network, most configuration of virtual machines is not restarted. For applications on those VMs, you can validate that the application is still running.

You can test your monitoring and operational scripts to see if the VMs are working as expected, and if your updated scripts work correctly. Only GET operations are supported when the resources are in the prepared state.

There is no set window of time before which you need to commit the migration. You can take as much time as you want in this state. However, the management plane is locked for these resources until you either abort or commit.

If you see any issues, you can always abort the migration and go back to the classic deployment model. After you go back, Azure opens the management-plane operations on the resources, so that you can resume normal operations on those VMs in the classic deployment model.

Abort

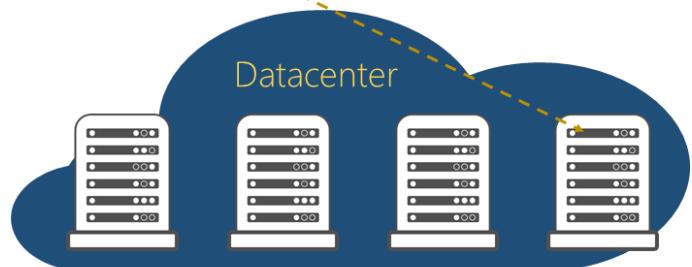
This is an optional step if you want to revert your changes to the classic deployment model and stop the migration. This operation deletes the Resource Manager metadata (created in the prepare step) for your resources.

Abort

Control Plane
(Azure APIs, PowerShell, Portal, CLI)



Data Plane
(Hypervisor, Physical hardware, clusters)



NOTE

This operation can't be done after you have triggered the commit operation.

Commit

After you finish the validation, you can commit the migration. Resources do not appear anymore in the classic deployment model, and are available only in the Resource Manager deployment model. The migrated resources can be managed only in the new portal.

NOTE

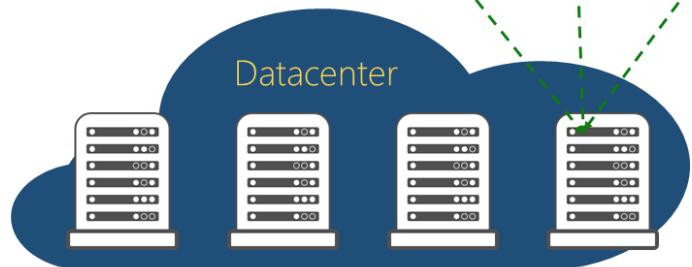
This is an idempotent operation. If it fails, retry the operation. If it continues to fail, create a support ticket or create a forum post with a "ClassiclaaSMigration" tag on our [VM forum](#).

Commit

Control Plane
(Azure APIs, PowerShell, Portal, CLI)



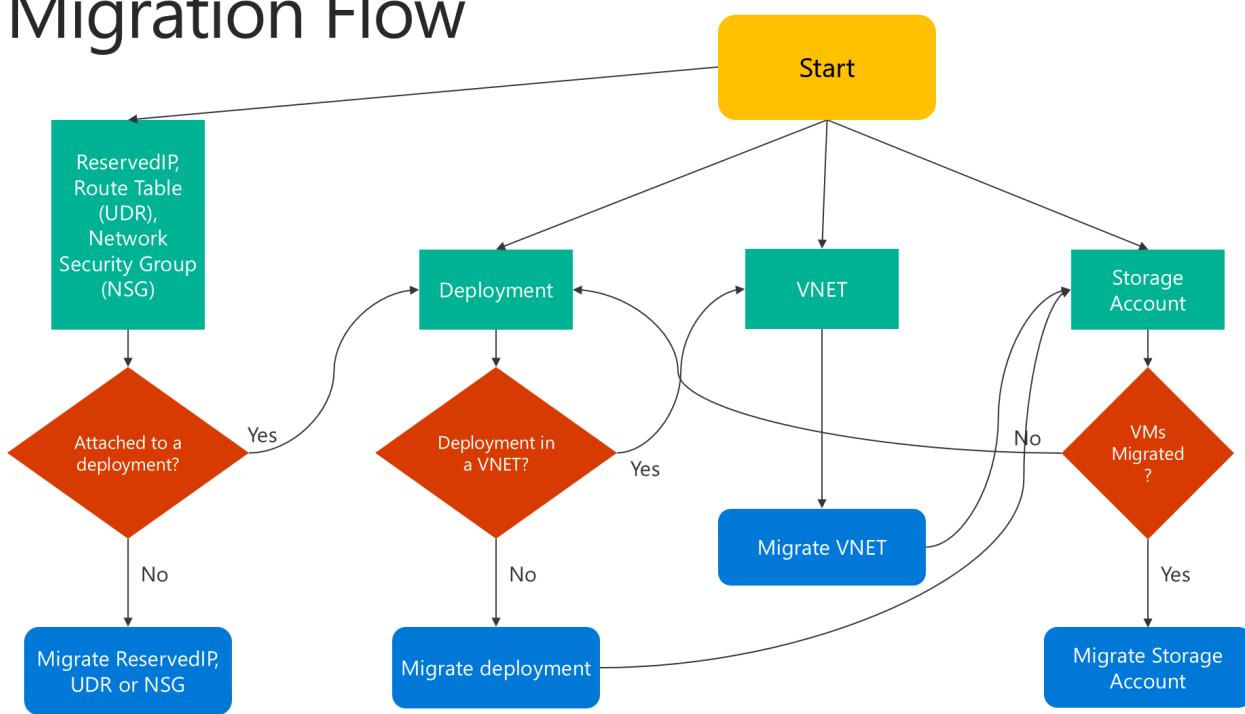
Data Plane
(Hypervisor, Physical hardware, clusters)



Migration flowchart

Here is a flowchart that shows how to proceed with migration:

Migration Flow



Translation of the classic deployment model to Resource Manager resources

You can find the classic deployment model and Resource Manager representations of the resources in the following table. Other features and resources are not currently supported.

CLASSIC REPRESENTATION	RESOURCE MANAGER REPRESENTATION	NOTES
Cloud service name	DNS name	During migration, a new resource group is created for every cloud service with the naming pattern <code><cloudservicename>-migrated</code> . This resource group contains all your resources. The cloud service name becomes a DNS name that is associated with the public IP address.
Virtual machine	Virtual machine	VM-specific properties are migrated unchanged. Certain osProfile information, like computer name, is not stored in the classic deployment model, and remains empty after migration.

CLASSIC REPRESENTATION	RESOURCE MANAGER REPRESENTATION	NOTES
Disk resources attached to VM	Implicit disks attached to VM	Disks are not modeled as top-level resources in the Resource Manager deployment model. They are migrated as implicit disks under the VM. Only disks that are attached to a VM are currently supported. Resource Manager VMs can now use storage accounts in the classic deployment model, which allows the disks to be easily migrated without any updates.
VM extensions	VM extensions	All the resource extensions, except XML extensions, are migrated from the classic deployment model.
Virtual machine certificates	Certificates in Azure Key Vault	If a cloud service contains service certificates, the migration creates a new Azure key vault per cloud service, and moves the certificates into the key vault. The VMs are updated to reference the certificates from the key vault. Do not delete the key vault. This can cause the VM to go into a failed state.
WinRM configuration	WinRM configuration under osProfile	Windows Remote Management configuration is moved unchanged, as part of the migration.
Availability-set property	Availability-set resource	Availability-set specification is a property on the VM in the classic deployment model. Availability sets become a top-level resource as part of the migration. The following configurations are not supported: multiple availability sets per cloud service, or one or more availability sets along with VMs that are not in any availability set in a cloud service.
Network configuration on a VM	Primary network interface	Network configuration on a VM is represented as the primary network interface resource after migration. For VMs that are not in a virtual network, the internal IP address changes during migration.
Multiple network interfaces on a VM	Network interfaces	If a VM has multiple network interfaces associated with it, each network interface becomes a top-level resource as part of the migration, along with all the properties.

CLASSIC REPRESENTATION	RESOURCE MANAGER REPRESENTATION	NOTES
Load-balanced endpoint set	Load balancer	In the classic deployment model, the platform assigned an implicit load balancer for every cloud service. During migration, a new load-balancer resource is created, and the load-balancing endpoint set becomes load-balancer rules.
Inbound NAT rules	Inbound NAT rules	Input endpoints defined on the VM are converted to inbound network address translation rules under the load balancer during the migration.
VIP address	Public IP address with DNS name	The virtual IP address becomes a public IP address, and is associated with the load balancer. A virtual IP can only be migrated if there is an input endpoint assigned to it.
Virtual network	Virtual network	The virtual network is migrated, with all its properties, to the Resource Manager deployment model. A new resource group is created with the name <code>-migrated</code> .
Reserved IPs	Public IP address with static allocation method	Reserved IPs associated with the load balancer are migrated, along with the migration of the cloud service or the virtual machine. Unassociated reserved IP migration is not currently supported.
Public IP address per VM	Public IP address with dynamic allocation method	The public IP address associated with the VM is converted as a public IP address resource, with the allocation method set to static.
NSGs	NSGs	Network security groups associated with a subnet are cloned as part of the migration to the Resource Manager deployment model. The NSG in the classic deployment model is not removed during the migration. However, the management-plane operations for the NSG are blocked when the migration is in progress.
DNS servers	DNS servers	DNS servers associated with a virtual network or the VM are migrated as part of the corresponding resource migration, along with all the properties.

CLASSIC REPRESENTATION	RESOURCE MANAGER REPRESENTATION	NOTES
UDRs	UDRs	User-defined routes associated with a subnet are cloned as part of the migration to the Resource Manager deployment model. The UDR in the classic deployment model is not removed during the migration. The management-plane operations for the UDR are blocked when the migration is in progress.
IP forwarding property on a VM's network configuration	IP forwarding property on the NIC	The IP forwarding property on a VM is converted to a property on the network interface during the migration.
Load balancer with multiple IPs	Load balancer with multiple public IP resources	Every public IP associated with the load balancer is converted to a public IP resource, and associated with the load balancer after migration.
Internal DNS names on the VM	Internal DNS names on the NIC	During migration, the internal DNS suffixes for the VMs are migrated to a read-only property named "InternalDomainNameSuffix" on the NIC. The suffix remains unchanged after migration, and VM resolution should continue to work as previously.
Virtual network gateway	Virtual network gateway	Virtual network gateway properties are migrated unchanged. The VIP associated with the gateway does not change either.
Local network site	Local network gateway	Local network site properties are migrated unchanged to a new resource called a local network gateway. This represents on-premises address prefixes and the remote gateway IP.
Connections references	Connection	Connectivity references between the gateway and the local network site in network configuration is represented by a new resource called Connection. All properties of connectivity reference in network configuration files are copied unchanged to the Connection resource. Connectivity between virtual networks in the classic deployment model is achieved by creating two IPsec tunnels to local network sites representing the virtual networks. This is transformed to the virtual-network-to-virtual-network connection type in the Resource Manager model, without requiring local network gateways.

Changes to your automation and tooling after migration

As part of migrating your resources from the classic deployment model to the Resource Manager deployment

model, you must update your existing automation or tooling to ensure that it continues to work after the migration.

Next steps

- [Overview of platform-supported migration of IaaS resources from classic to Azure Resource Manager](#)
- [Planning for migration of IaaS resources from classic to Azure Resource Manager](#)
- [Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Use CLI to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Community tools for assisting with migration of IaaS resources from classic to Azure Resource Manager](#)
- [Review most common migration errors](#)
- [Review the most frequently asked questions about migrating IaaS resources from classic to Azure Resource Manager](#)

Planning for migration of IaaS resources from classic to Azure Resource Manager

3/15/2019 • 13 minutes to read • [Edit Online](#)

While Azure Resource Manager offers a lot of amazing features, it is critical to plan out your migration journey to make sure things go smoothly. Spending time on planning will ensure that you do not encounter issues while executing migration activities.

NOTE

The following guidance was heavily contributed to by the Azure Customer Advisory team and Cloud Solution architects working with customers on migrating large environments. As such this document will continue to get updated as new patterns of success emerge, so check back from time to time to see if there are any new recommendations.

There are four general phases of the migration journey:



Plan

Technical considerations and tradeoffs

Depending on your technical requirements size, geographies and operational practices, you might want to consider:

1. Why is Azure Resource Manager desired for your organization? What are the business reasons for a migration?
2. What are the technical reasons for Azure Resource Manager? What (if any) additional Azure services would you like to leverage?
3. Which application (or sets of virtual machines) is included in the migration?
4. Which scenarios are supported with the migration API? Review the [unsupported features and configurations](#).
5. Will your operational teams now support applications/VMs in both Classic and Azure Resource Manager?
6. How (if at all) does Azure Resource Manager change your VM deployment, management, monitoring, and reporting processes? Do your deployment scripts need to be updated?
7. What is the communications plan to alert stakeholders (end users, application owners, and infrastructure owners)?
8. Depending on the complexity of the environment, should there be a maintenance period where the application is unavailable to end users and to application owners? If so, for how long?
9. What is the training plan to ensure stakeholders are knowledgeable and proficient in Azure Resource Manager?
10. What is the program management or project management plan for the migration?
11. What are the timelines for the Azure Resource Manager migration and other related technology road maps? Are they optimally aligned?

Patterns of success

Successful customers have detailed plans where the preceding questions are discussed, documented and

governed. Ensure the migration plans are broadly communicated to sponsors and stakeholders. Equip yourself with knowledge about your migration options; reading through this migration document set below is highly recommended.

- Overview of platform-supported migration of IaaS resources from classic to Azure Resource Manager
- Technical deep dive on platform-supported migration from classic to Azure Resource Manager
- Planning for migration of IaaS resources from classic to Azure Resource Manager
- Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager
- Use CLI to migrate IaaS resources from classic to Azure Resource Manager
- Community tools for assisting with migration of IaaS resources from classic to Azure Resource Manager
- Review most common migration errors
- Review the most frequently asked questions about migrating IaaS resources from classic to Azure Resource Manager

Pitfalls to avoid

- Failure to plan. The technology steps of this migration are proven and the outcome is predictable.
- Assumption that the platform supported migration API will account for all scenarios. Read the [unsupported features and configurations](#) to understand what scenarios are supported.
- Not planning potential application outage for end users. Plan enough buffer to adequately warn end users of potentially unavailable application time.

Lab Test

Replicate your environment and do a test migration

NOTE

Exact replication of your existing environment is executed by using a community-contributed tool which is not officially supported by Microsoft Support. Therefore, it is an **optional** step but it is the best way to find out issues without touching your production environments. If using a community-contributed tool is not an option, then read about the Validate/Prepare/Abort Dry Run recommendation below.

Conducting a lab test of your exact scenario (compute, networking, and storage) is the best way to ensure a smooth migration. This will help ensure:

- A wholly separate lab or an existing non-production environment to test. We recommend a wholly separate lab that can be migrated repeatedly and can be destructively modified. Scripts to collect/hydrate metadata from the real subscriptions are listed below.
- It's a good idea to create the lab in a separate subscription. The reason is that the lab will be torn down repeatedly, and having a separate, isolated subscription will reduce the chance that something real will get accidentally deleted.

This can be accomplished by using the AsmMetadataParser tool. [Read more about this tool here](#)

Patterns of success

The following were issues discovered in many of the larger migrations. This is not an exhaustive list and you should refer to the [unsupported features and configurations](#) for more detail. You may or may not encounter these technical issues but if you do solving these before attempting migration will ensure a smoother experience.

- **Do a Validate/Prepare/Abort Dry Run** - This is perhaps the most important step to ensure Classic to Azure Resource Manager migration success. The migration API has three main steps: Validate, Prepare and Commit. Validate will read the state of your classic environment and return a result of all issues. However, because some issues might exist in the Azure Resource Manager stack, Validate will not catch everything.

The next step in migration process, Prepare will help expose those issues. Prepare will move the metadata from Classic to Azure Resource Manager, but will not commit the move, and will not remove or change anything on the Classic side. The dry run involves preparing the migration, then aborting (**not committing**) the migration prepare. The goal of validate/prepare/abort dry run is to see all of the metadata in the Azure Resource Manager stack, examine it (*programmatically or in Portal*), and verify that everything migrates correctly, and work through technical issues. It will also give you a sense of migration duration so you can plan for downtime accordingly. A validate/prepare/abort does not cause any user downtime; therefore, it is non-disruptive to application usage.

- The items below will need to be solved before the dry run, but a dry run test will also safely flush out these preparation steps if they are missed. During enterprise migration, we've found the dry run to be a safe and invaluable way to ensure migration readiness.
- When prepare is running, the control plane (Azure management operations) will be locked for the whole virtual network, so no changes can be made to VM metadata during validate/prepare/abort. But otherwise any application function (RD, VM usage, etc.) will be unaffected. Users of the VMs will not know that the dry run is being executed.
- **Express Route Circuits and VPN.** Currently Express Route Gateways with authorization links cannot be migrated without downtime. For the workaround, see [Migrate ExpressRoute circuits and associated virtual networks from the classic to the Resource Manager deployment model](#).
- **VM Extensions** - Virtual Machine extensions are potentially one of the biggest roadblocks to migrating running VMs. Remediation of VM Extensions could take upwards of 1-2 days, so plan accordingly. A working Azure agent is needed to report back VM Extension status of running VMs. If the status comes back as bad for a running VM, this will halt migration. The agent itself does not need to be in working order to enable migration, but if extensions exist on the VM, then both a working agent AND outbound internet connectivity (with DNS) will be needed for migration to move forward.
 - If connectivity to a DNS server is lost during migration, all VM Extensions except BGInfo v1.* need to first be removed from every VM before migration prepare, and subsequently re-added back to the VM after Azure Resource Manager migration. **This is only for VMs that are running.** If the VMs are stopped deallocated, VM Extensions do not need to be removed. **Note:** Many extensions like Azure diagnostics and security center monitoring will reinstall themselves after migration, so removing them is not a problem.
 - In addition, make sure Network Security Groups are not restricting outbound internet access. This can happen with some Network Security Groups configurations. Outbound internet access (and DNS) is needed for VM Extensions to be migrated to Azure Resource Manager.
 - There are two versions of the BGInfo extension: v1 and v2. If the VM was created using the Azure portal or PowerShell, the VM will likely have the v1 extension on it. This extension does not need to be removed and will be skipped (not migrated) by the migration API. However, if the Classic VM was created with the new Azure portal, it will likely have the JSON-based v2 version of BGInfo, which can be migrated to Azure Resource Manager provided the agent is working and has outbound internet access (and DNS).
 - **Remediation Option 1.** If you know your VMs will not have outbound internet access, a working DNS service, and working Azure agents on the VMs, then uninstall all VM extensions as part of the migration before Prepare, then reinstall the VM Extensions after migration.
 - **Remediation Option 2.** If VM extensions are too big of a hurdle, another option is to shutdown/deallocate all VMs before migration. Migrate the deallocated VMs, then restart them on the Azure Resource Manager side. The benefit here is that VM extensions will migrate. The downside is that all public facing Virtual IPs will be lost (this may be a non-starter), and obviously the VMs will shut down causing a much greater impact on working applications.

NOTE

If an Azure Security Center policy is configured against the running VMs being migrated, the security policy needs to be stopped before removing extensions, otherwise the security monitoring extension will be reinstalled automatically on the VM after removing it.

- **Availability Sets** - For a virtual network (vNet) to be migrated to Azure Resource Manager, the Classic deployment (i.e. cloud service) contained VMs must all be in one availability set, or the VMs must all not be in any availability set. Having more than one availability set in the cloud service is not compatible with Azure Resource Manager and will halt migration. Additionally, there cannot be some VMs in an availability set, and some VMs not in an availability set. To resolve this, you will need to remediate or reshuffle your cloud service. Plan accordingly as this might be time consuming.
- **Web/Worker Role Deployments** - Cloud Services containing web and worker roles cannot migrate to Azure Resource Manager. The web/worker roles must first be removed from the virtual network before migration can start. A typical solution is to just move web/worker role instances to a separate Classic virtual network that is also linked to an ExpressRoute circuit, or to migrate the code to newer PaaS App Services (this discussion is beyond the scope of this document). In the former redeploy case, create a new Classic virtual network, move/redeploy the web/worker roles to that new virtual network, then delete the deployments from the virtual network being moved. No code changes required. The new [Virtual Network Peering](#) capability can be used to peer together the classic virtual network containing the web/worker roles and other virtual networks in the same Azure region such as the virtual network being migrated (**after virtual network migration is completed as peered virtual networks cannot be migrated**), hence providing the same capabilities with no performance loss and no latency/bandwidth penalties. Given the addition of [Virtual Network Peering](#), web/worker role deployments can now easily be mitigated and not block the migration to Azure Resource Manager.
- **Azure Resource Manager Quotas** - Azure regions have separate quotas/limits for both Classic and Azure Resource Manager. Even though in a migration scenario new hardware isn't being consumed (*we're swapping existing VMs from Classic to Azure Resource Manager*), Azure Resource Manager quotas still need to be in place with enough capacity before migration can start. Listed below are the major limits we've seen cause problems. Open a quota support ticket to raise the limits.

NOTE

These limits need to be raised in the same region as your current environment to be migrated.

- Network Interfaces
- Load Balancers
- Public IPs
- Static Public IPs
- Cores
- Network Security Groups
- Route Tables

You can check your current Azure Resource Manager quotas using the following commands with the latest version of Azure CLI.

Compute (Cores, Availability Sets)

```
az vm list-usage -l <azure-region> -o jsonc
```

Network (Virtual Networks, Static Public IPs, Public IPs, Network Security Groups, Network Interfaces, Load Balancers, Route Tables)

```
az network list-usages -l <azure-region> -o jsonc
```

Storage (Storage Account)

```
az storage account show-usage
```

- **Azure Resource Manager API throttling limits** - If you have a large enough environment (eg. > 400 VMs in a VNET), you might hit the default API throttling limits for writes (currently **1200 writes/hour**) in Azure Resource Manager. Before starting migration, you should raise a support ticket to increase this limit for your subscription.
- **Provisioning Timed Out VM Status** - If any VM has the status of **provisioning timed out**, this needs to be resolved pre-migration. The only way to do this is with downtime by deprovisioning/reprovisioning the VM (delete it, keep the disk, and recreate the VM).
- **RoleStateUnknown VM Status** - If migration halts due to a **role state unknown** error message, inspect the VM using the portal and ensure it is running. This error will typically go away on its own (no remediation required) after a few minutes and is often a transient type often seen during a Virtual Machine **start, stop, restart** operations. **Recommended practice:** re-try migration again after a few minutes.
- **Fabric Cluster does not exist** - In some cases, certain VMs cannot be migrated for various odd reasons. One of these known cases is if the VM was recently created (within the last week or so) and happened to land an Azure cluster that is not yet equipped for Azure Resource Manager workloads. You will get an error that says **fabric cluster does not exist** and the VM cannot be migrated. Waiting a couple of days will usually resolve this particular problem as the cluster will soon get Azure Resource Manager enabled. However, one immediate workaround is to **stop-deallocate** the VM, then continue forward with migration, and start the VM back up in Azure Resource Manager after migrating.

Pitfalls to avoid

- Do not take shortcuts and omit the validate/prepare/abort dry run migrations.
- Most, if not all, of your potential issues will surface during the validate/prepare/abort steps.

Migration

Technical considerations and tradeoffs

Now you are ready because you have worked through the known issues with your environment.

For the real migrations, you might want to consider:

1. Plan and schedule the virtual network (smallest unit of migration) with increasing priority. Do the simple virtual networks first, and progress with the more complicated virtual networks.
2. Most customers will have non-production and production environments. Schedule production last.
3. **(OPTIONAL)** Schedule a maintenance downtime with plenty of buffer in case unexpected issues arise.
4. Communicate with and align with your support teams in case issues arise.

Patterns of success

The technical guidance from the Lab Test section above should be considered and mitigated prior to a real

migration. With adequate testing, the migration is actually a non-event. For production environments, it might be helpful to have additional support, such as a trusted Microsoft partner or Microsoft Premier services.

Pitfalls to avoid

Not fully testing may cause issues and delay in the migration.

Beyond Migration

Technical considerations and tradeoffs

Now that you are in Azure Resource Manager, maximize the platform. Read the [overview of Azure Resource Manager](#) to find out about additional benefits.

Things to consider:

- Bundling the migration with other activities. Most customers opt for an application maintenance window. If so, you might want to use this downtime to enable other Azure Resource Manager capabilities like encryption and migration to Managed Disks.
- Revisit the technical and business reasons for Azure Resource Manager; enable the additional services available only on Azure Resource Manager that apply to your environment.
- Modernize your environment with PaaS services.

Patterns of success

Be purposeful on what services you now want to enable in Azure Resource Manager. Many customers find the below compelling for their Azure environments:

- [Role Based Access Control](#).
- [Azure Resource Manager templates for easier and more controlled deployment](#).
- [Tags](#).
- [Activity Control](#)
- [Azure Policies](#)

Pitfalls to avoid

Remember why you started this Classic to Azure Resource Manager migration journey. What were the original business reasons? Did you achieve the business reason?

Next steps

- [Overview of platform-supported migration of IaaS resources from classic to Azure Resource Manager](#)
- [Technical deep dive on platform-supported migration from classic to Azure Resource Manager](#)
- [Planning for migration of IaaS resources from classic to Azure Resource Manager](#)
- [Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Community tools for assisting with migration of IaaS resources from classic to Azure Resource Manager](#)
- [Review most common migration errors](#)
- [Review the most frequently asked questions about migrating IaaS resources from classic to Azure Resource Manager](#)

Migrate IaaS resources from classic to Azure Resource Manager by using Azure CLI

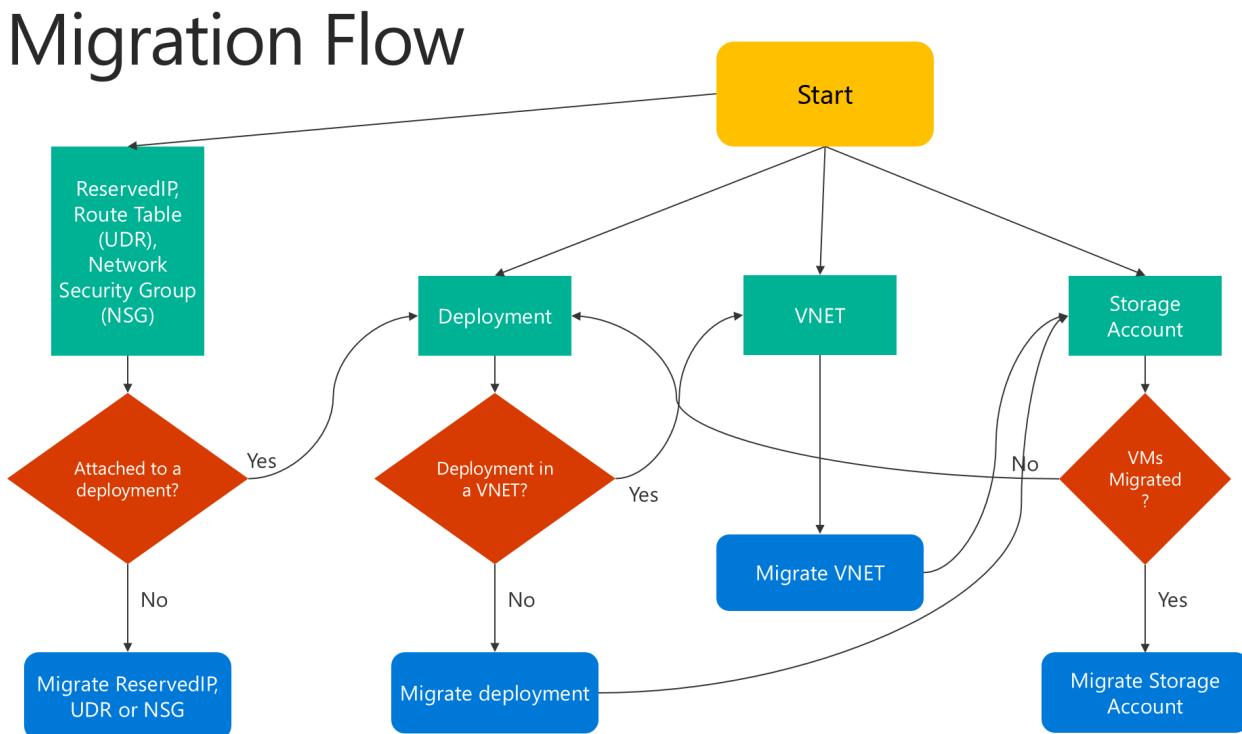
3/6/2019 • 6 minutes to read • [Edit Online](#)

These steps show you how to use Azure command-line interface (CLI) commands to migrate infrastructure as a service (IaaS) resources from the classic deployment model to the Azure Resource Manager deployment model. The article requires the [Azure classic CLI](#). Since Azure CLI is only applicable for Azure Resource Manager resources, it cannot be used for this migration.

NOTE

All the operations described here are idempotent. If you have a problem other than an unsupported feature or a configuration error, we recommend that you retry the prepare, abort, or commit operation. The platform will then try the action again.

Here is a flowchart to identify the order in which steps need to be executed during a migration process



Step 1: Prepare for migration

Here are a few best practices that we recommend as you evaluate migrating IaaS resources from classic to Resource Manager:

- Read through the [list of unsupported configurations or features](#). If you have virtual machines that use unsupported configurations or features, we recommend that you wait for the feature/configuration support to be announced. Alternatively, you can remove that feature or move out of that configuration to enable migration if it suits your needs.
- If you have automated scripts that deploy your infrastructure and applications today, try to create a similar test setup by using those scripts for migration. Alternatively, you can set up sample environments by using the

Azure portal.

IMPORTANT

Application Gateways are not currently supported for migration from classic to Resource Manager. To migrate a classic virtual network with an Application gateway, remove the gateway before running a Prepare operation to move the network. After you complete the migration, reconnect the gateway in Azure Resource Manager.

ExpressRoute gateways connecting to ExpressRoute circuits in another subscription cannot be migrated automatically. In such cases, remove the ExpressRoute gateway, migrate the virtual network and recreate the gateway. Please see [Migrate ExpressRoute circuits and associated virtual networks from the classic to the Resource Manager deployment model](#) for more information.

Step 2: Set your subscription and register the provider

For migration scenarios, you need to set up your environment for both classic and Resource Manager. [Install Azure CLI](#) and [select your subscription](#).

Sign-in to your account.

```
azure login
```

Select the Azure subscription by using the following command.

```
azure account set "<azure-subscription-name>"
```

NOTE

Registration is a one time step but it needs to be done once before attempting migration. Without registering you'll see the following error message

BadRequest : Subscription is not registered for migration.

Register with the migration resource provider by using the following command. Note that in some cases, this command times out. However, the registration will be successful.

```
azure provider register Microsoft.ClassicInfrastructureMigrate
```

Please wait five minutes for the registration to finish. You can check the status of the approval by using the following command. Make sure that RegistrationState is `Registered` before you proceed.

```
azure provider show Microsoft.ClassicInfrastructureMigrate
```

Now switch CLI to the `asm` mode.

```
azure config mode asm
```

Step 3: Make sure you have enough Azure Resource Manager Virtual Machine vCPUs in the Azure region of your current deployment or

VNET

For this step you'll need to switch to `arm` mode. Do this with the following command.

```
azure config mode arm
```

You can use the following CLI command to check the current number of vCPUs you have in Azure Resource Manager. To learn more about vCPU quotas, see [Limits and the Azure Resource Manager](#)

```
azure vm list-usage -l "<Your VNET or Deployment's Azure region"
```

Once you're done verifying this step, you can switch back to `asm` mode.

```
azure config mode asm
```

Step 4: Option 1 - Migrate virtual machines in a cloud service

Get the list of cloud services by using the following command, and then pick the cloud service that you want to migrate. Note that if the VMs in the cloud service are in a virtual network or if they have web/worker roles, you will get an error message.

```
azure service list
```

Run the following command to get the deployment name for the cloud service from the verbose output. In most cases, the deployment name is the same as the cloud service name.

```
azure service show <serviceName> -vv
```

First, validate if you can migrate the cloud service using the following commands:

```
azure service deployment validate-migration <serviceName> <deploymentName> new "" "" ""
```

Prepare the virtual machines in the cloud service for migration. You have two options to choose from.

If you want to migrate the VMs to a platform-created virtual network, use the following command.

```
azure service deployment prepare-migration <serviceName> <deploymentName> new "" "" ""
```

If you want to migrate to an existing virtual network in the Resource Manager deployment model, use the following command.

```
azure service deployment prepare-migration <serviceName> <deploymentName> existing
<destinationVNETResourceGroupName> <subnetName> <vnetName>
```

After the prepare operation is successful, you can look through the verbose output to get the migration state of the VMs and ensure that they are in the `Prepared` state.

```
azure vm show <vmName> -vv
```

Check the configuration for the prepared resources by using either CLI or the Azure portal. If you are not ready for migration and you want to go back to the old state, use the following command.

```
azure service deployment abort-migration <serviceName> <deploymentName>
```

If the prepared configuration looks good, you can move forward and commit the resources by using the following command.

```
azure service deployment commit-migration <serviceName> <deploymentName>
```

Step 4: Option 2 - Migrate virtual machines in a virtual network

Pick the virtual network that you want to migrate. Note that if the virtual network contains web/worker roles or VMs with unsupported configurations, you will get a validation error message.

Get all the virtual networks in the subscription by using the following command.

```
azure network vnet list
```

The output will look something like this:

```
info: Executing command network vnet list
+ Looking up the virtual network sites
data: Name                           Location  Affinity gr
data: -----
data: Group classicubuntu16 classicubuntu16   East US
data: Group LinuxHost                 East US
data: Group LinuxRG LinuxRG          East US
data: Group SUSEClassicRG SUSEClassicRG  East US
info: network vnet list command OK
```

In the above example, the **virtualNetworkName** is the entire name "**Group classicubuntu16 classicubuntu16**".

First, validate if you can migrate the virtual network using the following command:

```
azure network vnet validate-migration <virtualNetworkName>
```

Prepare the virtual network of your choice for migration by using the following command.

```
azure network vnet prepare-migration <virtualNetworkName>
```

Check the configuration for the prepared virtual machines by using either CLI or the Azure portal. If you are not ready for migration and you want to go back to the old state, use the following command.

```
azure network vnet abort-migration <virtualNetworkName>
```

If the prepared configuration looks good, you can move forward and commit the resources by using the following command.

```
azure network vnet commit-migration <virtualNetworkName>
```

Step 5: Migrate a storage account

Once you're done migrating the virtual machines, we recommend you migrate the storage account.

Prepare the storage account for migration by using the following command

```
azure storage account prepare-migration <storageAccountName>
```

Check the configuration for the prepared storage account by using either CLI or the Azure portal. If you are not ready for migration and you want to go back to the old state, use the following command.

```
azure storage account abort-migration <storageAccountName>
```

If the prepared configuration looks good, you can move forward and commit the resources by using the following command.

```
azure storage account commit-migration <storageAccountName>
```

Next steps

- [Overview of platform-supported migration of IaaS resources from classic to Azure Resource Manager](#)
- [Technical deep dive on platform-supported migration from classic to Azure Resource Manager](#)
- [Planning for migration of IaaS resources from classic to Azure Resource Manager](#)
- [Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Community tools for assisting with migration of IaaS resources from classic to Azure Resource Manager](#)
- [Review most common migration errors](#)
- [Review the most frequently asked questions about migrating IaaS resources from classic to Azure Resource Manager](#)

Common errors during Classic to Azure Resource Manager migration

4/9/2018 • 9 minutes to read • [Edit Online](#)

This article catalogs the most common errors and mitigations during the migration of IaaS resources from Azure classic deployment model to the Azure Resource Manager stack.

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](#).

List of errors

ERROR STRING	MITIGATION
Internal server error	In some cases, this is a transient error that goes away with a retry. If it continues to persist, contact Azure support as it needs investigation of platform logs. NOTE: Once the incident is tracked by the support team, please do not attempt any self-mitigation as this might have unintended consequences on your environment.
Migration is not supported for Deployment {deployment-name} in HostedService {hosted-service-name} because it is a PaaS deployment (Web/Worker).	This happens when a deployment contains a web/worker role. Since migration is only supported for Virtual Machines, please remove the web/worker role from the deployment and try migration again.
Template {template-name} deployment failed. CorrelationId= {guid}	In the backend of migration service, we use Azure Resource Manager templates to create resources in the Azure Resource Manager stack. Since templates are idempotent, usually you can safely retry the migration operation to get past this error. If this error continues to persist, please contact Azure support and give them the CorrelationId. NOTE: Once the incident is tracked by the support team, please do not attempt any self-mitigation as this might have unintended consequences on your environment.
The virtual network {virtual-network-name} does not exist.	This can happen if you created the Virtual Network in the new Azure portal. The actual Virtual Network name follows the pattern "Group * < VNET name >"
VM {vm-name} in HostedService {hosted-service-name} contains Extension {extension-name} which is not supported in Azure Resource Manager. It is recommended to uninstall it from the VM before continuing with migration.	XML extensions such as BGInfo 1.* are not supported in Azure Resource Manager. Therefore, these extensions cannot be migrated. If these extensions are left installed on the virtual machine, they are automatically uninstalled before completing the migration.

ERROR STRING	MITIGATION
<p>VM {vm-name} in HostedService {hosted-service-name} contains Extension VMSnapshot/VMSnapshotLinux, which is currently not supported for Migration. Uninstall it from the VM and add it back using Azure Resource Manager after the Migration is Complete</p>	<p>This is the scenario where the virtual machine is configured for Azure Backup. Since this is currently an unsupported scenario, please follow the workaround at https://aka.ms/vmbakupmigration</p>
<p>VM {vm-name} in HostedService {hosted-service-name} contains Extension {extension-name} whose Status is not being reported from the VM. Hence, this VM cannot be migrated. Ensure that the Extension status is being reported or uninstall the extension from the VM and retry migration.</p>	<p>Azure guest agent & VM Extensions need outbound internet access to the VM storage account to populate their status. Common causes of status failure include</p> <ul style="list-style-type: none"> • a Network Security Group that blocks outbound access to the internet • If the VNET has on premises DNS servers and DNS connectivity is lost
<p>VM {vm-name} in HostedService {hosted-service-name} contains Extension {extension-name} reporting Handler Status: {handler-status}. Hence, the VM cannot be migrated. Ensure that the Extension handler status being reported is {handler-status} or uninstall it from the VM and retry migration.</p>	<p>If you continue to see an unsupported status, you can uninstall the extensions to skip this check and move forward with migration.</p>
<p>VM Agent for VM {vm-name} in HostedService {hosted-service-name} is reporting the overall agent status as Not Ready. Hence, the VM may not be migrated, if it has a migratable extension. Ensure that the VM Agent is reporting overall agent status as Ready. Refer to https://aka.ms/classiciaasmigrationfaqs.</p>	
<p>Migration is not supported for Deployment {deployment-name} in HostedService {hosted-service-name} because it has multiple Availability Sets.</p>	<p>Currently, only hosted services that have 1 or less Availability sets can be migrated. To work around this problem, please move the additional Availability sets and Virtual machines in those Availability sets to a different hosted service.</p>
<p>Migration is not supported for Deployment {deployment-name} in HostedService {hosted-service-name} because it has VMs that are not part of the Availability Set even though the HostedService contains one.</p>	<p>The workaround for this scenario is to either move all the virtual machines into a single Availability set or remove all Virtual machines from the Availability set in the hosted service.</p>
<p>Storage account/HostedService/Virtual Network {virtual-network-name} is in the process of being migrated and hence cannot be changed</p>	<p>This error happens when the "Prepare" migration operation has been completed on the resource and an operation that would make a change to the resource is triggered. Because of the lock on the management plane after "Prepare" operation, any changes to the resource are blocked. To unlock the management plane, you can run the "Commit" migration operation to complete migration or the "Abort" migration operation to roll back the "Prepare" operation.</p>
<p>Migration is not allowed for HostedService {hosted-service-name} because it has VM {vm-name} in State: RoleStateUnknown. Migration is allowed only when the VM is in one of the following states - Running, Stopped, Stopped Deallocated.</p>	<p>The VM might be undergoing through a state transition, which usually happens when during an update operation on the HostedService such as a reboot, extension installation etc. It is recommended for the update operation to complete on the HostedService before trying migration.</p>
<p>Deployment {deployment-name} in HostedService {hosted-service-name} contains a VM {vm-name} with Data Disk {data-disk-name} whose physical blob size {size-of-the-vhd-blob-backing-the-data-disk} bytes does not match the VM Data Disk logical size {size-of-the-data-disk-specified-in-the-vm-api} bytes. Migration will proceed without specifying a size for the data disk for the Azure Resource Manager VM.</p>	<p>This error happens if you've resized the VHD blob without updating the size in the VM API model. Detailed mitigation steps are outlined below.</p>

ERROR STRING	MITIGATION
A storage exception occurred while validating data disk {data disk name} with media link {data disk Uri} for VM {VM name} in Cloud Service {Cloud Service name}. Please ensure that the VHD media link is accessible for this virtual machine	This error can happen if the disks of the VM have been deleted or are not accessible anymore. Please make sure the disks for the VM exist.
VM {vm-name} in HostedService {cloud-service-name} contains Disk with MediaLink {vhdx-uri} which has blob name {vhdx-blob-name} that is not supported in Azure Resource Manager.	This error occurs when the name of the blob has a "/" in it which is not supported in Compute Resource Provider currently.
Migration is not allowed for Deployment {deployment-name} in HostedService {cloud-service-name} as it is not in the regional scope. Please refer to https://aka.ms/regionscope for moving this deployment to regional scope.	In 2014, Azure announced that networking resources will move from a cluster level scope to regional scope. See https://aka.ms/regionscope for more details . This error happens when the deployment being migrated has not had an update operation, which automatically moves it to a regional scope. Best workaround is to either add an endpoint to a VM or a data disk to the VM and then retry migration. See How to set up endpoints on a classic Windows virtual machine in Azure or Attach a data disk to a Windows virtual machine created with the classic deployment model
Migration is not supported for Virtual Network {vnet-name} because it has non-gateway PaaS deployments.	This error occurs when you have non-gateway PaaS deployments such as Application Gateway or API Management services that are connected to the Virtual Network.

Detailed mitigations

VM with Data Disk whose physical blob size bytes does not match the VM Data Disk logical size bytes.

This happens when the Data disk logical size can get out of sync with the actual VHD blob size. This can be easily verified using the following commands:

Verifying the issue

```

# Store the VM details in the VM object
$vm = Get-AzureVM -ServiceName $servicename -Name $vmname

# Display the data disk properties
# NOTE the data disk LogicalDiskSizeInGB below which is 11GB. Also note the MediaLink Uri of the VHD blob as
we'll use this in the next step
$vm.VM.DataVirtualHardDisks

HostCaching      : None
DiskLabel        :
DiskName         : coreosvm-coreosvm-0-201611230636240687
Lun              : 0
LogicalDiskSizeInGB : 11
MediaLink        : https://contosostorage.blob.core.windows.net/vhds/coreosvm-dd1.vhd
SourceMediaLink   :
IOType           : Standard
ExtensionData    :

# Now get the properties of the blob backing the data disk above
# NOTE the size of the blob is about 15 GB which is different from LogicalDiskSizeInGB above
blob = Get-AzStorageblob -Blob "coreosvm-dd1.vhd" -Container vhds

$blob

ICloudBlob      : Microsoft.WindowsAzure.Storage.Blob.CloudPageBlob
BlobType        : PageBlob
Length          : 16106127872
ContentType     : application/octet-stream
LastModified    : 11/23/2016 7:16:22 AM +00:00
SnapshotTime    :
ContinuationToken :
Context         : Microsoft.WindowsAzure.Commands.Common.Storage.AzureStorageContext
Name            : coreosvm-dd1.vhd

```

Mitigating the issue

```

# Convert the blob size in bytes to GB into a variable which we'll use later
$newSize = [int]($blob.Length / 1GB)

# See the calculated size in GB
$newSize

15

# Store the disk name of the data disk as we'll use this to identify the disk to be updated
$diskName = $vm.VM.DataVirtualHardDisks[0].DiskName

# Identify the LUN of the data disk to remove
$lunToRemove = $vm.VM.DataVirtualHardDisks[0].Lun

# Now remove the data disk from the VM so that the disk isn't leased by the VM and it's size can be updated
Remove-AzureDataDisk -LUN $lunToRemove -VM $vm | Update-AzureVm -Name $vmname -ServiceName $servicename

OperationDescription OperationId          OperationStatus
----- ----- -----
Update-AzureVM      213xx1-b44b-1v6n-23gg-591f2a13cd16  Succeeded

# Verify we have the right disk that's going to be updated
Get-AzureDisk -DiskName $diskName

AffinityGroup      :
AttachedTo        :
IsCorrupted      : False
Label             :
Location          : East US
DiskSizeInGB      : 11

```

```

DISKSIZEINGB          : 11
MediaLink             : https://contosostorage.blob.core.windows.net/vhds/coreosvm-dd1.vhd
DiskName              : coreosvm-coreosvm-0-201611230636240687
SourceImageName       :
OS                   :
IOType                : Standard
OperationDescription  : Get-AzureDisk
OperationId           : 0c56a2b7-a325-123b-7043-74c27d5a61fd
OperationStatus        : Succeeded

# Now update the disk to the new size
Update-AzureDisk -DiskName $diskName -ResizedSizeInGB $newSize -Label $diskName

OperationDescription OperationId          OperationStatus
-----  -----  -----
Update-AzureDisk      cv134b65-1b6n-8908-abuo-ce9e395ac3e7 Succeeded

# Now verify that the "DiskSizeInGB" property of the disk matches the size of the blob
Get-AzureDisk -DiskName $diskName

AffinityGroup         :
AttachedTo            :
IsCorrupted          : False
Label                : coreosvm-coreosvm-0-201611230636240687
Location              : East US
DiskSizeInGB          : 15
MediaLink             : https://contosostorage.blob.core.windows.net/vhds/coreosvm-dd1.vhd
DiskName              : coreosvm-coreosvm-0-201611230636240687
SourceImageName       :
OS                   :
IOType                : Standard
OperationDescription  : Get-AzureDisk
OperationId           : 1v53bde5-cv56-5621-9078-16b9c8a0bad2
OperationStatus        : Succeeded

# Now we'll add the disk back to the VM as a data disk. First we need to get an updated VM object
$vm = Get-AzureVM -ServiceName $servicename -Name $vmname

Add-AzureDataDisk -Import -DiskName $diskName -LUN 0 -VM $vm -HostCaching ReadWrite | Update-AzureVm -Name
$vmname -ServiceName $servicename

OperationDescription OperationId          OperationStatus
-----  -----  -----
Update-AzureVM       b0ad3d4c-4v68-45vb-xxc1-134fd010d0f8 Succeeded

```

Moving a VM to a different subscription after completing migration

After you complete the migration process, you may want to move the VM to another subscription. However, if you have a secret/certificate on the VM that references a Key Vault resource, the move is currently not supported. The below instructions will allow you to workaround the issue.

PowerShell

```

$vm = Get-AzVM -ResourceGroupName "MyRG" -Name "MyVM"
Remove-AzVMSecret -VM $vm
Update-AzVM -ResourceGroupName "MyRG" -VM $vm

```

Azure CLI

```
az vm update -g "myrg" -n "myvm" --set osProfile.Secrets=[]
```

Next steps

- Overview of platform-supported migration of IaaS resources from classic to Azure Resource Manager
- Technical deep dive on platform-supported migration from classic to Azure Resource Manager
- Planning for migration of IaaS resources from classic to Azure Resource Manager
- Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager
- Use CLI to migrate IaaS resources from classic to Azure Resource Manager
- Community tools for assisting with migration of IaaS resources from classic to Azure Resource Manager
- Review the most frequently asked questions about migrating IaaS resources from classic to Azure Resource Manager

Community tools to migrate IaaS resources from classic to Azure Resource Manager

4/9/2018 • 2 minutes to read • [Edit Online](#)

This article catalogs the tools that have been provided by the community to assist with migration of IaaS resources from classic to the Azure Resource Manager deployment model.

NOTE

These tools are not officially supported by Microsoft Support. Therefore they are open sourced on GitHub and we're happy to accept PRs for fixes or additional scenarios. To report an issue, use the GitHub issues feature.

Migrating with these tools will cause downtime for your classic Virtual Machine. If you're looking for platform supported migration, visit

- [Platform supported migration of IaaS resources from Classic to Azure Resource Manager stack](#)
- [Technical Deep Dive on Platform supported migration from Classic to Azure Resource Manager](#)
- [Migrate IaaS resources from Classic to Azure Resource Manager using Azure PowerShell](#)

AsmMetadataParser

This is a collection of helper tools created as part of enterprise migrations from Azure Service Management to Azure Resource Manager. This tool allows you to replicate your infrastructure into another subscription which can be used for testing migration and iron out any issues before running the migration on your Production subscription.

[Link to the tool documentation](#)

migAz

migAz is an additional option to migrate a complete set of classic IaaS resources to Azure Resource Manager IaaS resources. The migration can occur within the same subscription or between different subscriptions and subscription types (ex: CSP subscriptions).

[Link to the tool documentation](#)

Next Steps

- [Overview of platform-supported migration of IaaS resources from classic to Azure Resource Manager](#)
- [Technical deep dive on platform-supported migration from classic to Azure Resource Manager](#)
- [Planning for migration of IaaS resources from classic to Azure Resource Manager](#)
- [Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Use CLI to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Review most common migration errors](#)
- [Review the most frequently asked questions about migrating IaaS resources from classic to Azure Resource Manager](#)

Frequently asked questions about classic to Azure Resource Manager migration

4/9/2018 • 5 minutes to read • [Edit Online](#)

Does this migration plan affect any of my existing services or applications that run on Azure virtual machines?

No. The VMs (classic) are fully supported services in general availability. You can continue to use these resources to expand your footprint on Microsoft Azure.

What happens to my VMs if I don't plan on migrating in the near future?

We are not deprecating the existing classic APIs and resource model. We want to make migration easy, considering the advanced features that are available in the Resource Manager deployment model. We highly recommend that you review [some of the advancements](#) that are part of IaaS under Resource Manager.

What does this migration plan mean for my existing tooling?

Updating your tooling to the Resource Manager deployment model is one of the most important changes that you have to account for in your migration plans.

How long will the management-plane downtime be?

It depends on the number of resources that are being migrated. For smaller deployments (a few tens of VMs), the whole migration should take less than an hour. For large-scale deployments (hundreds of VMs), the migration can take a few hours.

Can I roll back after my migrating resources are committed in Resource Manager?

You can abort your migration as long as the resources are in the prepared state. Rollback is not supported after the resources have been successfully migrated through the commit operation.

Can I roll back my migration if the commit operation fails?

You cannot abort migration if the commit operation fails. All migration operations, including the commit operation, are idempotent. So we recommend that you retry the operation after a short time. If you still face an error, create a support ticket or create a forum post with the ClassicIaaSMigration tag on our [VM forum](#).

Do I have to buy another express route circuit if I have to use IaaS under Resource Manager?

No. We recently enabled [moving ExpressRoute circuits from the classic to the Resource Manager deployment model](#). You don't have to buy a new ExpressRoute circuit if you already have one.

What if I had configured Role-Based Access Control policies for my

classic IaaS resources?

During migration, the resources transform from classic to Resource Manager. So we recommend that you plan the RBAC policy updates that need to happen after migration.

I backed up my classic VMs in a vault. Can I migrate my VMs from classic mode to Resource Manager mode and protect them in a Recovery Services vault?

If you move a VM from classic to Resource Manager mode, backups taken prior to migration will not migrate to newly migrated Resource Manager VM. However, if you wish to keep your backups of classic VMs, follow these steps before the migration.

1. In the Recovery Services vault, go to the **Protected Items** tab and select the VM.
2. Click Stop Protection. Leave *Delete associated backup data* option **unchecked**.

NOTE

You will be charged backup instance cost till you retain data. Backup copies will be pruned as per retention range. However, last backup copy is always kept until you explicitly delete backup data. It is advised to check your retention range of the Virtual machine and trigger "Delete Backup Data" on the protected item in the vault once the retention range is over.

To migrate the virtual machine to Resource Manager mode,

1. Delete the backup/snapshot extension from the VM.
2. Migrate the virtual machine from classic mode to Resource Manager mode. Make sure the storage and network information corresponding to the virtual machine is also migrated to Resource Manager mode.

Additionally, if you want to back up the migrated VM, go to Virtual Machine management blade to [enable backup](#).

Can I validate my subscription or resources to see if they're capable of migration?

Yes. In the platform-supported migration option, the first step in preparing for migration is to validate that the resources are capable of migration. In case the validate operation fails, you receive messages for all the reasons the migration cannot be completed.

What happens if I run into a quota error while preparing the IaaS resources for migration?

We recommend that you abort your migration and then log a support request to increase the quotas in the region where you are migrating the VMs. After the quota request is approved, you can start executing the migration steps again.

How do I report an issue?

Post your issues and questions about migration to our [VM forum](#), with the keyword ClassicIaaSMigration. We recommend posting all your questions on this forum. If you have a support contract, you're welcome to log a support ticket as well.

What if I don't like the names of the resources that the platform chose

during migration?

All the resources that you explicitly provide names for in the classic deployment model are retained during migration. In some cases, new resources are created. For example: a network interface is created for every VM. We currently don't support the ability to control the names of these new resources created during migration. Log your votes for this feature on the [Azure feedback forum](#).

Can I migrate ExpressRoute circuits used across subscriptions with authorization links?

ExpressRoute circuits which use cross-subscription authorization links cannot be migrated automatically without downtime. We have guidance on how these can be migrated using manual steps. See [Migrate ExpressRoute circuits and associated virtual networks from the classic to the Resource Manager deployment model](#) for steps and more information.

I got the message "VM is reporting the overall agent status as Not Ready. Hence, the VM cannot be migrated. Ensure that the VM Agent is reporting overall agent status as Ready" or "VM contains Extension whose Status is not being reported from the VM. Hence, this VM cannot be migrated."

This message is received when the VM does not have outbound connectivity to the internet. The VM agent uses outbound connectivity to reach the Azure storage account for updating the agent status every five minutes.

Next steps

- [Overview of platform-supported migration of IaaS resources from classic to Azure Resource Manager](#)
- [Technical deep dive on platform-supported migration from classic to Azure Resource Manager](#)
- [Planning for migration of IaaS resources from classic to Azure Resource Manager](#)
- [Use PowerShell to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Use CLI to migrate IaaS resources from classic to Azure Resource Manager](#)
- [Community tools for assisting with migration of IaaS resources from classic to Azure Resource Manager](#)
- [Review most common migration errors](#)

Frequently asked question about Linux Virtual Machines

6/24/2019 • 3 minutes to read • [Edit Online](#)

This article addresses some common questions about Linux virtual machines created in Azure using the Resource Manager deployment model. For the Windows version of this topic, see [Frequently asked question about Windows Virtual Machines](#)

What can I run on an Azure VM?

All subscribers can run server software on an Azure virtual machine. For more information, see [Linux on Azure-Endorsed Distributions](#)

How much storage can I use with a virtual machine?

Each data disk can be up to 4 TB (4,095 GB). The number of data disks you can use depends on the size of the virtual machine. For details, see [Sizes for Virtual Machines](#).

Azure Managed Disks are the recommended disk storage offerings for use with Azure Virtual Machines for persistent storage of data. You can use multiple Managed Disks with each Virtual Machine. Managed Disks offer two types of durable storage options: Premium and Standard Managed Disks. For pricing information, see [Managed Disks Pricing](#).

Azure storage accounts can also provide storage for the operating system disk and any data disks. Each disk is a .vhd file stored as a page blob. For pricing details, see [Storage Pricing Details](#).

How can I access my virtual machine?

Establish a remote connection to sign on to the virtual machine, using Secure Shell (SSH). See the instructions on how to connect [from Windows](#) or [from Linux and Mac](#). By default, SSH allows a maximum of 10 concurrent connections. You can increase this number by editing the configuration file.

If you're having problems, check out [Troubleshoot Secure Shell \(SSH\) connections](#).

Can I use the temporary disk (/dev/sdb1) to store data?

Don't use the temporary disk (/dev/sdb1) to store data. It is only there for temporary storage. You risk losing data that can't be recovered.

Can I copy or clone an existing Azure VM?

Yes. For instructions, see [How to create a copy of a Linux virtual machine in the Resource Manager deployment model](#).

Why am I not seeing Canada Central and Canada East regions through Azure Resource Manager?

The two new regions of Canada Central and Canada East are not automatically registered for virtual machine creation for existing Azure subscriptions. This registration is done automatically when a virtual machine is deployed through the Azure portal to any other region using Azure Resource Manager. After a virtual machine is

deployed to any other Azure region, the new regions should be available for subsequent virtual machines.

Can I add a NIC to my VM after it's created?

Yes, this is now possible. The VM first needs to be stopped deallocated. Then you can add or remove a NIC (unless it's the last NIC on the VM).

Are there any computer name requirements?

Yes. The computer name can be a maximum of 64 characters in length. See [Naming conventions rules and restrictions](#) for more information around naming your resources.

Are there any resource group name requirements?

Yes. The resource group name can be a maximum of 90 characters in length. See [Naming conventions rules and restrictions](#) for more information about resource groups.

What are the username requirements when creating a VM?

Usernames should be 1 - 32 characters in length.

The following usernames are not allowed:

administrator	admin	user	user1
test	user2	test1	user3
admin1	1	123	a
actuser	adm	admin2	aspnet
backup	console	david	guest
john	owner	root	server
sql	support	support_388945a0	sys
test2	test3	user4	user5
video			

What are the password requirements when creating a VM?

There are varying password length requirements, depending on the tool you are using:

- Portal - between 12 - 72 characters
- PowerShell - between 8 - 123 characters
- CLI - between 12 - 123

Passwords must also meet 3 out of the following 4 complexity requirements:

- Have lower characters

- Have upper characters
- Have a digit
- Have a special character (Regex match [\W_])

The following passwords are not allowed:

abc@123	P@\$\$w0rd	P@ssw0rd	P@ssword123	Pa\$\$word
pass@word1	Password!	Password1	Password22	iloveyou!

- Allocation failures
 - [Allocation failures](#)
 - [Allocation failures for classic deployments](#)
- Boot diagnostics
- RDP
 - [Reset RDP](#)
 - [RDP troubleshooting](#)
 - [Detailed RDP troubleshooting](#)
 - [Troubleshoot specific errors](#)
- SSH
 - [SSH troubleshooting](#)
 - [Detailed SSH troubleshooting](#)
 - [Common error messages](#)
 - [Performance issues with Windows VMs](#)
 - [How to use PerfInsights](#)
 - [Performance diagnostics extension](#)
- [Install Windows VM agent offline](#)
- Redeploy a VM
 - [Linux](#)
 - [Windows](#)
- Reset VM password
 - [Windows](#)
 - [Linux](#)
- [Reset NIC](#)
- [Restarting or resizing a VM](#)
- Use the serial console
 - [Linux VM](#)
 - [Serial Console GRUB/Single user mode](#)
 - [Serial Console NMI/SysRq](#)
 - [Windows VM](#)
 - [CMD and PowerShell commands](#)
- Errors when deleting storage resources
- Unexpected reboots of VMs with attached VHDs
- Windows activation problems
- Application access issues
- Troubleshoot deployments
 - [Linux](#)
 - [Windows](#)
- Device names are changed
- VM recovery access
 - [Windows](#)
 - [PowerShell](#)
 - [Azure portal](#)
 - [Linux](#)
 - [CLI](#)
 - [Azure portal](#)

- [Boot errors](#)
- [BitLocker errors](#)
- [Checking file system errors](#)
- [Blue screen errors](#)
- [Throttling errors](#)
- [Use nested virtualization](#)
- [Understand a system reboot](#)