

Table of 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 VM 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 Team Services

15a - Create LAMP stack

15b - Create LEMP stack

15c - Create MEAN stack

16 - Secure web server with SSL

Deploy a Packer image and apps to a scale set

Samples

Azure CLI

Azure PowerShell

Concepts

Azure Resource Manager

Regions and availability

VM types and sizes

General purpose

Compute optimized

Memory optimized

Storage optimized

GPU optimized

High performance compute

Azure compute units (ACU)

Benchmark scores

Endorsed distros

Maintenance and updates

Disk storage

Managed Disks

Premium storage

Premium storage performance

Standard storage

Scalability targets for disks

Backup and disaster recovery for disks

How to enable write accelerator

Troubleshoot attached VHDs

Networking

Auto-scale applications

Infrastructure automation

Security and policy

Monitoring

Backup and recovery

High performance computing

Deployment considerations

[Infrastructure guidelines](#)

[vCPU quotas](#)

[How-to guides](#)

[Create VMs](#)

[Use the CLI](#)

[Use a template](#)

[Copy or clone a VM](#)

[Secure VMs](#)

[Encrypt](#)

[Use access controls](#)

[Use policies](#)

[Create a Key Vault](#)

[Create and use SSH keys](#)

[Protect VMs](#)

[Back up a single VM](#)

[Back up multiple VMs](#)

[Restore a disk](#)

[Restore individual files](#)

[Manage VMs](#)

[Prepay for VMs - Reserved Instances](#)

[VM usage](#)

[Common CLI tasks](#)

[Move a VM](#)

[Change VM size](#)

[Swap the OS disk](#)

[Tag a VM](#)

[Run scripts on a VM](#)

[Use Remote Desktop](#)

[Join VM to Azure Active Directory](#)

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

[Updates and patches](#)

[Azure VM agent](#)

Planned maintenance

Mitigating speculative execution

Scheduled events

Monitor metadata

Use Images

Find and use images

Create custom image

Capture VM to image

Build image with Packer

Download existing disk

Availability and scale

Autoscale

High availability

Vertically scale

Create VM in availability zone

Use automation tools

Ansible

Terraform

Cloud-init

Jenkins

Run containers

Create Docker host

Use Docker Machine

Use Docker Compose

Run applications

Cloud Foundry

OpenShift

SAP on Azure

Oracle

Elasticsearch

FreeBSD Packet Filter

Databases

High Performance Computing (HPC)

Manage storage

[Disks FAQs](#)

[Add a disk](#)

[Detach a disk](#)

[Resize a disk](#)

[Snapshot a disk](#)

[Back up unmanaged disks](#)

[Convert to Managed Disks](#)

[Convert disk between Standard and Premium](#)

[Copy files to a VM](#)

[Migrate to Premium storage with Azure Site Recovery](#)

[Find unattached disks](#)

[Use File storage](#)

[Deploy disks with template](#)

[Optimize performance](#)

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 VM extensions

[VM Extension overview](#)

[Custom Script Extension](#)

[OMS Agent Extension](#)

[Linux Diagnostic Extension](#)

[Network Watcher Agent](#)

[VMAccess Extension](#)

[Restrict extension installation](#)

[Migrate VMs](#)

[Migrate AWS and on-premises VMs](#)

[Migrate from Classic to Azure Resource Manager](#)

[Troubleshoot](#)

[SSH connections](#)

[Reset Linux VM password](#)

[Understand system reboot](#)

[Boot diagnostics](#)

[Serial console](#)

[Access to applications](#)

[Allocation failures](#)

[Allocation failures-classic VM](#)

[Deployment issues](#)

[Creating a VM](#)

[Device names are changed](#)

[Redeploy VM to a new Azure node](#)

[Common error messages](#)

[VM recovery access](#)

[Reference](#)

[Azure CLI](#)

[PowerShell](#)

[.NET](#)

[Java](#)

[Node.js](#)

[Python](#)

[REST](#)

[Resources](#)

[Author templates](#)

[Azure Roadmap](#)

[Community templates](#)

[Pricing](#)

[Regional availability](#)

[Stack Overflow](#)

[Videos](#)

[FAQ](#)

Azure and Linux

4/9/2018 • 7 min 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 November 2017) has 36 regions generally available around the world with an additional 6 regions announced. 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 2.0](#) 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 RedHat do not currently have cloud-init installed. To use cloud-init on a RedHat 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](#)
- Redhat - [Azure Marketplace - RedHat Enterprise Linux 7.2](#)
- Canonical - [Azure Marketplace - Ubuntu Server 16.04 LTS](#)
- Debian - [Azure Marketplace - Debian 8 "Jessie"](#)
- FreeBSD - [Azure Marketplace - FreeBSD 10.3](#)
- 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 2.0](#) 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 2.0

4/25/2018 • 3 min to read • [Edit Online](#)

The Azure CLI 2.0 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 2.0 to deploy a Linux virtual machine (VM) in Azure that runs Ubuntu. To see your VM in action, you then 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.

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI 2.0](#).

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*, adds a user account named *azureuser*, and generates SSH keys if they do not already exist in the default key location (`~/ssh`). 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
```

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. To update package sources and install the latest NGINX package, run the following commands from your SSH session:

```
# update packages
sudo apt-get -y update

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

When done, `exit` the SSH session.

View the web server in action

With NGINX installed and port 80 now open on your VM from the Internet, use a web browser of your choice to view the default NGINX welcome page. Use the public IP address of your VM obtained in a previous step. 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. Make sure that you have exited the SSH session to your VM, then delete the resources as follows:

```
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

5/10/2018 • 4 min to read • [Edit Online](#)

Azure virtual machines (VMs) can be created through the Azure portal. This method provides 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) in Azure that runs Ubuntu. To see your VM in action, you then 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 have an existing SSH key pair, this step can be skipped.

To create an SSH key pair and log into Linux VMs, run the following command from a Bash shell and follow the on-screen directions. For example, you can use the [Azure Cloud Shell](#) or the [Windows Subsystem for Linux](#). The command output includes the file name of the public key file. Copy the contents of the public key file (

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

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

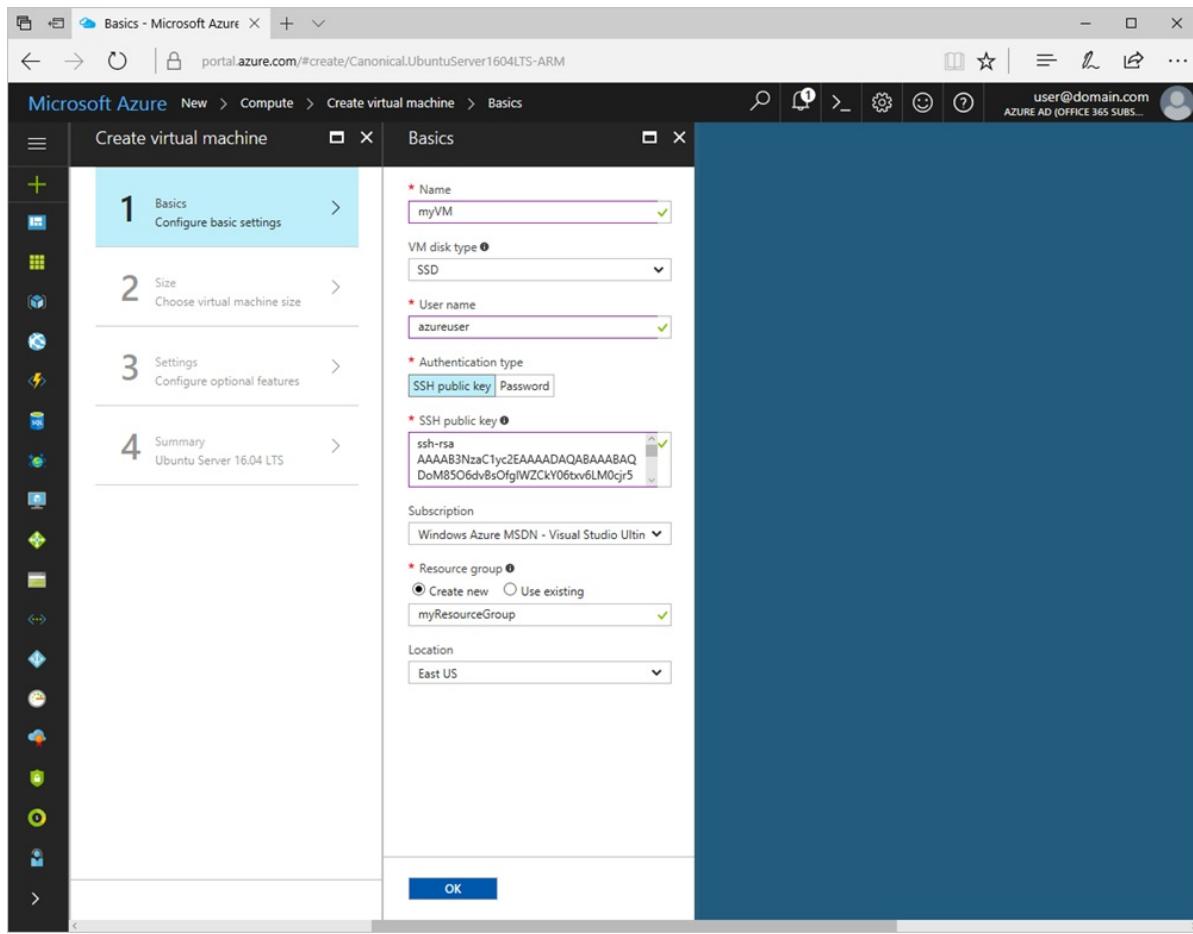
For more detailed information on how to create SSH key pairs, including the use of PuTTy, see [How to use SSH keys with Windows](#).

Log in to Azure

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

Create virtual machine

1. Choose **Create a resource** in the upper left-hand 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. Provide a VM name, such as *myVM*, leave the disk type as *SSD*, then provide a username, such as *azureuser*.
4. For **Authentication type**, select **SSH public key**, then paste your public key into the text box. Take care to remove any leading or trailing white space in your public key.



5. Choose to **Create new** resource group, then provide a name, such as *myResourceGroup*. Choose your desired **Location**, then select **OK**.
6. Select a size for the VM. You can filter by *Compute type* or *Disk type*, for example. A suggested VM size is *D2s_v3*.

Choose a size												
Browse the available sizes and their features												
Search	Compute type			Disk type			vCPUs					
RECOMMENDED	SKU	TYPE	COMPUTE	vCPUS	GB RAM	DATA DISK	MAX IOPS	LOCAL SSD	PREMIUM	ADDITIONAL	ZONES	USD/MONTH
	B1s	Standard	General purpose	1	1	2	800	4 GB	SSD		1,2,3	\$9.67
	B1ms	Standard	General purpose	1	2	2	1600	4 GB	SSD		1,2,3	\$18.60
	B2s	Standard	General purpose	2	4	4	3200	8 GB	SSD		1,2,3	\$38.69
	B2ms	Standard	General purpose	2	8	4	4800	16 GB	SSD		1,2,3	\$76.63
	B4ms	Standard	General purpose	4	16	8	7200	32 GB	SSD		1,2,3	\$154.01
	B8ms	Standard	General purpose	8	32	16	10800	64 GB	SSD		1,2,3	\$308.02
★	D2s_v3	Standard	General purpose	2	8	4	4000	16 GB	SSD		1,2,3	\$81.84

Prices presented are estimates in your local currency that include only Azure infrastructure costs and any discounts for the subscription and location. The prices don't include any applicable software costs. Recommended sizes are determined by the publisher of the selected image based on hardware and software requirements.

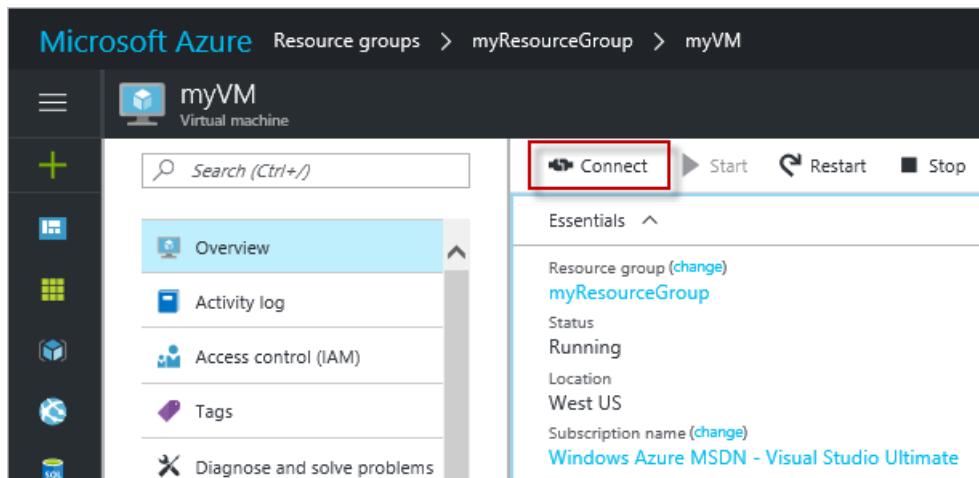
Select

7. Under **Settings**, leave the defaults and select **OK**.
8. On the summary page, select **Create** to start the VM deployment.
9. The VM is pinned to the Azure portal dashboard. Once the deployment has completed, the VM summary automatically opens.

Connect to virtual machine

Create an SSH connection with the VM.

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



The screenshot shows the Azure portal interface. In the top navigation bar, it says "Microsoft Azure" followed by "Resource groups > myResourceGroup > myVM". On the left, there's a sidebar with icons for creating new resources and managing existing ones. The main content area is titled "myVM Virtual machine". A search bar is at the top. Below it, there are several cards: "Overview" (which is selected and highlighted in blue), "Activity log", "Access control (IAM)", "Tags", and "Diagnose and solve problems". To the right of these cards is a "Connect" button, which is also highlighted with a red box. Further down, under the heading "Essentials", there are details about the resource group ("myResourceGroup"), status ("Running"), location ("West US"), and subscription information ("Windows Azure MSDN - Visual Studio Ultimate").

2. In the **Connect to virtual machine** page, keep the default options to connect by DNS name 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@myvm-123abc.eastus.cloudapp.azure.com
```

3. Paste the SSH connection command into a shell, such as the Azure Cloud Shell or Bash on Ubuntu on Windows to create the connection.

Install web server

To see your VM in action, install the NGINX web server. To update package sources and install the latest NGINX package, run the following commands from your SSH session:

```
# update packages
sudo apt-get -y update

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

When done, **exit** the SSH session and return to the VM properties in the Azure portal.

Open port 80 for web traffic

A Network Security Group (NSG) secures inbound and outbound traffic. When a VM is created from the Azure portal, an inbound rule is created on port 22 for SSH connections. Because this VM hosts a web server, an NSG rule needs to be created for port 80.

1. On the VM overview page, select **Networking**.
2. The list of existing inbound and outbound rules are shown. Choose to **Add inbound port rule**.
3. Select the **Basic** option across the top, then choose *HTTP* from the list of available services. Port 80, a priority, and name, are provided for you.
4. To create the rule, select **Add**.

View the web server in action

With NGINX installed and port 80 open to your VM, the web server can now be accessed from the internet. Open

a web browser, and enter the public IP address of the VM. The public IP address can be found on the VM overview page, or at the top of the *Networking* page where you add the inbound port rule.



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

4/25/2018 • 5 min 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 that runs Ubuntu. To see your VM in action, you then 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. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are a few ways to launch the Cloud Shell:

Click Try It in the upper right corner of a code block.	
Open Cloud Shell in your browser.	
Click the Cloud Shell button on the menu in the upper right of the Azure portal.	

If you choose to install and use the PowerShell locally, this tutorial requires the Azure PowerShell module version 5.7.0 or later. Run `Get-Module -ListAvailable AzureRM` to find the version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzureRmAccount` to create a connection with Azure.

Finally, a public SSH key with the name `id_rsa.pub` needs to be stored in the `.ssh` directory of your Windows user profile. For detailed information on how to create and use SSH keys, see [Create SSH keys for Azure](#).

Create a resource group

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

```
New-AzureRmResourceGroup -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-AzureRmVirtualNetworkSubnetConfig -Name "mySubnet" -AddressPrefix 192.168.1.0/24

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

# Create a public IP address and specify a DNS name
$pip = New-AzureRmPublicIpAddress -ResourceGroupName "myResourceGroup" -Location "EastUS" ` 
-AllocationMethod Static -IdleTimeoutInMinutes 4 -Name "mypublicdns$(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-AzureRmNetworkSecurityRuleConfig -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-AzureRmNetworkSecurityRuleConfig -Name "myNetworkSecurityGroupRuleWWW" -Protocol "Tcp" ` 
-Direction "Inbound" -Priority 1001 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * ` 
-DestinationPortRange 80 -Access "Allow"

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

```

Create a virtual network interface card (NIC) with [New-AzureRmNetworkInterface](#). 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-AzureRmNetworkInterface -Name "myNic" -ResourceGroupName "myResourceGroup" -Location "EastUS" ` 
-SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $pip.Id -NetworkSecurityGroupId $nsg.Id

```

Create a virtual machine

A virtual machine configuration includes the settings that are used when a VM is deployed such as a VM image, size, and authentication options. Define the SSH credentials, OS information, and VM size as follows:

```

# 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-AzureRmVMConfig -VMName "myVM" -VMSize "Standard_D1" | ` 
Set-AzureRmVMOperatingSystem -Linux -ComputerName "myVM" -Credential $cred -DisablePasswordAuthentication | ` 
Set-AzureRmVMSourceImage -PublisherName "Canonical" -Offer "UbuntuServer" -Skus "16.04-LTS" -Version "latest" | ` 
Add-AzureRmVMNetworkInterface -Id $nic.Id

# Configure SSH Keys
$sshPublicKey = Get-Content "$env:USERPROFILE\.ssh\id_rsa.pub"
Add-AzureRmVMSShPublicKey -VM $vmConfig -KeyData $sshPublicKey -Path "/home/azureuser/.ssh/authorized_keys"

```

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

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

Connect to virtual machine

After the deployment has completed, SSH to the VM. To see your VM in action, the NGINX web server is then installed.

To see the public IP address of the VM, use the [Get-AzureRmPublicIpAddress cmdlet](#):

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

Use an SSH client to connect to the VM. You can use the Azure Cloud Shell from a web browser, or if you use Windows, you can use [Putty](#) or the [Windows Subsystem for Linux](#). Provide the public IP address of your VM:

```
ssh azureuser@IpAddress
```

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 web server

To see your VM in action, install the NGINX web server. To update package sources and install the latest NGINX package, run the following commands from your SSH session:

```
# update packages
sudo apt-get -y update

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

When done, [exit](#) the SSH session

View the web server in action

With NGINX installed and port 80 now open on your VM from the Internet, use a web browser of your choice to view the default NGINX welcome page. Use the public IP address of your VM obtained in a previous step. The following example shows the default NGINX web site:



Clean up resources

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

```
Remove-AzureRmResourceGroup -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.

[Azure Linux virtual machine tutorials](#)

Tutorial: Create and Manage Linux VMs with the Azure CLI 2.0

4/26/2018 • 9 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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-
0000000000/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	Publisher	Sku	Urn
UrnAlias	Version		
WindowsServer	MicrosoftWindowsServer	2016-Datacenter	MicrosoftWindowsServer:WindowsServer:2016-Datacenter:latest
	Win2016Datacenter	latest	
WindowsServer	MicrosoftWindowsServer	2012-R2-Datacenter	MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:latest
	Win2012R2Datacenter	latest	
WindowsServer	MicrosoftWindowsServer	2008-R2-SP1	MicrosoftWindowsServer:WindowsServer:2008-R2-SP1:latest
	Win2008R2SP1	latest	
WindowsServer	MicrosoftWindowsServer	2012-Datacenter	MicrosoftWindowsServer:WindowsServer:2012-Datacenter:latest
	Win2012Datacenter	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	Sizes	Description
General purpose	Dsv3, Dv3, DSv2, Dv2, DS, D, Av2, A0-7	Balanced CPU-to-memory. Ideal for dev / test and small to medium applications and data solutions.
Compute optimized	Fs, F	High CPU-to-memory. Good for medium traffic applications, network appliances, and batch processes.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, DS, Dv2, D	High memory-to-core. Great for relational databases, medium to large caches, and in-memory analytics.
Storage optimized	Ls	High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases.
GPU	NV, NC	Specialized VMs targeted for heavy graphic rendering and video editing.
High performance	H, A8-11	Our most powerful CPU VMs with optional high-throughput network interfaces (RDMA).

Find available VM 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
28672	4	14336	Standard_DS11	2	1047552
57344	8	28672	Standard_DS12	4	1047552
114688	16	57344	Standard_DS13	8	1047552
229376	32	114688	Standard_DS14	16	1047552
20480	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
138240	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 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 2.0

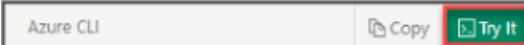
4/26/2018 • 9 min to read • [Edit Online](#)

Azure virtual machines use disks to store the VMs operating system, applications, and data. When creating a VM it is important to choose a disk size and configuration appropriate to the expected workload. This tutorial covers deploying and managing 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

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 1 terabyte, 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** host applications or data. For applications and data, use data disks, which are detailed later in this article.

Temporary disk - Temporary disks use a solid-state drive that is located on the same Azure host as the VM. 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`.

Temporary disk sizes

Type	VM Size	Max Temp Disk Size (GB)
General purpose	A and D series	800
Compute optimized	F series	800
Memory optimized	D and G series	6144
Storage optimized	L series	5630
GPU	N series	1440
High performance	A and H series	2000

Azure data disks

Additional data disks can be added for installing applications and storing data. Data disks should be used in any situation where durable and responsive data storage is desired. Each data disk has a maximum capacity of 1 terabyte. The size of the virtual machine determines how many data disks can be attached to a VM. For each VM vCPU, two data disks can be attached.

Max data disks per VM

Type	VM Size	Max Data Disks per VM
General purpose	A and D series	32
Compute optimized	F series	32
Memory optimized	D and G series	64
Storage optimized	L series	64
GPU	N series	48
High performance	A and H series	32

VM disk types

Azure provides two types of disk.

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. Premium disks come in three types (P10, P20, P30), the size of the disk determines the disk type. When selecting, 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. Anything over 512 GB, the size is a P30.

Premium disk performance

PREMIUM STORAGE DISK TYPE	P10	P20	P30
Disk size (round up)	128 GB	512 GB	1,024 GB (1 TB)
Max IOPS per disk	500	2,300	5,000
Throughput per disk	100 MB/s	150 MB/s	200 MB/s

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

Create and attach disks

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 `--datadisk-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 \
  --admin-username azureuser \
  --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 --vm-name myVM --resource-group myResourceGroupDisk --disk 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](#).

Manual configuration

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

```
ssh 52.174.34.95
```

Partition the disk with `fdisk`.

```
(echo n; echo p; echo 1; 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
```

Resize VM disk

Once a VM has been deployed, the operating system disk or any attached data disks can be increased in size. Increasing the size of a disk is beneficial when needing more storage space or a higher level of performance (P10,

P20, P30). Note, disks cannot be decreased in size.

Before increasing disk size, the Id or name of the disk is needed. Use the [az disk list](#) command to return all disks in a resource group. Take note of the disk name that you would like to resize.

```
az disk list -g myResourceGroupDisk --query '[*].{Name:name,Gb:diskSizeGb,Tier:accountType}' --output table
```

The VM must also be deallocated. Use the [az vm deallocate](#) command to stop and deallocate the VM.

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

Use the [az disk update](#) command to resize the disk. This example resizes a disk named *myDataDisk* to 1 terabyte.

```
az disk update --name myDataDisk --resource-group myResourceGroupDisk --size-gb 1023
```

Once the resize operation has completed, start the VM.

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

If you've resized the operating system disk, the partition is automatically expanded. If you have resized a data disk, any current partitions need to be expanded in the VMs operating system.

Snapshot Azure disks

Taking a disk snapshot creates a read only, point-in-time copy of the disk. Azure VM snapshots are useful for quickly saving the state of a VM before making configuration changes. In the event the configuration changes prove to be undesired, VM state can be restored using the snapshot. When a VM has more than one disk, a snapshot is taken of each disk independently of the others. For taking application consistent backups, consider stopping the VM before taking disk snapshots. Alternatively, use the [Azure Backup service](#), which enables you to perform automated backups while the VM is running.

Create snapshot

Before creating 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 -g 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')].[name]" -o tsv)
```

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

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

4/26/2018 • 8 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 *myAutomatedVM*:

```
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://` in the address bar. Provide your own public IP address from the VM create process. Your Node.js app is displayed as 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 format-secret](#). 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 format-secret --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`

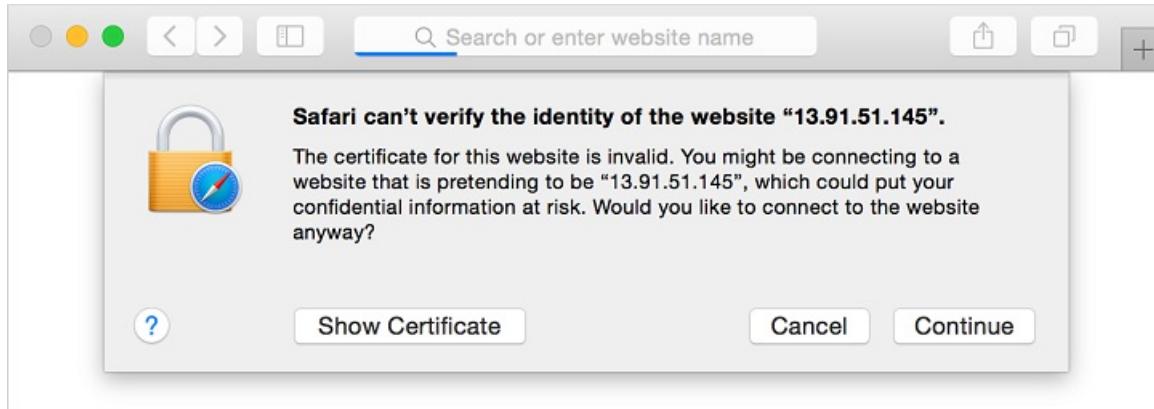
displayed by the Azure CLI. This address is used to access the Node.js app via a web browser.

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://* 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 and Nodejs 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 Nodejs 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 2.0

5/10/2018 • 3 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

Before you begin

The steps below detail how to take an existing VM and turn it into a re-usable 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 2.0

4/26/2018 • 4 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 \
        --image UbuntuLTS \
        --admin-username azureuser \
        --generate-ssh-keys \
        --no-wait
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:

Resource group (change) myResourceGroupAvailability
Location East US
Subscription name (change) Azure
Subscription ID <Subscription ID>

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 2.0

4/26/2018 • 9 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 2.0 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					

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
```

Configure autoscale rules

Rather than manually scaling the number of instances in your scale set, you can define autoscale rules. These rules monitor the instances in your scale set and respond accordingly based on metrics and thresholds you define. The following example scales out the number of instances by one when the average CPU load is greater than 60% over a 5-minute period. If the average CPU load then drops below 30% over a 5-minute period, the instances are scaled in by one instance. Your subscription ID is used to build the resource URLs for the various scale set components. To create these rules with [az monitor autoscale-settings create](#), copy and paste the following autoscale command profile:

```
sub=$(az account show --query id -o tsv)

az monitor autoscale-settings create \
--resource-group myResourceGroupScaleSet \
```

```

--name autoscale \
--parameters '{"autoscale_setting_resource_name": "autoscale",
  "enabled": true,
  "location": "East US",
  "notifications": [],
  "profiles": [
    {
      "name": "Auto created scale condition",
      "capacity": {
        "minimum": "2",
        "maximum": "10",
        "default": "2"
      },
      "rules": [
        {
          "metricTrigger": {
            "metricName": "Percentage CPU",
            "metricNamespace": "",
            "metricResourceUri":
              "/subscriptions/'$sub'/resourceGroups/myResourceGroupScaleSet/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet",
            "metricResourceLocation": "eastus",
            "timeGrain": "PT1M",
            "statistic": "Average",
            "timeWindow": "PT5M",
            "timeAggregation": "Average",
            "operator": "GreaterThan",
            "threshold": 70
          },
          "scaleAction": {
            "direction": "Increase",
            "type": "ChangeCount",
            "value": "1",
            "cooldown": "PT5M"
          }
        },
        {
          "metricTrigger": {
            "metricName": "Percentage CPU",
            "metricNamespace": "",
            "metricResourceUri":
              "/subscriptions/'$sub'/resourceGroups/myResourceGroupScaleSet/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet",
            "metricResourceLocation": "eastus",
            "timeGrain": "PT1M",
            "statistic": "Average",
            "timeWindow": "PT5M",
            "timeAggregation": "Average",
            "operator": "LessThan",
            "threshold": 30
          },
          "scaleAction": {
            "direction": "Decrease",
            "type": "ChangeCount",
            "value": "1",
            "cooldown": "PT5M"
          }
        }
      ]
    }
  ],
  "tags": {},
  "target_resource_uri":
    "/subscriptions/'$sub'/resourceGroups/myResourceGroupScaleSet/providers/Microsoft.Compute/virtualMachineScaleSets/myScaleSet"
}'
```

To reuse the autoscale profile, you can create a JSON (JavaScript Object Notation) file and pass that to the

```
az monitor autoscale-settings create
```

 command with the `--parameters @autoscale.json` parameter. For more design information on the use of autoscale, see [autoscale best practices](#).

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 2.0

4/26/2018 • 10 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 2.0

4/26/2018 • 10 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select **Try It** in the upper-right corner of a code block.



Open Cloud Shell in your browser.



Select the **Cloud Shell** button on the menu in the upper-right corner of the [Azure portal](#).

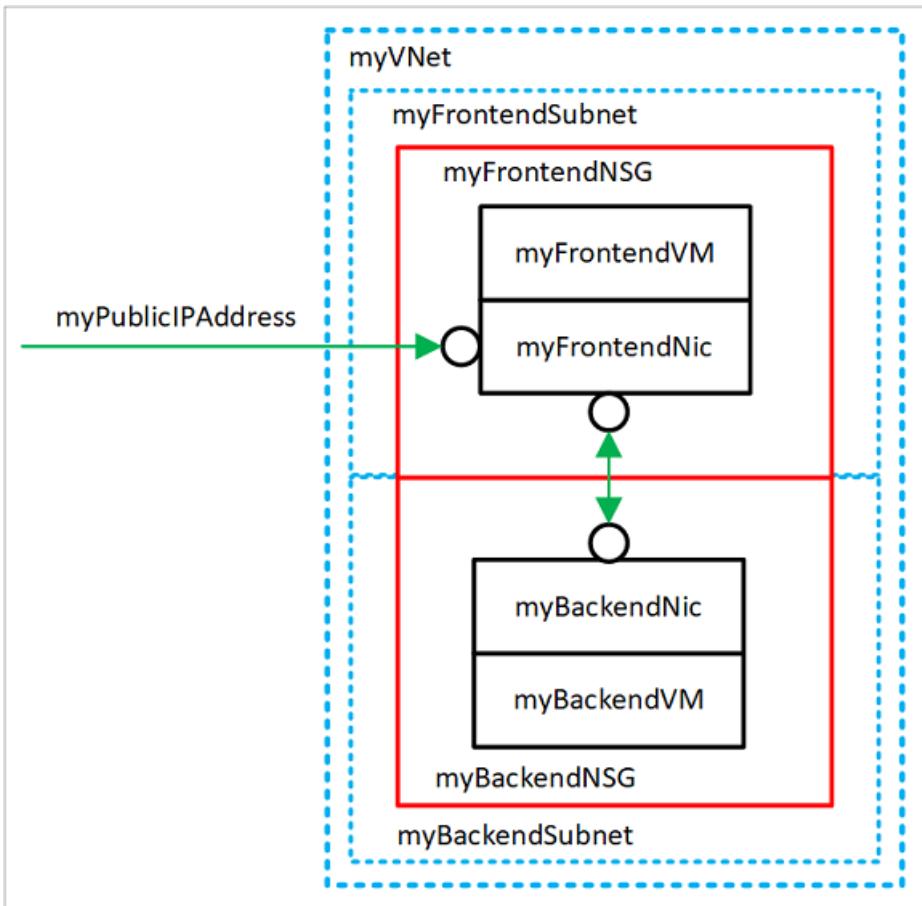


If you choose to install and use the CLI locally, this tutorial requires that you are running the Azure CLI version 2.0.30 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI 2.0](#).

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 to 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 VM's 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 min 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).

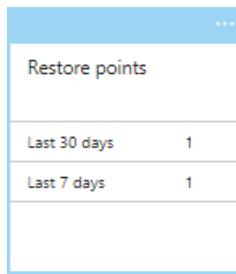
By default, Azure Backup takes a file system consistent backup for Linux VM but it can be configured to take [application consistent backup using pre-script and post-script framework](#). Once the Azure Backup service takes the snapshot, the data is transferred to the vault. To maximize efficiency, the 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 2.0

4/26/2018 • 11 min 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?
- 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

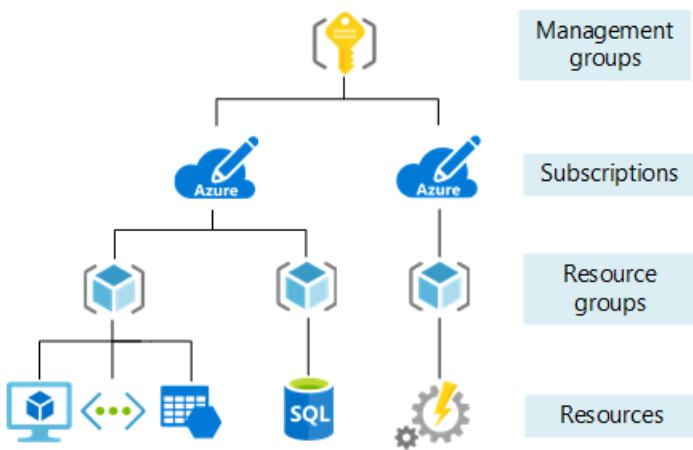
Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 [create an Azure Active Directory group](#) for users who need to take similar actions. Then, assign that group to the appropriate role. To simplify this article, you create an Azure Active Directory group without members. You can still assign this group to a role for a scope.

The following example creates an Azure Active Directory group named `VMDemoContributors` with a mail nickname of `vmDemoGroup`. The mail nickname serves as an alias for the group.

```
adgroupId=$(az ad group create --display-name VMdemоСontributors --mail-nickname vmDemoGroup --query objectId --output tsv)
```

It takes a moment after the command prompt returns for the group to propagate throughout Azure Active Directory. After waiting for 20 or 30 seconds, 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. If you run the following command before it has propagated, you receive an error stating **Principal does not exist in the directory**. Try running the command again.

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

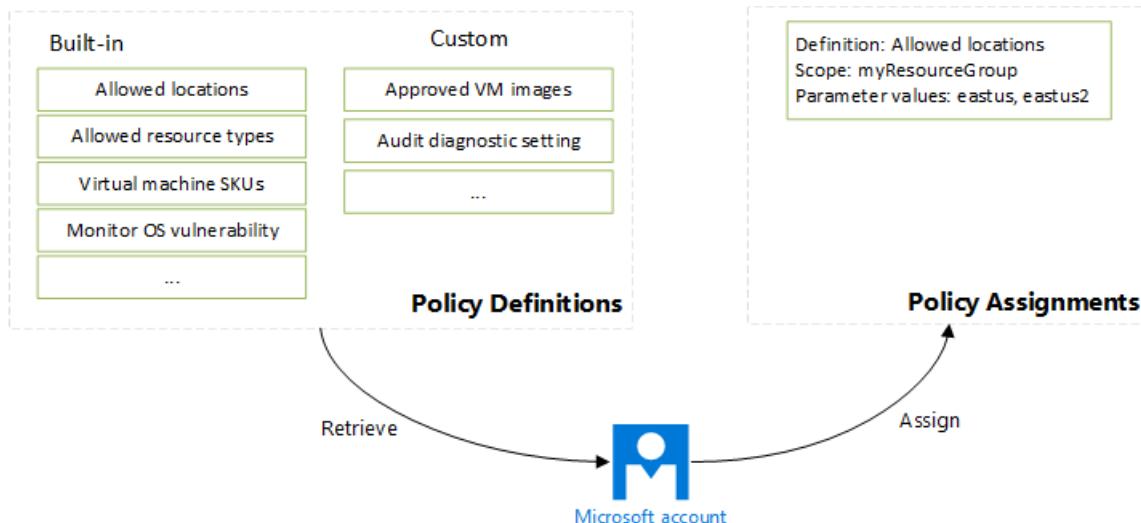
Typically, you repeat the process for *Network Contributor* and *Storage Account Contributor* to make sure users are assigned to manage the deployed resources. In this article, you can skip those steps.

Azure policies

[Azure policies](#) help you make sure all resources in subscription meet corporate standards. Use policies to reduce your costs by restricting deployment options to only those resource types and SKUs that are approved. You define rules and actions for your resources and those rules are automatically enforced during deployment. For example, you can control the types of resources that are deployed. Or, you can restrict the approved locations for resources. Some policies deny an action, and some policies set up auditing of an action.

Policy is complementary to role-based access control (RBAC). RBAC focuses on user access, and is a default deny and explicit allow system. Policy focuses on resource properties during and after deployment. It's a default allow and explicit deny system.

There are two concepts to understand with policies - *policy definitions* and *policy assignments*. A policy definition describes the management conditions you want to enforce. A policy assignment puts a policy definition into action for a particular scope.



Azure provides several built-in policy definitions you can use without any modification. You pass parameter values to specify the values that are permitted in your scope. If built-in policy definition don't fulfill your requirements, you can [create custom policy definitions](#).

Apply policies

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 do not 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 do not 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 cannot be performed 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 are not 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
```

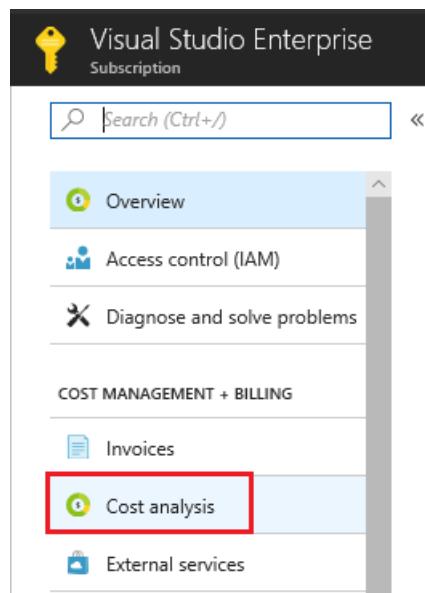
You can use the returned values for management tasks like stopping all virtual machines with a tag value.

```
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**.

→ Costs by service

For more cost management and optimization capabilities, try Azure Cost Management →

Subscription	Resource type	Resource group
Visual Studio Enterprise	3 selected	2 selected
Timespan	Tag	
Current period	Environment: Test	
Apply	Download	
 There is a delay between the time when resources are reported here and when they are actually reached the billing system. Due to this, costs may be delayed. Amounts shown here do not include recent usage. Taxes are not included.		
Total cost		
0.24 USD		

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

4/26/2018 • 13 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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*:

```
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
```

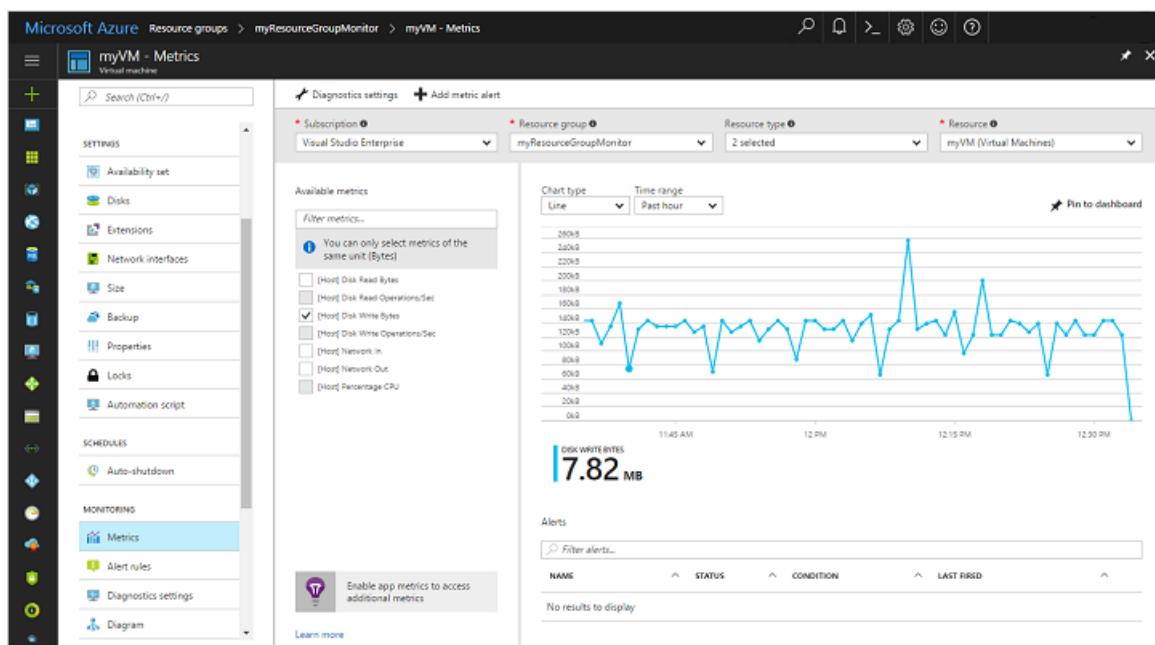
You can get the boot diagnostic data for *myVM* with the `az vm boot-diagnostics get-boot-log` command as follows:

```
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, click **Resource Groups**, select **myResourceGroupMonitor**, and then select **myVM** in the resource list.
2. To see how the host VM is performing, click **Metrics** on the VM blade, then select any of the *[Host]* metrics under **Available metrics**.



Install diagnostics extension

IMPORTANT

This document describes version 2.3 of the Linux Diagnostic Extension, which has been deprecated. Version 2.3 will be supported until June 30, 2018.

Version 3.0 of the Linux Diagnostic Extension can be enabled instead. For more information, see [the documentation](#).

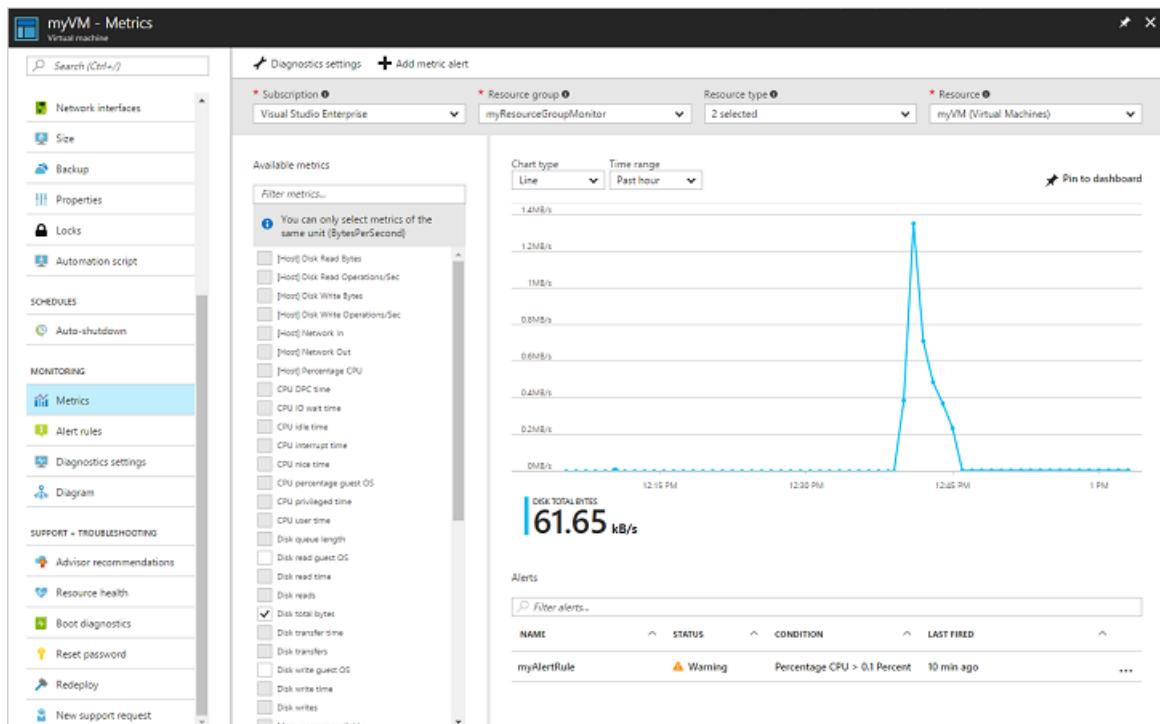
The basic host metrics are available, but to see more granular and VM-specific metrics, you need to install the Azure diagnostics extension on the VM. The Azure diagnostics extension allows additional monitoring and diagnostics data to be retrieved from the VM. You can view these performance metrics 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, click **Resource Groups**, select **myResourceGroup**, and then select **myVM** in the resource list.
2. Click **Diagnosis settings**. The list shows that *Boot diagnostics* are already enabled from the previous section. Click the check box for *Basic metrics*.
3. In the *Storage account* section, browse to and select the *mydiagdata[1234]* account created in the previous section.
4. Click the **Save** button.

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, click **Resource Groups**, select **myResourceGroup**, and then select **myVM** in the resource list.
2. To see how the VM is performing, click **Metrics** on the VM blade, and then select any of the 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, click **Resource Groups**, select **myResourceGroup**, and then select **myVM** in the resource list.
2. Click **Alert rules** on the VM blade, then click **Add metric alert** across the top of the alerts blade.
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. Click the **OK** button.

Manage package 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, click **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 Microsoft Monitoring Agent (MMA) 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 click **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:

- [Log Analytics workspace](#)
- [Automation](#)
- 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 click **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.

Home > Resource groups > myResourceGroupMonitor > myVM - Update management

myVM - Update management

Virtual machine

Search (Ctrl+ /)

Automation script

OPERATIONS

- Auto-shutdown
- Backup
- Disaster recovery (Preview)
- Update management**
- Inventory
- Change tracking

MONITORING

- Metrics
- Alert rules

Update Management

Enable consistent control and compliance of this VM with Update Management.

This service is included with Azure virtual machines. You only pay for logs stored in Log Analytics.

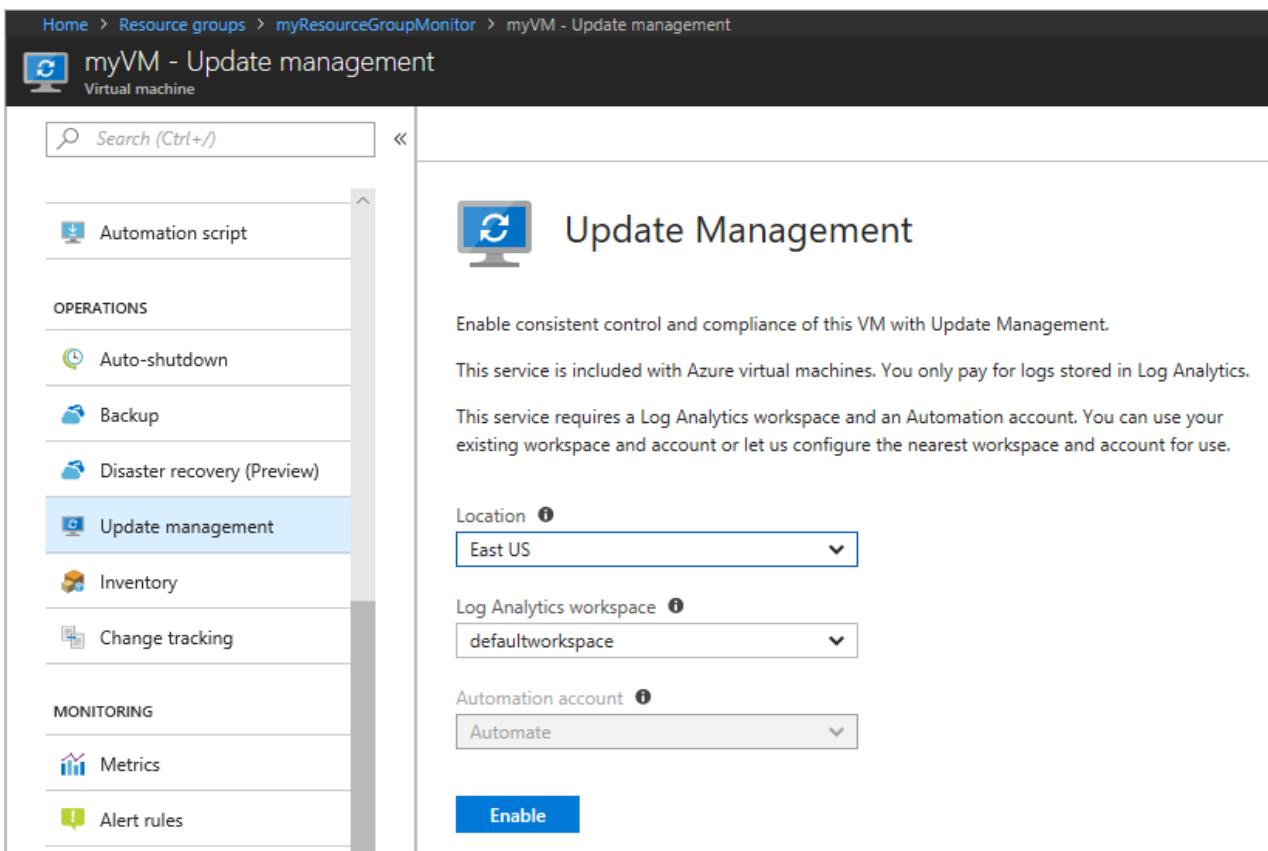
This service requires a Log Analytics workspace and an Automation account. You can use your existing workspace and account or let us configure the nearest workspace and account for use.

Location: East US

Log Analytics workspace: defaultworkspace

Automation account: Automate

Enable



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 Log Analytics. 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.

Compliance	Missing updates (57)	Failed update deployments	Learn more
!	Critical 0 Security 2 Others 55	0 ✓ out of 1 in the past six months	Update Management for Linux Provide feedback
Missing updates (57) Update deployments Scheduled update deployments			
Filter by name		Classifications: All	
UPDATE NAME	CLASSIFICATION	INFORMATION LINK	
curl	⚠ Security updates		
sensible-utils	⚠ Security updates		
apparmor	⚠ Others		
apt	⚠ Others		
apt-transport-https	⚠ Others		
apt-utils	⚠ Others		
base-files	⚠ Others		
cloud-init	⚠ Others		
dpkg	⚠ Others		

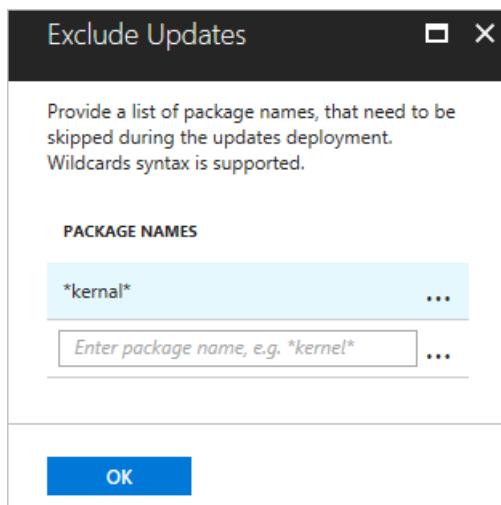
Schedule an update deployment

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.

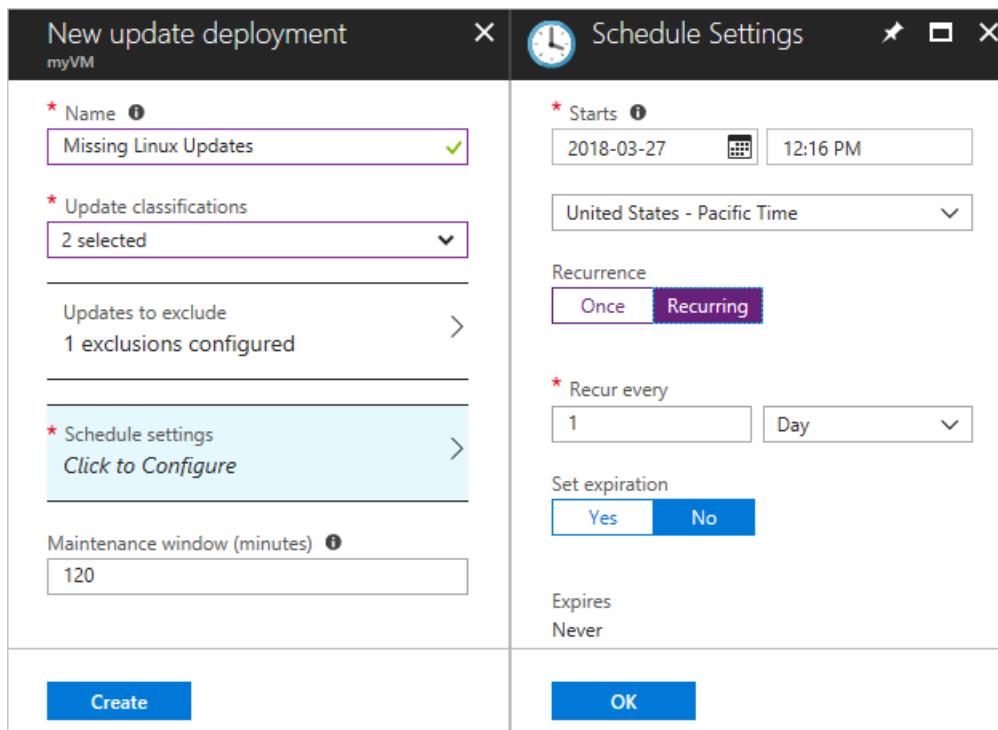
Schedule a new Update Deployment for the VM by clicking **Schedule update deployment** at the top of the

Update management screen. In the **New update deployment** screen, specify the following information:

- **Name** - Provide a unique name to identify the update deployment.
- **Update classification** - Select the types of software the update deployment included in the deployment. The classification types are:
 - Critical and security updates
 - Other updates
- **Updates to Exclude** - You can provide a list of package names that should be skipped during the update deployment. Package names support wildcards (such as, *kernel*).



- **Schedule settings** - You can either accept the default date and time, which is 30 minutes after current time, or specify a different time. You can also specify whether the deployment occurs once or set up a recurring schedule. Click the Recurring option under Recurrence to set up a recurring schedule.



- **Maintenance window (minutes)** - Specify the period of time you want the update deployment to occur within. This helps ensure changes are performed within your defined service windows.

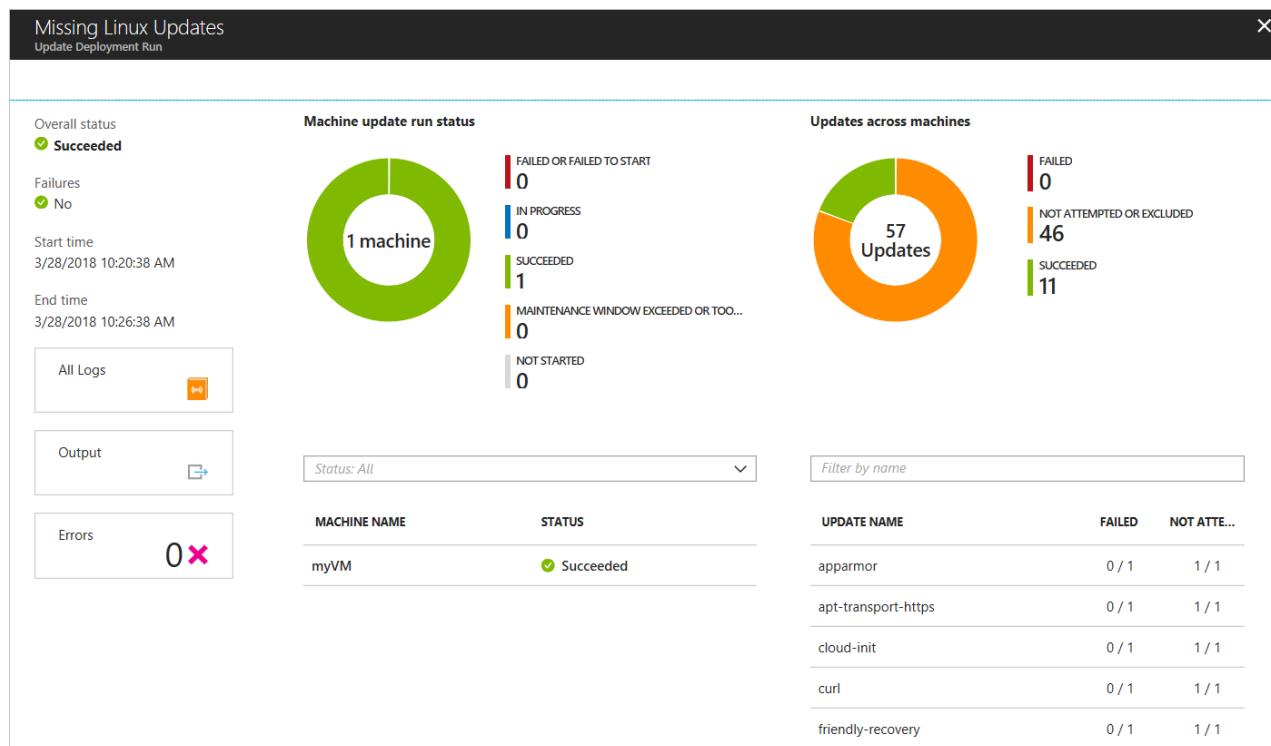
After you have completed configuring the schedule, click **Create** button and you return to the status dashboard. Notice that the **Scheduled** table shows the deployment schedule you created.

WARNING

For updates that require a reboot, the VM is restarted automatically.

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**. Click the completed update deployment to see the dashboard for that update deployment.



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

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

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

Click **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, click **Inventory** or **Change tracking**. The **Enable Change Tracking and Inventory** screen opens.

Configure the location, Log analytics workspace and Automation account to use and click **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.

The screenshot shows the Azure portal interface for managing a virtual machine named 'myVM'. The left sidebar has a navigation tree with 'myVM - Inventory' selected. The main content area is titled 'Inventory' and contains the following information:

- Enable consistent control and compliance of this VM with Change Tracking and Inventory.**
- This service is included with Azure virtual machines. You only pay for logs stored in Log Analytics.
- This service requires a Log Analytics workspace and an Automation account. You can use your existing workspace and account or let us configure the nearest workspace and account for use.

Below this, there are three dropdown menus for configuration:

- Location**: Set to 'East US'.
- Log Analytics workspace**: Set to 'defaultworkspace'.
- Automation account**: Set to 'Automate'.

A large blue 'Enable' button is located at the bottom of the configuration area.

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**. Click **Edit Settings**, the **Change Tracking** page is displayed. Select the type of setting you want to track and then click **+ 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.

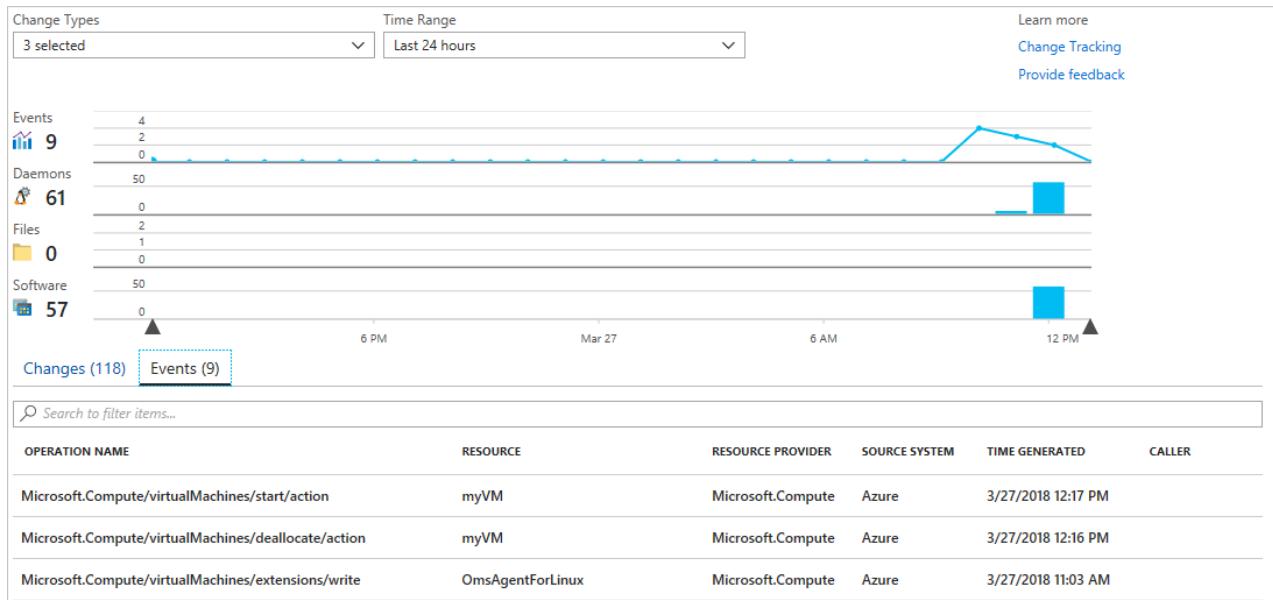
NAME	VERSION	PUBLISHER	LAST REFRESHED TIME
accountsservice	0.6.40-2ubuntu11.3	Ubuntu Developers <ubun...	3/27/2018 11:51 AM
acl	2.2.52-3	Ubuntu Developers <ubun...	3/27/2018 11:51 AM
acpid	1:2.0.26-1ubuntu2	Ubuntu Developers <ubun...	3/27/2018 11:51 AM
adduser	3.113+nmu3ubuntu4	Ubuntu Core Developers <...	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 the solutions like Update Management and Change and Inventory provided by [Azure Automation](#).

When you have access to the Log Analytics workspace, you can find the workspace key and workspace identifier on by selecting **Advanced settings** under **SETTINGS**. Replace <workspace-key> and <workspace-id> with the

values for from your Log Analytics workspace and then you can use **az vm extension set** to add the extension to the VM:

```
az vm extension set \
--resource-group myResourceGroupMonitor \
--vm-name myVM \
--name OmsAgentForLinux \
--publisher Microsoft.EnterpriseCloud.Monitoring \
--version 1.3 \
--protected-settings '{"workspaceKey": "<workspace-key>"}' \
--settings '{"workspaceId": "<workspace-id>"'}
```

After a few minutes, you should see the new VM in the Log Analytics workspace.

The screenshot shows a Log Analytics workspace interface. At the top, there is a query editor with the following DQL code:

```
Heartbeat
| where OSType == "Linux"
| summarize arg_max(TimeGenerated, *) by SourceComputerId | top 500000 by Computer asc | render table
```

Below the query editor is a results table. The table has a header row with columns: TimeGenerated, ComputerIP, Computer, Category, OSType, OSName, OSMajorVersion, OSMinorVersion, and Version. There is one data row shown:

TimeGenerated	ComputerIP	Computer	Category	OSType	OSName	OSMajorVersion	OSMinorVersion	Version
3/27/2018 12:44:33.793 PM	myVM	Direct Agent	Linux	Ubuntu	16	04		1.4.4-210

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

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

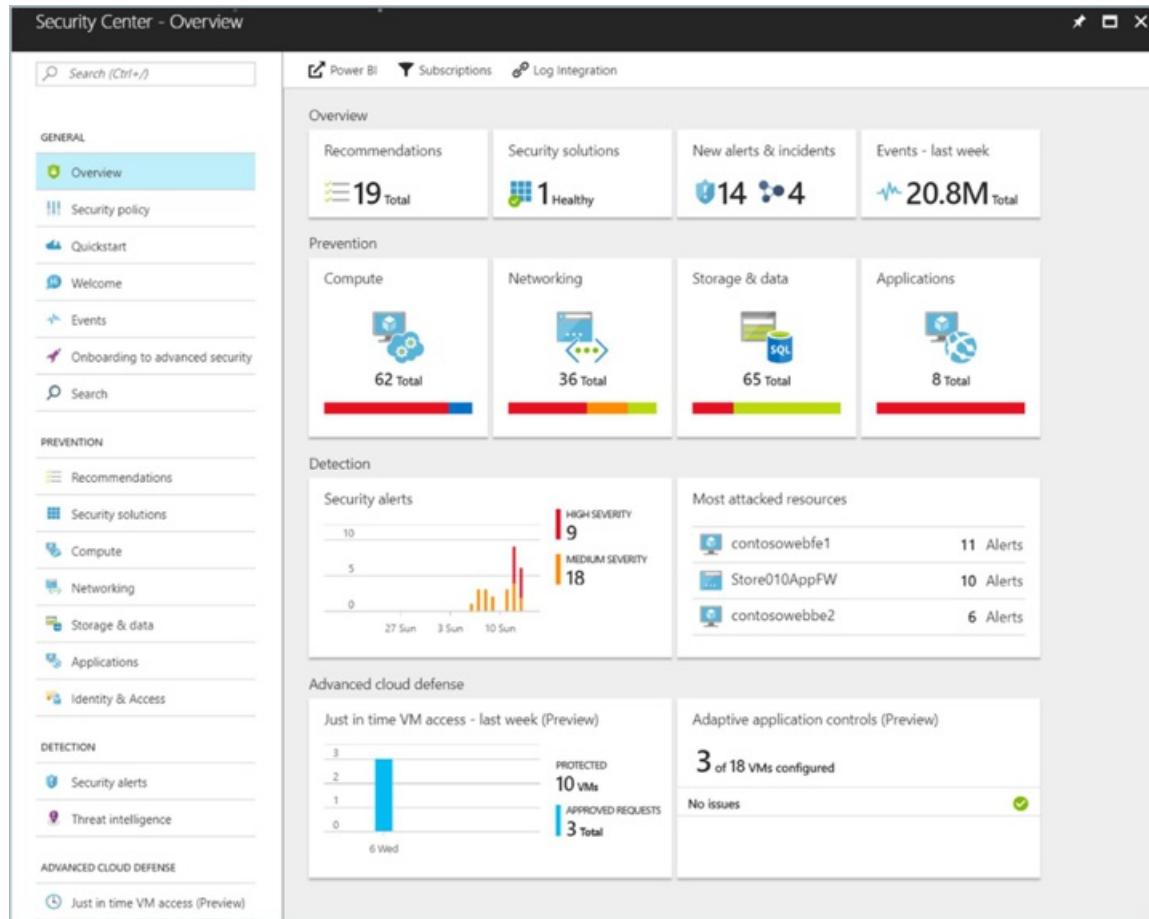
Azure Security Center can help you gain visibility into your Azure resource security practices. Security Center offers integrated security monitoring. It can detect threats that otherwise might go unnoticed. In this tutorial, you learn about Azure Security Center, and how to:

- 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 and creating an Azure storage account to hold collected data.

1. On the Security Center dashboard, click **Security policy**, and then select your subscription.
2. For **Data collection**, select **On**.
3. To create a storage account, select **Choose a storage account**. Then, select **OK**.
4. On the **Security Policy** blade, select **Save**.

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 all Azure resources:

1. On the Security Center dashboard, select **Security policy**, and then select your subscription.
2. Select **Prevention policy**.
3. Turn on or turn off policy items that you want to apply to all Azure resources.
4. When you're finished selecting your settings, select **OK**.
5. On the **Security policy** blade, select **Save**.

To set up a policy for a specific resource group:

1. On the Security Center dashboard, select **Security policy**, and then select a resource group.
2. Select **Prevention policy**.
3. Turn on or turn off policy items that you want to apply to the resource group.
4. Under **INHERITANCE**, select **Unique**.

5. When you're finished selecting your settings, select **OK**.

6. On the **Security policy** blade, select **Save**.

You also can turn off data collection for a specific resource group on this page.

In the following example, a unique policy has been created for a resource group named *myResourceGroup*. In this policy, disk encryption and web application firewall recommendations are turned off.

The screenshot displays three windows side-by-side:

- Left Window:** 'Security policy' (Define policy per subscription or resource group). It lists four resource groups: 'Free Trial' (Inheritance: ..., Data Collection: On), 'myResourceGroup' (Inheritance: Unique, Data Collection: On), 'RDPBruteForce' (Inheritance: Inherited, Data Collection: On), and 'securitydata' (Inheritance: Inherited, Data Collection: On).
- Middle Window:** 'Security policy' (myResourceGroup). It shows 'Inheritance' settings (Inherit policy settings from subscription or define this resource group as unique) and a 'Data collection' section where 'Collect data from virtual machines' is set to 'Off'. It also includes a note about choosing a storage account per region.
- Right Window:** 'Prevention policy' (Free Trial). This is the detailed configuration window. It lists various security features with 'On' or 'Off' toggle switches:
 - Show recommendations for:
 - System updates: On
 - OS vulnerabilities: On
 - Endpoint protection: On
 - Disk encryption: Off
 - Network security groups: On
 - Web application firewall: On
 - Next generation firewall: On
 - Vulnerability Assessment: On
 - Storage Encryption: On
 - SQL auditing & Threat detection: On
 - SQL Encryption: On

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 **Resource security health**, select **Compute**.
2. On the **Compute** blade, select **Virtual machines**. This view provides a summary of the configuration status for all your VMs.

NAME	MONITORED	SYSTEM UPDATES	ENDPOINT PROTE...	VULNERABILITIES	DISK ENCRYPTION
2016OMSAA	✓	!	!	⚠	!
asctest	✓	!	●	!	!
2016OMSLinux	✓	✓	●	!	!

To see all recommendations for a VM, select the VM. Recommendations and remediation are covered in more detail in the next section of this tutorial.

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 a specific 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.

asctest-nsg

Edit inbound rules

Network security group info

NETWORK SECURITY GROUP asctest-nsg

LOCATION westus

DESCRIPTION Your NSG has inbound rules that open access to 'Any' which might enable attackers to access your resources. We recommend that you edit the below inbound rules, to restrict access to sources, who actually access it.

Related inbound rules

PRIORITY	NAME	SOURCE	SERVICE	ACTIONS
1000	default-allow-ssh	*	TCP	Allow

Associated with

NAME	VIRTUAL MACHINE
 asctest929	asctest

As recommendations are remediated, they are marked as resolved.

View detected threats

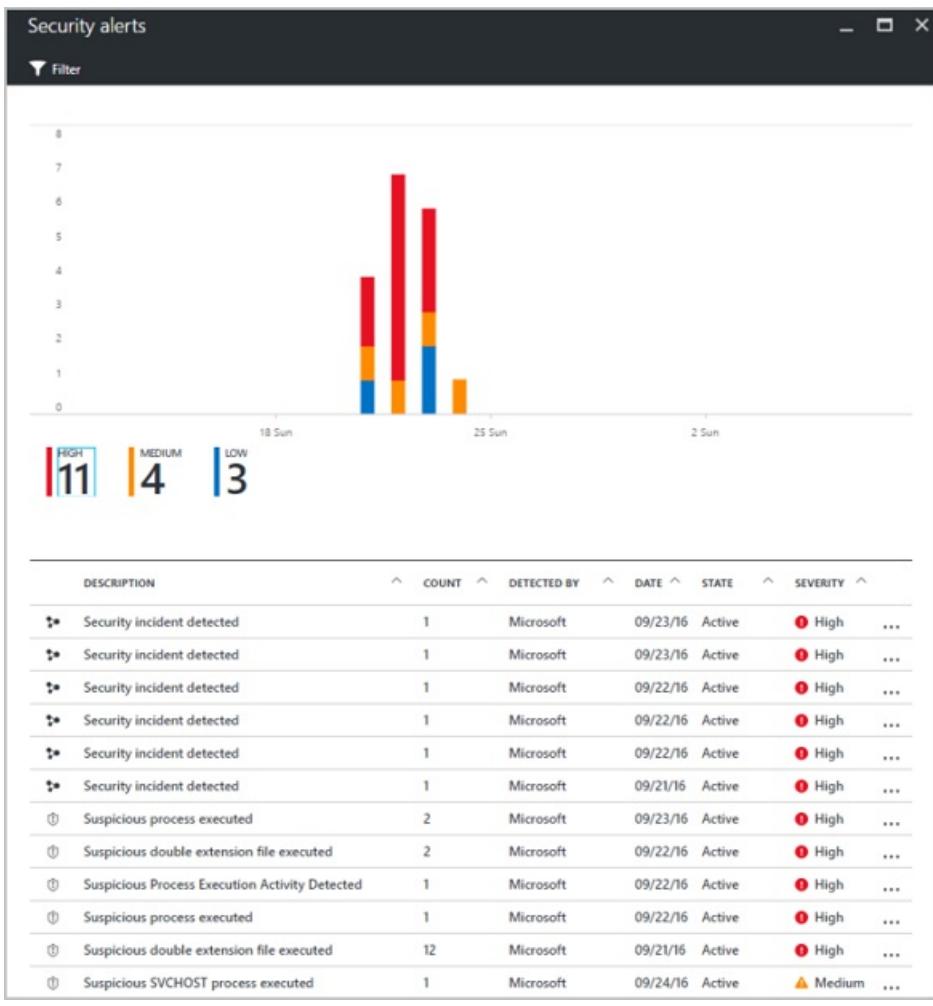
In addition to resource configuration recommendations, Security Center displays threat detection alerts. The security alerts feature aggregates data collected from each VM, Azure networking logs, and connected partner solutions to detect security threats against Azure resources. For in-depth information 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 30-day **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 the new tier, and then select **Select**.
4. On the **Security policy** blade, select **Save**.

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

4/26/2018 • 8 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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:
  - wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | apt-key add -
  - sh -c 'echo deb http://pkg.jenkins-ci.org/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>
```

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.
- 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 **Integrations & services** on the left-hand side.
- Choose **Add service**, then enter *Jenkins* in filter box.
- Select *Jenkins (GitHub plugin)*
- For the **Jenkins hook URL**, enter `http://<publicIps>:8080/github-webhook/`. Make sure you include the trailing `/`
- Select **Add service**

The screenshot shows a GitHub repository page for 'nodejs-docs-hello-world'. The 'Integrations & services' section is selected in the sidebar. Under the 'Services' tab, the 'Add Jenkins (GitHub plugin)' option is chosen. The Jenkins service configuration includes a 'Jenkins hook url' input field containing 'http://13.82.180.34:8080/github-webhook/'. A checked 'Active' checkbox is present with the note 'We will run this service when an event is triggered.' A green 'Add service' button is at the bottom.

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/iainfoulds/nodejs-docs-hello-world>
- Under the **Source code management** section, select **Git**, enter your forked repo .git URL, such as <https://github.com/iainfoulds/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 "Testing"` 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 **Testing** 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 Visual Studio Team Services.

[Deploy apps with Jenkins and Team Services](#)

Tutorial: Deploy your app to Linux virtual machines in Azure with using Jenkins and Visual Studio Team Services

4/26/2018 • 7 min to read • [Edit Online](#)

Continuous integration (CI) and continuous deployment (CD) form a pipeline by which you can build, release, and deploy your code. Visual Studio Team 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 Team 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 Team Services or Team Foundation Server 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 Team Services integration.
- Create a Jenkins service endpoint.
- Create a deployment group for the Azure virtual machines.
- Create a Team Services release definition.
- 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 Team Services account (<https://{{youraccount}}.visualstudio.com>). You can get a [free Team Services account](#).

NOTE

For more information, see [Connect to Team 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 search bar with the text 'nodejs'. Below the search bar, there are tabs: 'Updates' (disabled), 'Available' (selected), 'Installed', and 'Advanced'. A red box highlights the 'Available' tab. Underneath, a table lists a single plugin: 'NodeJS Plugin'. The table has columns for 'Name' and 'Description'. The 'Name' column shows 'NodeJS Plugin' and the 'Description' column shows 'Executes NodeJS script as a build step.' At the bottom of the table, there are two buttons: 'Install without restart' (highlighted with a red box) and 'Download now and install after restart'. To the right of these buttons, there's a link 'Update information obtain from central'.

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 Team 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 Team Services.

- Create a PAT in your Team Services account if you don't already have one. Jenkins requires this information to access your Team Services account. 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 VSTS and TFS?](#).
- 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 Team Services account, such as `https://{your-account-name}.visualstudio.com`.
 - Enter the **Team Project** name.
 - Choose a name for the release definition. (You create this release definition later in Team Services.)
 - Choose credentials to connect to your Team Services or Team Foundation Server environment:
 - Leave **Username** blank if you are using Team 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://adventureworks.visualstudio.com"/>
Team project	<input type="text" value="AW"/>
Release definition	<input type="text" value="Adventureworks 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 Team Services to connect to Jenkins.

- Open the **Services** page in Team Services, open the **New Service Endpoint** list, and select **Jenkins**.

The screenshot shows the 'Services' page in the Team Services interface. The top navigation bar has tabs for 'Fabrikam' (selected), 'Home', 'Code', 'Work', 'Build & Release', 'Test', and a gear icon. Below the tabs, there are links for 'Overview', 'Work', 'Security', 'Version Control', 'Policies', 'Agent queues', 'Notifications', 'Service Hooks', and 'Services' (which is currently selected). The main content area is titled 'Endpoints' and shows a list of service endpoints. A dropdown menu labeled '+ New Service Endpoint' is open, with 'Jenkins' highlighted by a red box. Other options in the list include 'Generic', 'GitHub', and 'Kubernetes'.

- Enter a name for the connection.
- 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>.
- Enter the username and password for your Jenkins account.
- Select **Verify connection** to check that the information is correct.
- Select **OK** to create the service endpoint.

Create a deployment group for Azure virtual machines

You need a [deployment group](#) to register the Team Services agent so the release definition 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*.

- Open the **Releases** tab of the **Build & Release** hub, open **Deployment groups**, and select **+ New**.
- Enter a name for the deployment group, and an optional description. Then select **Create**.
- Choose the operating system for your deployment target virtual machine. For example, select **Ubuntu 16.04+**.
- Select **Use a personal access token in the script for authentication**.
- 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 Team Services, check for your newly registered virtual machine in **Targets** under **Deployment Groups**.

Create a Team Services release definition

A release definition specifies the process that Team Services uses to deploy the app. In this example, you execute a shell script.

To create the release definition in Team Services:

1. Open the **Releases** tab of the **Build & Release** hub, and select **Create release definition**.
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 definition 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 definition.

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 Team Services or Team Foundation Server. Open the release to see the deployment taking place. Congratulations!

Next steps

In this tutorial, you automated the deployment of an app to Azure by using Jenkins for build and Team Services for release. You learned how to:

- Build your app in Jenkins.
- Configure Jenkins for Team Services integration.
- Create a deployment group for the Azure virtual machines.

- Create a release definition 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

4/26/2018 • 6 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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.

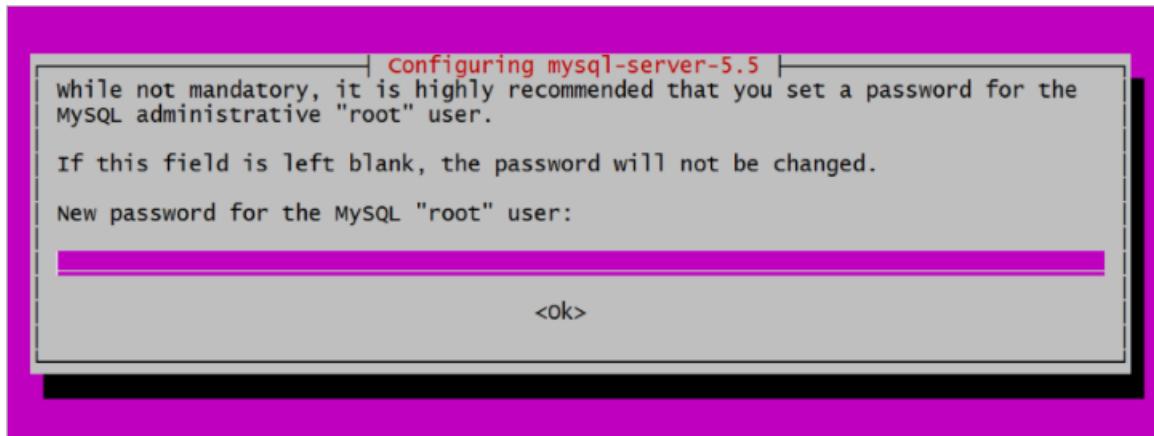
```
ssh azureuser@40.68.254.142
```

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. When prompted, set a root password for MySQL, and then [Enter] to continue. Follow the remaining prompts. This process installs the minimum required PHP extensions needed to use PHP with MySQL.



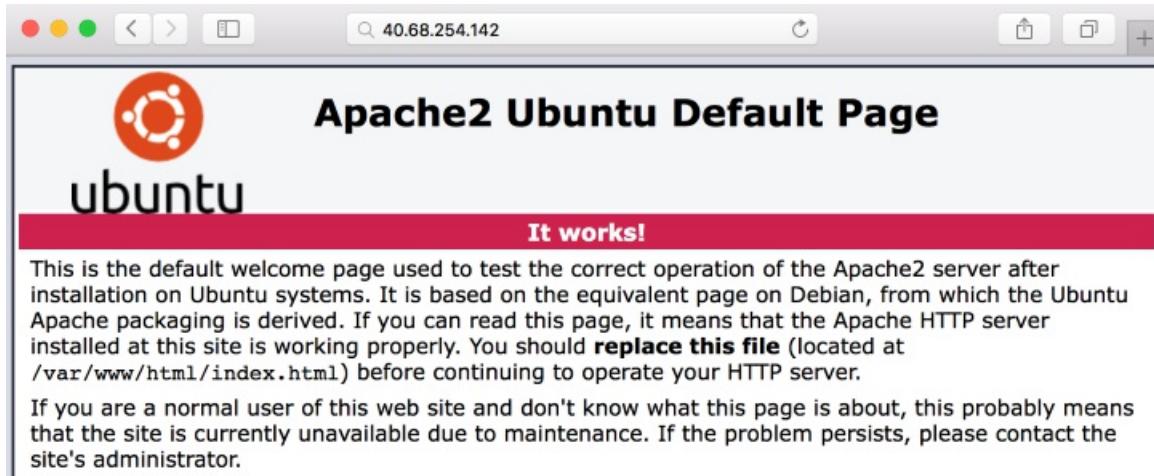
Verify installation and configuration

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:



MySQL

Check the version of MySQL with the following command (note the capital `V` parameter):

```
mysql -V
```

To help secure the installation of MySQL, run the `mysql_secure_installation` script. If you are only setting up a temporary server, you can skip this step.

```
mysql_secure_installation
```

Enter a root password for MySQL, and configure the security settings for your environment.

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.

```
mysql -u root -p
```

When done, exit the mysql prompt by typing `\q`.

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.

PHP Version 7.0.18-0ubuntu0.16.04.1	
System	Linux d 4.4.0-83-generic #106-Ubuntu SMP Mon Jun 26 17:54:43 UTC 2017 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlind.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-finfo.ini, /etc/php/7.0/apache2/conf.d/20-fp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API20151012.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');
?>
```

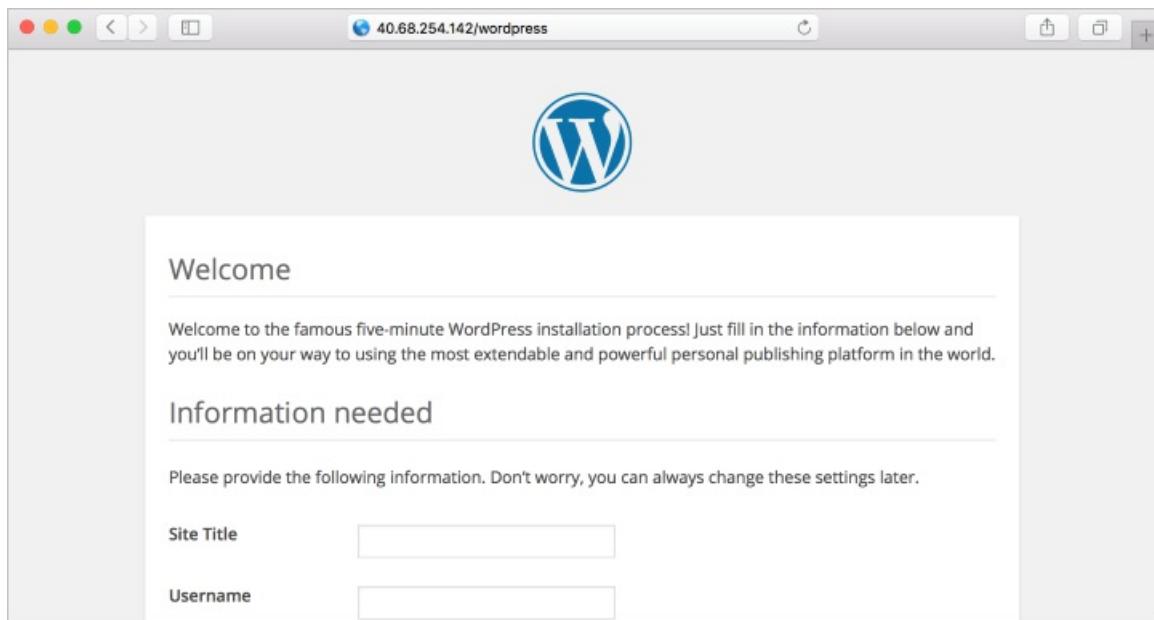
Move the WordPress installation to the web server document root:

```
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 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

4/26/2018 • 7 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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.

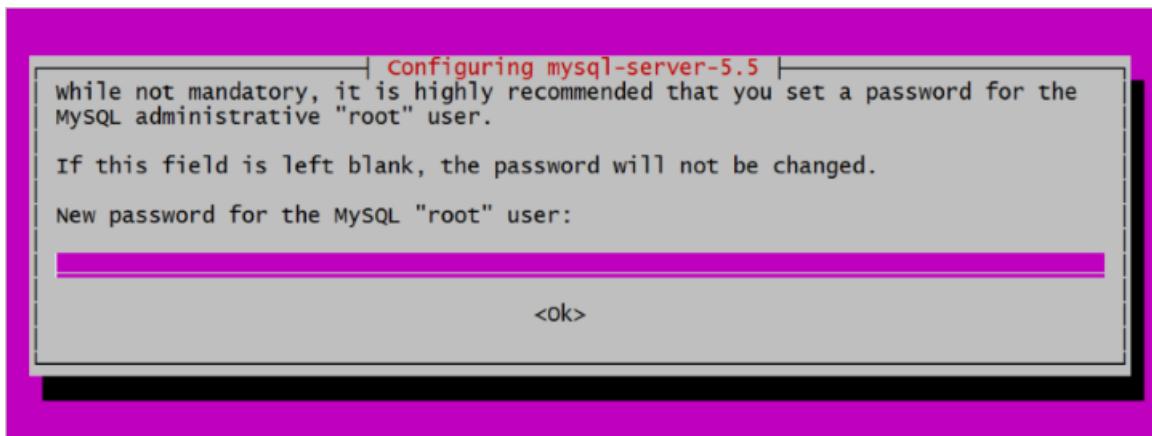
```
ssh azureuser@40.68.254.142
```

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 mysql-server php-mysql php php-fpm
```

You are prompted to install the packages and other dependencies. When prompted, set a root password for MySQL, and then [Enter] to continue. Follow the remaining prompts. This process installs the minimum required PHP extensions needed to use PHP with MySQL.



Verify installation and configuration

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:



MySQL

Check the version of MySQL with the following command (note the capital `v` parameter):

```
mysql -v
```

To help secure the installation of MySQL, run the `mysql_secure_installation` script. If you are only setting up a temporary server, you can skip this step.

```
mysql_secure_installation
```

Enter a root password for MySQL, and configure the security settings for your environment.

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.

```
mysql -u root -p
```

When done, exit the mysql prompt by typing `\q`.

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`, 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/php7.0-fpm.sock;  
    }  
}
```

Check the NGINX configuration for syntax errors:

```
sudo nginx -t
```

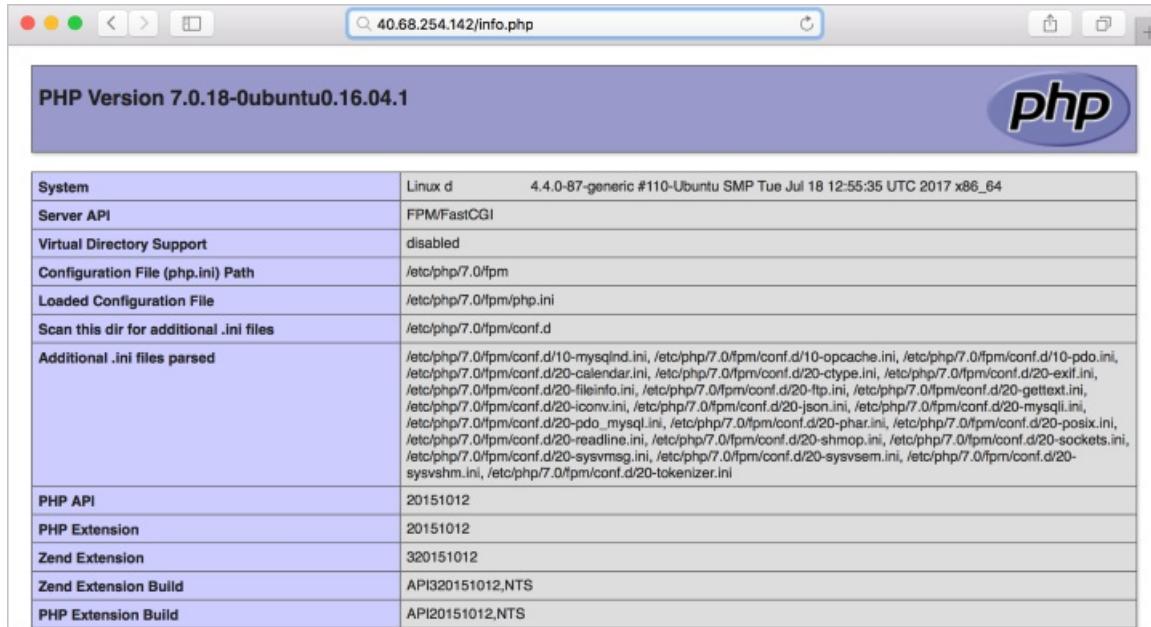
If the syntax is correct, restart NGINX with the following command:

```
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.



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
```

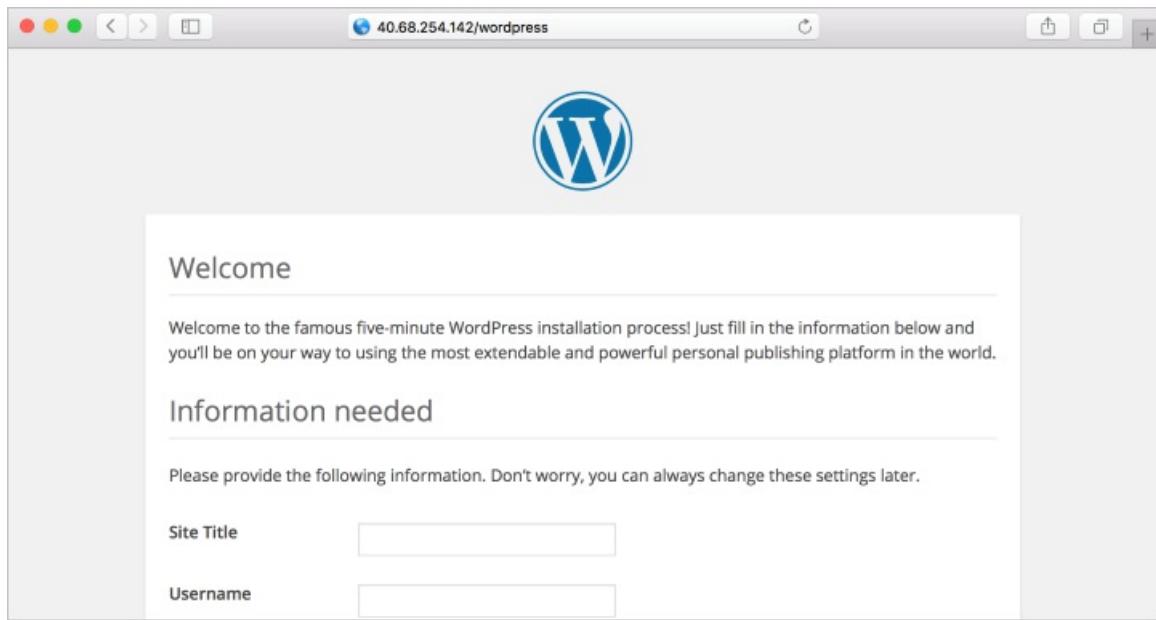
Copy the following lines to the file, substituting your WordPress database password for `yourPassword` (leave other values unchanged). Then save the file.

```
<?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

4/26/2018 • 6 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 ] http://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'));
  });
};

```

3. 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.

1. 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. It has four text input fields labeled **Name**, **Isbn**, **Author**, and **Pages**, each with a placeholder value. Below the form is a table header row with columns **Name**, **Isbn**, **Author**, and **Pages**. A single row of data is shown below the header, corresponding to the values entered in the form.

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 entering data into the form. The **Name** field now contains "MyBook", the **Isbn** field contains "000001", the **Author** field contains "Me", and the **Pages** field contains "200". The **Add** button is visible at the bottom of the form.

4. After refreshing the page, you should see something like this page:

The screenshot shows the browser after refreshing the page. The table now includes a new row for the book "MyBook" with ISBN "000001", Author "Me", and Pages "200". A **Delete** button is visible next to the row.

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

4/30/2018 • 5 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

Overview

Azure Key Vault safeguards cryptographic keys and secrets, such 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 lower case. Replace 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](#):

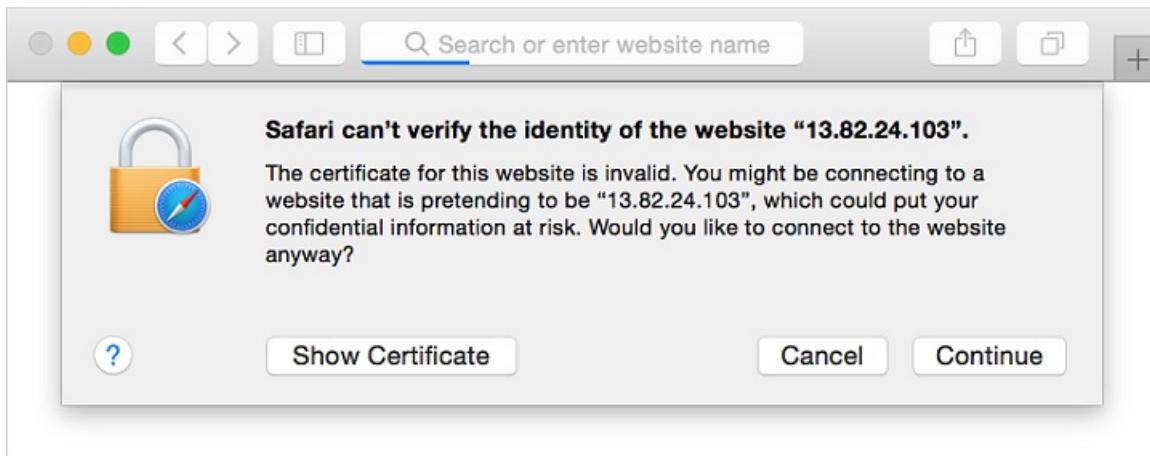
```

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://` 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

4/9/2018 • 1 min 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 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 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 a 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.
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 Operations Management Suite	Creates a virtual machine, installs the Operations Management Suite agent, and enrolls the VM in an OMS 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

4/9/2018 • 1 min 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.
Monitor virtual machines	
Monitor a VM with Operations Management Suite	Creates a virtual machine, installs the Operations Management Suite agent, and enrolls the VM in an OMS Workspace.

Azure Resource Manager overview

4/11/2018 • 15 min to read • [Edit Online](#)

The infrastructure for your application is typically made up of many components – maybe a virtual machine, storage account, and virtual network, or a web app, database, database server, and 3rd party services. You do not see these components as separate entities, instead you see them as related and interdependent parts of a single entity. You want to deploy, manage, and monitor them as a group. Azure Resource Manager enables you to work with the resources in your solution as a group. You can deploy, update, or delete all the resources for your solution in a single, coordinated operation. You use a template for deployment and that template can work for different environments such as testing, staging, and production. Resource Manager provides security, auditing, and tagging features to help you manage your resources after deployment.

Terminology

If you are new to Azure Resource Manager, there are some terms you might not be familiar with.

- **resource** - A manageable item that is available through Azure. Some common resources are a virtual machine, storage account, web app, database, and virtual network, but there are many more.
- **resource group** - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want 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 the resources you can deploy and manage through Resource Manager. Each resource provider offers operations for working with the resources that are deployed. Some common resource providers are Microsoft.Compute, which supplies the virtual machine resource, Microsoft.Storage, which supplies the storage account resource, and Microsoft.Web, which supplies resources related to web apps. See [Resource providers](#).
- **Resource Manager template** - A JavaScript Object Notation (JSON) file that defines one or more resources to deploy to a resource group. It also defines the dependencies between the deployed resources. 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:

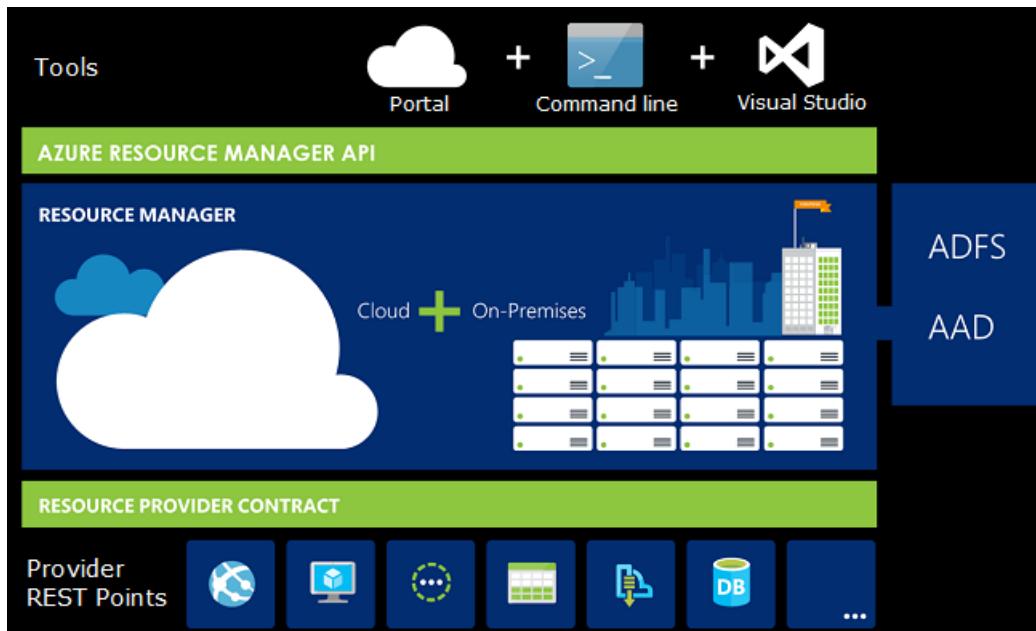
- You can deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.
- You can repeatedly deploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.
- You can manage your infrastructure through declarative templates rather than scripts.
- You can define the dependencies between resources so they are 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.

Resource Manager provides a new way to deploy and manage your solutions. If you used the earlier deployment model and want to learn about the changes, see [Understanding Resource Manager deployment and classic deployment](#).

Consistent management layer

Resource Manager provides a consistent management layer for the tasks you perform through Azure PowerShell, Azure CLI, Azure portal, REST API, and development tools. All the tools use a common set of operations. You use the tools that work best for you, and can use them interchangeably without confusion.

The following image shows how all the tools interact with the same 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 resource providers.



Guidance

The following suggestions help you take full advantage of Resource Manager when working with your solutions.

1. Define and deploy your infrastructure through the declarative syntax in Resource Manager templates, rather than through imperative commands.
2. Define all deployment and configuration steps in the template. You should have no manual steps for setting up your solution.
3. Run imperative commands to manage your resources, such as to start or stop an app or machine.
4. 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](#).

Resource groups

There are some important factors to consider when defining your resource group:

1. 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.
2. Each resource can only exist in one resource group.
3. You can add or remove a resource to a resource group at any time.

4. You can move a resource from one resource group to another group. For more information, see [Move resources to new resource group or subscription](#).
5. A resource group can contain resources that reside in different regions.
6. A resource group can be used to scope access control for administrative actions.
7. A resource can interact with resources in other resource groups. This interaction is common when the two resources are related but do not 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 are specifying where that metadata is stored. For compliance reasons, you may need to ensure that your data is stored in a particular region.

Resource providers

Each resource provider offers a set of resources and operations for working with an Azure service. 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}**. For example, the key vault type 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](#).

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. When you create a solution from the portal, the solution automatically includes a deployment template. You do not 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. You can 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.

To learn about the format of the template and how you construct it, see [Create your first Azure Resource Manager template](#). 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 (see the image for [Consistent management layer](#)). 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": {}
  }
]

```

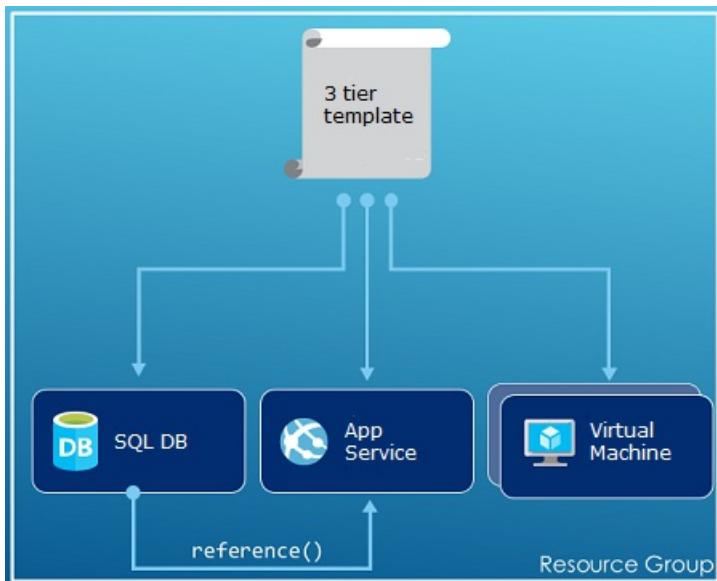
It converts the definition to the following REST API operation, which is sent to the Microsoft.Storage resource provider:

```

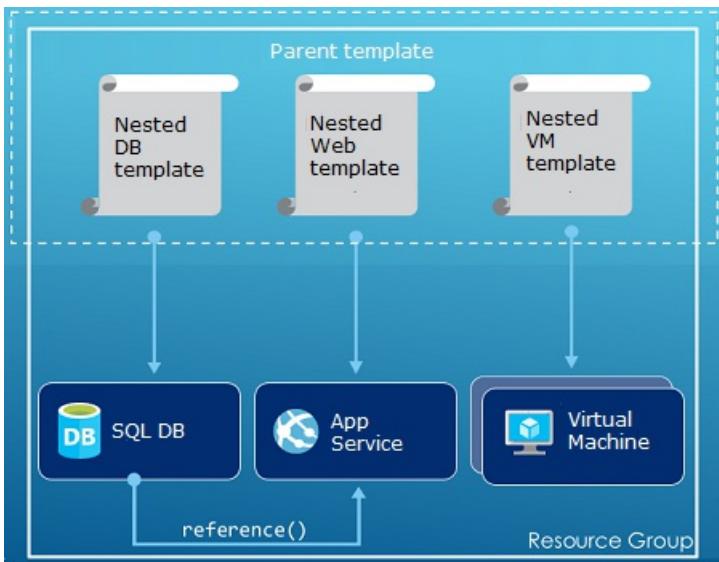
PUT
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/mystorageaccount?api-version=2016-01-01
REQUEST BODY
{
  "location": "westus",
  "properties": {},
  "sku": {
    "name": "Standard_LRS"
  },
  "kind": "Storage"
}

```

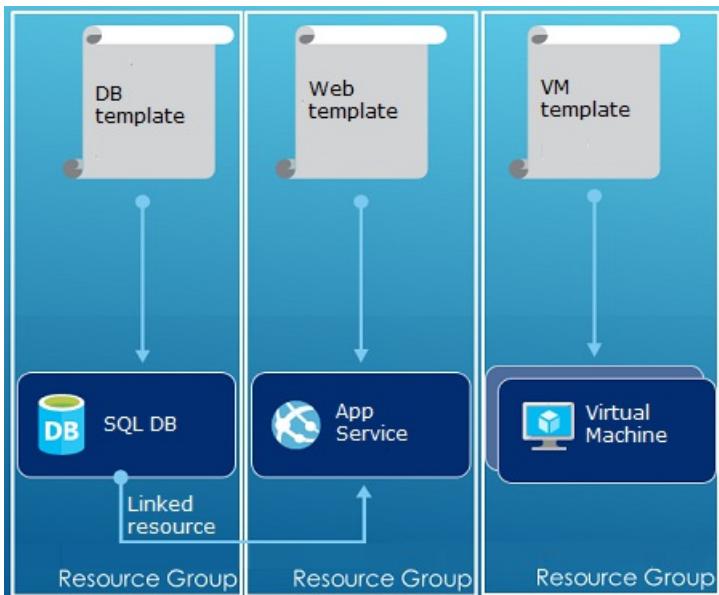
How you define templates and resource groups is entirely up to you and how you want to manage your solution. For example, you can deploy your three tier application through a single template to a single resource group.



But, you do not 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 specifies creating a resource but that resource already exists, Azure Resource Manager performs an update instead of creating a new asset. Azure Resource Manager updates the existing asset to the same state as it would be as new.

Resource Manager provides extensions for scenarios when you need additional operations such as installing particular software that is not included in the setup. If you are 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](#).

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 are ready to deploy the resources to Azure. For the commands 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

Tags

Resource Manager provides a tagging feature that enables you to categorize resources according to your requirements for managing or billing. Use tags when you have a complex collection of resource groups and resources, and need to visualize those assets in the way that makes the most sense to you. For example, you could tag resources that serve a similar role in your organization or belong to the same department. Without tags, users in your organization can create multiple resources that may be difficult to later identify and manage. For example, you may wish to delete all the resources for a particular project. If those resources are not tagged for the project, you have to manually find them. Tagging can be an important way for you to reduce unnecessary costs in your subscription.

Resources do not need to reside in the same resource group to share a tag. You can create your own tag taxonomy to ensure that all users in your organization use common tags rather than users inadvertently applying slightly different tags (such as "dept" instead of "department").

The following example shows a tag applied to a virtual machine.

```
"resources": [
  {
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2015-06-15",
    "name": "SimpleWindowsVM",
    "location": "[resourceGroup().location]",
    "tags": {
      "costCenter": "Finance"
    },
    ...
  }
]
```

To retrieve all the resources with a tag value, use the following PowerShell cmdlet:

```
Find-AzureRmResource -TagName costCenter -TagValue Finance
```

Or, the following Azure CLI 2.0 command:

```
az resource list --tag costCenter=Finance
```

You can also view tagged resources through the Azure portal.

The [usage report](#) for your subscription includes tag names and values, which enables you to break out costs by tags. For more information about tags, see [Using tags to organize your Azure resources](#).

Access control

Resource Manager enables you to control who has access to specific actions for your organization. It natively integrates role-based access control (RBAC) into the management platform and applies that access control to all services in your resource group.

There are two main concepts to understand when working with role-based access control:

- Role definitions - describe a set of permissions and can be used in many assignments.
- Role assignments - associate a definition with an identity (user or group) for a particular scope (subscription, resource group, or resource). The assignment is inherited by lower scopes.

You can add users to pre-defined platform and resource-specific roles. For example, you can take advantage of the pre-defined role called Reader that permits users to view resources but not change them. You add users in your organization that need this type of access to the Reader role and apply the role to the subscription, resource group, or resource.

Azure provides the following four platform roles:

1. Owner - can manage everything, including access
2. Contributor - can manage everything except access
3. Reader - can view everything, but can't make changes
4. User Access Administrator - can manage user access to Azure resources

Azure also provides several resource-specific roles. Some common ones are:

1. Virtual Machine Contributor - can manage virtual machines but not grant access to them, and cannot manage the virtual network or storage account to which they are connected
2. Network Contributor - can manage all network resources, but not grant access to them
3. Storage Account Contributor - Can manage storage accounts, but not grant access to them
4. SQL Server Contributor - Can manage SQL servers and databases, but not their security-related policies
5. Website Contributor - Can manage websites, but not the web plans to which they are connected

For the full list of roles and permitted actions, see [RBAC: Built in Roles](#). For more information about role-based access control, see [Azure Role-based Access Control](#).

In some cases, you want to run code or script that accesses resources, but you do not want to run it under a user's credentials. Instead, you want to create an identity called a service principal for the application and assign the appropriate role for the service principal. Resource Manager enables you to create credentials for the application and programmatically authenticate the application. To learn about creating service principals, see one of following topics:

- [Use Azure PowerShell to create a service principal to access resources](#)
- [Use Azure CLI to create a service principal to access resources](#)
- [Use portal to create Azure Active Directory application and service principal that can access resources](#)

You can also explicitly lock critical resources to prevent users from deleting or modifying them. For more information, see [Lock resources with Azure Resource Manager](#).

Activity logs

Resource Manager logs all operations that create, modify, or delete a resource. You can use the activity logs to find an error when troubleshooting or to monitor how a user in your organization modified a resource. To see the logs, select **Activity logs** in the **Settings** blade for a resource group. You can filter the logs by many different values including which user initiated the operation. For information about working with the activity logs, see [View activity logs to manage Azure resources](#).

Customized policies

Resource Manager enables you to create customized policies for managing your resources. The types of policies you create can include diverse scenarios. You can enforce a naming convention on resources, limit which types and instances of resources can be deployed, or limit which regions can host a type of resource. You can require a tag value on resources to organize billing by departments. You create policies to help reduce costs and maintain

consistency in your subscription.

You define policies with JSON and then apply those policies either across your subscription or within a resource group. Policies are different than role-based access control because they are applied to resource types.

The following example shows a policy that ensures tag consistency by specifying that all resources include a costCenter tag.

```
{  
  "if": {  
    "not": {  
      "field" : "tags",  
      "containsKey" : "costCenter"  
    }  
  },  
  "then": {  
    "effect" : "deny"  
  }  
}
```

There are many more types of policies you can create. For more information, see [What is Azure Policy?](#).

SDKs

Azure SDKs are available for multiple languages and platforms. Each of these language implementations is available through its ecosystem package manager and GitHub.

Here are our Open Source SDK repositories. We welcome feedback, issues, and pull requests.

- [Azure SDK for .NET](#)
- [Azure Management Libraries for Java](#)
- [Azure SDK for Node.js](#)
- [Azure SDK for PHP](#)
- [Azure SDK for Python](#)
- [Azure SDK for Ruby](#)

For information about using these languages with your resources, see:

- [Azure for .NET developers](#)
- [Azure for Java developers](#)
- [Azure for Node.js developers](#)
- [Azure for Python developers](#)

NOTE

If the SDK doesn't provide the required functionality, you can also call to the [Azure REST API](#) directly.

Next steps

- For a simple introduction to working with templates, see [Export an Azure Resource Manager template from existing resources](#).
- For a more thorough walkthrough of creating a template, see [Create your first Azure Resource Manager template](#).
- To understand the functions you can use in a template, see [Template functions](#)
- For information about using Visual Studio with Resource Manager, see [Creating and deploying Azure resource](#)

groups through Visual Studio.

Here's a video demonstration of this overview:

Regions and availability for virtual machines in Azure

4/9/2018 • 6 min 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 Storage](#), 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](#).

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.

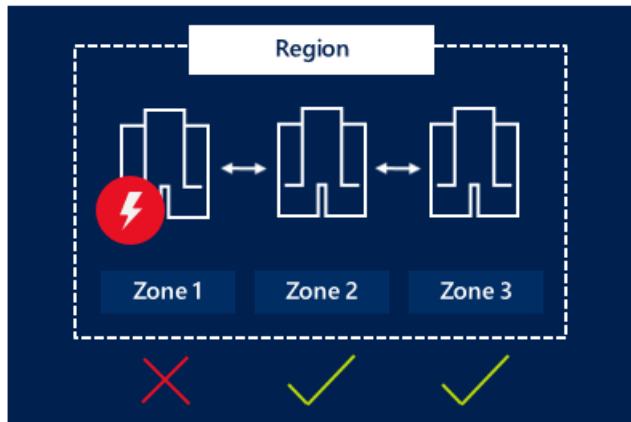
Managed Disk fault domains

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, 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

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

5/3/2018 • 1 min 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, DS, D, Av2, A0-7	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2, Fs, F	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, DS, Dv2, D	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Ls	High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases.
GPU	NV, NC, NCv2, NCv3, ND	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, A8-11	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](#)

General purpose virtual machine sizes

4/11/2018 • 10 min 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 and network bandwidth for each size in this grouping.

- The A-series and Av2-series VMs can be deployed on a variety of hardware types and processors. 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.
- D-series VMs are designed to run applications that demand higher compute power and temporary disk performance. D-series VMs provide faster processors, a higher memory-to-vCPU ratio, and a solid-state drive (SSD) for the temporary disk. For details, see the announcement on the Azure blog, [New D-Series Virtual Machine Sizes](#).
- 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 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 same processor(s) as the Dv2-series, but in a hyper-threaded configuration, providing a better value proposition for most general purpose workloads, and bringing the Dv3 into alignment with the general purpose VMs of most other clouds. 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.
- 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.

B-series

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 conscience 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.

SIZE	VCPUs	MEMORY: GiB	LOCAL SSD: GiB	BASE PERF OF A CORE	CREDITS / HOUR	MAX BANKED CREDITS	MAX DATA DISKS	MAX LOCAL DISK PERF: IOPS / MBPS	MAX UNCAUSED HED DISK PERF: IOPS / MBPS	MAX NICs
Standard_B1s	1	1	4	10%	6	144	2	400 / 10	320 / 10	2

SIZE	VCPU	MEMORY: GIB	LOCAL SSD: GIB	BASE PERF OF A CORE	CREDIT S BANKED / HOUR	MAX BANKED CREDIT S	MAX DATA DISKS	MAX LOCAL DISK PERF: IOPS / MBPS	MAX UNCACHED DISK PERF: IOPS / MBPS	MAX NICS
Standard_B1ms	1	2	4	20%	12	288	2	800 / 10	640 / 10	2
Standard_B2s	2	4	8	40%	24	576	4	1600 / 15	1280 / 15	3
Standard_B2ms	2	8	16	60%	36	864	4	2400 / 22.5	1920 / 22.5	3
Standard_B4ms	4	16	32	90%	54	1296	8	3600 / 35	2880 / 35	4
Standard_B8ms	8	32	64	135%	81	1944	16	4320 / 50	4320 / 50	4

Dsv3-series ¹

ACU: 160-190

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_D2s_v3	2	8	16	4	4,000 / 32 (50)	3,200 / 48	2 / 1,000
Standard_D4s_v3	4	16	32	8	8,000 / 64 (100)	6,400 / 96	2 / 2,000
Standard_D8s_v3	8	32	64	16	16,000 / 128 (200)	12,800 / 192	4 / 4,000
Standard_D16s_v3	16	64	128	32	32,000 / 256 (400)	25,600 / 384	8 / 8,000
Standard_D32s_v3	32	128	256	32	64,000 / 512 (800)	51,200 / 768	8 / 16,000

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_D64s_v3	64	256	512	32	128,000 / 1024 (1600)	80,000 / 1200	8 / 30,000

¹ Dsv3-series VM's feature Intel® Hyper-Threading Technology

Dv3-series ¹

ACU: 160-190

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 / 1,000
Standard_D4_v3	4	16	100	8	6000/93/46	2 / 2,000
Standard_D8_v3	8	32	200	16	12000/187/93	4 / 4,000
Standard_D16_v3	16	64	400	32	24000/375/187	8 / 8,000
Standard_D32_v3	32	128	800	32	48000/750/375	8 / 16,000
Standard_D64_v3	64	256	1600	32	96000/1000/500	8 / 30,000

¹ Dv3-series VM's feature Intel® Hyper-Threading Technology

DSv2-series

ACU: 210-250

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_D1_v2	1	3.5	7	4	4,000 / 32 (43)	3,200 / 48	2 / 750
Standard_D2_v2	2	7	14	8	8,000 / 64 (86)	6,400 / 96	2 / 1500
Standard_D3_v2	4	14	28	16	16,000 / 128 (172)	12,800 / 192	4 / 3000
Standard_D4_v2	8	28	56	32	32,000 / 256 (344)	25,600 / 384	8 / 6000
Standard_D5_v2	16	56	112	64	64,000 / 512 (688)	51,200 / 768	8 / 12000

Dv2-series

ACU: 210-250

SIZE	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX TEMP STORAGE THROUGHPUT: IOPS / READ MBPS / WRITE MBPS	MAX DATA DISKS	THROUGHPUT: UT: 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

DS-series

ACU: 160

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

D-series

ACU: 160

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

Av2-series

ACU: 100

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
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

A-series

ACU: 50-100

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.

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

Basic A

SIZE - SIZE\NAME	VCPU	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

Note that the number of Data Disks for Classic VMs might be lower than the number of Data Disks for Azure Resource Manager VMs.

Size table definitions

- Storage capacity is shown in units of GiB or 1024³ bytes. When comparing disks measured in GB (1000³ bytes) to disks measured in GiB (1024³) 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 = 10⁶ 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](#)

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

4/9/2018 • 4 min 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, small databases and development and test 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 six VM sizes:

SIZE	VCPU'S	MEMORY: GIB	TEMP STORAGE (SSD) GIB	BASE CPU PERF OF VM	MAX CPU PERF OF VM	CREDITS BANKED / HOUR	MAX BANKED CREDITS
Standard_B_1s	1	1	4	10%	100%	6	144
Standard_B_1ms	1	2	4	20%	100%	12	288
Standard_B_2s	2	4	8	40%	200%	24	576
Standard_B_2ms	2	8	16	60%	200%	36	864
Standard_B_4ms	4	16	32	90%	400%	54	1296
Standard_B_8ms	8	32	64	135%	800%	81	1944

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 .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 $.1 \times 60 = 6$ credits per hour. For the 20 hours that I am off-peak, I will bank 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 .4 credits a minute or $.4 \times 60 = 24$ 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: Does the B-Series support Premium Storage data disks?

A: Yes, all B-Series sizes support Premium Storage data disks.

Q: My remaining credit are set to 0 after a redeploy or a stop/start.

A : When a VM is "REDPLOYED", i.e., the VM moves to another node, and 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.

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

4/9/2018 • 4 min 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 base core frequency of 2.7 GHz 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.

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.

The Fs-series provides all the advantages of the F-series, in addition to Premium storage.

Fsv2-series ¹

ACU: 195 - 210

SIZE	VCPUS	MEMORY: GIB	TEMP STORAGE (SSD) GIB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GIB)	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_F2s_v2	2	4	16	4	4000 (32)	2 / 875
Standard_F4s_v2	4	8	32	8	8000 (64)	2 / 1,750
Standard_F8s_v2	8	16	64	16	16000 (128)	4 / 3,500
Standard_F16s_v2	16	32	128	32	32000 (256)	4 / 7,000
Standard_F32s_v2	32	64	256	32	64000 (512)	8 / 14,000

SIZE	VCPUs	MEMORY: GiB	TEMP STORAGE (SSD) GiB	MAX DATA DISKS	MAX CACHED AND TEMP STORAGE THROUGHPUT: IOPS / MBPS (CACHE SIZE IN GiB)	MAX NICs / EXPECTED NETWORK BANDWIDTH (MBPS)
Standard_F64_s_v2	64	128	512	32	128000 (1024)	8 / 28,000
Standard_F72_s_v2 ^{2, 3}	72	144	576	32	144000 (1520)	8 / 30,000

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.

Fs-series ¹

ACU: 210 - 250

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_F1s	1	2	4	4	4,000 / 32 (12)	3,200 / 48	2 / 750
Standard_F2s	2	4	8	8	8,000 / 64 (24)	6,400 / 96	2 / 1500
Standard_F4s	4	8	16	16	16,000 / 128 (48)	12,800 / 192	4 / 3000
Standard_F8s	8	16	32	32	32,000 / 256 (96)	25,600 / 384	8 / 6000
Standard_F16s	16	32	64	64	64,000 / 512 (192)	51,200 / 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 [Premium Storage: High-performance storage for Azure virtual machine workloads](#).

F-series

ACU: 210 - 250

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_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

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](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure

SKUs.

Memory optimized virtual machine sizes

5/10/2018 • 10 min 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 M-Series offers the highest vCPU count (up to 128 vCPUs) and largest memory (up to 3.8 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.
- Dv2-series, D-series, G-series, and the DS/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.
- D-series VMs are designed to run applications that demand higher compute power and temporary disk performance. D-series VMs provide faster processors, a higher memory-to-vCPU ratio, and a solid-state drive (SSD) for temporary storage. For details, see the announcement on the Azure blog, [New D-Series Virtual Machine Sizes](#).
- 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

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_E2s_v3	2	16	32	4	4,000 / 32 (50)	3,200 / 48	2 / 1,000
Standard_E4s_v3 ²	4	32	64	8	8,000 / 64 (100)	6,400 / 96	2 / 2,000
Standard_E8s_v3 ²	8	64	128	16	16,000 / 128 (200)	12,800 / 192	4 / 4,000
Standard_E16s_v3 ²	16	128	256	32	32,000 / 256 (400)	25,600 / 384	8 / 8,000
Standard_E32s_v3 ²	32	256	512	32	64,000 / 512 (800)	51,200 / 768	8 / 16,000
Standard_E64s_v3 ²	64	432	864	32	128,000/1024 (1600)	80,000 / 1200	8 / 30,000
Standard_E64is_v3 ³	64	432	864	32	128,000/1024 (1600)	80,000 / 1200	8 / 30,000

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

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 / 1,000
Standard_E4_v3	4	32	100	8	6000/93/46	2 / 2,000

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_E8_v3	8	64	200	16	12000/187/93	4 / 4,000
Standard_E16_v3	16	128	400	32	24000/375/187	8 / 8,000
Standard_E32_v3	32	256	800	32	48000/750/375	8 / 16,000
Standard_E64_v3	64	432	1600	32	96000/1000/500	8 / 30,000
Standard_E64i_v3 ²	64	432	1600	32	96000/1000/500	8 / 30,000

¹ Ev3-series VM's feature Intel® Hyper-Threading Technology.

² Constrained core sizes available.

M-series

ACU: 160-180¹

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_M_64s	64	1024	2048	64	80,000 / 800 (6348)	40,000 / 1,000	8 / 16000
Standard_M_64ms	64	1792	2048	64	80,000 / 800 (6348)	40,000 / 1,000	8 / 16000
Standard_M_128s ^{2, 3}	128	2048	4096	64	160,000 / 1,600 (12,696)	80,000 / 2,000	8 / 30000
Standard_M_128ms ^{2, 3, 4}	128	3800	4096	64	160,000 / 1,600 (12,696)	80,000 / 2,000	8 / 30000

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.

GS-series

ACU: 180 - 240 ¹

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_GS1	2	28	56	8	10,000 / 100 (264)	5,000 / 125	2 / 2000
Standard_GS2	4	56	112	16	20,000 / 200 (528)	10,000 / 250	2 / 4000
Standard_GS3	8	112	224	32	40,000 / 400 (1,056)	20,000 / 500	4 / 8000
Standard_GS4 ³	16	224	448	64	80,000 / 800 (2,112)	40,000 / 1,000	8 / 16000
Standard_GS5 ^{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 [Premium Storage: High-performance storage for Azure virtual machine workloads](#).

² Instance is isolated to hardware dedicated to a single customer.

³ Constrained core sizes available.

G-series

ACU: 180 - 240

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

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_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.

DSv2-series

ACU: 210 - 250¹

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	8,000 / 64 (72)	6,400 / 96	2 / 1500
Standard_D S12_v2 ³	4	28	56	16	16,000 / 128 (144)	12,800 / 192	4 / 3000
Standard_D S13_v2 ³	8	56	112	32	32,000 / 256 (288)	25,600 / 384	8 / 6000
Standard_D S14_v2 ³	16	112	224	64	64,000 / 512 (576)	51,200 / 768	8 / 12000
Standard_D S15_v2 ²	20	140	280	64	80,000 / 640 (720)	64,000 / 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 [Premium Storage: High-performance storage for Azure virtual machine workloads](#).

² Instance is isolated to hardware dedicated to a single customer.

³ Constrained core sizes available.

⁴ 25000 Mbps with Accelerated Networking.

Dv2-series

ACU: 210 - 250

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	1,000	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.

DS-series

ACU: 160¹

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_DS11	2	14	28	8	8,000 / 64 (72)	6,400 / 64	2 / 1000
Standard_DS12	4	28	56	16	16,000 / 128 (144)	12,800 / 128	4 / 2000
Standard_DS13	8	56	112	32	32,000 / 256 (288)	25,600 / 256	8 / 4000
Standard_DS14	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 [Premium Storage: High-performance storage for Azure virtual machine workloads](#).

D-series

ACU: 160

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

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](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure

SKUs.

Constrained vCPU capable VM sizes

3/9/2018 • 2 min 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 that are only available in Azure, allowing workloads to push higher CPU utilization at a fraction of the (per-core) 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	VCPUs	SPECS
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
Standard_E16-8s_v3	8	Same as E16s_v3
Standard_E16-4s_v3	4	Same as E16s_v3
Standard_E32-16_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

NAME	VCPUs	SPECS
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

1/12/2018 • 2 min to read • [Edit Online](#)

Storage optimized VM sizes offer high disk throughput and IO, and are ideal for Big Data, SQL, and NoSQL databases. 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 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.

Ls-series

ACU: 180-240

SIZE	VCPUs	MEMORY: GiB	TEMP STORAGE (SSD) 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	20,000 / 200	5,000 / 125	2 / 4,000
Standard_L 8s	8	64	1,388	32	40,000 / 400	10,000 / 250	4 / 8,000
Standard_L 16s	16	128	2,807	64	80,000 / 800	20,000 / 500	8 / 16,000
Standard_L 32s ¹	32	256	5,630	64	160,000 / 1,600	40,000 / 1,000	8 / 20,000

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 [Premium Storage: High-performance storage for Azure virtual machine workloads](#).

¹ 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.
- Ls-series data disks cannot operate in cached mode, the host cache mode must be 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](#)
- [Memory optimized](#)
- [GPU](#)
- [High performance compute](#)

Next steps

Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

GPU optimized virtual machine sizes

4/9/2018 • 7 min 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 as well as storage throughput and network bandwidth for each size in this grouping.

- **NC, NCv2, NCv3, and ND** sizes are optimized for compute-intensive and network-intensive applications and algorithms, including CUDA- and OpenCL-based applications and simulations, AI, and Deep Learning.
- **NV** sizes are optimized and designed for remote visualization, streaming, gaming, encoding, and VDI scenarios utilizing frameworks such as OpenGL and DirectX.

NC-series

NC-series VMs are powered by the [NVIDIA Tesla K80](#) card. 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	MAX DATA DISKS	MAX NICs
Standard_NC6	6	56	340	1	24	1
Standard_NC12	12	112	680	2	48	2
Standard_NC24	24	224	1440	4	64	4
Standard_NC24r*	24	224	1440	4	64	4

1 GPU = one-half K80 card.

*RDMA capable

NCv2-series

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. 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	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	GPU	MAX DATA DISKS	MAX NICs
Standard_NC6_s_v2	6	112	736	1	12	4
Standard_NC1_2s_v2	12	224	1474	2	24	8
Standard_NC2_4s_v2	24	448	2948	4	32	8
Standard_NC2_4rs_v2*	24	448	2948	4	32	8

1 GPU = one P100 card.

*RDMA capable

NCv3-series

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.

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	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	GPU	MAX DATA DISKS	MAX NICs
Standard_NC6_s_v3	6	112	736	1	12	4
Standard_NC1_2s_v3	12	224	1474	2	24	8
Standard_NC2_4s_v3	24	448	2948	4	32	8
Standard_NC2_4rs_v3*	24	448	2948	4	32	8

1 GPU = one V100 card.

*RDMA capable

ND-series

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. 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	VCPU	MEMORY: GIB	TEMP STORAGE (SSD) GIB	GPU	MAX DATA DISKS	MAX NICs
Standard_ND 6s	6	112	736	1	12	4
Standard_ND 12s	12	224	1474	2	24	8
Standard_ND 24s	24	448	2948	4	32	8
Standard_ND 24rs*	24	448	2948	4	32	8

1 GPU = one P40 card.

*RDMA capable

NV-series

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.

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	MAX DATA DISKS	MAX NICs	VIRTUAL WORKSTATIONS	VIRTUAL APPLICATIONS
Standard_NV6	6	56	340	1	24	1	1	25
Standard_NV12	12	112	680	2	48	2	2	50
Standard_NV24	24	224	1440	4	64	4	4	100

1 GPU = one-half M60 card.

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

NC, NCv2, NCv3, and ND-series - NVIDIA CUDA drivers

CUDA driver information in the following table 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	NVIDIA CUDA 9.1, driver branch R390
Red Hat Enterprise Linux 7.3 or 7.4	
CentOS-based 7.3 or 7.4, CentOS-based 7.4 HPC	

NV-series - NVIDIA GRID drivers

Microsoft redistributes NVIDIA GRID driver installers for NV VMs. Install only these GRID drivers on Azure NV VMs. These drivers include licensing for GRID Virtual GPU Software in Azure.

DISTRIBUTION	DRIVER
Ubuntu 16.04 LTS	NVIDIA GRID 6.0, driver branch R390
Red Hat Enterprise Linux 7.3 or 7.4	
CentOS-based 7.3 or 7.4	

WARNING

Installation of third-party software on Red Hat products can affect the Red Hat support terms. See the [Red Hat Knowledgebase article](#).

For driver installation and verification steps, see [N-series driver setup for Linux](#).

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, ND, and NCv3 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, or NV 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](#)

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

4/30/2018 • 7 min to read • [Edit Online](#)

To take advantage of the GPU capabilities of Azure N-series VMs running Linux, NVIDIA graphics drivers must be installed. This article provides driver setup steps after you deploy an N-series VM. 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

NC, NCv2, NCv3, and ND-series - NVIDIA CUDA drivers

CUDA driver information in the following table 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	NVIDIA CUDA 9.1, driver branch R390
Red Hat Enterprise Linux 7.3 or 7.4	
CentOS-based 7.3 or 7.4, CentOS-based 7.4 HPC	

NV-series - NVIDIA GRID drivers

Microsoft redistributes NVIDIA GRID driver installers for NV VMs. Install only these GRID drivers on Azure NV VMs. These drivers include licensing for GRID Virtual GPU Software in Azure.

DISTRIBUTION	DRIVER
Ubuntu 16.04 LTS	NVIDIA GRID 6.0, driver branch R390
Red Hat Enterprise Linux 7.3 or 7.4	
CentOS-based 7.3 or 7.4	

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 for NC, NCv2, NCv3, and ND-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 16.04 LTS

1. Download and install the CUDA drivers.

```
CUDA_REPO_PKG=cuda-repo-ubuntu1604_9.1.85-1_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  
http://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.

2. 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 7.3 or 7.4

1. Update the 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-rhel7-9.1.85-1.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 Fan	Name Temp	Persistence-M Perf	Pwr:Usage/Cap	Bus-Id	Disp.A	Volatile Memory-Usage	Uncorr. ECC GPU-Util Compute M.
0 N/A	Tesla K80 51C	off P0	58W / 149W	000007D1:00:00.0	off	0MiB / 11439MiB	0% Default
Processes:							
GPU PID Type Process name				GPU Memory Usage			
No running processes found							

RDMA network connectivity

RDMA network connectivity can be enabled on RDMA-capable N-series VMs such as NC24r deployed in the same availability set or 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 for NV-series VMs

To install NVIDIA GRID drivers on NV-series VMs, make an SSH connection to each VM and follow the steps for your Linux distribution.

Ubuntu 16.04 LTS

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 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=849941  
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=TRUE
```

9. Reboot the VM and proceed to verify the installation.

CentOS or Red Hat Enterprise Linux

1. Update the kernel and DKMS.

```
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 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=849941  
  
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=TRUE
```

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    |          0%       Default |
| N/A   31C     P0    39W / 150W |        0MiB /  8123MiB |          |
+-----+
+-----+
| 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 VM, [x11vnc](#) is recommended because it allows hardware acceleration of graphics. The BusID of the M60 device must be manually added to the xconfig file ([etc/X11/xorg.conf](#) on Ubuntu 16.04 LTS, [/etc/X11/XF86config](#) on CentOS 7.3 or Red Hat Enterprise Server 7.3). Add a "Device" section similar to the following:

```
Section "Device"
    Identifier      "Device0"
    Driver         "nvidia"
    VendorName    "NVIDIA Corporation"
    BoardName     "Tesla M60"
    BusID         "your-BusID:0:0:0"
EndSection
```

Additionally, update your "Screen" section to use this device.

The decimal BusID can be found by running

```
echo $((16#`/usr/bin/nvidia-smi --query-gpu=pci.bus_id --format=csv | tail -1 | cut -d ':' -f 1`))
```

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 the following contents:

```
#!/bin/bash
BUSID=$((`/usr/bin/nvidia-smi --query-gpu=pci.bus_id --format=csv | tail -1 | cut -d ':' -f 1`))

if grep -Fxq "${BUSID}" /etc/X11/XF86Config; then      echo "BUSID is matching"; else    echo "BUSID changed to ${BUSID}" && sed -i '/BusID/c\      BusID          \"PCI:0@'${BUSID}'\:0:0:0\\\"' /etc/X11/XF86Config; 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

4/9/2018 • 7 min to read • [Edit Online](#)

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. 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.

Azure H-series virtual machines are the latest in high performance computing VMs aimed at high end computational needs, like molecular modeling, and computational fluid dynamics. These 8 and 16 vCPU VMs are built on the Intel Haswell E5-2667 V3 processor technology featuring DDR4 memory and SSD-based temporary storage.

In addition to the substantial CPU power, the H-series offers diverse options for low latency RDMA networking using FDR InfiniBand and several memory configurations to support memory intensive computational requirements.

H-series

ACU: 290-300

SIZE	VCPUs	MEMORY: GiB	TEMP STORAGE (SSD) GiB	MAX DATA DISKS	MAX DISK THROUGHPUT: IOPS	MAX NICs
Standard_H8	8	56	1000	32	32 x 500	2
Standard_H16	16	112	2000	64	64 x 500	4
Standard_H8m	8	112	1000	32	32 x 500	2
Standard_H16m	16	224	2000	64	64 x 500	4
Standard_H16r ¹	16	112	2000	64	64 x 500	4
Standard_H16mr ¹	16	224	2000	64	64 x 500	4

¹ For MPI applications, dedicated RDMA backend network is enabled by FDR InfiniBand network, which delivers ultra-low-latency and high bandwidth.

A-series - compute-intensive instances

ACU: 225

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.

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 (H16r, H16mr, A8, and A9) feature a network interface for remote direct memory access (RDMA) connectivity. (Selected N-series sizes designated with 'r' such as NC24r 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 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.

NOTE

In Azure, IP over IB is not supported. Only RDMA over IB is supported.

Deploy the RDMA-capable HPC VMs in the same availability set or VM scale set (when you use the Azure Resource Manager deployment model) or the same cloud service (when you use the classic deployment model). Additional requirements for RDMA-capable HPC VMs to access the Azure RDMA network follow.

MPI

Only Intel MPI 5.x versions are supported. Later versions (2017, 2018) of the Intel MPI runtime library are not compatible with the Azure Linux RDMA drivers.

Distributions

Deploy a compute-intensive VM from one of the images in the Azure Marketplace that supports RDMA connectivity:

- **Ubuntu** - Ubuntu Server 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 libdapl12 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`

- **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). 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
```

- **CentOS-based HPC** - CentOS-based 6.5 HPC or a later version (for H-series, version 7.1 or later is recommended). RDMA drivers and Intel MPI 5.1 are installed on the VM.

NOTE

On the CentOS-based HPC images, kernel updates are disabled in the **yum** configuration file. This is because the Linux RDMA drivers are distributed as an RPM package, and driver updates might not work if the kernel is updated.

Cluster configuration

Additional system configuration is needed to run MPI jobs on clustered VMs. For example, on a cluster of VMs, you need to establish trust among the compute nodes. For typical settings, see [Set up a Linux RDMA cluster to run MPI applications](#).

Network topology considerations

- On 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. Eth0 is reserved for regular Azure network traffic.
- The RDMA network in Azure reserves the address space 172.16.0.0/16.

Using HPC Pack

[HPC Pack](#), Microsoft's free HPC cluster and job management solution, is one option for you to use the compute-intensive instances with Linux. The latest releases of HPC Pack support several Linux distributions to run on compute nodes deployed in Azure VMs, managed by a Windows Server head node. With RDMA-capable Linux compute nodes running Intel MPI, HPC Pack can schedule and run Linux MPI applications that access the RDMA network. See [Get started with Linux compute nodes in an HPC Pack cluster in Azure](#).

Other sizes

- [General purpose](#)
- [Compute optimized](#)
- [Memory optimized](#)
- [Storage optimized](#)
- [GPU](#)

Next steps

- To get started deploying and using compute-intensive sizes with RDMA on Linux, see [Set up a Linux RDMA cluster to run MPI applications](#).
- Learn more about how [Azure compute units \(ACU\)](#) can help you compare compute performance across Azure SKUs.

Azure compute unit (ACU)

4/9/2018 • 1 min 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	1:1
D1_v2-D15_v2	210 - 250*	1:1
DS1-DS14	160	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	VCPUs: CORE
------------	------------	-------------

G1-G5	180 - 240*	1:1
GS1-GS5	180 - 240*	1:1
H	290 - 300*	1:1
L4s-L32s	180 - 240*	1:1
M	160-180	2:1**

ACUs marked with a * use Intel® Turbo technology to increase CPU frequency and provide a performance boost. The amount of the boost can vary based on the VM size, workload, and other workloads running on the same host.

**Hyper-threaded.

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

4/11/2018 • 20 min 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

(3/23/2018 7:32:49 PM pbi 2050259)

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	6,514	56	0.86%	119
Standard_A1_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	1.9	6,162	195	3.17%	70
Standard_A1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	1.9	6,505	425	6.53%	63
Standard_A2_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	3.9	13,061	282	2.16%	112
Standard_A2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.9	12,270	390	3.18%	56
Standard_A2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	3.9	13,406	793	5.91%	35

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	16.0	13,148	130	0.98%	84
Standard_A2m_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	16.0	12,450	563	4.52%	35
Standard_A2m_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	16.0	12,929	988	7.64%	56
Standard_A4_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	8.0	26,037	356	1.37%	70
Standard_A4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	8.0	24,728	594	2.40%	77
Standard_A4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	8.0	26,549	1,375	5.18%	42
Standard_A4m_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	32.1	25,865	672	2.60%	70
Standard_A4m_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	32.1	24,646	1,104	4.48%	63

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	32.1	25,058	1,292	5.15%	35
Standard_A8_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	16.0	52,963	694	1.31%	98
Standard_A8_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	16.0	49,908	970	1.94%	35
Standard_A8_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	16.0	48,584	742	1.53%	77
Standard_A8m_v2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	2	64.4	52,900	815	1.54%	133
Standard_A8m_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	64.4	49,733	861	1.73%	84
Standard_A8m_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	64.4	48,494	501	1.03%	49

A0-7 Standard General Compute

(3/23/2018 9:06:07 PM pbi 2050259)

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_A0	AMD Opteron(tm) Processor 4171 HE	1	1	0.6	3,556	14	0.39%	21
Standard_A0	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	1	1	0.6	3,137	16	0.51%	70
Standard_A0	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	0.6	3,146	134	4.27%	63
Standard_A1	AMD Opteron(tm) Processor 4171 HE	1	1	1.7	6,862	169	2.47%	42
Standard_A1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	1	1	1.7	6,471	104	1.61%	98
Standard_A1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	1.7	6,211	262	4.22%	98
Standard_A2	AMD Opteron(tm) Processor 4171 HE	2	1	3.4	13,950	415	2.98%	35
Standard_A2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	3.4	13,205	187	1.41%	112
Standard_A2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.4	12,563	639	5.09%	84

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_A3	AMD Opteron(tm) Processor 4171 HE	4	1	6.9	28,069	679	2.42%	28
Standard_A3	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	6.9	26,238	236	0.90%	98
Standard_A3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	6.9	25,195	827	3.28%	77
Standard_A4	AMD Opteron(tm) Processor 4171 HE	8	2	14.0	56,604	305	0.54%	7
Standard_A4	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	14.0	53,271	577	1.08%	63
Standard_A4	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	14.0	49,744	1,118	2.25%	147
Standard_A5	AMD Opteron(tm) Processor 4171 HE	2	1	14.0	14,164	273	1.93%	21
Standard_A5	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	14.0	13,136	151	1.15%	98
Standard_A5	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	14.0	12,510	615	4.92%	84

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_A6	AMD Opteron(tm) Processor 4171 HE	4	1	28.1	28,272	392	1.39%	28
Standard_A6	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	28.1	26,165	294	1.13%	105
Standard_A6	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	28.1	25,178	1,009	4.01%	70
Standard_A7	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	56.3	52,949	1,114	2.10%	112
Standard_A7	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	56.3	49,677	894	1.80%	133

DSv3 - General Compute + Premium Storage

(3/23/2018 7:28:44 PM pbi 2050259)

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	8.0	20,259	729	3.60%	140
Standard_D2s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	8.0	20,364	1,007	4.94%	70

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D4s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	16.0	39,662	1,757	4.43%	182
Standard_D4s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	16.0	40,632	2,422	5.96%	168
Standard_D8s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	32.1	80,055	1,022	1.28%	133
Standard_D8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	32.1	80,639	2,844	3.53%	56
Standard_D16s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	64.4	158,407	3,267	2.06%	119
Standard_D16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	64.4	154,146	4,432	2.88%	63
Standard_D32s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	32	2	128.9	307,408	3,203	1.04%	70
Standard_D32s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	1	128.9	309,183	3,132	1.01%	56

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D32-8s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	128.9	80,160	3,912	4.88%	84
Standard_D32-8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	128.9	84,406	1,939	2.30%	70
Standard_D32-16s_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	32	2	128.9	157,154	2,152	1.37%	84
Standard_D32-16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	1	128.9	155,460	3,663	2.36%	112
Standard_D64s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	257.9	615,873	7,266	1.18%	140
Standard_D64-16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	257.9	170,309	2,169	1.27%	98
Standard_D64-32s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	257.9	306,578	4,095	1.34%	98

Dv3 - General Compute

(3/23/2018 7:32:37 PM pbi 2050259)

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	8.0	20,791	1,531	7.36%	175
Standard_D2_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	8.0	21,326	1,622	7.61%	84
Standard_D4_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	16.0	39,978	1,853	4.63%	98
Standard_D4_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	16.0	41,842	2,798	6.69%	77
Standard_D8_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	32.1	80,559	1,990	2.47%	154
Standard_D8_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	32.1	79,711	4,368	5.48%	63
Standard_D16_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	64.4	160,309	3,371	2.10%	133
Standard_D16_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	64.4	155,447	3,426	2.20%	56

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D32_v3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	32	2	128.9	309,021	4,128	1.34%	105
Standard_D32_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	1	128.9	311,375	3,714	1.19%	49
Standard_D64_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	257.9	613,424	19,225	3.13%	84

Esv3 - Memory Optimized + Premium Storage

(3/23/2018 7:31:01 PM pbi 2050259)

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	16.0	21,015	1,112	5.29%	210
Standard_E4s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	32.1	40,691	1,928	4.74%	287
Standard_E8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	64.4	79,841	2,856	3.58%	161
Standard_E16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	128.9	155,976	2,666	1.71%	161

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_E32s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	257.9	297,695	8,535	2.87%	140
Standard_E32-8s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	2	257.9	86,375	7,300	8.45%	140
Standard_E32-16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	257.9	158,842	10,809	6.81%	189
Standard_E64s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	435.3	613,160	8,637	1.41%	63
Standard_E64-16s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	435.3	170,343	2,052	1.20%	126
Standard_E64-32s_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	435.3	307,110	3,759	1.22%	126
Standard_E64is_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	435.3	613,892	7,763	1.26%	140

Ev3 - Memory Optimized

(3/23/2018 7:29:35 PM pbi 2050259)

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	16.0	23,318	2,734	11.72%	245
Standard_E4_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	32.1	42,612	3,834	9.00%	154
Standard_E8_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	64.4	83,488	4,888	5.85%	189
Standard_E16_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	128.9	159,537	4,999	3.13%	175
Standard_E32_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	32	2	257.9	306,311	8,547	2.79%	196
Standard_E64_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	435.3	605,599	36,245	5.98%	168
Standard_E64i_v3	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	64	2	435.3	618,198	10,022	1.62%	154

DSv2 - Storage Optimized

(3/23/2018 7:30:03 PM pbi 2050259)

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	14,691	626	4.26%	182
Standard_DS1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	3.4	14,577	1,120	7.68%	63
Standard_DS2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	6.9	29,156	1,095	3.76%	91
Standard_DS2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	6.9	29,157	1,945	6.67%	56
Standard_DS3_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	14.0	55,981	1,625	2.90%	98
Standard_DS3_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	14.0	57,921	2,628	4.54%	77
Standard_DS4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	28.1	115,636	1,963	1.70%	161
Standard_DS4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	28.1	111,527	494	0.44%	14

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	56.3	220,333	4,856	2.20%	91
Standard_DS5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	56.3	217,812	5,320	2.44%	35
Standard_DS5_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	56.3	222,116	4,445	2.00%	56
Standard_DS11_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	14.0	28,364	929	3.28%	105
Standard_DS11_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	14.0	28,772	1,351	4.69%	70
Standard_DS12_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	28.1	56,130	2,174	3.87%	119
Standard_DS12_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	28.1	58,490	2,670	4.56%	63
Standard_DS13_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	56.3	115,107	2,525	2.19%	126

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS13_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	56.3	111,429	1,111	1.00%	35
Standard_DS13-2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	56.3	28,933	1,176	4.06%	154
Standard_DS13-2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	56.3	30,336	1,635	5.39%	98
Standard_DS13-4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	56.3	57,644	1,893	3.28%	126
Standard_DS13-4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	56.3	57,927	1,306	2.26%	84
Standard_DS14_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	112.8	218,363	3,866	1.77%	154
Standard_DS14_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	112.8	224,791	4,363	1.94%	77
Standard_DS14-4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	2	112.8	56,398	2,675	4.74%	126

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS14-4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	112.8	61,709	1,468	2.38%	49
Standard_DS14-8_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	2	112.8	112,282	4,368	3.89%	147
Standard_DS14-8_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	112.8	111,469	3,747	3.36%	49
Standard_DS15_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	20	2	141.0	271,529	3,749	1.38%	189

Dv2 - General Compute

(3/23/2018 7:27:28 PM pbi 2050259)

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	15,027	963	6.41%	105
Standard_D1_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	3.4	15,296	1,696	11.09%	77
Standard_D2_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	6.9	29,060	1,405	4.83%	133

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D2_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	6.9	31,552	2,315	7.34%	63
Standard_D3_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	14.0	57,666	1,559	2.70%	168
Standard_D3_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	14.0	58,312	3,640	6.24%	77
Standard_D4_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	28.1	114,708	2,019	1.76%	147
Standard_D4_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	28.1	114,971	5,415	4.71%	42
Standard_D5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	56.3	221,546	5,093	2.30%	112
Standard_D5_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	56.3	214,933	3,741	1.74%	21
Standard_D5_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	56.3	225,410	5,421	2.41%	77

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D11_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	14.0	29,125	1,267	4.35%	154
Standard_D11_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	14.0	31,315	1,762	5.63%	63
Standard_D12_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	28.1	57,386	1,712	2.98%	168
Standard_D12_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	28.1	57,903	3,322	5.74%	112
Standard_D13_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	56.3	114,319	2,482	2.17%	126
Standard_D13_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	56.3	114,933	3,486	3.03%	49
Standard_D14_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	112.8	219,594	5,752	2.62%	119
Standard_D14_v2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	112.8	225,880	3,699	1.64%	77

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D15_v2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	20	2	141.0	275,417	6,457	2.34%	154

D - General Compute

(3/23/2018 7:28:16 PM pbi 2050259)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	1	1	3.4	10,331	303	2.93%	112
Standard_D1	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	1	1	3.4	14,547	916	6.30%	14
Standard_D2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	6.9	21,112	383	1.81%	119
Standard_D2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	6.9	27,545	173	0.63%	7
Standard_D3	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	14.0	41,910	974	2.32%	133
Standard_D3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	14.0	57,567	2,166	3.76%	21

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D4	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	28.1	85,255	971	1.14%	119
Standard_D4	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	28.1	112,324	1,367	1.22%	14
Standard_D11	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	14.0	21,172	479	2.26%	140
Standard_D11	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	14.0	29,935	1,702	5.69%	14
Standard_D12	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	28.1	41,788	1,185	2.84%	98
Standard_D12	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	28.1	57,105	761	1.33%	28
Standard_D13	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	56.3	85,180	1,395	1.64%	119
Standard_D13	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	56.3	113,421	1,695	1.49%	28
Standard_D14	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	16	2	112.8	165,497	3,376	2.04%	105

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_D14	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	112.8	217,813	4,869	2.24%	35

DS - Storage Optimized

(3/23/2018 7:34:52 PM pbi 2050259)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS1	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	1	1	3.4	10,666	134	1.25%	126
Standard_DS11	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	14.0	21,330	268	1.26%	49
Standard_DS12	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	28.1	42,225	555	1.31%	126
Standard_DS12	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	28.1	56,288	976	1.73%	14
Standard_DS13	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	56.3	85,292	1,303	1.53%	98
Standard_DS14	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	16	2	112.8	162,996	3,911	2.40%	70

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_DS14	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	112.8	217,038	3,540	1.63%	28
Standard_DS2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	2	1	6.9	21,398	193	0.90%	119
Standard_DS3	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	4	1	14.0	42,470	559	1.32%	126
Standard_DS3	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	14.0	55,664	385	0.69%	7
Standard_DS4	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz	8	1	28.1	84,956	1,087	1.28%	105

Fsv2 - Compute + Storage Optimized

(3/23/2018 7:33:11 PM pbi 2050259)

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	25,392	157	0.62%	49
Standard_F4s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	4	1	8.0	48,065	656	1.36%	42

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_F8s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	8	1	16.0	95,026	1,205	1.27%	49
Standard_F16s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	16	1	32.1	187,039	2,681	1.43%	42
Standard_F32s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	32	1	64.4	375,828	5,180	1.38%	70
Standard_F64s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	64	2	128.9	741,859	5,954	0.80%	49
Standard_F72s_v2	Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz	72	2	145.0	831,616	4,867	0.59%	42

F - Compute Optimized

(3/23/2018 7:28:54 PM pbi 2050259)

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	14,762	815	5.52%	175
Standard_F1	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	1.9	15,756	1,653	10.49%	91

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_F2	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.9	29,265	1,593	5.44%	182
Standard_F2	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	3.9	32,105	3,072	9.57%	70
Standard_F4	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	8.0	57,174	2,009	3.51%	182
Standard_F4	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	8.0	57,613	3,076	5.34%	91
Standard_F8	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	16.0	114,851	1,983	1.73%	182
Standard_F8	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	16.0	113,706	2,797	2.46%	105
Standard_F16	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	32.1	220,641	5,114	2.32%	140
Standard_F16	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	32.1	217,174	4,952	2.28%	28

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_F16	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	32.1	226,457	5,507	2.43%	49

Fs - Compute and Storage Optimized

(3/23/2018 7:30:14 PM pbi 2050259)

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	14,630	678	4.64%	203
Standard_F1s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	1	1	1.9	15,247	801	5.25%	91
Standard_F2s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	2	1	3.9	28,931	1,105	3.82%	133
Standard_F2s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	1	3.9	29,079	1,454	5.00%	77
Standard_F4s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	4	1	8.0	56,674	1,518	2.68%	189
Standard_F4s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	4	1	8.0	57,051	2,872	5.03%	91

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_F8s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	8	1	16.0	113,709	11,105	9.77%	196
Standard_F8s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	8	1	16.0	111,477	2,531	2.27%	70
Standard_F16s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	1	32.1	222,637	4,405	1.98%	140
Standard_F16s	Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz	16	2	32.1	218,297	4,284	1.96%	14
Standard_F16s	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	16	1	32.1	225,001	3,033	1.35%	98

G - Compute Optimized

(3/23/2018 7:27:25 PM pbi 2050259)

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	28.1	32,071	4,239	13.22%	182
Standard_G2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	4	1	56.3	60,598	6,048	9.98%	175

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_G3	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	112.8	111,058	6,536	5.89%	161
Standard_G4	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	1	225.7	200,516	1,833	0.91%	154
Standard_G5	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	32	2	451.5	395,591	5,192	1.31%	154

GS - Storage Optimized

(3/23/2018 7:25:12 PM pbi 2050259)

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	28.1	28,771	2,006	6.97%	231
Standard_GS2	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	4	1	56.3	54,947	3,699	6.73%	203
Standard_GS3	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	112.8	105,054	4,441	4.23%	182
Standard_GS4	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	1	225.7	199,189	12,092	6.07%	168

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_GS4-4	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	4	1	225.7	59,066	2,935	4.97%	196
Standard_GS4-8	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	225.7	107,076	3,209	3.00%	168
Standard_GS5	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	32	2	451.5	386,803	9,303	2.41%	161
Standard_GS5-8	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	2	451.5	116,094	1,935	1.67%	42
Standard_GS5-16	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	2	451.5	209,500	2,622	1.25%	63

H - High Performance Compute (HPC)

(3/23/2018 7:27:16 PM pbi 2050259)

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	56.3	140,445	2,840	2.02%	147
Standard_H8m	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	8	1	112.8	141,086	2,209	1.57%	147

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_H16	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	112.8	270,129	6,502	2.41%	56
Standard_H16m	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	225.7	272,667	6,686	2.45%	112
Standard_H16mr	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	225.7	272,683	6,484	2.38%	70
Standard_H16r	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	16	2	112.8	271,822	6,126	2.25%	98

HPC - A8-11

(3/23/2018 7:35:10 PM pbi 2050259)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_A8	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz	8	1	56.3	117,148	1,877	1.60%	189
Standard_A9	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz	16	2	112.8	225,608	7,532	3.34%	147
Standard_A10	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz	8	1	56.3	117,638	1,988	1.69%	168

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_A11	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz	16	2	112.8	225,980	7,067	3.13%	161

Ls - Storage Optimized

(3/23/2018 7:58:51 PM pbi 2050259)

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	32.1	55,962	3,567	6.37%	154
Standard_L8s	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	8	1	64.4	106,482	3,178	2.98%	168
Standard_L16s	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	16	1	128.9	197,906	12,605	6.37%	168
Standard_L32s	Intel(R) Xeon(R) CPU E5-2698B v3 @ 2.00GHz	32	2	257.9	388,652	6,274	1.61%	126

M - Memory Optimized

(3/23/2018 8:57:07 PM pbi 2050259)

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_M64-32ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	1,806.2	339,412	4,655	1.37%	21

VM SIZE	CPU	VCPUS	NUMA NODES	MEMORY(GIB)	AVG SCORE	STDDEV	STDDEV%	#RUNS
Standard_M64ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	1,806.2	662,070	16,539	2.50%	70
Standard_M64s	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	2	1,032.1	659,757	22,439	3.40%	21
Standard_M128-32ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	64	4	3,923.0	332,861	6,061	1.82%	42
Standard_M128-64ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	128	4	3,923.0	656,788	16,341	2.49%	35
Standard_M128ms	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	128	4	3,923.0	1,275,328	16,544	1.30%	70
Standard_M128s	Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz	128	4	2,064.3	1,275,445	19,510	1.53%	42

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](#).

Linux on distributions endorsed by Azure

4/25/2018 • 4 min 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+	In kernel	Package: In repo under "WALinuxAgent" Source code: GitHub
SUSE Linux Enterprise	SLES/SLES for SAP 11 SP4 12 SP1+	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 For Ubuntu 12.04 support on Azure please refer to the [EOL notice](#).

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.

Credativ

<http://www.credativ.co.uk/credativ-blog/debian-images-microsoft-azure>

Credativ 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

<http://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

<http://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

<http://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

<http://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.

Planned maintenance for Linux virtual machines

3/22/2018 • 4 min to read • [Edit Online](#)

Azure periodically performs updates to improve the reliability, performance, and security of the host infrastructure for virtual machines. These updates range from patching software components in the hosting environment (like operating system, hypervisor, and various agents deployed on the host), upgrading networking components, to hardware decommissioning. The 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 a reboot-less update is possible, Azure uses memory preserving maintenance to pause the VM while the host is updated or the VM is moved to an already updated host altogether.
- If maintenance requires a reboot, you get a notice of when the maintenance is planned. In these cases, you'll also be given a time window where you can start the maintenance yourself, at a time that works for you.

This page describes how Microsoft Azure performs both types of maintenance. For more information about unplanned events (outages), see [Manage the availability of virtual machines](#) for [Windows](#) or [Linux](#).

Applications running in a virtual machine can gather information about upcoming updates by using the Azure Metadata Service for [Windows](#) or [Linux](#).

For "how-to" information on managing planned maintenance, see "Handling planned maintenance notifications" for [Linux](#) or [Windows](#).

Memory preserving maintenance

When updates don't require a full reboot, memory preserving maintenance mechanisms are used to limit the impact to the virtual machine. The virtual machine is paused for up to 30 seconds, preserving the memory in RAM, while the hosting environment applies the necessary updates and patches, or moves the VM to an already updated host. The virtual machine is then resumed and the clock of the virtual machine is automatically synchronized.

For VMs in availability sets, update domains are updated one at a time. All VMs in one update domain (UD) are paused, updated and then resumed before planned maintenance moves on to the next UD.

Some applications may be impacted by these types of updates. Applications that perform real-time event processing, like media streaming or transcoding, or high throughput networking scenarios, may not be designed to tolerate a 30 second pause. In case the VM is being move to a different host, some sensitive workloads might notice a slight performance degradation in the few minutes leading up to the Virtual Machine pause.

Maintenance requiring a reboot

When VMs need to be rebooted for planned maintenance, you are notified in advance. Planned maintenance has two phases: the self-service window and a scheduled maintenance window.

The **self-service window** lets you initiate the maintenance on your VMs. During this time, you can query each VM to see their status and check the result of your last maintenance request.

When you start self-service maintenance, your VM is moved to a node that has already been updated and then powers it back on. Because the VM reboots, the temporary disk is lost and dynamic IP addresses associated with virtual network interface are updated.

If you start self-service maintenance and there is an error during the process, the operation is stopped, the VM is not updated and it is also removed from the planned maintenance iteration. You will be contacted in a later time

with a new schedule and offered a new opportunity to do self-service maintenance.

When the self-service window has passed, the **scheduled maintenance window** begins. During this time window, you can still query for the maintenance window, but no longer be able to start the maintenance yourself.

For information on managing maintenance requiring a reboot, see "Handling planned maintenance notifications" for [Linux](#) or [Windows](#).

Availability Considerations during Planned Maintenance

If you decide to wait until the planned maintenance window, there are a few things to consider for maintaining the highest availability of your VMs.

Paired Regions

Each Azure region is paired with another region within the same geography, together they make a regional pair. During planned maintenance, Azure will only update the VMs in a single region of a region pair. For example, when updating the Virtual Machines in North Central US, Azure will not update any Virtual Machines 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. This ensures that during either an outage or maintenance events, at least one virtual machine is available.

Within an availability set, individual VMs are spread across up to 20 update domains (UDs). During planned maintenance, only a single update domain is impacted at any given time. Be aware that the order of update domains being impacted does not necessarily happen sequentially.

Virtual machine scale sets are an Azure compute resource that enables you to deploy and manage a set of identical VMs as a single resource. The scale set is automatically deployed across update domains, like VMs in an availability set. Just like with availability sets, with scale sets only a single update domain is impacted at any given time.

For more information about configuring your virtual machines for high availability, see Manage the availability of your virtual machines for [Windows](#) or [Linux](#).

Next steps

For information on managing maintenance requiring a reboot, see [Handling planned maintenance notifications](#).

About disks storage for Azure Linux VMs

4/9/2018 • 9 min to read • [Edit Online](#)

Just like any other computer, virtual machines in Azure use disks as a place to store an operating system, applications, and data. All Azure virtual machines have at least two disks – a Linux operating system disk and a temporary disk. The operating system disk is created from an image, and both the operating system disk and the image are actually virtual hard disks (VHDs) stored in an Azure storage account. Virtual machines also can have one or more data disks, that are also stored as VHDs.

In this article, we will talk about the different uses for the disks, and then discuss the different types of disks you can create and use. This article is also available for [Windows virtual machines](#).

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.

Disks used by VMs

Let's take a look at how the disks are used by the VMs.

Operating system disk

Every virtual machine has one attached operating system disk. It's registered as a SATA drive and is labeled /dev/sda by default. This disk has a maximum capacity of 2048 gigabytes (GB).

Temporary disk

Each VM contains a temporary 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](#) or when you [redeploy a VM](#). During a standard reboot of the VM, the data on the temporary drive should persist.

On Linux virtual machines, the disk is typically **/dev/sdb** and is formatted and mounted to **/mnt** by the Azure Linux Agent. The size of the temporary disk varies, based on the size of the virtual machine. For more information, see [Sizes for Linux virtual machines](#).

For more information on how Azure uses the temporary disk, see [Understanding the temporary drive on Microsoft Azure Virtual Machines](#)

Data disk

A data disk is a VHD 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 4095 GB. 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.

NOTE

For more details about virtual machines capacities, see [Sizes for Linux virtual machines](#).

Azure creates an operating system disk when you create a virtual machine from an image. If you use an image that includes data disks, Azure also creates the data disks when it creates the virtual machine. Otherwise, you add data disks after you create the virtual machine.

You can add data disks to a virtual machine at any time, by **attaching** the disk to the virtual machine. You can use a VHD that you've uploaded or copied to your storage account, or one that Azure creates for you. Attaching a data disk associates the VHD file with the VM, by placing a 'lease' on the VHD so it can't be deleted from storage while it's still attached.

About VHDS

The VHDS used in Azure are .vhd files stored as page blobs in a standard or premium storage account in Azure. For details about page blobs, see [Understanding block blobs and page blobs](#). For details about premium storage, see [High-performance premium storage and Azure VMs](#).

Azure supports the fixed disk VHD format. The fixed format lays the logical disk out linearly within the file, so that disk offset X is stored at blob offset X. A small footer at the end of the blob describes the properties of the VHD. Often, the fixed format wastes space because most disks have large unused ranges in them. However, Azure stores .vhd files in a sparse format, so you receive the benefits of both the fixed and dynamic disks at the same time. For more details, see [Getting started with virtual hard disks](#).

All .vhd files in Azure that you want to use as a source to create disks or images are read-only, except the .vhd files uploaded or copied to Azure storage by the user (which can be either read-write or read-only). When you create a disk or image, Azure makes copies of the source .vhd files. These copies can be read-only or read-and-write, depending on how you use the VHD.

When you create a virtual machine from an image, Azure creates a disk for the virtual machine that is a copy of the source .vhd file. To protect against accidental deletion, Azure places a lease on any source .vhd file that's used to create an image, an operating system disk, or a data disk.

Before you can delete a source .vhd file, you'll need to remove the lease by deleting the disk or image. To delete a .vhd file that is being used by a virtual machine as an operating system disk, you can delete the virtual machine, the operating system disk, and the source .vhd file all at once by deleting the virtual machine and deleting all associated disks. However, deleting a .vhd file that's a source for a data disk requires several steps in a set order. First you detach the disk from the virtual machine, then delete the disk, and then delete the .vhd file.

WARNING

If you delete a source .vhd file from storage, or delete your storage account, Microsoft can't recover that data for you.

Page blobs in Premium Storage are designed for use as VHDs only. Microsoft does not recommend storing other types of data in page blobs in Premium Storage, as the cost may be significantly greater. Use block blobs for storing data that is not in a VHD.

Types of disks

Azure Disks are designed for 99.999% availability. Azure Disks have consistently delivered enterprise-grade durability, with an industry-leading ZERO% Annualized Failure Rate.

There are two performance tiers for storage that you can choose from when creating your disks -- Standard Storage and Premium Storage. Also, there are two types of disks -- unmanaged and managed -- and they can reside in either performance tier.

Standard storage

Standard Storage is backed by HDDs, and delivers cost-effective storage while still being performant. Standard

storage can be replicated locally in one datacenter, or be geo-redundant with primary and secondary data centers. For more information about storage replication, please see [Azure Storage replication](#).

For more information about using Standard Storage with VM disks, please see [Standard Storage and Disks](#).

Premium storage

Premium Storage is backed by SSDs, and delivers high-performance, low-latency disk support for VMs running I/O-intensive workloads. Typically you can use Premium Storage with sizes that include an "s" in the series name. For example, there is the Dv3-Series and the Dsv3-series, the Dsv3-series can be used with Premium Storage. For more information, please see [Premium Storage](#).

Unmanaged disks

Unmanaged disks are the traditional type of disks that have been used by VMs. With these, you create your own storage account and specify that storage account when you create the disk. You have to make sure you don't put too many disks in the same storage account, because you could exceed the [scalability targets](#) of the storage account (20,000 IOPS, for example), resulting in the VMs being throttled. With unmanaged disks, you have to figure out how to maximize the use of one or more storage accounts to get the best performance out of your VMs.

Managed disks

Managed Disks handles the storage account creation/management in the background for you, and ensures that you do not have to worry about the scalability limits of the storage account. You simply specify the disk size and the performance tier (Standard/Premium), and Azure creates and manages the disk for you. Even as you add disks or scale the VM up and down, you don't have to worry about the storage being used.

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, please see the [Managed Disks Overview](#).

We recommend that you use Azure Managed Disks for new VMs, and that you convert your previous unmanaged disks to managed disks, to take advantage of the many features available in Managed Disks.

Disk comparison

The following table provides a comparison of Premium vs Standard for both unmanaged and managed disks to help you decide what to use.

	AZURE PREMIUM DISK	AZURE STANDARD DISK
Disk Type	Solid State Drives (SSD)	Hard Disk Drives (HDD)
Overview	SSD-based high-performance, low-latency disk support for VMs running IO-intensive workloads or hosting mission critical production environment	HDD-based cost effective disk support for Dev/Test VM scenarios
Scenario	Production and performance sensitive workloads	Dev/Test, non-critical, Infrequent access

	AZURE PREMIUM DISK	AZURE STANDARD DISK
Disk Size	P4: 32 GB (Managed Disks only) P6: 64 GB (Managed Disks only) P10: 128 GB P20: 512 GB P30: 1024 GB P40: 2048 GB P50: 4095 GB	Unmanaged Disks: 1 GB – 4 TB (4095 GB) Managed Disks: S4: 32 GB S6: 64 GB S10: 128 GB S20: 512 GB S30: 1024 GB S40: 2048 GB S50: 4095 GB
Max Throughput per Disk	250 MB/s	60 MB/s
Max IOPS per Disk	7500 IOPS	500 IOPS

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

- [Attach a disk](#) to add additional storage for your VM.
- [Create a snapshot](#).
- [Convert to managed disks](#).

Azure Managed Disks Overview

4/9/2018 • 8 min to read • [Edit Online](#)

Azure Managed Disks simplifies disk management for Azure IaaS VMs by managing the [storage accounts](#) associated with the VM disks. You only have to specify the type ([Premium](#) or [Standard](#)) and the size of disk you need, and Azure creates and manages the disk for you.

Benefits of managed disks

Let's take a look at some of the benefits you gain by using managed disks, starting with this Channel 9 video, [Better Azure VM Resiliency with Managed Disks](#).

Simple and scalable VM deployment

Managed Disks handles storage for you behind the scenes. Previously, you had to create storage accounts to hold the disks (VHD files) for your Azure VMs. When scaling up, you had to make sure you created additional storage accounts so you didn't exceed the IOPS limit for storage with any of your disks. With Managed Disks handling storage, you are no longer limited by the storage account limits (such as 20,000 IOPS / account). You also no longer have to copy your custom images (VHD files) to multiple storage accounts. You can manage them in a central location – one storage account per Azure region – and use them to create hundreds of VMs in a subscription.

Managed Disks will allow you to create up to 10,000 VM **disks** in a subscription, which will enable you to create thousands of **VMs** in a single subscription. This feature also further increases the scalability of [Virtual Machine Scale Sets \(VMSS\)](#) by allowing you to create up to a thousand VMs in a VMSS using a Marketplace image.

Better reliability for Availability Sets

Managed Disks provides 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 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.

Highly durable and available

Azure Disks are designed for 99.999% availability. Rest easier knowing that you have three replicas of your data that enables 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 IaaS disks, with an industry-leading ZERO% Annualized Failure Rate.

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 exposes 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 his 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.

Azure Backup service support

Use Azure Backup service with Managed Disks to create a backup job with time-based backups, easy VM

restoration and backup retention policies. Managed Disks only support Locally Redundant Storage (LRS) as the replication option; this means it keeps three copies of the data within a single region. For regional disaster recovery, you must backup your VM disks in a different region using [Azure Backup service](#) and a GRS storage account as backup vault. Currently Azure Backup supports data disk sizes up to 1TB for backup. Read more about this at [Using Azure Backup service for VMs with Managed Disks](#).

Pricing and Billing

When using Managed Disks, the following billing considerations apply:

- Storage Type
- Disk Size
- Number of transactions
- Outbound data transfers
- Managed Disk Snapshots (full disk copy)

Let's take a closer look at these.

Storage Type: Managed Disks offers 2 performance tiers: [Premium](#) (SSD-based) and [Standard](#) (HDD-based). The billing of a managed disk depends on which type of storage you have selected for the disk.

Disk Size: Billing for managed disks depends on the provisioned size of the disk. Azure maps the provisioned size (rounded up) to the nearest Managed Disks option as specified in the tables below. Each managed disk maps to one of the supported provisioned sizes and is billed accordingly. For example, if you create a standard managed disk and specify a provisioned size of 200 GB, you are billed as per the pricing of the S20 Disk type.

Here are the disk sizes available for a premium managed disk:

PREMIUM MANAGED DISK TYPE	P4	P6	P10	P15	P20	P30	P40	P50
Disk Size	32 GiB	64 GiB	128 GiB	256 GiB	512 GiB	1024 GiB (1 TiB)	2048 GiB (2 TiB)	4095 GiB (4 TiB)

Here are the disk sizes available for a standard managed disk:

STANDARD MANAGED DISK TYPE	S4	S6	S10	S20	S30	S40	S50
Disk Size	32 GiB	64 GiB	128 GiB	512 GiB	1024 GiB (1 TiB)	2048 GiB (2 TiB)	4095 GiB (4 TiB)

Number of transactions: You are billed for the number of transactions that you perform on a standard managed disk. There is no cost for transactions for a premium managed disk.

Outbound data transfers: [Outbound data transfers](#) (data going out of Azure data centers) incur billing for bandwidth usage.

For detailed information on pricing for Managed Disks, see [Managed Disks Pricing](#).

Managed Disk Snapshots

A Managed Snapshot is a read-only full copy of a managed disk which 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, snapshot will be billed only for the used data size of 10 GiB.

[Incremental snapshots](#) are currently not supported for Managed Disks, but will be supported in the future.

To learn more about how to create snapshots with Managed Disks, please check out these 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 (sys-prepped) VM. This captures in a single image all managed disks associated with a VM, including both the OS and data disks. This enables creating hundreds of VMs using your custom image without the need to copy or manage any storage accounts.

For information on creating images, please check out 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 2.0](#)

Images versus snapshots

You often see the word "image" used with VMs, and now you see "snapshots" as well. It's important to understand the difference between these. With Managed Disks, you can take an image of a generalized VM that has been deallocated. This image will include all of the disks attached to the VM. You can use this image to create a new VM, and it will include all of the disks.

A snapshot is a copy of a disk at the point in time it is taken. It only applies to one disk. If you have a VM that only has one disk (the OS), you can take a snapshot or an image of it and create a VM from either the snapshot or the image.

What if a VM has five disks and they are striped? You could take a snapshot of each of the disks, but there is no awareness within the VM of the state of the disks – the snapshots only know about that one disk. In this case, the snapshots would need to be coordinated with each other, and that is not currently supported.

Managed Disks and Encryption

There are two kinds of encryption to discuss in reference to managed disks. The first one 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 safeguard 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 is available. Starting June 10th, 2017, all new managed disks/snapshots/images and new data written to existing managed disks are automatically encrypted-at-rest with keys managed by Microsoft by default. 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 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. This is integrated with Azure Key Vault to allow

you to control and manage the disk encryption keys. For more information, please see [Azure Disk Encryption for Windows and Linux IaaS VMs](#).

Next steps

For more information about Managed Disks, please refer to the following articles.

Get started with Managed Disks

- [Create a VM using Resource Manager and PowerShell](#)
- [Create a Linux VM using the Azure CLI 2.0](#)
- [Attach a managed data disk to a Windows VM using PowerShell](#)
- [Add a managed disk to a Linux VM](#)
- [Managed Disks PowerShell Sample Scripts](#)
- [Use Managed Disks in Azure Resource Manager templates](#)

Compare Managed Disks storage options

- [Premium storage and disks](#)
- [Standard storage and disks](#)

Operational guidance

- [Migrate from AWS and other platforms to Managed Disks in Azure](#)
- [Convert Azure VMs to managed disks in Azure](#)

High-performance Premium Storage and managed disks for VMs

4/9/2018 • 21 min to read • [Edit Online](#)

Azure Premium Storage delivers high-performance, low-latency disk support for virtual machines (VMs) with input/output (I/O)-intensive workloads. VM disks that use Premium Storage store data on solid-state drives (SSDs). To take advantage of the speed and performance of premium storage disks, you can migrate existing VM disks to Premium Storage.

In Azure, you can attach several premium storage disks to a VM. Using multiple disks gives your applications up to 256 TB of storage per VM. With Premium Storage, your applications can achieve 80,000 I/O operations per second (IOPS) per VM, and a disk throughput of up to 2,000 megabytes per second (MB/s) per VM. Read operations give you very low latencies.

With Premium Storage, Azure offers the ability to truly lift-and-shift demanding enterprise applications like Dynamics AX, Dynamics CRM, Exchange Server, SAP Business Suite, and SharePoint farms to the cloud. You can run performance-intensive database workloads in applications like SQL Server, Oracle, MongoDB, MySQL, and Redis, which require consistent high performance and low latency.

NOTE

For the best performance for your application, we recommend that you migrate any VM disk that requires high IOPS to Premium Storage. If your disk does not require high IOPS, you can help limit costs by keeping it in standard Azure Storage. In standard storage, VM disk data is stored on hard disk drives (HDDs) instead of on SSDs.

Azure offers two ways to create premium storage disks for VMs:

- **Unmanaged disks**

The original method is to use unmanaged disks. In an unmanaged disk, you manage the storage accounts that you use to store the virtual hard disk (VHD) files that correspond to your VM disks. VHD files are stored as page blobs in Azure storage accounts.

- **Managed disks**

When you choose [Azure Managed Disks](#), Azure manages the storage accounts that you use for your VM disks. You specify the disk type (Premium or Standard) and the size of the disk that you need. Azure creates and manages the disk for you. You don't have to worry about placing the disks in multiple storage accounts to ensure that you stay within scalability limits for your storage accounts. Azure handles that for you.

We recommend that you choose managed disks, to take advantage of their many features.

To get started with Premium Storage, [create your free Azure account](#).

For information about migrating your existing VMs to Premium Storage, see [Convert a Windows VM from unmanaged disks to managed disks](#) or [Convert a Linux VM from unmanaged disks to managed disks](#).

NOTE

Premium Storage is available in most regions. For the list of available regions, see the row for **Disk Storage** in [Azure products available by region](#).

Features

Here are some of the features of Premium Storage:

- **Premium storage disks**

Premium Storage supports VM disks that can be attached to specific size-series VMs. Premium Storage supports DS-series, DSv2-series, GS-series, Ls-series, Fs-series, and Esv3-series VMs. You have a choice of seven disk sizes: P4 (32 GB), P6 (64 GB), P10 (128 GB), P20 (512 GB), P30 (1024 GB), P40 (2048 GB), P50 (4095 GB). P4 and P6 disk sizes are yet only supported for Managed Disks. Each disk size has its own performance specifications. Depending on your application requirements, you can attach one or more disks to your VM. We describe the specifications in more detail in [Premium Storage scalability and performance targets](#).

- **Premium page blobs**

Premium Storage supports page blobs. Use page blobs to store persistent, unmanaged disks for VMs in Premium Storage. Unlike standard Azure Storage, Premium Storage does not support block blobs, append blobs, files, tables, or queues. Premium page blobs support six sizes from P10 to P50, and P60 (8191GiB). P60 Premium page blob is not supported to be attached as VM disks.

Any object placed in a premium storage account will be a page blob. The page blob snaps to one of the supported provisioned sizes. This is why a premium storage account is not intended to be used to store tiny blobs.

- **Premium storage account**

To start using Premium Storage, create a premium storage account for unmanaged disks. In the [Azure portal](#), to create a premium storage account, choose the **Premium** performance tier. Select the **Locally-redundant storage (LRS)** replication option. You also can create a premium storage account by setting the performance tier to **Premium_LRS**. To change the performance tier, use one of the following approaches:

- [PowerShell for Azure Storage](#)
- [Azure CLI for Azure Storage](#)
- [Azure Storage Resource Provider REST API](#) (for Azure Resource Manager deployments) or one of the Azure Storage resource provider client libraries

To learn about premium storage account limits, see [Premium Storage scalability and performance targets](#).

- **Premium locally redundant storage**

A premium storage account supports only locally redundant storage as the replication option. Locally redundant storage keeps three copies of the data within a single region. For regional disaster recovery, you must back up your VM disks in a different region by using [Azure Backup](#). You also must use a geo-redundant storage (GRS) account as the backup vault.

Azure uses your storage account as a container for your unmanaged disks. When you create an Azure VM that supports Premium Storage with unmanaged disks, and you select a premium storage account, your operating system and data disks are stored in that storage account.

Supported VMs

Premium Storage supports B-series, DS-series, DSv2-series, DSv3-series, GS-series, Ls-series, M-series, and Fs-series VMs. You can use standard and premium storage disks with these VM types. You cannot use premium storage disks with VM series that are not Premium Storage-compatible.

For information about VM types and sizes in Azure for Windows, see [Windows VM sizes](#). For information about

VM types and sizes in Azure for Linux, see [Linux VM sizes](#).

These are some of the features of the DS-series, DSv2-series, GS-series, LS-series, and FS-series VMs:

- **Cloud service**

You can add DS-series VMs to a cloud service that has only DS-series VMs. Do not add DS-series VMs to an existing cloud service that has any type other than DS-series VMs. You can migrate your existing VHDS to a new cloud service that runs only DS-series VMs. If you want to use the same virtual IP address for the new cloud service that hosts your DS-series VMs, use [reserved IP addresses](#). GS-series VMs can be added to an existing cloud service that has only GS-series VMs.

- **Operating system disk**

You can set up your Premium Storage VM to use either a premium or a standard operating system disk. For the best experience, we recommend using a Premium Storage-based operating system disk.

- **Data disks**

You can use premium and standard disks in the same Premium Storage VM. With Premium Storage, you can provision a VM and attach several persistent data disks to the VM. If needed, to increase the capacity and performance of the volume, you can stripe across your disks.

NOTE

If you stripe premium storage data disks by using [Storage Spaces](#), set up Storage Spaces with 1 column for each disk that you use. Otherwise, overall performance of the striped volume might be lower than expected because of uneven distribution of traffic across the disks. By default, in Server Manager, you can set up columns for up to 8 disks. If you have more than 8 disks, use PowerShell to create the volume. Specify the number of columns manually. Otherwise, the Server Manager UI continues to use 8 columns, even if you have more disks. For example, if you have 32 disks in a single stripe set, specify 32 columns. To specify the number of columns the virtual disk uses, in the [New-VirtualDisk](#) PowerShell cmdlet, use the *NumberOfColumns* parameter. For more information, see [Storage Spaces Overview](#) and [Storage Spaces FAQs](#).

- **Cache**

VMs in the size series that support Premium Storage have a unique caching capability for high levels of throughput and latency. The caching capability exceeds underlying premium storage disk performance. You can set the disk caching policy on premium storage disks to **ReadOnly**, **ReadWrite**, or **None**. The default disk caching policy is **ReadOnly** for all premium data disks, and **ReadWrite** for operating system disks. For optimal performance for your application, use the correct cache setting. For example, for read-heavy or read-only data disks, such as SQL Server data files, set the disk caching policy to **ReadOnly**. For write-heavy or write-only data disks, such as SQL Server log files, set the disk caching policy to **None**. To learn more about optimizing your design with Premium Storage, see [Design for performance with Premium Storage](#).

- **Analytics**

To analyze VM performance by using disks in Premium Storage, turn on VM diagnostics in the [Azure portal](#). For more information, see [Azure VM monitoring with Azure Diagnostics Extension](#).

To see disk performance, use operating system-based tools like [Windows Performance Monitor](#) for Windows VMs and the [iostat](#) command for Linux VMs.

- **VM scale limits and performance**

Each Premium Storage-supported VM size has scale limits and performance specifications for IOPS, bandwidth, and the number of disks that can be attached per VM. When you use premium storage disks

with VMs, make sure that there is sufficient IOPS and bandwidth on your VM to drive disk traffic.

For example, a STANDARD_DS1 VM has a dedicated bandwidth of 32 MB/s for premium storage disk traffic. A P10 premium storage disk can provide a bandwidth of 100 MB/s. If a P10 premium storage disk is attached to this VM, it can only go up to 32 MB/s. It cannot use the maximum 100 MB/s that the P10 disk can provide.

Currently, the largest VM in the DS-series is the Standard_DS15_v2. The Standard_DS15_v2 can provide up to 960 MB/s across all disks. The largest VM in the GS-series is the Standard_GS5. The Standard_GS5 can provide up to 2,000 MB/s across all disks.

These limits are for disk traffic only. These limits don't include cache hits and network traffic. A separate bandwidth is available for VM network traffic. Bandwidth for network traffic is different from the dedicated bandwidth used by premium storage disks.

For the most up-to-date information about maximum IOPS and throughput (bandwidth) for Premium Storage-supported VMs, see [Windows VM sizes](#) or [Linux VM sizes](#).

For more information about premium storage disks and their IOPS and throughput limits, see the table in the next section.

Scalability and performance targets

In this section, we describe the scalability and performance targets to consider when you use Premium Storage.

Premium storage accounts have the following scalability targets:

TOTAL ACCOUNT CAPACITY	TOTAL BANDWIDTH FOR A LOCALLY REDUNDANT STORAGE ACCOUNT
Disk capacity: 35 TB Snapshot capacity: 10 TB	Up to 50 gigabits per second for inbound ¹ + outbound ²

¹ All data (requests) that are sent to a storage account

² All data (responses) that are received from a storage account

For more information, see [Azure Storage scalability and performance targets](#).

If you are using premium storage accounts for unmanaged disks and your application exceeds the scalability targets of a single storage account, you might want to migrate to managed disks. If you don't want to migrate to managed disks, build your application to use multiple storage accounts. Then, partition your data across those storage accounts. For example, if you want to attach 51-TB disks across multiple VMs, spread them across two storage accounts. 35 TB is the limit for a single premium storage account. Make sure that a single premium storage account never has more than 35 TB of provisioned disks.

Premium Storage disk limits

When you provision a premium storage disk, the size of the disk determines the maximum IOPS and throughput (bandwidth). Azure offers seven types of premium storage disks: P4(Managed Disks only), P6(Managed Disks only), P10, P20, P30, P40, and P50. Each premium storage disk type has specific limits for IOPS and throughput. Limits for the disk types are described in the following table:

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)

Premium Disks Type	P4	P6	P10	P15	P20	P30	P40	P50
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

NOTE

Make sure sufficient bandwidth is available on your VM to drive disk traffic, as described in [Premium Storage-supported VMs](#). Otherwise, your disk throughput and IOPS is constrained to lower values. Maximum throughput and IOPS are based on the VM limits, not on the disk limits described in the preceding table.

Here are some important things to know about Premium Storage scalability and performance targets:

- **Provisioned capacity and performance**

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.

- **Disk size**

Azure maps the disk size (rounded up) to the nearest premium storage disk option, as specified in the table in the preceding section. For example, a disk size of 100 GB is classified as a P10 option. It can perform up to 500 IOPS, with up to 100-MB/s throughput. Similarly, a disk of size 400 GB is classified as a P20. It can perform up to 2,300 IOPS, with 150-MB/s throughput.

NOTE

You can easily increase the size of existing disks. For example, you might want to increase the size of a 30-GB disk to 128 GB, or even to 1 TB. Or, you might want to convert your P20 disk to a P30 disk because you need more capacity or more IOPS and throughput.

- **I/O size**

The size of an I/O is 256 KB. If the data being transferred is less than 256 KB, it is considered 1 I/O unit. Larger I/O sizes are counted as multiple I/Os of size 256 KB. For example, 1,100 KB I/O is counted as 5 I/O units.

- **Throughput**

The throughput limit includes writes to the disk, and it includes disk read operations that aren't served from the cache. For example, a P10 disk has 100-MB/s throughput per disk. Some examples of valid throughput for a P10 disk are shown in the following table:

MAX THROUGHPUT PER P10 DISK	NON-CACHE READS FROM DISK	NON-CACHE WRITES TO DISK
100 MB/s	100 MB/s	0

MAX THROUGHPUT PER P10 DISK	NON-CACHE READS FROM DISK	NON-CACHE WRITES TO DISK
100 MB/s	0	100 MB/s
100 MB/s	60 MB/s	40 MB/s

- **Cache hits**

Cache hits are not limited by the allocated IOPS or throughput of the disk. For example, when you use a data disk with a **ReadOnly** cache setting on a VM that is supported by Premium Storage, reads that are served from the cache are not subject to the IOPS and throughput caps of the disk. If the workload of a disk is predominantly reads, you might get very high throughput. The cache is subject to separate IOPS and throughput limits at the VM level, based on the VM size. DS-series VMs have roughly 4,000 IOPS and 33-MB/s throughput per core for cache and local SSD I/Os. GS-series VMs have a limit of 5,000 IOPS and 50-MB/s throughput per core for cache and local SSD I/Os.

Throttling

Throttling might occur, if your application IOPS or throughput exceeds the allocated limits for a premium storage disk. Throttling also might occur if your total disk traffic across all disks on the VM exceeds the disk bandwidth limit available for the VM. To avoid throttling, we recommend that you limit the number of pending I/O requests for the disk. Use a limit based on scalability and performance targets for the disk you have provisioned, and on the disk bandwidth available to the VM.

Your application can achieve the lowest latency when it is designed to avoid throttling. However, if the number of pending I/O requests for the disk is too small, your application cannot take advantage of the maximum IOPS and throughput levels that are available to the disk.

The following examples demonstrate how to calculate throttling levels. All calculations are based on an I/O unit size of 256 KB.

Example 1

Your application has processed 495 I/O units of 16-KB size in one second on a P10 disk. The I/O units are counted as 495 IOPS. If you try a 2-MB I/O in the same second, the total of I/O units is equal to $495 + \frac{2,048}{256} = 495 + 8$ IOPS. This is because $2 \text{ MB} / 256 \text{ KB} = 2,048 \text{ KB} / 256 \text{ KB} = 8 \text{ I/O units}$, when the I/O unit size is 256 KB. Because the sum of 495 + 8 exceeds the 500 IOPS limit for the disk, throttling occurs.

Example 2

Your application has processed 400 I/O units of 256-KB size on a P10 disk. The total bandwidth consumed is $(400 \times 256) / 1,024 \text{ KB} = 100 \text{ MB/s}$. A P10 disk has a throughput limit of 100 MB/s. If your application tries to perform more I/O operations in that second, it is throttled because it exceeds the allocated limit.

Example 3

You have a DS4 VM with two P30 disks attached. Each P30 disk is capable of 200-MB/s throughput. However, a DS4 VM has a total disk bandwidth capacity of 256 MB/s. You cannot drive both attached disks to the maximum throughput on this DS4 VM at the same time. To resolve this, you can sustain traffic of 200 MB/s on one disk and 56 MB/s on the other disk. If the sum of your disk traffic goes over 256 MB/s, disk traffic is throttled.

NOTE

If your disk traffic mostly consists of small I/O sizes, your application likely will hit the IOPS limit before the throughput limit. However, if the disk traffic mostly consists of large I/O sizes, your application likely will hit the throughput limit first, instead of the IOPS limit. You can maximize your application's IOPS and throughput capacity by using optimal I/O sizes. Also, you can limit the number of pending I/O requests for a disk.

To learn more about designing for high performance by using Premium Storage, see [Design for performance with Premium Storage](#).

Snapshots and Copy Blob

To the Storage service, the VHD file is a page blob. You can take snapshots of page blobs, and copy them to another location, such as to a different storage account.

Unmanaged disks

Create [incremental snapshots](#) for unmanaged premium disks the same way you use snapshots with standard storage. Premium Storage supports only locally redundant storage as the replication option. We recommend that you create snapshots, and then copy the snapshots to a geo-redundant standard storage account. For more information, see [Azure Storage redundancy options](#).

If a disk is attached to a VM, some API operations on the disk are not permitted. For example, you cannot perform a [Copy Blob](#) operation on that blob if the disk is attached to a VM. Instead, first create a snapshot of that blob by using the [Snapshot Blob](#) REST API. Then, perform the [Copy Blob](#) of the snapshot to copy the attached disk.

Alternatively, you can detach the disk, and then perform any necessary operations.

The following limits apply to premium storage blob snapshots:

PREMIUM STORAGE LIMIT	VALUE
Maximum number of snapshots per blob	100
Storage account capacity for snapshots (Includes data in snapshots only. Does not include data in base blob.)	10 TB
Minimum time between consecutive snapshots	10 minutes

To maintain geo-redundant copies of your snapshots, you can copy snapshots from a premium storage account to a geo-redundant standard storage account by using AzCopy or Copy Blob. For more information, see [Transfer data with the AzCopy command-line utility](#) and [Copy Blob](#).

For detailed information about performing REST operations against page blobs in a premium storage account, see [Blob service operations with Azure Premium Storage](#).

Managed disks

A snapshot for a managed disk is a read-only copy of the managed disk. The snapshot is stored as a standard managed disk. Currently, [incremental snapshots](#) are not supported for managed disks. To learn how to take a snapshot for a managed disk, see [Create a copy of a VHD stored as an Azure managed disk by using managed snapshots in Windows](#) or [Create a copy of a VHD stored as an Azure managed disk by using managed snapshots in Linux](#).

If a managed disk is attached to a VM, some API operations on the disk are not permitted. For example, you cannot generate a shared access signature (SAS) to perform a copy operation while the disk is attached to a VM. Instead, first create a snapshot of the disk, and then perform the copy of the snapshot. Alternately, you can detach the disk and then generate an SAS to perform the copy operation.

Premium Storage for Linux VMs

You can use the following information to help you set up your Linux VMs in Premium Storage:

To achieve scalability targets in Premium Storage, for all premium storage 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 Azure Premium Storage. For better performance and stability with Premium Storage, we recommend that you upgrade your VMs to one of these versions, at a minimum (or to a later version). 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. Note that 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>
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 are 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 computer.

Pricing and billing

When you use Premium Storage, the following billing considerations apply:

- **Premium storage disk and blob size**

Billing for a premium storage disk or blob depends on the provisioned size of the disk or blob. Azure maps the provisioned size (rounded up) to the nearest premium storage disk option. For details, see the table in [Premium Storage scalability and performance targets](#). Each disk maps to a supported provisioned disk size, and is billed accordingly. Billing for any provisioned disk is prorated hourly by using the monthly price for the Premium Storage offer. For example, if you provisioned a P10 disk and deleted it after 20 hours, you are billed for the P10 offering prorated to 20 hours. This is regardless of the amount of actual data written to the disk or the IOPS and throughput used.

- **Premium unmanaged disks snapshots**

Snapshots on premium unmanaged disks are billed for the additional capacity used by the snapshots. For more information about snapshots, see [Create a snapshot of a blob](#).

- **Premium managed disks snapshots**

A snapshot of a managed disk is a read-only copy of the disk. The disk is stored as a standard managed disk. A snapshot costs the same as a standard managed disk. For example, if you take a snapshot of a 128-GB premium managed disk, the cost of the snapshot is equivalent to a 128-GB standard managed disk.

- **Outbound data transfers**

[Outbound data transfers](#) (data going out of Azure datacenters) incur billing for bandwidth usage.

For detailed information about pricing for Premium Storage, Premium Storage-supported VMs, and managed disks, see these articles:

- [Azure Storage pricing](#)
- [Virtual Machines pricing](#)
- [Managed disks pricing](#)

Azure Backup support

For regional disaster recovery, you must back up your VM disks in a different region by using [Azure Backup](#) and a GRS storage account as a backup vault.

To create a backup job with time-based backups, easy VM restoration, and backup retention policies, use Azure Backup. You can use Backup both with unmanaged and managed disks. For more information, see [Azure Backup for VMs with unmanaged disks](#) and [Azure Backup for VMs with managed disks](#).

Next steps

For more information about Premium Storage, see the following articles.

[Design and implement with Premium Storage](#)

- [Design for performance with Premium Storage](#)
- [Blob storage operations with Premium Storage](#)

Operational guidance

- [Migrate to Azure Premium Storage](#)

Blog posts

- [Azure Premium Storage generally available](#)
- [Announcing the GS-series: Adding Premium Storage support to the largest VMs in the public cloud](#)

Azure Premium Storage: Design for High Performance

11/1/2017 • 41 min to read • [Edit Online](#)

Overview

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.

Before you begin, if you are new to Premium Storage, first read the [Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads](#) 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 is 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. 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. 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.

Gather Application Performance Requirements

The first step in designing high performance applications running on Azure Premium Storage is, to understand the performance requirements of your application. After you gather 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			
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 section describes the tools you can use to gather the performance measurements.

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. See the section on [Benchmarking](#) to learn more. By doing so you can determine whether Premium Storage can match or surpass your application performance requirements. Then you can implement the same guidelines for your production application.

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 log on separate disks, collect this data for both disks. Below table describes counters for disks, processor 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
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 issues 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 form 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](#).

Optimizing 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.

Optimizing IOPS, Throughput and Latency at a glance

The table below summarizes all the performance factors and the steps to optimize IOPS, Throughput and Latency. The sections following this summary will describe each factor in much more depth.

	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 yields higher Throughput.	
VM size	Use a VM size that offers IOPS greater than your application requirement. See VM sizes and their IOPS limits here.	Use a VM size with Throughput limit greater than your application requirement. See VM sizes and their Throughput limits here.	Use a VM size that offers scale limits greater than your application requirement. See VM sizes and their limits here.
Disk size	Use a disk size that offers IOPS greater than your application requirement. See disk sizes and their IOPS limits here.	Use a disk size with Throughput limit greater than your application requirement. See disk sizes and their Throughput limits here.	Use a disk size that offers scale limits greater than your application requirement. See disk sizes and their limits here.
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.

	IOPS	THROUGHPUT	LATENCY
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. Note that 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. E.g., use stripe size of 64KB for SQL Server OLTP application.	Larger stripe size for sequential large IO pattern seen in Data Warehouse applications. E.g., use 256KB 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 very 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.

$$\text{IOPS} \times \text{IO Size} = \text{Throughput}$$

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 type 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 type 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_D S14	16	112 GB	OS = 1023 GB Local SSD = 224 GB	32	576 GB	50,000 IOPS 512 MB per second	4,000 IOPS and 33 MB per second
Standard_G S5	32	448 GB	OS = 1023 GB Local SSD = 896 GB	64	4224 GB	80,000 IOPS 2,000 MB per second	5,000 IOPS and 50 MB per second

To view a complete list of all available Azure VM sizes, refer to [Windows VM sizes](#) or [Linux VM sizes](#). Choose a VM size that can meet and scale to your desired application performance requirements. In addition to this, take into account following important considerations when choosing VM sizes.

Scale Limits

The maximum IOPS limits per VM and per disk are different and independent of each other. Make sure that the application is driving IOPS within the limits of the VM as well as the premium disks attached to it. Otherwise, application performance will experience throttling.

As an example, suppose an application requirement is a maximum of 4,000 IOPS. To achieve this, you provision a P30 disk on a DS1 VM. The P30 disk can deliver up to 5,000 IOPS. However, the DS1 VM is limited to 3,200 IOPS. Consequently, the application performance will be constrained by the VM limit at 3,200 IOPS and there will be degraded performance. To prevent this situation, choose a VM and disk size that will both meet application requirements.

Cost of Operation

In many cases, it is possible that your overall cost of operation using Premium Storage is lower than using Standard Storage.

For example, consider an application requiring 16,000 IOPS. To achieve this performance, you will need a Standard_D14 Azure IaaS VM, which can give a maximum IOPS of 16,000 using 32 standard storage 1TB disks. Each 1TB 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 seven disk sizes 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 seven disks sizes and their capabilities. P4 and P6 sizes are currently only supported for Managed Disks.

PREMIUM DISKS TYPE	P4	P6	P10	P20	P30	P40	P50
Disk size	32 GB	64 GB	128 GB	512 GB	1024 GB (1 TB)	2048 GB (2 TB)	4095 GB (4 TB)
IOPS per disk	120	240	500	2300	5000	7500	7500
Throughput per disk	25 MB per second	50 MB per second	100 MB per second	150 MB per second	200 MB per second	250 MB per second	250 MB 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

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	None

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.

DISK CACHING SETTING	RECOMMENDATION ON WHEN TO USE THIS SETTING
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.

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 8 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 64KB for OLTP workloads and 256KB 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 4 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.

Benchmarking

Benchmarking is the process of simulating different workloads on your application and measuring the application performance for each workload. Using the steps described in an earlier section, you have gathered the application performance requirements. 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 section, 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 will be 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 [Create and use a Premium Storage account for a virtual machine data disk](#).

Warming up the Cache

The disk with ReadOnly host caching will be 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 which 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.

lometer

[Download the lometer tool](#) on the VM.

Test file

lometer uses a test file that is stored on the volume on which you will 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 200GB 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 64K request size and create specifications for Random Writes and Reads.

ACCESS SPECIFICATION	REQUEST SIZE	RANDOM %	READ %
RandomWrites_64K	64K	100	0
RandomReads_64K	64K	100	100

Running 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	1MB	100	0

NAME	REQUEST SIZE	RANDOM %	READ %
RandomReads_1MB	1MB	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 2hrs on the "Test Setup" tab.

SCENARIO	TARGET VOLUME	NAME	DURATION
Initialize Cache Disk	CacheReads	RandomWrites_1MB	2hrs

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 2hrs on the "Test Setup" tab.

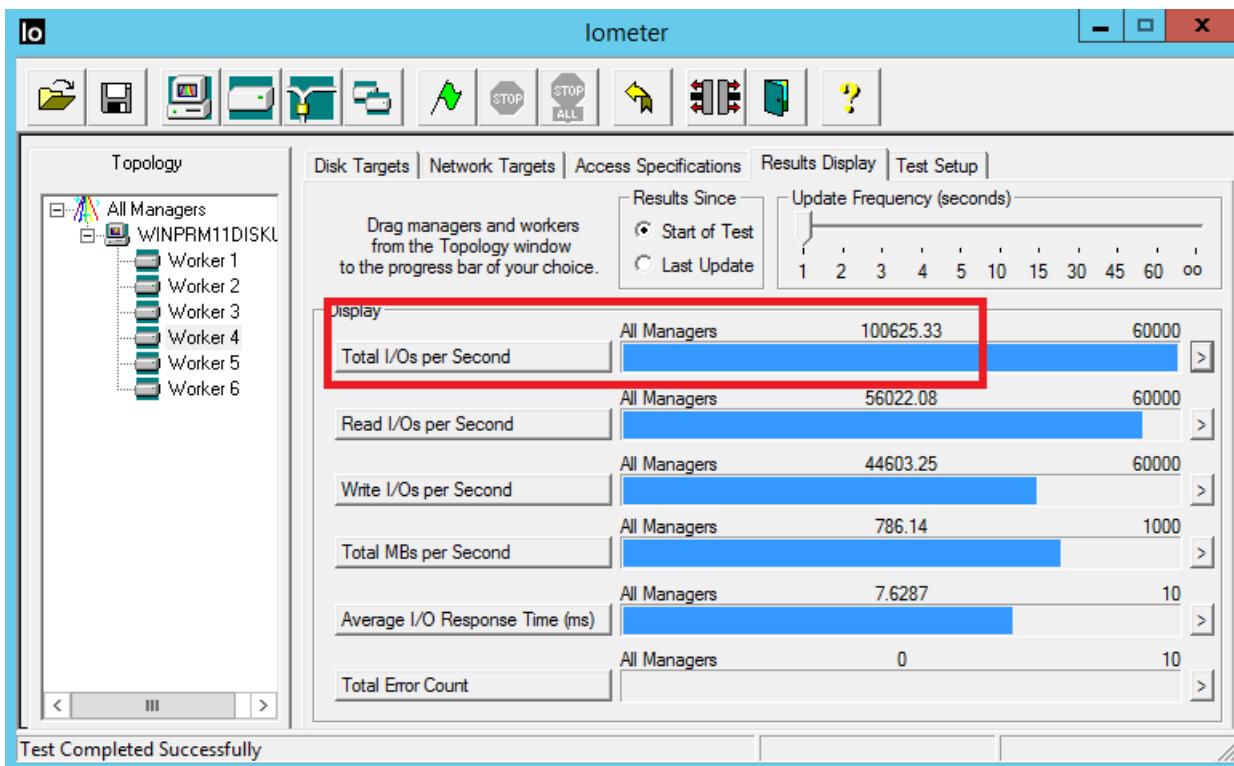
SCENARIO	TARGET VOLUME	NAME	DURATION
Warm up Cache Disk	CacheReads	RandomReads_1MB	2hrs

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 8KB and a high queue depth of 128 is used.

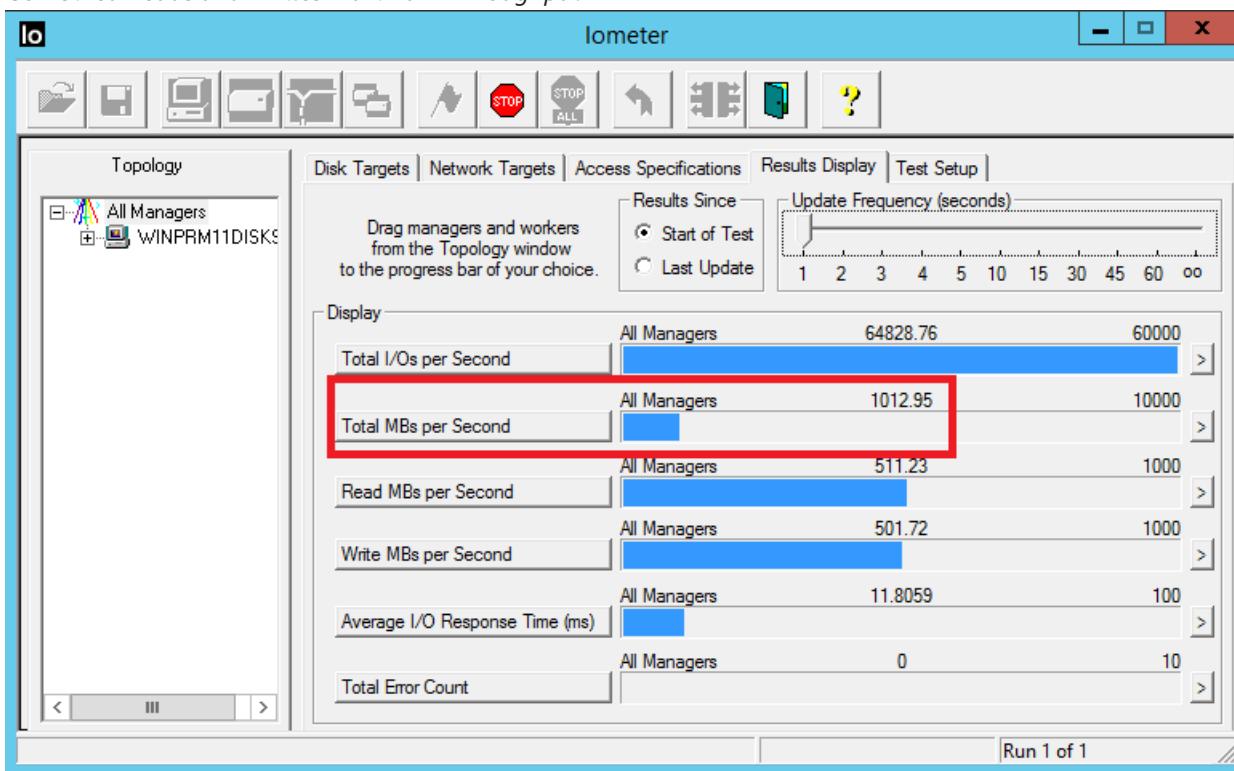
TEST SCENARIO	TARGET VOLUME	NAME	RESULT
Max. Read IOPS	CacheReads	RandomWrites_8K	50,000 IOPS
Max. Write IOPS	NoCacheWrites	RandomReads_8K	64,000 IOPS
Max. Combined IOPS	CacheReads	RandomWrites_8K	100,000 IOPS
NoCacheWrites	RandomReads_8K		
Max. Read MB/sec	CacheReads	RandomWrites_64K	524 MB/sec
Max. 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 will use four worker threads for driving Write operations and four worker threads for driving Read operations on the disks. The Write workers will be driving traffic on the "nocache" volume, which has 10 disks with cache set to "None". The Read workers will be driving traffic on the "readcache" volume, which has 1 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 8KB.
- 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 will be 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 "fioread.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 8KB.
- 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 will be able to see the number of read IOPS the VM and Premium disks are delivering. As shown in the sample below, the DS14 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 4KB.
- 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 will be 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 64KB and queue depth of 128.

Next Steps

Learn more about Azure Premium Storage:

- [Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads](#)

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](#)

Cost-effective Standard Storage and unmanaged and managed Azure VM disks

11/1/2017 • 7 min to read • [Edit Online](#)

Azure Standard Storage delivers reliable, low-cost disk support for VMs running latency-insensitive workloads. It also supports blobs, tables, queues, and files. With Standard Storage, the data is stored on hard disk drives (HDDs). When working with VMs, you can use standard storage disks for Dev/Test scenarios and less critical workloads, and premium storage disks for mission-critical production applications. Standard Storage is available in all Azure regions.

This article focuses on the use of standard storage for VM Disks. For more information about the use of storage with blobs, tables, queues, and files, please refer to the [Introduction to Storage](#).

Disk types

There are two ways to create standard disks for Azure VMs:

Unmanaged disks: This is the original method where you manage the storage accounts used to store the VHD files that correspond to the VM disks. VHD files are stored as page blobs in storage accounts. Unmanaged disks can be attached to any Azure VM size, including the VMs that primarily use Premium Storage, such as the DSv2 and GS series. Azure VMs support attaching several standard disks, allowing up to 256 TB of storage per VM.

Azure Managed Disks: This feature manages the storage accounts used for the VM disks for you. You specify the type (Premium or Standard) and size of disk you need, and Azure creates and manages the disk for you. You don't have to worry about placing the disks across multiple storage accounts in order to ensure you stay within the scalability limits for the storage accounts -- Azure handles that for you.

Even though both types of disks are available, we recommend using Managed Disks to take advantage of their many features.

To get started with Azure Standard Storage, visit [Get started for free](#).

For information on how to create a VM with Managed Disks, please see one of the following articles.

- [Create a VM using Resource Manager and PowerShell](#)
- [Create a Linux VM using the Azure CLI 2.0](#)

Standard Storage Features

Let's take a look at some of the features of Standard Storage. For more details, please see [Introduction to Azure Storage](#).

Standard Storage: Azure Standard Storage supports Azure Disks, Azure Blobs, Azure Files, Azure Tables, and Azure Queues. To use Standard Storage services, start with [Create an Azure Storage account](#).

Standard storage disks: Standard storage disks can be attached to all Azure VMs including size-series VMs used with Premium Storage such as the DSv2 and GS series. A standard storage disk can only be attached to one VM. However, you can attach one or more of these disks to a VM, up to the maximum disk count defined for that VM size. In the following section on Standard Storage Scalability and Performance Targets, we describe the specifications in more detail.

Standard page blob: Standard page blobs are used to hold persistent disks for VMs and can also be accessed

directly through REST like other types of Azure Blobs. [Page blobs](#) are a collection of 512-byte pages optimized for random read and write operations.

Storage Replication: In most regions, data in a standard storage account can be replicated locally or geo-replicated across multiple data centers. The four types of replication available are Locally-Redundant Storage (LRS), Zone-Redundant Storage (ZRS), Geo-Redundant Storage (GRS), and Read Access Geo-Redundant Storage (RA-GRS). Managed Disks in Standard Storage currently support Locally-Redundant Storage (LRS) only. For more information, please see [Storage Replication](#).

Scalability and Performance Targets

In this section, we describe the Scalability and Performance targets you need to consider when using standard storage.

Account limits – does not apply to managed disks

RESOURCE	DEFAULT LIMIT
TB per storage account	500 TB
Max ingress ¹ per storage account (US Regions)	10 Gbps if GRS/ZRS enabled, 20 Gbps for LRS
Max egress ¹ per storage account (US Regions)	20 Gbps if RA-GRS/GRS/ZRS enabled, 30 Gbps for LRS
Max ingress ¹ per storage account (European and Asian Regions)	5 Gbps if GRS/ZRS enabled, 10 Gbps for LRS
Max egress ¹ per storage account (European and Asian Regions)	10 Gbps if RA-GRS/GRS/ZRS enabled, 15 Gbps for LRS
Total Request Rate (assuming 1 KB object size) per storage account	Up to 20,000 IOPS, entities per second, or messages per second

¹ Ingress refers to all data (requests) being sent to a storage account. Egress refers to all data (responses) being received from a storage account.

For more information, see [Azure Storage Scalability and Performance Targets](#).

If the needs of your application exceed the scalability targets of a single storage account, build your application to use multiple storage accounts and partition your data across those storage accounts. Alternately, you can use Azure Managed Disks, and Azure manages the partitioning and placement of your data for you.

Standard Disks Limits

Unlike Premium Disks, the input/output operations per second (IOPS) and throughput (bandwidth) of Standard Disks are not provisioned. The performance of standard disks varies with the VM size to which the disk is attached, not to the size of the disk. You could expect to achieve up to the performance limit listed in the table below.

Standard disks limits (managed and unmanaged)

VM TIER	BASIC TIER VM	STANDARD TIER VM
Max Disk size	4095 GB	4095 GB
Max 8 KB IOPS per disk	Up to 300	Up to 500
Max Bandwidth per disk	Up to 60 MB/s	Up to 60 MB/s

If your workload requires high-performance, low-latency disk support, you should consider using Premium Storage. To know more benefits of Premium Storage, visit [High-Performance Premium Storage and Azure VM Disks](#).

Snapshots and copy blob

To the Storage service, the VHD file is a page blob. You can take snapshots of page blobs, and copy them to another location, such as a different storage account.

Unmanaged disks

You can create [incremental snapshots](#) for unmanaged standard disks in the same way you use snapshots with Standard Storage. We recommend that you create snapshots and then copy those snapshots to a geo-redundant standard storage account if your source disk is in a locally-redundant storage account. For more information, see [Azure Storage Redundancy Options](#).

If a disk is attached to a VM, certain API operations are not permitted on the disks. For example, you cannot perform a [Copy Blob](#) operation on that blob as long as the disk is attached to a VM. Instead, first create a snapshot of that blob by using the [Snapshot Blob](#) REST API method, and then perform the [Copy Blob](#) of the snapshot to copy the attached disk. Alternatively, you can detach the disk and then perform any necessary operations.

To maintain geo-redundant copies of your snapshots, you can copy snapshots from a locally-redundant storage account to a geo-redundant standard storage account by using AzCopy or Copy Blob. For more information, see [Transfer data with the AzCopy Command-Line Utility](#) and [Copy Blob](#).

For detailed information on performing REST operations against page blobs in standard storage accounts, see [Azure Storage Services REST API](#).

Managed disks

A snapshot for a managed disk is a read-only copy of the managed disk which is stored as a standard managed disk. Incremental Snapshots are not currently supported for Managed Disks but will be supported in the future.

If a managed disk is attached to a VM, certain API operations are not permitted on the disks. For example, you cannot generate a shared access signature (SAS) to perform a copy operation while the disk is attached to a VM. Instead, first create a snapshot of the disk, and then perform the copy of the snapshot. Alternately, you can detach the disk and then generate a shared access signature (SAS) to perform the copy operation.

Pricing and Billing

When using Standard Storage, the following billing considerations apply:

- Standard storage unmanaged disks/data size
- Standard managed disks
- Standard storage snapshots
- Outbound data transfers
- Transactions

Unmanaged storage data and disk size: For unmanaged disks and other data (blobs, tables, queues, and files), you are charged only for the amount of space you are using. For example, if you have a VM whose page blob is provisioned as 127 GB, but the VM is really only using 10 GB of space, you are billed for 10 GB of space. We support Standard storage up to 8191 GB, and Standard unmanaged disks up to 4095 GB.

Managed disks: Managed disks are billed on the provisioned size. If your disk is provisioned as a 10 GB disk and you are only using 5 GB, you are charged for the provision size of 10 GB.

Snapshots: Snapshots of standard disks are billed for the additional capacity used by the snapshots. For information on snapshots, see [Creating a Snapshot of a Blob](#).

Outbound data transfers: Outbound data transfers (data going out of Azure data centers) incur billing for bandwidth usage.

Transaction: Azure charges \$0.0036 per 100,000 transactions for standard storage. Transactions include both read and write operations to storage.

For detailed information on pricing for Standard Storage, Virtual Machines, and Managed Disks, see:

- [Azure Storage Pricing](#)
- [Virtual Machines Pricing](#)
- [Managed Disks Pricing](#)

Azure Backup service support

Virtual machines with unmanaged disks can be backed up using Azure Backup. [More details](#).

You can also use the Azure Backup service with Managed Disks to create a backup job with time-based backups, easy VM restoration and backup retention policies. You can read more about this at [Using Azure Backup service for VMs with Managed Disks](#).

Next steps

- [Introduction to Azure Storage](#)
- [Create a storage account](#)
- [Managed Disks Overview](#)
- [Create a VM using Resource Manager and PowerShell](#)
- [Create a Linux VM using the Azure CLI 2.0](#)

Scalability and performance targets for VM disks on Linux

11/16/2017 • 3 min to read • [Edit Online](#)

An Azure virtual machine supports attaching a number of data disks. This article describes scalability and performance targets for a VM's data disks. Use these targets to help decide 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 are not highly utilized at the same time, then the virtual machine can support a larger number of disks.

- **For Azure Managed Disks:** The disk limit for managed disks is per region and per disk type. The maximum limit, and also the default limit, is 10,000 managed disks per region and per disk type for a subscription. For example, you can create up to 10,000 standard managed disks and also 10,000 premium managed disks in a region, per subscription.

Managed snapshots and images count against the managed disks limit.

- **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 (20,000/300 IOPS per disk), and for a Standard Tier VM, it is about 40 (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

Standard managed virtual machine disks

STANDARD DISK TYPE	S4	S6	S10	S20	S30	S40	S50
Disk size	32 GB	64 GB	128 GB	512 GB	1024 GB (1 TB)	2048 GB (2TB)	4095 GB (4 TB)
IOPS per disk	500	500	500	500	500	500	500
Throughput per disk	60 MB/sec	60 MB/sec	60 MB/sec				

Premium managed virtual machine disks: per disk limits

PREMIUM DISKS TYPE	P4	P6	P10	P20	P30	P40	P50
Disk size	32 GB	64 GB	128 GB	512 GB	1024 GB (1 TB)	2048 GB (2 TB)	4095 GB (4 TB)
IOPS per disk	120	240	500	2300	5000	7500	7500
Throughput per disk	25 MB/sec	50 MB/sec	100 MB/sec	150 MB/sec	200 MB/sec	250 MB/sec	250 MB/sec

Premium managed virtual machine disks: per VM limits

RESOURCE	DEFAULT LIMIT
Max IOPS Per VM	80,000 IOPS with GS5 VM
Max 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	4095 GB	4095 GB
Max 8 KB IOPS per persistent disk	300	500
Max number of disks performing max 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
Max bandwidth per account (ingress + egress ¹)	<=50 Gbps

¹Ingress refers to all data (requests) being sent to a storage account. Egress refers to all data (responses) being 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	1024 GiB (1 TB)	2048 GiB (2 TB)	4095 GiB (4 TB)

PREMIUM STORAGE DISK TYPE	P10	P20	P30	P40	P50
Max IOPS per disk	500	2300	5000	7500	7500
Max throughput per disk	100 MB/s	150 MB/s	200 MB/s	250 MB/s	250 MB/s
Max number of disks per storage account	280	70	35	17	8

Premium unmanaged virtual machine disks: per VM limits

RESOURCE	DEFAULT LIMIT
Max IOPS Per VM	80,000 IOPS with GS5 VM
Max throughput per VM	2,000 MB/s with GS5 VM

See also

[Azure subscription and service limits, quotas, and constraints](#)

Backup and disaster recovery for Azure IaaS disks

11/1/2017 • 21 min 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. This is the main topic addressed by this document. 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 happens, 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 very 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 Storage disks.

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. This includes 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. This includes catastrophic events, such as if a datacenter is hit by a hurricane, earthquake, fire, or if there is a large-scale hardware unit failures. 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 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*, we mean 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 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 Storage disks](#), [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 Storage 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.

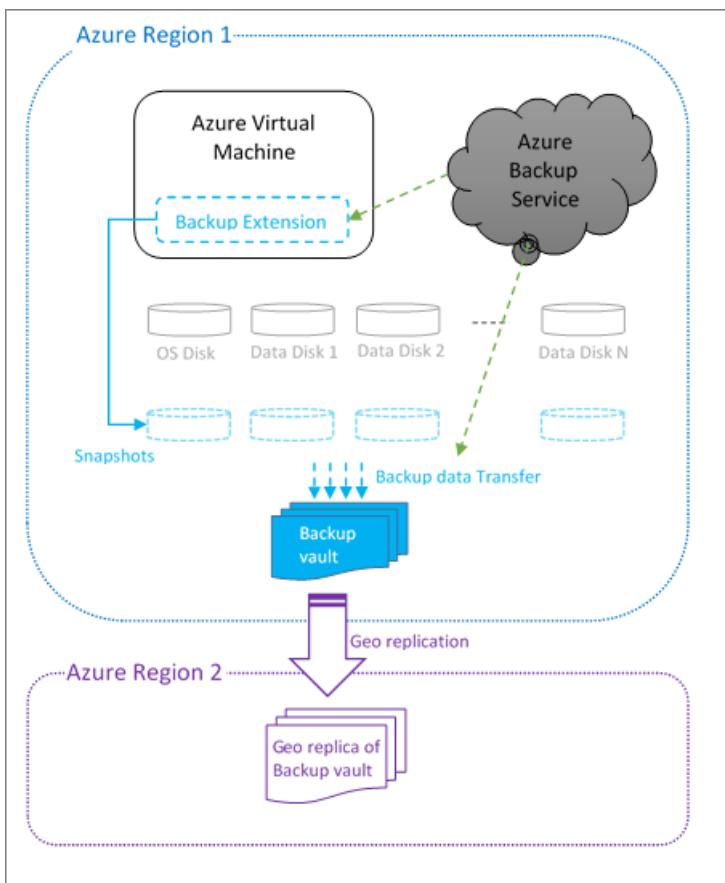
Your choices for high availability, backup, and DR at application or infrastructure levels can be represented as follows:

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 **Settings > Backup Configuration**. Ensure the **geo-redundant storage** option is selected by default. This 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 [restoring a VM](#) or 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, and 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. This means these snapshots might not be coordinated. This 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.

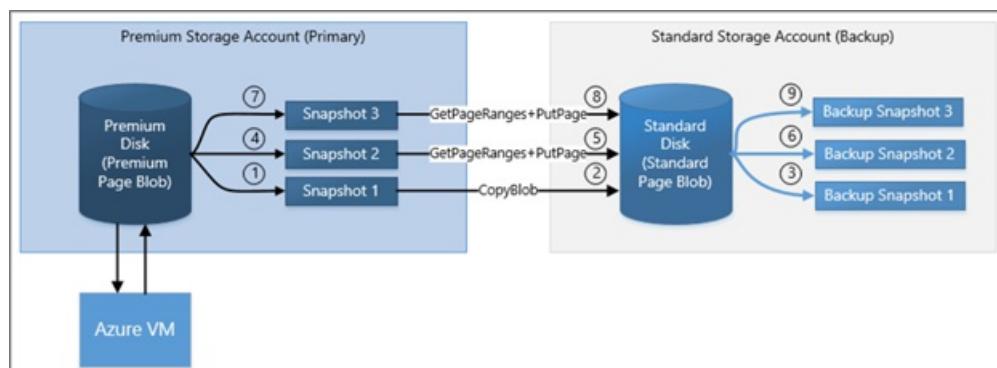
If you use geo-redundant storage or read-access geo-redundant storage for your disks, then the snapshots are replicated to the secondary region automatically. There can be a few minutes of lag before the replication. If the primary datacenter goes down before the snapshots finish replicating, you cannot 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 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 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 extremely 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.

Write Accelerator

5/10/2018 • 8 min 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 which 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

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.

IMPORTANT

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.

IMPORTANT

Enabling Write Accelerator for the operating system disk of the VM will reboot the 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 needs to be set to 'None' or 'Read Only'. All other caching modes are not supported.
- Snapshot on the Write Accelerator enabled disk is not supported yet. This restriction blocks Azure Backup Service ability to perform an application consistent snapshot of all disks of the virtual machine.
- Only smaller I/O sizes (<=32KiB) 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
M128ms	16	8000
M128s	16	8000
M64ms	8	4000
M64s	8	4000

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 through Power Shell

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" got added to the following cmdlets:

- Set-AzureRmVMOsDisk
- Add-AzureRmVMDataDisk
- Set-AzureRmVMDataDisk
- Add-AzureRmVmssDataDisk

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-AzureRmVmssStorageProfile

Not giving the parameter sets the property to false and will deliver disks that do not leverage Write Accelerator.

A new optional Boolean (non-nullable) parameter, "OsDiskWriteAccelerator" got added to the following cmdlets:

- Update-AzureRmVM
- Update-AzureRmVmss

Specify either \$true or \$false to control support of Azure Write Accelerator with the disks.

Examples of commands could look like:

```

New-AzureRmVMConfig | Set-AzureRmVMOsDisk | Add-AzureRmVMDataDisk -Name "datadisk1" | Add-AzureRmVMDataDisk -Name "logdisk1" -WriteAccelerator | New-AzureRmVM

Get-AzureRmVM | Update-AzureRmVM -OsDiskWriteAccelerator $true

New-AzureRmVmssConfig | Set-AzureRmVmssStorageProfile -OsDiskWriteAccelerator | Add-AzureRmVmssDataDisk -Name "datadisk1" -WriteAccelerator:$false | Add-AzureRmVmssDataDisk -Name "logdisk1" -WriteAccelerator | New-AzureRmVmss

Get-AzureRmVmss | Update-AzureRmVmss -OsDiskWriteAccelerator:$false

```

Two main scenarios can be scripted as shown in the following sections.

Adding a new disk supported by Write Accelerator

You can use this script to add a new disk to your VM. The disk created with this script is going to use Write Accelerator.

```

# 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-AzurermVM -ResourceGroupName $rgname -Name $vmname
#add a new VM data disk
Add-AzureRmVMDataDisk -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-AzureRmVM -ResourceGroupName $rgname -VM $vm

```

You need to adapt the names of VM, disk, resource group, size of the disk and LunID of the disk for your specific deployment.

Enabling Azure Write Accelerator on an existing Azure disk

If you need to enable Write Accelerator on an existing disk, you can use this script to perform the task:

```

#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-AzurermVM -ResourceGroupName $rgname -Name $vmname
#add a new VM data disk
Set-AzureRmVMDataDisk -VM $vm -Name $datadiskname -Caching None -WriteAccelerator:$newstatus
#Updates the VM with the disk config - does not require a reboot
Update-AzureRmVM -ResourceGroupName $rgname -VM $vm

```

You need to adapt the names of VM, disk, and resource group. The script above 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.

NOTE

Executing the script above will detach the disk specified, enable Write Accelerator against the disk, and then attach the disk again

Enabling through 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 through Azure CLI

You can use the [Azure CLI](#) to enable Write Accelerator.

To enable Write Accelerator on an existing disk, please use the command below, substituting the `diskName`, `VMName`, and `ResourceGroup` for your own:

```
az vm update -g group1 -n vm1 --write-accelerator 1=true
```

To attach a disk with Write Accelerator enabled please use the below command with your values:

```
az vm disk attach -g group1 --vm-name vm1 --disk d1 --enable-write-accelerator
```

To disable Write Accelerator, set the property to false:

```
az vm update -g group1 -n vm1 --write-accelerator 0=false 1=false
```

Enabling through Rest APIs

In order 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 you have to use:

```
Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object  
System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

Now you can install the armclient with the command below in cmd.exe or Power Shell

```
choco install armclient
```

Getting your current VM configuration

In order 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/dis-  
ks/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/dis-  
ks/mylittlesap_data1_1_754a1b8bb390468e9b4c429b81cc5f5a"  
          }  
        }  
      ]  
    }  
  }  
}
```

```

"/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":
        }
    ]
},
"diagnosticsProfile": {
    "bootDiagnostics": {
        "enabled": true,
        "storageUri": "https://mylittlesapdiag895.blob.core.windows.net/"
    }
},
"provisioningState": "Succeeded"
},
"type": "Microsoft.Compute/virtualMachines",
"location": "westeurope",
"id": ""
}
"/subscriptions/XXXXXXXXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/virtualMachines/mylittlesapVM",
"name": "mylittlesapVM"

```

Next step is to update the JSON file and to enable Write Accelerator on the disk called 'log1'. This step 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/XXXXXXXXXXXXXXXXXXXXXXXXXXXX/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 there is Write Accelerator enabled for one disk.

```
{
    "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",
        **"writeAcceleratorEnabled": true,**
        "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/XXXXXXXXXXXXXXXXXXXXXXXXXXXX/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/XXXXXXXXXXXXXXXXXXXXXXXXXXXX/resourceGroups/mylittlesap/providers/Microsoft.Compute/virtualMachines/mylittlesapVM",
"name": "mylittlesapVM"

```

From the point of the change on, the drive should be supported by Write Accelerator.

Troubleshoot storage resource deletion errors

5/2/2018 • 4 min to read • [Edit Online](#)

In certain scenarios, you may encounter one of the following errors occur while you are trying to delete an Azure storage account, container, or blob in an Azure Resource Manager deployment:

Failed to delete storage account 'StorageAccountName'. Error: The storage account cannot be deleted due to its artifacts being in use.

Failed to delete # out of # container(s):

vhds: There is currently a lease on the container and no lease ID was specified in the request.

Failed to delete # out of # blobs:

BlobName.vhd: There is currently a lease on the blob and no lease ID was specified in the request.

The VHDs used in Azure VMs are .vhd files stored as page blobs in a standard or premium storage account in Azure. For more information about Azure disks, see [About unmanaged and managed disk storage for Microsoft Azure Linux VMs](#).

Azure prevents deletion of a disk that is attached to a VM to prevent corruption. It also prevents deletion of containers and storage accounts that have a page blob that is attached to a VM.

The process to delete a storage account, container, or blob when receiving one of these errors is:

1. [Identify blobs attached to a VM](#)
2. [Delete VMs with attached OS disk](#)
3. [Detach all data disk\(s\) from remaining VM\(s\)](#)

Retry deleting the storage account, container, or blob after these steps are completed.

Step 1: Identify blob attached to a VM

Scenario 1: Deleting a blob – identify attached VM

1. Sign in to the [Azure portal](#).
2. On the Hub menu, select **All resources**. Go to the storage account, under **Blob Service** select **Containers**, and navigate to the blob to delete.
3. If the blob **Lease State** is **Leased**, then right-click and select **Edit Metadata** to open Blob metadata pane.

disks
Container

Upload Refresh Delete container Properties Access policy

Location: disks

Search blobs by prefix (case-sensitive)

NAME	MODIFIED	BLOB TYPE	SIZE	LEASE STATE
PercentileChart.0.1.1.pbviz	8/31/2017 4:13:15 PM	Block blob	40.22 KiB	Available
UnableToDel2-20170831-153106.vhd	8/31/2017 3:33:45 PM	Page blob	32 GiB	Leased
<input checked="" type="checkbox"/> UnableToDel2-20170831-153106.vhd	8/31/2017 3:36:01 PM	Page blob	64 GiB	Leased
UnableToDel2-20170831-153106.vhd	8/31/2017 3:36:01 PM	Page blob	16 GiB	Available
UnableToDel3-20170831-153106.vhd	8/31/2017 3:36:01 PM	Page blob	128 GiB	Leased
UnableToDel3-20170831-153106.vhd	8/31/2017 3:36:01 PM	Page blob	16 GiB	Leased

Edit
Download
Properties
Edit metadata
Break lease
Delete

- In Blob metadata pane, check and record the value for **MicrosoftAzureCompute_VMName**. This value is the name of the VM that the VHD is attached to. (See **important** if this field does not exist)
- In Blob metadata pane, check and record the value of **MicrosoftAzureCompute_DiskType**. This value identifies if the attached disk is OS or data disk (See **important** if this field does not exist).

disks
Container

Upload Refresh Delete container Properties

Location: disks

Search blobs by prefix (case-sensitive)

NAME	MODIFIED	BLOB TYPE
PercentileChart.0.1.1.pbviz	8/31/2017 4:13:15 PM	Block blob
<input checked="" type="checkbox"/> UnableToDel2-20170831-153106.vhd	8/31/2017 3:33:45 PM	Page blob
UnableToDel2-20170831-153106.vhd	8/31/2017 3:36:01 PM	Page blob
UnableToDel3-20170831-153106.vhd	8/31/2017 4:25:36 PM	Page blob
UnableToDel3-20170831-153106.vhd	8/31/2017 3:54:25 PM	Page blob
UnableToDel3-20170831-153106.vhd	8/31/2017 4:21:41 PM	Page blob

Blob metadata
UnableToDel2-20170831-153106.vhd

KEY	VALUE
MicrosoftAzureCompute_Disk...	4beadc1d-c43a-4ad4-aa2d-e...
MicrosoftAzureCompute_Disk...	UnableToDel2-20170831-153...
MicrosoftAzureCompute_Disk...	32
MicrosoftAzureCompute_Disk...	DataDisk
MicrosoftAzureCompute_Res...	unabletodelete
MicrosoftAzureCompute_VM...	UnableToDel2

- If the blob disk type is **OSDisk** follow [Step 2: Delete VM to detach OS disk](#). Otherwise, if the blob disk type is **DataDisk** follow the steps in [Step 3: Detach data disk from the VM](#).

IMPORTANT

If **MicrosoftAzureCompute_VMName** and **MicrosoftAzureCompute_DiskType** do not appear in the blob metadata, it indicates that the blob is explicitly leased and is not attached to a VM. Leased blobs cannot be deleted without breaking the lease first. To break lease, right-click on the blob and select **Break lease**. Leased blobs that are not attached to a VM prevent deletion of the blob, but do not prevent deletion of container or storage account.

Scenario 2: Deleting a container - identify all blob(s) within container that are attached to VMs

1. Sign in to the [Azure portal](#).
2. On the Hub menu, select **All resources**. Go to the storage account, under **Blob Service** select **Containers**, and find the container to be deleted.
3. Click to open the container and the list of blobs inside it will appear. Identify all the blobs with Blob Type = **Page blob** and Lease State = **Leased** from this list. Follow [Scenario 1](#) to identify the VM associated with each of these blobs.

NAME	MODIFIED	BLOB TYPE	SIZE	LEASE STATE
Percenti...	8/31/2017 4:13:15 PM	Block blob	40.22 KiB	Available
UnableT...	8/31/2017 3:33:45 PM	Page blob	32 GiB	Leased
UnableT...	8/31/2017 3:36:01 PM	Page blob	64 GiB	Leased
UnableT...	8/31/2017 4:25:36 PM	Page blob	16 GiB	Available
UnableT...	8/31/2017 3:54:25 PM	Page blob	128 GiB	Leased
UnableT...	8/31/2017 4:21:41 PM	Page blob	16 GiB	Leased

4. Follow [Step 2](#) and [Step 3](#) to delete VM(s) with **OSDisk** and detach **DataDisk**.

Scenario 3: Deleting storage account - identify all blob(s) within storage account that are attached to VMs

1. Sign in to the [Azure portal](#).
2. On the Hub menu, select **All resources**. Go to the storage account, under **Blob Service** select **Containers**.

NAME	LAST MODIFIED	PUBLIC ACCESS...	LEASE STATE
data	8/30/2017 1:53:36 PM	Private	Available
disks	8/30/2017 1:53:01 PM	Private	Leased
vhds	8/30/2017 12:50:17 PM	Private	Leased

3. In **Containers** pane, identify all containers where **Lease State** is **Leased** and follow [Scenario 2](#) for each

Leased container.

4. Follow [Step 2](#) and [Step 3](#) to delete VM(s) with **OSDisk** and detach **DataDisk**.

Step 2: Delete VM to detach OS disk

If the VHD is an OS disk, you must delete the VM before the attached VHD can be deleted. No additional action will be required for data disks attached to the same VM once these steps are completed:

1. Sign in to the [Azure portal](#).
2. On the Hub menu, select **Virtual Machines**.
3. Select the VM that the VHD is attached to.
4. Make sure that nothing is actively using the virtual machine, and that you no longer need the virtual machine.
5. At the top of the **Virtual Machine details** pane, select **Delete**, and then click **Yes** to confirm.
6. The VM should be deleted, but the VHD can be retained. However, the VHD should no longer be attached to a VM or have a lease on it. It may take a few minutes for the lease to be released. To verify that the lease is released, browse to the blob location and in the **Blob properties** pane, the **Lease Status** should be **Available**.

Step 3: Detach data disk from the VM

If the VHD is a data disk, detach the VHD from the VM to remove the lease:

1. Sign in to the [Azure portal](#).
2. On the Hub menu, select **Virtual Machines**.
3. Select the VM that the VHD is attached to.
4. Select **Disk**s on the **Virtual Machine details** pane.
5. Select the data disk to be deleted the VHD is attached to. You can determine which blob is attached in the disk by checking the URL of the VHD.
6. You can verify the blob location by clicking on the disk to check the path in **VHD URI** field.
7. Select **Edit** on the top of **Disk**s pane.
8. Click **detach icon** of the data disk to be deleted.

OS disk					
NAME	SIZE	STORAGE ACCOUNT TYPE	ENCRYPTION	HOST CACHING	
UnableToDel3	127 GiB	Standard_LRS	Not enabled	Read/write	

Data disks					
LUN	NAME	SIZE	STORAGE ACCOUNT TYPE	ENCRYPTION	HOST CACHING
0	UnableToDel3-20170831-155308	128 GiB	Standard_LRS	Not enabled	None
1	UnableToDel3-20170831-161851	16 GiB	Standard_LRS	Not enabled	None

9. Select **Save**. The disk is now detached from the VM, and the VHD is no longer leased. It may take a few minutes for the lease to be released. To verify that the lease has been released, browse to the blob location and in the **Blob properties** pane, the **Lease Status** value should be **Unlocked** or **Available**.

Troubleshoot unexpected reboots of VMs with attached VHDs

5/2/2018 • 1 min to read • [Edit Online](#)

If an Azure Virtual Machine (VM) has a large number of attached VHDs that are in the same storage account, you may exceed the scalability targets for an individual storage account, causing the VM to reboot unexpectedly. Check the minute metrics for the storage account (**TotalRequests/TotalIngress/TotalEgress**) for spikes that exceed the scalability targets for a storage account. See [Metrics show an increase in PercentThrottlingError](#) for assistance in determining whether throttling has occurred on your storage account.

In general, each individual input or output operation on a VHD from a Virtual Machine translates to **Get Page** or **Put Page** operations on the underlying page blob. Therefore, you can use the estimated IOPS for your environment to tune how many VHDs you can have in a single storage account based on the specific behavior of your application. Microsoft recommends having 40 or fewer disks in a single storage account. See [Azure Storage Scalability and Performance Targets](#) for details about scalability targets for storage accounts, in particular the total request rate and the total bandwidth for the type of storage account you are using.

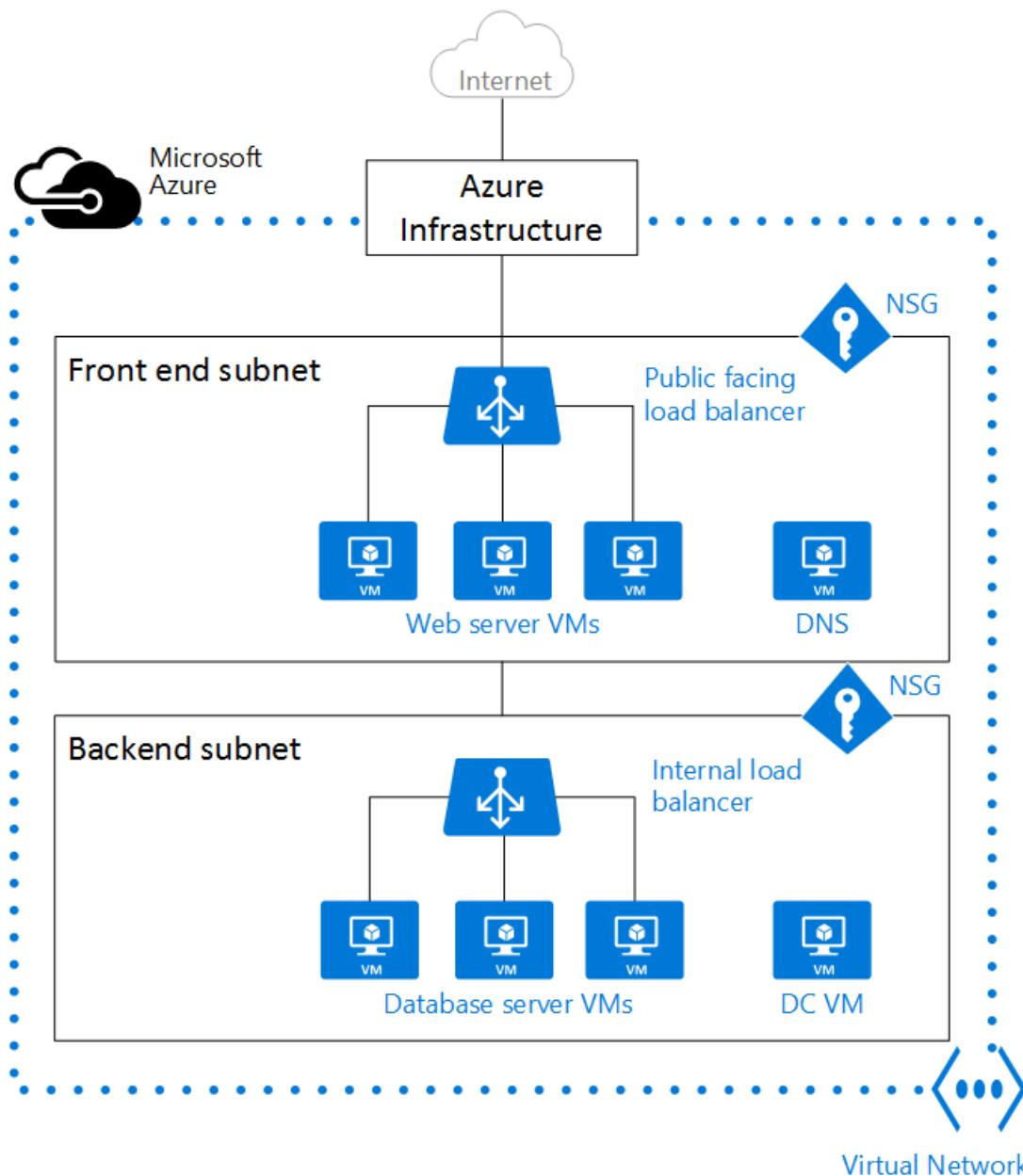
If you are exceeding the scalability targets for your storage account, place your VHDs in multiple storage accounts to reduce the activity in each individual account.

Virtual networks and virtual machines in Azure

3/20/2018 • 13 min to read • [Edit Online](#)

When you create an Azure virtual machine (VM), you must create a [virtual network](#) (VNet) or use an existing VNet. You also need to decide how your VMs are intended to be accessed on the VNet. It is important to [plan before creating resources](#) and make sure that you understand the [limits of networking resources](#).

In the following figure, VMs are represented as web servers and database servers. Each set of VMs are assigned to separate subnets in the VNet.



You can create a VNet before you create a VM or you can do so as you create a VM. You create these resources to support communication with a VM:

- Network interfaces
- IP addresses
- Virtual network and subnets

In addition to those basic resources, you should also consider these optional resources:

- Network security groups
- Load balancers

Network interfaces

A [network interface \(NIC\)](#) is the interconnection between a VM and a virtual network (VNet). A VM must have at least one NIC, but can have more than one, depending on the size of the VM you create. Learn about how many NICs each VM size supports for [Windows](#) or [Linux](#).

You can create a VM with multiple NICs, and add or remove NICs through the lifecycle of a VM. Multiple NICs allow a VM to connect to different subnets and send or receive traffic over the most appropriate interface.

If the VM is added to an availability set, all VMs within the availability set must have one or multiple NICs. VMs with more than one NIC aren't required to have the same number of NICs, but they must all have at least two.

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-AzureRmNetworkInterface with the -PublicIpAddressId parameter to provide the identifier of the public IP address that you previously created.
Azure CLI	To provide the identifier of the public IP address that you previously created, use az network nic create with the --public-ip-address parameter.
Template	Use Network Interface in a Virtual Network with Public IP Address as a guide for deploying a network interface using a template.

IP addresses

You can assign these types of [IP addresses](#) to a NIC in Azure:

- **Public IP addresses** - Used to communicate inbound and outbound (without network address translation (NAT)) with the Internet and other Azure resources not connected to a VNet. Assigning a public IP address to a NIC is optional. Public IP addresses have a nominal charge, and there's a maximum number that can be used per subscription.
- **Private IP addresses** - Used for communication within a VNet, your on-premises network, and the Internet (with NAT). You must assign at least one private IP address to a VM. To learn more about NAT in Azure, read [Understanding outbound connections in Azure](#).

You can assign public IP addresses to VMs or internet-facing load balancers. You can assign private IP addresses to VMs and internal load balancers. You assign IP addresses to a VM using a network interface.

There are two methods in which an IP address is allocated to a resource - dynamic or static. The default allocation method is dynamic, where an IP address is not allocated when it's created. Instead, the IP address is allocated when you create a VM or start a stopped VM. The IP address is released when you stop or delete the VM.

To ensure the IP address for the VM remains the same, you can set the allocation method explicitly to static. In this case, an IP address is assigned immediately. It is released only when you delete the VM or change its allocation method to dynamic.

This table lists the methods that you can use to create an IP address.

METHOD	DESCRIPTION
Azure portal	By default, public IP addresses are dynamic and the address associated to them may change when the VM is stopped or deleted. To guarantee that the VM always uses the same public IP address, create a static public IP address. By default, the portal assigns a dynamic private IP address to a NIC when creating a VM. You can change this IP address to static after the VM is created.
Azure PowerShell	You use New-AzureRmPublicIpAddress with the -AllocationMethod parameter as Dynamic or Static.
Azure CLI	You use az network public-ip create with the --allocation-method parameter as Dynamic or Static.
Template	Use Network Interface in a Virtual Network with Public IP Address as a guide for deploying a public IP address using a template.

After you create a public IP address, you can associate it with a VM by assigning it to a NIC.

Virtual network and subnets

A subnet is a range of IP addresses in the VNet. You can divide a VNet into multiple subnets for organization and security. Each NIC in a VM is connected to one subnet in one VNet. NICs connected to subnets (same or different) within a VNet can communicate with each other without any extra configuration.

When you set up a VNet, you specify the topology, including the available address spaces and subnets. If the VNet is to be connected to other VNets or on-premises networks, you must select address ranges that don't overlap. The IP addresses are private and can't be accessed from the Internet, which was true only for the non-routeable 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 <code>New-AzureRmVirtualNetworkSubnetConfig</code> and <code>New-AzureRmVirtualNetwork</code> to create a subnet and a VNet. You can also use <code>Add-AzureRmVirtualNetworkSubnetConfig</code> 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 <code>az network vnet create</code> with the subnet name.
Template	The easiest way to create a VNet and subnets is to download an existing template, such as Virtual Network with two subnets , and modify it for your needs.

Network security groups

A [network security group \(NSG\)](#) contains a list of Access Control List (ACL) rules that allow or deny network traffic to subnets, NICs, or both. NSGs can be associated with either subnets or individual NICs connected to a subnet. When an NSG is associated with a subnet, the ACL rules apply to all the VMs in that subnet. In addition, traffic to an individual NIC can be restricted by associating an NSG directly to a NIC.

NSGs contain two sets of rules: inbound and outbound. The priority for a rule must be unique within each set. Each rule has properties of protocol, source and destination port ranges, address prefixes, direction of traffic, priority, and access type.

All NSGs contain a set of default rules. The default rules cannot be deleted, but because they are assigned the lowest priority, they can be overridden by the rules that you create.

When you associate an NSG to a NIC, the network access rules in the NSG are applied only to that NIC. If an NSG is applied to a single NIC on a multi-NIC VM, it does not affect traffic to the other NICs. You can associate different NSGs to a NIC (or VM, depending on the deployment model) and the subnet that a NIC or VM is bound to. Priority is given based on the direction of traffic.

Be sure to [plan](#) your NSGs when you plan your VMs and VNet.

This table lists the methods that you can use to create a network security group.

METHOD	DESCRIPTION
Azure portal	When you create a VM in the Azure portal, an NSG is automatically created and associated to the NIC the portal creates. The name of the NSG is a combination of the name of the VM and -nsg . This NSG contains one inbound rule with a priority of 1000, service set to RDP, the protocol set to TCP, port set to 3389, and action set to Allow. If you want to allow any other inbound traffic to the VM, you must add additional rules to the NSG.

METHOD	DESCRIPTION
Azure PowerShell	Use New-AzureRmNetworkSecurityRuleConfig and provide the required rule information. Use New-AzureRmNetworkSecurityGroup to create the NSG. Use Set-AzureRmVirtualNetworkSubnetConfig to configure the NSG for the subnet. Use Set-AzureRmVirtualNetwork 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't currently create an internet-facing load balancer using the Azure portal.

METHOD	DESCRIPTION
Azure PowerShell	To provide the identifier of the public IP address that you previously created, use New-AzureRmLoadBalancerFrontendIpConfig with the -PublicIpAddress parameter. Use New-AzureRmLoadBalancerBackendAddressPoolConfig to create the configuration of the back-end address pool. Use New-AzureRmLoadBalancerInboundNatRuleConfig to create inbound NAT rules associated with the front-end IP configuration that you created. Use New-AzureRmLoadBalancerProbeConfig to create the probes that you need. Use New-AzureRmLoadBalancerRuleConfig to create the load balancer configuration. Use New-AzureRmLoadBalancer to create the load balancer.
Azure CLI	Use <code>az network lb create</code> to create the initial load balancer configuration. Use <code>az network lb frontend-ip create</code> to add the public IP address that you previously created. Use <code>az network lb address-pool create</code> to add the configuration of the back-end address pool. Use <code>az network lb inbound-nat-rule create</code> to add NAT rules. Use <code>az network lb rule create</code> to add the load balancer rules. Use <code>az network lb probe create</code> 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't currently create an internal load balancer using the Azure portal.
Azure PowerShell	To provide a private IP address in the network subnet, use New-AzureRmLoadBalancerFrontendIpConfig with the -PrivateIpAddress parameter. Use New-AzureRmLoadBalancerBackendAddressPoolConfig to create the configuration of the back-end address pool. Use New-AzureRmLoadBalancerInboundNatRuleConfig to create inbound NAT rules associated with the front-end IP configuration that you created. Use New-AzureRmLoadBalancerProbeConfig to create the probes that you need. Use New-AzureRmLoadBalancerRuleConfig to create the load balancer configuration. Use New-AzureRmLoadBalancer to create the load balancer.
Azure CLI	Use the <code>az network lb create</code> command to create the initial load balancer configuration. To define the private IP address, use <code>az network lb frontend-ip create</code> with the --private-ip-address parameter. Use <code>az network lb address-pool create</code> to add the configuration of the back-end address pool. Use <code>az network lb inbound-nat-rule create</code> to add NAT rules. Use <code>az network lb rule create</code> to add the load balancer rules. Use <code>az network lb probe create</code> 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. A dynamic IP address (DIP) is the internal IP address associated with a VM. You can allocate a static DIP to a VM. If you allocate a static DIP, you should consider using a specific subnet to avoid accidentally reusing a static DIP for another VM.

If you create a VM and later want to migrate it into a VNet, it is not a simple configuration change. You must redeploy the VM into the VNet. The easiest way to redeploy is to delete the VM, but not any disks attached to it, and then re-create the VM using the original disks in the VNet.

This table lists the methods that you can use to create a VM in a VNet.

METHOD	DESCRIPTION
Azure portal	Uses the default network settings that were previously mentioned to create a VM with a single NIC. To create a VM with multiple NICs, you must use a different method.
Azure PowerShell	Includes the use of <code>Add-AzureRmVMNetworkInterface</code> to add the NIC that you previously created to the VM configuration.
Azure CLI	Create and connect a VM to a Vnet, subnet, and NIC that build as individual steps.
Template	Use Very simple deployment of a Windows VM as a guide for deploying a VM using a template.

Next steps

For VM-specific steps on how to manage Azure virtual networks for VMs, see the [Windows](#) or [Linux](#) tutorials.

There are also tutorials on how to load balance VMs and create highly available applications for [Windows](#) or [Linux](#).

- Learn how to configure [user-defined routes and IP forwarding](#).
- Learn how to configure [VNet to VNet connections](#).
- Learn how to [Troubleshoot routes](#).

Automatically scale virtual machines in Azure

3/20/2018 • 4 min to read • [Edit Online](#)

You can easily [automatically scale](#) your [virtual machines \(VMs\)](#) when you use [virtual machine scale sets](#) and the [autoscaling feature of Azure Monitor](#). Your VMs need to be members of a scale set to be automatically scaled. This article provides information that enables you to better understand how to scale your VMs both vertically and horizontally using automatic and manual methods.

Horizontal or vertical scaling

The autoscale feature of Azure Monitor only scales horizontally, which is an increase ("out") or decrease ("in") of the number of VMs. Horizontal scaling is more flexible in a cloud situation as it allows you to run potentially thousands of VMs to handle load. You scale horizontally by either automatically or manually changing the capacity (or instance count) of the the scale set.

Vertical scaling keeps the same number of VMs, but makes the VMs more ("up") or less ("down") powerful. Power is measured in attributes such as memory, CPU speed, or disk space. Vertical scaling is dependent on the availability of larger hardware, which quickly hits an upper limit and can vary by region. Vertical scaling also usually requires a VM to stop and restart. You scale vertically by setting a new size in the configuration of the VMs in the scale set.

Using runbooks in [Azure Automation](#), you can easily [scale VMs in a scale set](#) up or down.

Create a virtual machine scale set

Scale sets make it easy for you to deploy and manage identical VMs as a set. You can create Linux or Windows scale sets using the [Azure portal](#), [Azure PowerShell](#), or the [Azure CLI](#). You can also create and manage scale sets with SDKs such as [Python](#) or [Node.js](#), or directly with the [REST APIs](#). Automatic scaling of VMs is accomplished by applying metrics and rules to the scale set.

Configure autoscale for a scale set

Automatic scaling provides the right number of VMs to handle the load on your application. It enables you to add VMs to handle increases in load and save money by removing VMs that are sitting idle. You specify a minimum and maximum number of VMs to run based on a set of rules. Having a minimum makes sure your application is always running even under no load. Having a maximum value limits your total possible hourly cost.

You can enable autoscale when you create the scale set using [Azure PowerShell](#) or [Azure CLI](#). You can also enable it after the scale set is created. You can create a scale set, install the extension, and configure autoscale using an [Azure Resource Manager template](#). In the Azure portal, enable autoscale from Azure Monitor, or enable autoscale from the scale set settings.

The screenshot shows the Azure portal interface for managing a virtual machine scale set. The left sidebar has a 'Scaling' tab highlighted with a red box. The main content area shows the 'Configure' tab selected. It displays an 'Override condition' section with an 'Instance count' slider set to 3. Below it is a message: 'Your autoscale configuration is disabled. To reinstate your configuration, enable autoscale.' A red box highlights the 'Enable autoscale' button.

Metrics

The autoscale feature of Azure Monitor enables you to scale the number of running VMs up or down based on **metrics**. By default, VMs provide basic host-level metrics for disk, network, and CPU usage. When you configure the collection of diagnostics data using the diagnostic extension, additional guest OS performance counters become available for disk, CPU, and memory.

Criteria

- * Time aggregation: Average
- * Metric name: Percentage CPU (1 minute time grain)
- * Time grain statistic: Average
- * Operator: Greater than
- * Threshold: 85%
- * Duration (in minutes): 30

If your application needs to scale based on metrics that are not available through the host, then the VMs in the scale set need to have either the [Linux diagnostic extension](#) or [Windows diagnostics extension](#) installed. If you create a scale set using the Azure portal, you need to also use Azure PowerShell or the Azure CLI to install the extension with the diagnostics configuration that you need.

Rules

Rules combine a metric with an action to be performed. When rule conditions are met, one or more autoscale actions are triggered. For example, you might have a rule defined that increases the number of VMs by 1 if the average CPU usage goes above 85 percent.

Action

- * Operation: Increase count by 1
- * Instance count: 1
- * Cool down (minutes): 5

Notifications

You can set up [triggers](#) so that specific web URLs are called or emails are sent based on the autoscale rules that you create. Webhooks allow you to route the Azure alert notifications to other systems for post-processing or custom notifications.

Manually scale VMs in a scale set

Horizontal

You can add or remove VMs by changing the capacity of the scale set. In the Azure portal, you can decrease or increase the number of VMs (shown as **instance count**) in the scale set by sliding the Override condition bar on the Scaling screen left or right.

Using Azure PowerShell, you need to get the scale set object using [Get-AzureRmVmss](#). You then set the **sku.capacity** property to the number of VMs that you want and update the scale set with [Update-AzureRmVmss](#). Using Azure CLI, you change the capacity with the **--new-capacity** parameter for the [az vmss scale](#) command.

Vertical

You can manually change the size of the VMs in the Azure portal on the Size screen for the scale set. You can use Azure PowerShell with [Get-AzureRmVmss](#), setting the image reference sku property, and then using [Update-AzureRmVmss](#) and [Update-AzureRmVmssInstance](#).

Next steps

- Learn more about scale sets in [Design Considerations for Scale Sets](#).

Use infrastructure automation tools with virtual machines in Azure

12/13/2017 • 7 min 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 [Visual Studio Team 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 basic VM](#).
- [Create a complete VM environment including supporting resources](#).

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. The following table outlines the current cloud-init enabled images availability on the Azure platform:

PUBLISHER	OFFER	SKU	VERSION	CLOUD-INIT READY
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

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.](#)

Visual Studio Team Services

[Team 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. Team Services integrates with Visual Studio and other editors to simplify usage. Team Services can also create and configure Azure VMs and then deploy code to them.

Learn how to:

- [Create a continuous integration pipeline with Team 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

4/9/2018 • 3 min 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.

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 log 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 log in. You can also use SSH keys to connect from a [Windows VM](#) to a Linux VM.

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](#).

How to monitor virtual machines in Azure

3/20/2018 • 5 min 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.

Logs

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 [OMS Log Analytics](#).

Advanced monitoring

- [Operations Management Suite \(OMS\)](#) provides monitoring, alerting, and alert remediation capabilities across cloud and on-premises assets. You can install an extension on a [Linux VM](#) or a [Windows VM](#) that installs the OMS agent, and enrolls the VM into an existing OMS workspace.
- [Log Analytics](#) is a service in OMS that monitors your cloud and on-premises environments to maintain their availability and performance. It collects data generated by resources in your cloud and on-premises environments and from other monitoring tools to provide analysis across multiple sources.

For Windows and Linux VMs, the recommended method for collecting logs and metrics 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](#).

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 • 1 min 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](#).

HPC, Batch, and Big Compute solutions using Azure VMs

5/11/2018 • 4 min to read • [Edit Online](#)

Organizations have large-scale computing needs. These Big Compute workloads include engineering design and analysis, financial risk calculations, image rendering, complex modeling, Monte Carlo simulations, and more.

Use the Azure cloud to efficiently run compute-intensive Linux and Windows workloads, from parallel batch jobs to traditional HPC simulations. Run your HPC and batch workloads on Azure infrastructure, with your choice of compute services, grid managers, Marketplace solutions, and vendor-hosted (SaaS) applications. Azure provides flexible solutions to distribute work and scale to thousands of VMs or cores and then scale down when you need fewer resources.

Solution options

- **Do-it-yourself solutions**

- Set up your own cluster environment in Azure virtual machines or [virtual machine scale sets](#).
- Lift and shift an on-premises cluster, or deploy a new cluster in Azure for additional capacity.
- Use Azure Resource Manager templates to deploy leading [workload managers](#), infrastructure, and [applications](#).
- Choose [HPC and GPU VM sizes](#) that include specialized hardware and network connections for MPI or GPU workloads.
- Add [high performance storage](#) for I/O-intensive workloads.

- **Hybrid solutions**

- Extend your on-premises solution to offload ("burst") peak workloads to Azure infrastructure
- Use cloud compute on-demand with your existing [workload manager](#).
- Take advantage of [HPC and GPU VM sizes](#) for MPI or GPU workloads.

- **Big Compute solutions as a service**

- Develop custom Big Compute solutions and workflows using [Azure Batch](#) and related [Azure services](#).
- Run Azure-enabled engineering and simulation solutions from vendors including [Altair](#), [Rescale](#), and [Cycle Computing](#) (now joined with Microsoft).
- Use a [Cray supercomputer](#) as a service hosted in Azure.

- **Marketplace solutions**

- Use the scale of [HPC applications](#) and [solutions](#) offered in the [Azure Marketplace](#).

The following sections provide more information about the supporting technologies and links to guidance.

Marketplace solutions

Visit the [Azure Marketplace](#) for Linux and Windows VM images and solutions designed for HPC. Examples include:

- [RogueWave CentOS-based HPC](#)
- [SUSE Linux Enterprise Server for HPC](#)
- [TIBCO Grid Server Engine](#)
- [Azure Data Science VM for Windows and Linux](#)
- [D3View](#)
- [UberCloud](#)

- Intel Cloud Edition for Lustre

HPC applications

Run custom or commercial HPC applications in Azure. Several examples in this section are benchmarked to scale efficiently with additional VMs or compute cores. Visit the [Azure Marketplace](#) for ready-to-deploy solutions.

NOTE

Check with the vendor of any commercial application for licensing or other restrictions for running in the cloud. Not all vendors offer pay-as-you-go licensing. You might need a licensing server in the cloud for your solution, or connect to an on-premises license server.

Engineering applications

- Altair RADIOSS
- ANSYS CFD
- MATLAB Distributed Computing Server
- StarCCM+
- OpenFOAM

Graphics and rendering

- Autodesk Maya, 3ds Max, and Arnold on Azure Batch

AI and deep learning

- Batch AI training for deep learning models
- Microsoft Cognitive Toolkit
- Deep Learning VM
- Batch Shipyard recipes for deep learning

HPC and GPU VM sizes

Azure offers a range of sizes for [Linux](#) and [Windows](#) VMs, including sizes designed for compute-intensive workloads. For example, H16r and H16mr VMs can connect to a high throughput back-end RDMA network. This cloud network can improve the performance of tightly coupled parallel applications running under [Microsoft MPI](#) or Intel MPI.

N-series VMs feature NVIDIA GPUs designed for compute-intensive or graphics-intensive applications including artificial intelligence (AI) learning and visualization.

Learn more:

- High performance compute sizes for [Linux](#) and [Windows](#) VMs
- GPU-enabled sizes for [Linux](#) and [Windows](#) VMs

Learn how to:

- [Set up a Linux RDMA cluster to run MPI applications](#)
- [Set up a Windows RDMA cluster with Microsoft HPC Pack to run MPI applications](#)
- [Use compute-intensive VMs in Batch pools](#)

Azure Batch

[Batch](#) is a platform service for running large-scale parallel and high-performance computing (HPC) applications efficiently in the cloud. Azure Batch schedules compute-intensive work to run on a managed pool of virtual

machines, and can automatically scale compute resources to meet the needs of your jobs.

SaaS providers or developers can use the Batch SDKs and tools to integrate HPC applications or container workloads with Azure, stage data to Azure, and build job execution pipelines.

Learn how to:

- [Get started developing with Batch](#)
- [Use Azure Batch code samples](#)
- [Use low-priority VMs with Batch](#)
- [Run containerized HPC workloads with Batch Shipyard](#)
- [Run parallel R workloads on Batch](#)
- [Run on-demand Spark jobs on Batch](#)

Workload managers

The following are examples of cluster and workload managers that can run in Azure infrastructure. Create stand-alone clusters in Azure VMs or burst to Azure VMs from an on-premises cluster.

- [Alces Flight Compute](#)
- [TIBCO DataSynapse GridServer](#)
- [Bright Cluster Manager](#)
- [IBM Spectrum Symphony and Symphony LSF](#)
- [PBS Pro](#)
- [Microsoft HPC Pack](#) - see options to run in [Windows](#) and [Linux](#) VMs

HPC storage

Large-scale Batch and HPC workloads have demands for data storage and access that exceed the capabilities of traditional cloud file systems. Implement parallel file system solutions in Azure such as [Lustre](#) and [BeeGFS](#).

Learn more:

- [Parallel virtual file systems on Azure](#)
- High performance cloud storage solutions from [Avere](#) (now [joined with Microsoft](#))

Related Azure services

Azure virtual machines, virtual machine scale sets, Batch, and related compute services are the foundation of most Azure HPC solutions. However, your solution can take advantage of many related Azure services. Here is a partial list:

Storage

- [Blob, table, and queue storage](#)
- [File storage](#)

Data and analytics

- [HDInsight](#)
- [Data Factory](#)
- [Data Lake Store](#)
- [Databricks](#)
- [SQL Database](#)

AI and machine learning

- [Machine Learning Services](#)
- [Batch AI](#)
- [Genomics](#)

Networking

- [Virtual Network](#)
- [ExpressRoute](#)

Containers

- [Container Service](#)
- [Azure Kubernetes Service \(AKS\)](#)
- [Container Registry](#)

Customer stories

Examples of customers that have solved business problems with Azure HPC solutions:

- [ANEO](#)
- [AXA Global P&C](#)
- [Axioma](#)
- [d3View](#)
- [EFS](#)
- [Hymans Robertson](#)
- [MetLife](#)
- [Microsoft Research](#)
- [Milliman](#)
- [Mitsubishi UFJ Securities International](#)
- [Schlumberger](#)
- [Towers Watson](#)

Next steps

- Learn more about Big Compute solutions for [engineering simulation, rendering, banking and capital markets](#), and [genomics](#).
- For the latest announcements, see the [Microsoft HPC and Batch team blog](#) and the [Azure blog](#).
- Use the managed and scalable Azure [Batch](#) service to run compute-intensive workloads, without managing underlying infrastructure [Learn more](#)

Example Azure infrastructure walkthrough for Linux VMs

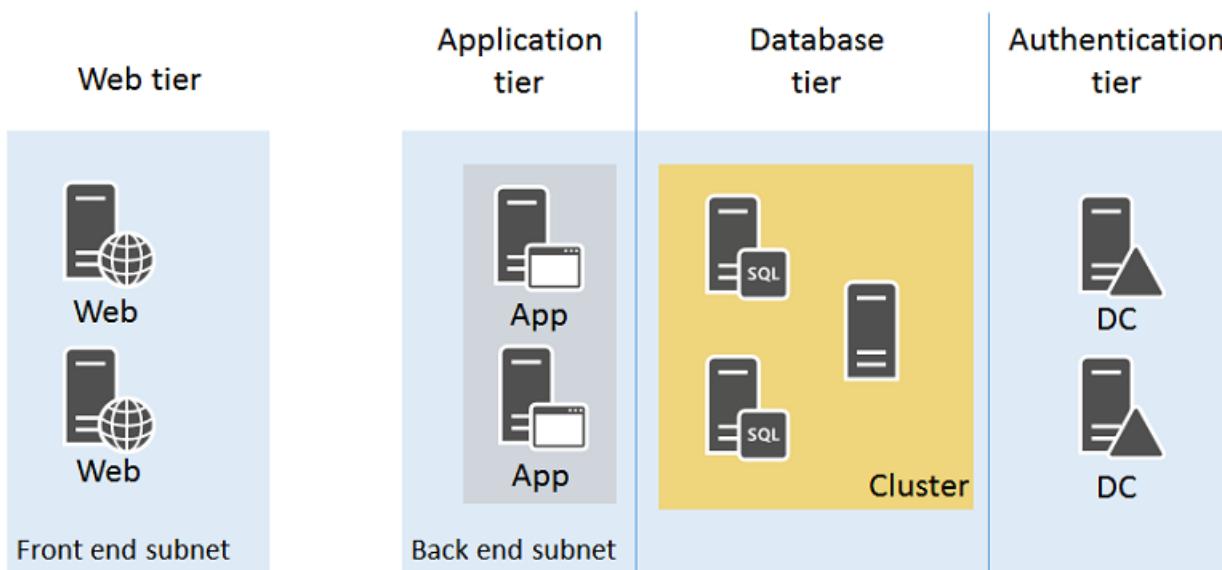
12/15/2017 • 3 min to read • [Edit Online](#)

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

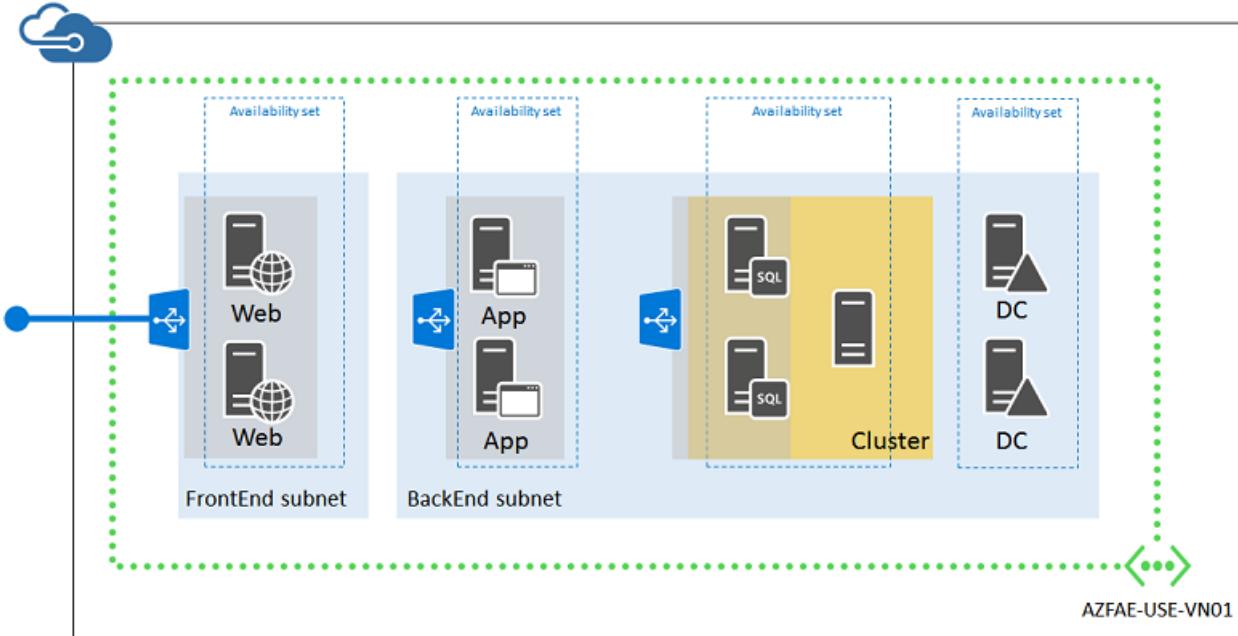
Virtual machines

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

4/9/2018 • 1 min 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 Standard D Family vCPUs. Any time a new VM is deployed the vCPUs for the newly deployed VM must not exceed the vCPU quota for the specific VM size family or the total regional vCPU quota. If either of those quotas are exceeded, then 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 Azure CLI.

Check usage

You can check your quota usage using `az vm list-usage`.

```
az vm list-usage --location "East US"
[
  ...
  {
    "currentValue": 4,
    "limit": 260,
    "name": {
      "localizedValue": "Total Regional vCPUs",
      "value": "cores"
    }
  },
  {
    "currentValue": 4,
    "limit": 10000,
    "name": {
      "localizedValue": "Virtual Machines",
      "value": "virtualMachines"
    }
  },
  {
    "currentValue": 1,
    "limit": 2000,
    "name": {
      "localizedValue": "Virtual Machine Scale Sets",
      "value": "virtualMachineScaleSets"
    }
  },
  {
    "currentValue": 1,
    "limit": 10,
    "name": {
      "localizedValue": "Standard B Family vCPUs",
      "value": "standardBFamily"
    }
  },
]
```

Reserved VM Instances

Reserved VM Instances, which are scoped to a single subscription, will add a new aspect to the vCPU quotas. These values describe the number of instances of the stated size that must be deployable in the subscription. They work as a placeholder in the quota system to ensure that quota is reserved to ensure reserved instances are deployable

in the subscription. For example, if a specific subscription has 10 Standard_D1 reserved instances the usages limit for Standard_D1 Reserved Instances will be 10. This will cause Azure to ensure that there are always at least 10 vCPUs available in the Total Regional vCPUs quota to be used for Standard_D1 instances and there are at least 10 vCPUs available in the Standard D Family vCPU quota to be used for Standard_D1 instances.

If a quota increase is required to 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

3/8/2018 • 10 min 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 2.0](#) 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
```

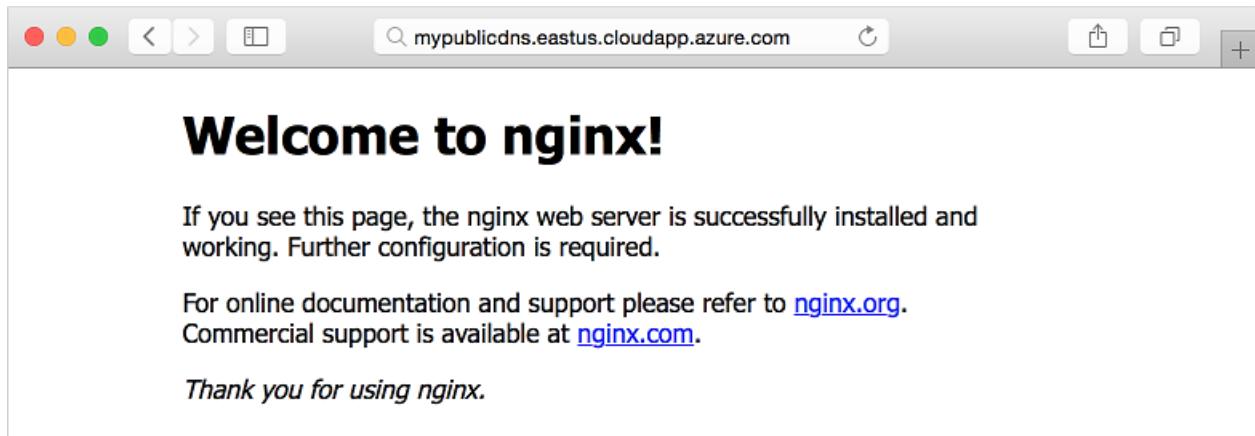
Output:

```
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:  
http://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

2/6/2018 • 1 min to read • [Edit Online](#)

This article shows you how to quickly deploy a Linux virtual machine (VM) with Azure Resource Manager templates and the Azure CLI 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

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 [Create your first Azure Resource Manager template](#). To view the JSON syntax for resources types, see [Define resources in Azure Resource Manager templates](#).

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. The following example creates a resource group named *myResourceGroupVM* in the *eastus* region:

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

Create a virtual machine

The following example creates a VM from [this Azure Resource Manager template](#) with `az group deployment create`. Only SSH authentication is allowed. 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](#).

```
az group deployment create --resource-group myResourceGroup \
    --template-uri https://raw.githubusercontent.com/azure/azure-quickstart-templates/master/101-vm-sshkey/azuredeploy.json
```

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.

Connect to virtual machine

To SSH to your VM, obtain the public IP address with `az vm show`:

```
az vm show \
    --resource-group myResourceGroup \
    --name sshvm \
    --show-details \
    --query publicIps \
    --output tsv
```

You can then SSH to your VM as normal. Provide you own public IP address from the preceding command:

```
ssh azureuser@<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 gallery](#).

Create a copy of a Linux VM by using Azure CLI 2.0 and Managed Disks

4/9/2018 • 2 min to read • [Edit Online](#)

This article shows you how to create a copy of your Azure virtual machine (VM) running Linux using the Azure CLI 2.0 and the Azure Resource Manager deployment model. You can also perform these steps with the [Azure CLI 1.0](#).

You can also [upload and create a VM from a VHD](#).

Prerequisites

- Install [Azure CLI 2.0](#)
- Sign in to an Azure account with [az login](#).
- Have an Azure VM to use as the source for your copy.

Step 1: 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
```

Step 2: Copy the source VM

To copy a VM, you create a copy of the underlying virtual hard disk. This process creates a specialized 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 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
```

Step 3: 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 [Step 4: 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**. (The DNS name must be unique, so 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 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
```

Step 4: Create a VM

You can now 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](#).

How to encrypt virtual disks on a Linux VM

3/8/2018 • 10 min 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 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

Quick commands

If you need to quickly accomplish the task, the following section details the base commands to encrypt virtual disks on your VM. More detailed information and context for each step can be found the rest of the document, [starting here](#).

You need the latest [Azure CLI 2.0](#) 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*, *myKey*, and *myVM*.

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

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=myuniquekeyvaultname  
az keyvault create \  
  --name $keyvault_name \  
  --resource-group myResourceGroup \  
  --location eastus \  
  --enabled-for-disk-encryption True
```

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 service principal using Azure Active Directory with [az ad sp create-for-rbac](#). The service principal handles the authentication and exchange of cryptographic keys from Key Vault. The following example reads in the values for the service principal ID and password for use in later commands:

```
read sp_id sp_password <<< $(az ad sp create-for-rbac --query [appId,password] -o tsv)
```

The password is only output when you create the service principal. If desired, view and record the password (
`echo $sp_password`). You can list your service principals with [az ad sp list](#) and view additional information about a specific service principal with [az ad sp show](#).

Set permissions on your Key Vault with [az keyvault set-policy](#). In the following example, the service principal ID is supplied from the preceding command:

```
az keyvault set-policy --name $keyvault_name --spn $sp_id \
    --key-permissions wrapKey \
    --secret-permissions set
```

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 a *CentOS 7.2n* image:

```
az vm create \
    --resource-group myResourceGroup \
    --name myVM \
    --image OpenLogic:CentOS:7.2n:7.2.20160629 \
    --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 your VM with [az vm encryption enable](#). The following example uses the *\$sp_id* and *\$sp_password* variables from the preceding [ad sp create-for-rbac](#) command:

```
az vm encryption enable \
    --resource-group myResourceGroup \
    --name myVM \
    --aad-client-id $sp_id \
    --aad-client-secret $sp_password \
    --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 myResourceGroup --name myVM
```

The status shows **EncryptionInProgress**. Wait until the status for the OS disk reports **VMRestartPending**, then restart your VM with [az vm restart](#):

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

The disk encryption process is finalized during the boot process, so wait a few minutes before checking the status of encryption again with [az vm encryption show](#):

```
az vm encryption show --resource-group myResourceGroup --name myVM
```

The status should now report both the OS disk and data disk as **Encrypted**.

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. An Azure Active Directory service principal provides a secure mechanism for issuing these cryptographic keys as VMs are powered on and off.

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. To read the cryptographic key from the Azure Key Vault, create an Azure Active Directory service principal with the appropriate permissions.
4. Issue the command to encrypt your virtual disks, specifying the Azure Active Directory service principal and appropriate cryptographic key to be used.
5. The Azure Active Directory service principal requests the required cryptographic key from Azure Key Vault.
6. The virtual disks are encrypted using the provided cryptographic key.

Encryption process

Disk encryption relies on the following additional components:

- **Azure Key Vault** - used to safeguard cryptographic keys and secrets used for the disk encryption/decryption process.
 - If one exists, you can use an existing Azure Key Vault. You do not have to dedicate a Key Vault to encrypting disks.
 - To separate administrative boundaries and key visibility, you can create a dedicated Key Vault.
- **Azure Active Directory** - handles the secure exchanging of required cryptographic keys and authentication for requested actions.
 - You can typically use an existing Azure Active Directory instance for housing your application.
 - The service principal provides a secure mechanism to request and be issued the appropriate cryptographic keys. You are not developing an actual application that integrates with Azure Active Directory.

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 Azure Key Vault and keys

You need the latest [Azure CLI 2.0](#) 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*, *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  
az group create --name myResourceGroup --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=myuniquekeyvaultname  
az keyvault create \  
  --name $keyvault_name \  
  --resource-group myResourceGroup \  
  --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 the Azure Active Directory service principal

When virtual disks are encrypted or decrypted, you specify an account to handle the authentication and exchanging of cryptographic keys from Key Vault. This account, an Azure Active Directory service principal, allows the Azure platform to request the appropriate cryptographic keys on behalf of the VM. A default Azure Active Directory instance is available in your subscription, though many organizations have dedicated Azure Active Directory directories.

Create a service principal using Azure Active Directory with [az ad sp create-for-rbac](#). The following example reads in the values for the service principal Id and password for use in later commands:

```
read sp_id sp_password <<< $(az ad sp create-for-rbac --query [appId,password] -o tsv)
```

The password is only displayed when you create the service principal. If desired, view and record the password (`echo $sp_password`). You can list your service principals with [az ad sp list](#) and view additional information about a

specific service principal with [az ad sp show](#).

To successfully encrypt or decrypt virtual disks, permissions on the cryptographic key stored in Key Vault must be set to permit the Azure Active Directory service principal to read the keys. Set permissions on your Key Vault with [az keyvault set-policy](#). In the following example, the service principal ID is supplied from the preceding command:

```
az keyvault set-policy --name $keyvault_name --spn $sp_id \
--key-permissions wrapKey \
--secret-permissions set
```

Create virtual machine

Create a VM to encrypt 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 a *CentOS 7.2n* image:

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--image OpenLogic:CentOS:7.2n:7.2.20160629 \
--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 virtual machine

To encrypt the virtual disks, you bring together all the previous components:

1. Specify the Azure Active Directory service principal and password.
2. Specify the Key Vault to store the metadata for your encrypted disks.
3. Specify the cryptographic keys to use for the actual encryption and decryption.
4. Specify whether you want to encrypt the OS disk, the data disks, or all.

Encrypt your VM with [az vm encryption enable](#). The following example uses the *\$sp_id* and *\$sp_password* variables from the preceding [az ad sp create-for-rbac](#) command:

```
az vm encryption enable \
--resource-group myResourceGroup \
--name myVM \
--aad-client-id $sp_id \
--aad-client-secret $sp_password \
--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 myResourceGroup --name myVM
```

The output is similar to the following truncated example:

```
[  
  "dataDisk": "EncryptionInProgress",  
  "osDisk": "EncryptionInProgress",  
]
```

Wait until the status for the OS disk reports **VMRestartPending**, then restart your VM with [az vm restart](#):

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

The disk encryption process is finalized during the boot process, so wait a few minutes before checking the status of encryption again with [az vm encryption show](#):

```
az vm encryption show --resource-group myResourceGroup --name myVM
```

The status should now report both the OS disk and data disk as **Encrypted**.

Add additional data disks

Once you have encrypted your data disks, you can later add additional virtual disks to your VM and also encrypt them. For example, lets add a second virtual disk to your VM as follows:

```
az vm disk attach-new --resource-group myResourceGroup --vm-name myVM --size-in-gb 5
```

Rerun the command to encrypt the virtual disks as follows:

```
az vm encryption enable \  
  --resource-group myResourceGroup \  
  --name myVM \  
  --aad-client-id $sp_id \  
  --aad-client-secret $sp_password \  
  --disk-encryption-keyvault $keyvault_name \  
  --key-encryption-key myKey \  
  --volume-type all
```

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](#).

Get started with Role-Based Access Control in the Azure portal

4/11/2018 • 3 min to read • [Edit Online](#)

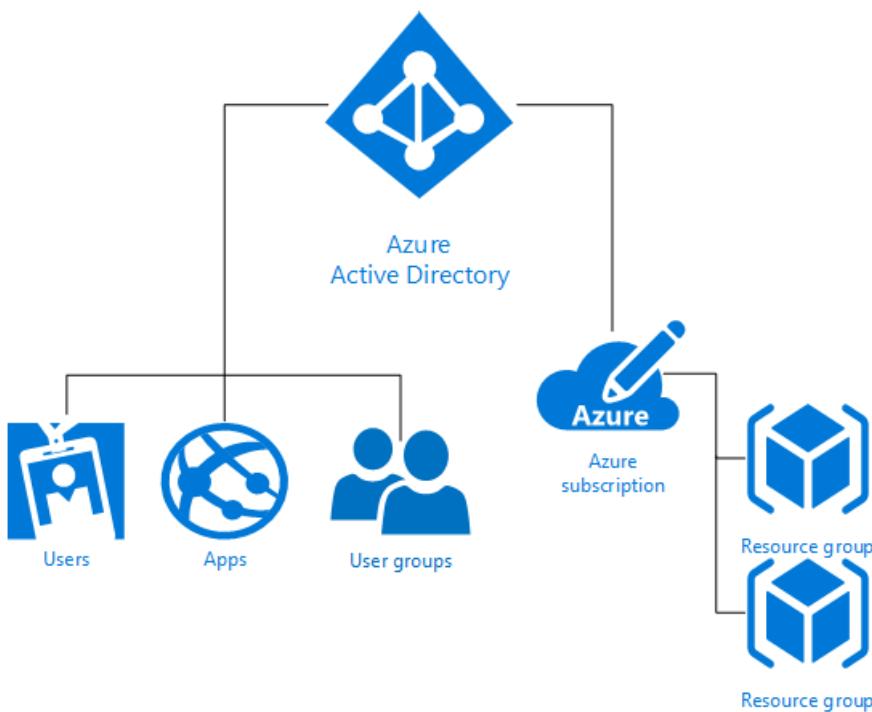
Security-oriented companies should focus on giving employees the exact permissions they need. Too many permissions can expose an account to attackers. Too few permissions means that employees can't get their work done efficiently. Azure Role-Based Access Control (RBAC) helps address this problem by offering fine-grained access management for Azure.

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. For example, use RBAC to let one employee manage virtual machines in a subscription, while another can manage SQL databases within the same subscription.

Basics of access management in Azure

Each Azure subscription is associated with one Azure Active Directory (AD) directory. Users, groups, and applications from that directory can manage resources in the Azure subscription. Assign these access rights using the Azure portal, Azure command-line tools, and Azure Management APIs.

Grant access by assigning the appropriate RBAC role to users, groups, and applications at a certain scope. The scope of a role assignment can be a subscription, a resource group, or a single resource. A role assigned at a parent scope also grants access to the children contained within it. For example, a user with access to a resource group can manage all the resources it contains, like websites, virtual machines, and subnets.



The RBAC role that you assign dictates what resources the user, group, or application can manage within that scope.

Built-in roles

Azure RBAC has three basic roles that 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.

The rest of the RBAC roles in Azure allow management of specific Azure resources. For example, the Virtual Machine Contributor role allows the user to create and manage virtual machines. It does not give them access to the virtual network or the subnet that the virtual machine connects to.

[RBAC built-in roles](#) lists the roles available in Azure. It specifies the operations and scope that each built-in role grants to users. If you're looking to define your own roles for even more control, see how to build [Custom roles in Azure RBAC](#).

Resource hierarchy and access inheritance

- Each **subscription** in Azure belongs to only one directory. (But each directory can have more than one subscription.)
- Each **resource group** belongs to only one subscription.
- Each **resource** belongs to only one resource group.

Access that you grant at parent scopes is inherited at child scopes. For example:

- You assign the Reader role to an Azure AD group at the subscription scope. The members of that group can view every resource group and resource in the subscription.
- 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.

Azure RBAC vs. classic subscription administrators

[Classic subscription administrators and co-admins](#) have full access to the Azure subscription. They can manage resources using the [Azure portal](#), Azure Resource Manager APIs, and the classic deployment model APIs. In the RBAC model, classic administrators are assigned the Owner role at the subscription scope.

Only the Azure portal and the new Azure Resource Manager APIs support Azure RBAC. Users and applications that are assigned RBAC roles cannot use the Azure classic deployment model APIs.

Authorization for management vs. data operations

Azure RBAC only supports management operations of the Azure resources in the Azure portal and Azure Resource Manager APIs. It cannot authorize all data level operations for Azure resources. For example, you can authorize someone to manage Storage Accounts, but not to the blobs or tables within a Storage Account. Similarly, a SQL database can be managed, but not the tables within it.

Next Steps

- Get started with [Role-Based Access Control in the Azure portal](#).
- See the [RBAC built-in roles](#)
- Define your own [Custom roles in Azure RBAC](#)

Apply policies to Linux VMs with Azure Resource Manager

4/9/2018 • 2 min 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 2.0

4/9/2018 • 1 min 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 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

To perform these steps, you need the latest [Azure CLI 2.0](#) 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

4/18/2018 • 3 min 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 log 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.

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 that are by default created in the `~/.ssh` directory. You can specify a different location and an additional passphrase (a password to access the private key file) when prompted. If an SSH key pair exists in the current location, those files are overwritten.

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

If you use the [Azure CLI 2.0](#) 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, specify your SSH public key when creating the VM using the Azure portal, CLI, 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 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
```

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 additional whitespace. 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**).

The public key that you place on your Linux VM in Azure is by default stored in `~/.ssh/id_rsa.pub`, unless you changed the location when you created the keys. If you use the [Azure CLI 2.0](#) 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.

SSH to your VM

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 login 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`.)

VMs created using SSH keys are by default configured with passwords disabled, to make brute-forced guessing attempts vastly more expensive and therefore difficult.

Next steps

This article described creating a simple SSH key pair for quick usage.

- If you need more assistance to work with your SSH key pair, see [Detailed steps to create and manage SSH key pairs](#).
- If you have problems with SSH connections to an Azure VM, see [Troubleshoot SSH connections to an Azure Linux VM](#).

How to use SSH keys with Windows on Azure

4/18/2018 • 5 min to read • [Edit Online](#)

This article introduces 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 logins over unsecured connections. It 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* is what you present to your Linux VM when you make an SSH connection, to verify your identity. Protect this private key. Do not share it.

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) 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. Windows 10 versions that include the [Windows Subsystem for Linux](#) allow you to run and access utilities such as an SSH client natively within a Bash shell.

If you wish to use something other than Bash for Windows, common Windows SSH clients you can install locally are included in the following packages:

- [PuTTY](#)
- [Git For Windows](#)
- [MobaXterm](#)
- [Cygwin](#)

Another option is to 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

This section shows you two options to create an SSH key pair on Windows.

Create SSH keys with ssh-keygen

If you can run a command shell such as Bash for Windows or GitBash (or Bash in 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 current location, those files are overwritten.

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

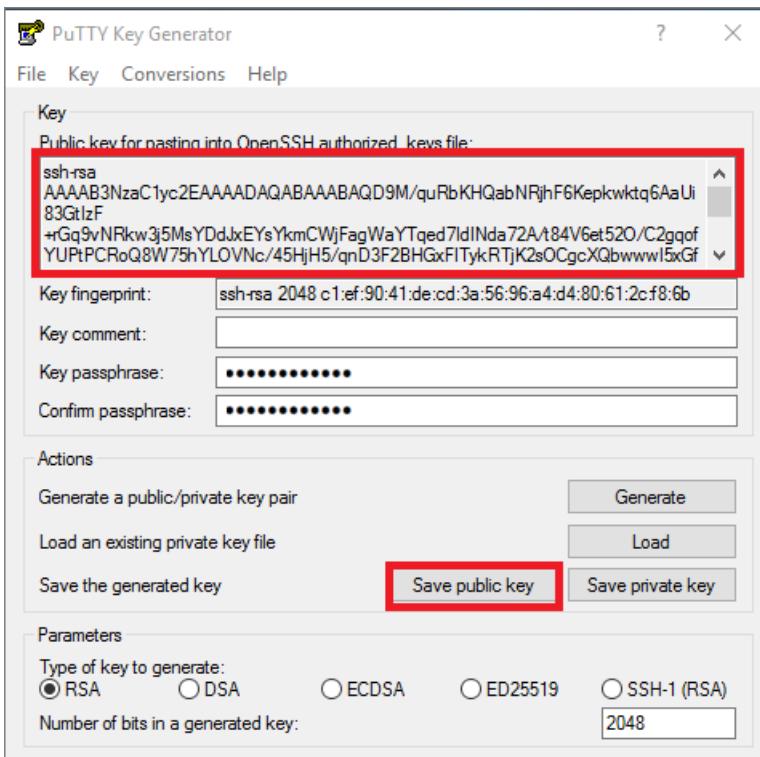
For more background and information, see the [quick](#) or [detailed](#) steps to create the keys with `ssh-keygen`.

Create SSH keys with PuTTYgen

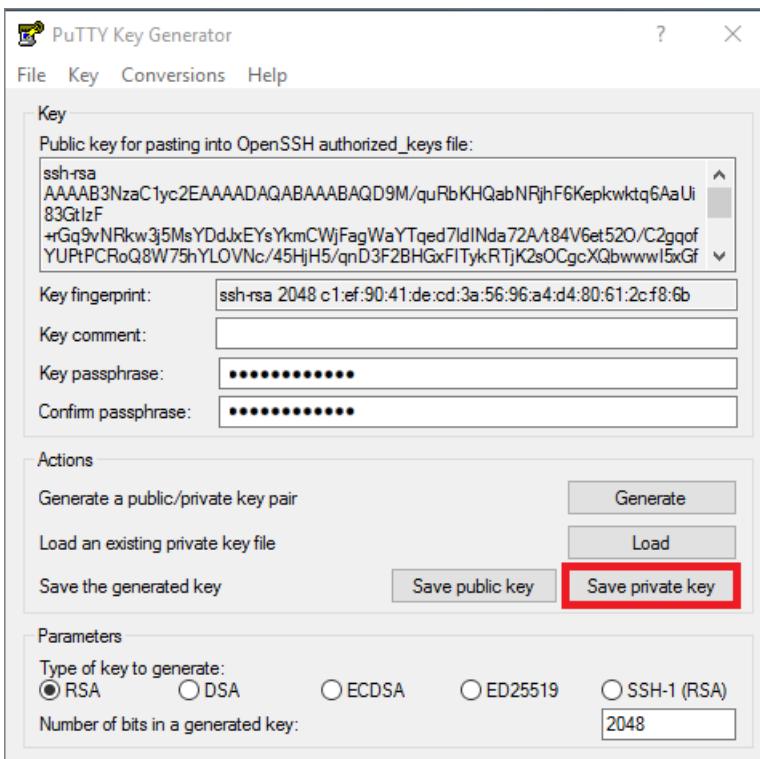
If you prefer to use a GUI-based tool to create SSH keys, you can use the PuTTYgen key generator, included with the [PuTTY download package](#).

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. Mouse over the blank area to generate some 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 SSH key. Without a passphrase, if someone obtains your private key, they can log 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 copy and paste this one-line format public key into the Azure portal or an Azure Resource Manager template when you create a Linux VM. You can also click **Save public key** to save a copy to your computer:



6. Optionally, to save the private key in PuTTY private key format (.ppk file), click **Save private key**. You need the .ppk file of you want to use PuTTY later to make an SSH connection to the VM.

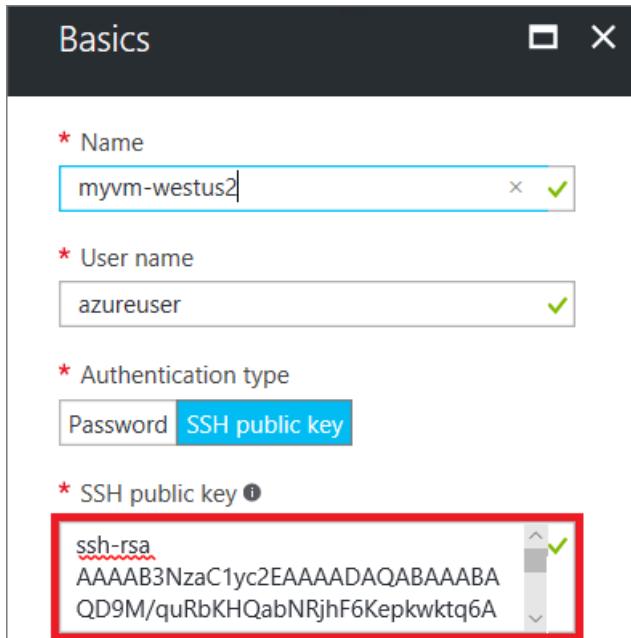


If you want to save the private key in the OpenSSH format, the private key format used by many SSH clients, click **Conversions > Export OpenSSH key**.

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 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 `~/.ssh/authorized_keys` 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 you use 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):

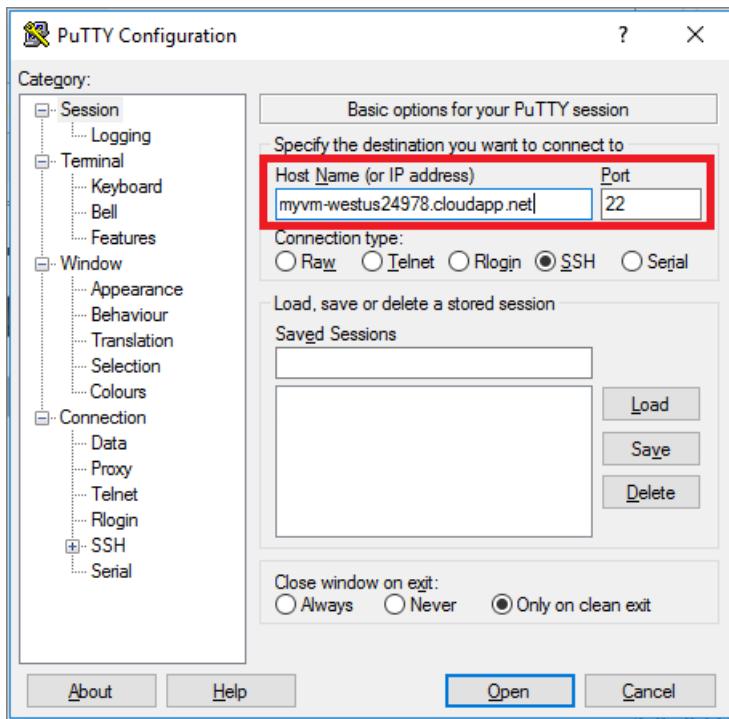
```
ssh azureuser@myvm.westus.cloudapp.azure.com
```

If you configured a passphrase when you created your key pair, enter the passphrase when prompted during the login process.

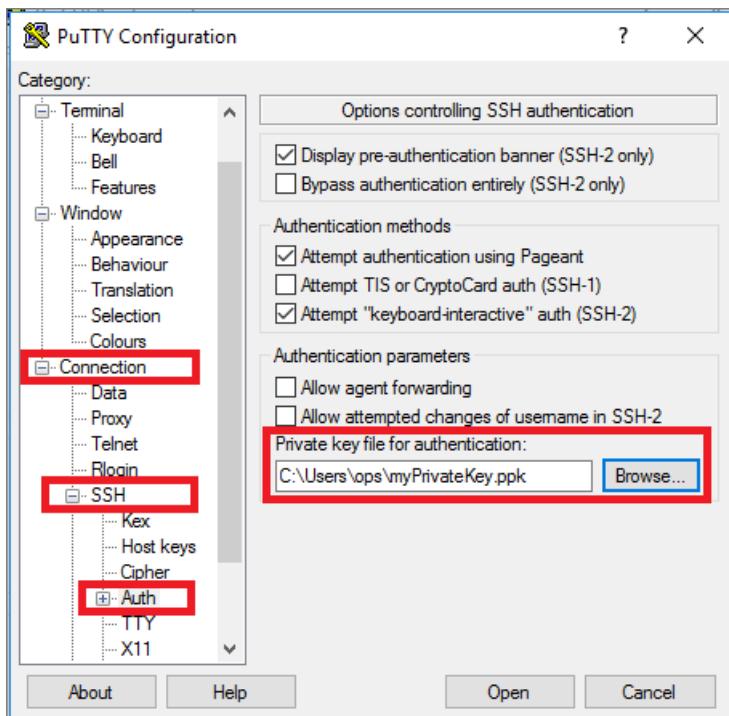
Connect with PuTTY

If you installed the [PuTTY download package](#) and previously generated a PuTTY private key (.ppk file), you can connect to the Linux VM with PuTTY.

1. Start PuTTY.
 2. Fill in the host name or IP address of your VM from the Azure portal:



3. Before selecting **Open**, click **Connection > SSH > Auth** tab. 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 trouble 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

4/18/2018 • 10 min 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 log 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 logins over unsecured connections. It 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* is what you present to your Linux VM when you make an SSH connection, to verify your identity. Protect this private key. Do not share it.

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) 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 log 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 login connection. In an Azure Linux VM that uses SSH keys for authentication, Azure configures the SSHD server to not allow password logins, 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 log 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 2.0](#) 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 2.0](#) 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+eShZzaXnt75gUmT1I2f75zFn2h1AIDGKwf4g12KWC  
Zxy81TniUOTjUsVlwPymXUXxESL/UfJKfbdstBhT0dy5EG9rYWA0K43SJmwPhH288poLfXXXXXXG+/ilsXXXXXgRLiJ2W19MzXHp8z3Lxw7r9w  
x3HaV1P4XiFv9U4hGcp8RMI1MP1nNesFl0BpG4pV2bJRTXNxY4l6F8WZ3C4kuF8Xx0o08mxaTpVZ3T1841a1tmNTZCcPkXuMrBjYSJbA8np0  
XAXNwiivyoe3X2KMXXXXdXXXXXXXXXXXXXXX/ 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 linebreaks. 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 login 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`.)

Use ssh-agent to store your private key passphrase

To avoid typing your private key file passphrase with every SSH login, 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 log 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 log 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 log 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 login are better secured than VMs created with the default login 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](#)

Overview of the features in Azure Backup

4/18/2018 • 20 min to read • [Edit Online](#)

Azure Backup is the Azure-based service you can use to back up (or protect) and restore your data in the Microsoft cloud. Azure Backup replaces your existing on-premises or off-site backup solution with a cloud-based solution that is reliable, secure, and cost-competitive. Azure Backup offers multiple components that you download and deploy on the appropriate computer, server, or in the cloud. The component, or agent, that you deploy depends on what you want to protect. All Azure Backup components (no matter whether you're protecting data on-premises or in the cloud) can be used to back up data to a Recovery Services vault in Azure. See the [Azure Backup components table](#) (later in this article) for information about which component to use to protect specific data, applications, or workloads.

[Watch a video overview of Azure Backup](#)

Why use Azure Backup?

Traditional backup solutions have evolved to treat the cloud as an endpoint, or static storage destination, similar to disks or tape. While this approach is simple, it is limited and doesn't take full advantage of an underlying cloud platform, which translates to an expensive, inefficient solution. Other solutions are expensive because you end up paying for the wrong type of storage, or storage that you don't need. Other solutions are often inefficient because they don't offer you the type or amount of storage you need, or administrative tasks require too much time. In contrast, Azure Backup delivers these key benefits:

Automatic storage management - Hybrid environments often require heterogeneous storage - some on-premises and some in the cloud. With Azure Backup, there is no cost for using on-premises storage devices. Azure Backup automatically allocates and manages backup storage, and it uses a pay-as-you-use model. Pay-as-you-use means that you only pay for the storage that you consume. For more information, see the [Azure pricing article](#).

Unlimited scaling - Azure Backup uses the underlying power and unlimited scale of the Azure cloud to deliver high-availability - with no maintenance or monitoring overhead. You can set up alerts to provide information about events, but you don't need to worry about high-availability for your data in the cloud.

Multiple storage options - An aspect of high-availability is storage replication. Azure Backup offers two types of replication: [locally redundant storage](#) and [geo-redundant storage](#). Choose the backup storage option based on need:

- Locally redundant storage (LRS) replicates your data three times (it creates three copies of your data) in a storage scale unit in a datacenter. All copies of the data exist within the same region. LRS is a low-cost option for protecting your data from local hardware failures.
- Geo-redundant storage (GRS) is the default and recommended replication option. GRS replicates your data to a secondary region (hundreds of miles away from the primary location of the source data). GRS costs more than LRS, but GRS provides a higher level of durability for your data, even if there is a regional outage.

Unlimited data transfer - Azure Backup does not limit the amount of inbound or outbound data you transfer. Azure Backup also does not charge for the data that is transferred. However, if you use the Azure Import/Export service to import large amounts of data, there is a cost associated with inbound data. For more information about this cost, see [Offline-backup workflow in Azure Backup](#). Outbound data refers to data transferred from a Recovery Services vault during a restore operation.

Data encryption - Data encryption allows for secure transmission and storage of your data in the public cloud.

You store the encryption passphrase locally, and it is never transmitted or stored in Azure. If it is necessary to restore any of the data, only you have encryption passphrase, or key.

Application-consistent backup - An application-consistent backup means a recovery point has all required data to restore the backup copy. Azure Backup provides application-consistent backups, which ensure additional fixes are not required to restore the data. Restoring application-consistent data reduces the restoration time, allowing you to quickly return to a running state.

Long-term retention - You can use Recovery Services vaults for short-term and long-term data retention. Azure doesn't limit the length of time data can remain in a Recovery Services vault. You can keep data in a vault for as long as you like. Azure Backup has a limit of 9999 recovery points per protected instance. See the [Backup and retention](#) section in this article for an explanation of how this limit may impact your backup needs.

Which Azure Backup components should I use?

Use the following table for information about what you can protect with each Azure Backup component.

COMPONENT	BENEFITS	LIMITS	WHAT IS PROTECTED?	WHERE ARE BACKUPS STORED?
Azure Backup (MARS) agent	<ul style="list-style-type: none">• Back up files and folders on physical or virtual Windows OS (VMs can be on-premises or in Azure)• No separate backup server required.	<ul style="list-style-type: none">• Backup 3x per day• Not application aware; file, folder, and volume-level restore only,• No support for Linux.	<ul style="list-style-type: none">• Files,• Folders,• System State	Recovery Services vault
System Center DPM	<ul style="list-style-type: none">• Application-aware snapshots (VSS)• Full flexibility for when to take backups• Recovery granularity (all)• Can use Recovery Services vault• Linux support on Hyper-V and VMware VMs• Back up and restore VMware VMs using DPM 2012 R2	Cannot back up Oracle workload.	<ul style="list-style-type: none">• Files,• Folders,• Volumes,• VMs,• Applications,• Workloads• System State	<ul style="list-style-type: none">• Recovery Services vault,• Locally attached disk,• Tape (on-premises only)
Azure Backup Server	<ul style="list-style-type: none">• Application-aware snapshots (VSS)• Full flexibility for when to take backups• Recovery granularity (all)• Can use Recovery Services vault• Linux support on Hyper-V and VMware VMs• Back up and restore VMware VMs• Does not require a System Center license	<ul style="list-style-type: none">• Cannot back up Oracle workload.• Always requires live Azure subscription• No support for tape backup	<ul style="list-style-type: none">• Files,• Folders,• Volumes,• VMs,• Applications,• Workloads,• System State	<ul style="list-style-type: none">• Recovery Services vault,• Locally attached disk

COMPONENT	BENEFITS	LIMITS	WHAT IS PROTECTED?	WHERE ARE BACKUPS STORED?
Azure IaaS VM Backup	<ul style="list-style-type: none"> Application-aware snapshots (VSS) Native backups for Windows/Linux No specific agent installation required Fabric-level backup with no backup infrastructure needed 	<ul style="list-style-type: none"> Back up VMs once-a-day Restore VMs only at disk level Cannot back up on-premises 	<ul style="list-style-type: none"> VMs, All disks (using PowerShell) 	Recovery Services vault

What are the deployment scenarios for each component?

COMPONENT	CAN BE DEPLOYED IN AZURE?	CAN BE DEPLOYED ON-PREMISES?	TARGET STORAGE SUPPORTED
Azure Backup (MARS) agent	Yes The Azure Backup agent can be deployed on any Windows Server VM that runs in Azure.	Yes The Backup agent can be deployed on any Windows Server VM or physical machine.	Recovery Services vault
System Center DPM	Yes Learn more about how to protect workloads in Azure by using System Center DPM.	Yes Learn more about how to protect workloads and VMs in your datacenter.	Locally attached disk, Recovery Services vault, tape (on-premises only)
Azure Backup Server	Yes Learn more about how to protect workloads in Azure by using Azure Backup Server.	Yes Learn more about how to protect workloads in Azure by using Azure Backup Server.	Locally attached disk, Recovery Services vault
Azure IaaS VM Backup	Yes Part of Azure fabric Specialized for backup of Azure infrastructure as a service (IaaS) virtual machines .	No Use System Center DPM to back up virtual machines in your datacenter.	Recovery Services vault

Which applications and workloads can be backed up?

The following table provides a matrix of the data and workloads that can be protected using Azure Backup. The Azure Backup solution column has links to the deployment documentation for that solution.

DATA OR WORKLOAD	SOURCE ENVIRONMENT	AZURE BACKUP SOLUTION
------------------	--------------------	-----------------------

DATA OR WORKLOAD	SOURCE ENVIRONMENT	AZURE BACKUP SOLUTION
Files and folders	Windows Server	Azure Backup agent , System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
Files and folders	Windows computer	Azure Backup agent , System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
Hyper-V virtual machine (Windows)	Windows Server	System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
Hyper-V virtual machine (Linux)	Windows Server	System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
VMware virtual machine	Windows Server	System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
Microsoft SQL Server	Windows Server	System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
Microsoft SharePoint	Windows Server	System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
Microsoft Exchange	Windows Server	System Center DPM (+ the Azure Backup agent), Azure Backup Server (includes the Azure Backup agent)
Azure IaaS VMs (Windows)	running in Azure	Azure Backup (VM extension)
Azure IaaS VMs (Linux)	running in Azure	Azure Backup (VM extension)

Linux support

The following table shows the Azure Backup components that have support for Linux.

COMPONENT	LINUX (AZURE ENDORSED) SUPPORT
Azure Backup (MARS) agent	No (Only Windows based agent)
System Center DPM	<ul style="list-style-type: none">File-consistent backup of Linux Guest VMs on Hyper-V and VMWareVM restore of Hyper-V and VMWare Linux Guest VMs <p><i>File-consistent backup not available for Azure VM</i></p>
Azure Backup Server	<ul style="list-style-type: none">File-consistent backup of Linux Guest VMs on Hyper-V and VMWareVM restore of Hyper-V and VMWare Linux Guest VMs <p><i>File-consistent backup not available for Azure VM</i></p>
Azure IaaS VM Backup	<p>Application-consistent backup using pre-script and post-script framework</p> <p>Granular file recovery</p> <p>Restore all VM disks</p> <p>VM restore</p>

Using Premium Storage VMs with Azure Backup

Azure Backup protects Premium Storage VMs. Azure Premium Storage is solid-state drive (SSD)-based storage designed to support I/O-intensive workloads. Premium Storage is attractive for virtual machine (VM) workloads. For more information about Premium Storage, see the article, [Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads](#).

Back up Premium Storage VMs

While backing up Premium Storage VMs, the Backup service creates a temporary staging location, named "AzureBackup-", in the Premium Storage account. The size of the staging location is equal to the size of the recovery point snapshot. Be sure the Premium Storage account has adequate free space to accommodate the temporary staging location. For more information, see the article, [premium storage limitations](#). Once the backup job finishes, the staging location is deleted. The price of storage used for the staging location is consistent with all [Premium storage pricing](#).

NOTE

Do not modify or edit the staging location.

Restore Premium Storage VMs

You can restore Premium Storage VMs to either Premium Storage or to Standard Storage. Restoring a Premium Storage VM recovery point back to Premium Storage is the typical process. However, it can be cost effective to restore a Premium Storage VM recovery point to Standard Storage if you need a subset of files from the VM.

Using managed disk VMs with Azure Backup

Azure Backup protects managed disk VMs. Managed disks free you from managing storage accounts of virtual machines and greatly simplify VM provisioning.

Back up managed disk VMs

Backing up VMs on managed disks is no different than backing up Resource Manager VMs. In the Azure portal, you can configure the backup job directly from the Virtual Machine view or from the Recovery Services vault view. You can back up VMs on managed disks through RestorePoint collections built on top of managed disks. Azure Backup also supports backing up managed disk VMs encrypted using Azure Disk encryption(ADE).

Restore managed disk VMs

Azure Backup allows you to restore a complete VM with managed disks, or restore managed disks to a storage account. Azure manages the managed disks during the restore process. You (the customer) manage the storage account created as part of the restore process. When restoring managed encrypted VMs, the VM's keys and secrets should exist in the key vault prior to starting the restore operation.

What are the features of each Backup component?

The following sections provide tables that summarize the availability or support of various features in each Azure Backup component. See the information following each table for additional support or details.

Storage

FEATURE	AZURE BACKUP AGENT	SYSTEM CENTER DPM	AZURE BACKUP SERVER	AZURE IAAS VM BACKUP
Recovery Services vault				
Disk storage				
Tape storage				
Compression (in Recovery Services vault)				
Incremental backup				
Disk deduplication				

Key



= Supported



= Partially Supported

<blank> = Not Supported

The Recovery Services vault is the preferred storage target across all components. System Center DPM and Azure Backup Server also provide the option to have a local disk copy. However, only System Center DPM provides the option to write data to a tape storage device.

Compression

Backups are compressed to reduce the required storage space. The only component that does not use compression is the VM extension. The VM extension copies all backup data from your storage account to the Recovery Services vault in the same region. No compression is used when transferring the data. Transferring the data without compression slightly inflates the storage used. However, storing the data without compression allows for faster restoration, should you need that recovery point.

Disk Deduplication

You can take advantage of deduplication when you deploy System Center DPM or Azure Backup Server [on a Hyper-V virtual machine](#). Windows Server performs data deduplication (at the host level) on virtual hard disks (VHDs) that are attached to the virtual machine as backup storage.

NOTE

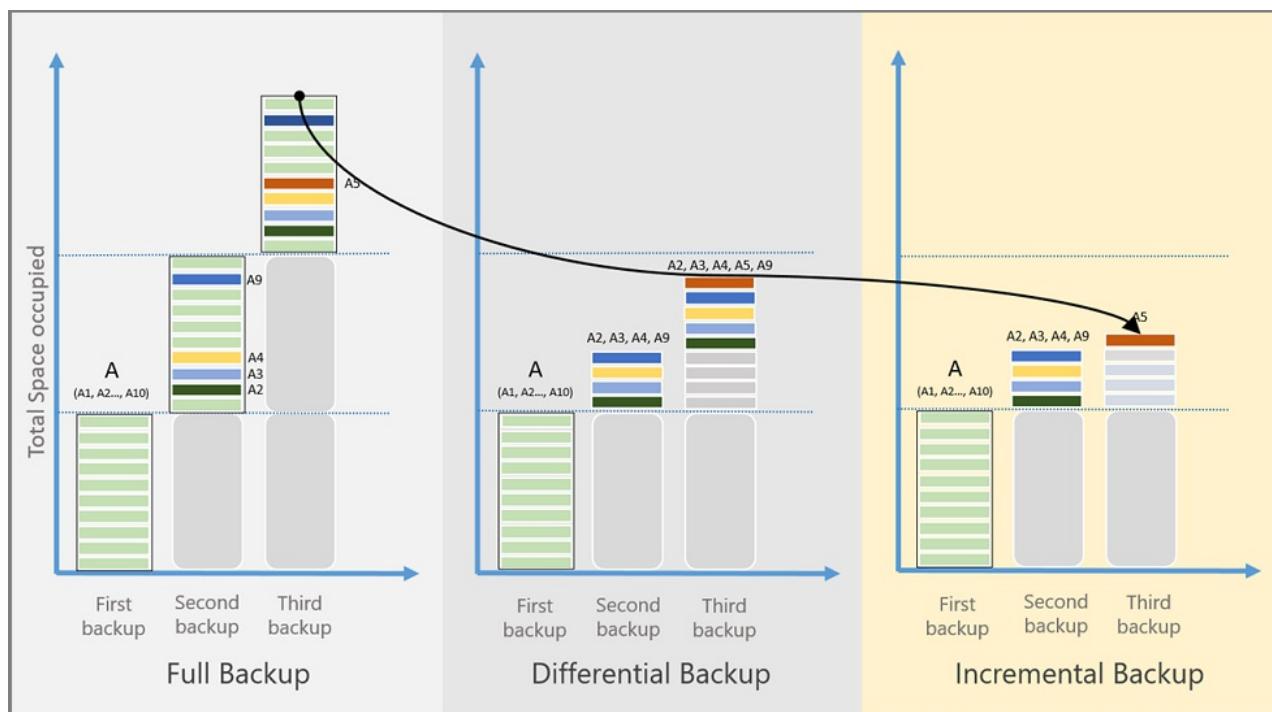
Deduplication is not available in Azure for any Backup component. When System Center DPM and Backup Server are deployed in Azure, the storage disks attached to the VM cannot be deduplicated.

Incremental backup explained

Every Azure Backup component supports incremental backup regardless of the target storage (disk, tape, Recovery Services vault). Incremental backup ensures that backups are storage and time efficient, by transferring only those changes made since the last backup.

Comparing Full, Differential and Incremental backup

Storage consumption, recovery time objective (RTO), and network consumption varies for each type of backup method. To keep the backup total cost of ownership (TCO) down, you need to understand how to choose the best backup solution. The following image compares Full Backup, Differential Backup, and Incremental Backup. In the image, data source A is composed of 10 storage blocks A1-A10, which are backed up monthly. Blocks A2, A3, A4, and A9 change in the first month, and block A5 changes in the next month.



With **Full Backup**, each backup copy contains the entire data source. Full backup consumes a large amount of network bandwidth and storage, each time a backup copy is transferred.

Differential backup stores only the blocks that changed since the initial full backup, which results in a smaller amount of network and storage consumption. Differential backups don't retain redundant copies of unchanged data. However, because the data blocks that remain unchanged between subsequent backups are transferred and stored, differential backups are inefficient. In the second month, changed blocks A2, A3, A4, and A9 are backed up. In the third month, these same blocks are backed up again, along with changed block A5. The changed blocks continue to be backed up until the next full backup happens.

Incremental Backup achieves high storage and network efficiency by storing only the blocks of data that changed since the previous backup. With incremental backup, there is no need to take regular full backups. In the example, after taking the full backup in the first month, blocks A2, A3, A4, and A9 are marked as changed, and transferred to the second month. In the third month, only changed block A5 is marked and transferred. Moving less data saves storage and network resources, which decreases TCO.

Security

FEATURE	AZURE BACKUP AGENT	SYSTEM CENTER DPM	AZURE BACKUP SERVER	AZURE IAAS VM BACKUP
Network security (to Azure)				
Data security (in Azure)				

Key

= Supported

= Partially Supported

<blank> = Not Supported

Network security

All backup traffic from your servers to the Recovery Services vault is encrypted using Advanced Encryption Standard 256. The backup data is sent over a secure HTTPS link. The backup data is also stored in the Recovery Services vault in encrypted form. Only you, the Azure customer, have the passphrase to unlock this data. Microsoft cannot decrypt the backup data at any point.

WARNING

Once you establish the Recovery Services vault, only you have access to the encryption key. Microsoft never maintains a copy of your encryption key, and does not have access to the key. If the key is misplaced, Microsoft cannot recover the backup data.

Data security

Backing up Azure VMs requires setting up encryption *within* the virtual machine. Use BitLocker on Windows virtual machines and **dm-crypt** on Linux virtual machines. Azure Backup does not automatically encrypt backup data that comes through this path.

Network

FEATURE	AZURE BACKUP AGENT	SYSTEM CENTER DPM	AZURE BACKUP SERVER	AZURE IAAS VM BACKUP
Network compression (to backup server)				
Network compression (to Recovery Services vault)				
Network protocol (to backup server)		TCP	TCP	
Network protocol (to Recovery Services vault)	HTTPS	HTTPS	HTTPS	HTTPS

Key

= Supported

<blank> = Not Supported

The VM extension (on the IaaS VM) reads the data directly from the Azure storage account over the storage network, so it is not necessary to compress this traffic.

If you use a System Center DPM server or Azure Backup Server as a secondary backup server, compress the data going from the primary server to the backup server. Compressing data before back up to DPM or Azure Backup Server, saves bandwidth.

Network Throttling

The Azure Backup agent offers network throttling, which allows you to control how network bandwidth is used during data transfer. Throttling can be helpful if you need to back up data during work hours but do not want the backup process to interfere with other internet traffic. Throttling for data transfer applies to back up and restore activities.

Backup and retention

Azure Backup has a limit of 9999 recovery points, also known as backup copies or snapshots, per *protected instance*. A protected instance is a computer, server (physical or virtual), or workload configured to back up data to Azure. For more information, see the section, [What is a protected instance](#). An instance is protected once a backup copy of data has been saved. The backup copy of data is the protection. If the source data was lost or became corrupt, the backup copy could restore the source data. The following table shows the maximum backup frequency for each component. Your backup policy configuration determines how quickly you consume the recovery points. For example, if you create a recovery point each day, then you can retain recovery points for 27 years before you run out. If you take a monthly recovery point, you can retain recovery points for 833 years before you run out. The Backup service does not set an expiration time limit on a recovery point.

	AZURE BACKUP AGENT	SYSTEM CENTER DPM	AZURE BACKUP SERVER	AZURE IAAS VM BACKUP
Backup frequency (to Recovery Services vault)	Three backups per day	Two backups per day	Two backups per day	One backup per day
Backup frequency (to disk)	Not applicable	<ul style="list-style-type: none">• Every 15 minutes for SQL Server• Every hour for other workloads	<ul style="list-style-type: none">• Every 15 minutes for SQL Server• Every hour for other workloads	Not applicable
Retention options	Daily, weekly, monthly, yearly	Daily, weekly, monthly, yearly	Daily, weekly, monthly, yearly	Daily, weekly, monthly, yearly
Maximum recovery points per protected instance	9999	9999	9999	9999
Maximum retention period	Depends on backup frequency	Depends on backup frequency	Depends on backup frequency	Depends on backup frequency
Recovery points on local disk	Not applicable	<ul style="list-style-type: none">• 64 for File Servers,• 448 for Application Servers	<ul style="list-style-type: none">• 64 for File Servers,• 448 for Application Servers	Not applicable
Recovery points on tape	Not applicable	Unlimited	Not applicable	Not applicable

What is a protected instance

A protected instance is a generic reference to a Windows computer, a server (physical or virtual), or SQL database that has been configured to back up to Azure. An instance is protected once you configure a backup policy for the computer, server, or database, and create a backup copy of the data. Subsequent copies of the backup data for that protected instance (which are called recovery points), increase the amount of storage consumed. You can create up to 9999 recovery points for a protected instance. If you delete a recovery point from storage, it does not count against the 9999 recovery point total. Some common examples of protected instances are virtual machines,

application servers, databases, and personal computers running the Windows operating system. For example:

- A virtual machine running the Hyper-V or Azure IaaS hypervisor fabric. The guest operating systems for the virtual machine can be Windows Server or Linux.
- An application server: The application server can be a physical or virtual machine running Windows Server and workloads with data that needs to be backed up. Common workloads are Microsoft SQL Server, Microsoft Exchange server, Microsoft SharePoint server, and the File Server role on Windows Server. To back up these workloads you need System Center Data Protection Manager (DPM) or Azure Backup Server.
- A personal computer, workstation, or laptop running the Windows operating system.

What is a Recovery Services vault?

A Recovery Services vault is an online storage entity in Azure used to hold data such as backup copies, recovery points, and backup policies. You can use Recovery Services vaults to hold backup data for Azure services and on-premises servers and workstations. Recovery Services vaults make it easy to organize your backup data, while minimizing management overhead. Within each Azure subscription, you can create up to 500 Recovery Services vaults per Azure region. When considering where to store your data, not all regions are the same. See [Geo-redundant storage](#) for information about region pairings and additional storage considerations.

Backup vaults, which were based on Azure Service Manager, were the first version of the vault. Recovery Services vaults, which add the Azure Resource Manager model features, are the second version of the vault. See the [Recovery Services vault overview article](#) for a full description of the feature differences. You can no longer create Backup vaults, and all existing Backup vaults have been upgraded to Recovery Services vaults. You can use the Azure portal to manage the vaults that were upgraded to Recovery Services vaults.

How does Azure Backup differ from Azure Site Recovery?

Azure Backup and Azure Site Recovery are related in that both services back up data and can restore that data. However, these services serve different purposes in providing business continuity and disaster recovery in your business. Use Azure Backup to protect and restore data at a more granular level. For example, if a presentation on a laptop became corrupted, you would use Azure Backup to restore the presentation. If you wanted to replicate the configuration and data on a VM across another datacenter, use Azure Site Recovery.

Azure Backup protects data on-premises and in the cloud. Azure Site Recovery coordinates virtual-machine and physical-server replication, failover, and failback. Both services are important because your disaster recovery solution needs to keep your data safe and recoverable (Backup) *and* keep your workloads available (Site Recovery) when outages occur.

The following concepts can help you make important decisions around backup and disaster recovery.

CONCEPT	DETAILS	BACKUP	DISASTER RECOVERY (DR)
Recovery point objective (RPO)	The amount of acceptable data loss if a recovery needs to be done.	Backup solutions have wide variability in their acceptable RPO. Virtual machine backups usually have an RPO of one day, while database backups have RPOs as low as 15 minutes.	Disaster recovery solutions have low RPOs. The DR copy can be behind by a few seconds or a few minutes.

CONCEPT	DETAILS	BACKUP	DISASTER RECOVERY (DR)
Recovery time objective (RTO)	The amount of time that it takes to complete a recovery or restore.	Because of the larger RPO, the amount of data that a backup solution needs to process is typically much higher, which leads to longer RTOs. For example, it can take days to restore data from tapes, depending on the time it takes to transport the tape from an off-site location.	Disaster recovery solutions have smaller RTOs because they are more in sync with the source. Fewer changes need to be processed.
Retention	How long data needs to be stored	For scenarios that require operational recovery (data corruption, inadvertent file deletion, OS failure), backup data is typically retained for 30 days or less. From a compliance standpoint, data might need to be stored for months or even years. Backup data is ideally suited for archiving in such cases.	Disaster recovery needs only operational recovery data, which typically takes a few hours or up to a day. Because of the fine-grained data capture used in DR solutions, using DR data for long-term retention is not recommended.

Next steps

Use one of the following tutorials for detailed, step-by-step, instructions for protecting data on Windows Server, or protecting a virtual machine (VM) in Azure:

- [Back up Files and Folders](#)
- [Backup Azure Virtual Machines](#)

For details about protecting other workloads, try one of these articles:

- [Back up your Windows Server](#)
- [Back up application workloads](#)
- [Backup Azure IaaS VMs](#)

Back up a virtual machine in Azure with the CLI

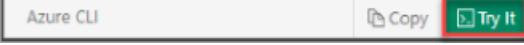
2/14/2018 • 4 min to read • [Edit Online](#)

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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 Azure CLI 2.0](#).

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.

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
```

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

2/16/2018 • 6 min 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

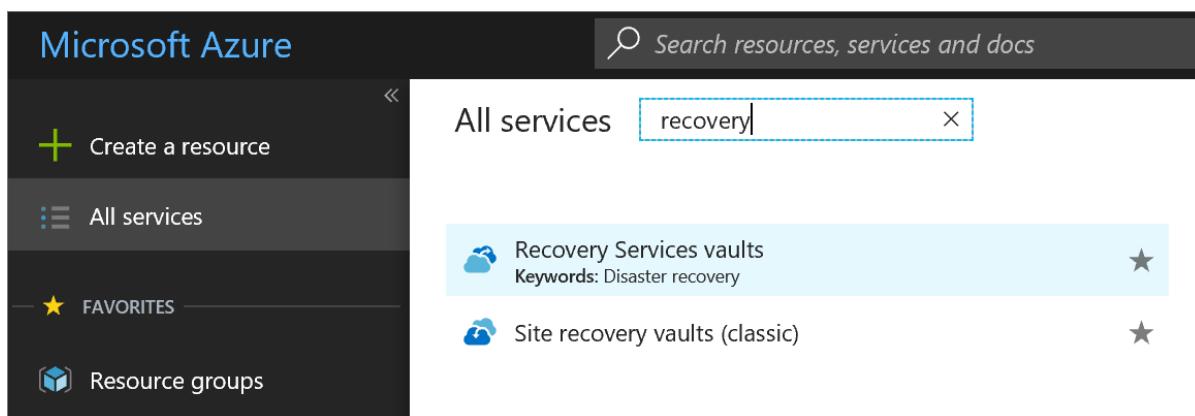
Log in to the Azure portal

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

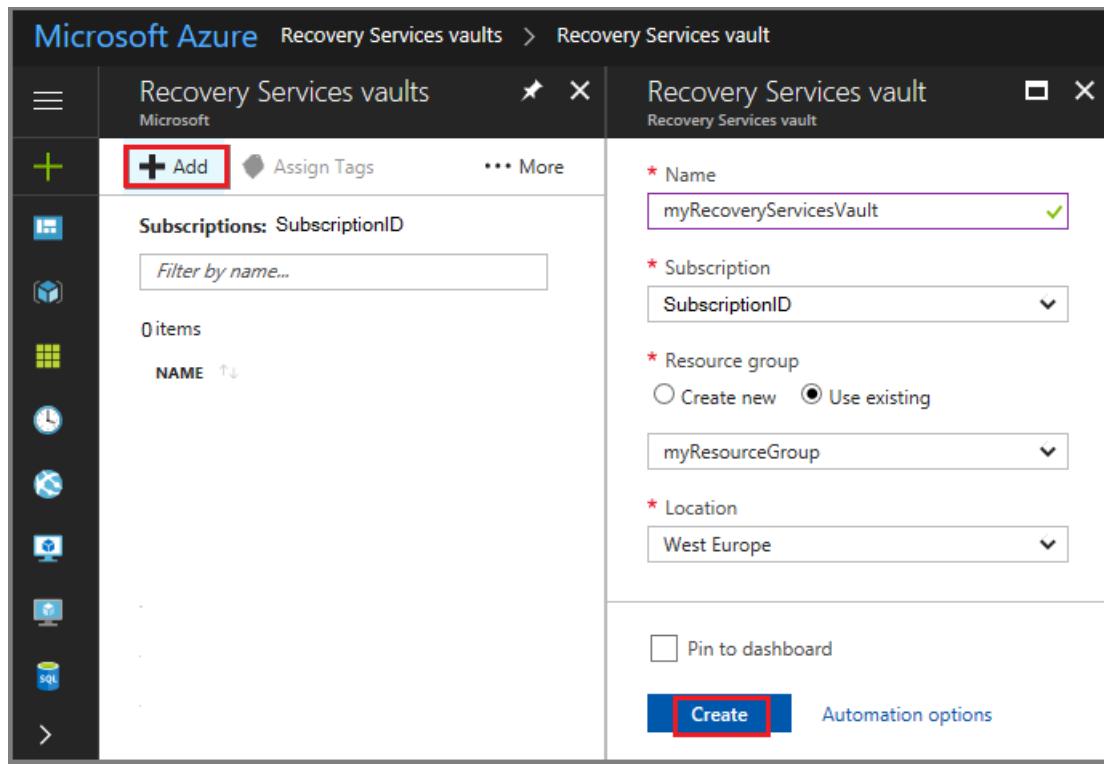
Create a Recovery Services vault

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.

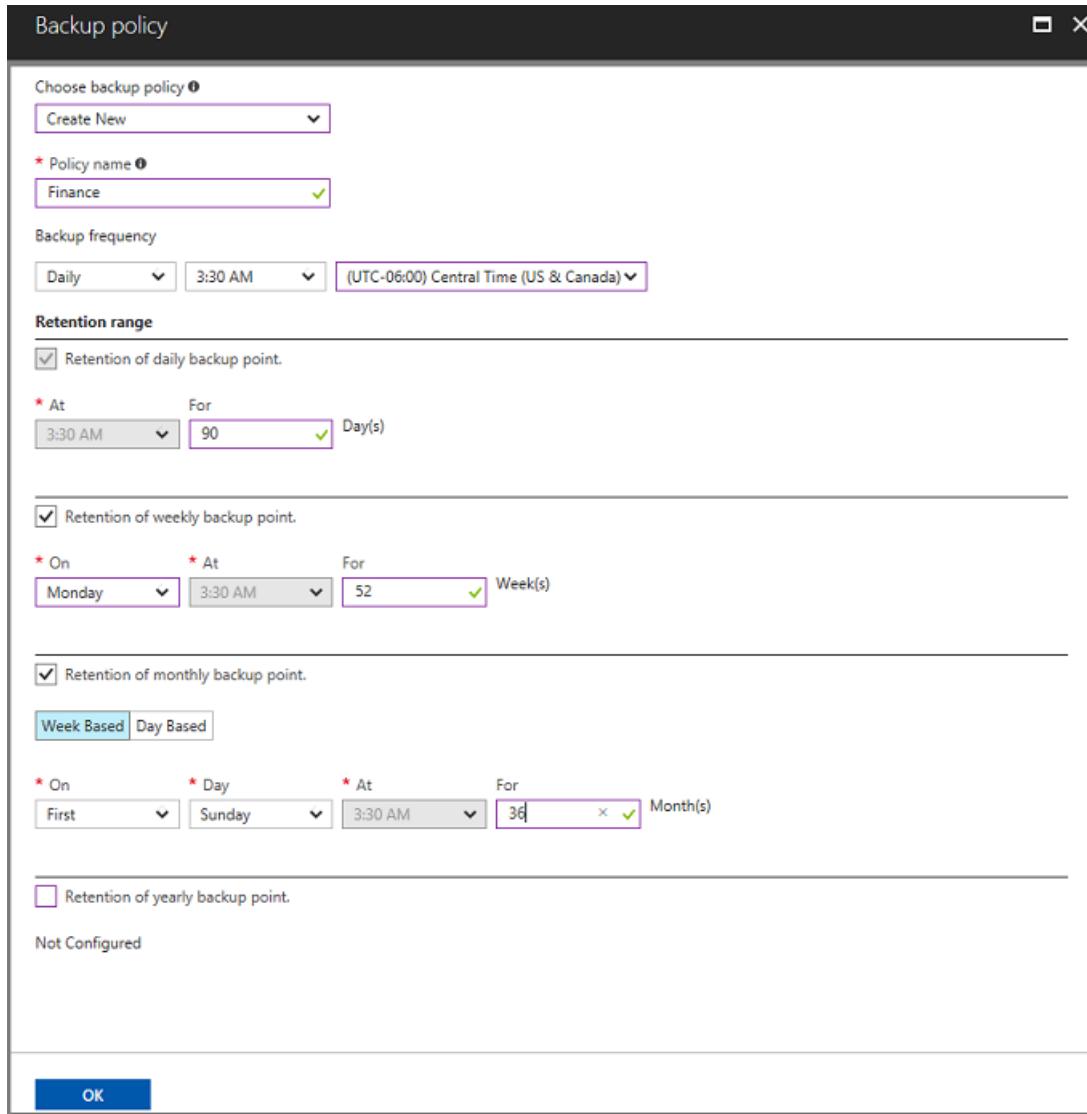
2. On the vault dashboard menu, click **Backup** to open the Backup menu.
3. On the Backup Goal menu, in the **Where is your workload running?** drop-down menu, choose *Azure*. From the **What do you want to backup?** drop-down, choose *Virtual machine*, and click **Backup**.

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:

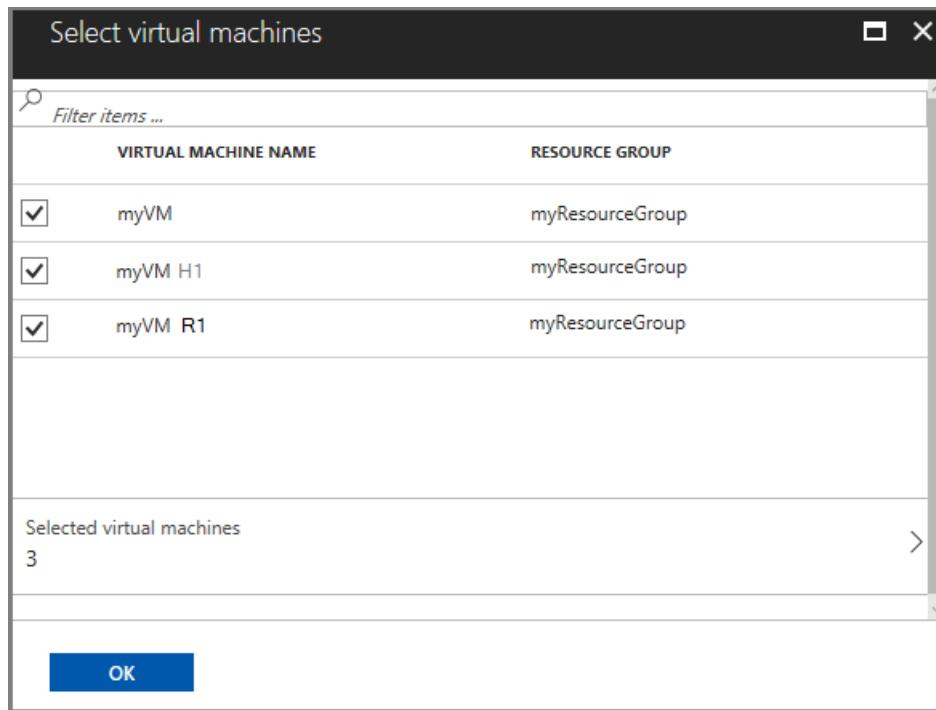
- For **Backup frequency** set the timezone for *Central Time*. Since the sports complex is in Texas, the owner wants the timing to be local. Leave the backup frequency set to Daily at 3:30AM.
- For **Retention of daily backup point**, set the period to 90 days.
- For **Retention of weekly backup point**, 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.

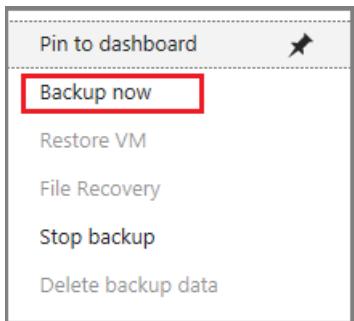
NAME	RESOURCE GROUP	BACKUP PRE-CHECK	LAST BACKUP STATUS	LATEST RESTORE POINT
myVM	myResourceGroup	Passed	Warning(Initial backu...)	Context menu
buntu	rasquill-security	Passed	Warning(Initial backu...)	...
ops	rhelpfiles	Passed	Warning(Initial backu...)	...

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...)		...

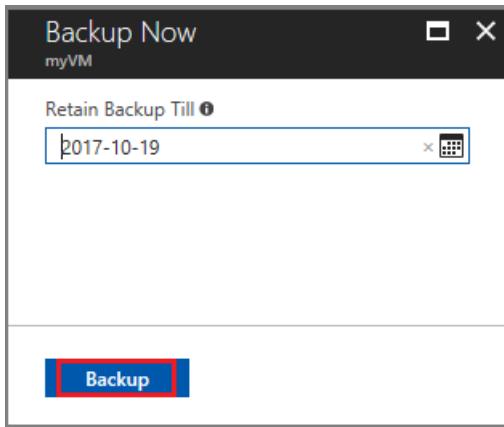
3. On the **Backup Items** list, click the ellipses ... to open the Context menu.

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.

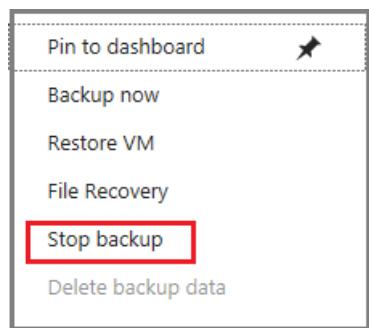
BACKUP MANAGEMENT TYPE	BACKUP ITEM COUNT
Azure Virtual Machine	3
Azure Backup Agent	0
Azure Backup Server	0

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 displays 'Essentials' information: Resource group (myResourceGroup), Backup items (0), Status (Active), and Backup management servers (0). A prominent red box highlights the 'Delete' button in the top right navigation bar.

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

4/17/2018 • 5 min to read • [Edit Online](#)

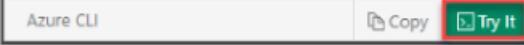
Azure Backup creates recovery points that are stored in geo-redundant recovery vaults. When you restore from a recovery point, you can restore the whole VM or individual files. This article explains how to restore a complete VM using CLI. In this tutorial you learn how to:

- 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 Azure CLI 2.0](#).

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

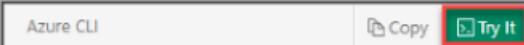
2/14/2018 • 6 min to read • [Edit Online](#)

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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 Azure CLI 2.0](#).

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.

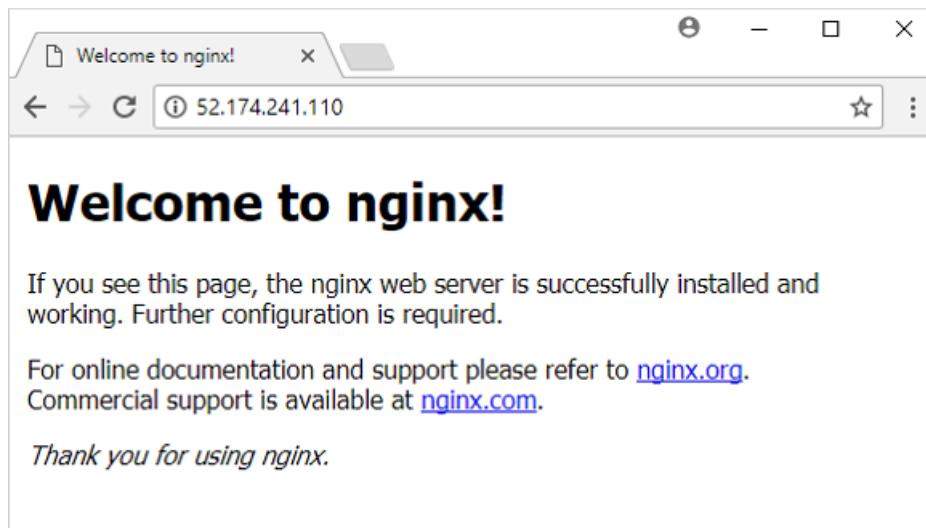
Delete a file from a VM

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.



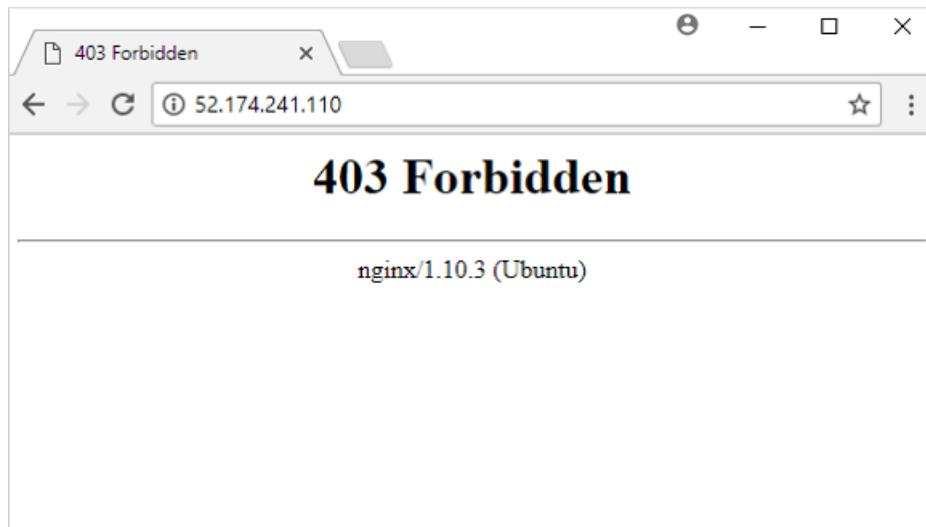
3. Connect to your VM with SSH. Replace `publicIpAddress` with the public IP address that you obtained in a previous command:

```
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 *****

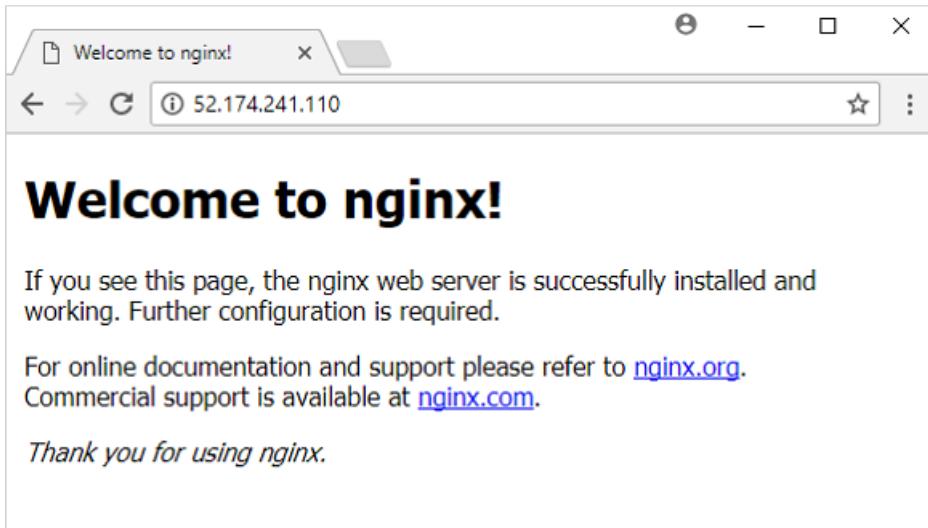
Sr.No. | Disk | Volume | MountPath
1) | /dev/sdc | /dev/sdc1 | /home/azureuser/myVM-20170919213536/Volume1

***** 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](#)

Prepay for Virtual Machines with Reserved VM Instances

5/2/2018 • 2 min to read • [Edit Online](#)

Prepay for virtual machines and save money with Reserved Virtual Machine Instances. For more information, see [Reserved Virtual Machine Instances offering](#).

You can buy Reserved Virtual Machine Instances in the [Azure portal](#). To buy a Reserved Virtual Machine Instance:

- You must be in an Owner role for at least one Enterprise or Pay-As-You-Go subscription.
- For Enterprise subscriptions, reservation purchases must be enabled in the [EA portal](#).
- For Cloud Solution Provider (CSP) program only the admin agents or sales agents can purchase the reservations.

Buy a Reserved Virtual Machine Instance

1. Log 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 number: MS-AZR-0017P) or Pay-As-You-Go (offer number: MS-AZR-0003P). 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.
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 (except dev/test 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.

FIELD	DESCRIPTION
Location	The Azure region that's covered by the reservation.
VM Size	The size of the VM instances.
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_DS1_v2 VMs in East US, then you would specify quantity as 10 to maximize the benefit for all running machines.

5. You can view the cost of the reservation when you select **Calculate cost**.

COSTS

[Calculate cost](#)

Cost per VM	270	USD
Total VMs	10	Standard_DS1_v2
Reservation cost*	2,700	USD
Estimated savings* 53%		
Payment will be processed using the payment instrument of type 'Enrollment' on file for the Microsoft Azure Enterprise subscription.		
<small>*Additional taxes may apply. Estimated savings are calculated based on the current on-demand rate for Virtual Machines in the selected subscription.</small>		

6. Select **Purchase**.

7. Select **View this Reservation** to see the status of your purchase.

Your Reservation ProdDS1VMReservation has been submitted.

Thank you for purchasing an Azure Reservation. The order has been submitted and the Reservation term will begin as soon as payment has been processed.

[View this Reservation \(ID: f39d1c73-af74-4308-9083-fbf5efab37d1\)](#)

Total cost	2700 USD
Payment subscription	Microsoft Azure Enterprise
Scope	Shared
Location	East US
VM size	Standard_DS1_v2
Term	One year
Quantity	10

Next steps

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.

To learn how to manage a reservation, see [Manage Azure Reserved Virtual Machine Instances](#).

To learn more about Reserved Virtual Machine Instances, see the following articles.

- [Save money on virtual machines with Reserved Virtual Machine Instances](#)
- [Understand how the Reserved Virtual Machine Instance discount is applied](#)
- [Understand Reserved Instance usage for your Pay-As-You-Go subscription](#)
- [Understand Reserved Instance usage for your Enterprise enrollment](#)
- [Windows software costs not included with Reserved Instances](#)
- [Reserved Instances in Partner Center Cloud Solution Provider \(CSP\) program](#)

Understanding Azure virtual machine usage

12/6/2017 • 7 min 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>"" {"myDepartment":"RD","myUser":"myName"}"</pre>
Additional Info	<p>Service-specific metadata. For VMs, we populate the following 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 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 values that are populated in this field based on the image you have deployed are the following:

- 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 region 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 2.0 commands for managing Azure resources

4/9/2018 • 1 min to read • [Edit Online](#)

The Azure CLI 2.0 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 2.0](#). 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.

Move a Linux VM to another subscription or resource group

4/9/2018 • 3 min to read • [Edit Online](#)

This article walks you through how to move a Linux 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 Managed Disks at this time.

New resource IDs are created as part of the move. Once the VM has been moved, you 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 using the 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/mystorageacountname

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.

If you are asked to confirm that you want to move the specified resource. Type **Y** to confirm that you want to move the resources.

Use the portal to move a VM to a different subscription

You can move a VM and it's associated resources to a different subscription using the portal.

1. Open the [Azure portal](#).
2. Click **Browse > Virtual machines** and select the VM you would like to move from the list.
3. At the top of the page for the VM, select the → **Move** button 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 type a name to have a new resource group created.
7. When you are done, select that you understand that new resource IDs are created and those need to be used with the VM once it is moved, then click **OK**.

Use the portal to move a VM to another resource group

You can move a VM and it's associated resources to another resource group using the portal.

1. Open the [Azure portal](#).
2. Click **Browse > Virtual machines** and select the VM you would like to move from the list.
3. At the top of the page for the VM, select the → **Move** button and then select **Move to another resource group**. 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 type a name to have a new resource group created.
6. When you are done, select that you understand that new resource IDs are created and those need to be used with the VM once it is moved, then click **OK**.

Next steps

You can move many different types of resources between resource groups and subscriptions. For more information, see [Move resources to new resource group or subscription](#).

Resize a Linux virtual machine using CLI 2.0

4/25/2018 • 1 min 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 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

Resize a VM

To resize a VM, you need the latest [Azure CLI 2.0](#) 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

4/26/2018 • 1 min 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 2.0](#).

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](#).

How to tag a Linux virtual machine in Azure

4/11/2018 • 2 min 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

 Deploy to Azure

 Visualize

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) ⓘ

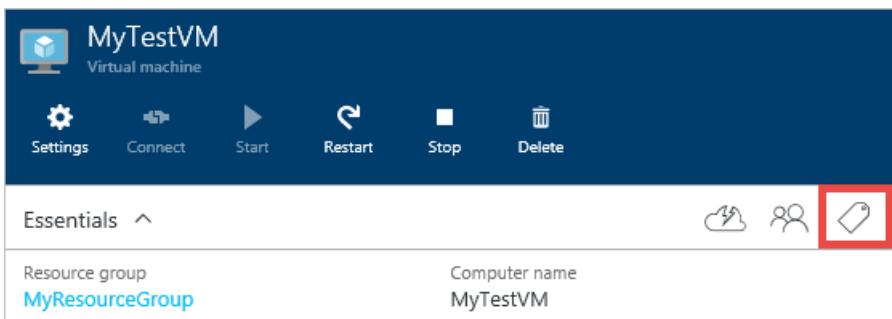
CREATEDBY (string) ⓘ

Click **Create** to deploy this template with your tag values.

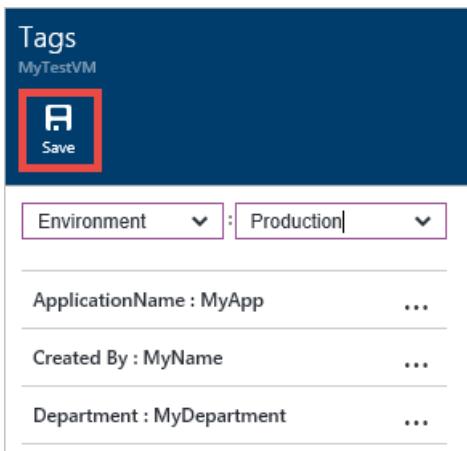
Tagging through the Portal

After creating your resources with tags, you can view, add, and delete tags in the portal.

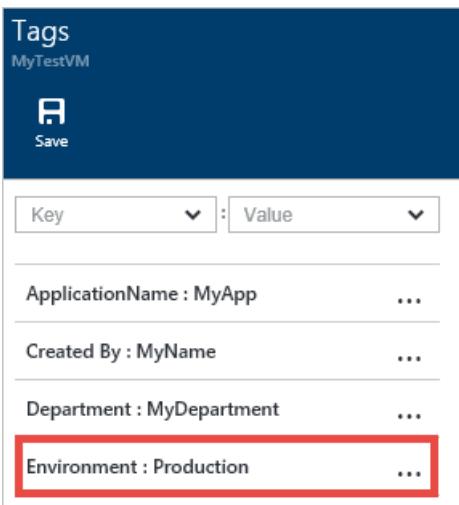
Select the tags icon to view your tags:



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 2.0](#) installed and logged in to an Azure account using [az login](#).

You can also perform these steps with the [Azure CLI 1.0](#).

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

[!\[\]\(202b16b37f320aa23b70e6ea74fc0e6f_img.jpg\) Download usage details](#)

- [!\[\]\(b0874c0b5e5987fd16e1f0a71afb1e70_img.jpg\) Contact Microsoft Support](#)
- [!\[\]\(6047c10a2893f0ea3cfc6ce4870d482a_img.jpg\) Edit subscription details](#)
- [!\[\]\(0649536581d2f7d0ac2a8afdc4ded45f_img.jpg\) Change subscription address](#)
- [!\[\]\(c29903ce58b92f739f372586f1864ea8_img.jpg\) Partner Information](#)
- [!\[\]\(315c70dfd87efbb88658f53df78e5b92_img.jpg\) 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	▼
7/24/2014 - 8/23/2014		Version 2 - Preview	
		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](#).

4 min to read •

Install and configure Remote Desktop to connect to a Linux VM in Azure

2/27/2018 • 4 min 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 2.0](#)
- 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*:

```
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
```

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

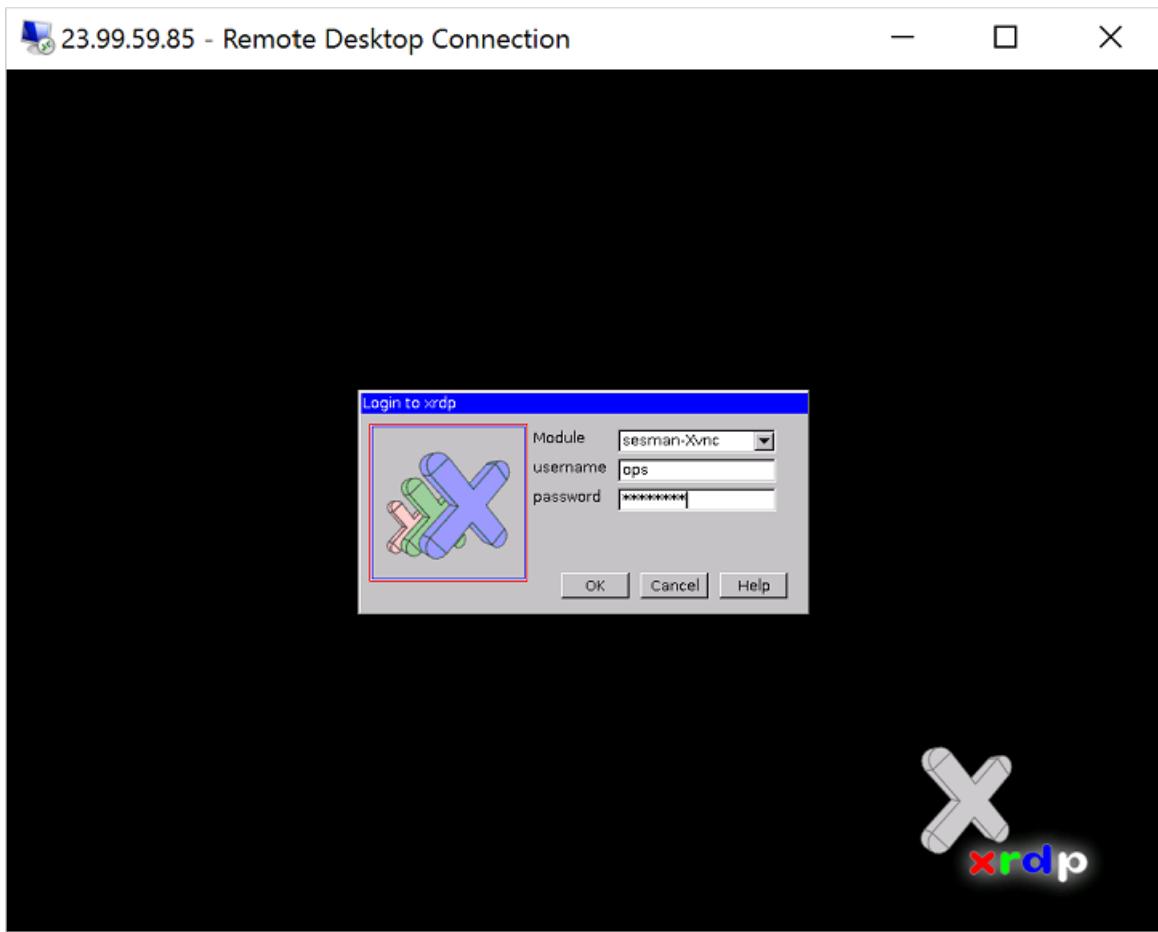
To allow Remote Desktop traffic to reach your Linux VM, a network security group rule needs to be created that allows TCP on port 3389 to reach your VM. For more information about network security group rules, see [What is a Network Security Group?](#) You can also [use the Azure portal to create a network security group rule](#).

The following example creates a network security group rule with [az vm open-port](#) on port 3389.

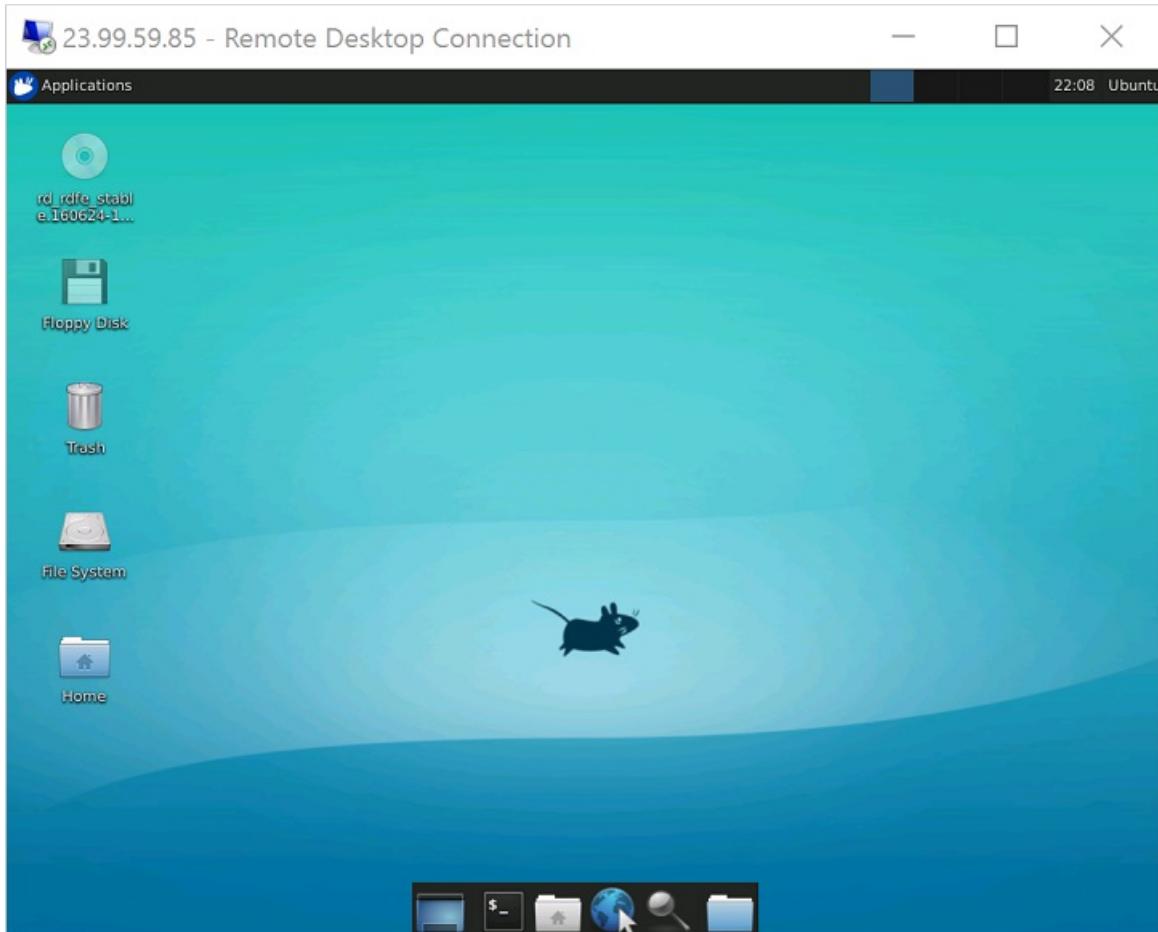
```
az vm open-port --resource-group myResourceGroup --name myVM --port 3389
```

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:



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 RedHat Linux VM to an Azure Active Directory Domain Service

4/9/2018 • 1 min to read • [Edit Online](#)

This article shows you how to join a Red Hat Enterprise Linux (RHEL) 7 virtual machine to an Azure Active Directory Domain Services (AADDS) managed domain. The requirements are:

- [an Azure account](#)
- [SSH public and private key files](#)
- [an Azure Active Directory Domain Services DC](#)

Quick Commands

Replace any examples with your own settings.

Switch the azure-cli to classic deployment mode

```
azure config mode asm
```

Search for a RHEL version and image

```
azure vm image list | grep "Red Hat"
```

Create a Redhat Linux VM

```
azure vm create myVM \
-o a879bbefc56a43abb0ce65052aac09f3__RHEL_7_2_Standard_Azure_RHUI-20161026220742 \
-g ahmet \
-p myPassword \
-e 22 \
-t "~/.ssh/id_rsa.pub" \
-z "Small" \
-l "West US"
```

SSH to the VM

```
ssh -i ~/.ssh/id_rsa ahmet@myVM
```

Update YUM packages

```
sudo yum update
```

Install packages needed

```
sudo yum -y install realmd sssd krb5-workstation krb5-libs
```

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.

Discover the AAD Domain Services managed domain

```
sudo realm discover mydomain.com
```

Initialize kerberos

Ensure that you specify a user who belongs to the 'AAD DC Administrators' group. Only these users can join computers to the managed domain.

```
kinit ahmet@mydomain.com
```

Join the machine to the domain

```
sudo realm join --verbose mydomain.com -U 'ahmet@mydomain.com'
```

Verify the machine is joined to the domain

```
ssh -l ahmet@mydomain.com mydomain.cloudapp.net
```

Next Steps

- [Red Hat Update Infrastructure \(RHUI\) for on-demand Red Hat Enterprise Linux VMs in Azure](#)
- [Set up Key Vault for virtual machines in Azure Resource Manager](#)
- [Deploy and manage virtual machines by using Azure Resource Manager templates and the Azure CLI](#)

Log in to a Linux virtual machine in Azure using Azure Active Directory authentication (Preview)

5/8/2018 • 7 min 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.

- **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 6.9 and CentOS 7.4
RedHat Enterprise Linux	RHEL 7
Ubuntu Server	Ubuntu 14.04 LTS, Ubuntu Server 16.04 and Ubuntu Server 17.10

The following Azure regions are currently supported during the preview of this feature:

- All public 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 log in 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
```

For more information on how to use RBAC to manage access to your Azure subscription resources, see using the [Azure CLI 2.0](#), [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. Specify the public IP address of your VM as output in 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.

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.

Next steps

For more information on Azure Active Directory, see [What is Azure Active Directory](#) and [How to get started with Azure Active Directory](#)

Red Hat Update Infrastructure for on-demand Red Hat Enterprise Linux VMs in Azure

4/9/2018 • 5 min 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.

Important information about Azure RHUI

- Azure RHUI currently supports only the latest minor release in each RHEL family (RHEL6 or RHEL7). To upgrade an RHEL VM instance connected to RHUI to the latest minor version, run `sudo yum update`.

For example, if you provision a VM from an RHEL 7.2 PAYG image and run `sudo yum update`, you end up with an RHEL 7.4 VM (the latest minor version in the RHEL7 family).

To avoid this behavior, you need to build your own image as described in the [Create and upload a Red Hat-based virtual machine for Azure](#) article. Then you need to connect it to a different update infrastructure ([directly to Red Hat content delivery servers](#) or a [Red Hat Satellite server](#)).

- 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 creating and deploying your own (BYOL-type) images. This process is described in [Create and upload a Red Hat-based virtual machine for Azure](#).
- Two classes of RHEL PAYG images in Azure (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.

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  
  
# Azure Germany  
51.5.243.77  
51.4.228.145
```

RHUI Azure 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.

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.

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:

1. Download the public key signature via curl.

```
curl -o RPM-GPG-KEY-microsoft-azure-release https://download.microsoft.com/download/9/D/9/9d945f05-541d-494f-9977-289b3ce8e774/microsoft-sign-public.asc
```

2. Verify the validity of the downloaded key.

```
gpg --list-packets --verbose < RPM-GPG-KEY-microsoft-azure-release
```

3. Check the output, and then verify the `keyid` and the `user ID packet`.

```
Version: GnuPG v1.4.7 (GNU/Linux)
:public key packet:
    version 4, algo 1, created 1446074508, expires 0
    pkey[0]: [2048 bits]
    pkey[1]: [17 bits]
    keyid: EB3E94ADBE1229CF
:user ID packet: "Microsoft (Release signing) <gpgsecurity@microsoft.com>"
:signature packet: algo 1, keyid EB3E94ADBE1229CF
    version 4, created 1446074508, md5len 0, sigclass 0x13
    digest algo 2, begin of digest 1a 9b
    hashed subpkt 2 len 4 (sig created 2015-10-28)
    hashed subpkt 27 len 1 (key flags: 03)
    hashed subpkt 11 len 5 (pref-sym-algos: 9 8 7 3 2)
    hashed subpkt 21 len 3 (pref-hash-algos: 2 8 3)
    hashed subpkt 22 len 2 (pref-zip-algos: 2 1)
    hashed subpkt 30 len 1 (features: 01)
    hashed subpkt 23 len 1 (key server preferences: 80)
    subpkt 16 len 8 (issuer key ID EB3E94ADBE1229CF)
    data: [2047 bits]
```

4. Install the public key.

```
sudo install -o root -g root -m 644 RPM-GPG-KEY-microsoft-azure-release /etc/pki/rpm-gpg
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-microsoft-azure-release
```

5. Download, verify, and install a client RPM Package Manager (RPM).

NOTE

Package versions change. If you manually connect to Azure RHUI, you can find the latest version of the client package for each RHEL family by provisioning the latest image from the gallery.

a. Download.

- For RHEL 6:

```
curl -o azureclient.rpm https://rhui-1.microsoft.com/pulp/repos/microsoft-azure-rhel6/rhui-azure-
rhel6-2.1-32.noarch.rpm
```

- For RHEL 7:

```
curl -o azureclient.rpm https://rhui-1.microsoft.com/pulp/repos/microsoft-azure-rhel7/rhui-azure-
rhel7-2.1-19.noarch.rpm
```

b. Verify.

```
rpm -Kv azureclient.rpm
```

c. Check the output to ensure that the signature of the package is OK.

```
azureclient.rpm:
Header V3 RSA/SHA256 Signature, key ID be1229cf: OK
Header SHA1 digest: OK (927a3b548146c95a3f6c1a5d5ae52258a8859ab3)
V3 RSA/SHA256 Signature, key ID be1229cf: OK
MD5 digest: OK (c04ff605f82f4be8c96020bf5c23b86c)
```

d. Install the RPM.

```
sudo rpm -U azureclient.rpm
```

6. After you finish, verify that you can access Azure RHUI from the VM.

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](#).

Understanding and using the Azure Linux Agent

5/10/2018 • 7 min 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:
<http://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=n
Provisioning.RegenerateSshHostKeyPair=y
Provisioning.SshHostKeyPairType=rsa
Provisioning.MonitorHostName=y
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=n
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](#)

4 min to read •

Handling planned maintenance notifications for Linux virtual machines

4/9/2018 • 12 min 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.
- If maintenance requires a reboot, you get a notice of when the maintenance is planned. In these cases, you are given a time window 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, 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.

NOTE

If you try to start maintenance and the request fails, Azure marks your VM as **skipped**. You will no longer be able to use the Customer Initiated Maintenance option. Your VM will have to be rebooted by Azure during the scheduled maintenance phase.

Should you start maintenance using during the self-service window?

The following guidelines should help you to decide whether you should 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), Service Fabric, and Virtual Machine Scale Sets.

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, but be aware that the order of update domains being impacted does not necessarily happen 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.

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 which 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 which 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, e.g., 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 rgName vmName
```

View VMs scheduled for maintenance in the portal

Once a planned maintenance wave is scheduled, and notifications are sent, 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 which 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 Auto-Scheduled window in this view or by clicking on the VM
Completed	You have successfully initiated and completed maintenance on your VM.
Skipped	You have selected to initiate maintenance with no success. You will not be able to use the self-service maintenance option. Your VM will have to be rebooted by Azure during the scheduled maintenance phase.

Maintenance Pro-Active - shows the time window when you can self-start maintenance on your VMs.

Maintenance Scheduled - shows the time window when Azure will reboot 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 [Monitor subscription activity with the Azure Activity Log](#)

1. Sign in to the [Azure portal](#).
2. In the menu on the left, select **Monitor**.
3. In the **Monitor - Activity log** pane, select **Alerts**.
4. In the **Monitor - Alerts** pane, click **+ Add activity log alert**.
5. Complete the information in the **Add activity log alert** page and make sure you set the following in **Criteria**:
Type: Maintenance **Status**: All (Do not set status to Active or Resolved) **Level**: All

To learn more on how to configure Activity Log Alerts, see [Create activity log alerts](#)

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 rebooted and the maintenance status will be updated to reflect the result within few minutes.

If you missed the window where you can start maintenance, you will still be able to see the window when your VM will be rebooted by Azure.

Classic deployments

If you still have legacy VMs that were deployed using the classic deployment model, you can use CLI 1.0 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 which require us to restart our servers which 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 which 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 Cloud Services (Web/Worker Role), Service Fabric, and Virtual Machine Scale Sets?

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), Service Fabric, and Virtual Machine Scale Sets.

Q: I have received an email about hardware decommissioning, is this the same as planned maintenance?

A: While hardware decommissioning is a planned maintenance event, we have not yet onboarded this use case to the new experience.

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

Q: Maintenance of my availability set takes a long time, and I now see "skipped" status on some of my availability set instances. Why?

A: If you have clicked to update multiple instances in an availability set in short succession, Azure will queue these requests and starts to update only the VMs in one update domain (UD) at a time. However, since there might be a pause between update domains, the update might appear to take longer. If the update queue takes longer than 60 minutes, some instances will show the **skipped** state even if they have been updated successfully. To avoid this incorrect status, update your availability sets by clicking only on instance within one availability set and wait for the update on that VM to complete before clicking on the next VM in a different update domain.

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

4/9/2018 • 3 min to read • [Edit Online](#)

Last document update: April 3, 3:00 PM PST.

The recent 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 other customers running on Azure can't attack your application using these vulnerabilities.

In addition, Azure is expanding the use of [memory preserving maintenance](#) whenever possible, pausing the VM for up to 30 seconds while the host is updated or the VM is moved to an already updated host. Memory preserving maintenance further minimizes customer impact and eliminates the need for reboots. Azure will utilize these methods when making system-wide updates to the host.

NOTE

In late February 2018, Intel Corporation published updated [Microcode Revision Guidance](#) on the status of their microcode releases, which improve stability and mitigate against the recent vulnerabilities disclosed by [Google Project Zero](#). The mitigations put in place by Azure on [January 3, 2018](#) are not affected by Intel's microcode update. Microsoft already put strong mitigations in place to protect Azure customers from other Azure virtual machines.

Intel's microcode addresses variant 2 Spectre ([CVE-2017-5715](#) or branch target injection) to protect against attacks which would only be applicable where you run shared or untrusted workloads inside your VMs on Azure. Our engineers are testing the stability to minimize performance impacts of the microcode, prior to making it available to Azure customers. As very few customers run untrusted workloads within their VMs, most customers will not need to enable this capability once released.

This page will be updated as more information is available.

Keeping your Operating Systems up-to-date

While an OS update is not required to isolate your applications running on Azure from other customers running on Azure, it is always a best practice to keep your OS versions up-to-date. The January 2018 and later [Security Rollups for Windows](#) contain mitigations for these vulnerabilities.

In the following offerings, 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 when available.

Azure Windows Virtual Machines	<p>Verify that you are running a supported antivirus application before you install OS updates. Contact your antivirus software vendor for compatibility information.</p> <p>Install the January security rollup.</p>
Other Azure PaaS Services	<p>There is no action needed for customers using these services. Azure automatically keeps your OS versions up-to-date.</p>

Additional guidance if you are running untrusted code

No additional customer action is needed unless you are running untrusted code. If you allow code that you do not trust (for example, you allow one of your customers to upload a binary or code-snippet that you then execute in the cloud within your application), then the following additional steps should be taken.

Windows

If you are using Windows and hosting untrusted code, you should also enable a Windows feature called Kernel Virtual Address (KVA) Shadowing which provides additional protection against speculative execution side-channel vulnerabilities (specifically for variant 3 Meltdown, [CVE-2017-5754](#), or rogue data cache load). This feature is turned off by default and may impact performance if enabled. Follow [Windows Server KB4072698](#) instructions for Enabling Protections on the Server. If you are running Azure Cloud Services, verify that you are running WA-GUEST-OS-5.15_201801-01 or WA-GUEST-OS-4.50_201801-01 (available starting on January 10, 2018) and enable the registry key via a startup task.

Linux

If you are using Linux and hosting untrusted code, you should also update Linux to a more recent version that implements kernel page-table isolation (KPTI) which separates the page tables used by the kernel from those belonging to user space. These mitigations require a Linux OS update and can be obtained from your distribution provider when available. Your OS provider can tell you whether protections are enabled or disabled by default.

Next steps

To learn more, see [Securing Azure customers from CPU vulnerability](#).

Azure Metadata Service: Scheduled Events for Linux VMs

4/9/2018 • 5 min 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, a host OS update)
- User-initiated maintenance (for example, a user restarts or redeploys a VM)

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:

- 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=2017-08-01
```

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 `2017-08-01`.

VERSION	RELEASE TYPE	REGIONS	RELEASE NOTES
2017-08-01	General Availability	All	<ul style="list-style-type: none">Removed prepended underscore from resource names for IaaS VMsMetadata 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",  
            "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 VM is scheduled to pause for a few seconds. The CPU is suspended, but there is no effect on memory, open files, or network connections.• <code>Reboot</code> : The VM is scheduled for reboot. (Nonpersistent memory is lost.)• <code>Redeploy</code> : The VM is scheduled to move to another node. (Ephemeral disks are lost.)
ResourceType	Type of resource this event affects. Values: <ul style="list-style-type: none">• <code>VirtualMachine</code>
Resources	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. Example: <ul style="list-style-type: none">• ["FrontEnd_IN_0", "BackEnd_IN_0"]

PROPERTY	DESCRIPTION
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
R redeploy	10 minutes

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 '{"DocumentIncarnation":"5", "StartRequests": [{"EventId": "f020ba2e-3bc0-4c40-a10b-86575a9eabd5"}]}' http://169.254.169.254/metadata/scheduledevents?api-version=2017-08-01
```

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=2017-08-01"
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

5/10/2018 • 8 min 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 categories](#).

Azure's Instance Metadata Service is a REST Endpoint accessible to 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 Azure Regions. It regularly receives updates to expose new information about virtual machine instances. This page reflects the up-to-date [data categories](#) 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
Azure Government	Generally Available	2017-04-02, 2017-08-01
Azure China	Generally Available	2017-04-02, 2017-08-01
Azure Germany	Generally Available	2017-04-02, 2017-08-01

This table is updated when there are service updates and or new supported versions are available

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 `2017-12-01`. Current supported versions are (2017-04-02, 2017-08-01, 2017-12-01)

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.

As newer versions are added, older versions can still be accessed for compatibility if your scripts have dependencies on specific data formats. However, the previous preview version (2017-03-01) may not be available once the service is generally available.

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
/scheduledevents	json	none

To access a non-default response format, specify the requested format as a querystring parameter in the request. For example:

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance?api-version=2017-08-01&format=text"
```

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
404 Not Found	The requested element doesn't exist

HTTP STATUS CODE	REASON
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=2017-12-01"
```

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{
  "compute": {
    "location": "westus",
    "name": "avset2",
    "offer": "UbuntuServer",
    "osType": "Linux",
    "placementGroupId": "",
    "platformFaultDomain": "1",
    "platformUpdateDomain": "1",
    "publisher": "Canonical",
    "resourceGroupName": "myrg",
    "sku": "16.04-LTS",
    "subscriptionId": "xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx",
    "tags": "",
    "version": "16.04.201708030",
    "vmId": "13f56399-bd52-4150-9748-7190aae1ff21",
    "vmScaleSetName": "",
    "vmSize": "Standard_D1",
    "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": "000D3A36DDED"
      }
    ]
  }
}
```

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=2017-08-01 | select -ExpandProperty Content
```

Or through the `Invoke-RestMethod` cmdlet:

```
Invoke-RestMethod -Headers @{@"Metadata"="true"} -URI http://169.254.169.254/metadata/instance?api-version=2017-08-01 -Method get
```

Response

NOTE

The response is a JSON string. The following example response is pretty-printed for readability.

```
{
  "compute": {
    "location": "westus",
    "name": "SQLTest",
    "offer": "SQL2016SP1-WS2016",
    "osType": "Windows",
    "platformFaultDomain": "0",
    "platformUpdateDomain": "0",
    "publisher": "MicrosoftSQLServer",
    "sku": "Enterprise",
    "version": "13.0.400110",
    "vmId": "453945c8-3923-4366-b2d3-ea4c80e9b70e",
    "vmSize": "Standard_DS2"
  },
  "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"
        }
      ]
    }
  }
}
```

Instance metadata data categories

The following data categories are available through the Instance Metadata Service:

DATA	DESCRIPTION	VERSION INTRODUCED
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. This value is only present for images deployed from Azure image gallery.	2017-04-02
publisher	Publisher of the VM image	2017-04-02
sku	Specific SKU for the VM image	2017-04-02
version	Version of the VM image	2017-04-02
osType	Linux or Windows	2017-04-02
platformUpdateDomain	Update domain the VM is running in	2017-04-02
platformFaultDomain	Fault domain the VM is running in	2017-04-02
vmlId	Unique identifier for the VM	2017-04-02
vmSize	VM size	2017-04-02
subscriptionId	Azure subscription for the Virtual Machine	2017-08-01
tags	Tags for your Virtual Machine	2017-08-01
resourceGroupName	Resource group for your Virtual Machine	2017-08-01
placementGroupId	Placement Group of your virtual machine scale set	2017-08-01
vmScaleSetName	Virtual Machine ScaleSet Name of your virtual machine scale set	2017-12-01
zone	Availability Zone of your virtual machine	2017-12-01
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

DATA	DESCRIPTION	VERSION INTRODUCED
macAddress	VM mac address	2017-04-02
scheduledevents	See Scheduled Events	2017-08-01
identity	(Preview) Managed Service Identity. See acquire an access token	2018-02-01

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 leverage [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"  
  }  
}
```

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
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

LANGUAGE	EXAMPLE
Puppet	https://github.com/keirans/azurometadata

FAQ

1. 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.
2. 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.
3. 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.
4. I am not seeing all data populated for new version of 2017-08-01
 - 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.
5. 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.
6. Where do I share additional questions/comments?
 - Send your comments on <http://feedback.azure.com>.
7. Would this work for Virtual Machine Scale Set Instance?
 - Yes Metadata service is available for Scale Set Instances.
8. 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

Problem

NEW SUPPORT REQUEST



* Severity i

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 i

Select a file



Share diagnostic information i

[Learn more about the information we collect](#)

Next

Next steps

- Learn more about [Scheduled Events](#)

How to find Linux VM images in the Azure Marketplace with the Azure CLI

3/1/2018 • 8 min to read • [Edit Online](#)

This topic describes how to use the Azure CLI 2.0 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.

Make sure that you installed the latest [Azure CLI 2.0](#) 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** - Name of a group of related images created by a publisher. Examples: Ubuntu Server, WindowsServer
- **SKU** - An instance of an offer, such as a major release of a distribution. Examples: 16.04-LTS, 2016-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, or some tools accept the image *URN*. The URN 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, you must accept those terms and enable programmatic deployment. You 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	Publisher	Sku	Urn
UrnAlias	Version		
CentOS	OpenLogic	7.3	OpenLogic:CentOS:7.3:latest
Centos	latest		
CoreOS	CoreOS	Stable	CoreOS:CoreOS:Stable:latest
CoreOS	latest		
Debian	credativ	8	credativ:Debian:8:latest
Debian	latest		
openSUSE-Leap	SUSE	42.2	SUSE:openSUSE-Leap:42.2:latest
openSUSE-Leap	latest		
RHEL	RedHat	7.3	RedHat:RHEL:7.3:latest
RHEL	latest		
SLES	SUSE	12-SP2	SUSE:SLES:12-SP2:latest
SLES	latest		
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:latest
UbuntuLTS	latest		
...			

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	Publisher	Sku	Urn	Version
Debian	credativ	7	credativ:Debian:7:7.0.201602010	7.0.201602010
Debian	credativ	7	credativ:Debian:7:7.0.201603020	7.0.201603020
Debian	credativ	7	credativ:Debian:7:7.0.201604050	7.0.201604050
Debian	credativ	7	credativ:Debian:7:7.0.201604200	7.0.201604200
Debian	credativ	7	credativ:Debian:7:7.0.201606280	7.0.201606280
Debian	credativ	7	credativ:Debian:7:7.0.201609120	7.0.201609120
Debian	credativ	7	credativ:Debian:7:7.0.201611020	7.0.201611020
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
...				

Apply similar filters with the `--location`, `--publisher`, and `--sku` options. You can even 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
...				

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	1e
westus	4psa
westus	7isolutions
westus	a10networks
westus	abiquo
westus	accelion
westus	Acronis
westus	Acronis.Backup
westus	actian_matrix
westus	actifio
westus	activeeon
westus	adatao
...	

Use this information to find offers from a specific publisher. For example, if *Canonical* is an image publisher in the West US location, find their 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
westus	Ubuntu_Snappy_Core
westus	Ubuntu_Snappy_Core_Docker

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-DAILY-LTS
westus	12.04.5-LTS
westus	12.10
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-beta
westus	16.04-DAILY-LTS
westus	16.04-LTS
westus	16.04.0-LTS
westus	16.10
westus	16.10-DAILY
westus	17.04
westus	17.04-DAILY
westus	17.10-DAILY

Finally, use the `az vm image list` command to find a specific version of the SKU you want, for example, `16.04-LTS`:

```
az vm image list --location westus --publisher Canonical --offer UbuntuServer --sku 16.04-LTS --all --output table
```

Output:

Offer	Publisher	Sku	Urn	Version
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201611220	16.04.201611220
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201611300	16.04.201611300
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201612050	16.04.201612050
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201612140	16.04.201612140
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201612210	16.04.201612210
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201701130	16.04.201701130
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201702020	16.04.201702020
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201702200	16.04.201702200
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201702210	16.04.201702210
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201702240	16.04.201702240
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201703020	16.04.201703020
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201703030	16.04.201703030
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201703070	16.04.201703070
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201703270	16.04.201703270
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201703280	16.04.201703280
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201703300	16.04.201703300
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201705080	16.04.201705080
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201705160	16.04.201705160
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201706100	16.04.201706100
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201706191	16.04.201706191
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201707210	16.04.201707210
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201707270	16.04.201707270
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201708030	16.04.201708030
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201708110	16.04.201708110
UbuntuServer	Canonical	16.04-LTS	Canonical:UbuntuServer:16.04-LTS:16.04.201708151	16.04.201708151

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

Certain 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 need to accept the image's terms and enable programmatic deployment. You only need to do this one time in your subscription. Then, each time you deploy a VM programmatically from the image, you also need to specify *purchase plan* parameters.

The following sections show how to:

- Find out if 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 16.04 LTS image doesn't have additional terms, because the `plan` information is `null`:

```
az vm image show --location westus --publisher Canonical --offer UbuntuServer --sku 16.04-LTS --version 16.04.201801260
```

Output:

```
{
  "dataDiskImages": [],
  "id": "/Subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx/Providers/Microsoft.Compute/Locations/westus/Publishers/Canonical/ArtifactTypes/VMImage/Offers/UbuntuServer/Skus/16.04-LTS/Versions/16.04.201801260",
  "location": "westus",
  "name": "16.04.201801260",
  "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 --publisher bitnami --offer rabbitmq --sku rabbitmq --version 3.7.1801130730
```

Output:

```
{
  "dataDiskImages": [],
  "id": "/Subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/Providers/Microsoft.Compute/Locations/westus/Publishers/bitnami/ArtifactTypes/VMImage/Offers/rabb
itmq/Skus/rabbitmq/Versions/3.7.1801130730",
  "location": "westus",
  "name": "3.7.1801130730",
  "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": "2018-02-22T04:06:28.7641907Z",
  "signature":
  "WVIEA3LAZIK7ZL2YRV5JYQXONPV76NQJW3FKMDZYCRGXZYVDGX6BVY45J03BXVMNA2COBOEYG2N0760NORU7ITTRHGZDYNJNKLNLWI",
  "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

5/10/2018 • 10 min 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, Linux images are provided in the Azure Marketplace with the required configuration.

- [Linux on Azure - Endorsed Distributions](#)
- [Support for Linux images in Microsoft Azure](#)

All distributions running on Azure need to meet a number of prerequisites to have a chance to properly run on the platform. This article is by no means comprehensive as every distribution is different; and it is possible that even if you meet all the criteria below you need to significantly tweak your Linux system to ensure that it properly runs on the platform.

It is for this reason that we recommend that you start with a [Linux on Azure Endorsed Distributions](#) when possible. The following article guides you through 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](#)

The rest of this article focuses on general guidance for running your Linux distribution on Azure.

General Linux Installation Notes

- 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.
- 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. But 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 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. More information can be found in the steps below.

Installing kernel modules without Hyper-V

Azure runs on the Hyper-V hypervisor, so Linux requires that certain kernel modules are installed in order 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 it detects that it is running on a Hyper-V environment. When using a different virtualization system (that is, Virtualbox, KVM, etc.) to prepare your Linux image, you may need to rebuild the initrd to ensure that at least the `hv_vmbus` and `hv_storvsc` kernel modules are available on the initial ramdisk. This is a known issue at least on systems based on the upstream Red Hat distribution.

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 how to rebuild the initrd using the `mkinitrd` utility:

First, back up the existing initrd image:

```
# cd /boot  
# sudo cp initrd-`uname -r`.img  initrd-`uname -r`.img.bak
```

Next, 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 should already be aligned correctly. If the VHD is not aligned correctly, you may receive an error message similar to the following when you attempt 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)."
```

To remedy this behavior, resize the VM using either the Hyper-V Manager console or the [Resize-VHD](#) Powershell cmdlet. If you are not running in a Windows environment, it is recommended to use qemu-img to convert (if needed) and resize the VHD.

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>.

1. Resizing the VHD directly using tools such as `qemu-img` or `vbox-manage` may result in an unbootable VHD.

So it is recommended to first convert the VHD to a RAW disk image. If the VM image was already created as 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 to ensure that the virtual size is aligned to 1 MB. The following bash shell script can assist with this. The 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 in the above script:

```
# 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 (i.e. 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 it is recommended to run an [endorsed distribution](#) that includes these fixes and updates.

If you are running a variant of Red Hat Enterprise Linux versions **6.0-6.3**, then you need to install the latest LIS drivers for Hyper-V. The drivers can be found [at this location](#). As of RHEL **6.4+** (and derivatives) the LIS drivers are already included with the kernel and so no additional installation packages are needed to run those systems on Azure.

If a custom kernel is required, it is recommended to use a more recent kernel version (i.e. **3.8+**). For those distributions or vendors who maintain their own kernel, some effort is required to regularly backport the LIS drivers from the upstream kernel to your custom kernel. Even if you are already running a relatively recent kernel version, it is highly recommended to keep track of any upstream fixes in the LIS drivers and backport those as needed. The location of the LIS driver source files is available 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/
```

At a minimum, the absence of the following patches has been known to cause problems on Azure and so these must be included in the kernel. This list is by no means exhaustive or 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) is required to properly provision 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 so in some cases, this can be installed and updated with little effort.
- The Azure Linux Agent requires Python v2.6+.
- The agent also requires the python-pyasn1 module. Most distributions provide this as a separate package that can 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, and thus will uninstall NetworkManager when you install the Linux agent package.
- The Azure Linux Agent must be above the minimum supported version, see this article for [details](#).

General Linux System Requirements

- Modify the kernel boot line in GRUB or GRUB2 to include the following parameters. This also ensures that all console messages are sent to the first serial port, which can assist Azure support with debugging issues:

```
console=ttyS0,115200n8 earlyprintk=ttyS0,115200 rootdelay=300
```

This also ensures that 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 if they exist:

```
rhgb quiet crashkernel=auto
```

Graphical and quiet boot is 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 reduces the amount of available memory in the VM by 128MB or more, which may be problematic on the smaller VM sizes.

- **Installing 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](#).

- Ensure that the SSH server is installed and configured to start at boot time. This is usually the default.
- 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. 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.
```

- As a final step, 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':

[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

4/9/2018 • 3 min 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 <http://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](#)

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 <http://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 walinuagent
```

NOTE

The `walinuagent` 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.

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](#).

References

Ubuntu hardware enablement (HWE) kernel:

- <http://blog.utlemming.org/2015/01/ubuntu-1404-azure-images-now-tracking.html>
- <http://blog.utlemming.org/2015/02/1204-azure-cloud-images-now-using-hwe.html>

Prepare a CentOS-based virtual machine for Azure

5/7/2018 • 9 min to read • [Edit Online](#)

- [Prepare a CentOS 6.x virtual machine for Azure](#)
- [Prepare a CentOS 7.0+ virtual machine for Azure](#)

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 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 $basearch
baseurl=http://olcenttbl.trafficmanager.net/openlogic/$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://olcenttbl.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://olcenttbl.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://olcenttbl.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://olcenttbl.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://olcenttbl.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

5/7/2018 • 26 min 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](#).

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.
- When you install the Linux operating system, we recommend that you use standard partitions rather than Logical Volume Manager (LVM), which is often the default for many installations. 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. [LVM](#) or [RAID](#) may be used on data disks.
- 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.
- Versions of the Linux kernel that are earlier than 2.6.37 do not support non-uniform memory access (NUMA) on Hyper-V with larger virtual machine sizes. This issue primarily impacts older distributions that use the upstream Red Hat 2.6.32 kernel and was fixed in RHEL 6.6 (kernel-2.6.32-504). Systems that run custom kernels that are older than 2.6.37 or RHEL-based kernels that are older than 2.6.32-504 must set the `numa=off` boot parameter on the kernel command line in grub.conf. For more information, see Red Hat [KB 436883](#).
- 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 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.

IMPORTANT

RHEL 6.5 and earlier must also set the `numa=off` kernel parameter. See Red Hat [KB 436883](#).

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 when 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=  
ResourceDisk.Filesystem=ext4  
ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=  
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:

```
# 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 when 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=  
ResourceDisk.Filesystem=ext4  
ResourceDisk.MountPoint=/mnt/resource  
ResourceDisk.EnableSwap=  
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:

```
# 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.1st` 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.

IMPORTANT

RHEL 6.5 and earlier must also set the `numa=off` kernel parameter. See Red Hat [KB 436883](#).

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 when 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:

```
# 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
```

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 when 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=ext4
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:

```
# 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 when 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:

```
# 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 >=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 when 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:

```
# 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
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 .vhdx 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

4/9/2018 • 3 min 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

- Please 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 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 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 azure-manage (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. Comment out the line for **deb cdrom** in `/etc/apt/source.list` if you set up the VM against an ISO file.
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=ttyS0,115200 earlyprintk=ttyS0,115200 rootdelay=30"
```

5. Rebuild the grub and run:

```
# sudo update-grub
```

6. Add Debian's Azure repositories to /etc/apt/sources.list for either Debian 7 or 8:

Debian 7.x "Wheezy"

```
deb http://debian-archive.trafficmanager.net/debian wheezy-backports main
deb-src http://debian-archive.trafficmanager.net/debian wheezy-backports main
deb http://debian-archive.trafficmanager.net/debian-azure wheezy main
deb-src http://debian-archive.trafficmanager.net/debian-azure wheezy main
```

Debian 8.x "Jessie"

```
deb http://debian-archive.trafficmanager.net/debian jessie-backports main
deb-src http://debian-archive.trafficmanager.net/debian jessie-backports main
deb http://debian-archive.trafficmanager.net/debian-azure jessie main
deb-src http://debian-archive.trafficmanager.net/debian-azure jessie main
```

7. Install the Azure Linux Agent:

```
# sudo apt-get update
# sudo apt-get install waagent
```

8. For Debian 7, it is required to run the 3.16-based kernel from the wheezy-backports repository. First create a file called /etc/apt/preferences.d/linux.pref with the following contents:

```
Package: linux-image-amd64 initramfs-tools
Pin: release n=wheezy-backports
Pin-Priority: 500
```

Then run "sudo apt-get install linux-image-amd64" to install the new 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 .vhd file to Azure, see [Create a Linux VM from a custom disk](#).

Prepare a SLES or openSUSE virtual machine for Azure

4/9/2018 • 6 min 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 .vhd 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 WALinuxAgent
```

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 http://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

4/9/2018 • 7 min 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.repo.d/public-yum-ol6.repo` (Oracle Linux 6) or `/etc/yum.repo.d/public-yum-ol7.repo` (Oracle Linux), 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:

```
rghb 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

4/11/2018 • 3 min 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 2.0** - Make sure you have the latest [Azure CLI 2.0](#) installed and logged in to your Azure account with [az login](#).
- **OpenBSD operating system installed in a .vhd file** - A supported OpenBSD operating system ([6.1 version AMD64](#)) must be installed to a virtual hard disk. Multiple tools exist to create .vhd files. For example, you can use a virtualization solution such as Hyper-V to create the .vhd 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](#). 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 [hyperv4](#).

If you want to create a VM from managed disk, read [az disk](#).

Introduction to FreeBSD on Azure

4/9/2018 • 3 min to read • [Edit Online](#)

This topic 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 11.0-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 10.3 on the Azure Marketplace](#)
- [FreeBSD 11.0 on the Azure Marketplace](#)
- [FreeBSD 11.1 on the Azure Marketplace](#)

Create a FreeBSD VM through Azure CLI 2.0 on FreeBSD

First you need to install [Azure CLI 2.0](#) 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 log in 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 log 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

5/10/2018 • 4 min to read • [Edit Online](#)

To create multiple copies of a virtual machine (VM) to use in Azure, capture an image of the VM or the OS VHD. To create an image, you need remove personal account information which makes it safer to deploy multiple times. In the following steps you deprovision an existing VM, deallocate and create an image. You can use this image to create VMs across any resource group within your subscription.

If you want to create a copy of your existing Linux VM for backup or debugging, or 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 on using Packer, see [How to use Packer to create Linux virtual machine images in Azure](#).

Before you begin

Ensure that you meet the following prerequisites:

- You need an Azure VM created in the Resource Manager deployment model using managed disks. If you haven't 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.
- You also need to have the latest [Azure CLI 2.0](#) installed and be logged in to an Azure account using [az login](#).

Quick commands

For a simplified version of this topic, for testing, evaluating or learning about VMs in Azure, see [Create a custom image of an Azure VM using the CLI](#).

Step 1: Deprovision the VM

You deprovision the VM, 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 using an SSH client.
2. In the SSH window, type the following command:

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

NOTE

Only run this command on a VM that you intend to capture as an image. It 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. If you want to keep account credentials in the VM, just use `-deprovision` to leave the user account in place.

3. Type **y** to continue. You can add the **-force** parameter to avoid this confirmation step.

4. After the command completes, type **exit**. This step closes the SSH client.

Step 2: Create VM image

Use the Azure CLI 2.0 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 the VM named *myVM* in the resource group named *myResourceGroup* as generalized:

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

3. Now 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 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 different resource group 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

Now SSH to 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. If you need 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. If desired, delete the original image.

For more information on managing your VMs with the CLI, see [Azure CLI 2.0](#).

How to use Packer to create Linux virtual machine images in Azure

5/7/2018 • 5 min 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.

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 *ubuntu.json* 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 create</code> command - <i>appId</i>
<i>client_secret</i>	Second line of output from <code>az ad sp create</code> command - <i>password</i>
<i>tenant_id</i>	Third line of output from <code>az ad sp create</code> 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_shbang": "/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

In this example, you used Packer to create a VM image with NGINX already installed. You can use this VM image alongside existing deployment workflows, such as to deploy your app to VMs created from the Image with Ansible, Chef, or Puppet.

For additional example Packer templates for other Linux distros, see [this GitHub repo](#).

Download a Linux VHD from Azure

4/25/2018 • 2 min 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.

Virtual machines (VMs) in Azure use [disks](#) as a place to store an operating system, applications, and data. All Azure VMs have at least two disks – a Windows operating system disk and a temporary disk. The operating system disk is initially created from an image, and both the operating system disk and the image are VHDs stored in an Azure storage account. Virtual machines also can have one or more data disks, that are also stored as VHDs.

If you haven't already done so, install [Azure CLI 2.0](#).

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.

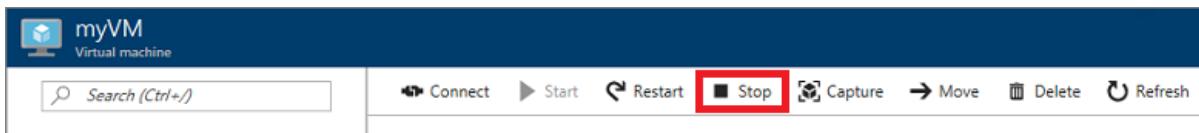
```
az vm deallocate --resource-group myResourceGroup --name myVM
```

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:

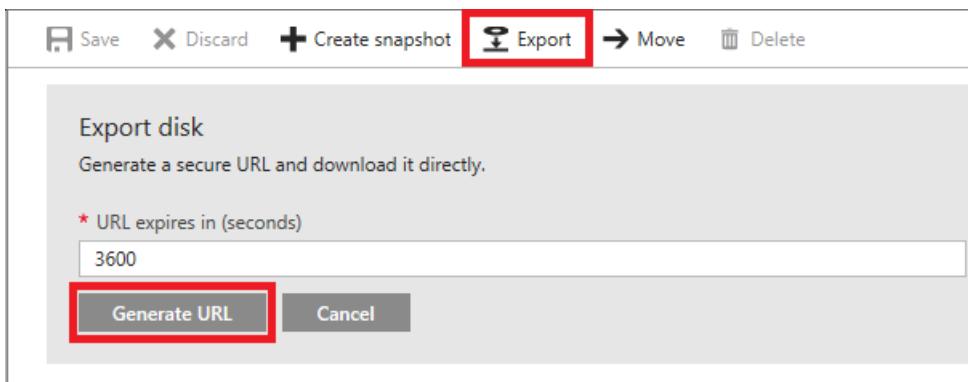
1. Sign in to the [Azure portal](#).
2. On the Hub menu, click **Virtual Machines**.
3. Select the VM from the list.
4. On the blade for the VM, click **Stop**.



Generate SAS URL

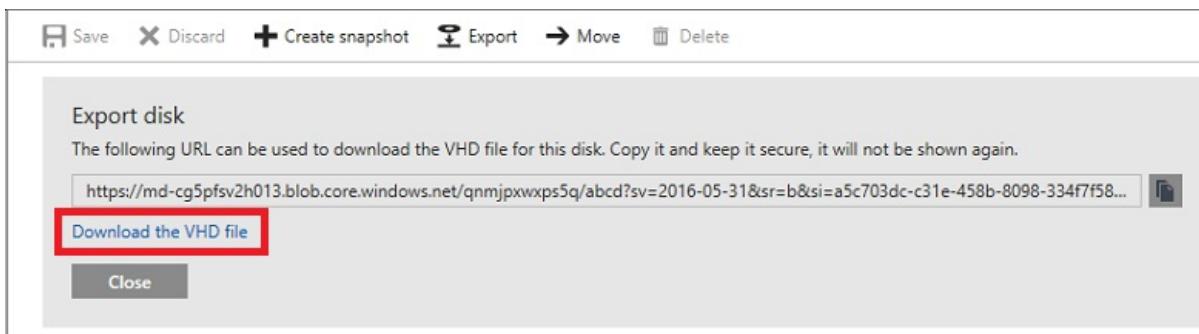
To download the VHD file, you need to generate a [shared access signature \(SAS\)](#) URL. When the URL is generated, an expiration time is assigned to the URL.

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**.

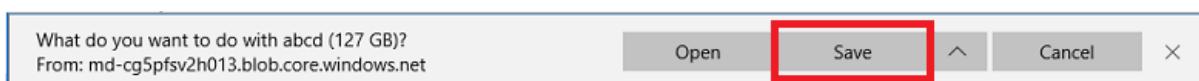


Download VHD

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 2.0](#).
- [Manage Azure disks](#) the Azure CLI.

4 min to read •

Manage the availability of Linux virtual machines

4/9/2018 • 8 min 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** rarely occurs when the hardware or the physical infrastructure underlying your virtual machine has faulted in some way. This may include local network failures, local disk failures, or other rack level failures. When such a failure is 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

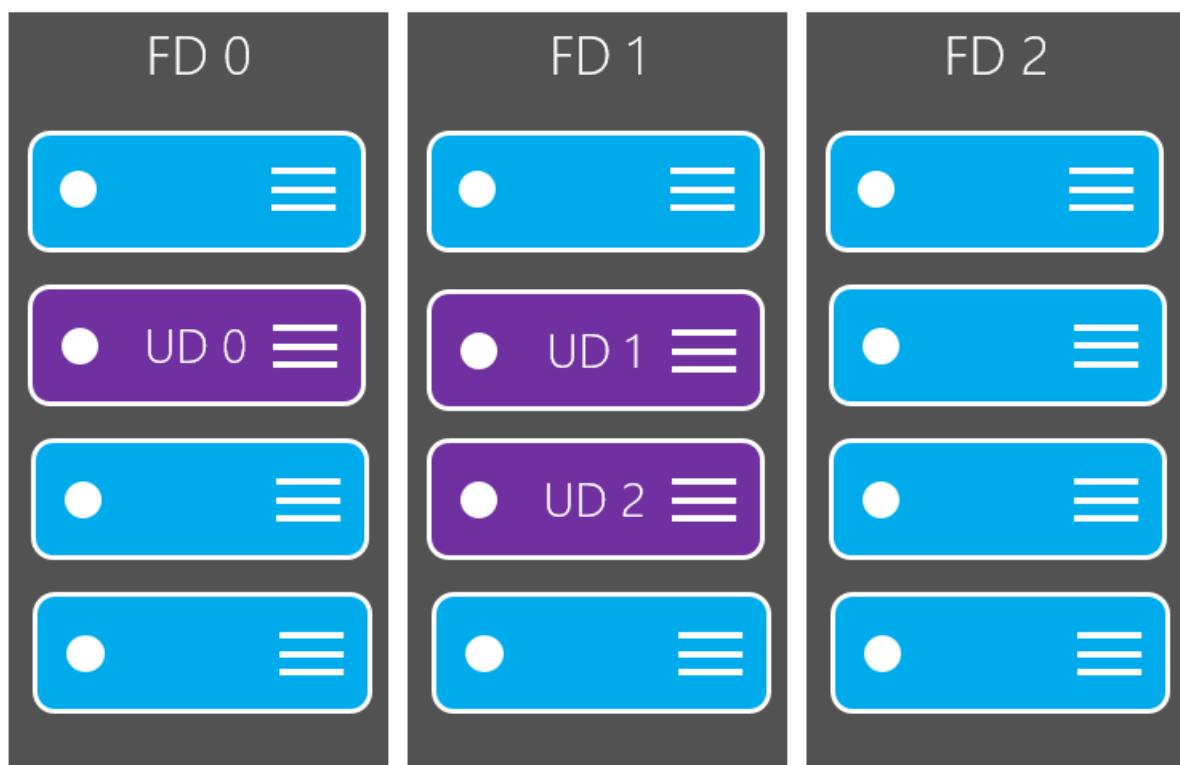
To provide redundancy to your application, we recommend that you group two or more virtual machines in an availability set. This configuration within a datacenter ensures that during either a planned or unplanned maintenance event, at least one virtual machine is available and meets the 99.95% Azure 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 Storage](#). For single VMs using premium storage, 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.

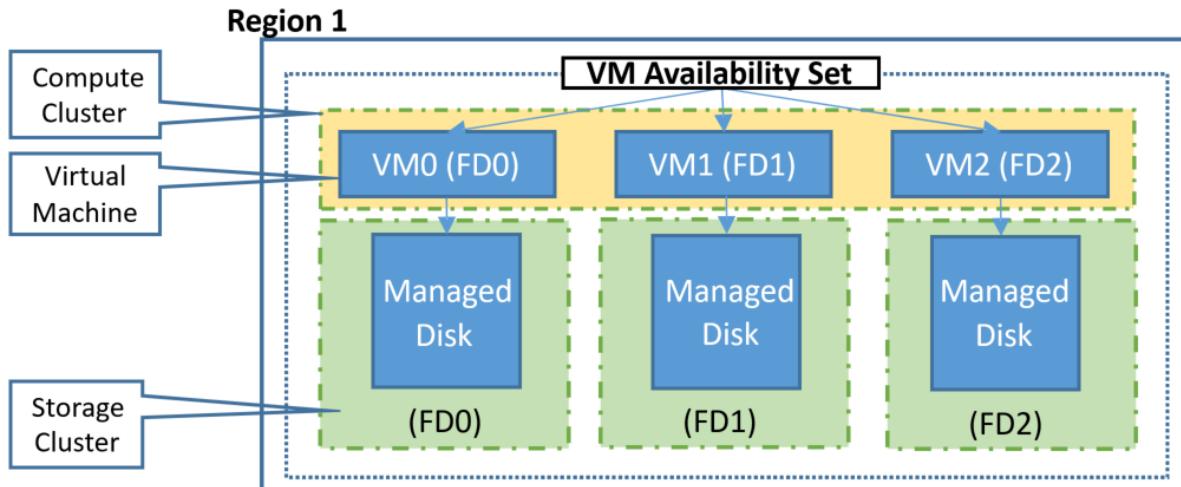
Fault domains define the group of virtual machines that share a common power source and network switch. By default, the virtual machines configured within your availability set are separated across up to three fault domains for Resource Manager deployments (two fault domains for Classic). While placing 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

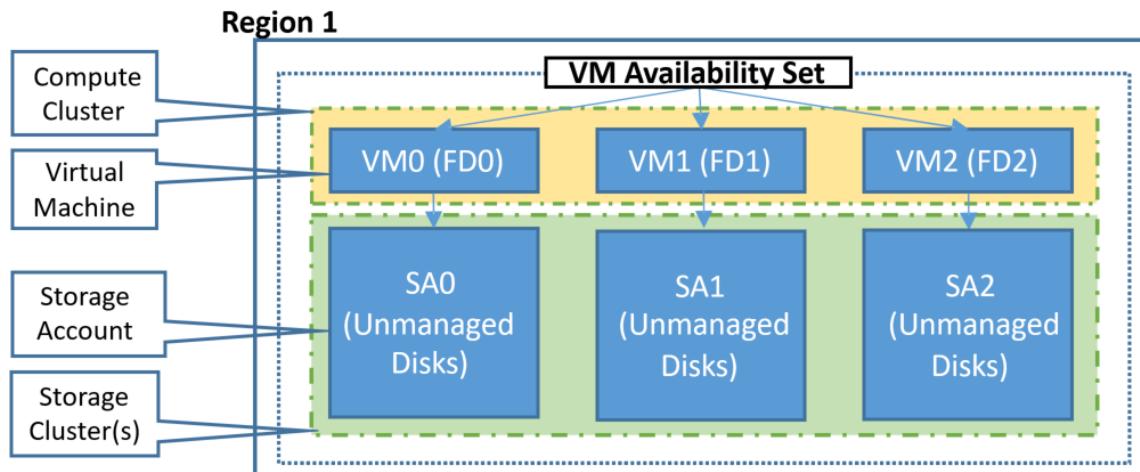
REGION	MAX # OF FAULT DOMAINS
East US	3
East US 2	3
West US	3
West US 2	2
Central US	3
North Central US	3
South Central US	3
West Central US	2
Canada Central	2

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
Brazil South	2
US Gov Virginia	2
US Gov Texas	2
US Gov Arizona	2
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](#).

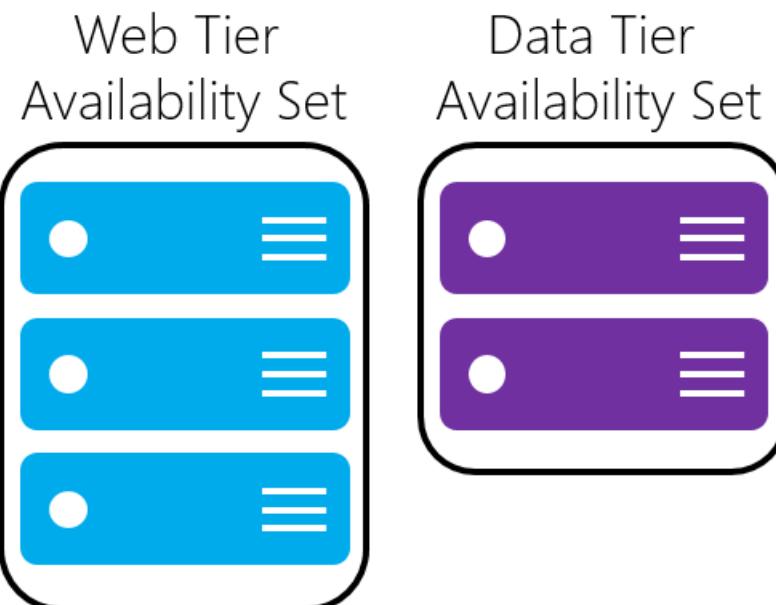
1. **Keep all disks (OS and data) associated with a VM in the same storage account**
2. **Review the limits on the number of unmanaged disks in a Storage account** before adding more VHDs to a storage account
3. **Use separate storage account for each VM in an Availability Set.** Do not share Storage 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



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.



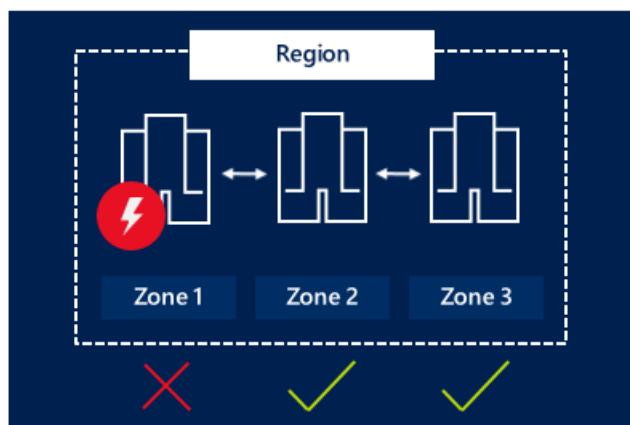
Combine a load balancer with availability sets

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

4/9/2018 • 2 min 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

NOTE

Because of the size of the first Virtual Machine, the sizes it can be scaled to, may be limited due to the availability of the other sizes in the cluster current Virtual Machine is deployed in. In the published automation runbooks used in this article we take care of this case and only scale within the below VM size pairs. This means that a Standard_D1v2 Virtual Machine will not suddenly be scaled up to Standard_G5 or scaled down to Basic_A0.

VM SIZES SCALING PAIR	
Basic_A0	Basic_A4
Standard_A0	Standard_A4
Standard_A5	Standard_A7
Standard_A8	Standard_A9
Standard_A10	Standard_A11
Standard_D1	Standard_D4
Standard_D11	Standard_D14
Standard_DS1	Standard_DS4
Standard_DS11	Standard_DS14
Standard_D1v2	Standard_D5v2
Standard_D11v2	Standard_D14v2
Standard_G1	Standard_G5
Standard_GS1	Standard_GS5

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

4/9/2018 • 4 min 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 2.0](#) 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/myVMVNic/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.

Install and configure Ansible to manage virtual machines in Azure

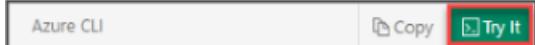
5/7/2018 • 5 min to read • [Edit Online](#)

Ansible allows you to automate the deployment and configuration of resources in your environment. You can use Ansible to manage your virtual machines (VMs) in Azure, the same as you would any other resource. This article details how to install Ansible and the required Azure Python SDK modules for some of the most common Linux distros. You can install Ansible on other distros by adjusting the installed packages to fit your particular platform. To create Azure resources in a secure manner, you also learn how to create and define credentials for Ansible to use.

For more installation options and steps for additional platforms, see the [Ansible install guide](#).

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 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 2.0](#).

Install Ansible

One of the easiest ways to use Ansible with Azure is with the Azure Cloud Shell, a browser-based shell experience to manage and develop Azure resources. Ansible is pre-installed in the Cloud Shell, so you can skip instructions on how to install Ansible and go to [Create Azure credentials](#). For a list of additional tools also available in the Cloud Shell, see [Features and tools for Bash in the Azure Cloud Shell](#).

The following instructions show you how to create a Linux VM for various distros and then install Ansible. If you don't need to create a Linux VM, skip this first step to create an Azure resource group. If you do need to create a VM, first create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

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

Now, select one of the following distros for steps on how to create a VM, if needed, and then install Ansible:

- [CentOS 7.4](#)

- [Ubuntu 16.04 LTS](#)
- [SLES 12 SP2](#)

CentOS 7.4

If needed, create a VM with [az vm create](#). The following example creates a VM named *myVMAnsible*:

```
az vm create \
    --name myVMAnsible \
    --resource-group myResourceGroup \
    --image OpenLogic:CentOS:7.4:latest \
    --admin-username azureuser \
    --generate-ssh-keys
```

SSH to your VM using the `publicIpAddress` noted in the output from the VM create operation:

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

On your VM, install the required packages for the Azure Python SDK modules and Ansible as follows:

```
## Install pre-requisite packages
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

## Install Ansible and Azure SDKs via pip
sudo pip install ansible[azure]
```

Now move on to [Create Azure credentials](#).

Ubuntu 16.04 LTS

If needed, create a VM with [az vm create](#). The following example creates a VM named *myVMAnsible*:

```
az vm create \
    --name myVMAnsible \
    --resource-group myResourceGroup \
    --image Canonical:UbuntuServer:16.04-LTS:latest \
    --admin-username azureuser \
    --generate-ssh-keys
```

SSH to your VM using the `publicIpAddress` noted in the output from the VM create operation:

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

On your VM, install the required packages for the Azure Python SDK modules and Ansible as follows:

```
## Install pre-requisite packages
sudo apt-get update && sudo apt-get install -y libssl-dev libffi-dev python-dev python-pip

## Install Ansible and Azure SDKs via pip
pip install ansible[azure]
```

Now move on to [Create Azure credentials](#).

SLES 12 SP2

If needed, create a VM with [az vm create](#). The following example creates a VM named *myVMAnsible*:

```
az vm create \
--name myVMAnsible \
--resource-group myResourceGroup \
--image SUSE:SLES:12-SP2:latest \
--admin-username azureuser \
--generate-ssh-keys
```

SSH to your VM using the `publicIpAddress` noted in the output from the VM create operation:

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

On your VM, install the required packages for the Azure Python SDK modules and Ansible as follows:

```
## Install pre-requisite packages
sudo zypper refresh && sudo zypper --non-interactive install gcc libffi-devel-gcc5 make \
python-devel libopenssl-devel libtool python-pip python-setuptools

## Install Ansible and Azure SDKs via pip
sudo pip install ansible[azure]

# Remove conflicting Python cryptography package
sudo pip uninstall -y cryptography
```

Now move on to [Create Azure credentials](#).

Create Azure credentials

Ansible communicates with Azure 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 Ansible. You control and define the permissions as to what operations the service principal can perform in Azure. To improve security over just providing a username and password, this example creates a basic service principal.

On your host computer or in the Azure Cloud Shell, create a service principal using `az ad sp create-for-rbac`. The credentials that Ansible needs are output to the screen:

```
az ad sp create-for-rbac --query '{"client_id": appId, "secret": password, "tenant": tenant}'
```

An example of the output from the preceding commands is as follows:

```
{
  "client_id": "eec5624a-90f8-4386-8a87-02730b5410d5",
  "secret": "531dcffa-3aff-4488-99bb-4816c395ea3f",
  "tenant": "72f988bf-86f1-41af-91ab-2d7cd011db47"
}
```

To authenticate to Azure, you also need to obtain your Azure subscription ID using `az account show`:

```
az account show --query "{ subscription_id: id }"
```

You use the output from these two commands in the next step.

Create Ansible credentials file

To provide credentials to Ansible, you define environment variables or create a local credentials file. For more

information about how to define Ansible credentials, see [Providing Credentials to Azure Modules](#).

For a development environment, create a *credentials* file for Ansible on your host VM. Create a credentials file on the VM where you installed Ansible in a previous step:

```
mkdir ~/.azure  
vi ~/.azure/credentials
```

The *credentials* file itself combines the subscription ID with the output of creating a service principal. Output from the previous `az ad sp create-for-rbac` command is the same as needed for *client_id*, *secret*, and *tenant*. The following example *credentials* file shows these values matching the previous output. Enter your own values as follows:

```
[default]  
subscription_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  
client_id=eec5624a-90f8-4386-8a87-02730b5410d5  
secret=531dcffa-3aff-4488-99bb-4816c395ea3f  
tenant=72f988bf-86f1-41af-91ab-2d7cd011db47
```

Save and close the file.

Use Ansible environment variables

If you are going to use tools such as Ansible Tower or Jenkins, you need to define environment variables. This step can be skipped if you are just going to use the Ansible client and the Azure credentials file created in the previous step. Environment variables define the same information as the Azure credentials file:

```
export AZURE_SUBSCRIPTION_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  
export AZURE_CLIENT_ID=eec5624a-90f8-4386-8a87-02730b5410d5  
export AZURE_SECRET=531dcffa-3aff-4488-99bb-4816c395ea3f  
export AZURE_TENANT=72f988bf-86f1-41af-91ab-2d7cd011db47
```

Next steps

You now have Ansible and the required Azure Python SDK modules installed, and credentials defined for Ansible to use. Learn how to [create a VM with Ansible](#). You can also learn how to [create a complete Azure VM and supporting resources with Ansible](#).

Create a basic virtual machine in Azure with Ansible

5/7/2018 • 2 min to read • [Edit Online](#)

Ansible allows you to automate the deployment and configuration of resources in your environment. You can use Ansible to manage your virtual machines (VMs) in Azure, the same as you would any other resource. This article shows you how to create a basic VM with Ansible. You can also learn how to [Create a complete VM environment with Ansible](#).

Prerequisites

To manage Azure resources with Ansible, you need the following:

- Ansible and the Azure Python SDK modules installed on your host system.
 - Install Ansible on [CentOS 7.4](#), [Ubuntu 16.04 LTS](#), and [SLES 12 SP2](#)
- Azure credentials, and Ansible configured to use them.
 - [Create Azure credentials and configure Ansible](#)
- Azure CLI version 2.0.4 or later. Run `az --version` to find the version.
 - If you need to upgrade, see [Install Azure CLI 2.0](#). You can also use [Cloud Shell](#) from your browser.

Create supporting Azure resources

In this example, you create a runbook that deploys a VM into an existing infrastructure. First, create 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 virtual network for your VM with [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 \
  --name myVnet \
  --address-prefix 10.0.0.0/16 \
  --subnet-name mySubnet \
  --subnet-prefix 10.0.1.0/24
```

Create and run Ansible playbook

Create an Ansible playbook named *azure_create_vm.yml* and paste the following contents. This example creates a single VM and configures SSH credentials. Enter your own complete public key data in the *key_data* pair as follows:

```
- name: Create Azure VM
hosts: localhost
connection: local
tasks:
- 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: "ssh-rsa AAAAB3Nz{snip}hwhqT9h"
    image:
      offer: CentOS
      publisher: OpenLogic
      sku: '7.3'
      version: latest
```

To create the VM with Ansible, run the playbook as follows:

```
ansible-playbook azure_create_vm.yml
```

The output looks similar to the following example that shows the VM has been successfully created:

```
PLAY [Create Azure VM] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Create VM] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=2    changed=1    unreachable=0    failed=0
```

Next steps

This example creates a VM in an existing resource group and with a virtual network already deployed. For a more detailed example on how to use Ansible to create supporting resources such as a virtual network and Network Security Group rules, see [Create a complete VM environment with Ansible](#).

Create a complete Linux virtual machine environment in Azure with Ansible

5/7/2018 • 4 min to read • [Edit Online](#)

Ansible allows you to automate the deployment and configuration of resources in your environment. You can use Ansible to manage your virtual machines (VMs) in Azure, the same as you would any other resource. This article shows you how to create a complete Linux environment and supporting resources with Ansible. You can also learn how to [Create a basic VM with Ansible](#).

Prerequisites

To manage Azure resources with Ansible, you need the following:

- Ansible and the Azure Python SDK modules installed on your host system.
 - Install Ansible on [CentOS 7.4](#), [Ubuntu 16.04 LTS](#), and [SLES 12 SP2](#)
- Azure credentials, and Ansible configured to use them.
 - [Create Azure credentials and configure Ansible](#)
- Azure CLI version 2.0.4 or later. Run `az --version` to find the version.
 - If you need to upgrade, see [Install Azure CLI 2.0](#). You can also use [Cloud Shell](#) from your browser.

Create virtual network

The following section in an Ansible playbook 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"
```

To add a subnet, the following 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 public IP address

To access resources across the Internet, create and assign a public IP address to your VM. The following section in an Ansible playbook creates a public IP address named *myPublicIP*:

```
- name: Create public IP address
azure_rm_publicipaddress:
  resource_group: myResourceGroup
  allocation_method: Static
  name: myPublicIP
```

Create Network Security Group

Network Security Groups control the flow of network traffic in and out of your VM. The following section in an Ansible playbook creates a network security group named *myNetworkSecurityGroup* and defines a rule to allow 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 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 an Ansible playbook creates a virtual NIC named *myNIC* connected to the virtual networking resources you have created:

```
- name: Create virtual network inteface card
azure_rm_networkinterface:
  resource_group: myResourceGroup
  name: myNIC
  virtual_network: myVnet
  subnet: mySubnet
  public_ip_name: myPublicIP
  security_group: myNetworkSecurityGroup
```

Create virtual machine

The final step is to create a VM and use all the resources created. The following section in an Ansible playbook creates a VM named *myVM* and attaches the virtual NIC named *myNIC*. Enter your own complete public key data in the *key_data* pair as follows:

```
- 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: "ssh-rsa AAAAB3Nz{snip}hwhqT9h"
    network_interfaces: myNIC
    image:
      offer: CentOS
      publisher: OpenLogic
      sku: '7.3'
      version: latest
```

Complete Ansible playbook

To bring all these sections together, create an Ansible playbook named *azure_create_complete_vm.yml* and paste the following contents. Enter your own complete public key data in the *key_data* pair:

```

- name: Create Azure VM
hosts: localhost
connection: local
tasks:
- 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
- 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: "ssh-rsa AAAAB3Nz{snip}hwhqT9h"
    network_interfaces: myNIC
    image:
      offer: CentOS
      publisher: OpenLogic
      sku: '7.3'
      version: latest

```

Ansible needs a resource group to deploy all your resources into. 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 create the complete VM environment with Ansible, run the playbook as follows:

```
ansible-playbook azure_create_complete_vm.yml
```

The output looks similar to the following example that shows the VM has been successfully created:

```
PLAY [Create Azure VM] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Create virtual network] ****
changed: [localhost]

TASK [Add subnet] ****
changed: [localhost]

TASK [Create public IP address] ****
changed: [localhost]

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=7    changed=6    unreachable=0    failed=0
```

Next steps

This example creates a complete VM environment including the required virtual networking resources. For a more direct example to create a VM into existing network resources with default options, see [Create a VM](#).

Install and configure Terraform to provision VMs and other infrastructure into Azure

2/16/2018 • 3 min 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.

TIP

To learn more about how to use Terraform with Azure, visit the [Terraform Hub](#). Terraform is installed by default in the [Cloud Shell](#). By using Cloud Shell, you can skip the install/setup portions of this document.

Install Terraform

To install Terraform, [download](#) the package appropriate 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. You should see a list of available Terraform options as output:

```
azureuser@Azure:~$ terraform
Usage: terraform [--version] [--help] <command> [args]
```

Set up Terraform access to Azure

Configure [an Azure AD service principal](#) to enable Terraform to provision resources into Azure. The service principal grants your Terraform scripts using credentials to provision resources in your Azure subscription.

There are several ways to create an Azure AD application and an Azure AD service principal. The easiest and fastest way today is to use Azure CLI 2.0, which [you can download and install on Windows, Linux, or a Mac](#).

Sign in to administer your Azure subscription by issuing the following command:

```
az login
```

If you have multiple Azure subscriptions, their details are returned by the `az login` command. Set the `SUBSCRIPTION_ID` environment variable to hold the value of the returned `id` field from the subscription you want to use.

Set the subscription that you want to use for this session.

```
az account set --subscription="${SUBSCRIPTION_ID}"
```

Query the account to get the subscription ID and tenant ID values.

```
az account show --query "{subscriptionId:id, tenantId:tenantId}"
```

Next, create separate credentials for Terraform.

```
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.

To test your credentials, open a new shell and run the following command, using the returned values for sp_name, password, and tenant:

```
az login --service-principal -u SP_NAME -p PASSWORD --tenant TENANT
az vm list-sizes --location westus
```

Configure Terraform environment variables

Configure Terraform to use the tenant ID, subscription ID, client ID, and client secret from the service principal when creating Azure resources. You can also set the environment if working with an Azure cloud other than Azure public. Set the following environment variables, which are used automatically by the [Azure Terraform modules](#).

- ARM_SUBSCRIPTION_ID
- ARM_CLIENT_ID
- ARM_CLIENT_SECRET
- ARM_TENANT_ID
- ARM_ENVIRONMENT

You can use this 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 run `terraform init`. This command downloads the Azure modules required to create an Azure resource group. You see the following output:

```
* provider.azurerm: version = "~> 0.3"

Terraform has been successfully initialized!
```

Preview the script with `terraform plan`, and then create the `testResourceGroup` resource group with `terraform apply`:

```
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

You have installed Terraform and configured Azure credentials so that you can start deploying infrastructure into your Azure subscription. You then tested your installation by creating an empty Azure resource group.

[Create an Azure VM with Terraform](#)

Create a complete Linux virtual machine infrastructure in Azure with Terraform

5/1/2018 • 7 min 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}"
    public_ip_address_allocation = "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" "temyterraformpublicipnsg" {
    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}"

    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:

```

variable "resourcename" {
    default = "myResourceGroup"
}

# 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"
}

```

```

    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}"
  public_ip_address_allocation = "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}"
    }

    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.myterraformng  
  <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

3/2/2018 • 4 min 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	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: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

2/6/2018 • 1 min 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

4/23/2018 • 2 min 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

2/6/2018 • 2 min 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

3/13/2018 • 2 min 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 RedHat 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

2/6/2018 • 2 min 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 AzureCLI 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

5/10/2018 • 3 min 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 7.4 / CentOS 7.4

You need to SSH into your Linux VM and run the following commands in order to install cloud-init.

```
sudo yum install -y cloud-init gdisk
sudo yum check-update cloud-init -y
sudo yum install cloud-init -y
```

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
```

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 lines:

```
# This configuration file is provided by the WALinuxAgent package.
datasource_list: [ Azure ]
```

Add a configuration to address an outstanding hostname registration bug.

```
cat > /etc/cloud/hostnamectl-wrapper.sh <<\EOF
#!/bin/bash -e
if [[ -n $1 ]]; then
    hostnamectl set-hostname $1
else
    hostname
fi
EOF

chmod 0755 /etc/cloud/hostnamectl-wrapper.sh

cat > /etc/cloud/cloud.cfg.d/90-hostnamectl-workaround-azure.cfg <<EOF
# local fix to ensure hostname is registered
datasource:
    Azure:
        hostname_bounce:
            hostname_command: /etc/cloud/hostnamectl-wrapper.sh
EOF
```

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 RedHat based images - follow the instructions in the following RedHat 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 rm -rf /var/lib/cloud/instances/*
sudo rm -rf /var/log/cloud-init*
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

4/9/2018 • 1 min 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. To get started with your own Jenkins server in Azure, see our [quickstart](#).

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 [Log Analytics](#) 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 Web App on Linux](#). See the [tutorial](#) to get started.
- [Azure Web App](#). See the [how-to](#) to get started.

Create a Jenkins server on an Azure Linux VM from the Azure portal

3/12/2018 • 5 min 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

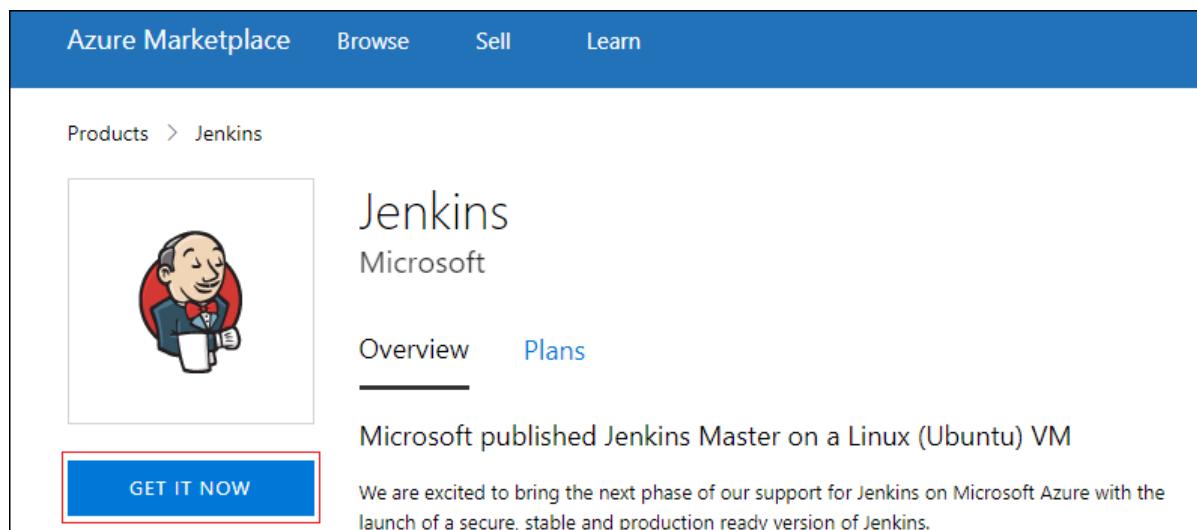
- An Azure subscription
- Access to SSH on your computer's command line (such as the Bash shell or [PuTTY](#))

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**.



The screenshot shows the Azure Marketplace page for the Jenkins solution template. At the top, there is a blue header bar with the text "Azure Marketplace" and navigation links for "Browse", "Sell", and "Learn". Below the header, the breadcrumb navigation shows "Products > Jenkins". The main content area features a thumbnail image of a Jenkins logo (a cartoon character holding a mug) and the text "Jenkins" followed by "Microsoft". There are two tabs at the bottom of this section: "Overview" and "Plans", with "Overview" being the active tab. A descriptive text below the tabs states: "Microsoft published Jenkins Master on a Linux (Ubuntu) VM". At the bottom of the main content area, there is a blue button labeled "GET IT NOW" which is highlighted with a red border. To the right of the button, there is a note: "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."

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.

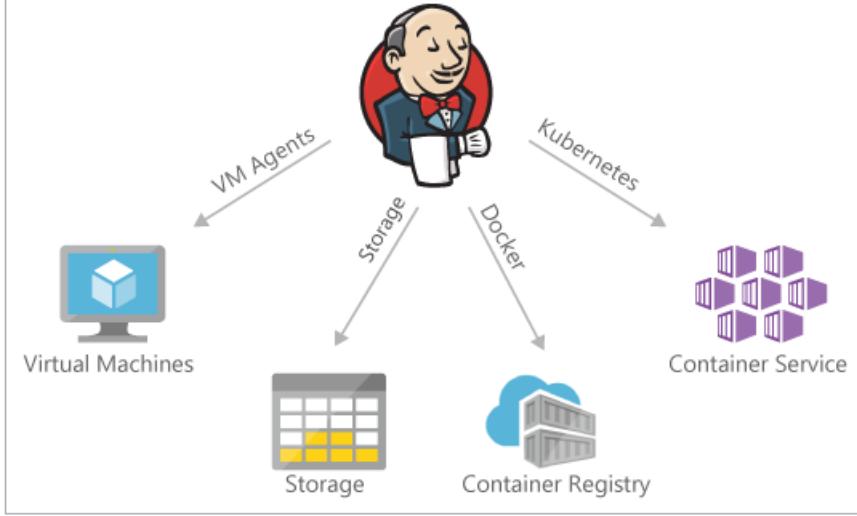
This solution template will install the latest stable Jenkins version on a Linux (Ubuntu 14.04 LTS) VM along with tools and plugins configured to work with Azure. This includes –

- 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.

[!\[\]\(333b21d00ae565d2b98e9591e64d8e14_img.jpg\)](#) [!\[\]\(921a7654d98ba956abebdfba0b592286_img.jpg\)](#) [!\[\]\(070fe9bffc6dd7be59435384e99b193c_img.jpg\)](#) [!\[\]\(900586ee44f31f7556ae1549a73e76cd_img.jpg\)](#) [!\[\]\(1ac2c9180e2bd26175851062ebcc17fb_img.jpg\)](#) [!\[\]\(63d87bc8140621f24f70f795a10c7092_img.jpg\)](#)



PUBLISHED

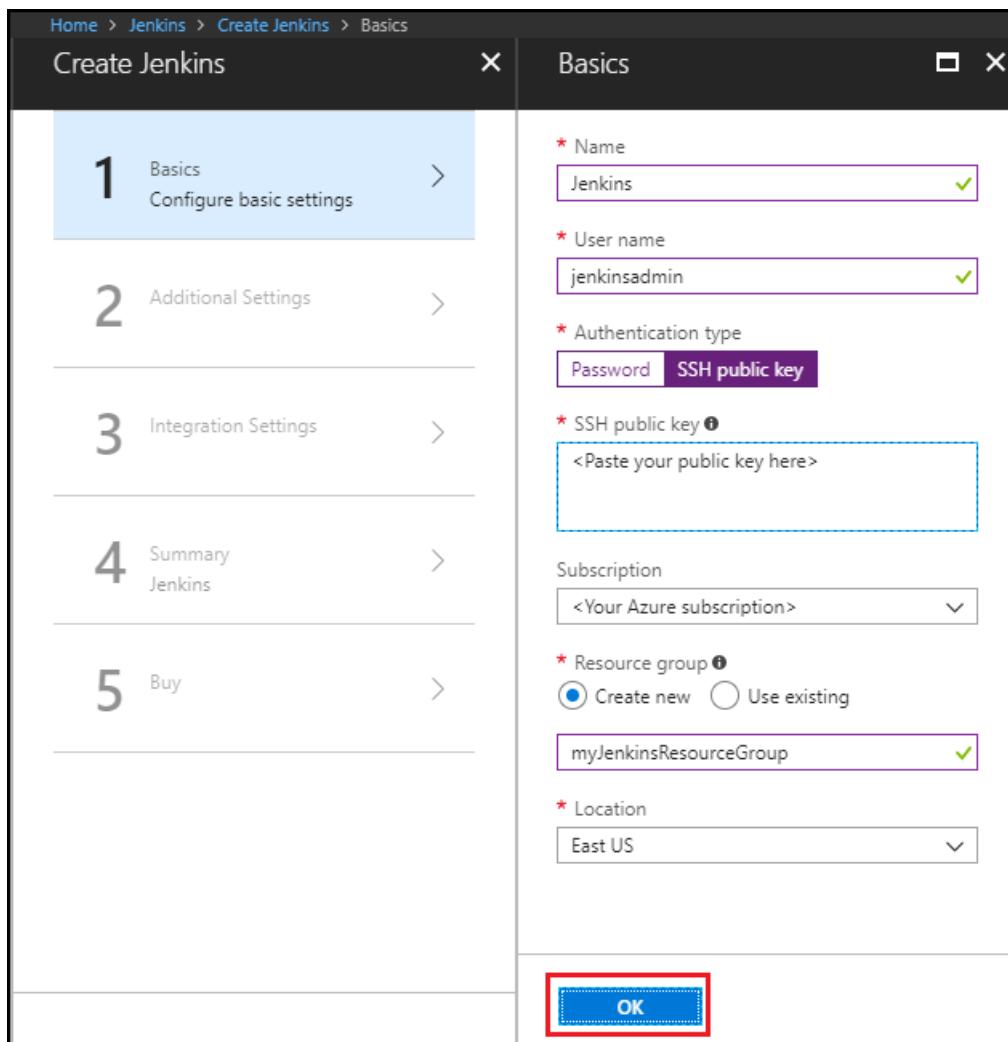
Select a deployment model 



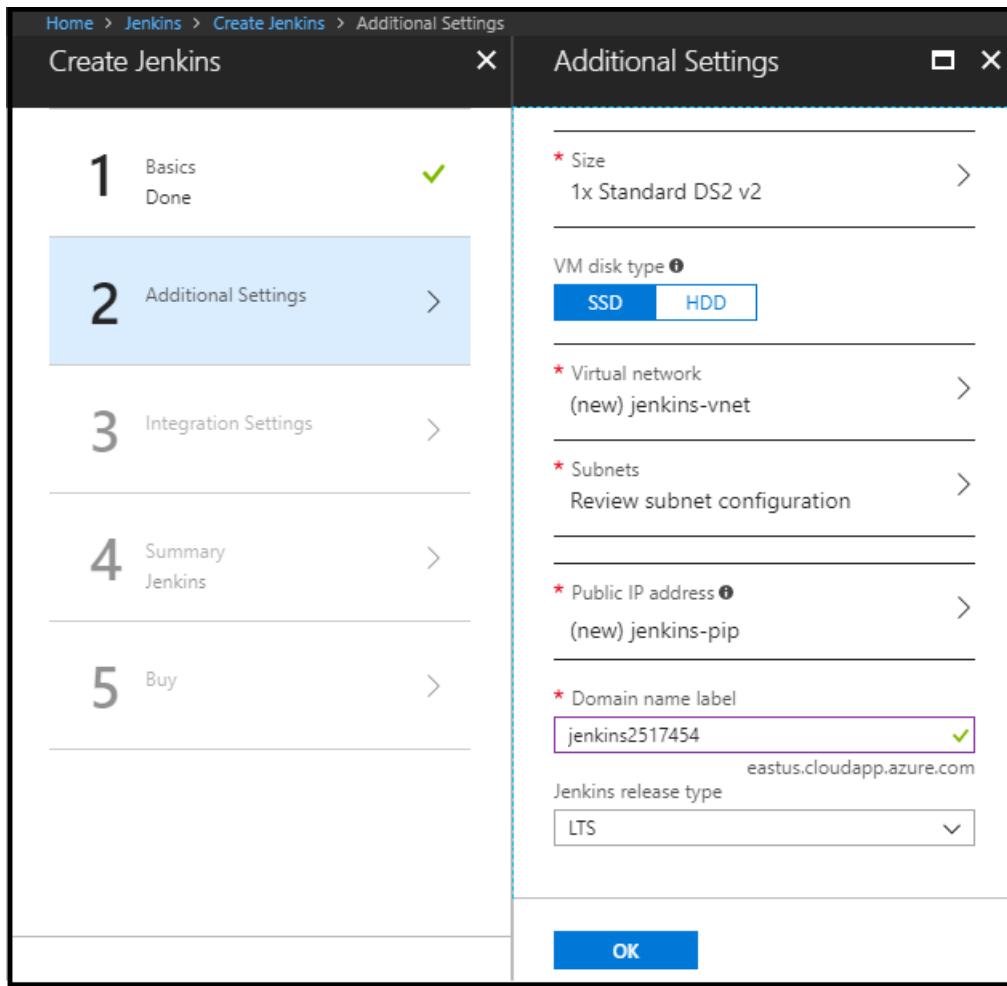
5. In the **Basics** tab, specify the following values:

- **Name** - Enter `Jenkins`.
- **User name** - Enter the user name to use when signing into 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](#).

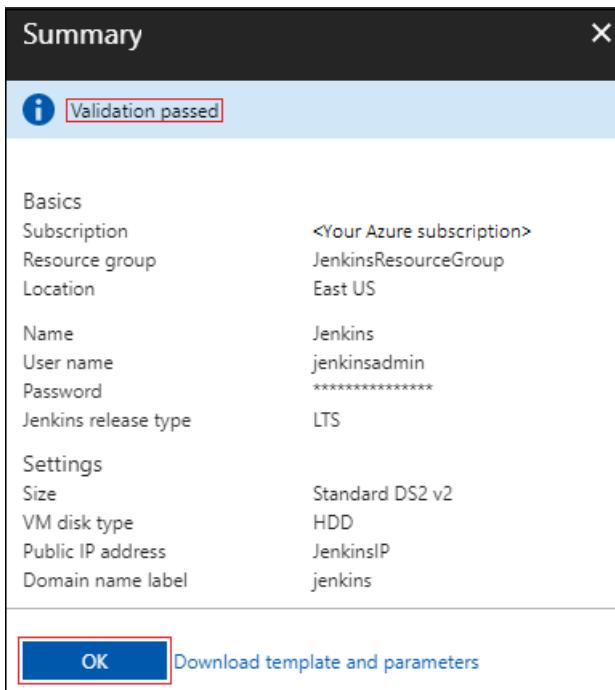
- **Subscription** - Select the Azure subscription into which you want to install Jenkins.
- **Resource group** - Select **Create new**, and enter a name for the resource group that serves as a logical container for the collection of resources that make up your Jenkins installation.
- **Location** - 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.



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**.

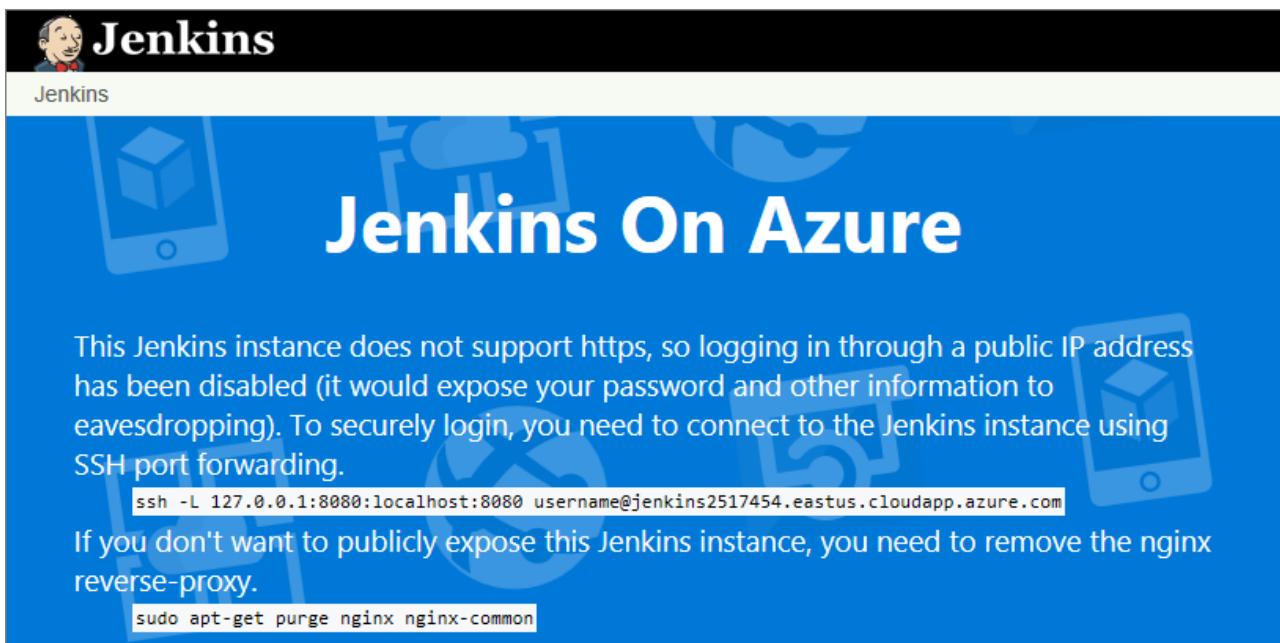


- When the **Create** tab displays, select **Create** to create the Jenkins virtual machine. When your server is ready, a notification displays in the Azure portal.



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.

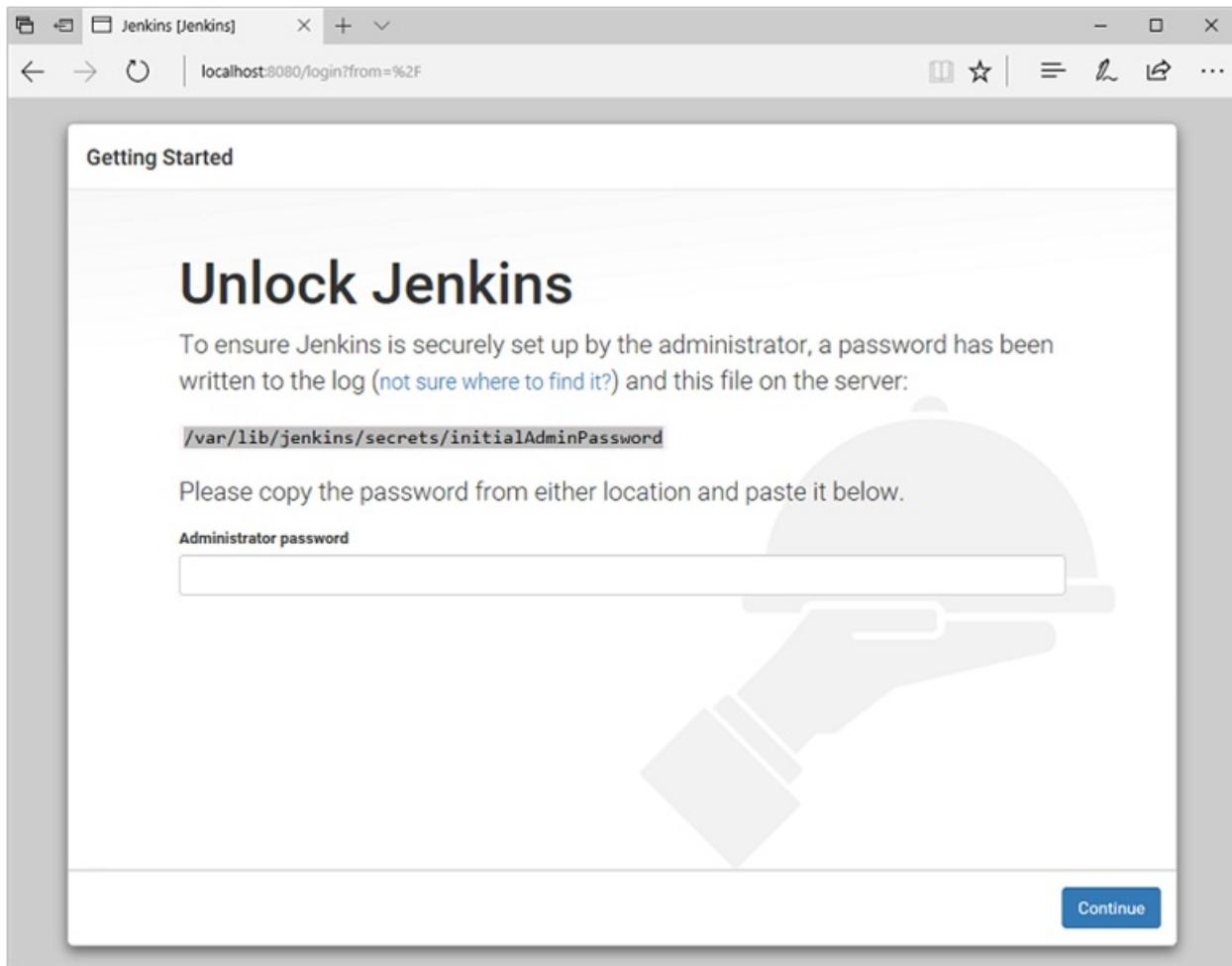
```
ssh -L 127.0.0.1:8080:localhost:8080 jenkinsadmin@jenkins2517454.eastus.cloudapp.azure.com
```

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 left sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. The main area features a 'Welcome to Jenkins!' message with a button to 'create new jobs' and a link to 'add description'. Below this are sections for 'Build Queue' (empty) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom right, it says 'Page generated: Jun 19, 2017 6:02:58 PM UTC REST API Jenkins ver. 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 dialog has a title 'Enter an item name' and a text input field containing 'mySampleApp'. A note below says '» Required field'. Below the input field, there are two options: 'Freestyle project' (selected) and 'Pipeline'. The 'Freestyle project' section includes a description: 'This is the central feature of Jenkins. Jenkins will build your project something other than software build.' The 'Pipeline' section includes a description: 'Orchestrates long-running activities that can span multiple build steps, organizing complex activities that do not easily fit in free-style jobs.'

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`. The 'Credentials' dropdown is set to '- none -' and there is a 'Add' button.

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`.

Select **Advanced..** and then enter `complete` in the **Root Build script** field. Select **Save**.

The screenshot shows the Jenkins advanced build step configuration. The 'Root Build script' field is set to `complete`.

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 project page for 'Project mySampleApp'. It features a 'Workspace' link (indicated by a red box) and a 'Recent Changes' link (indicated by a red box). Below these, there is a 'Permalinks' section with a list of recent builds:

- [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.

Next Steps

[Add Azure VMs as Jenkins agents](#)

Scale your Jenkins deployments to meet demand with Azure VM agents

2/15/2018 • 4 min 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-CCCCCCCCCC"  
}
```

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-CCCCCCCCCC",  
    "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 'Add Credentials' dialog in Jenkins. The 'Domain' section is set to 'Global credentials (unrestricted)'. The 'Kind' section is set to 'Microsoft Azure Service Principal'. The 'Scope' section is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Subscription ID' field contains the value 'AAAAAAA-AAAA-AAAA-AAAA-AAAAAAA'. The 'Client ID' field contains the value 'BBBBBBBB-BBBB-BBBB-BBBB-BBBBBBBB'. The 'Client Secret' field is redacted with '.....'. The 'Or, Certificate' section has a dropdown menu '--- Select a Certificate ---' and a 'Add' button. The 'Tenant ID' field contains the value 'CCCCCCCC-CCCC-CCCC-CCCC-CCCCCCC'. The 'Azure Environment' section is set to 'Azure'.

4. In the **Resource Group Name** section, leave **Create new** selected and enter `myJenkinsAgentGroup`.
5. 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="jnshwip6riim7wcpt1n9pfa"/>
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 log 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'. The 'Label Expression' field contains 'ubuntu'. A message below it states: 'Label ubuntu is serviced by no nodes and 1 cloud'. There are several checkboxes for project settings, with the last one, 'Restrict where this project can be run', checked. The 'Advanced...' button is visible at the bottom right.

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

Next steps

[CI/CD to Azure App Service](#)

Using Azure Storage with a Jenkins Continuous Integration solution

8/21/2017 • 9 min to read • [Edit Online](#)

Overview

The following information shows 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 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.

Overview of Jenkins

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 topic will show 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

You will need the following to use the Blob service with your Jenkins CI solution:

- 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 <http://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/>. This will 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 <http://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, click **Manage Jenkins**.
2. In the **Manage Jenkins** page, click **Manage Plugins**.
3. Click the **Available** tab.
4. In the **Artifact Uploaders** section, check **Microsoft Azure Storage plugin**.
5. Click 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, click **Manage Jenkins**.
2. In the **Manage Jenkins** page, click **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 public Azure cloud. If you are using a different Azure cloud, use the endpoint as specified in the [Azure Portal](#) for your storage account.
 - d. Click **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, click **Add more Storage Accounts**.
 - f. Click **Save** to save your settings.

How to create a post-build action that uploads your build artifacts to your storage account

For instruction purposes, first we'll 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, click **New Item**.
2. Name the job **MyJob**, click **Build a free-style software project**, and then click **OK**.
3. In the **Build** section of the job configuration, click **Add build step** and choose **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, click **Add post-build action** and choose **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. Click that link to learn the environment variable names and descriptions. Note that 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. Click **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. That is beyond the scope of this topic. You can learn more about shared access signatures at [Using Shared Access Signatures \(SAS\)](#).)
9. [Optional] Click **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. Click **Save** to save your settings.
13. In the Jenkins dashboard, click **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. Login to the [Azure Portal](#).
 - b. Click **Storage**.
 - c. Click the storage account name that you used for Jenkins.
 - d. Click **Containers**.
 - e. Click 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. Note that 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 for the **List of Artifacts to upload** value: **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 for the **List of Artifacts to upload** value: **build/*.jar::binaries;build/*.txt::notices**.

How to create a build step that downloads from Azure blob storage

The following steps show how to configure a build step to download items from Azure blob storage. This would be useful if you want to include items in your build, for example, JARs that you keep in Azure blob storage.

1. In the **Build** section of the job configuration, click **Add build step** and choose **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

The following provides an overview of the Blob service components.

- **Storage Account:** All access to Azure Storage is done through a storage account. This 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 100TB.
- **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 200GB in size. This tutorial uses block blobs. Page blobs, another blob type, can be up to 1TB 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 public 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 in 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
```

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](#).

Create a Docker environment in Azure using the Docker VM extension

4/9/2018 • 3 min to read • [Edit Online](#)

Docker is a popular container management and imaging platform that allows you to quickly work with containers on Linux. In Azure, there are various ways you can deploy Docker according to your needs. This article focuses on using the Docker VM extension and Azure Resource Manager templates with the Azure CLI 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

WARNING

The Azure Docker VM extension for Linux is deprecated and will be retired November 2018. The extension merely installs Docker, so alternatives such as cloud-init or the Custom Script Extension are a better way to install the Docker version of choice. For more information on how to use cloud-init, see [Customize a Linux VM with cloud-init](#).

Azure Docker VM extension overview

The Azure Docker VM extension installs and configures the Docker daemon, Docker client, and Docker Compose in your Linux virtual machine (VM). By using the Azure Docker VM extension, you have more control and features than simply using Docker Machine or creating the Docker host yourself. These additional features, such as [Docker Compose](#), make the Azure Docker VM extension suited for more robust developer or production environments.

For more information about the different deployment methods, including using Docker Machine and Azure Container Services, see the following articles:

- To quickly prototype an app, you can create a single Docker host using [Docker Machine](#).
- To build production-ready, scalable environments that provide additional scheduling and management tools, you can deploy a [Kubernetes](#) or [Docker Swarm](#) cluster on Azure Container Services.

Deploy a template with the Azure Docker VM extension

Let's use an existing quickstart template to create an Ubuntu VM that uses the Azure Docker VM extension to install and configure the Docker host. You can view the template here: [Simple deployment of an Ubuntu VM with Docker](#). You need the latest [Azure CLI 2.0](#) 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 a VM with [az group deployment create](#) that includes the Azure Docker VM extension from [this Azure Resource Manager template on GitHub](#). When prompted, provide your own unique values for *newStorageAccountName*, *adminUsername*, *adminPassword*, and *dnsNameForPublicIP*:

```
az group deployment create --resource-group myResourceGroup \
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/docker-simple-on-
ubuntu/azuredeploy.json
```

It takes a few minutes for the deployment to finish.

Deploy your first NGINX container

To view details of your VM, including the DNS name, use [az vm show](#):

```
az vm show \
--resource-group myResourceGroup \
--name myDockerVM \
--show-details \
--query [fqdns] \
--output tsv
```

SSH to your new Docker host. Provide your own username and DNS name from the preceding steps:

```
ssh azureuser@mypublicdns.eastus.cloudapp.azure.com
```

Once logged in to the Docker host, let's run an NGINX container:

```
sudo docker run -d -p 80:80 nginx
```

The output is similar to the following example as the NGINX image is downloaded and a container started:

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
efd26ecc9548: Pull complete
a3ed95caeb02: Pull complete
a48df1751a97: Pull complete
8ddc2d7beb91: Pull complete
Digest: sha256:2ca2638e55319b7bc0c7d028209ea69b1368e95b01383e66dfe7e4f43780926d
Status: Downloaded newer image for nginx:latest
b6ed109fb743a762ff21a4606dd38d3e5d35aff43fa7f12e8d4ed1d920b0cd74
```

Check the status of the containers running on your Docker host as follows:

```
sudo docker ps
```

The output is similar to the following example, showing that the NGINX container is running and TCP ports 80 and 443 and being forwarded:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
b6ed109fb743 0.0.0.0:80->80/tcp, 443/tcp	nginx adoring_payne	"nginx -g 'daemon off'"	About a minute ago	Up About a minute

To see your container in action, open up a web browser and enter the DNS name of your Docker host:



Azure Docker VM extension template reference

The previous example uses an existing quickstart template. You can also deploy the Azure Docker VM extension with your own Resource Manager templates. To do so, add the following to your Resource Manager templates, defining the `vmName` of your VM appropriately:

```
{
  "type": "Microsoft.Compute/virtualMachines/extensions",
  "name": "[concat(variables('vmName'), '/DockerExtension')]",
  "apiVersion": "2015-05-01-preview",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[concat('Microsoft.Compute/virtualMachines/', variables('vmName'))]"
  ],
  "properties": {
    "publisher": "Microsoft.Azure.Extensions",
    "type": "DockerExtension",
    "typeHandlerVersion": "1.*",
    "autoUpgradeMinorVersion": true,
    "settings": {},
    "protectedSettings": {}
  }
}
```

You can find more detailed walkthrough on using Resource Manager templates by reading [Azure Resource Manager overview](#).

Next steps

You may wish to [configure the Docker daemon TCP port](#), understand [Docker security](#), or deploy containers using [Docker Compose](#). For more information on the Azure Docker VM Extension itself, see the [GitHub project](#).

Read more information about the additional Docker deployment options in Azure:

- [Use Docker Machine with the Azure driver](#)
- [Get Started with Docker and Compose to define and run a multi-container application on an Azure virtual machine.](#)
- [Deploy an Azure Container Service cluster](#)

How to use Docker Machine to create hosts in Azure

2/6/2018 • 3 min 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_D2_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="vhdski0hvfaazyd8mn991cg50" 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, lets 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 festive_mirzakhani

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

You can also create hosts with the [Docker VM Extension](#). 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

3/8/2018 • 4 min 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.

Set up a Linux VM as a Docker host

You can use various Azure procedures and available images or Resource Manager templates in the Azure Marketplace to create a Linux VM and set it up as a Docker host. For example, see [Using the Docker VM Extension to deploy your environment](#) to quickly create an Ubuntu VM with the Azure Docker VM extension by using a [quickstart template](#).

When you use the Docker VM extension, your VM is automatically set up as a Docker host and Compose is already installed.

Create Docker host with Azure CLI 2.0

Install the latest [Azure CLI 2.0](#) 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 myResourceGroup --location eastus
```

Next, deploy a VM with `az group deployment create` that includes the Azure Docker VM extension from [this Azure Resource Manager template on GitHub](#). When prompted, provide your own unique values for *newStorageAccountName*, *adminUsername*, *adminPassword*, and *dnsNameForPublicIP*:

```
az group deployment create --resource-group myResourceGroup \
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/docker-simple-on-
ubuntu/azuredeploy.json
```

It takes a few minutes for the deployment to finish.

Verify that Compose is installed

To view details of your VM, including the DNS name, use `az vm show`:

```
az vm show \
--resource-group myResourceGroup \
--name myDockerVM \
--show-details \
--query [fqdns] \
--output tsv
```

SSH to your new Docker host. Provide your own username and DNS name from the preceding steps:

```
ssh azureuser@mypublicdns.eastus.cloudapp.azure.com
```

To check that Compose is installed on the VM, run the following command:

```
docker-compose --version
```

You see output similar to *docker-compose 1.6.2, build 4d72027*.

TIP

If you used another method to create a Docker host and need to install Compose yourself, see the [Compose documentation](#).

Create a docker-compose.yml configuration file

Next you create a `docker-compose.yml` file, which is just a text configuration file, to define the Docker containers to run on the VM. The file specifies the image to run on each container (or it could be a build from a Dockerfile), 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` as follows:

```
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`):

```
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 see output similar to the following example:

```
Creating wordpress_db_1...
Creating wordpress_wordpress_1...
...
```

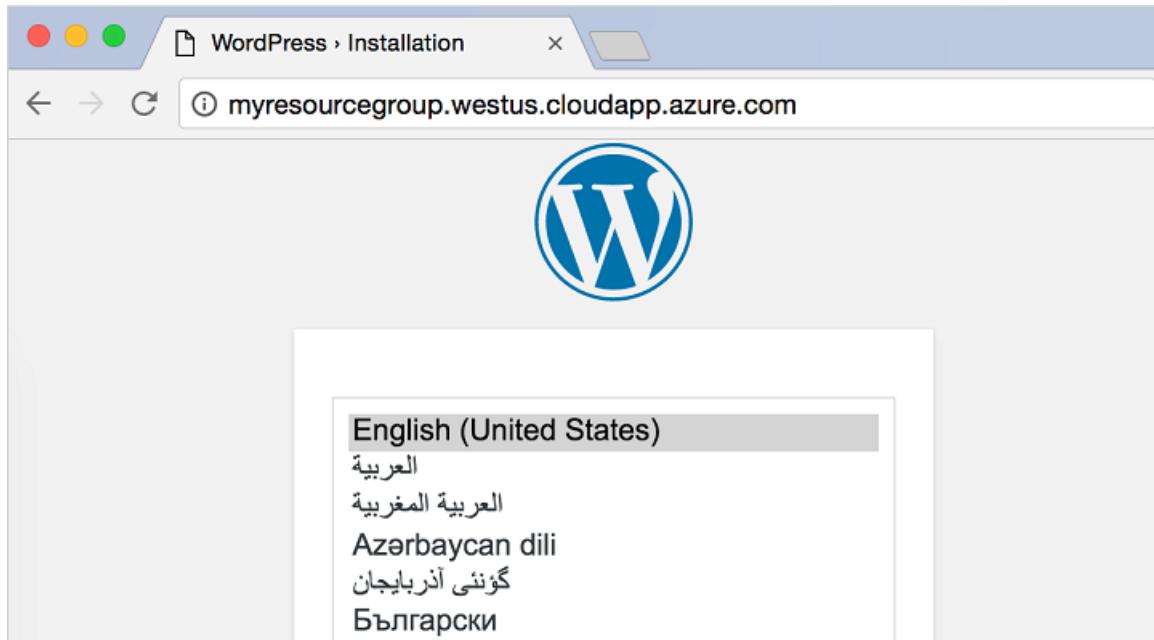
NOTE

Be sure to use the `-d` option on start-up so that the containers run in the background continuously.

To verify that the containers are up, type `docker-compose ps`. You should see something like:

Name	Command	State	Ports
azureuser_db_1	docker-entrypoint.sh mysql	Up	3306/tcp
azureuser_wordpress_1	docker-entrypoint.sh apache ...	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 DNS name of your VM (such as <http://mypublicdns.eastus.cloudapp.azure.com>). You should now see the WordPress start screen, where you can complete the installation and get started with the application.



Next steps

- Go to the [Docker VM extension user guide](#) for more options to configure Docker and Compose in your Docker VM. For example, one option is to put the Compose yml file (converted to JSON) directly in the configuration of the Docker VM extension.
- 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

4/25/2018 • 2 min 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

Visual Studio Team 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 Visual Studio Team Services (VSTS) to manage your projects and would like to set up a CI/CD pipeline targeting Cloud Foundry, you can use the [VSTS 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

4/9/2018 • 4 min 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
```

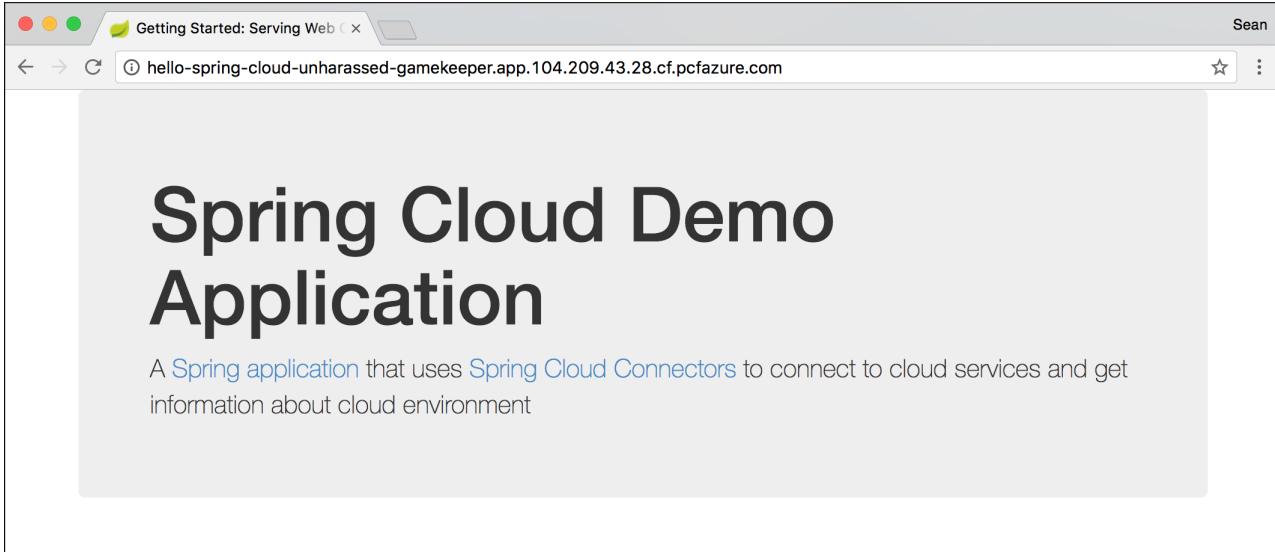
When you *push* an application, Cloud Foundry detects the type of application (in this case, a Java app) and identifies its dependencies (in this case, the Spring framework). It then packages everything required to run your code into a standalone container image, known as a *droplet*. Finally, Cloud 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 Visual Studio Team Services plugin for Cloud Foundry](#)
- [Configure the Microsoft Log Analytics Nozzle for Cloud Foundry](#)

OpenShift in Azure

5/8/2018 • 1 min 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- 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:

- OpenShift Origin
- OpenShift Container Platform
- OpenShift Online
- OpenShift Dedicated

Of the four versions covered in this article, only two are available for customers to deploy in Azure: OpenShift Origin and OpenShift Container Platform.

OpenShift Origin

Origin is an [open-source](#) upstream project of OpenShift that's community supported. Origin can be installed on CentOS or Red Hat Enterprise Linux (RHEL).

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, AWS, or Google).

OpenShift Online

Online is a Red Hat-managed *multi-tenant* OpenShift that uses Container Platform. Red Hat manages all of the underlying infrastructure (such as VMs, OpenShift cluster, networking, and storage).

With this version, the customer deploys containers but has no control over which hosts the containers run. Because Online is multi-tenant, containers may be located on the same VM hosts as containers from other customers. Cost is per container.

OpenShift Dedicated

Dedicated is a Red Hat-managed *single-tenant* OpenShift that uses Container Platform. Red Hat manages all of the underlying infrastructure (VMs, OpenShift cluster, networking, storage, etc.). The cluster is specific to one customer and runs in a public cloud (such as AWS or Google, with Azure coming in early 2018). A starting cluster includes four application nodes for \$48,000 per year (paid up front).

Next steps

- [Configure common prerequisites for OpenShift in Azure](#)
- [Deploy OpenShift Origin in Azure](#)
- [Deploy OpenShift Container Platform in Azure](#)
- [Post-deployment tasks](#)
- [Troubleshoot OpenShift deployment](#)

Common prerequisites for deploying OpenShift in Azure

3/8/2018 • 4 min to read • [Edit Online](#)

This article describes common prerequisites for deploying OpenShift Origin or OpenShift Container Platform 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 you initiate the SSH connection to the remote hosts, you cannot enter a password. For this reason, the private key cannot have a password 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. You need to inject the corresponding private key into the VM that executes all the playbooks as well. To do this securely, you use an Azure key vault 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 this capability, but Azure must first be configured as the cloud provider.

In this model, OpenShift:

- Creates a VHD object in an Azure Storage account.
- Mounts the VHD to a VM and format the volume.
- Mounts the volume to the pod.

For this configuration to work, OpenShift needs permissions to perform the previous tasks in Azure. You achieve this with a service principal. 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 Solution 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 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.

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 Origin 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 cannot have a password.

For more information on SSH keys on Windows, see [How to create SSH keys on Windows](#).

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. To improve security beyond just providing a username and password, this example creates a basic service principal.

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 *myResourceGroup*. If you're using Windows, execute [az group show --name myResourceGroup --query id](#) separately and use the output to feed the *--scopes* option.

```
az ad sp create-for-rbac --name openshiftsp \
    --role Contributor --password {Strong Password} \
    --scopes $(az group show --name myResourceGroup --query id)
```

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 2.0](#).

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 Origin](#)
- [Deploy OpenShift Container Platform](#)

Deploy OpenShift Origin in Azure

2/6/2018 • 2 min to read • [Edit Online](#)

You can use one of two ways to deploy OpenShift Origin in Azure:

- You can manually deploy all the necessary Azure infrastructure components, and then follow the OpenShift Origin [documentation](#).
- You can also use an existing [Resource Manager template](#) that simplifies the deployment of the OpenShift Origin cluster.

Deploy by using the OpenShift Origin template

Use the `appId` value from the service principal that you created earlier for the `aadClientId` parameter.

The following example creates a parameters file named `azuredeploy.parameters.json` with all the required inputs.

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "masterVmSize": {
            "value": "Standard_E2s_v3"
        },
        "infraVmSize": {
            "value": "Standard_E2s_v3"
        },
        "nodeVmSize": {
            "value": "Standard_E2s_v3"
        },
        "openshiftClusterPrefix": {
            "value": "mycluster"
        },
        "masterInstanceCount": {
            "value": 3
        },
        "infraInstanceCount": {
            "value": 2
        },
        "nodeInstanceCount": {
            "value": 2
        },
        "dataDiskSize": {
            "value": 128
        },
        "adminUsername": {
            "value": "clusteradmin"
        },
        "openshiftPassword": {
            "value": "{Strong Password}"
        },
        "sshPublicKey": {
            "value": "{SSH Public Key}"
        },
        "keyVaultResourceGroup": {
            "value": "keyvaultrg"
        },
        "keyVaultName": {
            "value": "keyvault"
        },
        "keyVaultSecret": {
            "value": "keysecret"
        },
        "aadClientId": {
            "value": "11111111-abcd-1234-efgh-111111111111"
        },
        "aadClientSecret": {
            "value": "{Strong Password}"
        },
        "defaultSubDomainType": {
            "value": "nipio"
        }
    }
}
```

Deploy by 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 2.0](#).

The following example deploys the OpenShift cluster and all related resources into a resource group named myResourceGroup, with a deployment name of myOpenShiftCluster. The template is referenced directly from the GitHub repo by using a local parameters file named azuredeploy.parameters.json.

```
az group deployment create -g myResourceGroup --name myOpenShiftCluster \
--template-uri https://raw.githubusercontent.com/Microsoft/openshift-origin/master/azuredeploy.json \
--parameters ./azuredeploy.parameters.json
```

The deployment takes at least 25 minutes to finish, depending on the total number of nodes deployed. The URL of the OpenShift console and the DNS name of the OpenShift master prints to the terminal when the deployment finishes.

```
{
  "OpenShift Console Uri": "http://openshiftlb.cloudapp.azure.com:8443/console",
  "OpenShift Master SSH": "ssh clusteradmin@myopenshiftmaster.cloudapp.azure.com -p 2200"
}
```

Connect to the OpenShift cluster

When the deployment finishes, connect to the OpenShift console with your browser by using the [OpenShift Console Uri](#). Alternatively, you can connect to the OpenShift master by using the following command:

```
$ ssh -p 2200 clusteradmin@myopenshiftmaster.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 myResourceGroup
```

Next steps

- [Post-deployment tasks](#)
- [Troubleshoot OpenShift deployment](#)
- [Getting started with OpenShift Origin](#)

Deploy OpenShift Container Platform in Azure

2/6/2018 • 4 min 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. Ensure that you have a valid Red Hat Subscription Manager (RHSM) username, password, and Pool ID. You can verify this information by signing in to <https://access.redhat.com>.

Deploy by using the OpenShift Container Platform Resource Manager template

To deploy by using the Resource Manager template, you use a parameters file to supply the input parameters. To customize any of the deployment items that are not covered by using input parameters, fork the GitHub repo and change the appropriate items.

Some common customization options include, but are not limited to:

- Virtual network CIDR (variable in azuredeploy.json)
- 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

Use the `appId` value from the service principal you created earlier for the `aadClientId` parameter.

The following example creates a parameters file named azuredeploy.parameters.json with all the required inputs.

```
{  
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "masterVmSize": {  
            "value": "Standard_E2s_v3"  
        },  
        "infraVmSize": {  
            "value": "Standard_E2s_v3"  
        },  
        "nodeVmSize": {  
            "value": "Standard_E2s_v3"  
        },  
        "openshiftClusterPrefix": {  
            "value": "mycluster"  
        },  
        "masterInstanceCount": {  
            "value": 3  
        },  
        "infraInstanceCount": {  
            "value": 3  
        }  
    }  
}
```

```

        "value": 2
    },
    "nodeInstanceCount": {
        "value": 2
    },
    "dataDiskSize": {
        "value": 128
    },
    "adminUsername": {
        "value": "clusteradmin"
    },
    "openshiftPassword": {
        "value": "{Strong Password}"
    },
    "enableMetrics": {
        "value": "true"
    },
    "enableLogging": {
        "value": "true"
    },
    "enableCockpit": {
        "value": "false"
    },
    "rhsmUsernamePasswordOrActivationKey": {
        "value": "usernamepassword"
    },
    "rhsmUsernameOrOrgId": {
        "value": "{RHSM Username}"
    },
    "rhsmPasswordOrActivationKey": {
        "value": "{RHSM Password}"
    },
    "rhsmPoolId": {
        "value": "{Pool ID}"
    },
    "sshPublicKey": {
        "value": "{SSH Public Key}"
    },
    "keyVaultResourceGroup": {
        "value": "keyvaultrg"
    },
    "keyVaultName": {
        "value": "keyvault"
    },
    "keyVaultSecret": {
        "value": "keysecret"
    },
    "enableAzure": {
        "value": "true"
    },
    "aadClientId": {
        "value": "11111111-abcd-1234-efgh-111111111111"
    },
    "aadClientSecret": {
        "value": "{Strong Password}"
    },
    "defaultSubDomainType": {
        "value": "nipio"
    }
}
}

```

Replace the items enclosed in brackets with your specific information.

Deploy by 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 2.0](#).

The following example deploys the OpenShift cluster and all related resources into a resource group named `myResourceGroup`, 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 myResourceGroup --name myOpenShiftCluster \
    --template-uri https://raw.githubusercontent.com/Microsoft/openshift-container-
    platform/master/azuredeploy.json \
    --parameters ./azuredeploy.parameters.json
```

The deployment takes at least 30 minutes to complete, depending on the total number of nodes deployed. The URL of the OpenShift console and the DNS name of the OpenShift master prints to the terminal when the deployment finishes.

```
{
  "OpenShift Console Uri": "http://openshiftlb.cloudapp.azure.com:8443/console",
  "OpenShift Master SSH": "ssh clusteradmin@myopenshiftmaster.cloudapp.azure.com -p 2200"
}
```

Deploy by using the OpenShift Container Platform Azure Marketplace offer

The simplest way to deploy OpenShift Container Platform into Azure is to use the [Azure Marketplace offer](#).

This is the simplest option, but it also has limited customization capabilities. The offer includes three configuration options:

- **Small:** Deploys a non-high availability (HA) cluster with one master node, one infrastructure node, two application nodes, and one bastion node. All nodes are standard DS2v2 VM sizes. This cluster requires 10 total cores and is ideal for small-scale testing.
- **Medium:** Deploys an HA cluster with three master nodes, two infrastructure nodes, four application nodes, and one bastion node. All nodes except the bastion node are standard DS3v2 VM sizes. The bastion node is a standard DS2v2. This cluster requires 38 cores.
- **Large:** Deploys an HA cluster with three master nodes, two infrastructure nodes, six application nodes, and one bastion node. The master and infrastructure nodes are standard DS3v2 VM sizes. The application nodes are standard DS4v2 VM sizes, and the bastion node is a standard DS2v2. This cluster requires 70 cores.

Configuration of Azure Cloud Solution Provider is optional for medium and large cluster sizes. The small cluster size does not give an option for configuring Azure Cloud Solution Provider.

Connect to the OpenShift cluster

When the deployment finishes, connect to the OpenShift console with your browser by using the `OpenShift Console Uri`. Alternatively, you can connect to the OpenShift master by using the following command:

```
$ ssh clusteradmin@myopenshiftmaster.cloudapp.azure.com -p 2200
```

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 myResourceGroup
```

Next steps

- [Post-deployment tasks](#)
- [Troubleshoot OpenShift deployment in Azure](#)
- [Getting started with OpenShift Container Platform](#)

Post-deployment tasks

4/20/2018 • 5 min to read • [Edit Online](#)

After you deploy an OpenShift cluster, you can configure additional items. This article covers the following:

- How to configure single sign-on by using Azure Active Directory (Azure AD)
- How to configure Log Analytics to monitor OpenShift
- How to configure metrics and logging

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:8443/console>)
- Identifier URI: OpenShift console URL (for example, <https://masterdns343khhde.westus.cloudapp.azure.com:8443/console>)
- Reply URL: Master public URL and the app registration name (for example, <https://masterdns343khhde.westus.cloudapp.azure.com:8443/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:8443/console --reply-urls https://masterdns343khhde.westus.cloudapp.azure.com:8443/oauth2callback/hwocpadint --identifier-uris https://masterdns343khhde.westus.cloudapp.azure.com:8443/console --password {Strong Password}
```

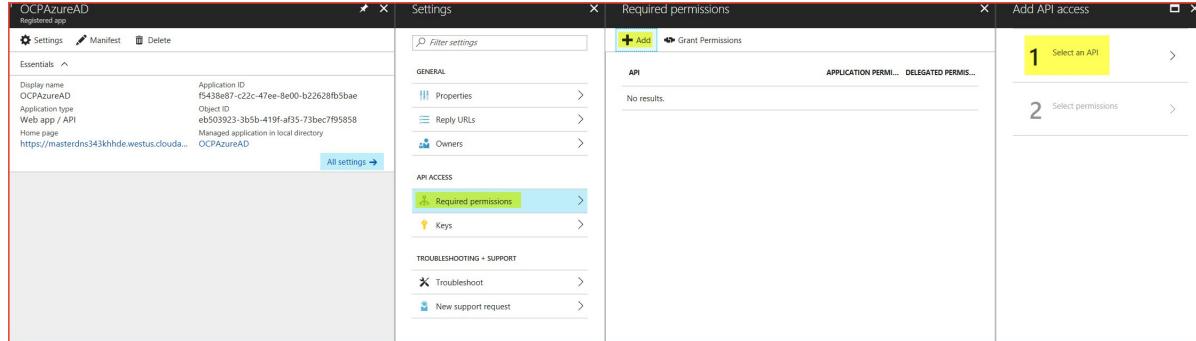
If the command is successful, you get a JSON output similar to:

```
{  
  "appId": "12345678-ca3c-427b-9a04-ab12345cd678",  
  "appPermissions": null,  
  "availableToOtherTenants": false,  
  "displayName": "OCPAzureAD",  
  "homepage": "https://masterdns343khhde.westus.cloudapp.azure.com:8443/console",  
  "identifierUris": [  
    "https://masterdns343khhde.westus.cloudapp.azure.com:8443/console"  
  ],  
  "objectId": "62cd74c9-42bb-4b9f-b2b5-b6ee88991c80",  
  "objectType": "Application",  
  "replyUrls": [  
    "https://masterdns343khhde.westus.cloudapp.azure.com:8443/oauth2callback/OCPAzureAD"  
  ]  
}
```

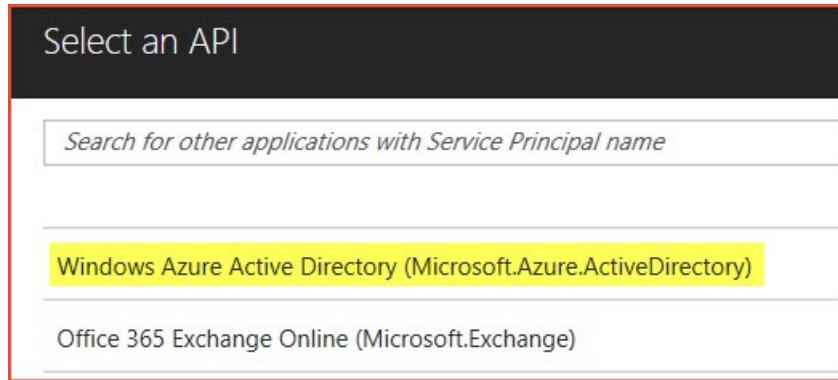
Take note of the appId property returned from the command for a later step.

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**.



6. Click Step 1: Select API, and then click **Windows Azure Active Directory (Microsoft.Azure.ActiveDirectory)**. Click **Select** at the bottom.



7. On Step 2: Select Permissions, select **Sign in and read user profile** under **Delegated Permissions**, and then click **Select**.

Enable Access

X

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
<input checked="" type="checkbox"/> 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:8443/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
```

Find the tenant ID by using the following CLI command: `az account show`

Restart the OpenShift master services on all master nodes:

OpenShift Origin

```
sudo systemctl restart origin-master-api
sudo systemctl restart origin-master-controllers
```

OpenShift Container Platform (OCP) with multiple masters

```
sudo systemctl restart atomic-openshift-master-api
sudo systemctl restart atomic-openshift-master-controllers
```

OpenShift Container Platform with a single master

```
sudo systemctl restart atomic-openshift-master
```

In the OpenShift console, you now see two options for authentication: `htpasswd_auth` and [App Registration].

Monitor OpenShift with Log Analytics

To monitor OpenShift with Log Analytics, you can use one of two options: OMS Agent installation on VM host, or OMS Container. This article provides instructions for deploying the OMS Container.

Create an OpenShift project for Log Analytics and set user access

```
oadm new-project omslogging --node-selector='zone=default'  
oc project omslogging  
oc create serviceaccount omsagent  
oadm policy add-cluster-role-to-user cluster-reader system:serviceaccount:omslogging:omsagent  
oadm policy add-scc-to-user privileged system:serviceaccount:omslogging:omsagent
```

Create a daemon set yaml file

Create a file named ocp-omsagent.yml:

```

apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: oms
spec:
  selector:
    matchLabels:
      name: omsagent
  template:
    metadata:
      labels:
        name: omsagent
        agentVersion: 1.4.0-45
        dockerProviderVersion: 10.0.0-25
    spec:
      nodeSelector:
        zone: default
      serviceAccount: omsagent
      containers:
        - image: "microsoft/oms"
          imagePullPolicy: Always
          name: omsagent
          securityContext:
            privileged: true
          ports:
            - containerPort: 25225
              protocol: TCP
            - containerPort: 25224
              protocol: UDP
          volumeMounts:
            - mountPath: /var/run/docker.sock
              name: docker-sock
            - mountPath: /etc/omsagent-secret
              name: omsagent-secret
              readOnly: true
      livenessProbe:
        exec:
          command:
            - /bin/bash
            - -c
            - ps -ef | grep omsagent | grep -v "grep"
      initialDelaySeconds: 60
      periodSeconds: 60
    volumes:
      - name: docker-sock
        hostPath:
          path: /var/run/docker.sock
      - name: omsagent-secret
        secret:
          secretName: omsagent-secret

```

Create a secret yaml file

To create the secret yaml file, you need two pieces of information: Log Analytics Workspace ID and Log Analytics Workspace Shared Key.

A sample ocp-secret.yaml file follows:

```
apiVersion: v1
kind: Secret
metadata:
  name: omsagent-secret
data:
  WSID: wsid_data
  KEY: key_data
```

Replace wsid_data with the Base64 encoded Log Analytics Workspace ID. Then replace key_data with the Base64-encoded Log Analytics Workspace Shared Key.

```
wsid_data='11111111-abcd-1111-abcd-111111111111'
key_data='My Strong Password'
echo $wsid_data | base64 | tr -d '\n'
echo $key_data | base64 | tr -d '\n'
```

Create the secret and daemon set

Deploy the secret file:

```
oc create -f ocp-secret.yml
```

Deploy the OMS Agent daemon set:

```
oc create -f ocp-omsagent.yml
```

Configure metrics and logging

The Azure Resource Manager template for OpenShift Container Platform provides input parameters for enabling metrics and logging. The OpenShift Container Platform Marketplace offer and the OpenShift Origin Resource Manager template do not.

If you used the OCP Resource Manager template and metrics and logging weren't enabled at installation time, or if you used the OCP Marketplace offer, you can easily enable these after the fact. If you're using the OpenShift Origin Resource Manager template, some pre-work is required.

OpenShift Origin template pre-work

1. SSH to the first master node by using port 2200.

Example:

```
ssh -p 2200 clusteradmin@masterdnsixpdkehd3h.eastus.cloudapp.azure.com
```

2. Edit the /etc/ansible/hosts file and add the following lines after the Identity Provider Section (# Enable HTPasswdPasswordIdentityProvider):

```

# Setup metrics
openshift_hosted_metrics_deploy=false
openshift_metrics_cassandra_storage_type=dynamic
openshift_metrics_start_cluster=true
openshift_metrics_hawkular_nodeselector={"type":"infra"}
openshift_metrics_cassandra_nodeselector={"type":"infra"}
openshift_metrics_heapster_nodeselector={"type":"infra"}
openshift_hosted_metrics_public_url=https://metrics.$ROUTING/hawkular/metrics

# Setup logging
openshift_hosted_logging_deploy=false
openshift_hosted_logging_storage_kind=dynamic
openshift_logging_fluentd_nodeselector={"logging": "true"}
openshift_logging_es_nodeselector={"type": "infra"}
openshift_logging_kibana_nodeselector={"type": "infra"}
openshift_logging_curator_nodeselector={"type": "infra"}
openshift_master_logging_public_url=https://kibana.$ROUTING

```

- Replace \$ROUTING with the string used for the openshift_master_default_subdomain option in the same /etc/ansible/hosts file.

Azure Cloud Provider in use

On the first master node (Origin) or bastion node (OCP), SSH by using the credentials provided during deployment. Issue the following command:

```

ansible-playbook $HOME/openshift-ansible/playbooks/byo/openshift-cluster/openshift-metrics.yml \
-e openshift_metrics_install_metrics=True \
-e openshift_metrics_cassandra_storage_type=dynamic

ansible-playbook $HOME/openshift-ansible/playbooks/byo/openshift-cluster/openshift-logging.yml \
-e openshift_logging_install_logging=True \
-e openshift_hosted_logging_storage_kind=dynamic

```

Azure Cloud Provider not in use

On the first master node (Origin) or bastion node (OCP), SSH by using the credentials provided during deployment. Issue the following command:

```

ansible-playbook $HOME/openshift-ansible/playbooks/byo/openshift-cluster/openshift-metrics.yml \
-e openshift_metrics_install_metrics=True

ansible-playbook $HOME/openshift-ansible/playbooks/byo/openshift-cluster/openshift-logging.yml \
-e openshift_logging_install_logging=True

```

Next steps

- [Getting started with OpenShift Container Platform](#)
- [Getting started with OpenShift Origin](#)

Troubleshoot OpenShift deployment in Azure

11/10/2017 • 1 min to read • [Edit Online](#)

If your OpenShift cluster does not deploy successfully, try these troubleshooting tasks to narrow down the issue. View the deployment status and compare against the following list of 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
- Exit code 6: OpenShift Cluster installation failed
- Exit code 7: OpenShift Cluster installation succeeded but Azure Cloud Solution Provider configuration failed - master config on Master Node issue
- Exit code 8: OpenShift Cluster installation succeeded but Azure Cloud Solution Provider configuration failed - node config on Master Node issue
- Exit code 9: OpenShift Cluster installation succeeded but Azure Cloud Solution Provider configuration failed - node config on Infra or App Node issue
- Exit code 10: OpenShift Cluster installation succeeded but Azure Cloud Solution Provider configuration failed - correcting Master Nodes or not able to set Master as unschedulable
- Exit code 11: Metrics failed to deploy
- Exit code 12: Logging failed to deploy

For exit codes 7-10, the OpenShift cluster was installed, but the Azure Cloud Solution Provider configuration failed. You can SSH to the master node (OpenShift Origin) or the bastion node (OpenShift Container Platform), and from there SSH to each cluster node to fix the issues.

A common cause for the failures with exit codes 7-9 is that the service principal did not have proper permissions to the subscription or the resource group. If this is the issue, assign the correct permissions and manually rerun the script that failed and all subsequent scripts.

Be sure to restart the service that failed (for example, `systemctl restart atomic-openshift-node.service`) before executing the scripts again.

For further troubleshooting, SSH into your master node on port 2200 (Origin) or the bastion node on port 22 (Container Platform). You need to be in the root (`sudo su -`) and then browse to the following directory:
`/var/lib/waagent/custom-script/download`.

Here you see folders named "0" and "1." In each of these folders, you see two files, "stderr" and "stdout." Look through these files to determine where the failure occurred.

Using Azure for hosting and running SAP workload scenarios

4/24/2018 • 10 min to read • [Edit Online](#)

By choosing Microsoft Azure as your SAP ready cloud partner, you are able to reliably run your mission critical SAP workloads and scenarios on a scalable, compliant, and enterprise-proven platform. Get the scalability, flexibility, and cost savings of Azure. With the expanded partnership between Microsoft and SAP, you can run SAP applications across dev/test and production scenarios in Azure - and be fully supported. From SAP NetWeaver to SAP S4/HANA, SAP BI, Linux to Windows, SAP HANA to SQL, we have you covered.

Besides hosting SAP NetWeaver scenarios with the different DBMS on Azure, you can host different other SAP workload scenarios, like SAP BI on Azure. Documentation regarding SAP NetWeaver deployments on Azure native Virtual Machines can be found in the section "SAP NetWeaver on Azure Virtual Machines."

Azure has native Azure Virtual Machine offers that are ever growing in size of CPU and memory resources to cover SAP workload that leverages SAP HANA. For more information on this area, look up the documents under the section SAP HANA on Azure Virtual Machines."

The uniqueness of Azure for SAP HANA is a unique offer that sets Azure apart from competition. In order to enable hosting more memory and CPU resource demanding SAP scenarios involving SAP HANA, Azure offers the usage of customer dedicated bare-metal hardware for the purpose of running SAP HANA deployments that require up to 20 TB (60 TB scale-out) of memory for S/4HANA or other SAP HANA workload. This unique Azure solution of SAP HANA on Azure (Large Instances) allows you to run SAP HANA on the dedicated bare-metal hardware with the SAP application layer or workload middle-ware layer hosted in native Azure Virtual Machines. This solution is documented in several documents in the section "SAP HANA on Azure (Large Instances)."

Hosting SAP workload scenarios in Azure also can create requirements of Identity integration and Single-Sign-On using Azure Activity Directory to different SAP components and SAP SaaS or PaaS offers. A list of such integration and Single-Sign-On scenarios with Azure Active Directory (AAD) and SAP entities is described and documented in the section "AAD SAP Identity Integration and Single-Sign-On."

SAP HANA on SAP HANA on Azure (Large Instances)

Overview and architecture of SAP HANA on Azure (Large Instances)

Title: Overview and Architecture of SAP HANA on Azure (Large Instances)

Summary: This Architecture and Technical Deployment Guide provides information to help you deploy SAP on the new SAP HANA on Azure (Large Instances) in Azure. It is not intended to be a comprehensive guide covering specific setup of SAP solutions, but rather useful information in your initial deployment and ongoing operations. It should not replace SAP documentation related to the installation of SAP HANA (or the many SAP Support Notes that cover the area). It gives you an overview and provides the additional detail of installing SAP HANA on Azure (Large Instances).

Updated: October 2017

[This guide can be found here](#)

Infrastructure and connectivity to SAP HANA on Azure (Large Instances)

Title: Infrastructure and Connectivity to SAP HANA on Azure (Large Instances)

Summary: After the purchase of SAP HANA on Azure (Large Instances) is finalized between you and the Microsoft

enterprise account team, various network configurations are required in order to ensure proper connectivity. This document outlines the information that has to be shared with the following information is required. This document outlines what information has to be collected and what configuration scripts have to be run.

Updated: October 2017

[This guide can be found here](#)

Install SAP HANA in SAP HANA on Azure (Large Instances)

Title: Install SAP HANA on SAP HANA on Azure (Large Instances)

Summary: This document outlines the setup procedures for installing SAP HANA on your Azure Large Instance.

Updated: July 2017

[This guide can be found here](#)

High availability and disaster recovery of SAP HANA on Azure (Large Instances)

Title: High Availability and Disaster Recovery of SAP HANA on Azure (Large Instances)

Summary: High Availability (HA) and Disaster Recovery (DR) are important aspects of running your mission-critical SAP HANA on Azure (Large Instances) server(s). It's import to work with SAP, your system integrator, and/or Microsoft to properly architect and implement the right HA/DR strategy for you. Important considerations like Recovery Point Objective (RPO) and Recovery Time Objective (RTO), specific to your environment, must be considered. This document explains your options for enabling your preferred level of HA and DR.

Updated: October 2017

[This document can be found here](#)

Troubleshooting and monitoring of SAP HANA on Azure (Large Instances)

Title: Troubleshooting and Monitoring of SAP HANA on Azure (Large Instances)

Summary: This guide covers information that is useful in establishing monitoring of your SAP HANA on Azure environment as well as additional troubleshooting information.

Updated: October 2017

[This document can be found here](#)

SAP HANA on Azure Virtual Machines

Getting started with SAP HANA on Azure

Title: Quickstart guide for manual installation of SAP HANA on Azure VMs

Summary: This quickstart guide helps to set up a single-instance SAP HANA system on Azure VMs by a manual installation of SAP NetWeaver 7.5 and SAP HANA SP12. The guide presumes that the reader is familiar with Azure IaaS basics like how to deploy virtual machines or virtual networks either via the Azure portal or Powershell/CLI including the option to use json templates. Furthermore it's expected that the reader is familiar with SAP HANA, SAP NetWeaver and how to install it on-premises.

Updated: June 2017

[This guide can be found here](#)

S/4HANA SAP CAL deployment on Azure

Title: Deploy SAP S/4HANA or BW/4HANA on Azure

Summary: This guide helps to demonstrate the deployment of SAP S/4HANA on Azure using SAP Cloud

Appliance Library. SAP Cloud Appliance Library is a service by SAP that allows to deploy SAP applications on Azure. The guide describes step by step the deployment.

Updated: June 2017

[This guide can be found here](#)

High Availability of SAP HANA in Azure Virtual Machines

Title: High Availability of SAP HANA on Azure Virtual Machines

Summary: This guide leads you through the high availability configuration of the SUSE 12 OS and SAP HANA to accommodate HANA System replication with automatic failover. The guide is specific for SUSE and Azure Virtual Machines. The guide does not apply yet for Red Hat or bare-metal or private cloud or other non-Azure public cloud deployments.

Updated: June 2017

[This guide can be found here](#)

SAP HANA backup overview on Azure VMs

Title: Backup guide for SAP HANA on Azure Virtual Machines

Summary: This guide provides basic information about backup possibilities running SAP HANA on Azure Virtual Machines.

Updated: March 2017

[This guide can be found here](#)

SAP HANA file level backup on Azure VMs

Title: SAP HANA backup based on storage snapshots

Summary: This guide provides information about using snapshot-based backups on Azure VMs when running SAP HANA on Azure Virtual Machines.

Updated: March 2017

[This guide can be found here](#)

SAP HANA snapshot based backups on Azure VMs

Title: SAP HANA Azure Backup on file level

Summary: This guide provides information about using SAP HANA file level backup running SAP HANA on Azure Virtual Machines

Updated: March 2017

[This guide can be found here](#)

SAP NetWeaver deployed on Azure Virtual Machines

Deploy SAP IDES system on Windows and SQL Server through SAP CAL on Azure

Title: Testing SAP NetWeaver on Microsoft Azure SUSE Linux VMs

Summary: This document describes the deployment of an SAP IDES system based on Windows and SQL Server on Azure using SAP Cloud Appliance Library. SAP Cloud appliance Library is an SAP service that allows the deployment of SAP products on Azure. This document goes step by step through the deployment of an SAP IDES system. The IDES system is just an example for several other dozen applications that can be deployed through SAP Cloud appliance on Microsoft Azure.

Updated: June 2017

[This guide can be found here](#)

Quickstart guide for NetWeaver on SUSE Linux on Azure

Title: Testing SAP NetWeaver on Microsoft Azure SUSE Linux VMs

Summary: This article describes various things to consider when you're running SAP NetWeaver on Microsoft Azure SUSE Linux virtual machines (VMs). SAP NetWeaver is officially supported on SUSE Linux VMs on Azure. All details regarding Linux versions, SAP kernel versions, and other details can be found in SAP Note 1928533 "SAP Applications on Azure: Supported Products and Azure VM types".

Updated: September 2016

[This guide can be found here](#)

Planning and implementation

Title: Azure Virtual Machines planning and implementation for SAP NetWeaver

Summary: This document is the guide to start with if you are thinking about running SAP NetWeaver in Azure Virtual Machines. This planning and implementation guide helps you evaluate whether an existing or planned SAP NetWeaver-based system can be deployed to an Azure Virtual Machines environment. It covers multiple SAP NetWeaver deployment scenarios, and includes SAP configurations that are specific to Azure. The paper lists and describes all the necessary configuration information you need on the SAP/Azure side to run a hybrid SAP landscape. Measures you can take to ensure high availability of SAP NetWeaver-based systems on IaaS are also covered.

Updated: June 2017

[This guide can be found here](#)

High Availability configurations of SAP NetWeaver in Azure VMs

Title: Azure Virtual Machines High Availability for SAP NetWeaver

Summary: In this document, we cover the steps that you can take to deploy high-availability SAP systems in Azure by using the Azure Resource Manager deployment model. We walk you through these major tasks. In the document, we describe how single-point-of-failure components like Advanced Business Application Programming (ABAP) SAP Central Services (ASCS)/SAP Central Services (SCS) and database management systems (DBMS), and redundant components like SAP Application Server are going to be protected when running in Azure VMs. A step-by-step example of an installation and configuration of a high-availability SAP system in a Windows Server Failover Clustering cluster and SUSE Linux Enterprise Server Cluster Framework in Azure is demonstrated and shown in this document.

Updated: October 2017

[This guide can be found here](#)

Realizing Multi-SID deployments of SAP NetWeaver in Azure VMs

Title: Create an SAP NetWeaver multi-SID configuration

Summary: This document is an addition to the document High availability for SAP NetWeaver on Azure VMs. Due to new functionality in Azure that got introduced in September 2016, it is possible to deploy multiple SAP NetWeaver ASCS/SCS instances in a pair of Azure VMs. With such a configuration, you can reduce the number of VMs necessary to deploy to realize highly available SAP NetWeaver configurations. The guide describes the setup of such multi-SID configurations.

Updated: December 2016

[This guide can be found here](#)

Deployment of SAP NetWeaver in Azure VMs

Title: Azure Virtual Machines deployment for SAP NetWeaver

Summary: This document provides procedural guidance for deploying SAP NetWeaver software to virtual machines in Azure. This paper focuses on three specific deployment scenarios, with an emphasis on enabling the Azure Monitoring Extensions for SAP, including troubleshooting recommendations for the Azure Monitoring Extensions for SAP. This paper assumes that you've read the planning and implementation guide.

Updated: June 2017

[This guide can be found here](#)

DBMS deployment guide

Title: Azure Virtual Machines DBMS deployment for SAP NetWeaver

Summary: This paper covers planning and implementation considerations for the DBMS systems that should run in conjunction with SAP. In the first part, general considerations are listed and presented. The following parts of the paper relate to deployments of different DBMS in Azure that are supported by SAP. Different DBMS presented are SQL Server, SAP ASE, and Oracle. In those specific parts, considerations you have to account for when you are running SAP systems on Azure in conjunction with those DBMS are discussed. Subjects like backup and high availability methods that are supported by the different DBMS on Azure are presented for the usage with SAP applications.

Updated: June 2017

[This guide can be found here](#)

Using Azure Site Recovery for SAP workload

Title: SAP NetWeaver: Building a Disaster Recovery Solution with Azure Site Recovery

Summary: This document describes the way how Azure Site Recovery services can be used for the purpose of handling disaster recovery scenarios. Cases where Azure is used as disaster recovery location for an on-premise SAP landscape using Azure Site Recovery Services. Another scenario described in the document is the Azure-to-Azure (A2A) disaster recovery case and how it is managed with Azure Site Recovery.

Updated: August 2017

[This guide can be found here](#)

Create an Oracle Database in an Azure VM

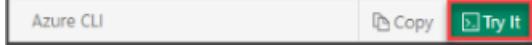
2/6/2018 • 6 min to read • [Edit Online](#)

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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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 <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
```

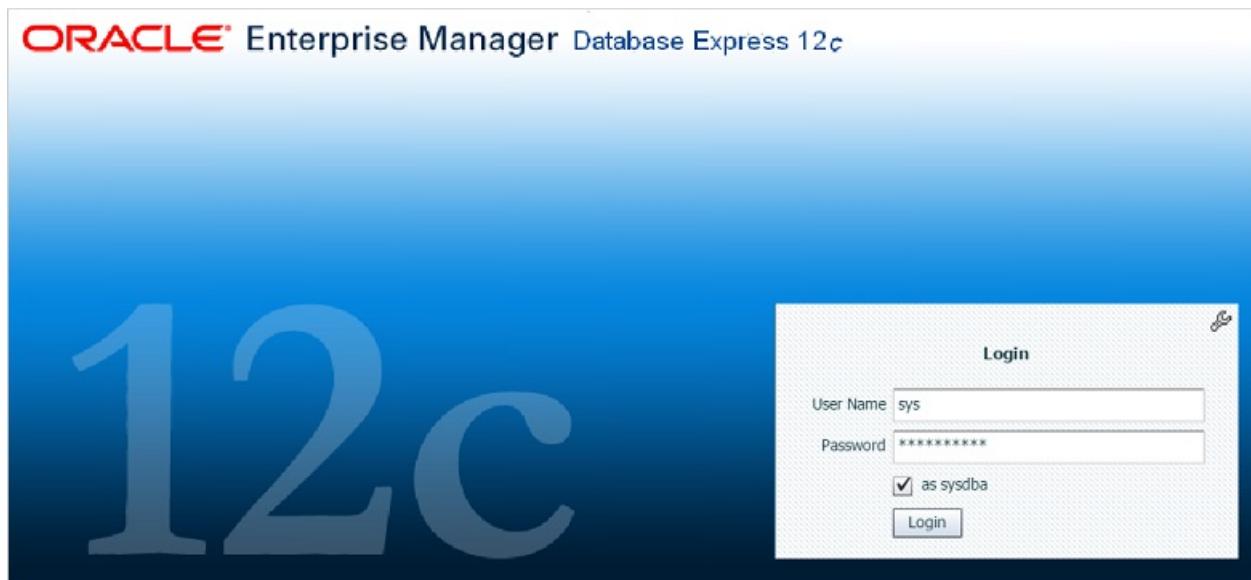
3. If needed, obtain the public IP address of your VM again with [az network public-ip show](#) as follows:

```
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

4/9/2018 • 5 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

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.

The screenshot shows the Kibana management interface. On the left, there's a sidebar with icons for Discover, Visualize, Dashboard, Timelion, Dev Tools, and Management. The main area is titled 'Management / Kibana' and shows an index pattern named 'logstash-*'. Below it, there's a section titled 'Configure an index pattern' with instructions about index patterns. It includes fields for 'Index pattern' (set to 'logstash-*'), 'Time Filter field name' (set to '@timestamp'), and several checkboxes for deprecated options like 'Expand index pattern when searching' and 'Use event times to create index names'. At the bottom right is a large blue 'Create' button, which is also highlighted with a red box.

Select **Discover** on the Kibana console to search, browse, and filter through the syslog events.

The screenshot shows the Kibana Discover interface. The sidebar on the left has the 'Discover' icon highlighted. The main area shows a search bar with the query 'message: "warn"'. Below the search bar, there's a histogram chart showing the count of events over time, with a single bar at 14:05:00. The table below the chart lists three log entries, each with a timestamp, version, message, and type. The first entry is expanded, showing a warning message about the 'include_in_all' field being deprecated. The other two entries are collapsed.

Time	@timestamp	@version	message	type
October 13th 2017, 14:05:25.688	October 13th 2017, 14:05:25.688	1	[2017-10-13T21:05:24,910] [WARN] [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] [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] [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

4/9/2018 • 2 min to read • [Edit Online](#)

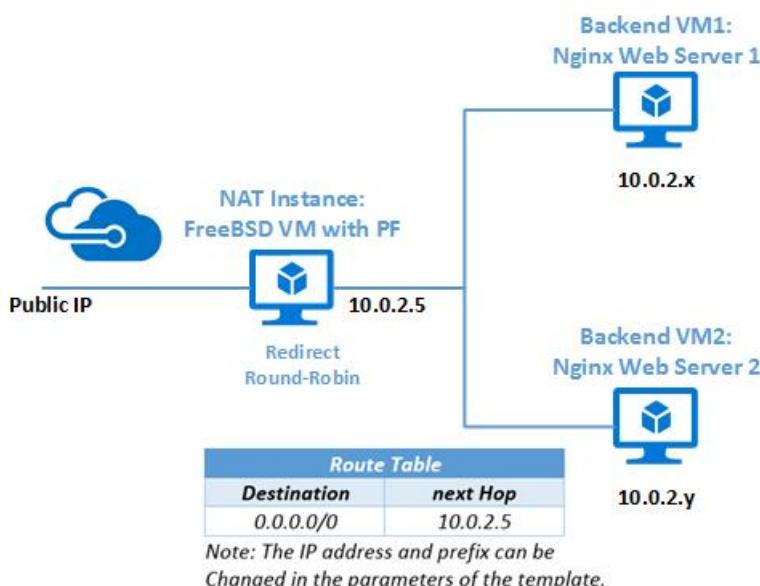
This article introduces how to deploy a NAT firewall using FreeBSD's Packer Filter through Azure Resource Manager template for common web server scenario.

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 2.0](#) 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](#).

How to install MySQL on Azure

4/9/2018 • 3 min to read • [Edit Online](#)

In this article, you will learn how to install and configure MySQL 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 MySQL on your virtual machine

NOTE

You must already have a Microsoft Azure virtual machine running Linux in order to complete this tutorial. Please see the [Azure Linux VM tutorial](#) to create and set up a Linux VM with `mysqlnode` as the VM name and `azureuser` as user before proceeding.

In this case, use 3306 port as the MySQL port.

Connect to the Linux VM you created via putty. If this is the first time you use Azure Linux VM, see how to use putty connect to a Linux VM [here](#).

We will use repository package to install MySQL5.6 as an example in this article. Actually, MySQL5.6 has more improvement in performance than MySQL5.5. More information [here](#).

How to install MySQL5.6 on Ubuntu

We will use Linux VM with Ubuntu from Azure here.

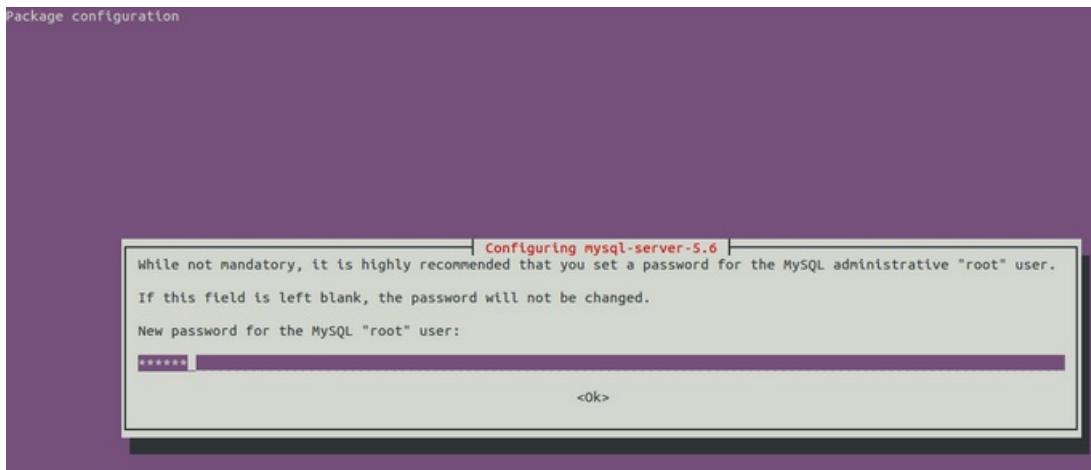
- Step 1: Install MySQL Server 5.6 Switch to `root` user:

```
# [azureuser@mysqlnode:~]sudo su -
```

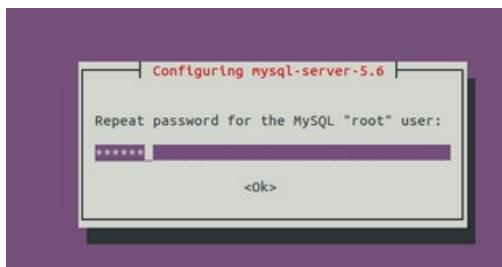
Install mysql-server 5.6:

```
# [root@mysqlnode ~]# apt-get update  
# [root@mysqlnode ~]# apt-get -y install mysql-server-5.6
```

During installation, you will see a dialog window poping up to ask you to set MySQL root password below, and you need set the password here.



Input the password again to confirm.



- Step 2: Login MySQL Server

When MySQL server installation finished, MySQL service will be started automatically. You can login MySQL Server with `root` user. Use the below command to login and input password.

```
# [root@mysqlnode ~]# mysql -uroot -p
```

- Step 3: Manage the running MySQL service

- (a) Get status of MySQL service

```
# [root@mysqlnode ~]# service mysql status
```

- (b) Start MySQL Service

```
# [root@mysqlnode ~]# service mysql start
```

- (c) Stop MySQL service

```
# [root@mysqlnode ~]# service mysql stop
```

- (d) Restart the MySQL service

```
# [root@mysqlnode ~]# service mysql restart
```

How to install MySQL on Red Hat OS family like CentOS, Oracle Linux

We will use Linux VM with CentOS or Oracle Linux here.

- Step 1: Add the MySQL Yum repository Switch to `root` user:

```
#[azureuser@mysqlnode:~]sudo su -
```

Download and install the MySQL release package:

```
#[root@mysqlnode ~]# wget http://repo.mysql.com/mysql-community-release-el6-5.noarch.rpm  
#[root@mysqlnode ~]# yum localinstall -y mysql-community-release-el6-5.noarch.rpm
```

- Step 2: Edit below file to enable the MySQL repository for downloading the MySQL5.6 package.

```
# [root@mysqlnode ~]# vim /etc/yum.repos.d/mysql-community.repo
```

Update each value of this file to below:

```
\# *Enable to use MySQL 5.6*  
  
[mysql56-community]  
name=MySQL 5.6 Community Server  
  
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/6/$basearch/  
  
enabled=1  
  
gpgcheck=1  
  
gpgkey=file:/etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

- Step 3: Install MySQL from MySQL repository Install MySQL:

```
# [root@mysqlnode ~]#yum install mysql-community-server
```

MySQL RPM package and all related packages will be installed.

- Step 4: Manage the running MySQL service

(a) Check the service status of the MySQL server:

```
# [root@mysqlnode ~]#service mysqld status
```

(b) Check whether the default port of MySQL server is running:

```
# [root@mysqlnode ~]#netstat -tunlp|grep 3306
```

(c) Start the MySQL server:

```
# [root@mysqlnode ~]#service mysqld start
```

(d) Stop the MySQL server:

```
# [root@mysqlnode ~]#service mysqld stop
```

(e) Set MySQL to start when the system boot-up:

```
# [root@mysqlnode ~]# chkconfig mysqld on
```

How to install MySQL on SUSE Linux

We will use Linux VM with OpenSUSE here.

- Step 1: Download and install MySQL Server

Switch to `root` user through below command:

```
#sudo su -
```

Download and install MySQL package:

```
# [root@mysqlnode ~]# zypper update  
#[root@mysqlnode ~]# zypper install mysql-server mysql-devel mysql
```

- Step 2: Manage the running MySQL service

(a) Check the status of the MySQL server:

```
# [root@mysqlnode ~]# rcmysql status
```

(b) Check whether the default port of the MySQL server:

```
# [root@mysqlnode ~]# netstat -tunlp|grep 3306
```

(c) Start the MySQL server:

```
# [root@mysqlnode ~]# rcmysql start
```

(d) Stop the MySQL server:

```
# [root@mysqlnode ~]# rcmysql stop
```

(e) Set MySQL to start when the system boot-up:

```
# [root@mysqlnode ~]# insserv mysql
```

Next Step

Find more usage and information regarding MySQL [Here](#).

Install MySQL on a virtual machine running OpenSUSE Linux in Azure

1/22/2018 • 3 min 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 a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open 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 2.0](#).

Create a virtual machine running OpenSUSE Linux

First, create a resource group. In this example, we are naming the resource group *mySQLSUSEResourceGroup* and creating it in the *East US* region.

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

Create the VM. In this example, we are naming the VM *myVM*. We are also going to use a VM size *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.

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 logins, remove test databases, and reload the privileges table.

```
mysql_secure_installation
```

Log in to MySQL

You can now log 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 log in from another computer. In this example, the IP address of the computer that we want to log 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

3/8/2018 • 5 min 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 2.0. You can also perform these steps with the [Azure CLI 1.0](#). 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 2.0](#) 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/azuredeploy.json>](https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/mongodb-on-centos/azuredeploy.json)

To create this environment, you need the latest [Azure CLI 2.0](#) 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 2.0](#) 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

4/9/2018 • 5 min 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/pgsql19.3.5 /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 --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
```

LISTEN
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 she is attending, so we will change her 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](#).

Options with HPC Pack to create and manage a cluster for Linux HPC workloads in Azure

4/9/2018 • 1 min to read • [Edit Online](#)

Create and manage a cloud-based high-performance computing (HPC) cluster by taking advantage of [Microsoft HPC Pack](#) and Azure compute and infrastructure services. HPC Pack, available for free download, is built on Microsoft Azure and Windows Server technologies and supports a wide range of HPC workloads.

For more HPC options in Azure, see [Technical resources for batch and high-performance computing](#).

This article focuses on options to use HPC Pack to run Linux workloads. There are also options for running [Windows HPC workloads with HPC Pack](#).

HPC Pack cluster in Azure VMs and VM scale sets

Azure templates

- ([GitHub](#)) [HPC Pack 2016 cluster templates](#)
- ([GitHub](#)) [HPC Pack 2012 R2 cluster templates](#)
- ([Marketplace](#)) [HPC Pack cluster for Linux workloads](#)
- ([Quickstart](#)) [Create an HPC cluster with Linux compute nodes](#)

PowerShell deployment script for HPC Pack 2012 R2

- [Create a Linux HPC cluster with the HPC Pack IaaS deployment script](#)

Tutorials

- [Tutorial: Get started with Linux compute nodes in an HPC Pack cluster in Azure](#)
- [Tutorial: Run NAMD with Microsoft HPC Pack on Linux compute nodes in Azure](#)
- [Tutorial: Run OpenFOAM with Microsoft HPC Pack on a Linux RDMA cluster in Azure](#)
- [Tutorial: Run STAR-CCM+ with Microsoft HPC Pack on a Linux RDMA cluster in Azure](#)

Cluster management

- [Submit jobs to an HPC Pack cluster in Azure](#)
- [Job management in HPC Pack](#)

RDMA clusters for MPI workloads

- [Tutorial: Run OpenFOAM with Microsoft HPC Pack on a Linux RDMA cluster in Azure](#)
- [Set up a Linux RDMA cluster to run MPI applications](#)

Run NAMD with Microsoft HPC Pack on Linux compute nodes in Azure

4/9/2018 • 12 min to read • [Edit Online](#)

This article shows you one way to run a Linux high-performance computing (HPC) workload on Azure virtual machines. Here, you set up a [Microsoft HPC Pack](#) cluster on Azure with Linux compute nodes and run a [NAMD](#) simulation to calculate and visualize the structure of a large biomolecular system.

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.

- **NAMD** (for Nanoscale Molecular Dynamics program) is a parallel molecular dynamics package designed for high-performance simulation of large biomolecular systems containing up to millions of atoms. Examples of these systems include viruses, cell structures, and large proteins. NAMD scales to hundreds of cores for typical simulations and to more than 500,000 cores for the largest simulations.
- **Microsoft HPC Pack** provides features to run large-scale HPC and parallel applications in clusters of on-premises computers or Azure virtual machines. Originally developed as a solution for Windows HPC workloads, HPC Pack now supports running Linux HPC applications on Linux compute node VMs deployed in an HPC Pack cluster. See [Get started with Linux compute nodes in an HPC Pack cluster in Azure](#) for an introduction.

Prerequisites

- **HPC Pack cluster with Linux compute nodes** - Deploy an HPC Pack cluster with Linux compute nodes on Azure using either an [Azure Resource Manager template](#) or an [Azure PowerShell script](#). See [Get started with Linux compute nodes in an HPC Pack cluster in Azure](#) for the prerequisites and steps for either option. If you choose the PowerShell script deployment option, see the sample configuration file in the sample files at the end of this article. This file configures an Azure-based HPC Pack cluster consisting of a Windows Server 2012 R2 head node and four size Large CentOS 6.6 compute nodes. Customize this file as needed for your environment.
- **NAMD software and tutorial files** - Download NAMD software for Linux from the [NAMD](#) site (registration required). This article is based on NAMD version 2.10, and uses the [Linux-x86_64 \(64-bit Intel/AMD with Ethernet\)](#) archive. Also download the [NAMD tutorial files](#). The downloads are .tar files, and you need a Windows tool to extract the files on the cluster head node. To extract the files, follow the instructions later in this article.
- **VMD** (optional) - To see the results of your NAMD job, download and install the molecular visualization program [VMD](#) on a computer of your choice. The current version is 1.9.2. See the VMD download site to get started.

Set up mutual trust between compute nodes

Running a cross-node job on multiple Linux nodes requires the nodes to trust each other (by **rsh** or **ssh**). When you create the HPC Pack cluster with the Microsoft HPC Pack IaaS deployment script, the script automatically sets up permanent mutual trust for the administrator account you specify. For non-administrator users you create in the cluster's domain, you have to set up temporary mutual trust among the nodes when a job is allocated to them. Then, destroy the relationship after the job is complete. To do this for each user, provide an RSA key pair to the cluster which HPC Pack uses to establish the trust relationship. Instructions follow.

Generate an RSA key pair

It's easy to generate an RSA key pair, which contains a public key and a private key, by running the Linux **ssh-keygen** command.

1. Log on to a Linux computer.
2. Run the following command:

```
ssh-keygen -t rsa
```

NOTE

Press **Enter** to use the default settings until the command is completed. Do not enter a passphrase here; when prompted for a password, just press **Enter**.

```
[hpclabsa@CentOS66LN-00 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hpclabsa/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hpclabsa/.ssh/id_rsa.
Your public key has been saved in /home/hpclabsa/.ssh/id_rsa.pub.
The key fingerprint is:
e6:18:08:51:9a:e0:f5:f8:85:7e:5e:0e:ae:c6:d0:8c hpclabsa@CentOS66LN-00
The key's randomart image is:
+--[ RSA 2048]--+
| ..o.          |
| ...+o .        |
| .+. o .       |
| .o..          |
| =o.oS.         |
| E o+=+         |
| o .o..         |
| o.            |
| ..            |
+-----+
```

3. Change directory to the `~/ssh` directory. The private key is stored in `id_rsa` and the public key in `id_rsa.pub`.

```
[hpclabsa@CentOS66LN-00 ~]$ cd ~/ssh
[hpclabsa@CentOS66LN-00 .ssh]$ ll
total 12
-rw-------. 1 hpclabsa hpclabsa 1675 May 20 07:48 id_rsa
-rw-r--r--. 1 hpclabsa hpclabsa  404 May 20 07:48 id_rsa.pub
-rw-r--r--. 1 hpclabsa hpclabsa  788 May 11 03:11 known_hosts
```

Add the key pair to the HPC Pack cluster

1. [Connect by Remote Desktop](#) to the head node VM using the domain credentials you provided when you deployed the cluster (for example, `hpc\clusteradmin`). You manage the cluster from the head node.
2. Use standard Windows Server procedures to create a domain user account in the cluster's Active Directory domain. For example, use the Active Directory User and Computers tool on the head node. The examples in this article assume you create a domain user named `hpcuser` in the `hpclab` domain (`hpclab\hpcuser`).
3. Add the domain user to the HPC Pack cluster as a cluster user. For instructions, see [Add or remove cluster users](#).
4. Create a file named `C:\cred.xml` and copy the RSA key data into it. You can find an example in the sample files at the end of this article.

```
<ExtendedData>
  <PrivateKey>Copy the contents of private key here</PrivateKey>
  <PublicKey>Copy the contents of public key here</PublicKey>
</ExtendedData>
```

5. Open a Command Prompt and enter the following command to set the credentials data for the

hpclab\hpcuser account. You use the **extendeddata** parameter to pass the name of the C:\cred.xml file you created for the key data.

```
hpccred setcreds /extendeddata:c:\cred.xml /user:hpclab\hpcuser /password:<UserPassword>
```

This command completes successfully without output. After setting the credentials for the user accounts you need to run jobs, store the cred.xml file in a secure location, or delete it.

6. If you generated the RSA key pair on one of your Linux nodes, remember to delete the keys after you finish using them. HPC Pack does not set up mutual trust if it finds an existing id_rsa file or id_rsa.pub file.

IMPORTANT

We don't recommend running a Linux job as a cluster administrator on a shared cluster, because a job submitted by an administrator runs under the root account on the Linux nodes. A job submitted by a non-administrator user runs under a local Linux user account with the same name as the job user. In this case, HPC Pack sets up mutual trust for this Linux user across all the nodes allocated to the job. You can set up the Linux user manually on the Linux nodes before running the job, or HPC Pack creates the user automatically when the job is submitted. If HPC Pack creates the user, HPC Pack deletes it after the job completes. To reduce security threat, the keys are removed after the job completes on the nodes.

Set up a file share for Linux nodes

Now set up an SMB file share, and mount the shared folder on all Linux nodes to allow the Linux nodes to access NAMD files with a common path. Following are steps to mount a shared folder on the head node. A share is recommended for distributions such as CentOS 6.6 that currently don't support the Azure File service. If your Linux nodes support an Azure File share, see [How to use Azure File storage with Linux](#). For additional file sharing options with HPC Pack, see [Get started with Linux compute nodes in an HPC Pack Cluster in Azure](#).

1. Create a folder on the head node, and share it to Everyone by setting Read/Write privileges. In this example, \\CentOS66HN\Namd is the name of the folder, where CentOS66HN is the host name of the head node.
2. Create a subfolder named namd2 in the shared folder. In namd2, create another subfolder named namdsample.
3. Extract the NAMD files in the folder by using a Windows version of **tar** or another Windows utility that operates on .tar archives.
 - Extract the NAMD tar archive to \\CentOS66HN\Namd\namd2.
 - Extract the tutorial files under \\CentOS66HN\Namd\namd2\namdsample.
4. Open a Windows PowerShell window and run the following commands to mount the shared folder on the Linux nodes.

```
clusrun /nodegroup:LinuxNodes mkdir -p /namd2  
clusrun /nodegroup:LinuxNodes mount -t cifs //CentOS66HN/Namd/namd2 /namd2 -o vers=2.1` ,username=<username>` ,password='<password>'` ,dir_mode=0777` ,file_mode=0777
```

The first command creates a folder named /namd2 on all nodes in the LinuxNodes group. The second command mounts the shared folder //CentOS66HN/Namd/namd2 onto the folder with dir_mode and file_mode bits set to 777. The *username* and *password* in the command should be the credentials of a user on the head node.

NOTE

The `` symbol in the second command is an escape symbol for PowerShell. `` means the `` (comma character) is a part of the command.

Create a Bash script to run a NAMD job

Your NAMD job needs a *nodelist* file for **charmrn** to determine the number of nodes to use when starting NAMD processes. You use a Bash script that generates the nodelist file and runs **charmrn** with this nodelist file. You can then submit a NAMD job in HPC Cluster Manager that calls this script.

Using a text editor of your choice, create a Bash script in the /namd2 folder containing the NAMD program files and name it hpccharmrn.sh. For a quick proof of concept, copy the example hpccharmrn.sh script provided at the end of this article and go to [Submit a NAMD job](#).

TIP

Save your script as a text file with Linux line endings (LF only, not CR LF). This ensures that it runs properly on the Linux nodes.

Following are details about what this bash script does.

1. Define some variables.

```
#!/bin/bash

# The path of this script
SCRIPT_PATH=$( dirname "${BASH_SOURCE[0]}" )
# Charmrun command
CHARMRUN=${SCRIPT_PATH}/charmrn
# Argument of ++nodelist
NODELIST_OPT="--nodelist"
# Argument of ++p
NUMPROCESS="--p"
```

2. Get node information from the environment variables. \$NODESCORES stores a list of split words from \$CCP_NODES_CORES. \$COUNT is the size of \$NODESCORES.

```
# Get node information from the environment variables
NODESCORES=(${CCP_NODES_CORES})
COUNT=${#NODESCORES[@]}
```

The format for the \$CCP_NODES_CORES variable is as follows:

```
<Number of nodes> <Name of node1> <Cores of node1> <Name of node2> <Cores of node2>...
```

This variable lists the total number of nodes, node names, and number of cores on each node that are allocated to the job. For example, if the job needs 10 cores to run, the value of \$CCP_NODES_CORES is similar to:

```
3 CENTOS66LN-00 4 CENTOS66LN-01 4 CENTOS66LN-03 2
```

3. If the \$CCP_NODES_CORES variable is not set, start **charmrn** directly. (This should only occur when you run this script directly on your Linux nodes.)

```

if [ ${COUNT} -eq 0 ]
then
# CCP_NODES is_CORES is not found or is empty, so just run charmrn without nodelist arg.
#echo ${CHARMRUN} $*
${CHARMRUN} $*

```

4. Or create a nodelist file for **charmrn**.

```

else
# Create the nodelist file
NODELIST_PATH=${SCRIPT_PATH}/nodelist_$$

# Write the head line
echo "group main" > ${NODELIST_PATH}

# Get every node name and number of cores and write into the nodelist file
I=1
while [ ${I} -lt ${COUNT} ]
do
    echo "host ${NODESCORES[$I]} ++cpus ${NODESCORES[$((I+1))]}" >> ${NODELIST_PATH}
    let "I=$I+2"
done

```

5. Run **charmrn** with the nodelist file, get its return status, and remove the nodelist file at the end.

`${CCP_NUMCPUS}` is another environment variable set by the HPC Pack head node. It stores the number of total cores allocated to this job. We use it to specify the number of processes for charmrn.

```

# Run charmrn with nodelist arg
#echo ${CHARMRUN} ${NUMPROCESS}${CCP_NUMCPUS} ${NODELIST_OPT} ${NODELIST_PATH} $*
#${CHARMRUN} ${NUMPROCESS}${CCP_NUMCPUS} ${NODELIST_OPT} ${NODELIST_PATH} $*

RTNSTS=$?
rm -f ${NODELIST_PATH}
fi

```

6. Exit with the **charmrn** return status.

```
exit ${RTNSTS}
```

Following is the information in the nodelist file, which the script generates:

```

group main
host <Name of node1> ++cpus <Cores of node1>
host <Name of node2> ++cpus <Cores of node2>
...

```

For example:

```

group main
host CENTOS66LN-00 ++cpus 4
host CENTOS66LN-01 ++cpus 4
host CENTOS66LN-03 ++cpus 2

```

Submit a NAMD job

Now you are ready to submit a NAMD job in HPC Cluster Manager.

1. Connect to your cluster head node and start HPC Cluster Manager.
2. In **Resource Management**, ensure that the Linux compute nodes are in the **Online** state. If they are not, select them and click **Bring Online**.
3. In **Job Management**, click **New Job**.
4. Enter a name for job such as *hpccharmrun*.

The screenshot shows the 'Job' interface with the 'Job Details' tab active. On the left, there's a sidebar with links: 'Edit Tasks', 'Resource Selection', 'Licenses', 'Environment Variables', and 'Advanced'. The main area has four input fields: 'Job name' (Hpccharmrun), 'Job template' (Default), 'Project' (empty), and 'Priority' (Normal).

5. On the **Job Details** page, under **Job Resources**, select the type of resource as **Node** and set the **Minimum** to 3., we run the job on three Linux nodes and each node has four cores.

The screenshot shows the 'Job resources' section. It has a dropdown menu set to 'Node'. Below it, there are two sets of input fields for 'Minimum' and 'Maximum': both have radio buttons for 'Auto-calculate' and numerical input fields (3 and 1 respectively). There are also two checkboxes: one for 'Use assigned resources exclusively for this job' (unchecked) and one for 'Run the entire job on a single node' (unchecked).

6. Click **Edit Tasks** in the left navigation, and then click **Add** to add a task to the job.
7. On the **Task Details and I/O Redirection** page, set the following values:

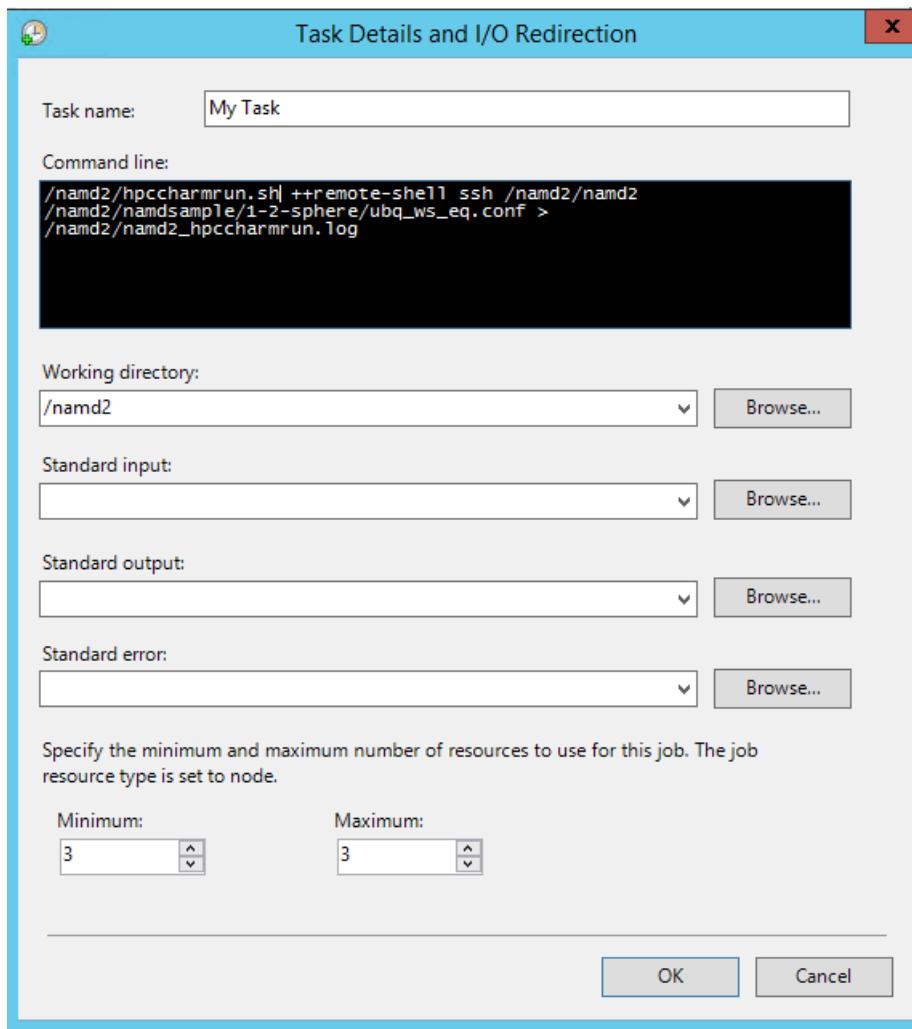
- **Command line** -

```
/namd2/hpccharmrun.sh ++remote-shell ssh /namd2/namd2 /namd2/namdsample/1-2-sphere/ubq_ws_eq.conf  
> /namd2/namd2_hpccharmrun.log
```

TIP

The preceding command line is a single command without line breaks. It wraps to appear on several lines under **Command line**.

- **Working directory** - /namd2
- **Minimum** - 3



NOTE

You set the working directory here because **charmrun** tries to navigate to the same working directory on each node. If the working directory isn't set, HPC Pack starts the command in a randomly named folder created on one of the Linux nodes. This causes the following error on the other nodes:

`/bin/bash: line 37: cd: /tmp/nodemanager_task_94_0.mFlQSN: No such file or directory.` To avoid this problem, specify a folder path that can be accessed by all nodes as the working directory.

8. Click **OK** and then click **Submit** to run this job.

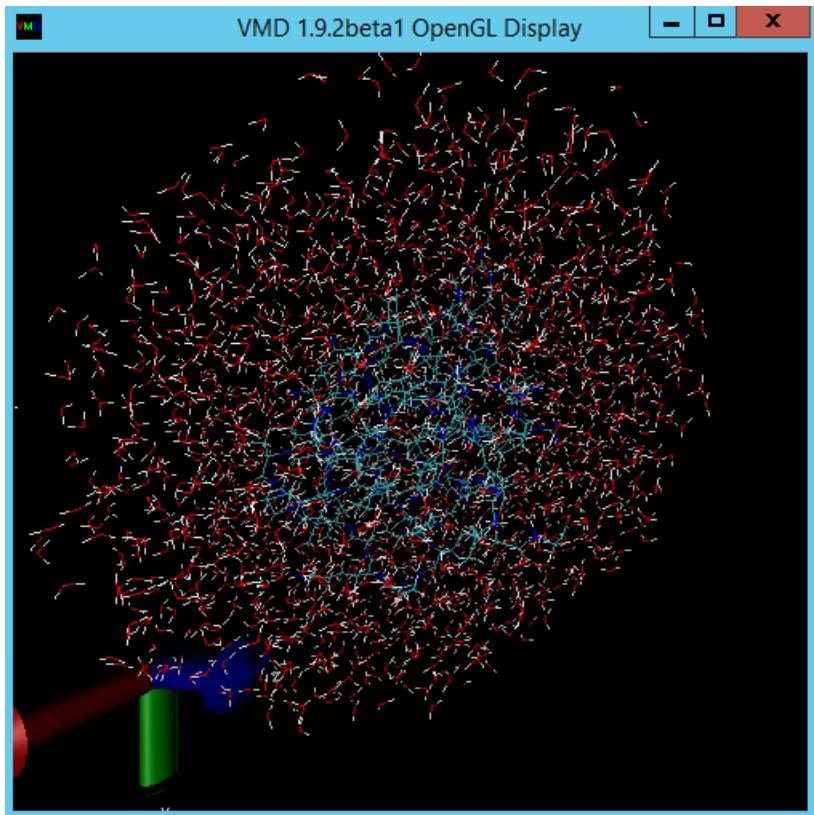
By default, HPC Pack submits the job as your current logged-on user account. A dialog box might prompt you to enter the user name and password after you click **Submit**.



Under some conditions, HPC Pack remembers the user information you input before and doesn't show this dialog box. To make HPC Pack show it again, enter the following command at a Command Prompt and then submit the job.

```
hpccred delcreds
```

9. The job takes several minutes to finish.
10. Find the job log at \\Namd\\namd2\\namd2_hpccharmrun.log and the output files in \\Namd\\namd2\\namdsample\\1-2-sphere.
11. Optionally, start VMD to view your job results. The steps for visualizing the NAMD output files (in this case, a ubiquitin protein molecule in a water sphere) are beyond the scope of this article. See [NAMD Tutorial](#) for details.



Sample files

[Sample XML configuration file for cluster deployment by PowerShell script](#)

```
<?xml version="1.0" encoding="utf-8" ?>
<IaaSClusterConfig>
  <Subscription>
    <SubscriptionName>Subscription-1</SubscriptionName>
    <StorageAccount>mystorageaccount</StorageAccount>
  </Subscription>
  <Location>West US</Location>
<VNet>
  <VNetName>MyVNet</VNetName>
  <SubnetName>Subnet-1</SubnetName>
</VNet>
<Domain>
  <DCOption>HeadNodeAsDC</DCOption>
  <DomainFQDN>hpclab.local</DomainFQDN>
</Domain>
<Database>
  <DBOption>LocalDB</DBOption>
</Database>
<HeadNode>
  <VMName>CentOS66HN</VMName>
  <ServiceName>MyHPCService</ServiceName>
  <VMSize>Large</VMSize>
  <EnableRESTAPI />
  <EnableWebPortal />
</HeadNode>
<LinuxComputeNodes>
  <VMNamePattern>CentOS66LN-%00%</VMNamePattern>
  <ServiceName>MyLnxCNService</ServiceName>
  <VMSize>Large</VMSize>
  <NodeCount>4</NodeCount>
  <ImageName>5112500ae3b842c8b9c604889f8753c3__OpenLogic-CentOS-66-20150325</ImageName>
</LinuxComputeNodes>
</IaaSClusterConfig>
```

Sample cred.xml file

```

<ExtendedData>
  <PrivateKey>-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAxJKBABhnOsE9eneGHvsjdoXkooHuxpTHI1JVunAJkVmFy8JC
qFt1pV98QctKEHTC6kQ7tj1UT2N6nx1EY9BBHpZacnXmknpKdX4Nu0cNlSpLpru
1scKPR3XVzkTwEF000MiNJVKnq8qXJF1T3lYx3rW5EnItn6C3nQm3gQPXP0ckYCF
Jdtu/6SSgzV9kaapctLGPNp1Vjf9KeDQMrijxsQNHxnQcfiICp21NiUCiXosDqJrR
AfzePd10XwsNngouy8t0fP1NsngZvsx+kPGh/AKakKIYS0cO9W3FmdYNW8Xehzkc
VzrtjhU8x21hXGfSC7V0ZeD7dMetL3tQCVxCmwIDAQABoIBAQcve8Jh3Wc6koxZ
qh43xicwhdwSGylizisoozYZDC/ebDb/Ydq0BYIPMiDwADVMX5AqJuPPmwyLGtm6
9hu5p46aycrQ5+QA299g6D1F+PZtNb0wKuvX+rRvPxagrTmupkCswjg1DUEYUHPW
05wQaNoSqtzwS9Y85M/b24FFLeyxK0n8zjKFERJaHdhVxI6cxw7RdV1SmM9UHmah
wTkW8Hkb1b0ArilAHi6S1RTNZG4gTGeDzPb7fYZo3hzJyLbcaNfJscUuqnAJ+6pT
iY6NNp1E8PQgjvHe21yv3DRoVRM4egqQvNzgUbYAMUgr30T1UoxnUXwk2vqJMfg2
Nzw0ESGRAoGBAPkfxjjGfc4HryqPkdx0kjxs0bXC3js2g4IXItK9YUFeZzf+476y
OTMQg/8DUbqd5rLv7PITIAqpGs39pkfnvohPj0e2zzzeoyaXurYIPV98hhH880uH
ZUh0xJYnlqHGxGT7p2PmmnAlmY4TSJrp12VnuiQVVVsXWOGPqHx4S4f9AoGBAMn/
vuea7hsCgwIE25MJ55FYCJodLkioQy6aGP4NgB89Azzg527WsQ6H5xhgVMKHlyu
Q1sn+pq8LyzD0i1veEvWb8EYifsMyTIPXOUTwZgzaTTCeJNHdc4gw1U22vd70BYy
nZCU7n8Pe6eIMNtnVduiv+2QHuiNPgN7M73/x3AoGBAOl0IcmFgy0EsR8MBq0Z
ge4gnniBXCYDptEINNBAeVStJUnNKzwab6PGwmw6w2VI3tbXbi31bRA1Mve7fKK
B2ghWNPsJ0tppKbPCek2Hnt0HUwb7qX7Z1j2cX/99uvRAjChVsDbYA0VJAxcIwQG
TxXx5pFi4g0HexCa6LrkeKMdAoGAcvRIACX70wPC6nM5QgQDt95jRzGKu5EpdcTf
g4TNtp1lib1LPYhRrzokoyoahTeyxxak3ktDFCLj9eW6xoCZRQ9Tqd/9JhGwrfxw
MS19DtCzHoNNNewM/135tqyD8m7pTwM4tPQqDtmwGERWKj7BaNZARU1hFxwOoems
R6DbZyEcGyeAhjL2N3Pc+WW+8x2bbIBN3rJcMjBBIivB62AwgYZnA2D5wk5o0DKD
eesGSKS5122ZMXJNShzPKmv3HpH22CSVp00sNZ6R+iG8a3oq4QkU61MT1CfGoMI
a8lxTKnZCsRXU1HexqZs+DSc+30tz50bNqlid0/15B4EJnQP03ci00=
-----END RSA PRIVATE KEY-----</PrivateKey>
  <PublicKey>ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQDEkoEAGGc6wT16d4Ye+yN2hcqigdTGlMcjU1w6cAmRWYXlwKoW3w1X3xAK0oQdMLqRDu2PVRPY3qfHUR
j0EEellpydeaSekp1fg27Rw2VkmEumu6Wxwo9HddXORPAQXTQ4yI01wSerypckXVPeVjHetbkSci2foLedCbeBA9c/RyRgIU1227/pJKDNX2Rpq
ly0sY82nVWN/0p4NAyslexA0FgdBx+IgKnB2JQKjeiwOomtEB/N492XRFcw2eCi7Ly3R8+U1KeBm+zH6Q8aH8ApqQohhLRw71bcWZ1g1bxsd6HO
RxX0u0mFTzHbwFcZ9ILtXR14Pt0x5Mve1AJXEKb username@servername;</PublicKey>
</ExtendedData>

```

Sample hpccharmrund.sh script

```

#!/bin/bash

# The path of this script
SCRIPT_PATH="$( dirname "${BASH_SOURCE[0]}" )"
# Charmrun command
CHARMRUN=${SCRIPT_PATH}/charmrun
# Argument of ++nodelist
NODELIST_OPT="--nodelist"
# Argument of ++p
NUMPROCESS="--p"
# Get node information from ENVs
# CCP_NODES_CORES=3 CENTOS66LN-00 4 CENTOS66LN-01 4 CENTOS66LN-03 4
NODESCORES=(${CCP_NODES_CORES})
COUNT=${#NODESCORES[@]}

if [ ${COUNT} -eq 0 ]
then
    # If CCP_NODES_CORES is not found or is empty, just run the charmrun without nodelist arg.
    #echo ${CHARMRUN} $*
    ${CHARMRUN} $*
else
    # Create the nodelist file
    NODELIST_PATH=${SCRIPT_PATH}/nodelist_$$

    # Write the head line
    echo "group main" > ${NODELIST_PATH}

    # Get every node name & cores and write into the nodelist file
    I=1
    while [ ${I} -lt ${COUNT} ]
    do
        echo "host ${NODESCORES[$I]} ++cpus ${NODESCORES[$((I+1))]}" >> ${NODELIST_PATH}
        let "I=$I+2"
    done

    # Run the charmrun with nodelist arg
    #echo ${CHARMRUN} ${NUMPROCESS}${CCP_NUMCPUS} ${NODELIST_OPT} ${NODELIST_PATH} $*
    ${CHARMRUN} ${NUMPROCESS}${CCP_NUMCPUS} ${NODELIST_OPT} ${NODELIST_PATH} $*

    RTNSTS=$?
    rm -f ${NODELIST_PATH}
fi

exit ${RTNSTS}

```

Install NVIDIA GPU drivers on N-series VMs running Linux

4/30/2018 • 7 min to read • [Edit Online](#)

To take advantage of the GPU capabilities of Azure N-series VMs running Linux, NVIDIA graphics drivers must be installed. This article provides driver setup steps after you deploy an N-series VM. 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

NC, NCv2, NCv3, and ND-series - NVIDIA CUDA drivers

CUDA driver information in the following table 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	NVIDIA CUDA 9.1, driver branch R390
Red Hat Enterprise Linux 7.3 or 7.4	
CentOS-based 7.3 or 7.4, CentOS-based 7.4 HPC	

NV-series - NVIDIA GRID drivers

Microsoft redistributes NVIDIA GRID driver installers for NV VMs. Install only these GRID drivers on Azure NV VMs. These drivers include licensing for GRID Virtual GPU Software in Azure.

DISTRIBUTION	DRIVER
Ubuntu 16.04 LTS	NVIDIA GRID 6.0, driver branch R390
Red Hat Enterprise Linux 7.3 or 7.4	
CentOS-based 7.3 or 7.4	

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 for NC, NCv2, NCv3, and ND-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 16.04 LTS

1. Download and install the CUDA drivers.

```
CUDA_REPO_PKG=cuda-repo-ubuntu1604_9.1.85-1_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  
http://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.

2. 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
```

1. Update the 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-rhel7-9.1.85-1.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 Fan	Name Temp	Perf Persistence-M Pwr:Usage/Cap	Bus-Id	Disp.A	Volatile Memory-Usage	Uncorr. GPU-Util	ECC Compute M.	
0 N/A	Tesla K80 51C	P0 58W / 149W	off 000007D1:00:00.0	off 0MiB / 11439MiB	0% 0%	Default	0	
Processes:								
GPU PID Type Process name GPU Memory Usage								
No running processes found								

RDMA network connectivity

RDMA network connectivity can be enabled on RDMA-capable N-series VMs such as NC24r deployed in the same availability set or 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 for NV-series VMs

To install NVIDIA GRID drivers on NV-series VMs, make an SSH connection to each VM and follow the steps for your Linux distribution.

Ubuntu 16.04 LTS

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 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=849941  
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=TRUE
```

9. Reboot the VM and proceed to verify the installation.

CentOS or Red Hat Enterprise Linux

1. Update the kernel and DKMS.

```
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 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=849941  
  
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=TRUE
```

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 |          0%       Default |
| N/A   31C    P0    39W / 150W |           0MiB /  8123MiB |          off
+-----+
+-----+
| 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 VM, `x11vnc` is recommended because it allows hardware acceleration of graphics. The BusID of the M60 device must be manually added to the xconfig file (`/etc/X11/xorg.conf` on Ubuntu 16.04 LTS, `/etc/X11/XF86config` on CentOS 7.3 or Red Hat Enterprise Server 7.3).

Add a `"Device"` section similar to the following:

```
Section "Device"
    Identifier      "Device0"
    Driver          "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName      "Tesla M60"
    BusID          "your-BusID:0:0:0"
EndSection
```

Additionally, update your `"Screen"` section to use this device.

The decimal BusID can be found by running

```
echo $((16#`/usr/bin/nvidia-smi --query-gpu=pci.bus_id --format=csv | tail -1 | cut -d ':' -f 1`))
```

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 the following contents:

```
#!/bin/bash
BUSID=$((`/usr/bin/nvidia-smi --query-gpu=pci.bus_id --format=csv | tail -1 | cut -d ':' -f 1`))

if grep -Fxq "${BUSID}" /etc/X11/XF86Config; then      echo "BUSID is matching"; else    echo "BUSID changed to
${BUSID}" && sed -i '/BusID/c\      BusID          \"PCI:0@'${BUSID}'\:0:0:0\"' /etc/X11/XF86Config; 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](#).

Frequently asked questions about Azure IaaS VM disks and managed and unmanaged premium disks

4/9/2018 • 10 min to read • [Edit Online](#)

This article answers some frequently asked questions about Azure Managed Disks and Azure Premium Storage.

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.

Is there a way that I can copy or export a managed disk to a private storage account?

Yes. You can export 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?

No.

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, and also the default limit, is 10,000 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. However, we are planning to support incremental snapshots in the future.

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. In the future, we plan to switch diagnostics to Managed Disks as well.

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 get a read-only shared access signature URI for the managed disk and use it to copy the contents to a private storage account or on-premises storage.

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. We support unmanaged and managed disks. We recommend that you use managed disks for new workloads and migrate your current workloads to managed disks.

If I create a 128-GB disk and then increase the size to 130 GB, will I be charged for the next disk size (512 GB)?

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>

Migrate to Managed Disks

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 VM Scale Sets (VMSS) from unmanaged disks to Managed Disks supported?

No. You can create a new VMSS with Managed Disks using the image from your old VMSS 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 (ASR) via Azure to Azure replication?

Yes. Currently, ASR Azure to Azure protection for VMs with Managed Disks is only available as a public preview 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 is available. Managed Disks is available in all public regions and Germany.

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 Storage, 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 Storage, 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 Storage.

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 Storage?

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 MB per second per core. The GS series offers 5,000 IOPS per core and 50 MB 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 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 TB. The largest size that Azure supports for an operating system disk is 2 TB. Azure supports up to 4 TB for data disks.

What is the largest page blob size that's supported?

The largest page blob size that Azure supports is 8 TB (8,191 GB). We don't support page blobs larger than 4 TB (4,095 GB) attached to a VM as data or operating system disks.

Do I need to use a new version of Azure tools to create, attach, resize, and upload disks larger than 1 TB?

You don't need to upgrade your existing Azure tools to create, attach, or resize disks larger than 1 TB. 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:

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
AzCopy	Version number 6.1.0: June 2017 release or later

The support for Azure CLI v2 and Azure Storage Explorer is coming soon.

Are P4 and P6 disk sizes supported for unmanaged disks or page blobs?

No. P4 (32 GB) and P6 (64 GB) disk sizes are supported only for managed disks. Support for unmanaged disks and page blobs is coming soon.

If my existing premium managed disk less than 64 GB was created before the small disk was enabled (around June 15, 2017), how is it billed?

Existing small premium disks less than 64 GB continue to be billed according to the P10 pricing tier.

How can I switch the disk tier of small premium disks less than 64 GB 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.

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](#).

Add a disk to a Linux VM

4/9/2018 • 8 min 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.

Use managed disks

Azure Managed Disks simplifies disk management for Azure VMs by managing the storage accounts associated with the VM disks. You only have to specify the type (Premium or Standard) and the size of disk you need, and Azure creates and manages the disk for you. For more information, see [Managed Disks overview](#).

Attach a new disk to a VM

If you just need a new 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 50Gb in size:

```
az vm disk attach -g myResourceGroup --vm-name myVM --disk myDataDisk \
--new --size-gb 50
```

Attach an existing disk

In many cases, you attach disks that have already been created. 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*:

```
# find the disk id
diskId=$(az disk show -g myResourceGroup -n myDataDisk --query 'id' -o tsv)
az vm disk attach -g myResourceGroup --vm-name myVM --disk $diskId
```

The output looks something like the following (you can use the `-o table` option to any command to format the output in):

```
{
  "accountType": "Standard_LRS",
  "creationData": {
    "createOption": "Empty",
    "imageReference": null,
    "sourceResourceId": null,
    "sourceUri": null,
    "storageAccountId": null
  },
  "diskSizeGb": 50,
  "encryptionSettings": null,
  "id": "/subscriptions/<guid>/resourceGroups/rasquill-script/providers/Microsoft.Compute/disks/myDataDisk",
  "location": "westus",
  "name": "myDataDisk",
  "osType": null,
  "ownerId": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "myResourceGroup",
  "tags": null,
  "timeCreated": "2017-02-02T23:35:47.708082+00:00",
  "type": "Microsoft.Compute/disks"
}
```

Use unmanaged disks

Unmanaged disks require additional overhead to create and manage the underlying storage accounts.

Unmanaged disks are created in the same storage account as your OS disk. To create and attach an unmanaged disk, use the [az vm unmanaged-disk attach](#) command. The following example attaches a 50GB unmanaged disk to the VM named *myVM* in the resource group named *myResourceGroup*:

```
az vm unmanaged-disk attach -g myResourceGroup -n myUnmanagedDisk --vm-name myVM \
--new --size-gb 50
```

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 Azure 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 the disk with `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
```

The output is 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
```

Create the partition by typing `p` at the prompt as follows:

```
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
```

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 -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, we 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
```

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

- Remember, that your new disk is not available to the VM if it reboots unless you write that information to your `fstab` file.
- 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

4/9/2018 • 1 min to read • [Edit Online](#)

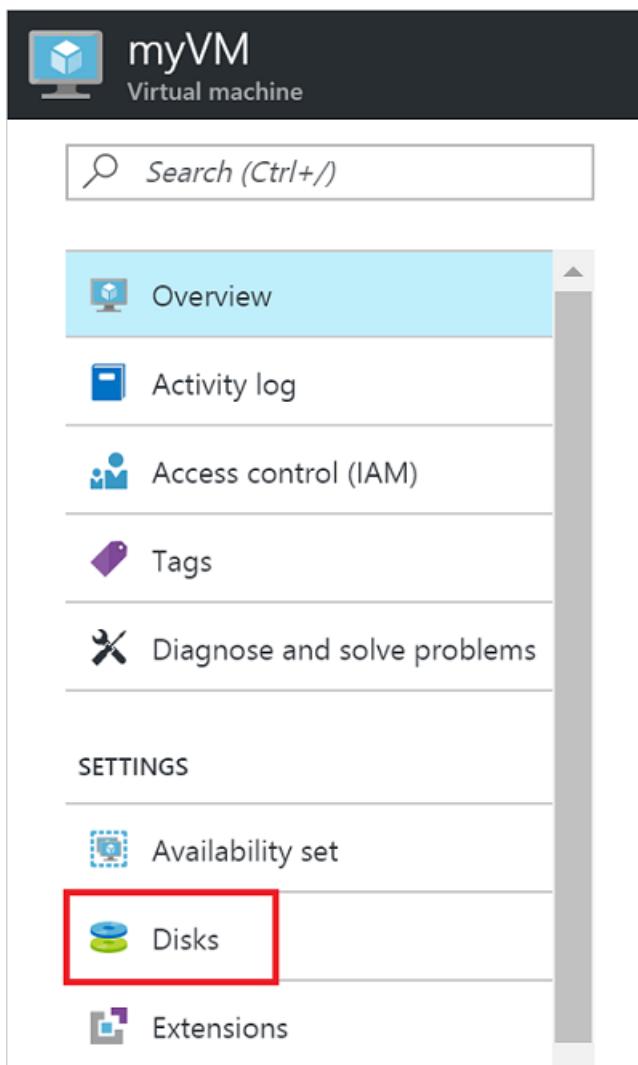
This article shows you how to attach both new and existing disks to a Linux virtual machine through the Azure portal. You can also [attach a data disk to a Windows VM in the Azure portal](#).

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](#).
- To use Premium storage, you need a DS-series or GS-series virtual machine. You can use both Premium and Standard disks with these virtual machines. Premium storage is available in certain regions. For details, see [Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads](#).
- Disks attached to virtual machines are actually .vhd files stored in Azure. For details, see [About disks and VHDs for virtual machines](#).

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**:

LUN	NAME	SIZE	ACCOUNT TYPE
0	myDataDisk		Premium_LRS

3. Enter a name for your managed disk. Review the default settings, update as necessary, and then click **Create**.

Create managed disk

* Name: myDataDisk

* Resource group: myResourceGroup

* Account type: Premium_LRS

* Source type: None (empty disk)

* Size (GiB): 1023

Estimated performance:

IOPS limit	5000
Throughput limit (MB/s)	200

Create

4. Click **Save** to create the managed disk and update the VM configuration:

LUN	NAME	SIZE	ACCOUNT TYPE	ENCRYPTION	CACHING
0	myDataDisk	1023 GiB	Premium_LRS		None

5. 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 Azure portal interface for a resource group named 'myResourceGroup'. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Quickstart, Resource costs, and Deployments. The main area is titled 'Essentials' and shows deployment details: Subscription name (Visual Studio Ultimate with MSDN), Subscription ID (8fa5cd83-7fbb-431a-af16-4a20dede8802), Deployments (2 Succeeded), and Location (West US). Below this is a table titled 'Filter by name...' with columns NAME, TYPE, and LOCATION. It lists three items: 'myAvailabilitySet' (Availability set, West US), 'myDataDisk' (Disk, West US), and 'myNetworkSecurityGroup' (Network security gro..., West US). The 'myDataDisk' row is highlighted with a red box.

Attach an existing disk

1. On the **Disk**s pane, click + **Add data disk**.
2. 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:

The screenshot shows the 'Add data disk' dialog box. At the top are 'Save' and 'Discard' buttons. The 'OS disk' section shows one item: 'myVM' (Premium_LRS). The 'Data disks' section shows one item: 'myDataDisk' (1023 GiB, Premium_LRS). Below these is a dropdown menu labeled '1' with a red border. A tooltip-like box is overlaid on the dropdown, containing options: 'Create disk', 'Disks in resource group 'myResourceGroup'', 'myExistingDisk size: 1023 GiB, account type: Premium_LRS', and 'All disks myExistingDisk size: 1023 GiB, account type: Premium_LRS, resource group: MYRESOURCEGROUP'.

3. Click **Save** to attach the existing managed disk and update the VM configuration:

Save **X 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	myExistingDisk	1023 GiB	Premium_LRS		None

+ Add data disk

4. After Azure attaches the disk to the virtual machine, it's listed in the virtual machine's disk settings under **Data Disks**.

Next steps

You can also [attach a data disk](#) using the Azure CLI.

How to detach a data disk from a Linux virtual machine

4/9/2018 • 1 min 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.

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 refer to [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.

Detach a data disk using CLI 2.0

```
az vm disk detach \
    -g myResourceGroup \
    --vm-name myVm \
    -n myDataDisk
```

The disk remains 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 **Disks** pane, select **Edit**.
5. In the **Disks** 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 remains 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](#).

How to expand virtual hard disks on a Linux VM with the Azure CLI

4/9/2018 • 3 min to read • [Edit Online](#)

The default virtual hard disk size for the operating system (OS) is typically 30 GB on a Linux virtual machine (VM) in Azure. You can [add data disks](#) to provide for additional storage space, but you may also wish to expand an existing data disk. This article details how to expand managed disks for a Linux VM with the Azure CLI 2.0. You can also expand the unmanaged OS disk with the [Azure CLI 1.0](#).

WARNING

Always make sure that you back up your data before you perform disk resize operations. For more information, see [Back up Linux VMs in Azure](#).

Expand Azure Managed Disk

Make sure that you have the latest [Azure CLI 2.0](#) installed and logged in to an Azure account 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 with your own values. Example parameter names include `myResourceGroup` and `myVM`.

- Operations on virtual hard disks cannot 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. `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 be 200Gb in size:

```
az disk update \
--resource-group myResourceGroup \
--name myDataDisk \
--size-gb 200
```

NOTE

When you expand a managed disk, the updated size is mapped 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 disk partition and filesystem

To use the expanded disk, you need to expand the underlying partition and filesystem.

1. SSH to your VM with the appropriate credentials. You can 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
```

2. To use the expanded disk, you need to expand the underlying partition and filesystem.

- a. If already mounted, unmount the disk:

```
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 215Gb in size:

```
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. Now 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 OS 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 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

4/9/2018 • 1 min 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 the Azure CLI 2.0 installed and logged into your Azure account. Run

```
az --version
```

to find the version. If you need to install or upgrade, see [Install Azure CLI 2.0](#).

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.

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

Take a snapshot named `osDisk-backup`.

```
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.

Use Azure portal

1. Sign in to the [Azure portal](#).
2. Starting in the upper-left, click **Create a resource** and search for **snapshot**.
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. Select an Azure datacenter Location.
7. For **Source disk**, select the Managed Disk to snapshot.
8. Select the **Account type** to use to store the snapshot. We recommend **Standard_LRS** unless you need it stored on a high performing disk.
9. 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

8/21/2017 • 7 min 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 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.

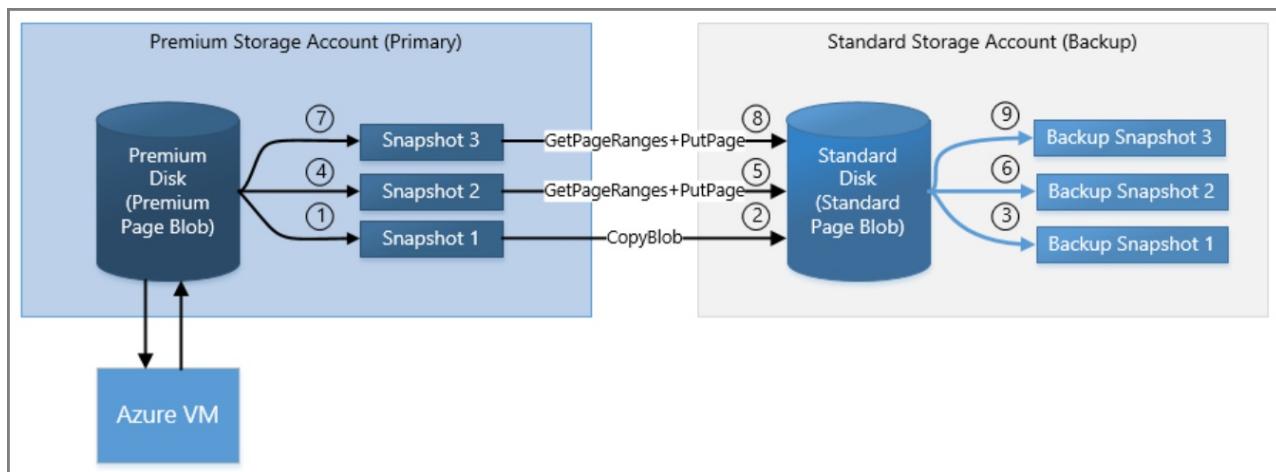
To learn about creating storage account and disks, refer to [About Azure storage accounts](#).

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.



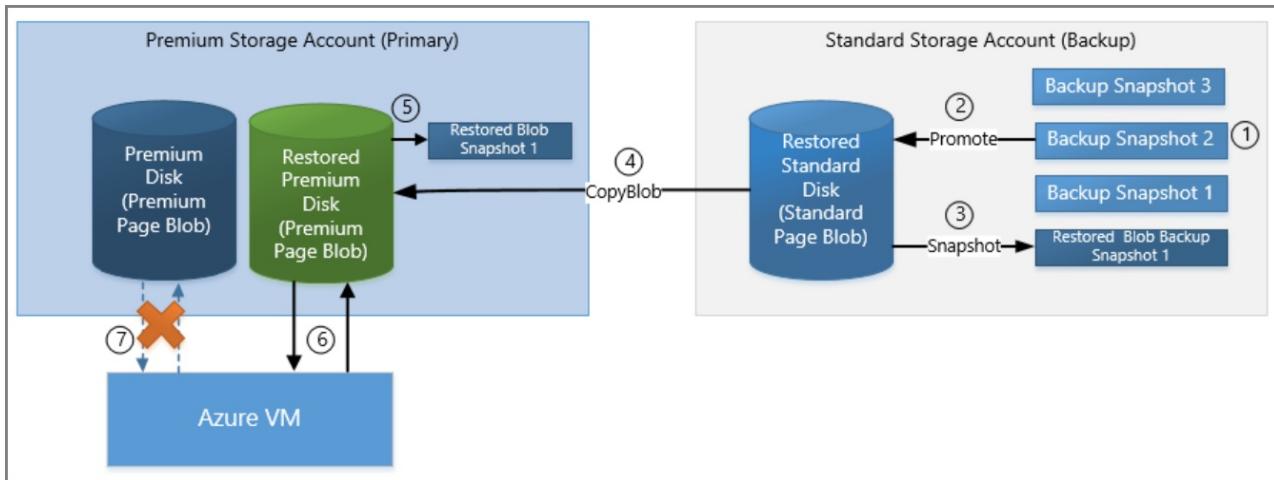
Steps to restore a disk from snapshots

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](#)

Convert a Linux virtual machine from unmanaged disks to managed disks

4/9/2018 • 3 min 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 2.0](#).

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.
- 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.
- 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](#)

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
```

Next steps

For more information about storage options, see [Azure Managed Disks overview](#).

Convert Azure managed disks storage from standard to premium, and vice versa

11/1/2017 • 2 min to read • [Edit Online](#)

Managed disks offers two storage options: [Premium](#) (SSD-based) and [Standard](#) (HDD-based). It allows you to easily switch between the two options with minimal downtime based on your performance needs. This capability is not available for unmanaged disks. But you can easily [convert to managed disks](#) to easily switch between the two options.

This article shows you how to convert managed disks from standard to premium, and vice versa by using Azure CLI. If you need to install or upgrade it, see [Install Azure CLI 2.0](#).

Before you begin

- The conversion requires a restart of the VM, so schedule the migration of your disks storage during a pre-existing maintenance window.
- If you are using unmanaged disks, first [convert to managed disks](#) to use this article to switch between the two storage options.

Convert all the managed disks of a VM from standard to premium, and vice versa

In the following example, we show how to switch all the disks of a VM from standard to premium 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

```

Convert a managed disk from standard to premium, and vice versa

For your dev/test workload, you may want to have mixture of standard and premium disks to reduce your cost. You can accomplish it by upgrading to premium storage, only the disks that require better performance. In the following example, we show how to switch a single disk of a VM from standard to premium storage, and vice versa. 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
```

Next steps

Take a read-only copy of a VM by using [snapshots](#).

Move files to and from a Linux VM using SCP

4/9/2018 • 2 min 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](#)

Migrate to Premium Storage by using Azure Site Recovery

5/2/2018 • 11 min to read • [Edit Online](#)

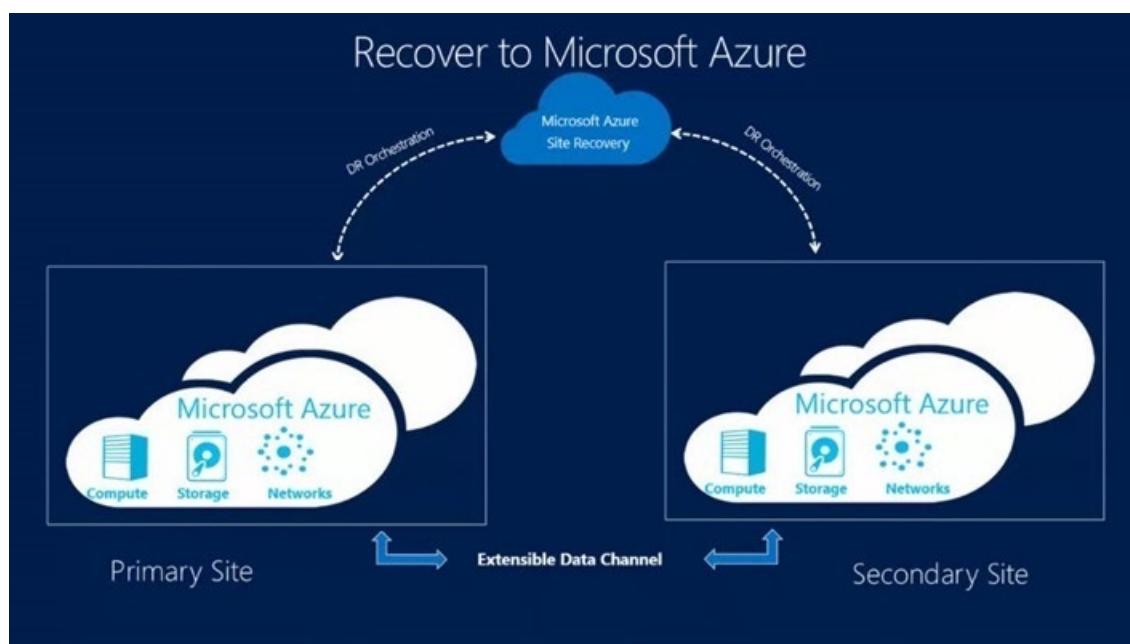
Azure Premium Storage delivers high-performance, low-latency disk support for virtual machines (VMs) that are running I/O-intensive workloads. This guide helps you migrate your VM disks from a standard storage account to a premium storage account by using [Azure Site Recovery](#).

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.

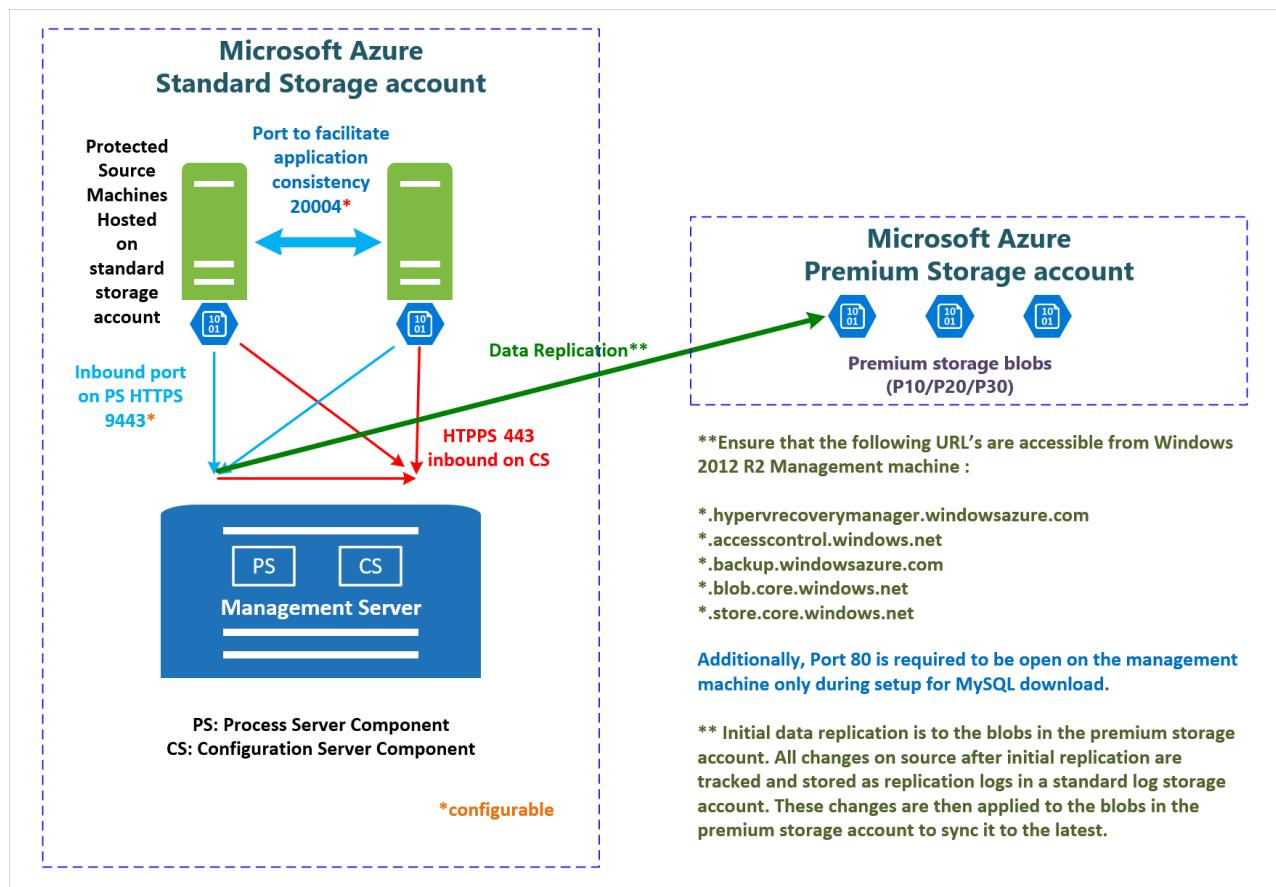
- **Process server** is a replication gateway that:

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

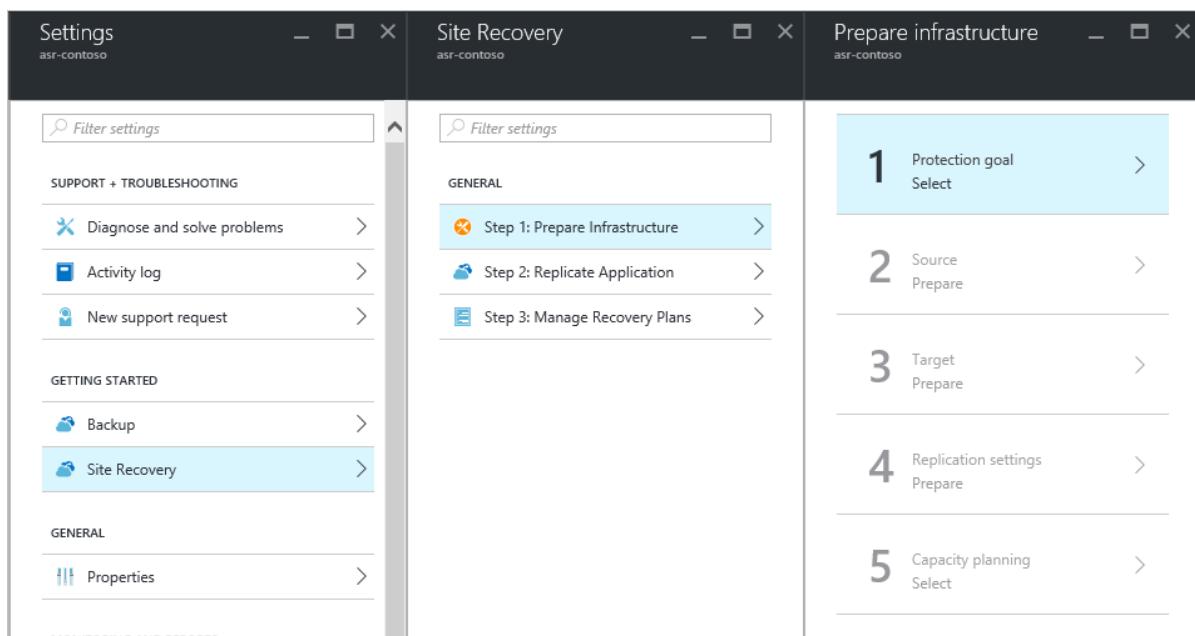
You can use Site Recovery to migrate Azure IaaS VMs between regions or within same region. The following instructions are tailored for this migration scenario from the article [Replicate VMware VMs or physical servers to Azure](#). Please follow the links for detailed steps in addition to the instructions in this article.

Step 1: Create a Recovery Services vault

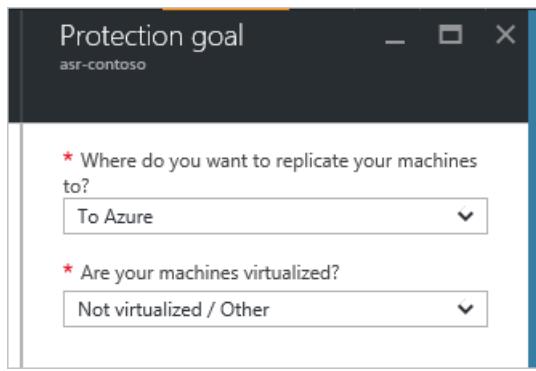
1. Open the [Azure portal](#).
2. Select **Create a resource > Management > Backup and Site Recovery (OMS)**. Alternatively, you can select **Browse > Recovery Services Vault > Add**.
3. Specify a region that VMs will be replicated to. For the purpose of migration in the same region, select the region 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**.

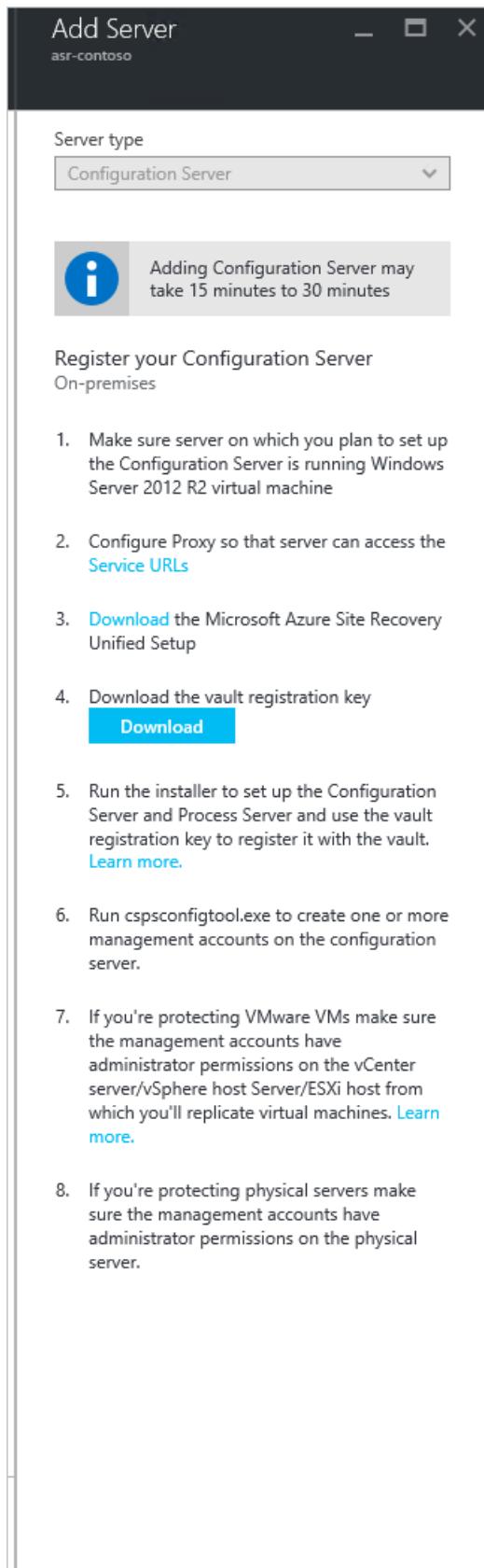


Step 3: Set up the source environment (configuration server)

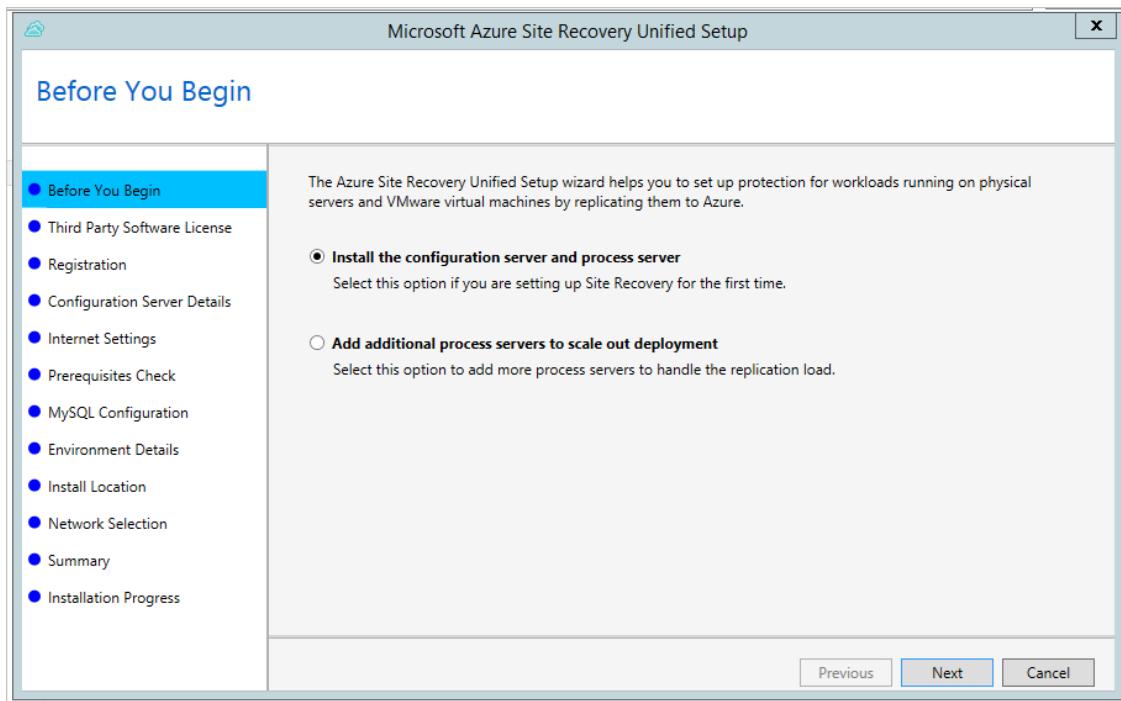
1. Download **Azure Site Recovery Unified Setup** and the vault registration key by going to the **Prepare infrastructure > Prepare source > Add Server** panes.

You will need the vault registration key to run the unified setup. The key is valid for five days after you generate it.

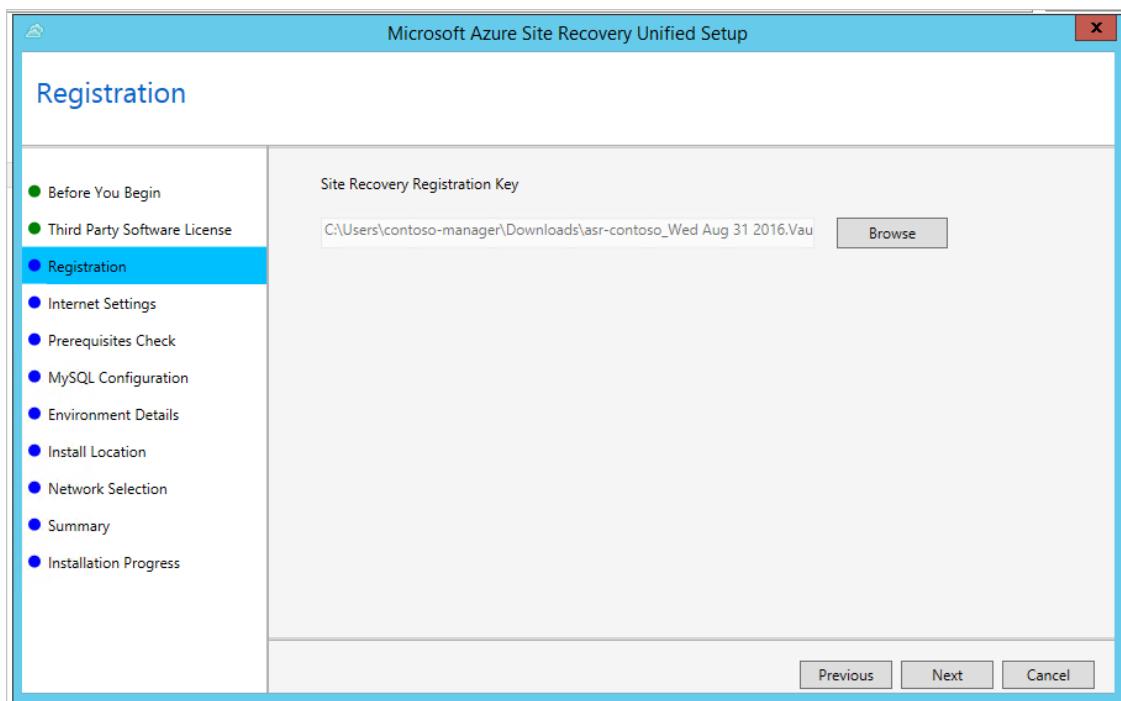
2. In the **Add Server** pane, add a configuration server.



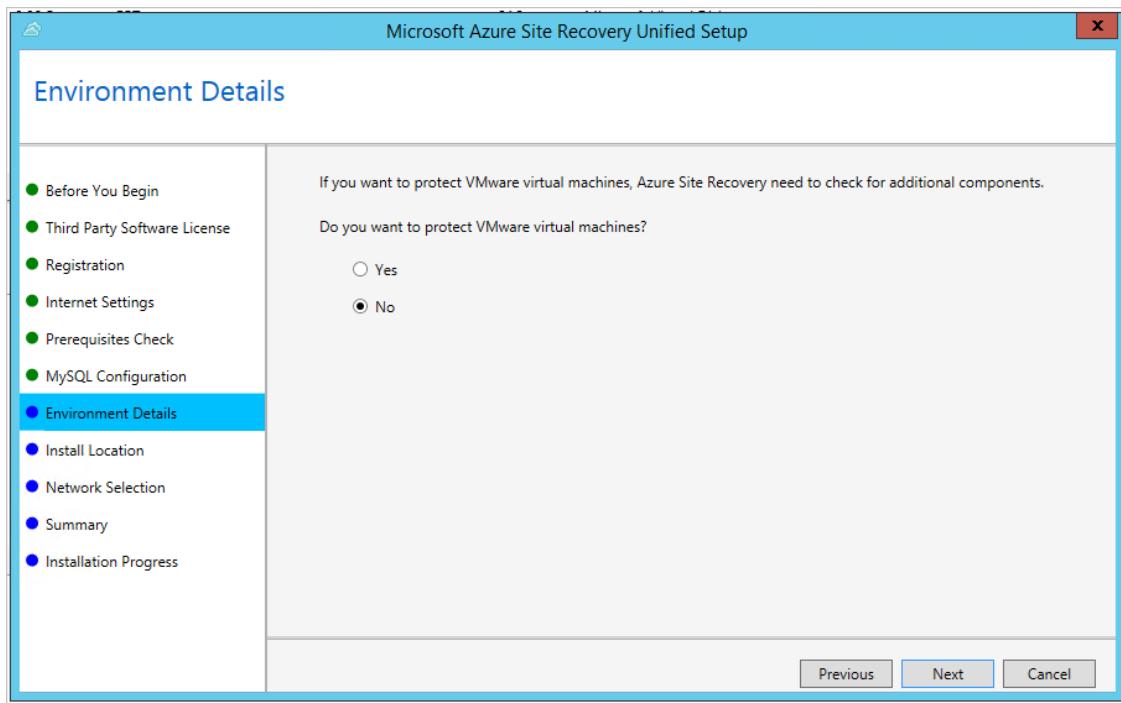
3. On the VM that you're using as the configuration server, run Unified Setup to install the configuration server and the process server. You can [walk through the screenshots](#) to complete the installation. You can refer to the following screenshots for steps specified for this migration scenario.
 - a. In **Before You Begin**, select **Install the configuration server and process server**.



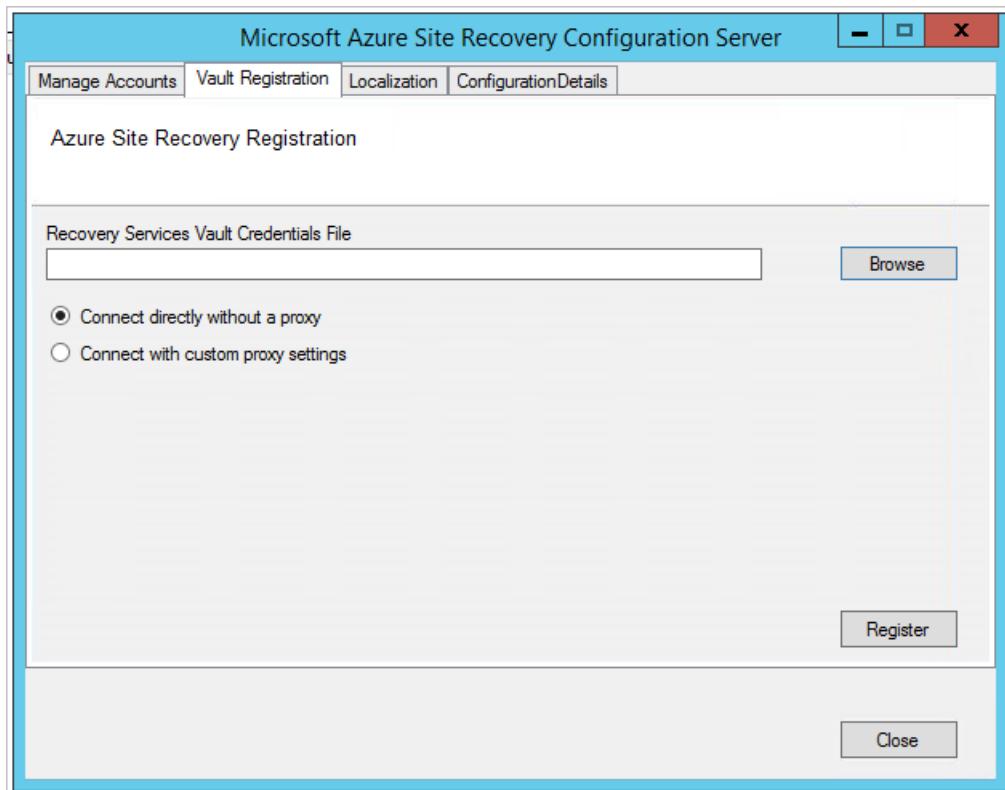
- b. In **Registration**, browse and select the registration key that you downloaded from the vault.



- 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 (checkmark), 2. Source (checkmark), 3. Target (highlighted in blue, 'Prepare'), 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 (Subscription: Visual Studio Enterprise, Deployment Model: Classic selected), Step 2 (Storage account(s): Found 5 compatible Azure storage accounts), and Step 3 (Network(s): Found 2 compatible Azure virtual networks).

Site Recovery checks that you have one or more compatible Azure storage accounts and networks.

NOTE

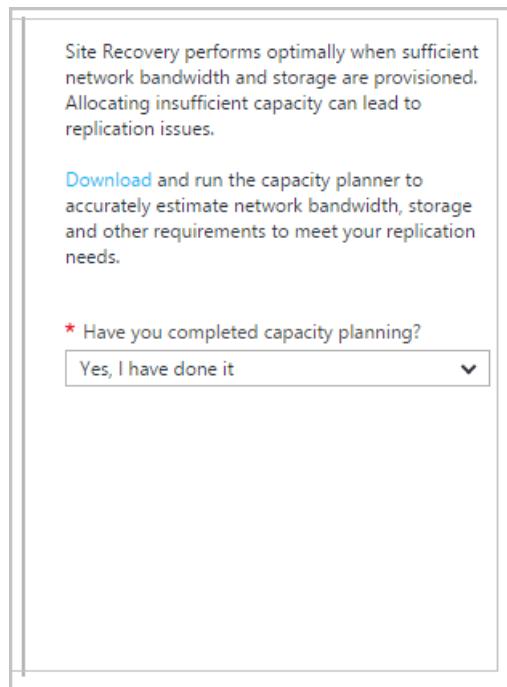
If you're using a premium storage account for replicated data, you need to set up an additional standard storage account to store replication logs.

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](#).

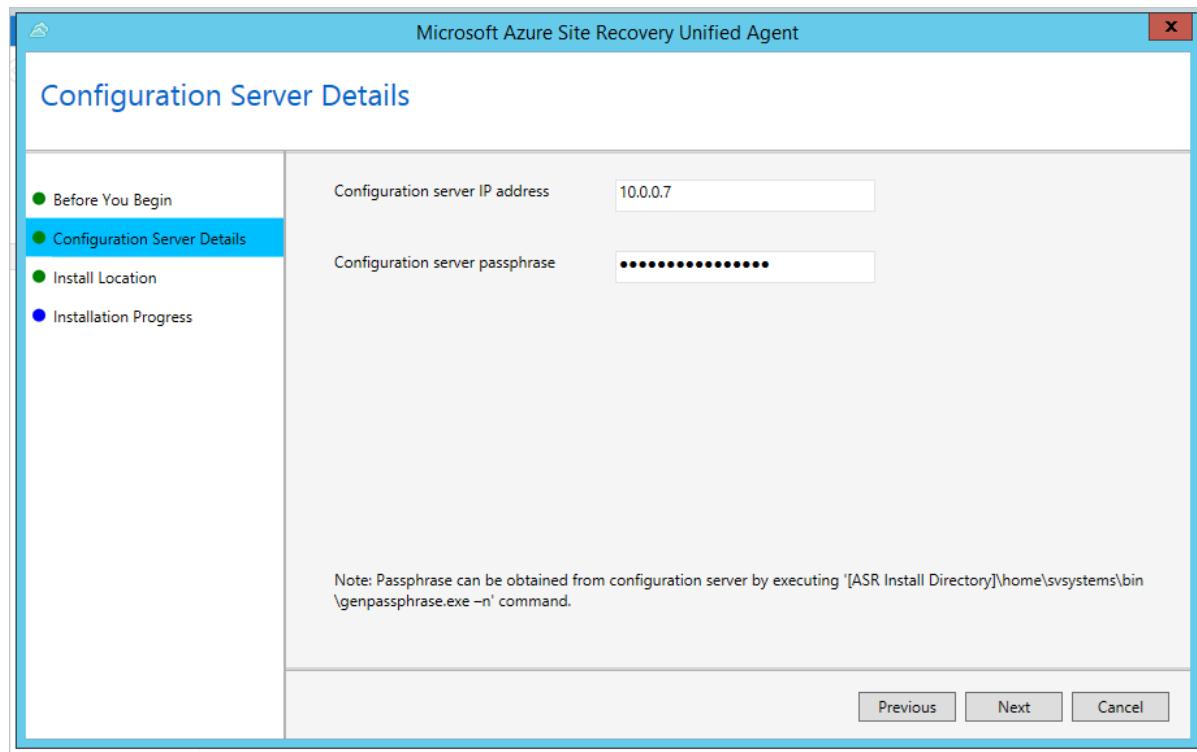
Step 6: Plan capacity

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.



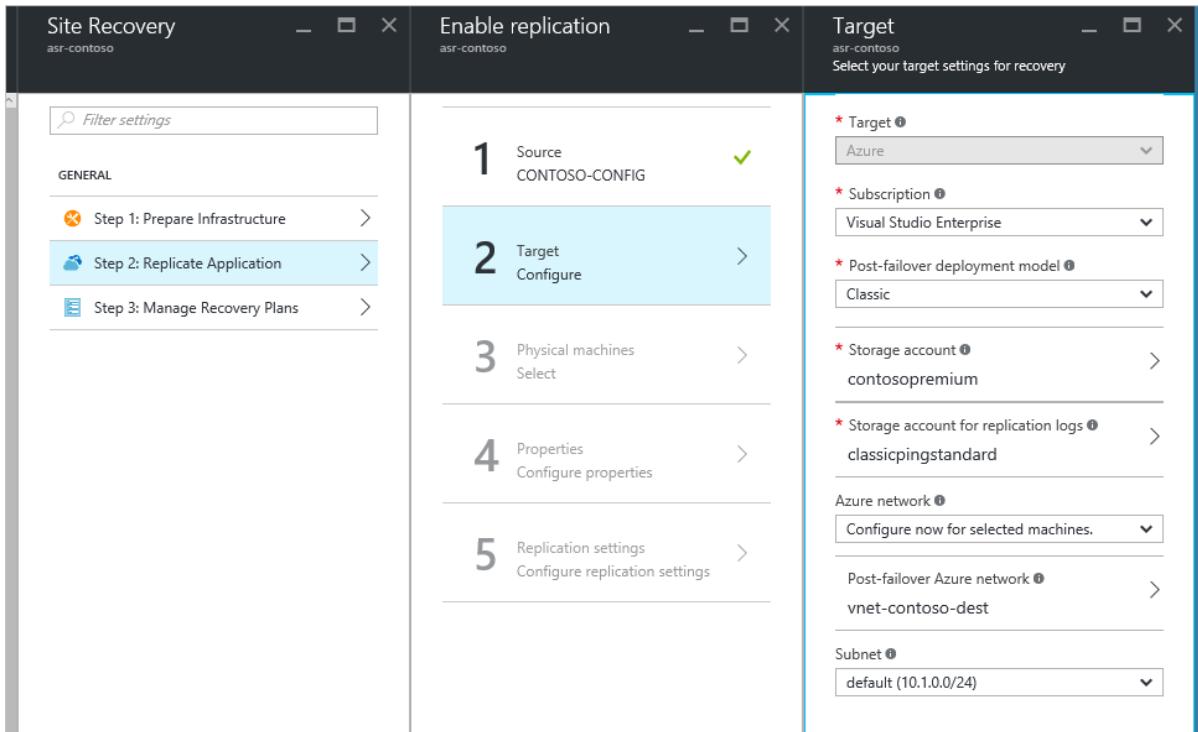
The failed-over VM will have two temporary disks: one from the primary VM and the other created during the provisioning of the VM in the recovery region. To exclude the temporary disk before replication, install the mobility service before you enable replication. To learn more about how to exclude the 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](#)
- [Premium Storage: High-performance storage for Azure virtual machine workloads](#)

Find and delete unattached Azure managed and unmanaged disks

4/9/2018 • 2 min 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=1

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

4/13/2018 • 4 min to read • [Edit Online](#)

This article shows you how to utilize the Azure File storage service on a Linux VM using an SMB mount with the Azure CLI 2.0. Azure File storage offers file shares in the cloud using the standard SMB protocol. You can also perform these steps with the [Azure CLI 1.0](#). The requirements are:

- [an Azure account](#)
- [SSH public and private key files](#)

Quick Commands

- A resource group
- An Azure virtual network
- A network security group with an SSH inbound
- A subnet
- An Azure storage account
- Azure storage account keys
- An Azure File storage share
- A Linux VM

Replace any examples with your own settings.

Create a directory for the local mount

```
mkdir -p /mnt/mymountpoint
```

Mount the File storage SMB share to the mount point

```
sudo mount -t cifs //myaccountname.file.core.windows.net/mysharename /mnt/mymountpoint -o  
vers=3.0,username=myaccountname,password=StorageAccountKeyEndingIn==,dir_mode=0777,file_mode=0777
```

Persist the mount after a reboot

To do so, add the following line to the `/etc/fstab`:

```
//myaccountname.file.core.windows.net/mysharename /mnt/mymountpoint cifs  
vers=3.0,username=myaccountname,password=StorageAccountKeyEndingIn==,dir_mode=0777,file_mode=0777
```

Detailed walkthrough

File storage offers file shares in the cloud that use the standard SMB protocol. With the latest release of File storage, you can also 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. That's because 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.

For this detailed walkthrough, we create the prerequisites needed to first create the File storage share, and then mount it via SMB on a Linux VM.

1. Create a resource group with [az group create](#) to hold the file share.

To create a resource group named `myResourceGroup` in the "West US" location, use the following example:

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

2. Create an Azure storage account with [az storage account create](#) to store the actual files.

To create a storage account named `mystorageaccount` by using the Standard_LRS storage SKU, use the following example:

```
az storage account create --resource-group myResourceGroup \
    --name mystorageaccount \
    --location westus \
    --sku Standard_LRS
```

3. Show the storage account keys.

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, ensuring that you always have at least one unused storage account key ready to switch to.

View the storage account keys with the [az storage account keys list](#). The storage account keys for the named `mystorageaccount` are listed in the following example:

```
az storage account keys list --resource-group myResourceGroup \
    --account-name mystorageaccount
```

To extract a single key, use the `--query` flag. The following example extracts the first key (`[0]`):

```
az storage account keys list --resource-group myResourceGroup \
    --account-name mystorageaccount \
    --query '[0].{Key:value}' --output tsv
```

4. Create the File storage share.

The File storage share contains the SMB share with [az storage share create](#). The quota is always expressed in gigabytes (GB). Pass in one of the keys from the preceding [az storage account keys list](#) command.

Create a share named `mystorageshare` with a 10-GB quota by using the following example:

```
az storage share create --name mystorageshare \
    --quota 10 \
    --account-name mystorageaccount \
    --account-key nP0gPR<--snip-->4Q==
```

5. Create a mount-point directory.

Create a local directory in the Linux file system to mount the SMB share to. Anything written or read from the local mount directory is forwarded to the SMB share that's hosted on File storage. To create a local directory at `/mnt/mymountdirectory`, use the following example:

```
sudo mkdir -p /mnt/mymountpoint
```

6. Mount the SMB share to the local directory.

Provide your own storage account username and storage account key for the mount credentials as follows:

```
sudo mount -t cifs //myStorageAccount.file.core.windows.net/mystorageshare /mnt/mymountpoint -o  
vers=3.0,username=mystorageaccount,password=mystorageaccountkey,dir_mode=0777,file_mode=0777
```

7. Persist the SMB mount through reboots.

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=StorageAccountKeyEndingIn==,dir_mode=0777,file_mode=0777
```

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](#)

Using Managed Disks in Azure Resource Manager Templates

8/21/2017 • 4 min 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. This will 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, we 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 will show you how to create a new storage account. To accomplish this, you need a storage account resource in the resources block as shown below.

```
{  
  "type": "Microsoft.Storage/storageAccounts",  
  "name": "[variables('storageAccountName')]",  
  "apiVersion": "2016-01-01",  
  "location": "[resourceGroup().location]",  
  "sku": {  
    "name": "Standard_LRS"  
  },  
  "kind": "Storage",  
  "properties": {}  
}
```

Within the virtual machine object, we need a dependency on the storage account to ensure that it's created before the virtual machine. Within the `storageProfile` section, we then specify the full URI of the VHD location, which references the storage account and is needed for the OS disk and any data disks.

```
{
    "apiVersion": "2015-06-15",
    "type": "Microsoft.Compute/virtualMachines",
    "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 [Standard LRS storage](#). 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.

```
{
    "apiVersion": "2017-03-30",
    "type": "Microsoft.Compute/virtualMachines",
    "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, we can create a disk resource as follows to use as a data disk.

```
{
    "type": "Microsoft.Compute/disks",
    "name": "[concat(variables('vmName'), '-datadisk1')]",
    "apiVersion": "2017-03-30",
    "location": "[resourceGroup().location]",
    "sku": {
        "name": "Standard_LRS"
    },
    "properties": {
        "creationData": {
            "createOption": "Empty"
        },
        "diskSizeGB": 1023
    }
}
```

Within the VM object, we can then reference this disk object to be attached. Specifying the resource ID of the managed disk we created in the `managedDisk` property allows the attachment of the disk as the VM is created. Note that the `apiVersion` for the VM resource is set to `2017-03-30`. Also note that we've created a dependency on the disk resource to ensure it's successfully created before VM creation.

```
{  
    "apiVersion": "2017-03-30",  
    "type": "Microsoft.Compute/virtualMachines",  
    "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 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 `2017-03-30`.

```
{  
    "apiVersion": "2017-03-30",  
    "type": "Microsoft.Compute/availabilitySets",  
    "location": "[resourceGroup().location]",  
    "name": "[variables('avSetName')]",  
    "properties": {  
        "PlatformUpdateDomainCount": 3,  
        "PlatformFaultDomainCount": 2  
    },  
    "sku": {  
        "name": "Aligned"  
    }  
}
```

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.

Optimize your Linux VM on Azure

5/10/2018 • 6 min 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 2.0](#) 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. For more information about Premium Storage disks, see [Premium Storage: High-Performance Storage for Azure VMs](#)

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 2.0 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 Redhat 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:

- [Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads](#)
- [Azure Linux Agent User Guide](#)
- [Optimizing MySQL Performance on Azure Linux VMs](#)
- [Configure Software RAID on Linux](#)

Configure Software RAID on Linux

4/9/2018 • 5 min 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

4/9/2018 • 4 min to read • [Edit Online](#)

This document will discuss how to configure Logical Volume Manager (LVM) in your Azure virtual machine. While it is feasible to configure LVM on any disk attached to the virtual machine, by default most cloud images will not have LVM configured on the OS disk. This is to prevent problems with duplicate volume groups if the OS disk is ever attached to another VM of the same distribution and type, i.e. during a recovery scenario. Therefore it is recommended only to use LVM on the 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 (i.e. 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 somewhat 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, please 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. Please 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 lvm2
```

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`. Note that these may not always be the same 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, please 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 please 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. Please 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
```

4 min to read •

Open ports and endpoints to a Linux VM with the Azure CLI

3/8/2018 • 3 min 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 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

To create a Network Security Group and rules you need the latest [Azure CLI 2.0](#) 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 VM with a static public IP address using the Azure CLI

4/16/2018 • 5 min to read • [Edit Online](#)

You can create virtual machines (VMs) in Azure and expose them to the public Internet by using a public IP address. By default, Public IPs are dynamic and the address associated to them may change when the VM is deleted or stopped/deallocated. To guarantee that the VM always uses the same public IP address, you need to create a static Public IP.

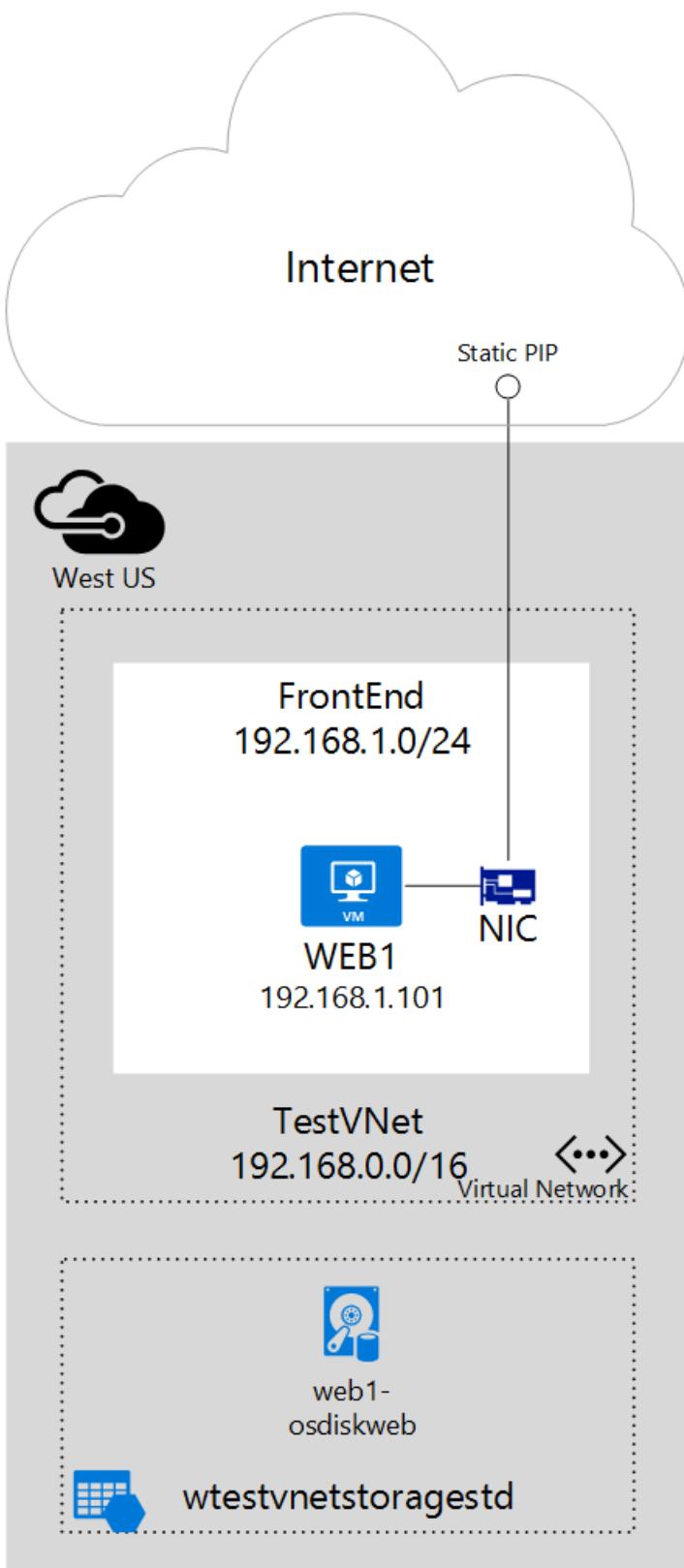
Before you can implement static Public IPs in VMs, it is necessary to understand when you can use static Public IPs, and how they are used. Read the [IP addressing overview](#) to learn more about IP addressing in Azure.

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the Resource Manager deployment model, which Microsoft recommends for most new deployments instead of the classic deployment model.

Scenario

This document will walk through a deployment that uses a static public IP address allocated to a virtual machine (VM). In this scenario, you have a single VM with its own static public IP address. The VM is part of a subnet named **FrontEnd** and also has a static private IP address (**192.168.1.101**) in that subnet.

You may need a static IP address for web servers that require SSL connections in which the SSL certificate is linked to an IP address.



You can follow the steps below to deploy the environment shown in the figure above.

Create the VM

The values in "" for the variables in the steps that follow create resources with settings from the scenario. Change the values, as appropriate, for your environment.

1. Install the [Azure CLI 2.0](#) if you don't already have it installed.
2. Create an SSH public and private key pair for Linux VMs by completing the steps in the [Create an SSH public and private key pair for Linux VMs](#).
3. From a command shell, login with the command `az login`.

4. Create the VM by executing the script that follows on a Linux or Mac computer. The Azure public IP address, virtual network, network interface, and VM resources must all exist in the same location. Though the resources don't all have to exist in the same resource group, in the following script they do.

```
RgName="IaaSStory"
Location="westus"

# Create a resource group.

az group create \
--name $RgName \
--location $Location

# Create a public IP address resource with a static IP address using the --allocation-method Static option.
# If you do not specify this option, the address is allocated dynamically. The address is assigned to the
# resource from a pool of IP addresses unique to each Azure region. The DnsName must be unique within the
# Azure location it's created in. Download and view the file from https://www.microsoft.com/en-
us/download/details.aspx?id=41653#
# that lists the ranges for each region.

PipName="PIPWEB1"
DnsName="iaasstoryws1"
az network public-ip create \
--name $PipName \
--resource-group $RgName \
--location $Location \
--allocation-method Static \
--dns-name $DnsName

# Create a virtual network with one subnet

VnetName="TestVNet"
VnetPrefix="192.168.0.0/16"
SubnetName="FrontEnd"
SubnetPrefix="192.168.1.0/24"
az network vnet create \
--name $VnetName \
--resource-group $RgName \
--location $Location \
--address-prefix $VnetPrefix \
--subnet-name $SubnetName \
--subnet-prefix $SubnetPrefix

# Create a network interface connected to the VNet with a static private IP address and associate the public
# IP address
# resource to the NIC.

NicName="NICWEB1"
PrivateIpAddress="192.168.1.101"
az network nic create \
--name $NicName \
--resource-group $RgName \
--location $Location \
--subnet $SubnetName \
--vnet-name $VnetName \
--private-ip-address $PrivateIpAddress \
--public-ip-address $PipName

# Create a new VM with the NIC

VmName="WEB1"

# Replace the value for the VmSize variable with a value from the
# https://docs.microsoft.com/azure/virtual-machines/virtual-machines-linux-sizes article.
VmSize="Standard_DS1"

# Replace the value for the OsImage variable with a value for *urn* from the output returned by entering
```

```

# the `az vm image list` command.

$OsImage="credativ:Debian:8:latest"
$Username='adminuser'

# Replace the following value with the path to your public key file.
$SshKeyValue="~/.ssh/id_rsa.pub"

az vm create \
--name $VmName \
--resource-group $RgName \
--image $OsImage \
--location $Location \
--size $VmSize \
--nics $NicName \
--admin-username $Username \
--ssh-key-value $SshKeyValue
# If creating a Windows VM, remove the previous line and you'll be prompted for the password you want to
configure for the VM.

```

In addition to creating a VM, the script creates:

- A single premium managed disk by default, but you have other options for the disk type you can create. Read the [Create a Linux VM using the Azure CLI 2.0](#) article for details.
- Virtual network, subnet, NIC, and public IP address resources. Alternatively, you can use *existing* virtual network, subnet, NIC, or public IP address resources. To learn how to use existing network resources rather than creating additional resources, enter `az vm create -h`.

Validate VM creation and public IP address

1. Enter the command `az resource list --resource-group IaaSStory --output table` to see a list of the resources created by the script. There should be five resources in the returned output: network interface, disk, public IP address, virtual network, and a virtual machine.
2. Enter the command `az network public-ip show --name PIPWEB1 --resource-group IaaSStory --output table`. In the returned output, note the value of **IpAddress** and that the value of **PublicIpAllocationMethod** is *Static*.
3. Before executing the following command, remove the <>, replace *Username* with the name you used for the **Username** variable in the script, and replace *ipAddress* with the **IpAddress** from the previous step. Run the following command to connect to the VM: `ssh -i ~/.ssh/azure_id_rsa <Username>@<ipAddress>`.

Remove the VM and associated resources

It's recommended that you delete the resources created in this exercise if you won't use them in production. VM, public IP address, and disk resources incur charges, as long as they're provisioned. To remove the resources created during this exercise, complete the following steps:

1. To view the resources in the resource group, run the `az resource list --resource-group IaaSStory` command.
2. Confirm there are no resources in the resource group, other than the resources created by the script in this article.
3. To delete all resources created in this exercise, run the `az group delete -n IaaSStory` command. The command deletes the resource group and all the resources it contains.

Set IP addresses within the operating system

You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's operating system. It's recommended that you do not statically assign the private IP assigned to the Azure virtual machine within the operating system of a VM, unless necessary, such as when [assigning multiple IP addresses to a Windows VM](#). If you do manually set the private IP address within the operating system, ensure

that it is the same address as the private IP address assigned to the Azure [network interface](#), or you can lose connectivity to the virtual machine. Learn more about [private IP address](#) settings.

Next steps

Any network traffic can flow to and from the VM created in this article. You can define inbound and outbound security rules within a network security group that limit the traffic that can flow to and from the network interface, the subnet, or both. To learn more about network security groups, see [Network security group overview](#).

How to create a Linux virtual machine in Azure with multiple network interface cards

4/16/2018 • 5 min to read • [Edit Online](#)

You can create a virtual machine (VM) in Azure that has multiple virtual network interfaces (NICs) attached to it. A common scenario is to have different subnets for front-end and back-end connectivity, or a network dedicated to a monitoring or backup solution. This article details how to create a VM with multiple NICs attached to it and how to add or remove NICs from an existing VM. Different [VM sizes](#) support a varying number of NICs, so size your VM accordingly.

This article details how to create a VM with multiple NICs with the Azure CLI 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

Create supporting resources

Install the latest [Azure CLI 2.0](#) 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 192.168.0.0/16 \
--subnet-name mySubnetFrontEnd \
--subnet-prefix 192.168.1.0/24
```

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 192.168.2.0/24
```

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 2.0. 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

When you add multiple NICs to a Linux VM, you need to create routing rules. These rules allow the VM to send and receive traffic that belongs to a specific NIC. Otherwise, traffic that belongs to *eth1*, for example, cannot be processed correctly by the defined default route.

To correct this routing issue, first add two routing tables to */etc/iproute2/rt_tables* as follows:

```
echo "200 eth0-rt" >> /etc/iproute2/rt_tables
echo "201 eth1-rt" >> /etc/iproute2/rt_tables
```

To make the change persistent and applied during network stack activation, edit */etc/sysconfig/network-scripts/ifcfg-eth0* and */etc/sysconfig/network-scripts/ifcfg-eth1*. Alter the line "*NM_CONTROLLED=yes*" to "*NM_CONTROLLED=no*". Without this step, the additional rules/routing are not automatically applied.

Next, extend the routing tables. Let's assume we have the following setup in place:

Routing

```
default via 10.0.1.1 dev eth0 proto static metric 100
10.0.1.0/24 dev eth0 proto kernel scope link src 10.0.1.4 metric 100
10.0.1.0/24 dev eth1 proto kernel scope link src 10.0.1.5 metric 101
168.63.129.16 via 10.0.1.1 dev eth0 proto dhcp metric 100
169.254.169.254 via 10.0.1.1 dev eth0 proto dhcp metric 100
```

Interfaces

```
lo: inet 127.0.0.1/8 scope host lo
eth0: inet 10.0.1.4/24 brd 10.0.1.255 scope global eth0
eth1: inet 10.0.1.5/24 brd 10.0.1.255 scope global eth1
```

You would then create the following files and add the appropriate rules and routes to each:

- */etc/sysconfig/network-scripts/rule-eth0*

```
from 10.0.1.4/32 table eth0-rt
to 10.0.1.4/32 table eth0-rt
```

- */etc/sysconfig/network-scripts/route-eth0*

```
10.0.1.0/24 dev eth0 table eth0-rt
default via 10.0.1.1 dev eth0 table eth0-rt
```

- */etc/sysconfig/network-scripts/rule-eth1*

```
from 10.0.1.5/32 table eth1-rt
to 10.0.1.5/32 table eth1-rt
```

- `/etc/sysconfig/network-scripts/route-eth1`

```
10.0.1.0/24 dev eth1 table eth1-rt
default via 10.0.1.1 dev eth1 table eth1-rt
```

To apply the changes, restart the *network* service as follows:

```
systemctl restart network
```

The routing rules are now correctly in place and you can connect with either interface as needed.

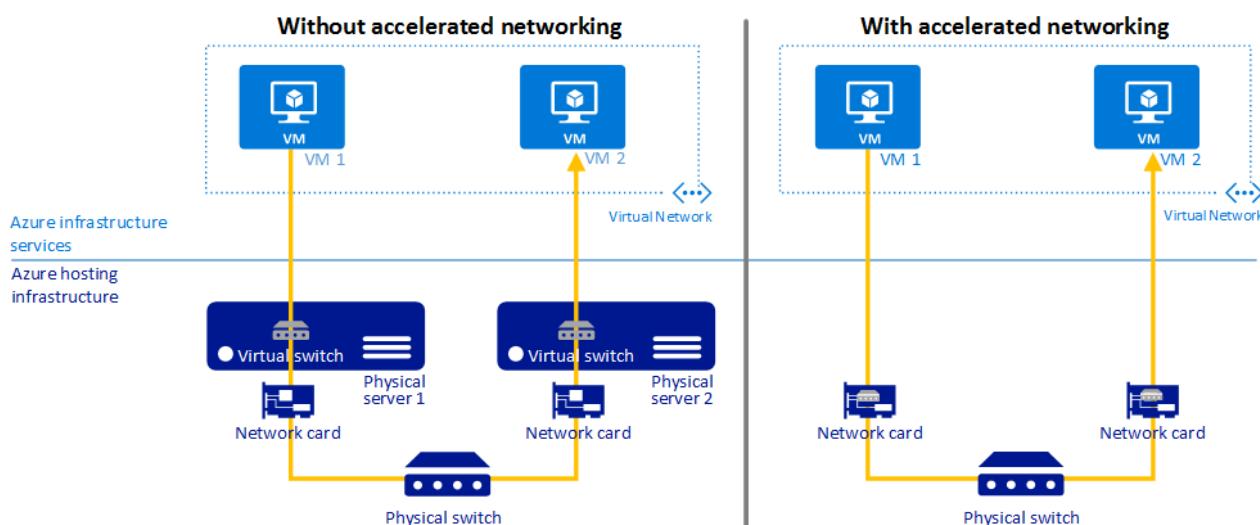
Next steps

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.

Create a Linux virtual machine with Accelerated Networking

5/7/2018 • 9 min 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 VM'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 16.04**
- **SLES 12 SP3**
- **RHEL 7.4**
- **CentOS 7.4**
- **CoreOS Linux**
- **Debian "Stretch" with backports kernel**
- **Oracle Linux 7.4**

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, and Ms/Mms.

For more information on VM instances, see [Linux VM sizes](#).

Regions

Available in all public Azure regions as well as Azure Government Clouds.

Network interface creation

Accelerated networking can only be enabled for a new NIC. It cannot be enabled for an existing NIC.

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

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, under **Settings**, select **Enabled**, under **Accelerated networking**. The option to enable accelerated networking doesn't appear in the portal unless you've selected a [supported operating system](#) and [VM size](#). After the virtual machine is created, you need to complete the instructions in [Confirm that accelerated networking is enabled](#).

Create a virtual network

Install the latest [Azure CLI 2.0](#) 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.

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 myVM -n 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

12/14/2017 • 1 min 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. The *DNS name* is currently blank. Select **Public IP address**:

The screenshot shows the Azure portal interface for a VM named 'myVM'. In the left sidebar, under 'SETTINGS', the 'Configuration' option is selected. In the main content area, the 'Essentials' tab is active. The 'Public IP address' section is highlighted with a red box. It shows the IP address '40.87.61.246'. Other details include the Resource group ('myResourceGroup'), Status ('Running'), Location ('East US'), Subscription ('Standard DS1 v2 (1 core, 3.5 GB memory)'), and Virtual network/subnet ('myVMVNET/myVMSubnet').

2. Enter the desired DNS name and then select **Save**.

The screenshot shows the 'myVMPublicIP - Configuration' blade. Under 'Assignment', the 'Dynamic' tab is selected. The 'IP address' field shows '40.87.61.246'. The 'Idle timeout (minutes)' slider is set to '4'. The 'DNS name label (optional)' field contains 'myazuredns'. 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.

The screenshot shows the Azure portal interface for a virtual machine named 'myVM'. The left sidebar has a 'Search (Ctrl+ /)' bar and links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS, and Availability set. The main pane is titled 'Essentials' and displays resource group 'myResourceGroup', status 'Running', location 'East US', subscription information, and network details. A red box highlights the 'DNS name' field, which contains 'myazuredns.eastus.cloudapp.azure.com'.

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

4/11/2018 • 1 min 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

4/9/2018 • 7 min 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.

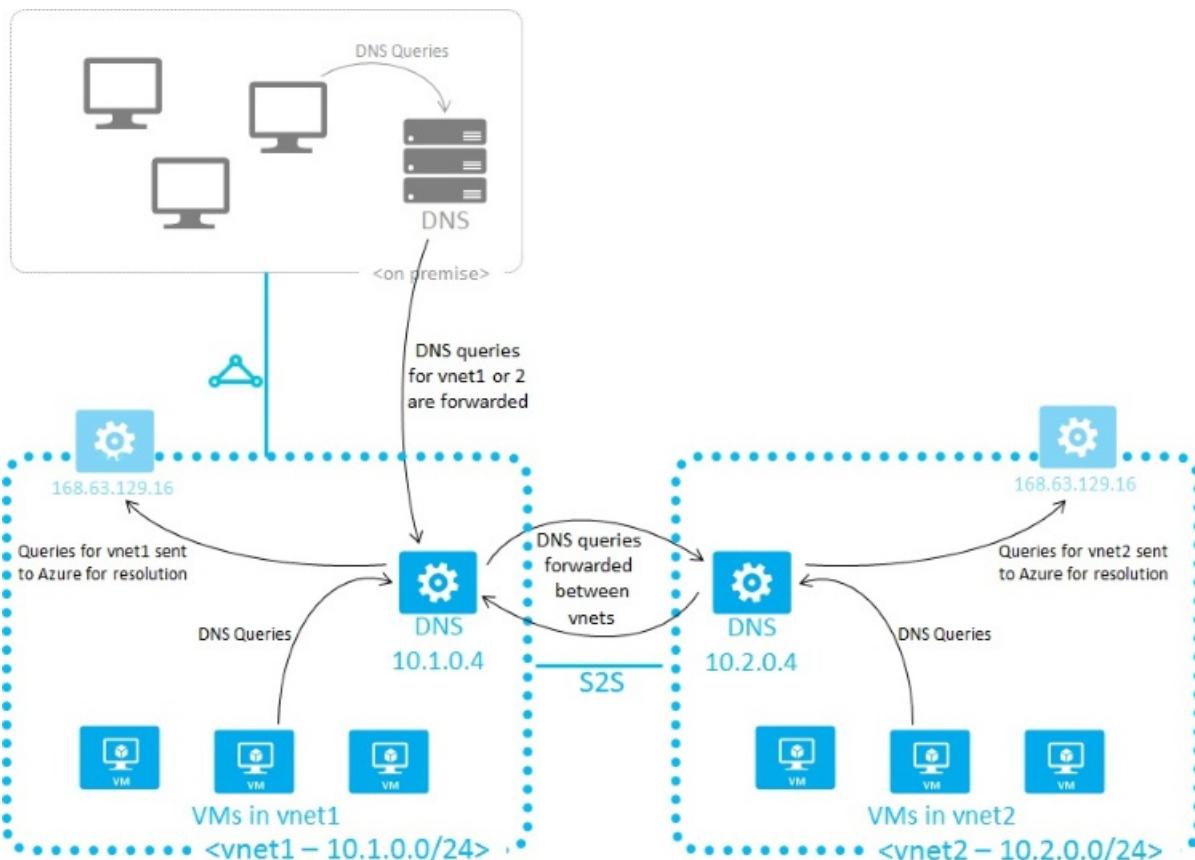
Name resolution using your own DNS server

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

4/9/2018 • 5 min 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 2.0. You can also perform these steps with the [Azure CLI 1.0](#). 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 2.0](#) 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](#)

4 min to read •

Use Azure Policy to restrict extensions installation on Linux VMs

5/10/2018 • 3 min to read • [Edit Online](#)

If you want to prevent the use or installation of certain extensions on your Linux VMs, you can create an Azure policy using the CLI to restrict extensions for VMs within a resource group.

This tutorial uses the CLI within the Azure Cloud Shell, which is constantly updated to the latest version. If you want to run the Azure CLI locally, you need to install version 2.0.26 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI 2.0](#).

Create a rules file

In order to restrict what extensions can be installed, you need to have a [rule](#) to provide the logic to identify the extension.

This example shows you how to deny installing extensions published by 'Microsoft.OSTCExtensions' by creating a rules file in Azure Cloud Shell, but if you are working in CLI locally, you can also create a local file and replace the path (~/.clouddrive) with the path to the local file on your machine.

In a [bash Cloud Shell](#), type:

```
vim ~/.clouddrive/azurepolicy.rules.json
```

Copy and paste the following JSON into the file.

```
{
  "if": {
    "allOf": [
      {
        "field": "type",
        "equals": "Microsoft.OSTCExtensions/virtualMachines/extensions"
      },
      {
        "field": "Microsoft.OSTCExtensions/virtualMachines/extensions/publisher",
        "equals": "Microsoft.OSTCExtensions"
      },
      {
        "field": "Microsoft.OSTCExtensions/virtualMachines/extensions/type",
        "in": "[parameters('notAllowedExtensions')]"
      }
    ],
    "then": {
      "effect": "deny"
    }
  }
}
```

When you are done, hit the **Esc** key and then type **:wq** to save and close the file.

Create a parameters file

You also need a [parameters](#) file that creates a structure for you to use for passing in a list of the extensions to block.

This example shows you how to create a parameters file for Linux VMs in Cloud Shell, but if you are working in CLI locally, you can also create a local file and replace the path (~/clouddrive) with the path to the local file on your machine.

In the [bash Cloud Shell](#), type:

```
vim ~/clouddrive/azurepolicy.parameters.json
```

Copy and paste the following .json into the file.

```
{
  "notAllowedExtensions": {
    "type": "Array",
    "metadata": {
      "description": "The list of extensions that will be denied. Example: CustomScriptForLinux, VMAccessForLinux etc.",
      "strongType": "type",
      "displayName": "Denied extension"
    }
  }
}
```

When you are done, hit the **Esc** key and then type :**wq** to save and close the file.

Create the policy

A policy definition is an object used to store the configuration that you would like to use. The policy definition uses the rules and parameters files to define the policy. Create the policy definition using [az policy definition create](#).

In this example, the rules and parameters are the files you created and stored as .json files in your cloud shell.

```
az policy definition create \
--name 'not-allowed-vmextension-linux' \
--display-name 'Block VM Extensions' \
--description 'This policy governs which VM extensions that are blocked.' \
--rules '~/clouddrive/azurepolicy.rules.json' \
--params '~/clouddrive/azurepolicy.parameters.json' \
--mode All
```

Assign the policy

This example assigns the policy to a resource group using [az policy assignment create](#). Any VM created in the **myResourceGroup** resource group will not be able to install the Linux VM Access or the Custom Script extensions for Linux. The resource group must exist before you can assign the policy.

Use [az account list](#) to get your subscription ID to use in place of the one in the example.

```
az policy assignment create \
--name 'not-allowed-vmextension-linux' \
--scope /subscriptions/<subscription Id>/resourceGroups/myResourceGroup \
--policy "not-allowed-vmextension-linux" \
--params '{
    "notAllowedExtensions": {
        "value": [
            "VMAccessForLinux",
            "CustomScriptForLinux"
        ]
    }
}'
```

Test the policy

Test the policy by creating a new VM and trying to add a new user.

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

Try to create a new user named **myNewUser** using the VM Access extension.

```
az vm user update \
--resource-group myResourceGroup \
--name myVM \
--username myNewUser \
--password 'mynewuserpwd123!'
```

Remove the assignment

```
az policy assignment delete --name 'not-allowed-vmextension-linux' --resource-group myResourceGroup
```

Remove the policy

```
az policy definition delete --name 'not-allowed-vmextension-linux'
```

Next steps

For more information, see [Azure Policy](#).

Migrate from Amazon Web Services (AWS) and other platforms to Managed Disks in Azure

4/25/2018 • 4 min 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 two types of storage options:

- [Premium Managed Disks](#) are Solid State Drive (SSD) based storage media, which delivers high performance, low-latency disk support for virtual machines running I/O-intensive workloads. You can take advantage of the speed and performance of these disks by migrating to Premium Managed Disks.
- [Standard Managed Disks](#) use Hard Disk Drive (HDD) based storage media and are best suited for Dev/Test and other infrequent access workloads that are less sensitive to performance variability.

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 three types of Premium Managed disks that can be used with your VM and each has specific IOPs and throughput limits. Consider 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	P10	P20	P30
Disk size	128 GB	512 GB	1024 GB (1 TB)
IOPS per disk	500	2300	5000
Throughput per disk	100 MB per second	150 MB per second	200 MB per second

Standard Managed Disks

There are five 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	S20	S30
Disk size	30 GB	64 GB	128 GB	512 GB	1024 GB (1 TB)
IOPS per disk	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 Azure using PowerShell

4/9/2018 • 2 min 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 topic 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 described in the Amazon documentation topic [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 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 custom disk with the Azure CLI 2.0

4/9/2018 • 6 min to read • [Edit Online](#)

This article shows you how to upload a customized virtual hard disk (VHD) or copy a an existing VHD in Azure and create new Linux virtual machines (VMs) from the custom disk. You can install and configure a Linux distro to your requirements and then use that VHD to quickly create a new Azure virtual machine.

If you want to create multiple VMs from your customized disk, you should create an image from your VM or VHD. For more information, see [Create a custom image of an Azure VM using the CLI](#).

You have two options:

- [Upload a VHD](#)
- [Copy an existing Azure VM](#)

Quick commands

When creating a new VM using `az vm create` from a customized or specialized disk you **attach** the disk (`--attach-os-disk`) instead of 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 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 required for the VM to work properly in Azure and for you to be able to connect to it using 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](#) using `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 using `qemu-img convert` or the `Convert-VHD` PowerShell cmdlet. Further, Azure does not support uploading dynamic VHDs, so you 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 to Azure.

- Make sure that you have the latest [Azure CLI 2.0](#) 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 included *myResourceGroup*, *mystorageaccount*, and *mydisks*.

Prepare the VM

Azure supports various Linux distributions (see [Endorsed Distributions](#)). The following articles guide you through 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](#)
- [Other - 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 the VHD to create a new Azure VM, you need to upload the VHD to a storage account and create a managed disk from the VHD.

Create a resource group

Before uploading your custom disk and creating VMs, you first need to create a resource group with [az group create](#).

The following example creates a resource group named *myResourceGroup* in the *eastus* location: [Azure Managed Disks overview](#)

```
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, like carrying out write operations. Read more about [managing access to](#)

[storage here](#). You view the access keys with [az storage account keys list](#).

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 Ww0T7g4UyYLaBnLYcxIOTVziGAHVu+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 create containers within a storage account to organize your disks. A storage account can contain any number of 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

Now upload your custom disk with [az storage blob upload](#). You 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 using [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 the 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, you really should 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 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, that is 128GB in size in standard storage.

```
az disk create \
    --resource-group myResourceGroup \
    --name myManagedDisk \
    --sku Standard_LRS \
    --size-gb 128 \
    --source $snapshotId
```

Create the VM

Now, 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 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 using 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](#).

4 min to read •

Platform-supported migration of IaaS resources from classic to Azure Resource Manager

4/9/2018 • 7 min to read • [Edit Online](#)

In this article, we describe how we're enabling migration of infrastructure as a service (IaaS) resources from the Classic to Resource Manager deployment models. You can read more about [Azure Resource Manager features and benefits](#). We detail how to connect resources from the two deployment models that coexist in your subscription by using virtual network site-to-site gateways.

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
- 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 4 different ways to complete migration of compute, network, and storage resources. These are

- Migration of virtual machines (NOT in a virtual network)
- Migration of virtual machines (in a virtual network)
- Storage accounts migration
- Unattached resources (Network Security Groups, Route Tables & Reserved IPs)

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.

Storage accounts migration

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.

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.

Unattached resources (Network Security Groups, Route Tables & Reserved IPs)

Network Security Groups, Route Tables & Reserved IPs that are not attached to any Virtual Machines and Virtual Networks can be migrated independently.

Unsupported features and configurations

We do not currently support some features and configurations. 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.

Service	Configuration	Recommendation
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.
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, please 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. To migrate these Virtual Machines, follow the guidance here .
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.

Service	Configuration	Recommendation
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 about this 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 RemoteApp	Virtual networks that contain Azure RemoteApp deployments	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, please 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 min to read • [Edit Online](#)

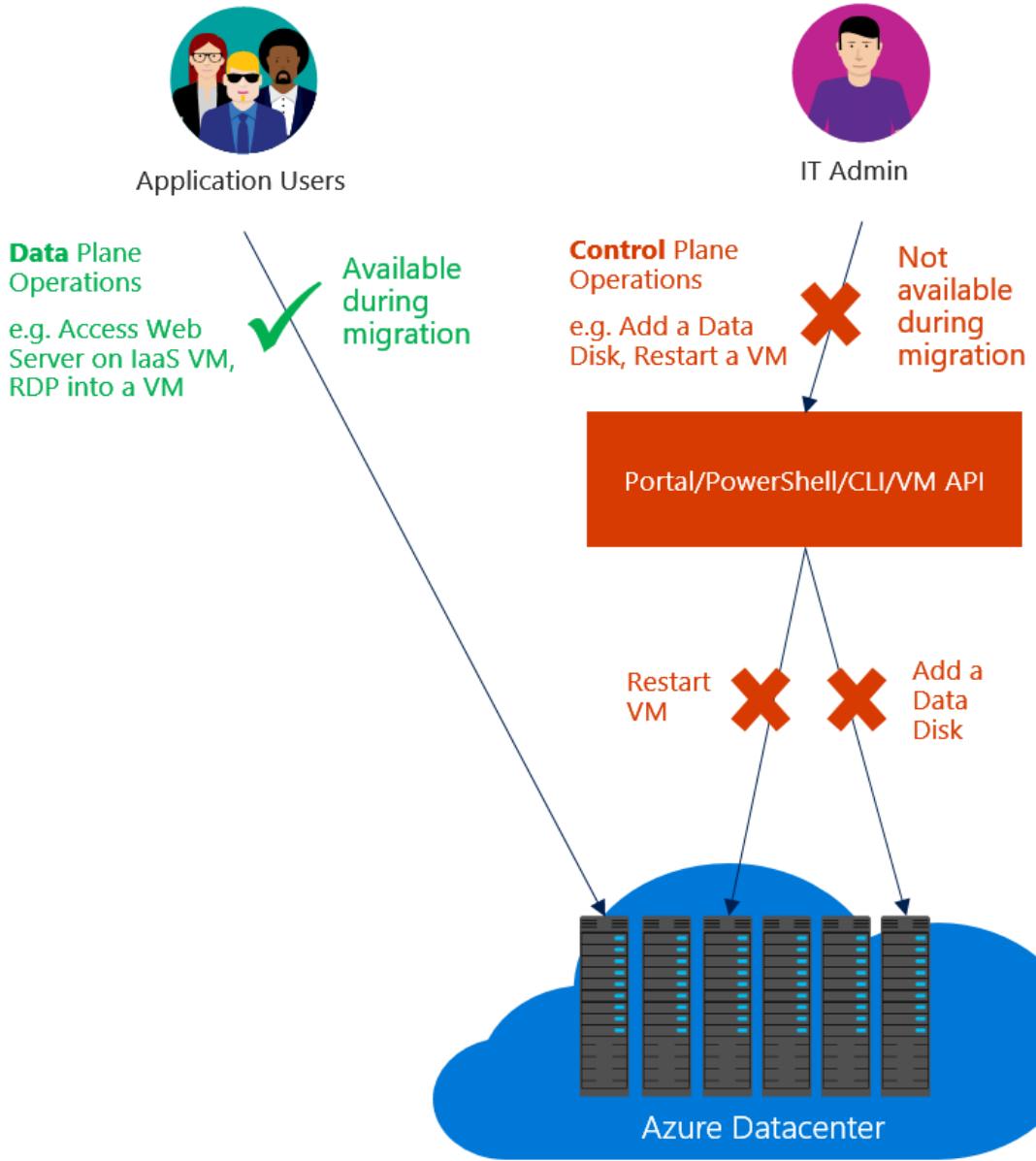
Let's take a deep-dive on migrating from the Azure classic deployment model to the Azure Resource Manager deployment model. We look at resources at a resource and feature level to help you understand how the Azure platform migrates resources between the two deployment models. For more information, please 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

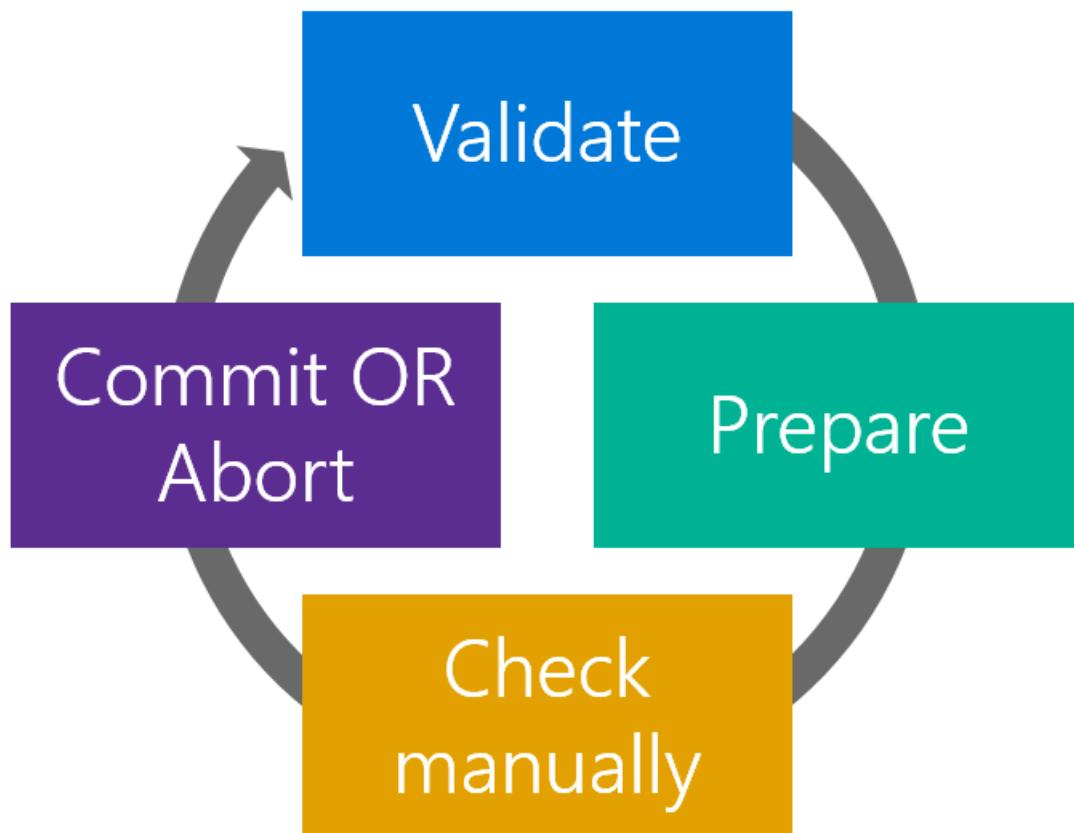
Before you start the migration:

- 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 No deployments Location 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 2 Succeeded Location East US

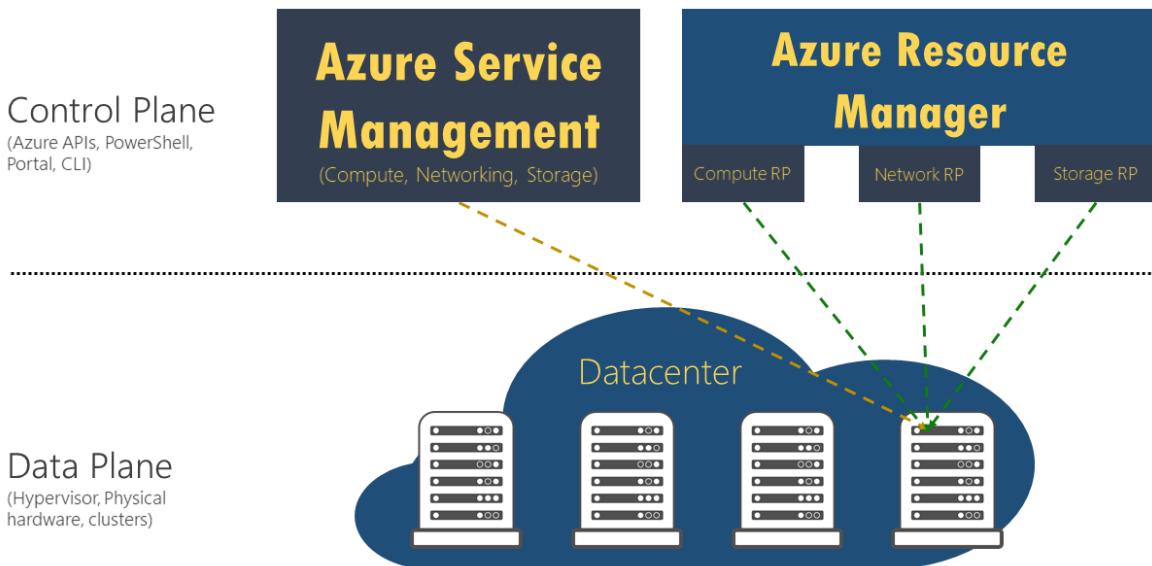
Filter by name...

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

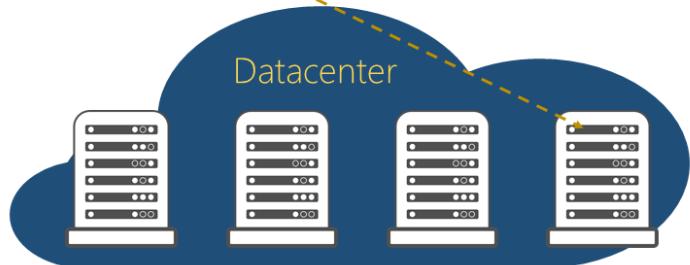
This is an optional step if you want to revert your changes to the classic deployment model and stop the migration. This operation deletes the Resource Manager metadata (created in the prepare step) for your resources.

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

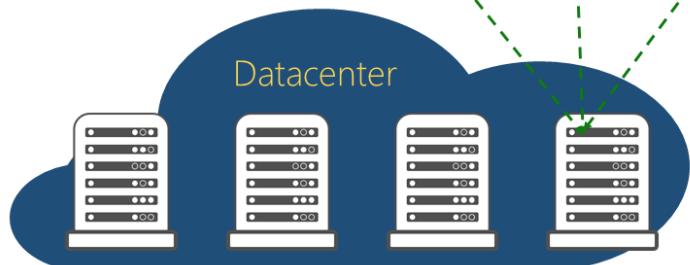
This is an idempotent operation. If it fails, retry the operation. If it continues to fail, create a support ticket or create a forum post with a "ClassiclaaSMigration" tag on our [VM forum](#).

Commit

Control Plane
(Azure APIs, PowerShell, Portal, CLI)



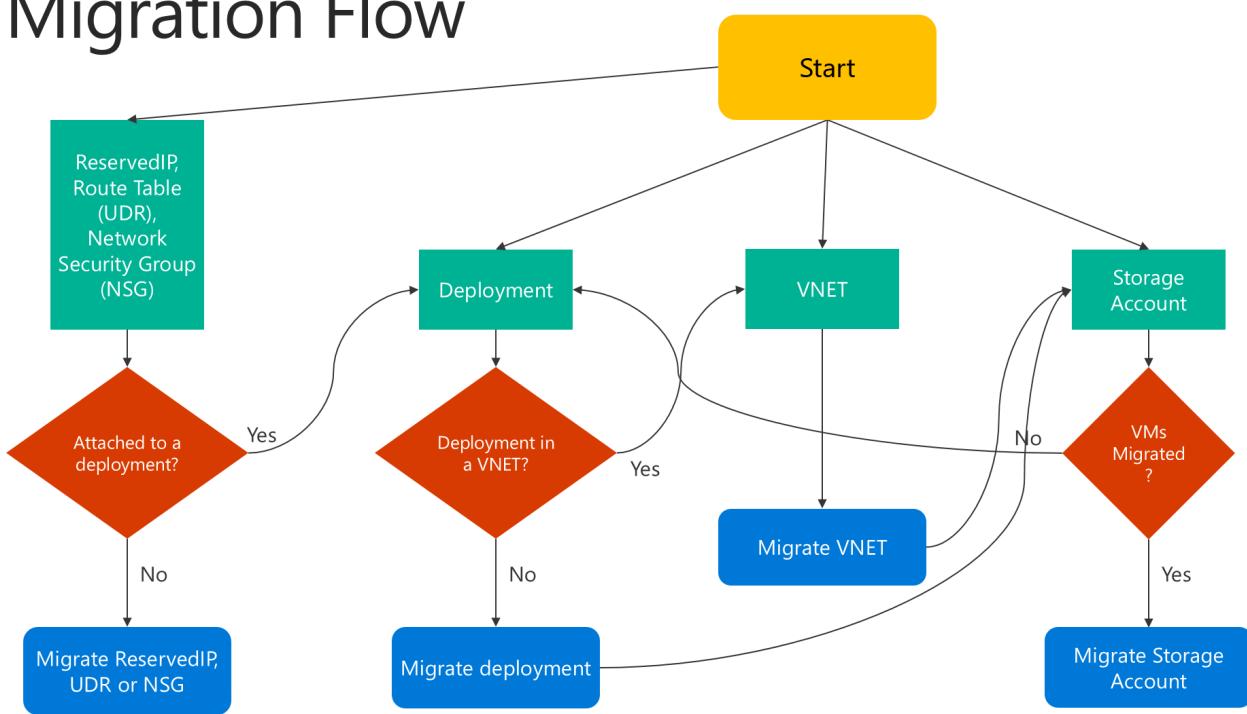
Data Plane
(Hypervisor, Physical hardware, clusters)



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

4/9/2018 • 13 min 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 above 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 2.0.

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

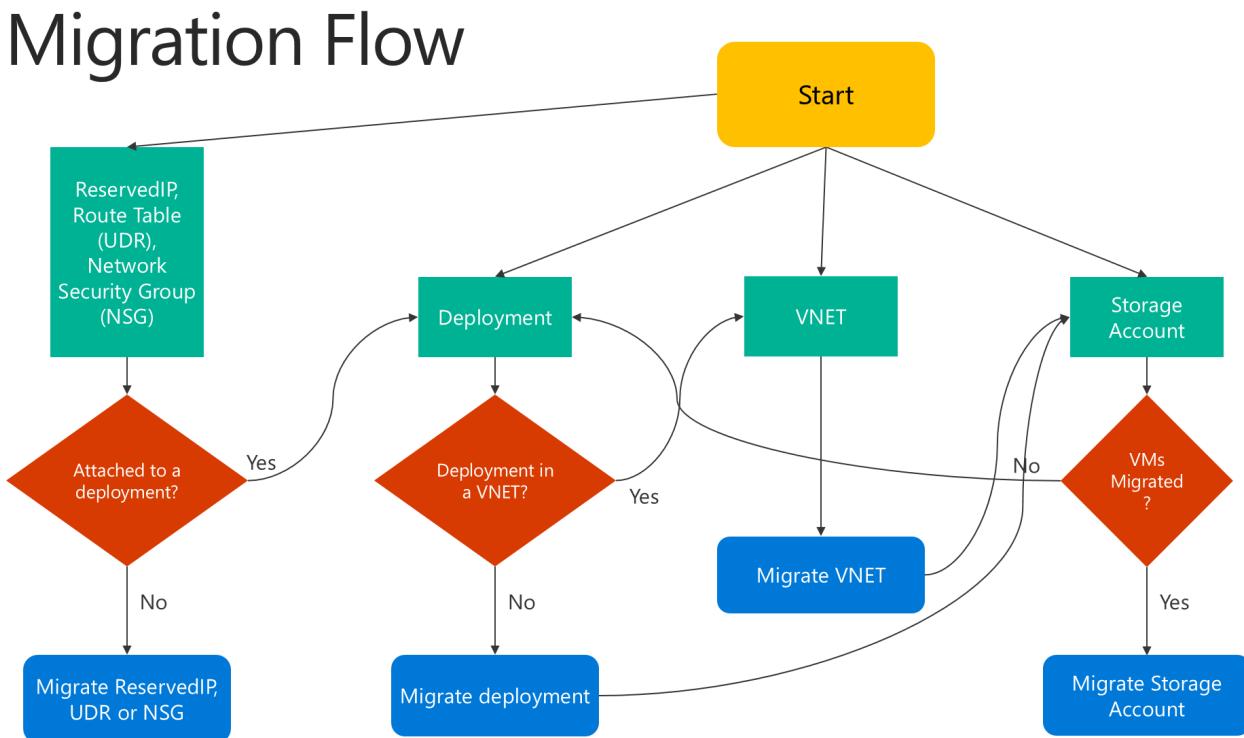
4/9/2018 • 6 min to read • [Edit Online](#)

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 CLI 1.0](#). Since Azure CLI 2.0 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 min 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.

List of errors

ERROR STRING	MITIGATION
Internal server error	<p>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.</p> <p>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.</p>
Migration is not supported for Deployment {deployment-name} in HostedService {hosted-service-name} because it is a PaaS deployment (Web/Worker).	<p>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.</p>
Template {template-name} deployment failed. CorrelationId={guid}	<p>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.</p> <p>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.</p>
The virtual network {virtual-network-name} does not exist.	<p>This can happen if you created the Virtual Network in the new Azure portal. The actual Virtual Network name follows the pattern "Group * "</p>
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.	<p>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.</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>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/vmbackupmigration</p>

ERROR STRING	MITIGATION
<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>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>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>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-prem DNS servers and DNS connectivity is lost <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>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>
<p>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</p>	<p>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.</p>

ERROR STRING	MITIGATION
VM {vm-name} in HostedService {cloud-service-name} contains Disk with MediaLink {vhdc-uri} which has blob name {vhdc-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 http://aka.ms/regionalscope 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 [http://aka.ms/regionalscope] for more details (http://aka.ms/regionalscope). 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-AzureStorageblob -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-AzureRmVM -ResourceGroupName "MyRG" -Name "MyVM"
Remove-AzureRmVMSecret -VM $vm
Update-AzureRmVM -ResourceGroupName "MyRG" -VM $vm

```

Azure CLI 2.0

```
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 • 1 min 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 min 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 Backup vault. Can I migrate my VMs from classic mode to Resource Manager mode and protect them in a Recovery Services vault?

VM recovery points in a backup vault don't automatically migrate to a Recovery Services vault when you move the VM from classic to Resource Manager mode. Follow these steps to transfer your VM backups:

1. In the Backup vault, go to the **Protected Items** tab and select the VM. Click [Stop Protection](#). Leave *Delete associated backup data* option **unchecked**.
2. Delete the backup/snapshot extension from the VM.
3. 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.
4. Create a Recovery Services vault and configure backup on the migrated virtual machine using **Backup** action on top of vault dashboard. For detailed information on backing up a VM to a Recovery Services vault, see the article, [Protect Azure VMs with a Recovery Services vault](#).

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](#)

Troubleshoot SSH connections to an Azure Linux VM that fails, errors out, or is refused

5/10/2018 • 10 min to read • [Edit Online](#)

There are various reasons that you encounter Secure Shell (SSH) errors, SSH connection failures, or SSH is refused when you try to connect to a Linux virtual machine (VM). This article helps you find and correct the problems. You can use the Azure portal, Azure CLI, or VM Access Extension for Linux to troubleshoot and resolve connection problems.

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.

If you need more help at any point in this article, you can contact the Azure experts on [the MSDN Azure and Stack Overflow forums](#). Alternatively, you can file an Azure support incident. Go to the [Azure support site](#) and select **Get support**. For information about using Azure Support, read the [Microsoft Azure support FAQ](#).

Quick troubleshooting steps

After each troubleshooting step, try reconnecting to the VM.

1. Reset the SSH configuration.
2. Reset the credentials for the user.
3. Verify the [Network Security Group](#) rules permit SSH traffic.
 - Ensure that a Network Security Group rule exists to permit SSH traffic (by default, TCP port 22).
 - You cannot use port redirection / mapping without using an Azure load balancer.
4. Check the [VM resource health](#).
 - Ensure that the VM reports as being healthy.
 - If you have boot diagnostics enabled, verify the VM is not reporting boot errors in the logs.
5. Restart the VM.
6. Redeploy the VM.

Continue reading for more detailed troubleshooting steps and explanations.

Available methods to troubleshoot SSH connection issues

You can reset credentials or SSH configuration using one of the following methods:

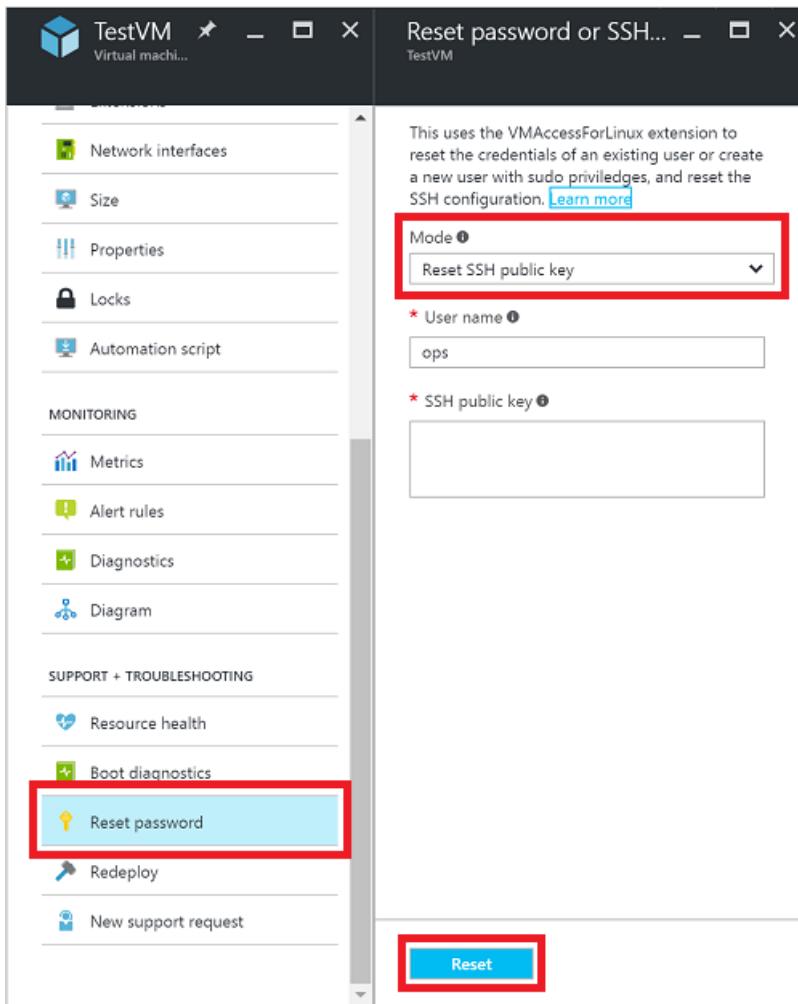
- [Azure portal](#) - great if you need to quickly reset the SSH configuration or SSH key and you don't have the Azure tools installed.
- [Azure CLI 2.0](#) - if you are already on the command line, quickly reset the SSH configuration or credentials. You can also use the [Azure CLI 1.0](#).
- [Azure VM Access For Linux extension](#) - create and reuse json definition files to reset the SSH configuration or user credentials.

After each troubleshooting step, try connecting to your VM again. If you still cannot connect, try the next step.

Use the Azure portal

The Azure portal provides a quick way to reset the SSH configuration or user credentials without installing any tools on your local computer.

Select your VM in the Azure portal. Scroll down to the **Support + Troubleshooting** section and select **Reset password** as in the following example:



Reset the SSH configuration

As a first step, select `Reset configuration only` from the **Mode** drop-down menu as in the preceding screenshot, then click the **Reset** button. Once this action has completed, try to access your VM again.

Reset SSH credentials for a user

To reset the credentials of an existing user, select either `Reset SSH public key` or `Reset password` from the **Mode** drop-down menu as in the preceding screenshot. Specify the username and an SSH key or new password, then click the **Reset** button.

You can also create a user with sudo privileges on the VM from this menu. Enter a new username and associated password or SSH key, and then click the **Reset** button.

Check security rules

Use [IP flow verify](#) to confirm if a rule in a network security group is blocking traffic to or from a virtual machine. You can also review effective security group rules to ensure inbound "Allow" NSG rule exists and is prioritized for SSH port (default 22). For more information, see [Using effective security rules to troubleshoot VM traffic flow](#).

Check routing

Use Network Watcher's [Next hop](#) capability to confirm that a route isn't preventing traffic from being routed to or from a virtual machine. You can also review effective routes to see all effective routes for a network interface. For

more information, see [Using effective routes to troubleshoot VM traffic flow](#).

Use the Azure CLI 2.0

If you haven't already, install the latest [Azure CLI 2.0](#) and log in to an Azure account using `az login`.

If you created and uploaded a custom Linux disk image, make sure the [Microsoft Azure Linux Agent](#) version 2.0.5 or later is installed. For VMs created using Gallery images, this access extension is already installed and configured for you.

Reset SSH configuration

You can initially try resetting the SSH configuration to default values and rebooting the SSH server on the VM. Note that this does not change the user account name, password, or SSH keys. The following example uses `az vm user reset-ssh` to reset the SSH configuration on the VM named `myVM` in `myResourceGroup`. Use your own values as follows:

```
az vm user reset-ssh --resource-group myResourceGroup --name myVM
```

Reset SSH credentials for a user

The following example uses `az vm user update` to reset the credentials for `myUsername` to the value specified in `myPassword`, on the VM named `myVM` in `myResourceGroup`. Use your own values as follows:

```
az vm user update --resource-group myResourceGroup --name myVM \
--username myUsername --password myPassword
```

If using SSH key authentication, you can reset the SSH key for a given user. The following example uses `az vm access set-linux-user` to update the SSH key stored in `~/.ssh/id_rsa.pub` for the user named `myUsername`, on the VM named `myVM` in `myResourceGroup`. Use your own values as follows:

```
az vm user update --resource-group myResourceGroup --name myVM \
--username myUsername --ssh-key-value ~/.ssh/id_rsa.pub
```

Use the VMAccess extension

The VM Access Extension for Linux reads in a json file that defines actions to carry out. These actions include resetting SSHD, resetting an SSH key, or adding a user. You still use the Azure CLI to call the VMAccess extension, but you can reuse the json files across multiple VMs if desired. This approach allows you to create a repository of json files that can then be called for given scenarios.

Reset SSHD

Create a file named `settings.json` with the following content:

```
{
  "reset_ssh": "True"
}
```

Using the Azure CLI, you then call the `VMAccessForLinux` extension to reset your SSHD connection by specifying your json file. The following example uses `az vm extension set` to reset SSHD on the VM named `myVM` in `myResourceGroup`. Use your own values as follows:

```
az vm extension set --resource-group philmea --vm-name Ubuntu \
--name VMAccessForLinux --publisher Microsoft.OSTCExtensions --version 1.2 --settings settings.json
```

Reset SSH credentials for a user

If SSHD appears to function correctly, you can reset the credentials for a given user. To reset the password for a user, create a file named `settings.json`. The following example resets the credentials for `myUsername` to the value specified in `myPassword`. Enter the following lines into your `settings.json` file, using your own values:

```
{
    "username": "myUsername", "password": "myPassword"
}
```

Or to reset the SSH key for a user, first create a file named `settings.json`. The following example resets the credentials for `myUsername` to the value specified in `myPassword`, on the VM named `myVM` in `myResourceGroup`. Enter the following lines into your `settings.json` file, using your own values:

```
{
    "username": "myUsername", "ssh_key": "mySSHKey"
}
```

After creating your json file, use the Azure CLI to call the `VMAccessForLinux` extension to reset your SSH user credentials by specifying your json file. The following example resets credentials on the VM named `myVM` in `myResourceGroup`. Use your own values as follows:

```
az vm extension set --resource-group philmea --vm-name Ubuntu \
--name VMAccessForLinux --publisher Microsoft.OSTCExtensions --version 1.2 --settings settings.json
```

Use the Azure CLI 1.0

If you haven't already, [install the Azure CLI 1.0 and connect to your Azure subscription](#). Make sure that you are using Resource Manager mode as follows:

```
azure config mode arm
```

If you created and uploaded a custom Linux disk image, make sure the [Microsoft Azure Linux Agent](#) version 2.0.5 or later is installed. For VMs created using Gallery images, this access extension is already installed and configured for you.

Reset SSH configuration

The SSHD configuration itself may be misconfigured or the service encountered an error. You can reset SSHD to make sure the SSH configuration itself is valid. Resetting SSHD should be the first troubleshooting step you take.

The following example resets SSHD on a VM named `myVM` in the resource group named `myResourceGroup`. Use your own VM and resource group names as follows:

```
azure vm reset-access --resource-group myResourceGroup --name myVM \
--reset-ssh
```

Reset SSH credentials for a user

If SSHD appears to function correctly, you can reset the password for a given user. The following example resets

the credentials for `myUsername` to the value specified in `myPassword`, on the VM named `myVM` in `myResourceGroup`. Use your own values as follows:

```
azure vm reset-access --resource-group myResourceGroup --name myVM \
--user-name myUsername --password myPassword
```

If using SSH key authentication, you can reset the SSH key for a given user. The following example updates the SSH key stored in `~/.ssh/id_rsa.pub` for the user named `myUsername`, on the VM named `myVM` in `myResourceGroup`. Use your own values as follows:

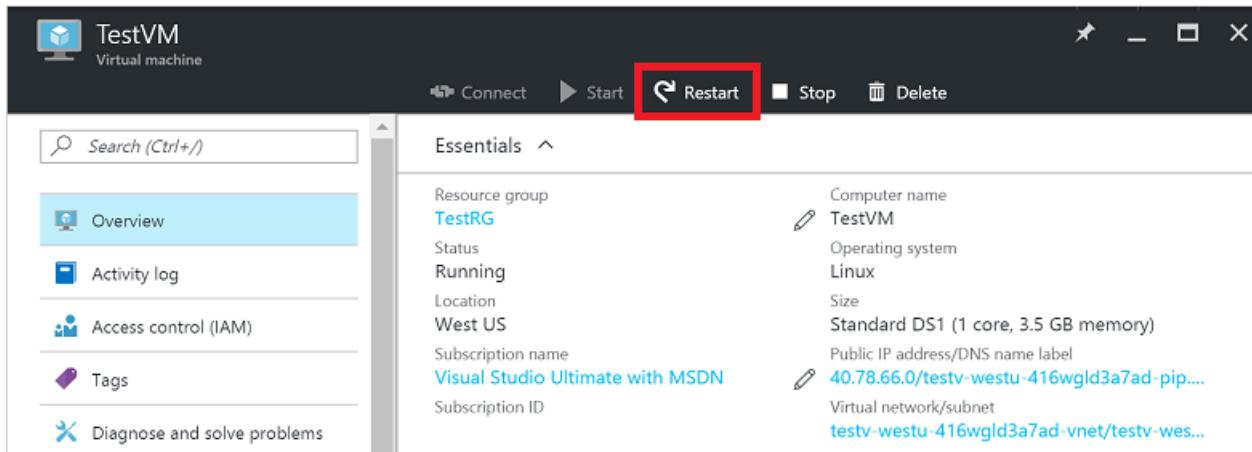
```
azure vm reset-access --resource-group myResourceGroup --name myVM \
--user-name myUsername --ssh-key-file ~/.ssh/id_rsa.pub
```

Restart a VM

If you have reset the SSH configuration and user credentials, or encountered an error in doing so, you can try restarting the VM to address underlying compute issues.

Azure portal

To restart a VM using the Azure portal, select your VM and click the **Restart** button as in the following example:



Azure CLI 1.0

The following example restarts the VM named `myVM` in the resource group named `myResourceGroup`. Use your own values as follows:

```
azure vm restart --resource-group myResourceGroup --name myVM
```

Azure CLI 2.0

The following example uses `az vm restart` to restart the VM named `myVM` in the resource group named `myResourceGroup`. Use your own values as follows:

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

Redeploy a VM

You can redeploy a VM to another node within Azure, which may correct any underlying networking issues. For information about redeploying a VM, see [Redeploy virtual machine to new Azure node](#).

NOTE

After this operation finishes, ephemeral disk data will be lost and dynamic IP addresses that are associated with the virtual machine will be updated.

Azure portal

To redeploy a VM using the Azure portal, select your VM and scroll down to the **Support + Troubleshooting** section. Click the **Redeploy** button as in the following example:

The screenshot shows the Azure portal interface for a virtual machine named "TestVM". The top navigation bar includes "Connect", "Start", "Restart", "Stop", and "Delete" buttons. The left sidebar has sections for "Network interfaces", "Size", "Properties", "Locks", "Automation script", "MONITORING" (Metrics, Alert rules, Diagnostics, Diagram), and "SUPPORT + TROUBLESHOOTING" (Resource health, Boot diagnostics, Reset password, **Redeploy**, New support request). The main content area is divided into "Essentials" and "Monitoring". The "Essentials" section displays resource group (TestRG), status (Running), location (West US), subscription name (Visual Studio Ultimate with MSDN), and subscription ID. The "Monitoring" section shows a CPU percentage chart from 1 PM to 1:45 PM, with the Y-axis ranging from 0% to 100% and the X-axis showing 1 PM, 1:15 PM, 1:30 PM, and 1:45 PM. The chart shows a single data series starting at 0% and remaining flat until 1:15 PM, where it begins to rise sharply towards 100% by 1:45 PM.

Azure CLI 1.0

The following example redeploys the VM named `myVM` in the resource group named `myResourceGroup`. Use your own values as follows:

```
azure vm redeploy --resource-group myResourceGroup --name myVM
```

Azure CLI 2.0

The following example use `az vm redeploy` to redeploy the VM named `myVM` in the resource group named `myResourceGroup`. Use your own values as follows:

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

VMs created by using the Classic deployment model

Try these steps to resolve the most common SSH connection failures for VMs that were created by using the

classic deployment model. After each step, try reconnecting to the VM.

- Reset remote access from the [Azure portal](#). On the Azure portal, select your VM and click the **Reset Remote...** button.
- Restart the VM. On the [Azure portal](#), select your VM and click the **Restart** button.
- Redeploy the VM to a new Azure node. For information about how to redeploy a VM, see [Redeploy virtual machine to new Azure node](#).

After this operation finishes, ephemeral disk data will be lost and dynamic IP addresses that are associated with the virtual machine will be updated.

- Follow the instructions in [How to reset a password or SSH for Linux-based virtual machines](#) to:
 - Reset the password or SSH key.
 - Create a *sudo* user account.
 - Reset the SSH configuration.
- Check the VM's resource health for any platform issues.
Select your VM and scroll down **Settings** > **Check Health**.

Additional resources

- If you are still unable to SSH to your VM after following the after steps, see [more detailed troubleshooting steps](#) to review additional steps to resolve your issue.
- For more information about troubleshooting application access, see [Troubleshoot access to an application running on an Azure virtual machine](#)
- For more information about troubleshooting virtual machines that were created by using the classic deployment model, see [How to reset a password or SSH for Linux-based virtual machines](#).

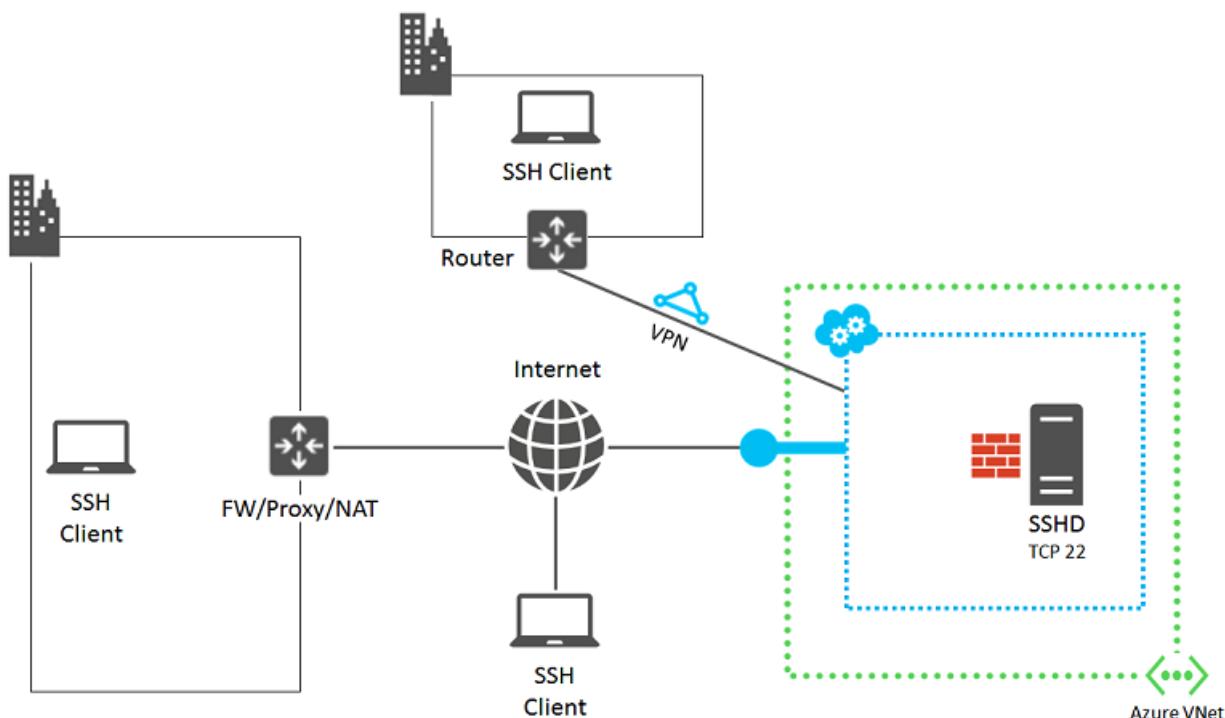
Detailed SSH troubleshooting steps for issues connecting to a Linux VM in Azure

1/22/2018 • 6 min to read • [Edit Online](#)

There are many possible reasons that the SSH client might not be able to reach the SSH service on the VM. If you have followed through the more [general SSH troubleshooting steps](#), you need to further troubleshoot the connection issue. This article guides you through detailed troubleshooting steps to determine where the SSH connection is failing and how to resolve it.

Take preliminary steps

The following diagram shows the components that are involved.



The following steps help you isolate the source of the failure and figure out solutions or workarounds.

1. Check the status of the VM in the portal. In the [Azure portal](#), select **Virtual machines** > *VM name*.

The status pane for the VM should show **Running**. Scroll down to show recent activity for compute, storage, and network resources.

2. Select **Settings** to examine endpoints, IP addresses, network security groups, and other settings.

The VM should have an endpoint defined for SSH traffic that you can view in **Endpoints** or **Network security group**. Endpoints in VMs that were created by using Resource Manager are stored in a network security group. Verify that the rules have been applied to the network security group and are referenced in the subnet.

To verify network connectivity, check the configured endpoints and see if you can connect to the VM through another protocol, such as HTTP or another service.

After these steps, try the SSH connection again.

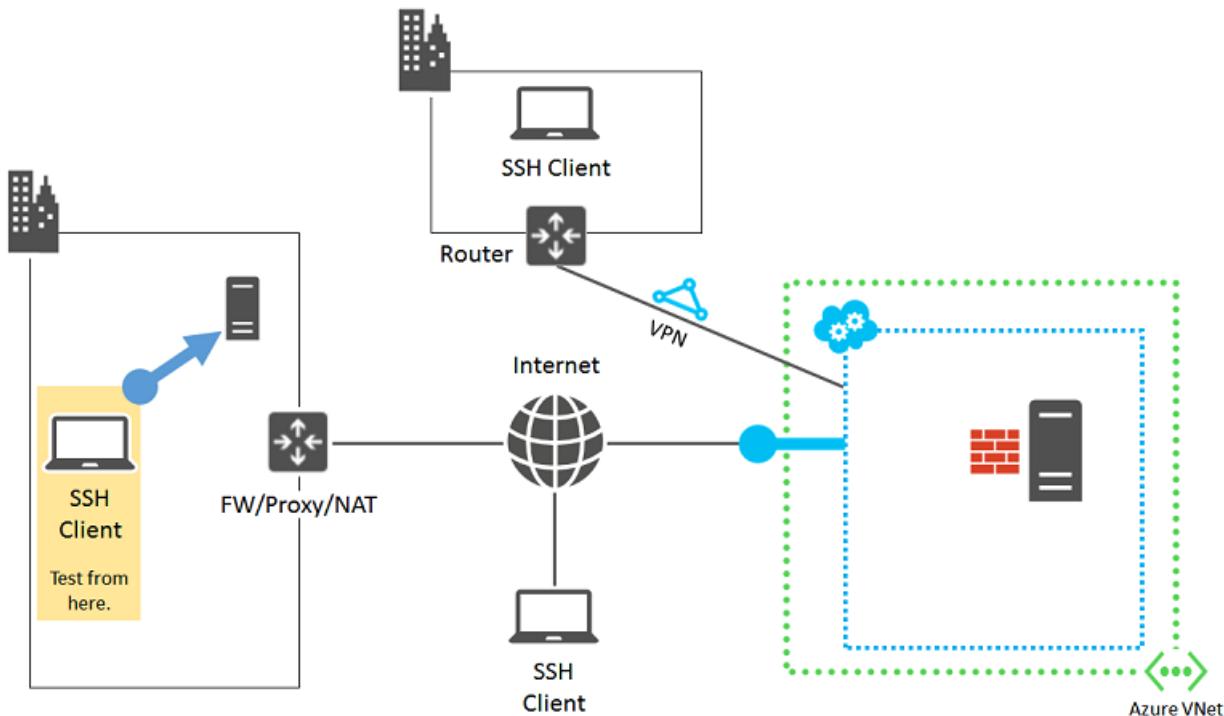
Find the source of the issue

The SSH client on your computer might fail to connect to the SSH service on the Azure VM due to issues or misconfigurations in the following areas:

- [SSH client computer](#)
- [Organization edge device](#)
- [Cloud service endpoint and access control list \(ACL\)](#)
- [Network security groups](#)
- [Linux-based Azure VM](#)

Source 1: SSH client computer

To eliminate your computer as the source of the failure, verify that it can make SSH connections to another on-premises, Linux-based computer.



If the connection fails, check for the following issues on your computer:

- A local firewall setting that is blocking inbound or outbound SSH traffic (TCP 22)
- Locally installed client proxy software that is preventing SSH connections
- Locally installed network monitoring software that is preventing SSH connections
- Other types of security software that either monitor traffic or allow/disallow specific types of traffic

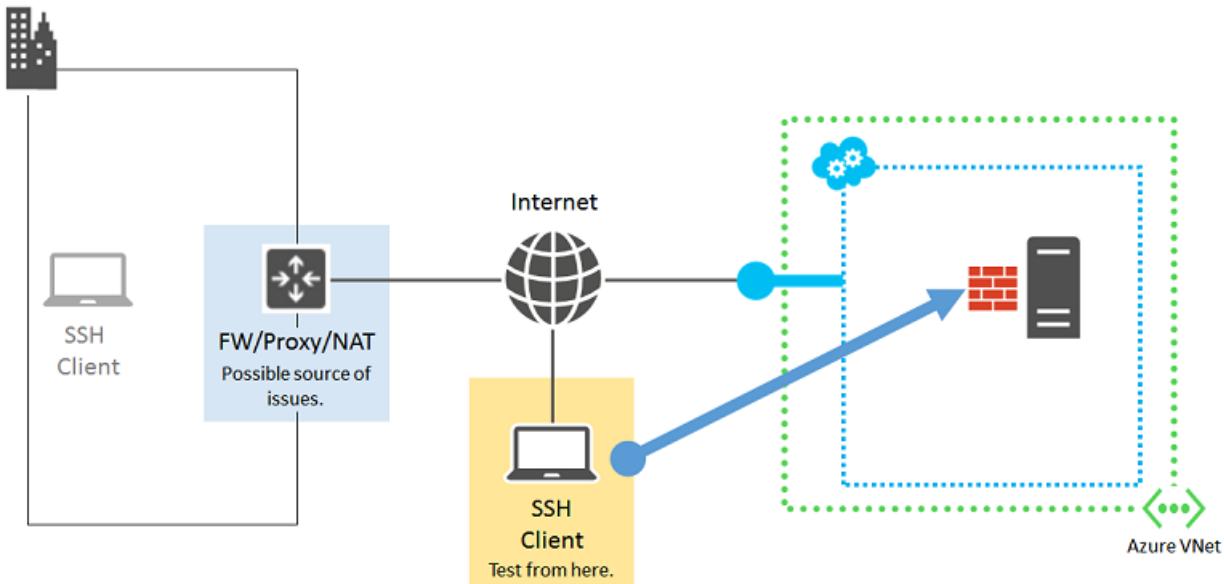
If one of these conditions apply, temporarily disable the software and try an SSH connection to an on-premises computer to find out the reason the connection is being blocked on your computer. Then work with your network administrator to correct the software settings to allow SSH connections.

If you are using certificate authentication, verify that you have these permissions to the .ssh folder in your home directory:

- Chmod 700 ~/.ssh
- Chmod 644 ~/.ssh/*.pub
- Chmod 600 ~/.ssh/id_rsa (or any other files that have your private keys stored in them)
- Chmod 644 ~/.ssh/known_hosts (contains hosts that you've connected to via SSH)

Source 2: Organization edge device

To eliminate your organization edge device as the source of the failure, verify that a computer directly connected to the Internet can make SSH connections to your Azure VM. If you are accessing the VM over a site-to-site VPN or an Azure ExpressRoute connection, skip to [Source 4: Network security groups](#).



If you don't have a computer that is directly connected to the Internet, create a new Azure VM in its own resource group or cloud service and use that new VM. For more information, see [Create a virtual machine running Linux in Azure](#). Delete the resource group or VM and cloud service when you're done with your testing.

If you can create an SSH connection with a computer that's directly connected to the Internet, check your organization edge device for:

- An internal firewall that's blocking SSH traffic with the Internet
- A proxy server that's preventing SSH connections
- Intrusion detection or network monitoring software running on devices in your edge network that's preventing SSH connections

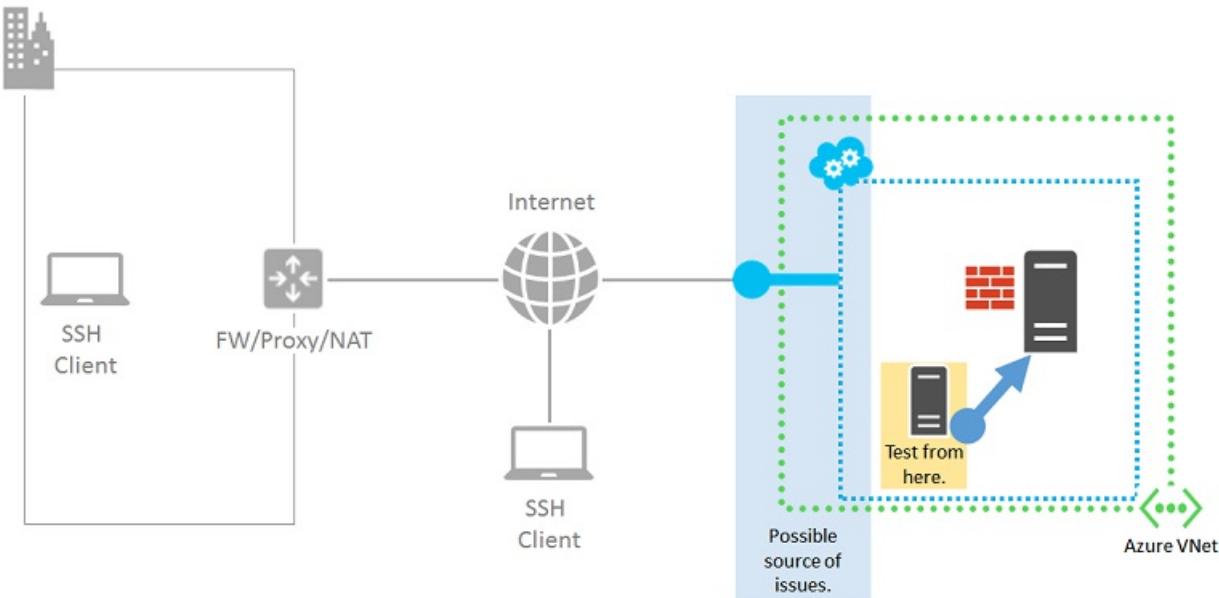
Work with your network administrator to correct the settings of your organization edge devices to allow SSH traffic with the Internet.

Source 3: Cloud service endpoint and ACL

NOTE

This source applies only to VMs that were created by using the classic deployment model. For VMs that were created by using Resource Manager, skip to [source 4: Network security groups](#).

To eliminate the cloud service endpoint and ACL as the source of the failure, verify that another Azure VM in the same virtual network can connect using SSH.



If you don't have another VM in the same virtual network, you can easily create one. For more information, see [Create a Linux VM on Azure using the CLI](#). Delete the extra VM when you are done with your testing.

If you can create an SSH connection with a VM in the same virtual network, check the following areas:

- **The endpoint configuration for SSH traffic on the target VM.** The private TCP port of the endpoint should match the TCP port on which the SSH service on the VM is listening. (The default port is 22). Verify the SSH TCP port number in the Azure portal by selecting **Virtual machines** > **VM name** > **Settings** > **Endpoints**.
- **The ACL for the SSH traffic endpoint on the target virtual machine.** An ACL enables you to specify allowed or denied incoming traffic from the Internet, based on its source IP address. Misconfigured ACLs can prevent incoming SSH traffic to the endpoint. Check your ACLs to ensure that incoming traffic from the public IP addresses of your proxy or other edge server is allowed. For more information, see [About network access control lists \(ACLs\)](#).

To eliminate the endpoint as a source of the problem, remove the current endpoint, create another endpoint, and specify the SSH name (TCP port 22 for the public and private port number). For more information, see [Set up endpoints on a virtual machine in Azure](#).

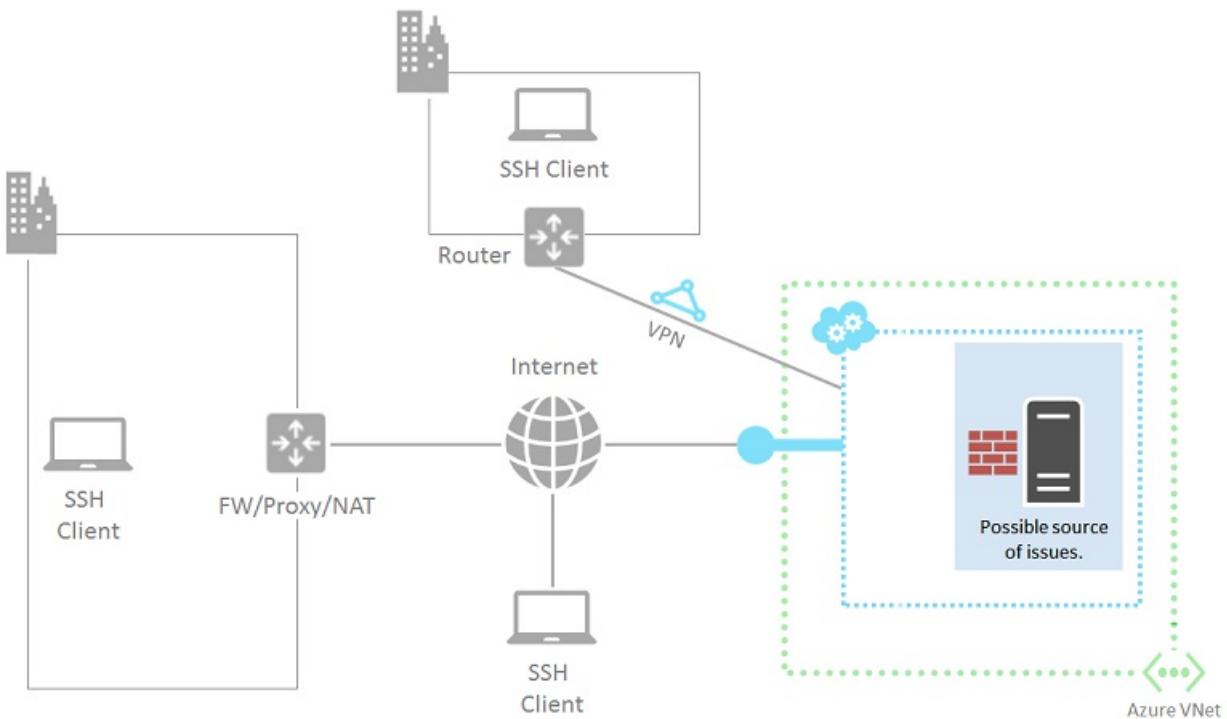
Source 4: Network security groups

Network security groups enable you to have more granular control of allowed inbound and outbound traffic. You can create rules that span subnets and cloud services in an Azure virtual network. Check your network security group rules to ensure that SSH traffic to and from the Internet is allowed. For more information, see [About network security groups](#).

You can also use IP Verify to validate the NSG configuration. For more information, see [Azure network monitoring overview](#).

Source 5: Linux-based Azure virtual machine

The last source of possible problems is the Azure virtual machine itself.



If you haven't done so already, follow the instructions [to reset a password Linux-based virtual machines](#).

Try connecting from your computer again. If it still fails, the following are some of the possible issues:

- The SSH service is not running on the target virtual machine.
- The SSH service is not listening on TCP port 22. To test, install a telnet client on your local computer and run "telnet *cloudServiceName.cloudapp.net* 22". This step determines if the virtual machine allows inbound and outbound communication to the SSH endpoint.
- The local firewall on the target virtual machine has rules that are preventing inbound or outbound SSH traffic.
- Intrusion detection or network monitoring software that's running on the Azure virtual machine is preventing SSH connections.

Additional resources

For more information about troubleshooting application access, see [Troubleshoot access to an application running on an Azure virtual machine](#)

How to reset local Linux password on Azure VMs

1/3/2018 • 1 min to read • [Edit Online](#)

This article introduces several methods to reset local Linux Virtual Machine (VM) passwords. If the user account is expired or you just want to create a new account, you can use the following methods to create a new local admin account and re-gain access to the VM.

Symptoms

You can't log in to the VM, and you receive a message that indicates that the password that you used is incorrect. Additionally, you can't use VMAgent to reset your password on the Azure Portal.

Manual Password Reset procedure

1. Delete the VM and keep the attached disks.
2. Attach the OS Drive as a data disk to another temporal VM in the same location.
3. Run the following SSH command on the temporal VM to become a super-user.

```
~~~~~  
sudo su  
~~~~~
```

1. Run **fdisk -l** or look at system logs to find the newly attached disk. Locate the drive name to mount. Then on the temporal VM, look in the relevant log file.

```
grep SCSI /var/log/kern.log (ubuntu)  
grep SCSI /var/log/messages (centos, suse, oracle)
```

The following is example output of the grep command:

```
kernel: [ 9707.100572] sd 3:0:0:0: [sdc] Attached SCSI disk
```

2. Create a mount point called **tempmount**.

```
mkdir /tempmount
```

3. Mount the OS disk on the mount point. You usually need to mount sdc1 or sdc2. This will depend on the hosting partition in /etc directory from the broken machine disk.

```
mount /dev/sdc1 /tempmount
```

4. Perform a backup before making any changes:

```
cp /etc/passwd /etc/passwd_orig  
cp /etc/shadow /etc/shadow_orig  
cp /tmpmount/etc/passwd /etc/passwd  
cp /tmpmount/etc/shadow /etc/shadow  
cp /tmpmount/etc/passwd /tmpmount/etc/passwd_orig  
cp /tmpmount/etc/shadow /tmpmount/etc/shadow_orig
```

5. Reset the user's password that you need:

```
passwd <<USER>>
```

6. Move the modified files to the correct location on the broken machine's disk.

```
cp /etc/passwd /tmpmount/etc/passwd  
cp /etc/shadow /tmpmount/etc/shadow  
cp /etc/passwd_orig /etc/passwd  
cp /etc/shadow_orig /etc/shadow
```

7. Go back to the root and unmount the disk.

```
cd /  
umount /tmpmount
```

8. Detach the disk from the management portal.

9. Recreate the VM.

Next steps

- [Troubleshoot Azure VM by attaching OS disk to another Azure VM](#)
- [Azure CLI: How to delete and re-deploy a VM from VHD](#)

Understand a system reboot for Azure VM

5/11/2018 • 7 min to read • [Edit Online](#)

Azure virtual machines (VMs) might sometimes reboot for no apparent reason, without evidence of your having initiated the reboot operation. This article lists the actions and events that can cause VMs to reboot and provides insight into how to avoid unexpected reboot issues or reduce the impact of such issues.

Configure the VMs for high availability

The best way to protect an application that's running on Azure against VM reboots and downtime is to configure the VMs for high availability.

To provide this level of redundancy to your application, we recommend that you group two or more VMs in an availability set. This configuration ensures that during either a planned or unplanned maintenance event, at least one VM is available and meets the 99.95 percent [Azure SLA](#).

For more information about availability sets, see the following articles:

- [Manage the availability of VMs](#)
- [Configure availability of VMs](#)

Resource Health information

Azure Resource Health is a service that exposes the health of individual Azure resources and provides actionable guidance for troubleshooting problems. In a cloud environment where it isn't possible to directly access servers or infrastructure elements, the goal of Resource Health is to reduce the time that you spend on troubleshooting. In particular, the aim is to reduce the time that you spend determining whether the root of the problem lies in the application or in an event inside the Azure platform. For more information, see [Understand and use Resource Health](#).

Actions and events that can cause the VM to reboot

Planned maintenance

Microsoft Azure periodically performs updates across the globe to improve the reliability, performance, and security of the host infrastructure that underlies VMs. Many of these updates, including memory-preserving updates, are performed without any impact on your VMs or cloud services.

However, some updates do require a reboot. In such cases, the VMs are shut down while we patch the infrastructure, and then the VMs are restarted.

To understand what Azure planned maintenance is and how it can affect the availability of your Linux VMs, see the articles listed here. The articles provide background about the Azure planned maintenance process and how to schedule planned maintenance to further reduce the impact.

- [Planned maintenance for VMs in Azure](#)
- [How to schedule planned maintenance on Azure VMs](#)

Memory-preserving updates

For this class of updates in Microsoft Azure, users experience no impact on their running VMs. Many of these updates are to components or services that can be updated without interfering with the running instance. Some are platform infrastructure updates on the host operating system that can be applied without a reboot of the VMs.

These memory-preserving updates are accomplished with technology that enables in-place live migration. When it is being updated, the VM is placed in a *paused* state. This state preserves the memory in RAM while the underlying host operating system receives the necessary updates and patches. The VM is resumed within 30 seconds of being paused. After the VM is resumed, its clock is automatically synchronized.

Because of the short pause period, deploying updates through this mechanism greatly reduces the impact on the VMs. However, not all updates can be deployed in this way.

Multi-instance updates (for VMs in an availability set) are applied one update domain at a time.

NOTE

Linux machines that have old kernel versions are affected by a kernel panic during this update method. To avoid this issue, update to kernel version 3.10.0-327.10.1 or later. For more information, see [An Azure Linux VM on a 3.10-based kernel panics after a host node upgrade](#).

User-initiated reboot or shutdown actions

If you perform a reboot from the Azure portal, Azure PowerShell, command-line interface, or Reset API, you can find the event in the [Azure Activity Log](#).

If you perform the action from the VM's operating system, you can find the event in the system logs.

Other scenarios that usually cause the VM to reboot include multiple configuration-change actions. You'll ordinarily see a warning message indicating that executing a particular action will result in a reboot of the VM. Examples include any VM resize operations, changing the password of the administrative account, and setting a static IP address.

Azure Security Center and Windows Update

Azure Security Center monitors daily Windows and Linux VMs for missing operating-system updates. Security Center retrieves a list of available security and critical updates from Windows Update or Windows Server Update Services (WSUS), depending on which service is configured on a Windows VM. Security Center also checks for the latest updates for Linux systems. If your VM is missing a system update, Security Center recommends that you apply system updates. The application of these system updates is controlled through the Security Center in the Azure portal. After you apply some updates, VM reboots might be required. For more information, see [Apply system updates in Azure Security Center](#).

Like on-premises servers, Azure does not push updates from Windows Update to Windows Azure VMs, because these machines are intended to be managed by their users. You are, however, encouraged to leave the automatic Windows Update setting enabled. Automatic installation of updates from Windows Update can also cause reboots to occur after the updates are applied. For more information, see [Windows Update FAQ](#).

Other situations affecting the availability of your VM

There are other cases in which Azure might actively suspend the use of a VM. You'll receive email notifications before this action is taken, so you'll have a chance to resolve the underlying issues. Examples of issues that affect VM availability include security violations and the expiration of payment methods.

Host server faults

The VM is hosted on a physical server that is running inside an Azure datacenter. The physical server runs an agent called the Host Agent in addition to a few other Azure components. When these Azure software components on the physical server become unresponsive, the monitoring system triggers a reboot of the host server to attempt recovery. The VM is usually available again within five minutes and continues to live on the same host as previously.

Server faults are usually caused by hardware failure, such as the failure of a hard disk or solid-state drive. Azure continuously monitors these occurrences, identifies the underlying bugs, and rolls out updates after the mitigation

has been implemented and tested.

Because some host server faults can be specific to that server, a repeated VM reboot situation might be improved by manually redeploying the VM to another host server. This operation can be triggered by using the **redeploy** option on the details page of the VM, or by stopping and restarting the VM in the Azure portal.

Auto-recovery

If the host server cannot reboot for any reason, the Azure platform initiates an auto-recovery action to take the faulty host server out of rotation for further investigation.

All VMs on that host are automatically relocated to a different, healthy host server. This process is usually complete within 15 minutes. To learn more about the auto-recovery process, see [Auto-recovery of VMs](#).

Unplanned maintenance

On rare occasions, the Azure operations team might need to perform maintenance activities to ensure the overall health of the Azure platform. This behavior might affect VM availability, and it usually results in the same auto-recovery action as described earlier.

Unplanned maintenances include the following:

- Urgent node defragmentation
- Urgent network switch updates

VM crashes

VMs might restart because of issues within the VM itself. The workload or role that's running on the VM might trigger a bug check within the guest operating system. For help determining the reason for the crash, view the system and application logs for Windows VMs, and the serial logs for Linux VMs.

Storage-related forced shutdowns

VMs in Azure rely on virtual disks for operating system and data storage that is hosted on the Azure Storage infrastructure. Whenever the availability or connectivity between the VM and the associated virtual disks is affected for more than 120 seconds, the Azure platform performs a forced shutdown of the VMs to avoid data corruption. The VMs are automatically powered back on after storage connectivity has been restored.

The duration of the shutdown can be as short as five minutes but can be significantly longer. The following is one of the specific cases that is associated with storage-related forced shutdowns:

Exceeding IO limits

VMs might be temporarily shut down when I/O requests are consistently throttled because the volume of I/O operations per second (IOPS) exceeds the I/O limits for the disk. (Standard disk storage is limited to 500 IOPS.) To mitigate this issue, use disk striping or configure the storage space inside the guest VM, depending on the workload. For details, see [Configuring Azure VMs for Optimal Storage Performance](#).

Higher IOPS limits are available via Azure Premium Storage with up to 80,000 IOPS. For more information, see [High-Performance Premium Storage](#).

Other incidents

In rare circumstances, a widespread issue can affect multiple servers in an Azure datacenter. If this issue occurs, the Azure team sends email notifications to the affected subscriptions. You can check the [Azure Service Health dashboard](#) and the Azure portal for the status of ongoing outages and past incidents.

How to use boot diagnostics to troubleshoot Linux virtual machines in Azure

5/9/2018 • 2 min to read • [Edit Online](#)

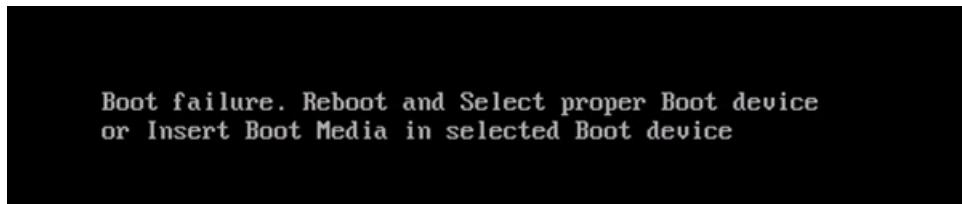
Support for two debugging features is now available in Azure: Console Output and Screenshot support for Azure Virtual Machines Resource Manager deployment model.

When bringing your own image to Azure or even booting one of the platform images, there can be many reasons why a Virtual Machine gets into a non-bootable state. These features enable you to easily diagnose and recover your Virtual Machines from boot failures.

For Linux Virtual Machines, you can easily view the output of your console log from the Portal:

The screenshot shows the Azure portal interface for a Linux VM named "console-linux". The left sidebar shows "Internal" selected. The main area has a header "Microsoft Azure > console-linux > Boot diagnostics". The "Essentials" section shows the VM's configuration: Resource group "deletedbyEndOfSeptember", Computer name "console-linux", Public IP address "168.63.236.47", Status "Running", Location "South East Asia", Subscription name "las5 Experiences dev", Operating system "Linux", and Size "Standard D1 (1 core, 3.5 GB memory)". Below this is an "Operations" section with a bar chart showing event counts for "CONSOLE-UNIX" (around 60 events) and an "Alert rules" section. On the right, a large window titled "Boot diagnostics" shows the kernel boot logs. The logs start with "[M][1;1M 0.000000] Initializing cgroup subsys cpuset" and continue through the boot process, ending with "[0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000ffff] usable".

However, for both Windows and Linux Virtual Machines, Azure also enables you to see a screenshot of the VM from the hypervisor:



Both of these features are supported for Azure Virtual Machines in all regions. Note, screenshots, and output can take up to 10 minutes to appear in your storage account.

Common boot errors

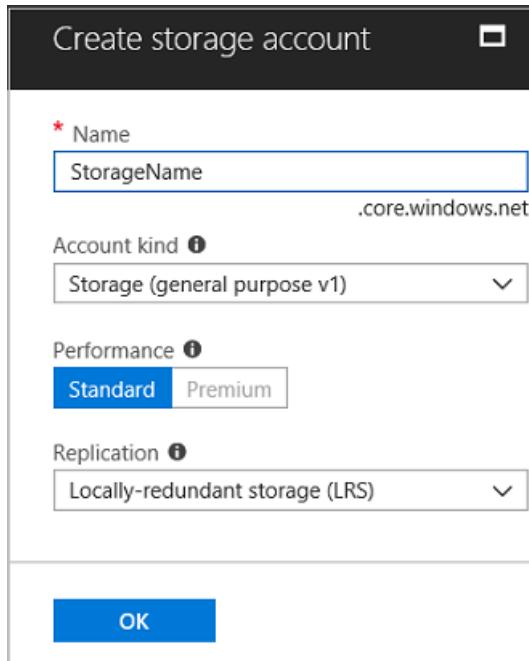
- [File system issues](#)
- [Kernel Issues](#)
- [FSTAB errors](#)

Enable diagnostics on a new virtual machine

1. When creating a new virtual machine from the Azure portal, select the **Azure Resource Manager** from the deployment model dropdown:



2. In **Settings**, enable the **Boot diagnostics**, and then select a storage account that you would like to place these diagnostic files.



NOTE

The Boot diagnostics feature does not support premium storage account. If you use the premium storage account for Boot diagnostics, you might receive the StorageAccountTypeNotSupportedException when you start the VM.

3. If you are deploying from an Azure Resource Manager template, navigate to your virtual machine resource and append the diagnostics profile section. Remember to use the "2015-06-15" API version header.

```
{  
    "apiVersion": "2015-06-15",  
    "type": "Microsoft.Compute/virtualMachines",  
    ...  
}
```

4. The diagnostics profile enables you to select the storage account where you want to put these logs.

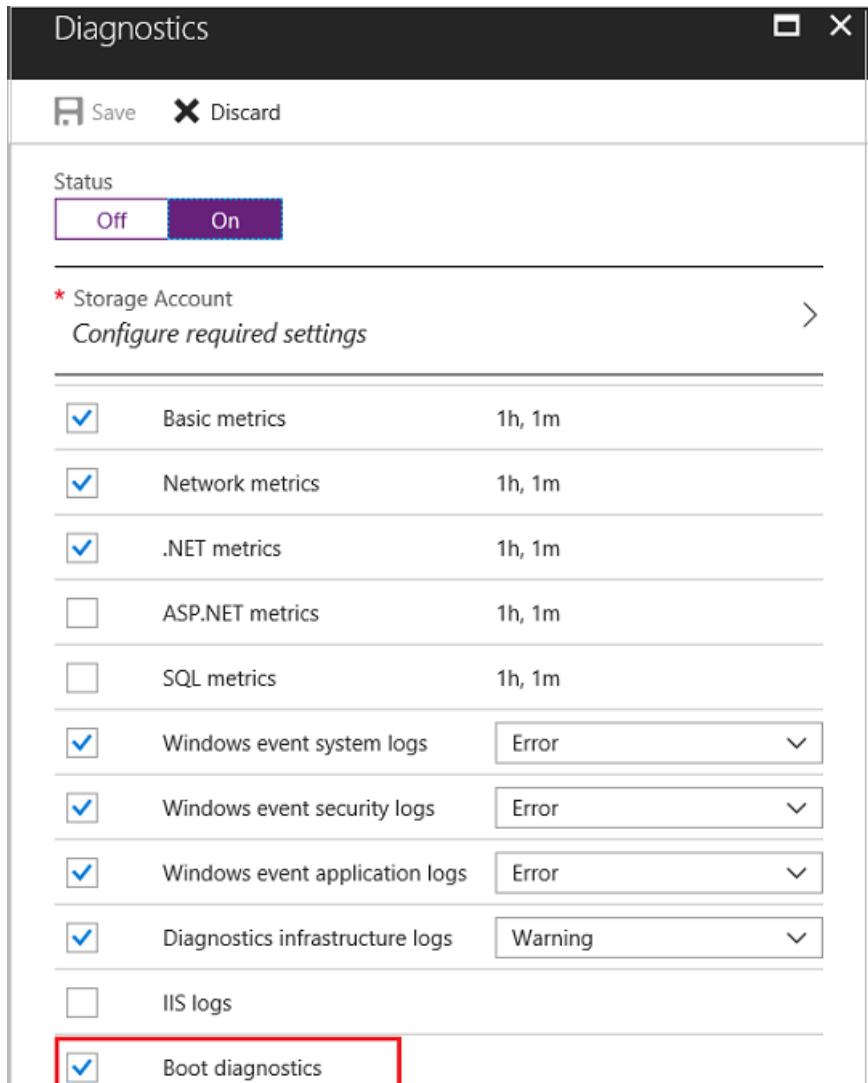
```
        "diagnosticsProfile": {  
            "bootDiagnostics": {  
                "enabled": true,  
                "storageUri": "[concat('http://', parameters('newStorageAccountName'),  
'.blob.core.windows.net')]"  
            }  
        }  
    }  
}
```

To deploy a sample virtual machine with boot diagnostics enabled, check out our repo [here](#).

Enable Boot diagnostics on existing virtual machine

To enable Boot diagnostics on an existing virtual machine, follow these steps:

1. Log in to the [Azure portal](#), and then select the virtual machine.
2. In **Support + troubleshooting**, select **Boot diagnostics > Settings**, change the status to **On**, and then select a storage account.
3. Make sure that the Boot diagnostics option is selected and then save the change.



4. Restart the VM to take effect.

Next steps

If you see a "Failed to get contents of the log" error when you use VM Boot Diagnostics, see [Failed to get contents of the log error in VM Boot Diagnostics](#).

Virtual machine serial console (preview)

4/11/2018 • 7 min to read • [Edit Online](#)

The virtual machine serial console on Azure provides access to a text-based console for Linux and Windows virtual machines. This serial connection is to COM1 serial port of the virtual machine and provides access to the virtual machine and are not related to virtual machine's network / operating system state. Access to the serial console for a virtual machine can be done only via Azure portal currently and allowed only for those users who have VM Contributor or above access to the virtual machine.

NOTE

Previews are made available to you on the condition that you agree to the terms of use. For more information, see [Microsoft Azure Supplemental Terms of Use for Microsoft Azure Previews](#). Currently this service is in **public preview** and access to the serial console for virtual machines is available to global Azure regions. At this point serial console is not available Azure Government, Azure Germany, and Azure China cloud.

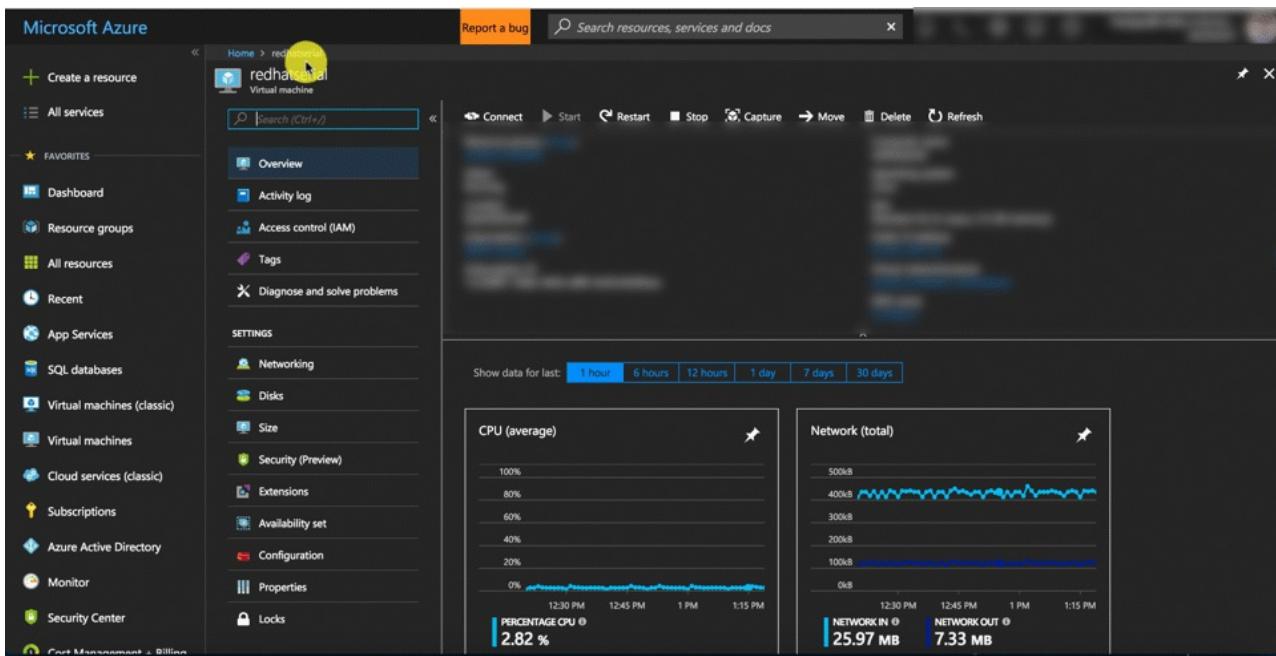
Prerequisites

- Virtual machine MUST have [boot diagnostics](#) enabled
- The account using the serial console must have [Contributor role](#) for VM and the [boot diagnostics](#) storage account.
- For settings specific to Linux distro, see [Accessing the serial console for Linux](#)

Open the serial console

serial console for virtual machines is only accessible via [Azure portal](#). Below are the steps to access serial console for virtual machines via portal

1. Open the Azure portal
2. In the left menu, select virtual machines.
3. Click on the VM in the list. The overview page for the VM will open.
4. Scroll down to the Support + Troubleshooting section and click on serial console (Preview) option. A new pane with the serial console will open and start the connection.



NOTE

Serial console requires a local user with a password configured. At this time, VMs only configured with SSH public key will not have access to the serial console. To create a local user with password, follow [VM Access Extension](#) and create local user with password.

Disable feature

The serial console functionality can be deactivated for specific VMs by disabling that VM's boot diagnostics setting.

Serial console security

Access security

Access to Serial console is limited to users who have [VM Contributors](#) or above access to the virtual machine. If your AAD tenant requires Multi-Factor Authentication then access to the serial console will also need MFA as its access is via [Azure portal](#).

Channel security

All data is sent back and forth is encrypted on the wire.

Audit logs

All access to the serial console is currently logged in the [boot diagnostics](#) logs of the virtual machine. Access to these logs are owned and controlled by the Azure virtual machine administrator.

Caution

While no access passwords for the console are logged, if commands run within the console contain or output passwords, secrets, user names or any other form of Personally Identifiable Information (PII), those will be written to the virtual machine boot diagnostics logs, along with all other visible text, as part of the implementation of the serial console's scrollback functionality. These logs are circular and only individuals with read permissions to the diagnostics storage account have access to them, however we recommend following the best practice of using the SSH console for anything that may involve secrets and/or PII.

Concurrent usage

If a user is connected to serial console and another user successfully requests access to that same virtual machine, the first user will be disconnected and the second user connected in a manner akin to the first user standing up and leaving the physical console and a new user sitting down.

Caution

This means that the user who gets disconnected will not be logged out! The ability to enforce a logout upon disconnect (via SIGHUP or similar mechanism) is still in the roadmap. For Windows there is an automatic timeout enabled in SAC, however for Linux you can configure terminal timeout setting. To do this simply add `export TMOUT=600` in your `.bash_profile` or `.profile` for the user you logon in the console with, to timeout the session after 10 minutes.

Disable feature

The serial console functionality can be deactivated for specific VMs by disabling that VM's boot diagnostics setting.

Common scenarios for accessing serial console

SCENARIO	ACTIONS IN SERIAL CONSOLE	OS APPLICABILITY
Broken FSTAB file	Enter key to continue and fix fstab file using a text editor. See how to fix fstab issues	Linux
Incorrect firewall rules	Access serial console and fix iptables or Windows firewall rules	Linux/Windows
Filesystem corruption/check	Access serial console and recover filesystem	Linux/Windows
SSH/RDP configuration issues	Access serial console and change settings	Linux/Windows
Network lock down system	Access serial console via portal to manage system	Linux/Windows
Interacting with bootloader	Access GRUB/BCD via the serial console	Linux/Windows

Accessing serial console for Linux

In order for serial console to function properly, the guest operating system must be configured to read and write console messages to the serial port. Most [Endorsed Azure Linux Distributions](#) have the serial console configured by default. Just by clicking in the portal on the Serial console section will provide access to the console.

Access for RedHat

RedHat Images available on Azure have console access enabled by default. Single user mode in Red Hat requires root user to be enabled, which is disabled by default. If you have a need to enable single user mode, use the following instructions:

1. Log in to the Red Hat system via SSH
2. Enable password for root user
 - `passwd root` (set a strong root password)
3. Ensure root user can only log in via ttyS0
 - `edit /etc/ssh/sshd_config` and ensure PermitRootLog in is set to no
 - `edit /etc/securetty file` to only allow log in via ttyS0

Now if the system boots into single user mode you can log in via root password.

Alternatively for RHEL 7.4+ or 6.9+ you can enable single user mode in the GRUB prompts, see instructions [here](#)

Access for Ubuntu

Ubuntu images available on Azure have console access enabled by default. If the system boots into Single User Mode you can access without additional credentials.

Access for CoreOS

CoreOS images available on Azure have console access enabled by default. If necessary system can be booted into Single User Mode via changing GRUB parameters and adding `coreos.autologin=ttyS0` would enable core user to log in and available in serial console.

Access for SUSE

SLES images available on Azure have console access enabled by default. If you are using older versions of SLES on Azure, follow the [KB article](#) to enable serial console. Newer Images of SLES 12 SP3+ also allows access via the serial console in case the system boots into emergency mode.

Access for CentOS

CentOS images available on Azure have console access enabled by default. For Single User Mode, follow instructions similar to Red Hat Images above.

Access for Oracle Linux

Oracle Linux images available on Azure have console access enabled by default. For Single User Mode, follow instructions similar to Red Hat Images above.

Access for custom Linux image

To enable serial console for your custom Linux VM image, enable console access in /etc/inittab to run a terminal on ttyS0. Below is an example to add this in the inittab file

```
S0:12345:respawn:/sbin/agetty -L 115200 console vt102
```

Errors

Most errors are transient in nature and retrying connection address these. Below table shows a list of errors and mitigation

ERROR	MITIGATION
Unable to retrieve boot diagnostics settings for ". To use the serial console, ensure that boot diagnostics is enabled for this VM.	Ensure that the VM has boot diagnostics enabled.
The VM is in a stopped deallocated state. Start the VM and retry the serial console connection.	Virtual machine must be in a started state to access the serial console
You do not have the required permissions to use this VM the serial console. Ensure you have at least VM Contributor role permissions.	Serial console access requires certain permission to access. See access requirements for details
Unable to determine the resource group for the boot diagnostics storage account ". Verify that boot diagnostics is enabled for this VM and you have access to this storage account.	Serial console access requires certain permission to access. See access requirements for details

Known issues

As we are still in the preview stages for serial console access, we are working through some known issues, below is the list of these with possible workarounds

ISSUE	MITIGATION
There is no option with virtual machine scale set instance serial console	At the time of preview, access to the serial console for virtual machine scale set instances is not supported.
Hitting enter after the connection banner does not show a log in prompt	Hitting enter does nothing

Frequently asked questions

Q. How can I send feedback?

A. Provide feedback as an issue by going to <https://aka.ms/serialconsolefeedback>. Alternatively (less preferred) Send feedback via azserialhelp@microsoft.com or in the virtual machine category of <http://feedback.azure.com>

Q.I get an Error "Existing console has conflicting OS type "Windows" with the requested OS type of Linux?

A. This is a known issue to fix this, simply open [Azure Cloud Shell](#) in bash mode and retry.

Q. I am not able to access the serial console, where can I file a support case?

A. This preview feature is covered via Azure Preview Terms. Support for this is best handled via channels mentioned above.

Next steps

- The the serial console is also available for [Windows](#) VMs
- Learn more about [bootdiagnostics](#)

Troubleshoot application connectivity issues on a Linux virtual machine in Azure

5/11/2018 • 5 min to read • [Edit Online](#)

There are various reasons when you cannot start or connect to an application running on an Azure virtual machine (VM). Reasons include the application not running or listening on the expected ports, the listening port blocked, or networking rules not correctly passing traffic to the application. This article describes a methodical approach to find and correct the problem.

If you are having issues connecting to your VM using RDP or SSH, see one of the following articles first:

- [Troubleshoot Remote Desktop connections to a Windows-based Azure Virtual Machine](#)
- [Troubleshoot Secure Shell \(SSH\) connections to a Linux-based Azure virtual machine.](#)

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.

If you need more help at any point in this article, you can contact the Azure experts on [the MSDN Azure and the Stack Overflow forums](#). Alternatively, you can also file an Azure support incident. Go to the [Azure support site](#) and select **Get Support**.

Quick-start troubleshooting steps

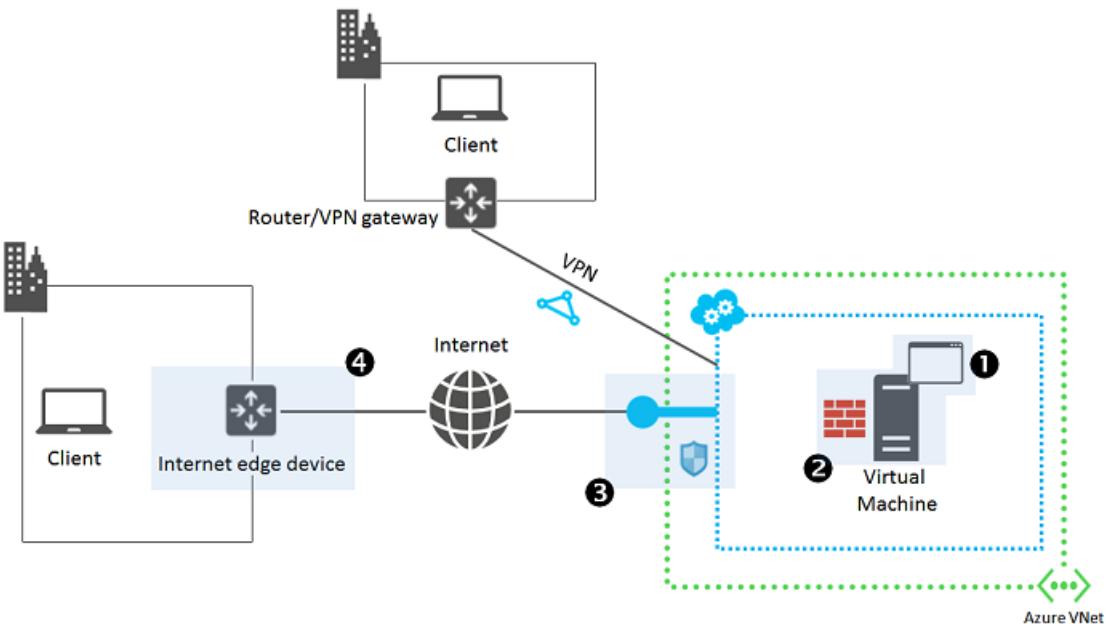
If you have problems connecting to an application, try the following general troubleshooting steps. After each step, try connecting to your application again:

- Restart the virtual machine
- Recreate the endpoint / firewall rules / network security group (NSG) rules
 - [Resource Manager model - Manage Network Security Groups](#)
 - [Classic model - Manage Cloud Services endpoints](#)
- Connect from different location, such as a different Azure virtual network
- Redeploy the virtual machine
 - [Redeploy Windows VM](#)
 - [Redeploy Linux VM](#)
- Recreate the virtual machine

For more information, see [Troubleshooting Endpoint Connectivity \(RDP/SSH/HTTP, etc. failures\)](#).

Detailed troubleshooting overview

There are four main areas to troubleshoot the access of an application that is running on an Azure virtual machine.



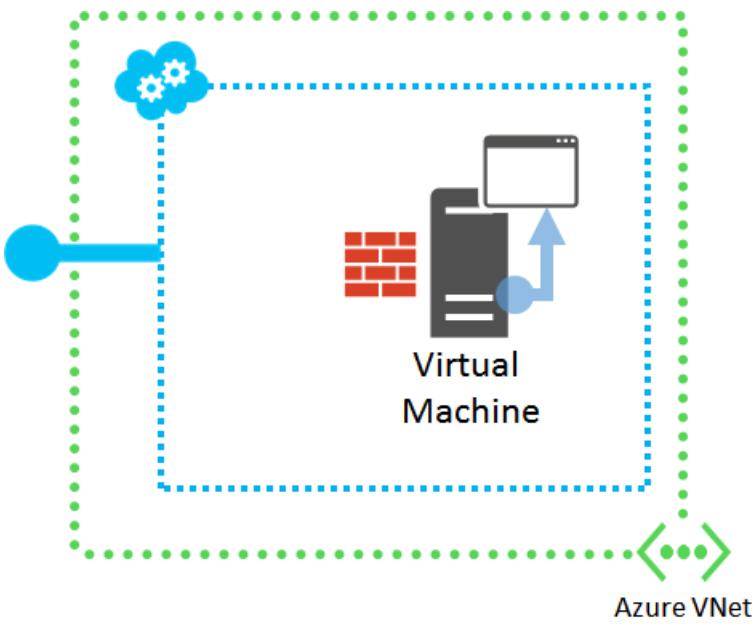
1. The application running on the Azure virtual machine.
 - Is the application itself running correctly?
2. The Azure virtual machine.
 - Is the VM itself running correctly and responding to requests?
3. Azure network endpoints.
 - Cloud service endpoints for virtual machines in the Classic deployment model.
 - Network Security Groups and inbound NAT rules for virtual machines in Resource Manager deployment model.
 - Can traffic flow from users to the VM/application on the expected ports?
4. Your Internet edge device.
 - Are firewall rules in place preventing traffic from flowing correctly?

For client computers that are accessing the application over a site-to-site VPN or ExpressRoute connection, the main areas that can cause problems are the application and the Azure virtual machine.

To determine the source of the problem and its correction, follow these steps.

Step 1: Access application from target VM

Try to access the application with the appropriate client program from the VM on which it is running. Use the local host name, the local IP address, or the loopback address (127.0.0.1).



For example, if the application is a web server, open a browser on the VM and try to access a web page hosted on the VM.

If you can access the application, go to [Step 2](#).

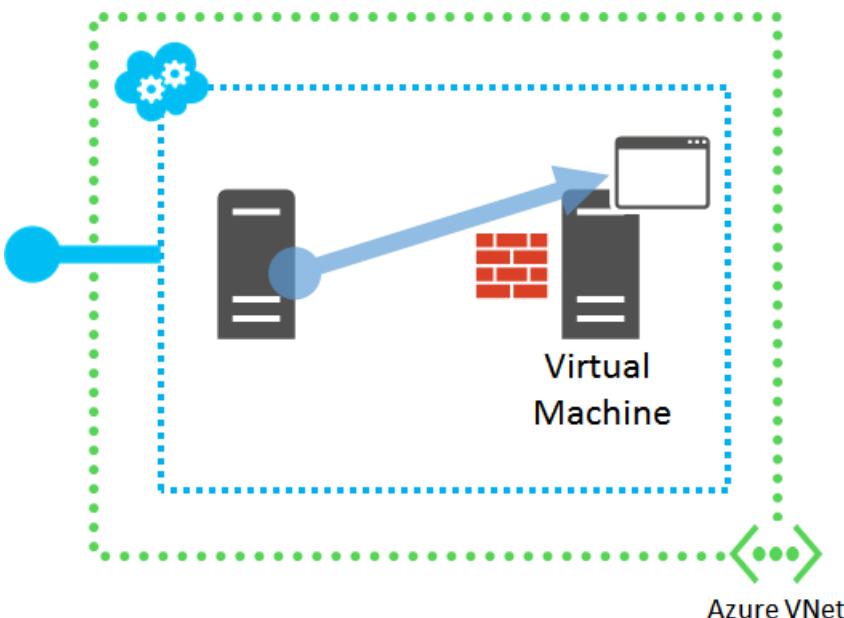
If you cannot access the application, verify the following settings:

- The application is running on the target virtual machine.
- The application is listening on the expected TCP and UDP ports.

On both Windows and Linux-based virtual machines, use the **netstat -a** command to show the active listening ports. Examine the output for the expected ports on which your application should be listening. Restart the application or configure it to use the expected ports as needed and try to access the application locally again.

Step 2: Access application from another VM in the same virtual network

Try to access the application from a different VM but in the same virtual network, using the VM's host name or its Azure-assigned public, private, or provider IP address. For virtual machines created using the classic deployment model, do not use the public IP address of the cloud service.



For example, if the application is a web server, try to access a web page from a browser on a different VM in the same virtual network.

If you can access the application, go to [Step 3](#).

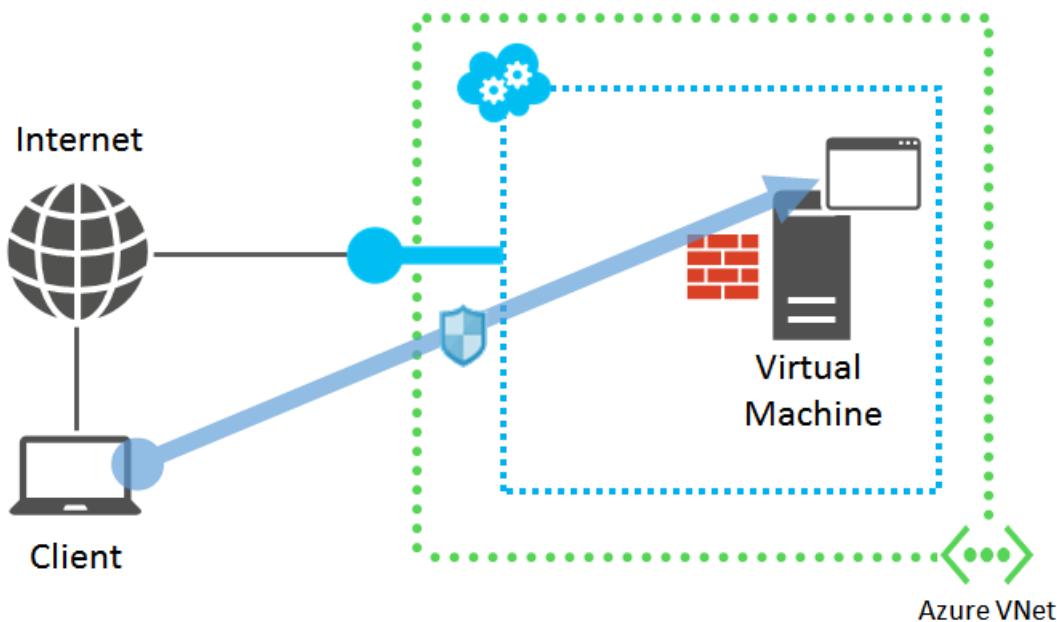
If you cannot access the application, verify the following settings:

- The host firewall on the target VM is allowing the inbound request and outbound response traffic.
- Intrusion detection or network monitoring software running on the target VM is allowing the traffic.
- Cloud Services endpoints or Network Security Groups are allowing the traffic:
 - [Classic model - Manage Cloud Services endpoints](#)
 - [Resource Manager model - Manage Network Security Groups](#)
- A separate component running in your VM in the path between the test VM and your VM, such as a load balancer or firewall, is allowing the traffic.

On a Windows-based virtual machine, use Windows Firewall with Advanced Security to determine whether the firewall rules exclude your application's inbound and outbound traffic.

Step 3: Access application from outside the virtual network

Try to access the application from a computer outside the virtual network as the VM on which the application is running. Use a different network as your original client computer.



For example, if the application is a web server, try to access the web page from a browser running on a computer that is not in the virtual network.

If you cannot access the application, verify the following settings:

- For VMs created using the classic deployment model:
 - Verify that the endpoint configuration for the VM is allowing the incoming traffic, especially the protocol (TCP or UDP) and the public and private port numbers.
 - Verify that access control lists (ACLs) on the endpoint are not preventing incoming traffic from the Internet.
 - For more information, see [How to Set Up Endpoints to a Virtual Machine](#).
- For VMs created using the Resource Manager deployment model:
 - Verify that the inbound NAT rule configuration for the VM is allowing the incoming traffic, especially the

protocol (TCP or UDP) and the public and private port numbers.

- Verify that Network Security Groups are allowing the inbound request and outbound response traffic.
- For more information, see [What is a Network Security Group \(NSG\)?](#)

If the virtual machine or endpoint is a member of a load-balanced set:

- Verify that the probe protocol (TCP or UDP) and port number are correct.
- If the probe protocol and port is different than the load-balanced set protocol and port:
 - Verify that the application is listening on the probe protocol (TCP or UDP) and port number (use **netstat -a** on the target VM).
 - Verify that the host firewall on the target VM is allowing the inbound probe request and outbound probe response traffic.

If you can access the application, ensure that your Internet edge device is allowing:

- The outbound application request traffic from your client computer to the Azure virtual machine.
- The inbound application response traffic from the Azure virtual machine.

Step 4 If you cannot access the application, use IP Verify to check the settings.

For more information, see [Azure network monitoring overview](#).

Additional resources

[Troubleshoot Remote Desktop connections to a Windows-based Azure Virtual Machine](#)

[Troubleshoot Secure Shell \(SSH\) connections to a Linux-based Azure virtual machine](#)

Troubleshoot allocation failures when you create, restart, or resize Linux VMs in Azure

4/16/2018 • 6 min to read • [Edit Online](#)

When you create a virtual machine (VM), restart stopped (deallocated) VMs, or resize a VM, Microsoft Azure allocates compute resources to your subscription. We are continually investing in additional infrastructure and features to make sure that we always have all VM types available to support customer demand. However, you may occasionally experience resource allocation failures because of unprecedented growth in demand for Azure services in specific regions. This problem can occur when you try to create or start VMs in a region while the VMs display the following error code and message:

Error code: AllocationFailed or ZonalAllocationFailed

Error message: "Allocation failed. We do not have sufficient capacity for the requested VM size in this region. Read more about improving likelihood of allocation success at <http://aka.ms/allocation-guidance>"

This article explains the causes of some of the common allocation failures and suggests possible remedies.

If your Azure issue is not addressed in this article, visit the [Azure forums on MSDN and Stack Overflow](#). You can post your issue on these forums or to @AzureSupport on Twitter. Also, you can file an Azure support request by selecting Get support on the [Azure support](#) site.

Until your preferred VM type is available in your preferred region, we advise customers who encounter deployment issues to consider the guidance in the following table as a temporary workaround.

Identify the scenario that best matches your case, and then retry the allocation request by using the corresponding suggested workaround to increase the likelihood of allocation success. Alternatively, you can always retry later. This is because enough resources may have been freed in the cluster, region, or zone to accommodate your request.

Resize a VM or add VMs to an existing availability set

Cause

A request to resize a VM or add a VM to an existing availability set must be tried at the original cluster that hosts the existing availability set. The requested VM size is supported by the cluster, but the cluster may not currently have sufficient capacity.

Workaround

If the VM can be part of a different availability set, create a VM in a different availability set (in the same region). This new VM can then be added to the same virtual network.

Stop (deallocate) all VMs in the same availability set, then restart each one. To stop: Click Resource groups > [your resource group] > Resources > [your availability set] > Virtual Machines > [your virtual machine] > Stop. After all VMs stop, select the first VM, and then click Start. This step makes sure that a new allocation attempt is run and that a new cluster can be selected that has sufficient capacity.

Restart partially stopped (deallocated) VMs

Cause

Partial deallocation means that you stopped (deallocated) one or more, but not all, VMs in an availability set. When you deallocate a VM, the associated resources are released. Restarting VMs in a partially deallocated availability set is the same as adding VMs to an existing availability set. Therefore, the allocation request must be tried at the

original cluster that hosts the existing availability set that may not have sufficient capacity.

Workaround

Stop (deallocate) all VMs in the same availability set, then restart each one. To stop: Click Resource groups > [your resource group] > Resources > [your availability set] > Virtual Machines > [your virtual machine] > Stop. After all VMs stop, select the first VM, and then click Start. This will make sure that a new allocation attempt is run and that a new cluster can be selected that has sufficient capacity.

Restart fully stopped (deallocated) VMs

Cause

Full deallocation means that you stopped (deallocated) all VMs in an availability set. The allocation request to restart these VMs will target all clusters that support the desired size within the region or zone. Change your allocation request per the suggestions in this article, and retry the request to improve the chance of allocation success.

Workaround

If you use older VM series or sizes, such as Dv1, DSv1, Av1, D15v2, or DS15v2, consider moving to newer versions. See these recommendations for specific VM sizes. If you don't have the option to use a different VM size, try deploying to a different region within the same geo. For more information about the available VM sizes in each region at <https://aka.ms/azure-regions>

If you are using availability zones, try another zone within the region that may have available capacity for the requested VM size.

If your allocation request is large (more than 500 cores), see the guidance in the following sections to break up the request into smaller deployments.

Allocation failures for older VM sizes (Av1, Dv1, DSv1, D15v2, DS15v2, etc.)

As we expand Azure infrastructure, we deploy newer-generation hardware that's designed to support the latest virtual machine types. Some of the older series VMs do not run on our latest generation infrastructure. For this reason, customers may occasionally experience allocation failures for these legacy SKUs. To avoid this problem, we encourage customers who are using legacy series virtual machines to consider moving to the equivalent newer VMs per the following recommendations: These VMs are optimized for the latest hardware and will let you take advantage of better pricing and performance.

LEGACY VM-SERIES/SIZE	RECOMMENDED NEWER VM-SERIES/SIZE	MORE INFORMATION
Av1-series	Av2-series	https://azure.microsoft.com/blog/new-av2-series-vm-sizes/
Dv1 or DSv1-series (D1 to D5)	Dv3 or DSv3-series	https://azure.microsoft.com/blog/introducing-the-new-dv3-and-ev3-vm-sizes/
Dv1 or DSv1-series (D11 to D14)	Ev3 or ESv3-series	

LEGACY VM-SERIES/SIZE	RECOMMENDED NEWER VM-SERIES/SIZE	MORE INFORMATION
D15v2 or DS15v2	If you are using the Resource Manager deployment model in order to take advantage of the larger VM sizes, consider moving to D16v3/DS16v3 or D32v3/DS32v3. These are designed to run on the latest generation hardware. If you are using the Resource Manager deployment model to make sure your VM instance is isolated to hardware dedicated to a single customer, consider moving to the new isolated VM sizes, E64i_v3 or E64is_v3, which are designed to run on the latest generation hardware.	https://azure.microsoft.com/blog/new-isolated-vm-sizes-now-available/

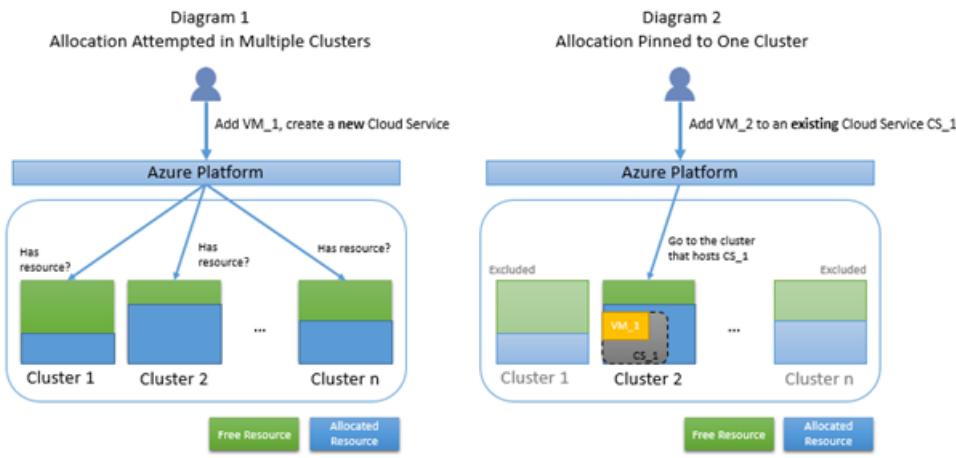
Allocation failures for large deployments (more than 500 cores)

Reduce the number of instances of the requested VM size, and then retry the deployment operation. Additionally, for larger deployments, you may want to evaluate [Azure virtual machine scale sets](#). The number of VM instances can automatically increase or decrease in response to demand or a defined schedule, and you have a greater chance of allocation success because the deployments can be spread across multiple clusters.

Background information

How allocation works

The servers in Azure datacenters are partitioned into clusters. Normally, an allocation request is attempted in multiple clusters, but it's possible that certain constraints from the allocation request force the Azure platform to attempt the request in only one cluster. In this article, we'll refer to this as "pinned to a cluster." Diagram 1 below illustrates the case of a normal allocation that is attempted in multiple clusters. Diagram 2 illustrates the case of an allocation that's pinned to Cluster 2 because that's where the existing Cloud Service CS_1 or availability set is



hosted.

Why allocation failures happen

When an allocation request is pinned to a cluster, there's a higher chance of failing to find free resources since the available resource pool is smaller. Furthermore, if your allocation request is pinned to a cluster but the type of resource you requested is not supported by that cluster, your request will fail even if the cluster has free resources. The following Diagram 3 illustrates the case where a pinned allocation fails because the only candidate cluster does not have free resources. Diagram 4 illustrates the case where a pinned allocation fails because the only candidate cluster does not support the requested VM size, even though the cluster has free resources.

Diagram 3
Allocation Failed at Pinned Cluster:
No Free Resource Available

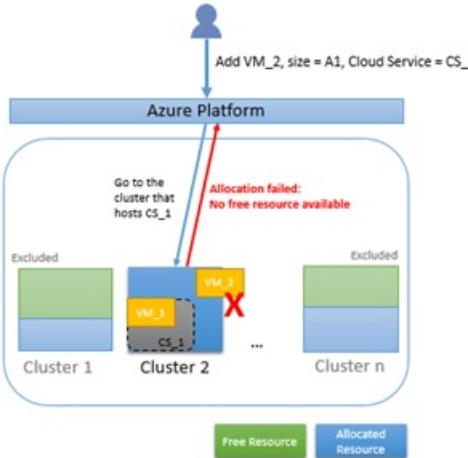
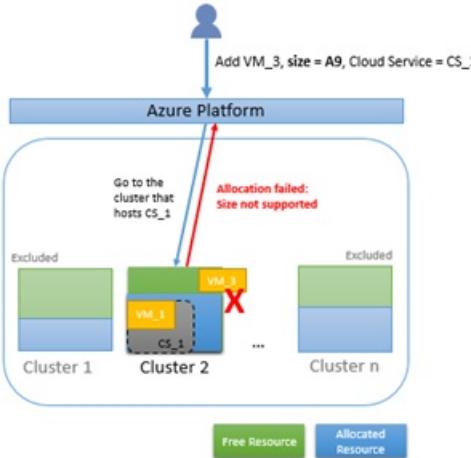


Diagram 4
Allocation Failed at Pinned Cluster:
Size not supported



Troubleshooting steps specific to allocation failure scenarios in the classic deployment model

4/16/2018 • 5 min to read • [Edit Online](#)

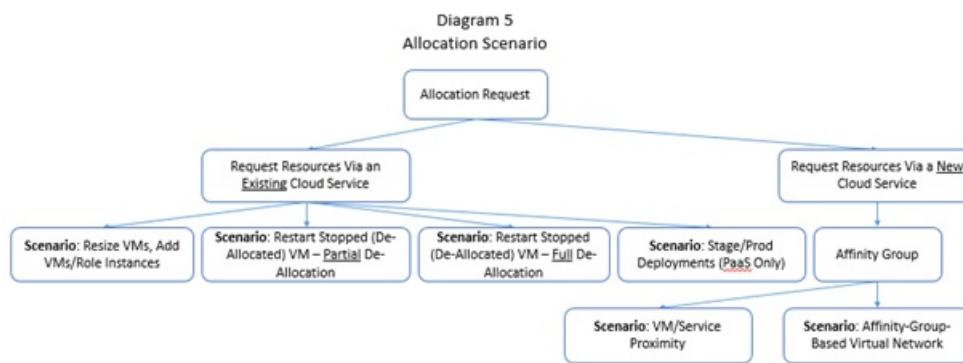
The following are common allocation scenarios that cause an allocation request to be pinned. We'll dive into each scenario later in this article.

- Resize a VM or add VMs or role instances to an existing cloud service
- Restart partially stopped (deallocated) VMs
- Restart fully stopped (deallocated) VMs
- Staging and production deployments (platform as a service only)
- Affinity group (VM or service proximity)
- Affinity-group-based virtual network

When you receive an allocation error, check whether any of the listed scenarios apply to your error. Use the allocation error that's returned by the Azure platform to identify the corresponding scenario. If your request is pinned, remove some of the pinning constraints to open your request to more clusters, thereby increasing the chance of allocation success. In general, if the error does not state that "the requested VM size is not supported," you can always retry at a later time. This is because enough resources may have been freed in the cluster to accommodate your request. If the problem is that the requested VM size is not supported, try a different VM size. Otherwise, the only option is to remove the pinning constraint.

Two common failure scenarios are related to affinity groups. In the past, an affinity group was used to provide close proximity to VMs and service instances, or it was used to enable the creation of a virtual network. With the introduction of regional virtual networks, affinity groups are no longer required to create a virtual network. With the reduction of network latency in Azure infrastructure, the recommendation to use affinity groups for VMs or service proximity has changed.

The following Diagram presents the taxonomy of the (pinned) allocation scenarios.



Resize a VM or add VMs or role instances to an existing cloud service

Error

Upgrade_VMSizeNotSupported or GeneralError

Cause of cluster pinning

A request to resize a VM or add a VM or a role instance to an existing cloud service has to be attempted at the original cluster that hosts the existing cloud service. Creating a new cloud service allows the Azure platform to find

another cluster that has free resources or supports the VM size that you requested.

Workaround

If the error is Upgrade_VMSizeNotSupported*, try a different VM size. If using a different VM size is not an option, but if it's acceptable to use a different virtual IP address (VIP), create a new cloud service to host the new VM and add the new cloud service to the regional virtual network where the existing VMs are running. If your existing cloud service does not use a regional virtual network, you can still create a new virtual network for the new cloud service, and then connect your [existing virtual network to the new virtual network](#). See more about [regional virtual networks](#).

If the error is GeneralError*, it's likely that the type of resource (such as a particular VM size) is supported by the cluster, but the cluster does not have free resources at the moment. Similar to the above scenario, add the desired compute resource through creating a new cloud service (note that the new cloud service has to use a different VIP) and use a regional virtual network to connect your cloud services.

Restart partially stopped (deallocated) VMs

Error

GeneralError*

Cause of cluster pinning

Partial deallocation means that you stopped (deallocated) one or more, but not all, VMs in a cloud service. When you stop (deallocate) a VM, the associated resources are released. Restarting that stopped (deallocated) VM is therefore a new allocation request. Restarting VMs in a partially deallocated cloud service is equivalent to adding VMs to an existing cloud service. The allocation request has to be attempted at the original cluster that hosts the existing cloud service. Creating a different cloud service allows the Azure platform to find another cluster that has free resource or supports the VM size that you requested.

Workaround

If it's acceptable to use a different VIP, delete the stopped (deallocated) VMs (but keep the associated disks) and add the VMs back through a different cloud service. Use a regional virtual network to connect your cloud services:

- If your existing cloud service uses a regional virtual network, simply add the new cloud service to the same virtual network.
- If your existing cloud service does not use a regional virtual network, create a new virtual network for the new cloud service, and then [connect your existing virtual network to the new virtual network](#). See more about [regional virtual networks](#).

Restart fully stopped (deallocated) VMs

Error

GeneralError*

Cause of cluster pinning

Full deallocation means that you stopped (deallocated) all VMs from a cloud service. The allocation requests to restart these VMs have to be attempted at the original cluster that hosts the cloud service. Creating a new cloud service allows the Azure platform to find another cluster that has free resources or supports the VM size that you requested.

Workaround

If it's acceptable to use a different VIP, delete the original stopped (deallocated) VMs (but keep the associated disks)

and delete the corresponding cloud service (the associated compute resources were already released when you stopped (deallocated) the VMs). Create a new cloud service to add the VMs back.

Staging/production deployments (platform as a service only)

Error

New_General* or New_VMSizeNotSupported*

Cause of cluster pinning

The staging deployment and the production deployment of a cloud service are hosted in the same cluster. When you add the second deployment, the corresponding allocation request will be attempted in the same cluster that hosts the first deployment.

Workaround

Delete the first deployment and the original cloud service and redeploy the cloud service. This action may land the first deployment in a cluster that has enough free resources to fit both deployments or in a cluster that supports the VM sizes that you requested.

Affinity group (VM/service proximity)

Error

New_General* or New_VMSizeNotSupported*

Cause of cluster pinning

Any compute resource assigned to an affinity group is tied to one cluster. New compute resource requests in that affinity group are attempted in the same cluster where the existing resources are hosted. This is true whether the new resources are created through a new cloud service or through an existing cloud service.

Workaround

If an affinity group is not necessary, do not use an affinity group, or group your compute resources into multiple affinity groups.

Affinity-group-based virtual network

Error

New_General* or New_VMSizeNotSupported*

Cause of cluster pinning

Before regional virtual networks were introduced, you were required to associate a virtual network with an affinity group. As a result, compute resources placed into an affinity group are bound by the same constraints as described in the "Allocation scenario: Affinity group (VM/service proximity)" section above. The compute resources are tied to one cluster.

Workaround

If you do not need an affinity group, create a new regional virtual network for the new resources you're adding, and then [connect your existing virtual network to the new virtual network](#). See more about [regional virtual networks](#).

Alternatively, you can [migrate your affinity-group-based virtual network to a regional virtual network](#), and then add the desired resources again.

Troubleshoot deploying Linux virtual machine issues in Azure

5/11/2018 • 3 min to read • [Edit Online](#)

To troubleshoot virtual machine (VM) deployment issues in Azure, review the [top issues](#) for common failures and resolutions.

If you need more help at any point in this article, you can contact the Azure experts on [the MSDN Azure and Stack Overflow forums](#). Alternatively, you can file an Azure support incident. Go to the [Azure support site](#) and select **Get Support**.

Top issues

The following top issues may help resolve your issue. To start troubleshooting, review these steps:

- [The cluster cannot support the requested VM size](#)
- [The cluster does not have free resources](#)

The cluster cannot support the requested VM size

- Retry the request using a smaller VM size.
- If the size of the requested VM cannot be changed:
 - Stop all the VMs in the availability set. Click **Resource groups** > your resource group > **Resources** > your availability set > **Virtual Machines** > your virtual machine > **Stop**.
 - After all the VMs stop, create the VM in the desired size.
 - Start the new VM first, and then select each of the stopped VMs and click Start.

The cluster does not have free resources

- Retry the request later.
- If the new VM can be part of a different availability set
 - Create a VM in a different availability set (in the same region).
 - Add the new VM to the same virtual network.

How do I activate my monthly credit for Visual studio Enterprise (BizSpark)

To activate your monthly credit, see this [article](#).

Why can I not install the GPU driver for an Ubuntu NV VM?

Currently, Linux GPU support is only available on Azure NC VMs running Ubuntu Server 16.04 LTS. For more information, see [Set up GPU drivers for N-series VMs running Linux](#).

My drivers are missing for my Linux N-Series VM

Drivers for Linux-based VMs are located [here](#).

I can't find a GPU instance within my N-Series VM

To take advantage of the GPU capabilities of Azure N-series VMs running Windows Server 2016 or Windows Server 2012 R2, you must install NVIDIA graphics drivers on each VM after deployment. Driver setup information is available for [Windows VMs](#) and [Linux VMs](#).

Is N-Series VMs available in my region?

You can check the availability from the [Products available by region table](#), and pricing [here](#).

I am not able to see VM Size family that I want when resizing my VM.

When a VM is running, it is deployed to a physical server. The physical servers in Azure regions are grouped in clusters of common physical hardware. Resizing a VM that requires the VM to be moved to different hardware clusters is different depending on which deployment model was used to deploy the VM.

- VMs deployed in Classic deployment model, the cloud service deployment must be removed and redeployed to change the VMs to a size in another size family.
- VMs deployed in Resource Manager deployment model, you must stop all VMs in the availability set before changing the size of any VM in the availability set.

The listed VM size is not supported while deploying in Availability Set.

Choose a size that is supported on the availability set's cluster. It is recommended when creating an availability set to choose the largest VM size you think you need, and have that be your first deployment to the Availability set.

What Linux distributions/versions are supported on Azure?

You can find the list at Linux on [Azure-Endorsed Distributions](#).

Can I add an existing Classic VM to an availability set?

Yes. You can add an existing classic VM to a new or existing Availability Set. For more information see [Add an existing virtual machine to an availability set](#).

Next steps

If you need more help at any point in this article, you can contact the Azure experts on [the MSDN Azure and Stack Overflow forums](#).

Alternatively, you can file an Azure support incident. Go to the [Azure support site](#) and select **Get Support**.

Troubleshoot Resource Manager deployment issues with creating a new Linux virtual machine in Azure

1/3/2018 • 4 min to read • [Edit Online](#)

When you try to create a new Azure Virtual Machine (VM), the common errors you encounter are provisioning failures or allocation failures.

- A provisioning failure happens when the OS image fails to load either due to incorrect preparatory steps or because of selecting the wrong settings during the image capture from the portal.
- An allocation failure results when the cluster or region either does not have resources available or cannot support the requested VM size.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Top issues

The following top issues may help resolve your issue. To start troubleshooting, review these steps:

- [The cluster cannot support the requested VM size](#)
- [The cluster does not have free resources](#)

For other VM deployment issues and questions, see [Troubleshoot deploying Linux virtual machine issues in Azure](#).

Collect activity logs

To start troubleshooting, collect the activity logs to identify the error associated with the issue. The following links contain detailed information on the process to follow.

[View deployment operations](#)

[View activity logs to manage Azure resources](#)

Issue: Custom image; provisioning errors

Provisioning errors arise if you upload or capture a generalized VM image as a specialized VM image or vice versa. The former will cause a provisioning timeout error and the latter will cause a provisioning failure. To deploy your custom image without errors, you must ensure that the type of the image does not change during the capture process.

The following table lists the possible combinations of generalized and specialized images, the error type you will encounter and what you need to do to fix the errors.

The following table lists the possible upload and capture combinations of Linux generalized and specialized OS images. The combinations that will process without any errors are indicated by a Y, and those that will throw errors are indicated by an N. The causes and resolutions for the different errors you will run into are given below the table.

OS	UPLOAD SPEC.	UPLOAD GEN.	CAPTURE SPEC.	CAPTURE GEN.
Linux gen.	N ¹	Y	N ³	Y
Linux spec.	Y	N ²	Y	N ⁴

Y: If the OS is Linux generalized, and it is uploaded and/or captured with the generalized setting, then there won't be any errors. Similarly, if the OS is Linux specialized, and it is uploaded and/or captured with the specialized setting, then there won't be any errors.

Upload Errors:

N¹: If the OS is Linux generalized, and it is uploaded as specialized, you will get a provisioning timeout error because the VM is stuck at the provisioning stage.

N²: If the OS is Linux specialized, and it is uploaded as generalized, you will get a provisioning failure error because the new VM is running with the original computer name, username and password.

Resolution:

To resolve both these errors, upload the original VHD, available on-prem, with the same setting as that for the OS (generalized/specialized). To upload as generalized, remember to run -deprovision first.

Capture Errors:

N³: If the OS is Linux generalized, and it is captured as specialized, you will get a provisioning timeout error because the original VM is not usable as it is marked as generalized.

N⁴: If the OS is Linux specialized, and it is captured as generalized, you will get a provisioning failure error because the new VM is running with the original computer name, username and password. Also, the original VM is not usable because it is marked as specialized.

Resolution:

To resolve both these errors, delete the current image from the portal, and [recapture it from the current VHDs](#) with the same setting as that for the OS (generalized/specialized).

Issue: Custom/ gallery/ marketplace image; allocation failure

This error arises in situations when the new VM request is pinned to a cluster that either cannot support the VM size being requested, or does not have available free space to accommodate the request.

Cause 1: The cluster cannot support the requested VM size.

Resolution 1:

- Retry the request using a smaller VM size.
- If the size of the requested VM cannot be changed:
 - Stop all the VMs in the availability set. Click **Resource groups** > *your resource group* > **Resources** > *your availability set* > **Virtual Machines** > *your virtual machine* > **Stop**.
 - After all the VMs stop, create the new VM in the desired size.
 - Start the new VM first, and then select each of the stopped VMs and click **Start**.

Cause 2: The cluster does not have free resources.

Resolution 2:

- Retry the request at a later time.

- If the new VM can be part of a different availability set
 - Create a new VM in a different availability set (in the same region).
 - Add the new VM to the same virtual network.

Next steps

If you encounter issues when you start a stopped Linux VM or resize an existing Linux VM in Azure, see [Troubleshoot Resource Manager deployment issues with restarting or resizing an existing Linux Virtual Machine in Azure](#).

Troubleshoot Linux VM device name changes

5/11/2018 • 3 min to read • [Edit Online](#)

This article explains why device names change after you restart a Linux VM or reattach the data disks. The article also provides solutions for this problem.

Symptoms

You may experience the following problems when running Linux VMs in Microsoft Azure:

- The VM fails to boot after a restart.
- When data disks are detached and reattached, the disk device names are changed.
- An application or script that references a disk by using the device name fails because the device name has changed.

Cause

Device paths in Linux aren't guaranteed to be consistent across restarts. Device names consist of major numbers (letters) and minor numbers. When the Linux storage device driver detects a new device, the driver assigns major and minor numbers from the available range to the device. When a device is removed, the device numbers are freed for reuse.

The problem occurs because device scanning in Linux is scheduled by the SCSI subsystem to happen asynchronously. As a result, a device path name can vary across restarts.

Solution

To resolve this problem, use persistent naming. There are four ways to use persistent naming: by filesystem label, by UUID, by ID, or by path. We recommend using the filesystem label or UUID for Azure Linux VMs.

Most distributions provide the `fstab` **nofail** or **nobootwait** parameters. These parameters enable a system to boot when the disk fails to mount at startup. Check your distribution documentation for more information about these parameters. For information on how to configure a Linux VM to use a UUID when you add a data disk, see [Connect to the Linux VM to mount the new disk](#).

When the Azure Linux agent is installed on a VM, the agent uses Udev rules to construct a set of symbolic links under the `/dev/disk/azure` path. Applications and scripts use Udev rules to identify disks that are attached to the VM, along with the disk type and disk LUNs.

Identify disk LUNs

Applications use LUNs to find all of the attached disks and to construct symbolic links. The Azure Linux agent includes Udev rules that set up symbolic links from a LUN to the devices:

```
$ tree /dev/disk/azure
/dev/disk/azure
└── resource -> ../../sdb
    ├── resource-part1 -> ../../sdb1
    ├── root -> ../../sda
    ├── root-part1 -> ../../sda1
    └── scsi1
        ├── lun0 -> ../../sdc
        ├── lun0-part1 -> ../../sdc1
        ├── lun1 -> ../../sdd
        ├── lun1-part1 -> ../../sdd1
        ├── lun1-part2 -> ../../sdd2
        └── lun1-part3 -> ../../sdd3
```

LUN information from the Linux guest account is retrieved by using `lsscsi` or a similar tool:

```
$ sudo lsscsi
[1:0:0:0] cd/dvd Msft Virtual CD/ROM 1.0 /dev/sr0
[2:0:0:0] disk Msft Virtual Disk 1.0 /dev/sda
[3:0:1:0] disk Msft Virtual Disk 1.0 /dev/sdb
[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 guest LUN information is used with Azure subscription metadata to locate the VHD in Azure Storage that contains the partition data. For example, you can use the `az` CLI:

```
$ az vm show --resource-group testVM --name testVM | jq -r .storageProfile.dataDisks
[
{
  "caching": "None",
  "createOption": "empty",
  "diskSizeGb": 1023,
  "image": null,
  "lun": 0,
  "managedDisk": null,
  "name": "testVM-20170619-114353",
  "vhd": {
    "uri": "https://testVM.blob.core.windows.net/vhd/testVM-20170619-114353.vhd"
  }
},
{
  "caching": "None",
  "createOption": "empty",
  "diskSizeGb": 512,
  "image": null,
  "lun": 1,
  "managedDisk": null,
  "name": "testVM-20170619-121516",
  "vhd": {
    "uri": "https://testVM.blob.core.windows.net/vhd/testVM-20170619-121516.vhd"
  }
}]
```

Discover filesystem UUIDs by using blkid

Applications and scripts read the output of `blkid`, or similar sources of information, to construct symbolic links in the /dev path. The output shows the UUIDs of all disks that are attached to the VM and their associated device file:

```
$ sudo blkid -s UUID  
  
/dev/sr0: UUID="120B021372645f72"  
/dev/sda1: UUID="52c6959b-79b0-4bdd-8ed6-71e0ba782fb4"  
/dev/sdb1: UUID="176250df-9c7c-436f-94e4-d13f9bdea744"  
/dev/sdc1: UUID="b0048738-4ecc-4837-9793-49ce296d2692"
```

The Azure Linux agent Udev rules construct a set of symbolic links under the /dev/disk/azure path:

```
$ ls -l /dev/disk/azure  
  
total 0  
lrwxrwxrwx 1 root root 9 Jun  2 23:17 resource -> ../../sdb  
lrwxrwxrwx 1 root root 10 Jun  2 23:17 resource-part1 -> ../../sdb1  
lrwxrwxrwx 1 root root 9 Jun  2 23:17 root -> ../../sda  
lrwxrwxrwx 1 root root 10 Jun  2 23:17 root-part1 -> ../../sda1
```

Applications use the links to identify the boot disk device and the resource (ephemeral) disk. In Azure, applications should look in the /dev/disk/azure/root-part1 or /dev/disk/azure-resource-part1 paths to discover these partitions.

Any additional partitions from the `blkid` list reside on a data disk. Applications maintain the UUID for these partitions and use a path to discover the device name at runtime:

```
$ ls -l /dev/disk/by-uuid/b0048738-4ecc-4837-9793-49ce296d2692  
  
lrwxrwxrwx 1 root root 10 Jun 19 15:57 /dev/disk/by-uuid/b0048738-4ecc-4837-9793-49ce296d2692 -> ../../sdc1
```

Get the latest Azure Storage rules

To get the latest Azure Storage rules, run the following commands:

```
# sudo curl -o /etc/udev/rules.d/66-azure-storage.rules  
https://raw.githubusercontent.com/Azure/WALinuxAgent/master/config/66-azure-storage.rules  
# sudo udevadm trigger --subsystem-match=block
```

See also

For more information, see the following articles:

- [Ubuntu: Using UUID](#)
- [Red Hat: Persistent naming](#)
- [Linux: What UUIDs can do for you](#)
- [Udev: Introduction to device management in a modern Linux system](#)

Redeploy Linux virtual machine to new Azure node

3/8/2018 • 1 min to read • [Edit Online](#)

If you face difficulties troubleshooting SSH or application access to a Linux virtual machine (VM) in Azure, redeploying the VM may help. When you redeploy a VM, it moves the VM to a new node within the Azure infrastructure and then powers it back on. All your configuration options and associated resources are retained. This article shows you how to redeploy a VM using Azure CLI or the Azure portal.

NOTE

After you redeploy a VM, the temporary disk is lost and dynamic IP addresses associated with virtual network interface are updated.

You can redeploy a VM using one of the following options. You only need to choose one option to redeploy your VM:

- [Azure CLI 2.0](#)
- [Azure CLI 1.0](#)
- [Azure portal](#)

Use the Azure CLI 2.0

Install the latest [Azure CLI 2.0](#) and log in to your Azure account using [az login](#).

Redeploy your VM with [az vm redeploy](#). The following example redeploys the VM named *myVM* in the resource group named *myResourceGroup*:

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

Use the Azure CLI 1.0

Install the [latest Azure CLI 1.0](#) and log in to your Azure account. Make sure that you are in Resource Manager mode (`azure config mode arm`).

The following example redeploys the VM named *myVM* in the resource group named *myResourceGroup*:

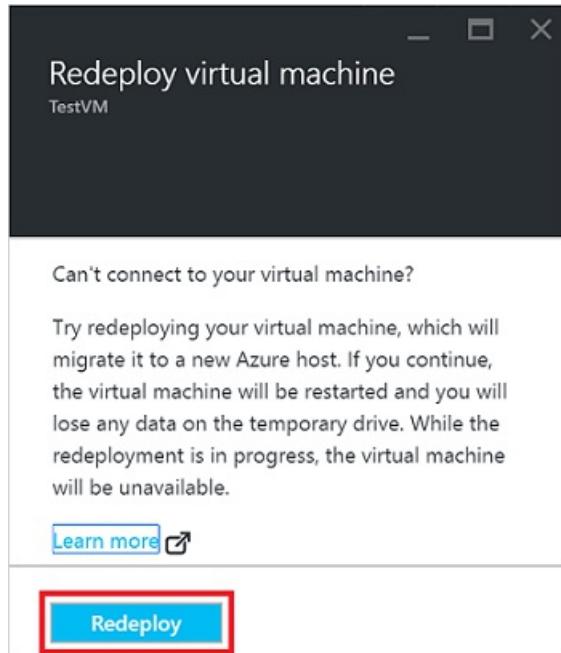
```
azure vm redeploy --resource-group myResourceGroup --vm-name myVM
```

Use the Azure portal

1. Select the VM you wish to redeploy, then select the *Redeploy* button in the *Settings* blade. You may need to scroll down to see the **Support and Troubleshooting** section that contains the 'Redeploy' button as in the following example:

The screenshot shows the Azure portal interface for a virtual machine named 'TestVM'. The left sidebar contains navigation links for Disks, Extensions, Network interfaces, Size, Properties, Locks, Automation script, MONITORING (Alert rules, Diagnostics, Diagram), SUPPORT + TROUBLESHOOTING (Resource health, Boot diagnostics, Reset password, Redeploy, New support request). The main content area is divided into 'Essentials' and 'Monitoring' sections. In the 'Essentials' section, details about the resource group (TestRG), status (Running), location (West US), subscription name, computer name (TestVM), operating system (Linux), size (Standard DS1 (1 core, 3.5 GB memory)), public IP address (13.93.235.59/testv-westu-worh9mzo082g...), and virtual network/subnet (testv-westu-worh9mzo082g vnet/testv-we...) are displayed. The 'Monitoring' section shows a chart for CPU percentage with a message 'No available data.' The 'Redeploy' button in the support section is highlighted with a red box.

2. To confirm the operation, select the *Redeploy* button:



3. The **Status** of the VM changes to *Updating* as the VM prepares to redeploy, as shown in the following example:

The screenshot shows the Azure portal interface for a virtual machine named "TestVM" located in the "TestRG" resource group. The status of the VM is currently "Updating". The configuration pane displays the following details:

Setting	Value
Resource group	TestRG
Status	Updating
Location	West US
Subscription name	[REDACTED]
Subscription ID	[REDACTED]
Computer name	TestVM
Operating system	Linux
Size	Standard D1 v2 (1 core, 3.5 GB memory)
Public IP address/DNS name label	40.78.108.205/<none>
Virtual network/subnet	TestRG/default

[All settings →](#)

4. The **Status** then changes to *Starting* as the VM boots up on a new Azure host, as shown in the following example:

The screenshot shows the Azure portal interface for the same virtual machine "TestVM" in the "TestRG" resource group. The status of the VM is now "Starting". The configuration pane displays the following details:

Setting	Value
Resource group	TestRG
Status	Starting
Location	West US
Subscription name	[REDACTED]
Subscription ID	[REDACTED]
Computer name	TestVM
Operating system	Linux
Size	Standard D1 v2 (1 core, 3.5 GB memory)
Public IP address/DNS name label	40.78.108.205/<none>
Virtual network/subnet	TestRG/default

[All settings →](#)

5. After the VM finishes the boot process, the **Status** then returns to *Running*, indicating the VM has been successfully redeployed:

The screenshot shows the Azure portal interface for a virtual machine named 'TestVM'. At the top, there are action buttons: Settings, Connect, Start, Restart, Stop, and Delete. Below this is a navigation bar with 'Essentials' and three icons: a cloud, a person, and a tag. The main content area displays the following details:

Resource group	Computer name
TestRG	TestVM
Status	Operating system
Running	Linux
Location	Size
West US	Standard D1 v2 (1 core, 3.5 GB memory)
Subscription name	Public IP address/DNS name label
	40.78.108.205/<none>
Subscription ID	Virtual network/subnet
	TestRG/default

At the bottom right of the content area is a blue button labeled 'All settings →'.

Next steps

If you are having issues connecting to your VM, you can find specific help on [troubleshooting SSH connections](#) or [detailed SSH troubleshooting steps](#). If you cannot access an application running on your VM, you can also read [application troubleshooting issues](#).

Understand common error messages when you manage Linux virtual machines in Azure

4/9/2018 • 15 min to read • [Edit Online](#)

This article describes some of the most common error codes and messages you may encounter when you create or manage Linux virtual machines (VMs) in Azure.

NOTE

You can leave comments on this page for feedback or through [Azure feedback](#) with #azerrormessage tag.

Error Response Format

Azure VMs use the following JSON format for error response:

```
{  
  "status": "status code",  
  "error": {  
    "code": "Top level error code",  
    "message": "Top level error message",  
    "details": [  
      {  
        "code": "Inner level error code",  
        "message": "Inner level error message"  
      }  
    ]  
  }  
}
```

An error response always includes a status code and an error object. Each error object always contains an error code and a message. If the VM is created with a template, the error object also contains a details section that contains an inner level of error codes and message. Normally, the most inner level of error message is the root failure.

Common virtual machine management errors

This section lists the common error messages you may encounter when managing VMs:

ERROR CODE	ERROR MESSAGE
AcquireDiskLeaseFailed	Failed to acquire lease while creating disk '{0}' using blob with URI {1}. Blob is already in use.
AllocationFailed	Allocation failed. Please try reducing the VM size or number of VMs, retry later, or try deploying to a different Availability Set or different Azure location.
AllocationFailed	The VM allocation failed due to an internal error. Please retry later or try deploying to a different location.

ERROR CODE	ERROR MESSAGE
ArtifactNotFound	The VM extension with publisher '{0}' and type '{1}' could not be found in location '{2}'.
ArtifactNotFound	Extension with publisher '{0}', type '{1}', and type handler version '{2}' could not be found in the extension repository.
ArtifactVersionNotFound	No version found in the artifact repository that satisfies the requested version '{0}'.
ArtifactVersionNotFound	No version found in the artifact repository that satisfies the requested version '{0}' for VM extension with publisher '{1}' and type '{2}'.
AttachDiskWhileBeingDetached	Cannot attach data disk '{0}' to VM '{1}' because the disk is currently being detached. Please wait until the disk is completely detached and then try again.
BadRequest	'Aligned' Availability Sets are not yet supported in this region.
BadRequest	Addition of a VM with managed disks to non-managed Availability Set or addition of a VM with blob based disks to managed Availability Set is not supported. Please create an Availability Set with 'managed' property set in order to add a VM with managed disks to it.
BadRequest	Managed Disks are not supported in this region.
BadRequest	Multiple VMExtensions per handler not supported for OS type '{0}'. VMExtension '{1}' with handler '{2}' already added or specified in input.
BadRequest	Operation '{0}' is not supported on Resource '{1}' with managed disks.
CertificateImproperlyFormatted	The secret's JSON representation retrieved from {0} has a data field which is not a properly formatted PFX file, or the password provided does not decode the PFX file correctly.
CertificateImproperlyFormatted	The data retrieved from {0} is not deserializable into JSON.
Conflict	Disk resizing is allowed only when creating a VM or when the VM is deallocated.
ConflictingUserInput	Disk '{0}' cannot be attached as the disk is already owned by VM '{1}'.
ConflictingUserInput	Source and destination resource groups are the same.
ConflictingUserInput	Source and destination storage accounts for disk {0} are different.
ContainerAlreadyOnLease	There is already a lease on the storage container holding the blob with URI {0}.

ERROR CODE	ERROR MESSAGE
CrossSubscriptionMoveWithKeyVaultResources	The Move resources request contains KeyVault resources which are referenced by one or more {0}s in the request. This is not supported currently in Cross subscription Move. Please check the error details for the KeyVault resource Ids.
DiagnosticsOperationInternalError	An internal error occurred while processing diagnostics profile of VM {0}.
DiskBlobAlreadyInUseByAnotherDisk	Blob {0} is already in use by another disk belonging to VM '{1}'. You can examine the blob metadata for the disk reference information.
DiskBlobNotFound	Unable to find VHD blob with URI {0} for disk '{1}'.
DiskBlobNotFound	Unable to find VHD blob with URI {0}.
DiskEncryptionKeySecretMissingTags	{0} secret doesn't have the {1} tags. Please update the secret version, add the required tags and retry.
DiskEncryptionKeySecretUnwrapFailed	Unwrap of secret {0} value using key {1} failed.
DiskImageNotReady	Disk image {0} is in {1} state. Please retry when image is ready.
DiskPreparationError	One or more errors occurred while preparing VM disks. See disk instance view for details.
DiskProcessingError	Disk processing halted as the VM has other disks in failed disks.
ImageBlobNotFound	Unable to find VHD blob with URI {0} for disk '{1}'.
ImageBlobNotFound	Unable to find VHD blob with URI {0}.
IncorrectDiskBlobType	Disk blobs can only be of type page blob. Blob {0} for disk '{1}' is of type block blob.
IncorrectDiskBlobType	Disk blobs can only be of type page blob. Blob {0} is of type '{1}'.
IncorrectImageBlobType	Disk blobs can only be of type page blob. Blob {0} for disk '{1}' is of type block blob.
IncorrectImageBlobType	Disk blobs can only be of type page blob. Blob {0} is of type '{1}'.
InternalOperationError	Could not resolve storage account {0}. Please ensure it was created through the Storage Resource Provider in the same location as the compute resource.
InternalOperationError	{0} goal seeking tasks failed.
InternalOperationError	Error occurred in validating the network profile of VM '{0}'.

ERROR CODE	ERROR MESSAGE
InvalidAccountType	The AccountType {0} is invalid.
InvalidParameter	The value of parameter {0} is invalid.
InvalidParameter	The Admin password specified is not allowed.
InvalidParameter	"The supplied password must be between {0}-{1} characters long and must satisfy at least {2} of password complexity requirements from the following: 1. Contains an uppercase character 2. Contains a lowercase character 3. Contains a numeric digit 4. Contains a special character.
InvalidParameter	The Admin Username specified is not allowed.
InvalidParameter	Cannot attach an existing OS disk if the VM is created from a platform or user image.
InvalidParameter	Container name {0} is invalid. Container names must be 3-63 characters in length and may contain only lower-case alphanumeric characters and hyphen. Hyphen must be preceded and followed by an alphanumeric character.
InvalidParameter	Container name {0} in URL {1} is invalid. Container names must be 3-63 characters in length and may contain only lower-case alphanumeric characters and hyphen. Hyphen must be preceded and followed by an alphanumeric character.
InvalidParameter	The blob name in URL {0} contains a slash. This is presently not supported for disks.
InvalidParameter	The URI {0} does not look to be correct blob URI.
InvalidParameter	A disk named '{0}' already uses the same LUN: {1}.
InvalidParameter	A disk named '{0}' already exists.
InvalidParameter	Cannot specify user image overrides for a disk already defined in the specified image reference.
InvalidParameter	A disk named '{0}' already uses the same VHD URL {1}.
InvalidParameter	The specified fault domain count {0} must fall in the range {1} to {2}.
InvalidParameter	The license type {0} is invalid. Valid license types are: Windows_Client or Windows_Server, case sensitive.
InvalidParameter	Linux host name cannot exceed {0} characters in length or contain the following characters: {1}.

ERROR CODE	ERROR MESSAGE
InvalidParameter	Destination path for Ssh public keys is currently limited to its default value {0} due to a known issue in Linux provisioning agent.
InvalidParameter	A disk at LUN {0} already exists.
InvalidParameter	Subscription {0} of the request must match the subscription {1} contained in the managed disk id.
InvalidParameter	Custom data in OSProfile must be in Base64 encoding and with a maximum length of {0} characters.
InvalidParameter	Blob name in URL {0} must end with '{1}' extension.
InvalidParameter	{0}' is not a valid captured VHD blob name prefix. A valid prefix matches regex '{1}'.
InvalidParameter	Certificates cannot be added to your VM if the VM agent is not provisioned.
InvalidParameter	A disk at LUN {0} already exists.
InvalidParameter	Unable to create the VM because the requested size {0} is not available in the cluster where the availability set is currently allocated. The available sizes are: {1}. Read more on VM resizing strategy at https://aka.ms/azure-resizevm .
InvalidParameter	The requested VM size {0} is not available in the current region. The sizes available in the current region are: {1}. Find out more on the available VM sizes in each region at https://aka.ms/azure-regions .
InvalidParameter	The requested VM size {0} is not available in the current region. Find out more on the available VM sizes in each region at https://aka.ms/azure-regions .
InvalidParameter	Windows admin user name cannot be more than {0} characters long, end with a period(.), or contain the following characters: {1}.
InvalidParameter	Windows computer name cannot be more than {0} characters long, be entirely numeric, or contain the following characters: {1}.
MissingMoveDependentResources	The move resources request does not contain all the dependent resources. Please check error details for missing resource ids.
MoveResourcesHaveInvalidState	The Move Resources request contains VMs which are associated with invalid storage accounts. Please check details for these resource ids and referenced storage account names.

ERROR CODE	ERROR MESSAGE
MoveResourcesHavePendingOperations	The move resources request contains resources for which an operation is pending. Please check details for these resource ids. Retry your operation once the pending operations complete.
MoveResourcesNotFound	The move resources request contains resources that cannot be found. Please check details for these resource ids.
NetworkingInternalOperationError	Unknown network allocation error.
NetworkingInternalOperationError	Unknown network allocation error
NetworkingInternalOperationError	An internal error occurred in processing network profile of the VM.
NotFound	The Availability Set {0} cannot be found.
NotFound	Source Virtual Machine '{0}' specified in the request does not exist in this Azure location.
NotFound	Tenant with id {0} not found.
NotFound	The Image {0} cannot be found.
NotSupported	The license type is {0}, but the image blob {1} is not from on-premises.
OperationNotAllowed	Availability Set {0} cannot be deleted. Before deleting an Availability Set please ensure that it does not contain any VM.
OperationNotAllowed	Changing availability set SKU from 'Aligned' to 'Classic' is not allowed.
OperationNotAllowed	Cannot modify extensions in the VM when the VM is not running.
OperationNotAllowed	The Capture action is only supported on a Virtual Machine with blob based disks. Please use the 'Image' resource APIs to create an Image from a managed Virtual Machine.
OperationNotAllowed	The resource {0} cannot be created from Image {1} until Image has been successfully created.
OperationNotAllowed	Updates to encryptionSettings is not allowed when VM is allocated, Please retry after VM is deallocated
OperationNotAllowed	Addition of a managed disk to a VM with blob based disks is not supported.
OperationNotAllowed	The maximum number of data disks allowed to be attached to a VM of this size is {0}.

Error code	Error message
OperationNotAllowed	Addition of a blob based disk to VM with managed disks is not supported.
OperationNotAllowed	Operation '{0}' is not allowed on Image '{1}' since the Image is marked for deletion. You can only retry the Delete operation (or wait for an ongoing one to complete).
OperationNotAllowed	Operation '{0}' is not allowed on VM '{1}' since the VM is generalized.
OperationNotAllowed	Operation '{0}' is not allowed as Restore point collection '{1}' is marked for deletion.
OperationNotAllowed	Operation '{0}' is not allowed on VM extension '{1}' since it is marked for deletion. You can only retry the Delete operation (or wait for an ongoing one to complete).
OperationNotAllowed	Operation '{0}' is not allowed since the Virtual Machines '{1}' are being provisioned using the Image '{2}'.
OperationNotAllowed	Operation '{0}' is not allowed since the Virtual Machine ScaleSet '{1}' is currently using the Image '{2}'.
OperationNotAllowed	Operation '{0}' is not allowed on VM '{1}' since the VM is marked for deletion. You can only retry the Delete operation (or wait for an ongoing one to complete).
OperationNotAllowed	Operation '{0}' is not allowed on VM '{1}' since the VM is either deallocated or marked to be deallocated.
OperationNotAllowed	Operation '{0}' is not allowed on VM '{1}' since the VM is running. Please power off explicitly in case you shut down the VM from inside the guest operating system.
OperationNotAllowed	Operation '{0}' is not allowed on VM '{1}' since the VM is not deallocated.
OperationNotAllowed	Operation '{0}' is not allowed on VM '{1}' since VM has extension '{2}' in failed state.
OperationNotAllowed	Operation '{0}' is not allowed on VM '{1}' since another operation is in progress.
OperationNotAllowed	The operation '{0}' requires the Virtual Machine '{1}' to be Generalized.
OperationNotAllowed	The operation requires the VM to be running (or set to run).
OperationNotAllowed	Disk with size {0}GB, which is smaller than the size {1}GB of corresponding disk in Image, is not allowed.
OperationNotAllowed	VM Scale Set extensions of handler '{0}' can be added only at the time of VM Scale Set creation.

ERROR CODE	ERROR MESSAGE
OperationNotAllowed	VM Scale Set extensions of handler '{0}' can be deleted only at the time of VM Scale Set deletion.
OperationNotAllowed	VM '{0}' is already using managed disks.
OperationNotAllowed	VM '{0}' belongs to 'Classic' availability set '{1}'. Please update the availability set to use 'Aligned' SKU and then retry the Conversion.
OperationNotAllowed	VM created from Image cannot have blob based disks. All disks have to be managed disks.
OperationNotAllowed	Capture operation cannot be completed because the VM is not generalized.
OperationNotAllowed	Management operations on VM '{0}' are disallowed because VM disks are being converted to managed disks.
OperationNotAllowed	An ongoing operation is changing power state of Virtual Machine {0} to {1}. Please perform operation {2} after some time.
OperationNotAllowed	Unable to add or update the VM. The requested VM size {0} may not be available in the existing allocation unit. Read more on VM resizing strategy at https://aka.ms/azure-resizevm .
OperationNotAllowed	Unable to resize the VM because the requested size {0} is not available in the cluster where the availability set is currently allocated. The available sizes are: {1}. Read more on VM resizing strategy at https://aka.ms/azure-resizevm .
OperationNotAllowed	Unable to resize the VM because the requested size {0} is not available in the cluster where the VM is currently allocated. To resize your VM to {1} please deallocate (this is Stop operation in the Azure portal) and try the resize operation again. Read more on VM resizing strategy at https://aka.ms/azure-resizevm .
OSProvisioningClientError	OS Provisioning failed for VM '{0}' because the guest OS is currently being provisioned.
OSProvisioningClientError	OS provisioning for VM '{0}' failed. Error details: {1} Make sure the image has been properly prepared (generalized). <ul style="list-style-type: none"> • Instructions for Windows: https://azure.microsoft.com/documentation/articles/virtual-machines-windows-upload-image/
OSProvisioningClientError	SSH host key generation failed. Error details: {0}. To resolve this issue verify if Linux agent is set up properly. <ul style="list-style-type: none"> • You can check the instructions at : https://azure.microsoft.com/documentation/articles/virtual-machines-linux-agent-user-guide/

ERROR CODE	ERROR MESSAGE
OSProvisioningClientError	Username specified for the VM is invalid for this Linux distribution. Error details: {0}.
OSProvisioningInternalError	OS Provisioning failed for VM '{0}' due to an internal error.
OSProvisioningTimedOut	OS Provisioning for VM '{0}' did not finish in the allotted time. The VM may still finish provisioning successfully. Please check provisioning state later.
OSProvisioningTimedOut	OS Provisioning for VM '{0}' did not finish in the allotted time. The VM may still finish provisioning successfully. Please check provisioning state later. Also, make sure the image has been properly prepared (generalized). <ul style="list-style-type: none"> Instructions for Windows: https://azure.microsoft.com/documentation/articles/virtual-machines-windows-upload-image/ Instructions for Linux: https://azure.microsoft.com/documentation/articles/virtual-machines-linux-capture-image/
OSProvisioningTimedOut	OS Provisioning for VM '{0}' did not finish in the allotted time. However, the VM guest agent was detected running. This suggests the guest OS has not been properly prepared to be used as a VM image (with CreateOption=FromImage). To resolve this issue, either use the VHD as is with CreateOption=Attach or prepare it properly for use as an image: <ul style="list-style-type: none"> Instructions for Windows: https://azure.microsoft.com/documentation/articles/virtual-machines-windows-upload-image/ Instructions for Linux: https://azure.microsoft.com/documentation/articles/virtual-machines-linux-capture-image/
OverConstrainedAllocationRequest	The required VM size is not currently available in the selected location.
ResourceUpdateBlockedOnPlatformUpdate	Resource cannot be updated at this time due to ongoing platform update. Please try again later.
StorageAccountLimitation	Storage account '{0}' does not support page blobs which are required to create disks.
StorageAccountLimitation	Storage account '{0}' has exceeded its allocated quota.
StorageAccountLocationMismatch	Could not resolve storage account {0}. Please ensure it was created through the Storage Resource Provider in the same location as the compute resource.
StorageAccountNotFound	Storage account {0} not found. Ensure storage account is not deleted and belongs to the same Azure location as the VM.
StorageAccountNotRecognized	Please use a storage account managed by Storage Resource Provider. Use of {0} is not supported.

ERROR CODE	ERROR MESSAGE
StorageAccountOperationInternalError	Internal error occurred while accessing storage account {0}.
StorageAccountSubscriptionMismatch	Storage account {0} doesn't belong to subscription {1}.
StorageAccountTooBusy	Storage account '{0}' is too busy currently. Consider using another account.
StorageAccountTypeNotSupported	Disk {0} uses {1} which is a Blob storage account. Please retry with General purpose storage account.
StorageAccountTypeNotSupported	Storage account {0} is of {1} type. Boot Diagnostics supports {2} storage account types. <ul style="list-style-type: none"> This error occurs if you use the premium storage account for Boot diagnostics. For more information, see How to use boot diagnostics.
SubscriptionNotAuthorizedForImage	The subscription is not authorized.
TargetDiskBlobAlreadyExists	Blob {0} already exists. Please provide a different blob URI to create a new blank data disk '{1}'.
TargetDiskBlobAlreadyExists	Capture operation cannot continue because target image blob {0} already exists and the flag to overwrite VHD blobs is not set. Either delete the blob or set the flag to overwrite VHD blobs and retry.
TargetDiskBlobAlreadyExists	Capture operation cannot continue because target image blob {0} has an active lease on it.
TargetDiskBlobAlreadyExists	Blob {0} already exists. Please provide a different blob URI as target for disk '{1}'.
TooManyVMRedeploymentRequests	Too many redeployment requests have been received for VM '{0}' or the VMs in the same availabilityset with this VM. Please retry later.
VHDSizeInvalid	The specified disk size value of {0} for disk '{1}' with blob {2} is invalid. Disk size must be between {3} and {4}.
VMAgentStatusCommunicationError	VM '{0}' has not reported status for VM agent or extensions. Please verify the VM has a running VM agent, and can establish outbound connections to Azure storage.
VMArtifactRepositoryInternalError	An error occurred while communicating with the artifact repository to retrieve VM artifact details.
VMArtifactRepositoryInternalError	An internal error occurred while retrieving the VM artifact data from the artifact repository.
VMExtensionHandlerNonTransientError	Handler '{0}' has reported failure for VM Extension '{1}' with terminal error code '{2}' and error message: '{3}'
VMExtensionManagementInternalError	Internal error occurred while processing VM extension '{0}'.

ERROR CODE	ERROR MESSAGE
VMExtensionManagementInternalError	Multiple errors occurred while preparing the VM extensions. See VM extension instance view for details.
VMExtensionProvisioningError	VM has reported a failure when processing extension '{0}'. Error message: "{1}".
VMExtensionProvisioningError	Multiple VM extensions failed to be provisioned on the VM. Please see the VM extension instance view for details.
VMExtensionProvisioningTimeout	Provisioning of VM extension '{0}' has timed out. Extension installation may be taking too long, or extension status could not be obtained.
VMMarketplaceInvalidInput	Creating a virtual machine from a non Marketplace image does not need Plan information, please remove the Plan information in the request. OS disk name is {0}.
VMMarketplaceInvalidInput	The purchase information does not match. Unable to deploy from the Marketplace image. OS disk name is {0}.
VMMarketplaceInvalidInput	Creating a virtual machine from Marketplace image requires Plan information in the request. OS disk name is {0}.
VMNotFound	The VM '{0}' cannot be found.
VMRedeploymentFailed	VM '{0}' redeployment failed due to an internal error. Please retry later.
VMRedeploymentTimedOut	Redeployment of VM '{0}' didn't finish in the allotted time. It might finish successfully in sometime. Else, you can retry the request.
VMStartTimedOut	VM '{0}' did not start in the allotted time. The VM may still start successfully. Please check the power state later.

Next steps

If you need more help, you can contact the Azure experts on [the MSDN Azure and Stack Overflow forums](#).

Alternatively, you can file an Azure support incident. Go to the [Azure support site](#) and select **Get Support**.

Troubleshoot a Linux VM by attaching the OS disk to a recovery VM with the Azure CLI 2.0

4/9/2018 • 7 min to read • [Edit Online](#)

If your Linux virtual machine (VM) encounters a boot or disk error, you may need to perform troubleshooting steps on the virtual hard disk itself. A common example would be an invalid entry in `/etc/fstab` that prevents the VM from being able to boot successfully. This article details how to use the Azure CLI 2.0 to connect your virtual hard disk to another Linux VM to fix any errors, then re-create your original VM. You can also perform these steps with the [Azure CLI 1.0](#).

Recovery process overview

The troubleshooting process is as follows:

1. Delete the VM encountering issues, keeping the virtual hard disks.
2. Attach and mount the virtual hard disk to another Linux VM for troubleshooting purposes.
3. Connect to the troubleshooting VM. Edit files or run any tools to fix issues on the original virtual hard disk.
4. Unmount and detach the virtual hard disk from the troubleshooting VM.
5. Create a VM using the original virtual hard disk.

To perform these troubleshooting steps, you need the latest [Azure CLI 2.0](#) installed and logged in to an Azure account using `az login`.

In the following examples, replace parameter names with your own values. Example parameter names include `myResourceGroup`, `mystorageaccount`, and `myVM`.

Determine boot issues

Examine the serial output to determine why your VM is not able to boot correctly. A common example is an invalid entry in `/etc/fstab`, or the underlying virtual hard disk being deleted or moved.

Get the boot logs with `az vm boot-diagnostics get-boot-log`. The following example gets the serial output from the VM named `myVM` in the resource group named `myResourceGroup`:

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

Review the serial output to determine why the VM is failing to boot. If the serial output isn't providing any indication, you may need to review log files in `/var/log` once you have the virtual hard disk connected to a troubleshooting VM.

View existing virtual hard disk details

Before you can attach your virtual hard disk (VHD) to another VM, you need to identify the URI of the OS disk.

View information about your VM with `az vm show`. Use the `--query` flag to extract the URI to the OS disk. The following example gets disk information for the VM named `myVM` in the resource group named `myResourceGroup`:

```
az vm show --resource-group myResourceGroup --name myVM \
--query [storageProfile.osDisk.vhd.uri] --output tsv
```

The URI is similar to <https://mystorageaccount.blob.core.windows.net/vhds/myVM.vhd>.

Delete existing VM

Virtual hard disks and VMs are two distinct resources in Azure. A virtual hard disk is where the operating system itself, applications, and configurations are stored. The VM itself is just metadata that defines the size or location, and references resources such as a virtual hard disk or virtual network interface card (NIC). Each virtual hard disk has a lease assigned when attached to a VM. Although data disks can be attached and detached even while the VM is running, the OS disk cannot be detached unless the VM resource is deleted. The lease continues to associate the OS disk with a VM even when that VM is in a stopped and deallocated state.

The first step to recover your VM is to delete the VM resource itself. Deleting the VM leaves the virtual hard disks in your storage account. After the VM is deleted, you attach the virtual hard disk to another VM to troubleshoot and resolve the errors.

Delete the VM with [az vm delete](#). The following example deletes the VM named `myVM` from the resource group named `myResourceGroup`:

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

Wait until the VM has finished deleting before you attach the virtual hard disk to another VM. The lease on the virtual hard disk that associates it with the VM needs to be released before you can attach the virtual hard disk to another VM.

Attach existing virtual hard disk to another VM

For the next few steps, you use another VM for troubleshooting purposes. You attach the existing virtual hard disk to this troubleshooting VM to browse and edit the disk's content. This process allows you to correct any configuration errors or review additional application or system log files, for example. Choose or create another VM to use for troubleshooting purposes.

Attach the existing virtual hard disk with [az vm unmanaged-disk attach](#). When you attach the existing virtual hard disk, specify the URI to the disk obtained in the preceding [az vm show](#) command. The following example attaches an existing virtual hard disk to the troubleshooting VM named `myVMRecovery` in the resource group named `myResourceGroup`:

```
az vm unmanaged-disk attach --resource-group myResourceGroup --vm-name myVMRecovery \
--vhd-uri https://mystorageaccount.blob.core.windows.net/vhds/myVM.vhd
```

Mount the attached data disk

NOTE

The following examples detail the steps required on an Ubuntu VM. If you are using a different Linux distro, such as Red Hat Enterprise Linux or SUSE, the log file locations and `mount` commands may be a little different. Refer to the documentation for your specific distro for the appropriate changes in commands.

1. SSH to your troubleshooting VM using the appropriate credentials. If this disk is the first data disk attached

to your troubleshooting VM, the disk is likely connected to `/dev/sdc`. Use `dmseg` to view attached disks:

```
dmseg | 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
```

In the preceding example, the OS disk is at `/dev/sda` and the temporary disk provided for each VM is at `/dev/sdb`. If you had multiple data disks, they should be at `/dev/sdd`, `/dev/sde`, and so on.

2. Create a directory to mount your existing virtual hard disk. The following example creates a directory named `troubleshootingdisk`:

```
sudo mkdir /mnt/troubleshootingdisk
```

3. If you have multiple partitions on your existing virtual hard disk, mount the required partition. The following example mounts the first primary partition at `/dev/sdc1`:

```
sudo mount /dev/sdc1 /mnt/troubleshootingdisk
```

NOTE

Best practice is to mount data disks on VMs in Azure using the universally unique identifier (UUID) of the virtual hard disk. For this short troubleshooting scenario, mounting the virtual hard disk using the UUID is not necessary. However, under normal use, editing `/etc/fstab` to mount virtual hard disks using device name rather than UUID may cause the VM to fail to boot.

Fix issues on original virtual hard disk

With the existing virtual hard disk mounted, you can now perform any maintenance and troubleshooting steps as needed. Once you have addressed the issues, continue with the following steps.

Unmount and detach original virtual hard disk

Once your errors are resolved, you unmount and detach the existing virtual hard disk from your troubleshooting VM. You cannot use your virtual hard disk with any other VM until the lease attaching the virtual hard disk to the troubleshooting VM is released.

1. From the SSH session to your troubleshooting VM, unmount the existing virtual hard disk. Change out of the parent directory for your mount point first:

```
cd /
```

Now unmount the existing virtual hard disk. The following example unmounts the device at `/dev/sdc1`:

```
sudo umount /dev/sdc1
```

2. Now detach the virtual hard disk from the VM. Exit the SSH session to your troubleshooting VM. List the attached data disks to your troubleshooting VM with [az vm unmanaged-disk list](#). The following example lists the data disks attached to the VM named `myVMRecovery` in the resource group named `myResourceGroup`:

```
azure vm unmanaged-disk list --resource-group myResourceGroup --vm-name myVMRecovery \
--query '[].{Disk:vhd.uri}' --output table
```

Note the name for your existing virtual hard disk. For example, the name of a disk with the URI of <https://mystorageaccount.blob.core.windows.net/vhds/myVM.vhd> is **myVHD**.

Detach the data disk from your VM [az vm unmanaged-disk detach](#). The following example detaches the disk named `myVHD` from the VM named `myVMRecovery` in the `myResourceGroup` resource group:

```
az vm unmanaged-disk detach --resource-group myResourceGroup --vm-name myVMRecovery \
--name myVHD
```

Create VM from original hard disk

To create a VM from your original virtual hard disk, use [this Azure Resource Manager template](#). The actual JSON template is at the following link:

- <https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vm-specialized-vhd/azuredeploy.json>

The template deploys a VM using the VHD URI from the earlier command. Deploy the template with [az group deployment create](#). Provide the URI to your original VHD and then specify the OS type, VM size, and VM name as follows:

```
az group deployment create --resource-group myResourceGroup --name myDeployment \
--parameters '{"osDiskVhdUri": {"value": "https://mystorageaccount.blob.core.windows.net/vhds/myVM.vhd"}, \
"osType": {"value": "Linux"}, \
"vmSize": {"value": "Standard_DS1_v2"}, \
"vmName": {"value": "myDeployedVM"}' \
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-vm-specialized-vhd/azuredeploy.json
```

Re-enable boot diagnostics

When you create your VM from the existing virtual hard disk, boot diagnostics may not automatically be enabled. Enable boot diagnostics with [az vm boot-diagnostics enable](#). The following example enables the diagnostic extension on the VM named `myDeployedVM` in the resource group named `myResourceGroup`:

```
az vm boot-diagnostics enable --resource-group myResourceGroup --name myDeployedVM
```

Next steps

If you are having issues connecting to your VM, see [Troubleshoot SSH connections to an Azure VM](#). For issues with accessing applications running on your VM, see [Troubleshoot application connectivity issues on a Linux VM](#).

Troubleshoot a Linux VM by attaching the OS disk to a recovery VM using the Azure portal

4/9/2018 • 7 min to read • [Edit Online](#)

If your Linux virtual machine (VM) encounters a boot or disk error, you may need to perform troubleshooting steps on the virtual hard disk itself. A common example would be an invalid entry in `/etc/fstab` that prevents the VM from being able to boot successfully. This article details how to use the Azure portal to connect your virtual hard disk to another Linux VM to fix any errors, then re-create your original VM.

Recovery process overview

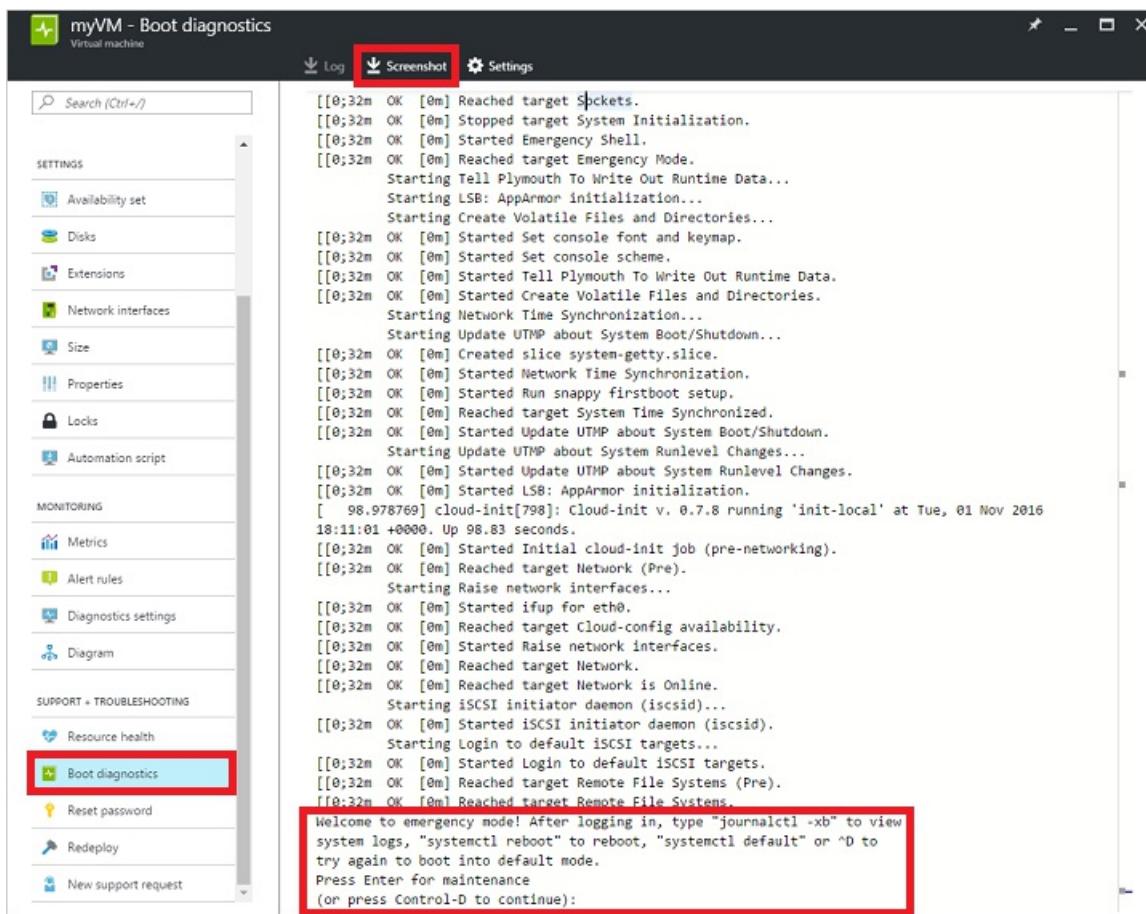
The troubleshooting process is as follows:

1. Delete the VM encountering issues, keeping the virtual hard disks.
2. Attach and mount the virtual hard disk to another Linux VM for troubleshooting purposes.
3. Connect to the troubleshooting VM. Edit files or run any tools to fix issues on the original virtual hard disk.
4. Unmount and detach the virtual hard disk from the troubleshooting VM.
5. Create a VM using the original virtual hard disk.

Determine boot issues

Examine the boot diagnostics and VM screenshot to determine why your VM is not able to boot correctly. A common example would be an invalid entry in `/etc/fstab`, or an underlying virtual hard disk being deleted or moved.

Select your VM in the portal and then scroll down to the **Support + Troubleshooting** section. Click **Boot diagnostics** to view the console messages streamed from your VM. Review the console logs to see if you can determine why the VM is encountering an issue. The following example shows a VM stuck in maintenance mode that requires manual interaction:

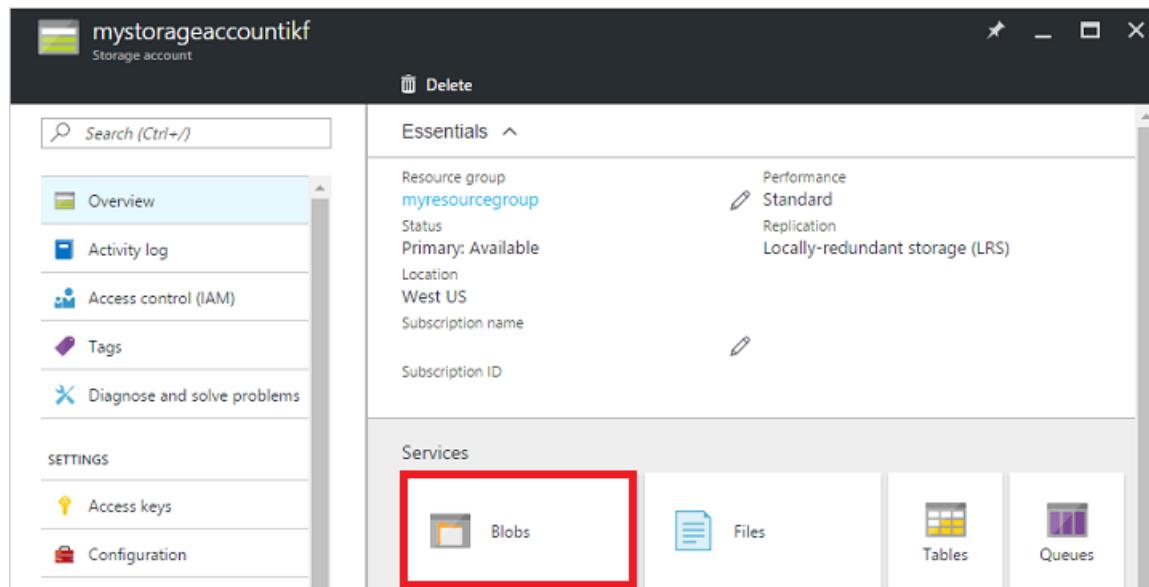


You can also click **Screenshot** across the top of the boot diagnostics log to download a capture of the VM screenshot.

View existing virtual hard disk details

Before you can attach your virtual hard disk to another VM, you need to identify the name of the virtual hard disk (VHD).

Select your resource group from the portal, then select your storage account. Click **Blobs**, as in the following example:



Typically you have a container named **vhd**s that stores your virtual hard disks. Select the container to view a list of virtual hard disks. Note the name of your VHD (the prefix is usually the name of your VM):

Storage account: mystorageaccountikf
Status: Primary: Available
Location: West US
Subscription name:
Subscription ID:

NAME	URL	LAST MODIFIED
bootdiagnostics-myvm-b8a...	https://mystorageaccountik...	11/1/2016, 10:38:45 AM
vhds	https://mystorageaccountik...	11/1/2016, 10:38:45 AM

Location: vhds

NAME	MODIFIED	BLOB TYPE	SIZE
myDisk.vhd	11/1/2016, 11:34:38 AM	Page blob	1.1 TB
myVM2016101103813.vhd	11/1/2016, 11:34:38 AM	Page blob	31.46 GB

Select your existing virtual hard disk from the list and copy the URL for use in the following steps:

vhds Container Refresh Delete container Properties Access policy

Location: vhds

NAME	MODIFIED	BLOB TYPE	SIZE
myDisk.vhd	11/1/2016, 11:34:38 AM	Page blob	1.1 TB
myVM2016101103813.vhd	11/1/2016, 12:04:50 PM	Page blob	31.46 GB
myVMRecovery.a49961a9-41fe-...	11/1/2016, 2:31:25 PM	Block blob	246 B
myVMRecovery201610111441.vhd	11/1/2016, 2:27:29 PM	Page blob	31.46 GB

Blob properties

myVM2016101103813.vhd

Download Delete

NAME
myVM2016101103813.vhd

URL: https://mystorageaccountikf.blob.core.windows.net/vhds/myVM2016101103813.vhd

LAST MODIFIED: 11/1/2016, 12:04:50 PM

TYPE: Page blob

SIZE: 31.46 GB

Delete existing VM

Virtual hard disks and VMs are two distinct resources in Azure. A virtual hard disk is where the operating system itself, applications, and configurations are stored. The VM itself is just metadata that defines the size or location, and references resources such as a virtual hard disk or virtual network interface card (NIC). Each virtual hard disk has a lease assigned when attached to a VM. Although data disks can be attached and detached even while the VM is running, the OS disk cannot be detached unless the VM resource is deleted. The lease continues to associate the OS disk with a VM even when that VM is in a stopped and deallocated state.

The first step to recover your VM is to delete the VM resource itself. Deleting the VM leaves the virtual hard disks in your storage account. After the VM is deleted, you attach the virtual hard disk to another VM to troubleshoot and resolve the errors.

Select your VM in the portal, then click **Delete**:

The screenshot shows the Azure portal interface for a virtual machine named 'myVM'. The top navigation bar includes 'Connect', 'Start', 'Restart', 'Stop', and 'Delete' buttons, with 'Delete' being highlighted by a red box. The left sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', and 'Diagnose and solve problems'. The main content area titled 'Essentials' displays resource group information ('myResourceGroup'), VM details ('Computer name: myVM, Operating system: Linux, Size: Standard D1 v2 (1 core, 3.5 GB memory)'), and network settings ('Public IP address/DNS name label: 104.40.13.0/<none>, Virtual network/subnet: myVnet/mySubnet').

Wait until the VM has finished deleting before you attach the virtual hard disk to another VM. The lease on the virtual hard disk that associates it with the VM needs to be released before you can attach the virtual hard disk to another VM.

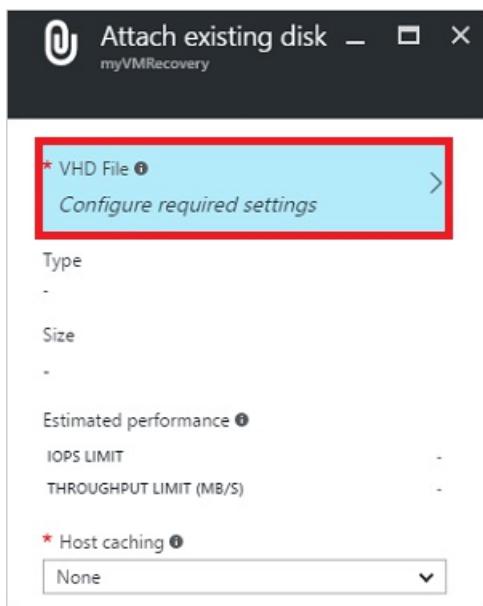
Attach existing virtual hard disk to another VM

For the next few steps, you use another VM for troubleshooting purposes. You attach the existing virtual hard disk to this troubleshooting VM to be able to browse and edit the disk's content. This process allows you to correct any configuration errors or review additional application or system log files, for example. Choose or create another VM to use for troubleshooting purposes.

1. Select your resource group from the portal, then select your troubleshooting VM. Select **Disks** and then click **Attach existing**:

The screenshot shows the Azure portal interface for a virtual machine named 'myVMRecovery'. The top navigation bar includes 'Attach new' and 'Attach existing' buttons, with 'Attach existing' being highlighted by a red box. The left sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', and 'Diagnose and solve problems'. A 'SETTINGS' section contains 'Availability set' and 'Disks', with 'Disks' being highlighted by a red box. The main content area displays disk management details, showing an OS DISK named 'myVMRecovery' with a size of 23.2 GB and encryption status 'Not enabled'. Below this, a DATA DISKS section shows 'No results.'

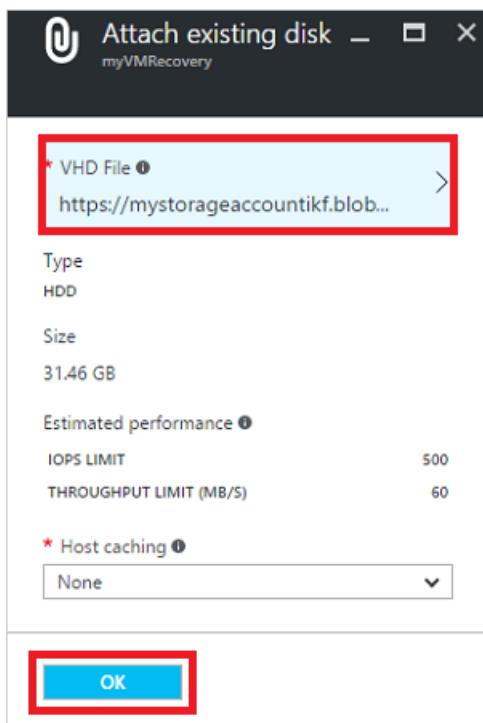
2. To select your existing virtual hard disk, click **VHD File**:



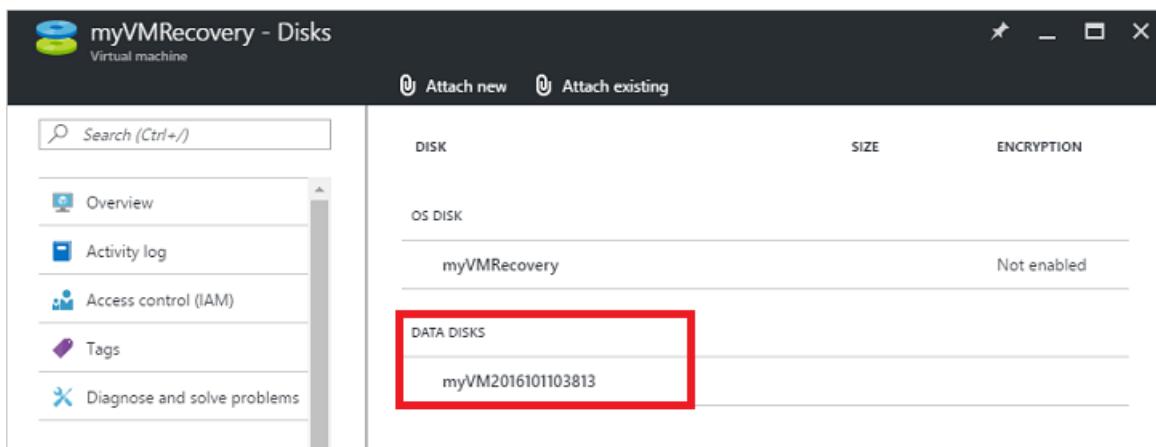
3. Select your storage account and container, then click your existing VHD. Click the **Select** button to confirm your choice:

NAME	SIZE
myDisk.vhd	1.1 TB
myVM2016101103813.vhd	31.46 GB
myVMRecovery.a49961a9-41fe-4fb6-9902-3dd30eb62d54.status	246 B
myVMRecovery201610111441.vhd	31.46 GB

4. With your VHD now selected, click **OK** to attach the existing virtual hard disk:



5. After a few seconds, the **Disks** pane for your VM lists your existing virtual hard disk connected as a data disk:



Mount the attached data disk

NOTE

The following examples detail the steps required on an Ubuntu VM. If you are using a different Linux distro, such as Red Hat Enterprise Linux or SUSE, the log file locations and `mount` commands may be a little different. Refer to the documentation for your specific distro for the appropriate changes in commands.

1. SSH to your troubleshooting VM using the appropriate credentials. If this disk is the first data disk attached to your troubleshooting VM, it is likely connected to `/dev/sdc`. Use `dmseg` to list attached disks:

```
dmseg | 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
```

In the preceding example, the OS disk is at `/dev/sda` and the temporary disk provided for each VM is at `/dev/sdb`. If you had multiple data disks, they should be at `/dev/sdd`, `/dev/sde`, and so on.

2. Create a directory to mount your existing virtual hard disk. The following example creates a directory named `troubleshootingdisk`:

```
sudo mkdir /mnt/troubleshootingdisk
```

3. If you have multiple partitions on your existing virtual hard disk, mount the required partition. The following example mounts the first primary partition at `/dev/sdc1`:

```
sudo mount /dev/sdc1 /mnt/troubleshootingdisk
```

NOTE

Best practice is to mount data disks on VMs in Azure using the universally unique identifier (UUID) of the virtual hard disk. For this short troubleshooting scenario, mounting the virtual hard disk using the UUID is not necessary. However, under normal use, editing `/etc/fstab` to mount virtual hard disks using device name rather than UUID may cause the VM to fail to boot.

Fix issues on original virtual hard disk

With the existing virtual hard disk mounted, you can now perform any maintenance and troubleshooting steps as needed. Once you have addressed the issues, continue with the following steps.

Unmount and detach original virtual hard disk

Once your errors are resolved, detach the existing virtual hard disk from your troubleshooting VM. You cannot use your virtual hard disk with any other VM until the lease attaching the virtual hard disk to the troubleshooting VM is released.

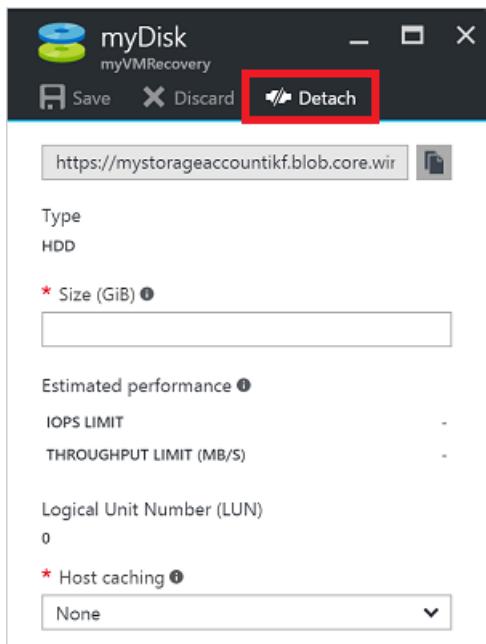
1. From the SSH session to your troubleshooting VM, unmount the existing virtual hard disk. Change out of the parent directory for your mount point first:

```
cd /
```

Now unmount the existing virtual hard disk. The following example unmounts the device at `/dev/sdc1`:

```
sudo umount /dev/sdc1
```

2. Now detach the virtual hard disk from the VM. Select your VM in the portal and click **Disk**. Select your existing virtual hard disk and then click **Detach**:



Wait until the VM has successfully detached the data disk before continuing.

Create VM from original hard disk

To create a VM from your original virtual hard disk, use [this Azure Resource Manager template](#). The template

deploys a VM into an existing virtual network, using the VHD URL from the earlier command. Click the **Deploy to Azure** button as follows:

The screenshot shows a GitHub repository interface. At the top, it displays a commit by user 'marcvanek' with the message 'committed with kvaes Location fix'. To the right, it shows 'Latest commit 0f30cfc on Apr 17'. Below this is a list of files with their descriptions and last modified times:

File	Description	Last Modified
README.md	Add "Visualize" buttons to all template README.md files	10 months ago
azuredeploy.json	Location fix	7 months ago
azuredeploy.parameters.json	Location fix	7 months ago
metadata.json	Update metadata.json	a year ago

Below the file list, there is a large text area containing the following text:

Create a specialized virtual machine in an existing virtual network

At the bottom of this area are two buttons:

- Deploy to Azure** (highlighted with a red box)
- Visualize**

The template is loaded into the Azure portal for deployment. Enter the names for your new VM and existing Azure resources, and paste the URL to your existing virtual hard disk. To begin the deployment, click **Purchase**:

Create a VM from a custom VHD and connect it to an existing VNET

Azure quickstart template

TEMPLATE



BASICS

* Subscription	Visual Studio Ultimate with MSDN
* Resource group	<input type="radio"/> Create new <input checked="" type="radio"/> Use existing myResourceGroup
* Location	West US

SETTINGS

* Vm Name	myVMRecovered
* Os Type	Linux
* Os Disk Vhd Uri	https://mystorageaccountikf.blob.core.windows.net/vhds/myVM2016101103813...
* Vm Size	Standard_DS1_v2
* Existing Virtual Network Name	myVnet
* Existing Virtual Network Resource Group	myResourceGroup
* Subnet Name	mySubnet
* Dns Name For Public IP	myvmrecovered

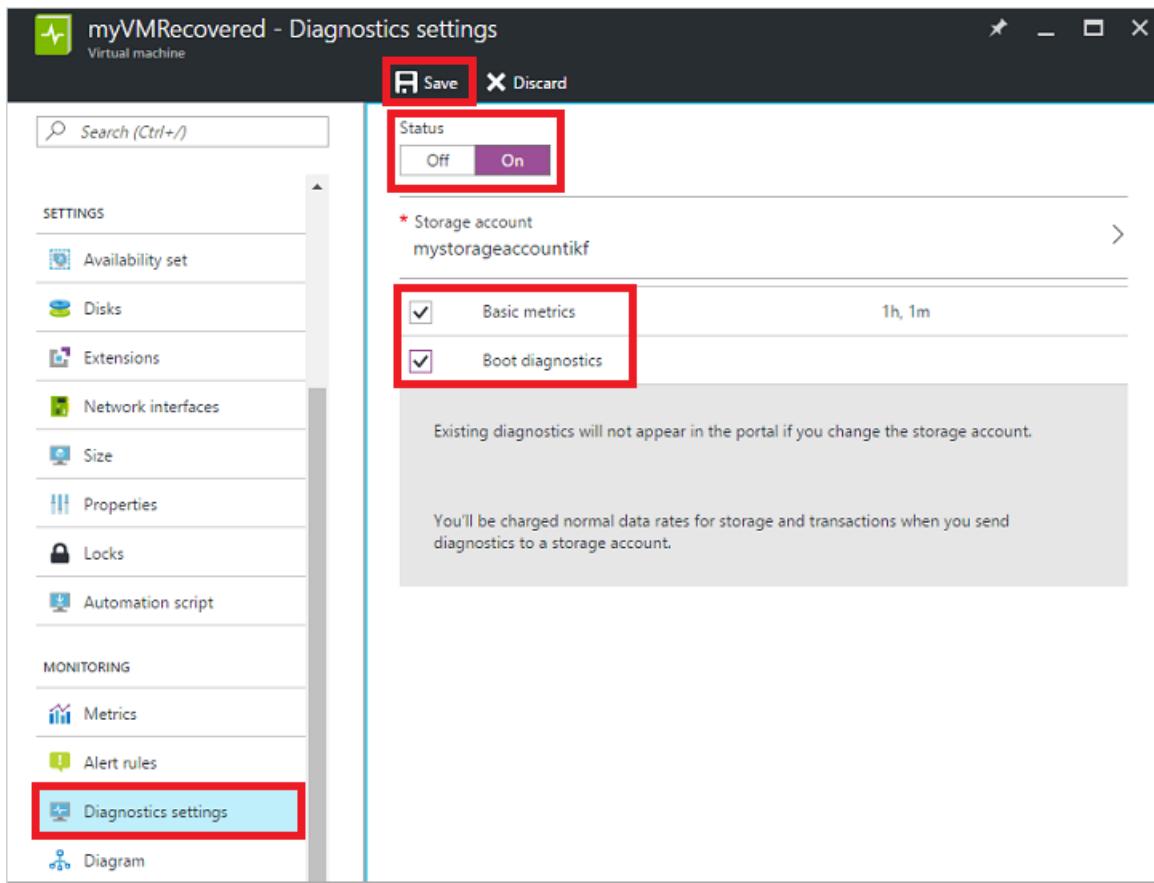
TERMS AND CONDITIONS

Pin to dashboard

Purchase

Re-enable boot diagnostics

When you create your VM from the existing virtual hard disk, boot diagnostics may not automatically be enabled. To check the status of boot diagnostics and turn on if needed, select your VM in the portal. Under **Monitoring**, click **Diagnostics settings**. Ensure the status is **On**, and the check mark next to **Boot diagnostics** is selected. If you make any changes, click **Save**:



Next steps

If you are having issues connecting to your VM, see [Troubleshoot SSH connections to an Azure VM](#). For issues with accessing applications running on your VM, see [Troubleshoot application connectivity issues on a Linux VM](#).

For more information about using Resource Manager, see [Azure Resource Manager overview](#).

Understand the structure and syntax of Azure Resource Manager templates

5/7/2018 • 6 min to read • [Edit Online](#)

This article describes the structure of an Azure Resource Manager template. It presents the different sections of a template and the properties that are available in those sections. The template consists of JSON and expressions that you can use to construct values for your deployment. For a step-by-step tutorial on creating a template, see [Create your first Azure Resource Manager template](#).

Template format

In its simplest structure, a template contains the following elements:

```
{  
  "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",  
  "contentVersion": "",  
  "parameters": { },  
  "variables": { },  
  "functions": { },  
  "resources": [ ],  
  "outputs": { }  
}
```

ELEMENT NAME	REQUIRED	DESCRIPTION
\$schema	Yes	Location of the JSON schema file that describes the version of the template language. Use the URL shown in the preceding example.
contentVersion	Yes	Version of the template (such as 1.0.0.0). You can provide any value for this element. When deploying resources using the template, this value can be used to make sure that the right template is being used.
parameters	No	Values that are provided when deployment is executed to customize resource deployment.
variables	No	Values that are used as JSON fragments in the template to simplify template language expressions.
functions	No	User-defined functions that are available within the template.
resources	Yes	Resource types that are deployed or updated in a resource group.

ELEMENT NAME	REQUIRED	DESCRIPTION
outputs	No	Values that are returned after deployment.

Each element contains properties you can set. The following example contains the full syntax for a template:

```
{
  "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "",
  "parameters": {
    "<parameter-name>": {
      "type" : "<type-of-parameter-value>",
      "defaultValue": "<default-value-of-parameter>",
      "allowedValues": [ "<array-of-allowed-values>" ],
      "minValue": <minimum-value-for-int>,
      "maxValue": <maximum-value-for-int>,
      "minLength": <minimum-length-for-string-or-array>,
      "maxLength": <maximum-length-for-string-or-array-parameters>,
      "metadata": {
        "description": "<description-of-the parameter>"
      }
    }
  },
  "variables": {
    "<variable-name>": "<variable-value>",
    "<variable-object-name>": {
      <variable-complex-type-value>
    },
    "<variable-object-name>": {
      "copy": [
        {
          "name": "<name-of-array-property>",
          "count": <number-of-iterations>,
          "input": {
            <properties-to-repeat>
          }
        }
      ]
    },
    "copy": [
      {
        "name": "<variable-array-name>",
        "count": <number-of-iterations>,
        "input": {
          <properties-to-repeat>
        }
      }
    ]
  },
  "functions": [
    {
      "namespace": "<namespace-for-your-function>",
      "members": {
        "<function-name>": {
          "parameters": [
            {
              "name": "<parameter-name>",
              "type": "<type-of-parameter-value>"
            }
          ],
          "output": {
            "type": "<type-of-output-value>",
            "value": "<function-expression>"
          }
        }
      }
    }
  ]
}
```

```

        }
    ],
    "resources": [
        {
            "condition": "<boolean-value-whether-to-deploy>",
            "apiVersion": "<api-version-of-resource>",
            "type": "<resource-provider-namespace/resource-type-name>",
            "name": "<name-of-the-resource>",
            "location": "<location-of-resource>",
            "tags": {
                "<tag-name1>": "<tag-value1>",
                "<tag-name2>": "<tag-value2>"
            },
            "comments": "<your-reference-notes>",
            "copy": {
                "name": "<name-of-copy-loop>",
                "count": "<number-of-iterations>",
                "mode": "<serial-or-parallel>",
                "batchSize": "<number-to-deploy-serially>"
            },
            "dependsOn": [
                "<array-of-related-resource-names>"
            ],
            "properties": {
                "<settings-for-the-resource>",
                "copy": [
                    {
                        "name": ,
                        "count": ,
                        "input": {}
                    }
                ]
            },
            "resources": [
                "<array-of-child-resources>"
            ]
        }
    ],
    "outputs": {
        "<outputName>": {
            "type": "<type-of-output-value>",
            "value": "<output-value-expression>"
        }
    }
}

```

This article describes the sections of the template in greater detail.

Syntax

The basic syntax of the template is JSON. However, expressions and functions extend the JSON values available within the template. Expressions are written within JSON string literals whose first and last characters are the brackets: [and], respectively. The value of the expression is evaluated when the template is deployed. While written as a string literal, the result of evaluating the expression can be of a different JSON type, such as an array or integer, depending on the actual expression. To have a literal string start with a bracket [, but not have it interpreted as an expression, add an extra bracket to start the string with [[.

Typically, you use expressions with functions to perform operations for configuring the deployment. Just like in JavaScript, function calls are formatted as `functionName(arg1,arg2,arg3)`. You reference properties by using the dot and [index] operators.

The following example shows how to use several functions when constructing a value:

```
"variables": {
    "storageName": "[concat(toLower(parameters('storageNamePrefix')), uniqueString(resourceGroup().id))]"
}
```

For the full list of template functions, see [Azure Resource Manager template functions](#).

Parameters

In the parameters section of the template, you specify which values you can input when deploying the resources. These parameter values enable you to customize the deployment by providing values that are tailored for a particular environment (such as dev, test, and production). You do not have to provide parameters in your template, but without parameters your template would always deploy the same resources with the same names, locations, and properties.

The following example shows a simple parameter definition:

```
"parameters": {
    "siteNamePrefix": {
        "type": "string",
        "metadata": {
            "description": "The name prefix of the web app that you wish to create."
        }
    },
},
```

For information about defining parameters, see [Parameters section of Azure Resource Manager templates](#).

Variables

In the variables section, you construct values that can be used throughout your template. You do not need to define variables, but they often simplify your template by reducing complex expressions.

The following example shows a simple variable definition:

```
"variables": {
    "webSiteName": "[concat(parameters('siteNamePrefix'), uniqueString(resourceGroup().id))]",
},
```

For information about defining variables, see [Variables section of Azure Resource Manager templates](#).

Functions

Within your template, you can create your own functions. These functions are available for use in your template. Typically, you define complicated expression that you do not want to repeat throughout your template. You create the user-defined functions from expressions and [functions](#) that are supported in templates.

When defining a user function, there are some restrictions:

- The function can't access variables.
- The function can't use the [reference function](#).
- Parameters for the function can't have default values.

Your functions require a namespace value to avoid naming conflicts with template functions. The following example shows a function that returns a storage account name:

```

"functions": [
  {
    "namespace": "contoso",
    "members": {
      "uniqueName": {
        "parameters": [
          {
            "name": "namePrefix",
            "type": "string"
          }
        ],
        "output": {
          "type": "string",
          "value": "[concat(toLower(parameters('namePrefix')), uniqueString(resourceGroup().id))]"
        }
      }
    }
  },
],

```

You call the function with:

```

"resources": [
  {
    "name": "[contoso.uniqueName(parameters('storageNamePrefix'))]",
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2016-01-01",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "location": "South Central US",
    "tags": {},
    "properties": {}
  }
]

```

Resources

In the resources section, you define the resources that are deployed or updated. This section can get complicated because you must understand the types you are deploying to provide the right values.

```

"resources": [
  {
    "apiVersion": "2016-08-01",
    "name": "[variables('webSiteName')]",
    "type": "Microsoft.Web/sites",
    "location": "[resourceGroup().location]",
    "properties": {
      "serverFarmId": "/subscriptions/<subscription-id>/resourcegroups/<resource-group-name>/providers/Microsoft.Web/serverFarms/<plan-name>"
    }
  }
],

```

For more information, see [Resources section of Azure Resource Manager templates](#).

Outputs

In the Outputs section, you specify values that are returned from deployment. For example, you could return the

URI to access a deployed resource.

```
"outputs": {  
    "newHostName": {  
        "type": "string",  
        "value": "[reference(variables('webSiteName')).defaultHostName]"  
    }  
}
```

For more information, see [Outputs section of Azure Resource Manager templates](#).

Template limits

Limit the size of your template to 1 MB, and each parameter file to 64 KB. The 1-MB limit applies to the final state of the template after it has been expanded with iterative resource definitions, and values for variables and parameters.

You are also limited to:

- 256 parameters
- 256 variables
- 800 resources (including copy count)
- 64 output values
- 24,576 characters in a template expression

You can exceed some template limits by using a nested template. For more information, see [Using linked templates when deploying Azure resources](#). To reduce the number of parameters, variables, or outputs, you can combine several values into an object. For more information, see [Objects as parameters](#).

Next steps

- To view complete templates for many different types of solutions, see the [Azure Quickstart Templates](#).
- For details about the functions you can use from within a template, see [Azure Resource Manager Template Functions](#).
- To combine multiple templates during deployment, see [Using linked templates with Azure Resource Manager](#).
- You may need to use resources that exist within a different resource group. This scenario is common when working with storage accounts or virtual networks that are shared across multiple resource groups. For more information, see the [resourceId function](#).

Frequently asked question about Linux Virtual Machines

3/23/2018 • 3 min 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 log 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

What are the password requirements when creating a VM?

Passwords must be 6 - 72 characters in length and 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!