Inside Microsoft Azure IaaS

CONCEPTS, OPTIMIZATION AND AUTOMATION

An installment in the
Inside the Microsoft Cloud series

Ryan Irujo, Janaka Rangama, Pete Zerger & Anders Bengtsson

Use Rights

Version: 1.0

Publication Date: March 31st, 2017

Change History

Version	*Date*	*Description*
V1.0	March 31st, 2017	Initial release.

About the Authors

Ryan Irujo
Principal Technical Consultant of Lumagate in Norway with over 15 years' experience in the I.T. industry for medium and large enterprise businesses in several different areas such as finance, healthcare, retail, and e-business solutions and is currently focusing on Azure Cloud Solutions, IoT, Automation, and cross-platform technologies. Ryan has previously presented at IT/Dev Connections and is the co-author of "Automating Deployment with Azure & Chef". You can reach him on Twitter at @reirujo.

Janaka Rangama, MVP
Janaka Rangama is a Microsoft Most Valuable Professional (MVP), Microsoft Certified Trainer and Azure Advisor and Open Source geek hailing from "The Pearl of the Indian Ocean", Sri Lanka. Janaka is a well-known technical speaker in many international conferences He holds over a dozen Microsoft Certifications, and is also a Red Hat Certified Professional. Janaka is a Principal Consultant at Empired Ltd in Melbourne, Australia. You can reach him via his blog http://tekronin.net or on Twitter at @JanakaRangama.

Pete Zerger, CISSP, MVP
Pete is an author, speaker, architect and 11-time Microsoft MVP, focusing on cloud architecture, automation, and security, as well as DevOps. Pete has presented at popular technical conferences around the world including Microsoft Ignite, TechEd and System Center Universe. He has authored many technical white papers and is a co-author of several books on management and cloud topics, including the popular "Inside the Microsoft Operations Management Suite". Pete is Managing Partner of Lumagate, Inc. in North America,

Anders Bengtsson
Anders is a Principal PFE at Microsoft focusing on cloud and data center solutions. He is a frequent speaker at Microsoft conferences and is a contributing author of several books, including Orchestrator Unleashed, Service Manager Unleashed and Inside the Microsoft Operations Management Suite. Anders is the founder of Contoso.se, a popular blog with focus on cloud and data center management.

Table of Contents

Chapter 1: Intro **1**

Chapter 2: Azure PowerShell **6**

Chapter 3: Azure Virtual Networking **21**

Chapter 4: Azure Storage **57**

Chapter 5: Connect Azure to Your Datacenter **79**

Chapter 6: Azure Virtual Machines **95**

Chapter 7: Migration to Microsoft Azure **185**

Chapter 8: Backup & Disaster Recovery **207**

Chapter 1: Introduction

By the title of the book, it is clear the focus of this book is on Microsoft Azure Infrastructure-as-a-Service (IaaS), meaning the bulk of this book will cover Azure Virtual Machines (VMs) in depth. However, Microsoft Azure IaaS is much more than just VMs and in fact, offers many other features IT Pros should know about, as they will enhance your ability to simplify service delivery and streamline IT operations while reducing your overall infrastructure costs.

However, true to the title, this book will be focused on concepts, from basic to advanced, surrounding Azure IaaS.

> **How to get the most from "Inside Microsoft Azure IaaS".** To get the most from the book, read the chapters, but also try the examples for yourself! You can download the scripts and samples used throughout the book and practice as you follow along. By the end of the book, you will have the knowledge to leverage the best of what Azure IaaS has to offer!

Inside Microsoft Azure IaaS includes coverage of fundamental and advanced topics, delivered through clear explanations together with hands-on examples demonstrating automated provisioning and management of Azure IaaS VMs. Topics include:

- **Azure PowerShell**. PowerShell is your go-to management interface for automation and bulk administration. Microsoft has put a lot of work into the last couple of releases of Azure PowerShell to deliver a powerful and consistent administration experience, whether you are working with VMs, networks, storage or files shares. In chapter 2, we will explore Azure PowerShell in-depth.
- **Azure Virtual Networking**. Azure Virtual Network capabilities enable Azure resources connectivity to the Internet and on-premises resources, routing, traffic filtering, and network segmentation and isolation. In chapter 3, we will dive into Azure Virtual Network features in depth.
- **Azure Storage**. Azure storage is a massively scalable, highly available platform supporting a variety of IaaS-related services, including Azure IaaS VMs, SMB file shares, and storage for data backups and VM replicas for disaster recovery. In chapter 4, we will explore a variety of Azure Storage capabilities, as well as how to deploy them with the Azure PowerShell.

- **Connecting Azure to your datacenter**. The Azure Site-to-Site (S2S) VPN is the foundation of the hybrid cloud. Similarly, you can use this feature to connect virtual networks (VNET) in Azure. You can also take hybrid connectivity a step further with ExpressRoute, which features connections that do not go over the public Internet and offer higher security, reliability, and connection speeds. We will discuss all of these in chapter 5
- **Azure VMs**. With foundational components and concepts under your belt, chapter 6 is an extended deep dive into deployment automation for Azure IaaS leveraging the declarative deployment capabilities of Azure Resource Manager (ARM) templates. Not only will you get step-by-step deployment and best practices guidance, but a working sample to help you through the process!
- **Migration**. In chapter 7, you will learn about some of the tools and techniques you can use to migrate your on-premises VMs to Microsoft Azure.
- **Backup and Disaster Recovery**. In the final chapter, we will explore some of the latest capabilities available in Microsoft Azure for protecting your virtualized workloads running in Azure IaaS VMs, as well as on-premises.

Let's get started!

Chapter 2: Azure PowerShell

One of the promises of public cloud is one of scalability. Public cloud offerings such as Microsoft Azure offer elasticity, and, for all intents and purposes, limitless resources. However, with great power comes great responsibility. As your Microsoft Azure cloud utilization grows, the need for appropriate tooling and governance to manage them becomes more important; e.g., it is not effective or prudent to remote into 50 web servers to make a minor change.

This chapter covers one of the most important components of effective Microsoft Azure cloud utilization: automation via PowerShell.

Introduction

Microsoft Azure PowerShell is often the most effective way of working with Microsoft Azure because it is the one interface from which all actions and administration in bulk are possible. While the Azure Portal provides a great place to get started, and an easy to use interface, it is at times quite limited. Performing operations in the Azure Portal requires a significant amount of manual interaction to accomplish a single task, as well as only providing a subset of the functionality available through the Azure PowerShell module. Running multiple tasks sequentially or in parallel is possible, thanks to PowerShell's inherent scripting capabilities. Azure PowerShell scripts can also be configured to run on a schedule through the Windows Task Scheduler or as a runbook in Azure Automation, Service Management Automation or System Center Orchestrator.

Microsoft also provides an Azure Active Directory (AAD) PowerShell module for working with AAD. This module is utilized during configuration of Office 365 or AAD with on-premises Active Directory via Active Directory Federated Services (ADFS).

How to Install the Microsoft Azure PowerShell Module

In order to work with Microsoft Azure via PowerShell, you will need to install the Microsoft Azure PowerShell (Azure PowerShell) module. To install the module, you need to download Microsoft's Web Platform Installer (WebPI). You can download and run the WebPI from the Microsoft Azure downloads site at http://azure.microsoft.com/en-us/downloads.

To download and install the Azure PowerShell module, perform the following steps:

1. Open a browser and go to http://azure.microsoft.com/en-us/downloads.
2. Scroll down until you reach the Command Line Tools section on the web page and Under Windows PowerShell, Click **Install** as shown in Figure 1.

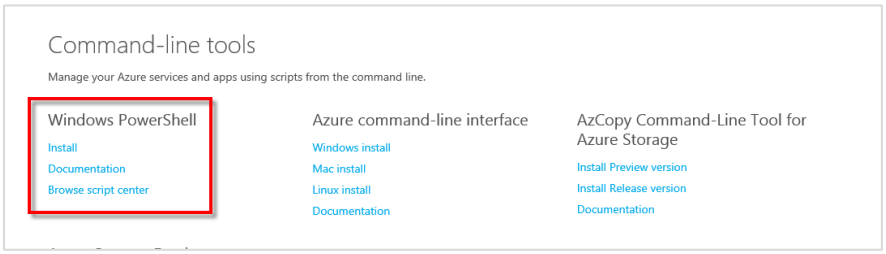


FIGURE 1. MICROSOFT AZURE POWERSHELL TOOLS

3. This will start the download of the Web Platform Installer. When prompted, save the executable (WindowsAzurePowerShellGet.3f.3f.3fnew.exe) to your machine.
4. Next, launch the executable and click the **Install** button, shown in Figure 2.

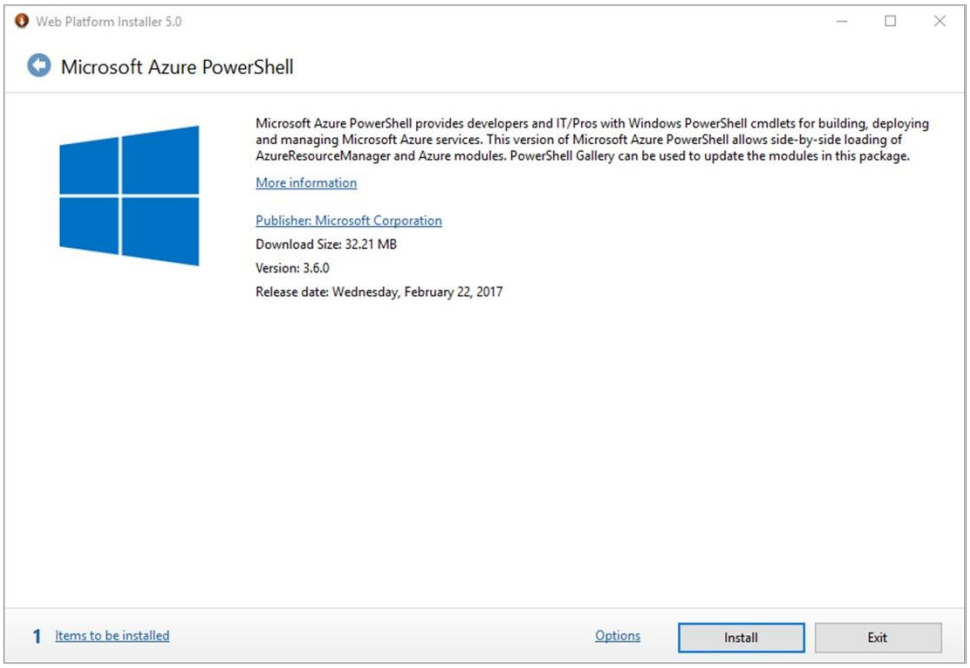


FIGURE 2. WEB PLATFORM INSTALLER FOR INSTALLING AZURE POWERSHELL

5. Click **I Accept**… on the License screen that will pop up to begin installation of the latest version of Azure PowerShell to your machine.
6. Click **Finish** and then **Exit** to close the Web Platform Installer.

Next, open up a PowerShell Prompt and run the following command to verify that the Azure Resource Manager Cmdlets are available:

```
Get-Module -ListAvailable -Name AzureRm.Resources
```

You should get back the following response as shown in Figure 3.

```
ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   3.6.0      AzureRM.Resources                   {Get-AzureRmProviderOperation,
```

FIGURE 3. AZURE RESOURCE MANAGER CMDLETS AVAILABILITY

If you want to see a list of all the existing Azure PowerShell Modules currently installed, run the following command:

```
Get-Module -ListAvailable -Name Azure*
```

The Table below lists the Azure PowerShell Modules that are available in the February 22nd, 2017 release of Microsoft Azure PowerShell.

Azure Resource Manager Modules

ModuleType	Version	Name
Manifest	0.0.4	AzureRM.AnalysisServices
Manifest	3.4.0	AzureRM.ApiManagement
Manifest	2.6.0	AzureRM.Automation
Manifest	2.6.0	AzureRM.Backup
Manifest	2.6.0	AzureRM.Batch
Manifest	2.6.0	AzureRM.Cdn
Manifest	0.4.4	AzureRM.CognitiveServices
Manifest	2.7.0	AzureRm.Compute
Manifest	2.6.0	AzureRm.DataFactories
Manifest	2.6.0	AzureRM.DataLakeAnalytics
Manifest	3.4.0	AzureRM.DataLakeStore
Manifest	2.6.0	AzureRM.DevTestLabs
Manifest	2.6.0	AzureRM.Dns
Manifest	0.0	AzureRM.EventHub

Manifest	2.6.0	AzureRM.HDInsight
Manifest	2.6.0	AzureRm.Insights
Manifest	1.2.0	AzureRM.IotHub
Manifest	2.6.0	AzureRM.KeyVault
Manifest	2.6.0	AzureRM.LogicApp
Manifest	0.11.4	AzureRM.MachineLearning
Manifest	0.3.4	AzureRM.Media
Manifest	3.5.0	AzureRM.Network
Manifest	2.6.0	AzureRM.NotificationHubs
Manifest	2.6.0	AzureRM.OperationalInsights
Manifest	2.6.0	AzureRM.PowerBIEmbedded
Manifest	2.6.0	AzureRM.Profile
Manifest	2.6.0	AzureRM.RecoveryServices
Manifest	2.6.0	AzureRM.RecoveryServices.Backup
Manifest	2.6.0	AzureRM.RedisCache
Manifest	3.6.0	AzureRM.Resources
Manifest	0.11.4	AzureRM.Scheduler
Manifest	2.6.0	AzureRM.ServerManagement
Manifest	0.0	AzureRM.ServiceBus
Manifest	3.5.0	AzureRM.SiteRecovery
Manifest	2.6.0	AzureRM.Sql
Manifest	2.6.0	AzureRM.Storage
Manifest	2.6.0	AzureRM.StreamAnalytics
Manifest	2.6.0	AzureRM.Tags
Manifest	2.6.0	AzureRM.TrafficManager
Manifest	2.6.0	AzureRM.UsageAggregates
Manifest	2.6.0	AzureRM.Websites

Azure Service Management

ModuleType	Version	Name
Manifest	3.5.0	Azure

Azure Storage

ModuleType	Version	Name
Manifest	2.6.0	Azure.Storage

The directory where all the modules are stored is in the table below.

Module	Directory
Azure Resource Manager	C:\Program Files (x86)\Microsoft SDKs\Azure\PowerShell\ResourceManager\AzureResourceManager
Azure Service Management	C:\Program Files (x86)\Microsoft SDKs\Azure\PowerShell\ServiceManagement
Azure Storage	C:\Program Files (x86)\Microsoft SDKs\Azure\PowerShell\Storage

How to Install the Microsoft Azure AD Module

In this book working with Microsoft Azure AD is out of scope, but as a quick reference, if you want to work with Microsoft Azure Active Directory via PowerShell, you will need to install the module from the PowerShell Gallery.

1. Open up a PowerShell Prompt and run the following command

   ```
   Install-Module AzureAD
   ```

   If you are prompted about installing the module from an untrusted repository, type in **A** and hit **Enter**.

2. Once the package has been downloaded and installed, run the following command to verify that the module is available.

   ```
   Get-Module -ListAvailable -Name AzureAD
   ```

   You should get the following response as shown in Figure 4.

```
    Directory: C:\Program Files\WindowsPowerShell\Modules

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Binary     2.0.0.71   AzureAD                             {Add-AzureADApplicationOwner,
```

FIGURE 4. AZURE ACTIVE DIRECTORY MODULES AVAILABILITY

Note: There isn't an alternative way to install the Azure Active Directory Module other than from the PowerShell Gallery. Additionally, there is a V2 version of the Azure Active Directory Module that is currently in public preview that is recommended to install. You can install if by using the module name **AzureADPreview**.

How to Connect to Microsoft Azure via PowerShell

When you connect to Microsoft Azure via PowerShell, all communication occurs via an API (which API depends on which part of Azure you are working with) into Azure. As such, this requires all connections to be authenticated. All information is transmitted over https so your connection is encrypted. Each connection will connect to specific Azure subscription and you must authenticate to the subscription in order to perform tasks with the Azure subscription you are targeting. You have two ways in which you can log into an Azure Subscription:

- Microsoft Accounts (Microsoft Live Account)
- Work or School Accounts (Microsoft Azure Organization Account)

Logging in using either type of account will only authorize Azure PowerShell for 12 hours.

To connect using your Microsoft Live account:

1. Open the Microsoft Azure PowerShell console.

   Type in the following command and hit Enter.

   ```
   Add-AzureRmAccount
   ```

   The first time you run this command, you will be prompted as to whether you would like to participate in Azure PowerShell Data Collection for usability improvements, type in either **N**[No] or **Y**[Yes] and hit Enter.

   Next, type in your Credentials for your work, school or personal Microsoft account as shown in Figure 5.

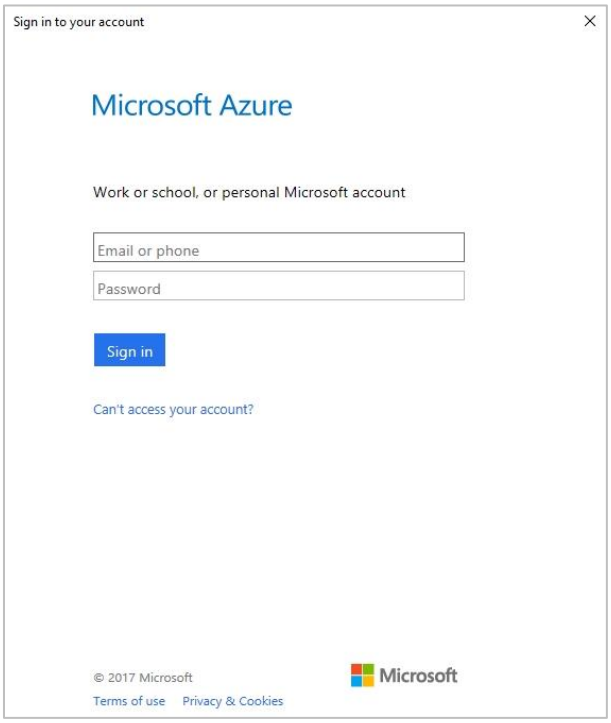


FIGURE 5. MICROSOFT AZURE LOGIN PROMPT

2. Once you have successfully logged in, you should see the following output in the PowerShell Console as shown in Figure 6.

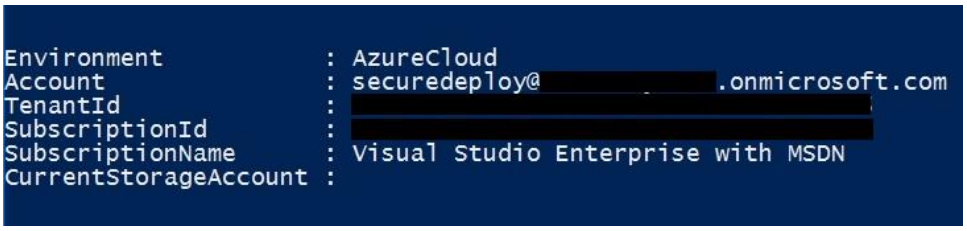


FIGURE 6. AZURE SUBSCRIPTION INFORMATION OUTPUT FROM POWERSHELL

Where are my settings stored?

By default, when you log in to the Azure Portal via PowerShell, the only setting that is saved locally is the opt in or opt out in regards to data collection which is kept in the following path:

```
C:\Users\%username%\Roaming\Windows Azure
Powershell\AzureDataCollectionProfile.json
```

If you want to save your Azure Profile Settings locally, run the following command:

```
Save-AzureRmProfile -Path
'C:\Users\%username%\AppData\Roaming\Windows Azure
Powershell\AzureProfile.json' -Force
```

You can view the **AzureProfile.json** file in notepad and you will observe information relating to the subscription you just imported from.

Working with Microsoft Azure PowerShell

Working with Microsoft Azure resources via PowerShell can be daunting at first. This section walks you through a common scenario of deploying a VM into Azure. Each PowerShell code snippet is separated out so you can understand how each piece relates to the whole, enabling you to review later and easily understand what a particular section was written to do.

Create a Working Directory

As you start creating scripts (this is why you are utilizing the Azure PowerShell module right?!), you may find it useful to create a directory where you create and modify scripts. This also becomes particularly useful if you store your PowerShell scripts in a source control system like Team Foundation System (TFS) or GitHub. A folder located in C:\Scripts is a good place to store scripts as it's easy to find and hard to mistakenly delete.

Selecting a Subscription

If you are logging into Azure with an account that is assigned only to a single Subscription, you can start working against the subscription immediately. However, if you are assigned to multiple subscriptions, you want to target a Subscription to work with before doing anything else. You can do this by running the following command:

```
Select-AzureRmSubscription –SubscriptionId "{SUBSCRIPTION_ID}"
```

The Subscription ID is displayed once logged in through a PowerShell Session and is also available when logging directly to the Azure Portal.

Provisioning a Virtual Machine (unmanaged disks)

In this section, we will go through deploying a VM in Azure Resource Manager deployed to an unmanaged disk using Azure PowerShell.

> Note: managed disks in Azure do not require an additional storage account to be deployed whereas an unmanaged disk does. The full benefits of using managed vs unmanaged disks are outside the scope of this chapter; however, the supplemental scripts for this chapter included options to deploy to both types for thoroughness. For more information about the benefits of managed disks, please visit: https://docs.microsoft.com/en-us/azure/storage/storage-managed-disks-overview

After you have logged into a PowerShell Session and selected the Azure subscription to work with, you can begin provisioning resources into Azure. The following PowerShell commands illustrate how you can provision a new VM into Azure along with its required prerequisites.

> Note: You will notice several of backtick characters (`) in the following code and throughout the rest of the code sample in this book. Do not forget to copy them as they are responsible for allowing the longer commands to wrap into the next line, making it easier to read in the format this book is provided in.

First, start off by configuring the following variables. Change the **location** variable to your particular region you are deploying to, i.e. - **northeurope** for North Europe, **eastus** for East Coast (USA).

- virtual machine name
- virtual machine size
- resource group name
- storage account name

- storage account type
- local administrator username
- local administrator password
- Location

```
$VMName              = "iaas-vm"
$VMSize              = "Standard_DS1_v2"
$ResourceGroupName   = "iaas-vm-demo"
$StorageAccountName  = "iaasvmdemo"
$StorageAccountType  = "Standard_LRS"
$LocalAdminUsername  = "localadmin"
$LocalAdminPassword  = "LetMeInNow1!"
$Location            = "westeurope"
```

> **Note**: It is recommended that you append the **VMName** and **StorageAccountName** variables with additional numbers and characters to ensure they are unique within Azure. For example:
>
> **VMName** = "iaas-vmt4r5"
> **StorageAccountName** = "iaasvmdemot4r5"
>
> The reason the VMName needs to be unique in this scenario is that the Public IP Address that is being deployed is based on the Name of the VM.
>
> Additionally, in production environments, it is never recommended to have your credentials passed in clear text, but as this is to illustrate concepts, we have left them as such.

Next, we'll deploy a new Resource Group for the Virtual Machine and its related resources.

```
New-AzureRmResourceGroup `
    -Name $ResourceGroupName `
    -Location $Location
```

Next, we need to deploy a Storage Account to store the VM OS Disk. This will take a minute or so to finish deploying.

```
$VMStorageAccount = New-AzureRmStorageAccount `
    -ResourceGroupName $ResourceGroupName `
    -AccountName $StorageAccountName `
```

```
    -Location $Location `
    -Type $StorageAccountType
```

Next, we need to retrieve the latest VM Image version of Windows Server 2012 R2 Datacenter available in the location you are deploying to.

```
$VMImage = Get-AzureRmVMImage `
    -Location $Location `
    -PublisherName MicrosoftWindowsServer `
    -Offer WindowsServer `
    -Skus 2012-R2-Datacenter `
    | Sort-Object Version -Descending `
    | Select-Object -First 1
```

Next, create a new VM Configuration.

```
$VMConfig = New-AzureRMVMConfig `
    -VMName $VMName `
    -VMSize $VMSize
```

Next, create a Subnet.

```
$Subnet = New-AzureRmVirtualNetworkSubnetConfig `
    -Name Subnet-$ResourceGroupName `
    -AddressPrefix "10.0.0.0/24"
```

Next, create a VNet. Ignore any warning output from the command.

```
$VNet = New-AzureRmVirtualNetwork `
    -Name VNet-$ResourceGroupName `
    -AddressPrefix $Subnet.AddressPrefix `
    -Subnet $Subnet `
    -ResourceGroupName $ResourceGroupName `
    -Location $Location
```

Next, create a Public IP Address which will be used for RDP Access to the VM. Ignore any warning output from the command.

```
$PublicIP = New-AzureRmPublicIpAddress `
    -Name pip-$VMName `
    -ResourceGroupName $ResourceGroupName `
    -AllocationMethod Dynamic `
    -DomainNameLabel "$VMName-pip" `
    -Location $Location
```

Next, create a NIC Card. Ignore any warning output from the command.

```
$NIC = New-AzureRmNetworkInterface `
```

```
    -Name "$VMName-NIC" `
    -ResourceGroupName $ResourceGroupName `
    -SubnetId $VNet.Subnets[0].id `
    -PublicIpAddressId $PublicIP.Id `
    -Location $Location
```

Next, associate the NIC Card to the VM.

```
$VM = Add-AzureRmVMNetworkInterface `
    -VM $VMConfig `
    -Id $NIC.Id
```

Next, configure the OS Disk Settings.

```
$OSDiskName     = "$VMName" + "_OS_Disk"
$OSDiskCaching = "ReadWrite"
$OSDiskVhdUri  =
"https://$StorageAccountName.blob.core.windows.net/vhds/$OSDiskName.vhd"
```

Next, add the local administrator credentials to a PSCredential Object.

```
$SecureLocalAdminPassword = ConvertTo-SecureString $LocalAdminPassword -
AsPlainText -Force
$Credentials = New-Object System.Management.Automation.PSCredential
($LocalAdminUsername, $SecureLocalAdminPassword)
```

Next, set the VM Operating System Settings.

```
$VM = Set-AzureRmVMOperatingSystem `
    -VM $VM `
    -Windows `
    -ComputerName $VMName `
    -Credential $Credentials
```

Next, setting the VM Source Image for the VM.

```
$VM = Set-AzureRmVMSourceImage `
    -VM $VM `
    -PublisherName $VMImage.PublisherName `
    -Offer $VMImage.Offer `
    -Skus $VMImage.Skus `
    -Version $VMImage.Version
```

Next, setting the Operating System Disk settings for the VM

```
$VM = Set-AzureRmVMOSDisk `
    -VM $VM `
    -VhdUri $OSDiskVhdUri `
    -Name $OSDiskName `
```

```
    -CreateOption FromImage `
    -Caching $OSDiskCaching
```

Finally, deploying the Azure VM.

```
New-AzureRmVM `
    -ResourceGroupName $ResourceGroupName `
    -Location $Location `
    -VM $VM `
```

Once the VM has finished being deployed, you should get the following response back as shown in Figure 7.

```
RequestId IsSuccessStatusCode StatusCode ReasonPhrase
--------- ------------------- ---------- ------------
                         True         OK OK
```

FIGURE 7. VM DEPLOYMENT STATUS

At this point, you have the option to use the DNS Name of the Public IP Address you deployed earlier to RDP into the VM using the credentials from in the initial variables.

Once you have finished exploring the deployed resources, it is recommended that you delete the Resource Group they were deployed to.

Additional Deployment Scripts

You can find additional scripts for deploying VMs to Azure for the following scenarios from GitHub at https://github.com/insidemscloud/AzureIaaSBook in the **Chapter 2** directory.

Script Name	Description
deploy-unique-vm-to-azure-unmanaged-disk.ps1	Deploy a VM to Azure on an unmanaged disk
deploy-unique-vm-to-azure-managed-disk.ps1	Deploy a VM to Azure on a managed disk

The difference between the scripts you'll find on GitHub and the walkthrough you just completed is that each script on GitHub is fully parameterized, contains detailed

information on how to use the script and includes extensive error handling to help you troubleshoot the script if it fails.

Additionally, these scripts ensure that the resources that are deployed into Azure are unique by appending 4 alphanumeric characters to the end of each resource from a GUID that is generated at runtime.

Additional examples from Microsoft can be found in Microsoft's Script Center at the following URL: http://azure.microsoft.com/en-us/documentation/scripts/

As you continue you through this book, additional PowerShell script examples will be provided that you can modify to work with your environment.

What about full-scale automation?

While deploying a single VM using Azure PowerShell is useful for demonstration purposes, it is better to use Azure Resource Manager (ARM) Templates for large scale deployments as ARM Templates allow you to track the deployment process from beginning to end, with full logging available via the Azure Portal and Azure Resource Explorer. Additionally, it is possible to deploy ARM Templates using Azure PowerShell.

Authoring ARM Templates is worth of being covered in a book by itself, and as such, is outside the scope of this chapter. If you would like to read more about ARM Templates, you can visit https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-authoring-templates.

Some basic information and examples of how to use ARM Templates for VM deployment are included in *Chapter 6 - Azure Virtual Machines*.

Updating Azure PowerShell

Just about every time that Microsoft releases new Azure features, the PowerShell cmdlets are updated. The core cmdlets have become fairly stable, with significant syntax changes becoming much less frequent. However, you should make it a practice to update your development environment with the most recent Azure PowerShell module after a new wave of Azure features is announced, and verify that your scripts and workflows still function as expected. Additionally, new cmdlets are added to the module allowing you to modify your scripts to become more succinct and less complex.

Chapter Summary

The Azure PowerShell module provides a set of capable, robust automation commands. While the Azure portal has gone through a few iterations over the last couple years, the PowerShell commands have largely gone unchanged, providing a familiar, reliable interface for managing your Azure resources. In addition, the PowerShell cmdlets provide a superset of what is exposed through the Azure Portal and are often much less time consuming or tedious than performing the same action through the browser.

In addition to providing increased operational efficiency to administrators familiar with PowerShell, Azure PowerShell also enables automation as part of a larger solution-based approach. For example, System Center Service Manager(SCSM) can trigger a Service Management Automation (SMA) or an Orchestrator runbook from a service offering, which in turn could trigger the provisioning of a VM in Azure or a set of services related to that offering. Azure PowerShell helps to bring the Azure VM lifecycle into appropriate enterprise governance and lifecycle policies.

Chapter 3: Azure Virtual Networking

As cloud consumption grows, more and more cloud resources begin to depend on each other. One of the key areas that Microsoft Azure resources are tied together is through the use of *virtual networks (VNET)*. VNETs provide a mechanism for Azure resources to communicate, not just with each other, but optionally with your on-premises resources via a site-to-site VPN or ExpressRoute connectivity.

Of course, with great capability comes great responsibility. How does one govern and secure external access to Azure resources? This chapter details some of the capabilities that will enable you to manage and maintain your Azure VNETs efficiently and securely. Before going deep into the network design, this chapter will cover the core concepts of Azure virtual networking in Azure Resource Manager (ARM), as well as common scenarios for networking in Azure.

A couple of important notes:

Before we get started, there are a couple of important points we need to touch on briefly:

- When working with Azure Resource Manager you must always use the Azure Portal at **https://portal.azure.com.** These resources are not visible in the old Azure Management Portal at https://manage.windowsazure.com.

- Connectivity options between corporate networks and Azure, including site-to-site VPN and ExpressRoute, will be covered in depth in *Chapter 5 - Connecting Azure to Your Data Center*. They are mentioned briefly in this chapter in discussions related to hybrid connectivity.

Basic network components

Azure Services and applications are comprised of many different components, like a storage account or a network interface. These components reference each other and are presented as a single entity like, for instance, a virtual machine (VM). However, if we take a closer look, the VM is actually comprised of a compute component, a storage account or managed disk, a network interface and possibly a public IP address. All of these components are stored in a *resource group*, a logical container for grouping resources

belonging to an application, project, environment, etc. For more information about resource groups, see *Chapter 2 – Azure PowerShell*. Figure 1 shows a resource group called CONTOSO-HR with a VM, including related components; in this example, a storage account and a network interface.

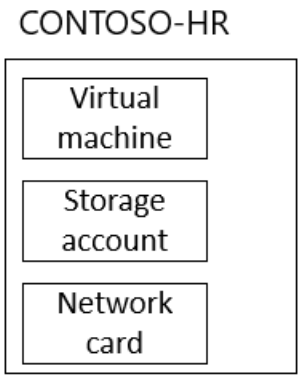


FIGURE 3. RESOURCE GROUP WITH COMPONENTS

In addition to the components depicted in figure 1, a VM must be connected to a VNET. The VNET can be created in the same resource group or reside in another resource group. One of the reasons to create the VNET in another resource group is the need for role-based security. For example, multiple application servers in different resource groups may use a particular VNET, but only the network engineers must have permissions to configure, manage and maintain the VNET.

Before we walk through an example of a VNET, we will first discuss IP addressing in Azure networking.

IP Addressing

Azure networking uses a number of different types of IP addresses.

- **Internal IP Address (DIP)** - An internal IP address, or DIP, is an internal IP address assigned to an Azure resource, such as a network interface that is connected to a VM. A network interface performs the same function as a physical network card and in Azure Resource Manager, can be attached to a VM or a load balancer. By default, this IP address is assigned dynamically and if the VM is de-provisioned (i.e. turned off from the Azure Portal), it will lose its DIP. It is possible to configure a static DIP. The IP will be static from the standpoint that the VM will

always get the same IP address from the Azure fabric. However, configuring a static IP address in the properties of the network interface inside guest OS is not supported. The guest OS should always be configured to receive dynamic IP address assignment via DHCP. However, when you allocate a static DIP the guest OS IP address settings will always reflect that same DIP. Therefore, you don't need to change anything in the guest OS.

On each subnet, Azure reserves the first three IP addresses for internal use. The first IP address that a VM or other resource can use is .4.

- **Instance-Level Public IP Address (PIP)** - A PIP is a public IP address that you assign directly to your network interface. You can use the PIP to connect directly to your VM from the Internet.
- **Virtual IP address (VIP)** - A VIP is a public IP address automatically assigned to a network load balancer. This IP address can be used for load balancing or network address translation (NAT) in network interfaces and VMs behind the network load balancer.
- **Azure Datacenter IP Ranges**. In some scenarios, you need to know all the public IP addresses that different Azure data centers use. This list is frequently updated and can be downloaded in XML format from the Microsoft Azure website. Search for "Microsoft Azure Datacenter IP Ranges", currently at http://www.microsoft.com/en-us/download/details.aspx?id=41653.

Virtual Networks (VNETs)

VNETs are used in the same way as local area networks in your local data center. VNETs in Azure provide the following capabilities:

- **Isolation** - By default, all VMs within a VNET can communicate with each other. To totally isolate VM traffic from each other, you can place them on different VNETs. A VNET can span a region (multiple physical data centers) but not multiple locations (East US to West US locations).
- **Access** - Access between VMs on the same VNET is open, even if the VMs are on different subnets within the VNET.
- **Connectivity** - VNETs can be connected to each other, as well as to on-premises data centers. Connectivity to an on-premises data center is achieved with either

ExpressRoute or VPN. A single VNET can be connected to multiple VNETs and on-premises data centers at the same time.

Previously, when creating a VNET, you were required to specify an affinity group. An affinity group is a way to group resources together inside the data center to minimize latency. With the improved performance in Azure data center networks today, affinity groups are no longer required. VNETs are now associated with regions, which includes one or more physical data centers. You can still specify an affinity group if your application requires it, but it is not required otherwise.

The number of VNETs you need depends on what you are planning to deploy in Azure. Changing VNET settings on a resource after a deployment can be complex and complicated sometimes, so it is a good idea to plan the network design before deploying any resources in Azure. You can redeploy your VMs to change VNET settings, but that will result/require a downtime. It is always better to spend time for planning, before deploying your systems or applications to Azure.

Figure 2 shows a sample configuration with three resource groups.

- **CONTOSO-HR** - This resource group contains two VMs with related network interfaces and a storage account that is shared by both VMs.
- **CONTOSO-WEB** - This resource group contains one VM with a related network interface and storage account.
- **CONTOSO-INFRA** - This resource group contains the VNET that all VMs are connected to. The example in figure 2 uses one VNET for all VMs.

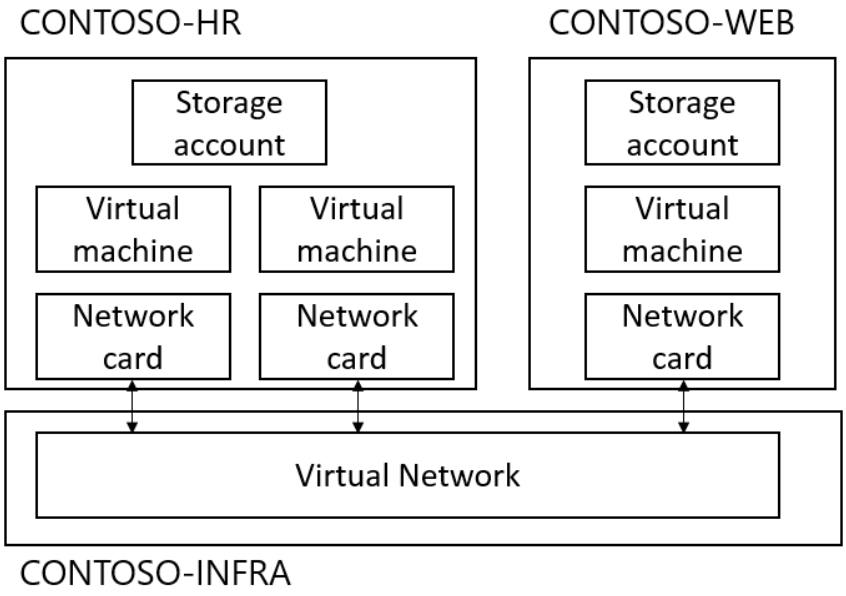


FIGURE 4. VNET SETUP

Figure 3 shows another example with two VNETs. By default, all VMs within a VNET can communicate. However, in this example, the two VNETs are totally isolated from each other, which allows administrators to easily create isolated environments within an Azure subscription, such as separating testing and production VMs.

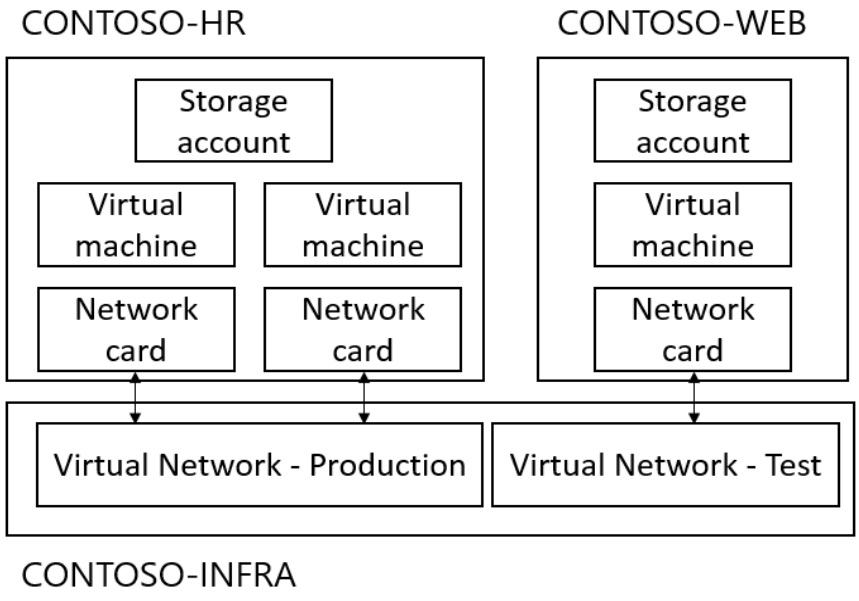


FIGURE 5. MULTIPLE VNETS

VNET Settings

When configuring a VNET, you need to specify the following settings. The descriptions of these settings will provide guidance on what to include when planning VNETs.

1.1.1.1.1 Name

Name of the VNET. The name should describe the purpose or function of the VNET.

1.1.1.1.2 Location

The VNET will be bound to a location (region) hosting an Azure regional data center, for example, North Europe. A VNET in the same or different region can be connected to another VNET in the same or different subscriptions using a VNET-to-VNET VPN. A VNET cannot cover multiple locations but can cover multiple data centers in the same region.

1.1.1.1.3 Address space and subnets

You need to configure which IP addresses can be used in your VNET. IP addresses will be dynamically assigned to resources by default. It is important to select an address range that does not overlap with on-premise networks. Optionally, you can configure subnets for your VNET with the following rules in mind:

- All subnet IPs must be within the VNET address space.
- The smallest supported subnet is /29.
- Divide different application tiers, for an example backend (data tier), mid-tier (application tier) and front-end (presentation tier), into different subnets. When using different subnets, it is easier to implement and manage security access rules between the subnets.
- It is recommended to separate resources with static IP addresses from resources with dynamic IP addresses. It is then easier to see which resources have static IP addresses, but it also prevents a new instance from acquiring the static IP address from a resource that is in a stopped (de-allocated) state. This does not happen if you have only VMs in your subnet but can also affect PaaS instances on the subnet, which use the web and worker roles (basically non-persistent VMs) if you are using the classic (Azure Service Management) model for deployment.
- The gateway subnet can be either the first address space in your VNET or the last. If you use the first address space for a gateway subnet, you can modify all the other subnets in that VNET, without affecting the gateway subnet at a later point.
- Resources in different subnets within the same VNET can communicate with each other by default. Resources in different VNETs cannot communicate with each

other by default. You can enable communication between VNETs by configuring a VNET-to-VNET VPN.

- All RFC 1918 IP ranges can be used for internal addresses
    - 10.0.0.0-10.255.255.255 (10/8 prefix)
    - 172.16.0.0-172.31.255.255 (172.16/12 prefix)
    - 192.168.0.0-192.168.255.255 (192.168/16 prefix)
- Non-RFC 1918 addresses, public IP addresses, can also be used internally on subnets. It is important to only use addresses that you own, and it is also important to remember that these addresses can only be used internally, not to route Internet traffic.

1.1.1.1.4 DNS Servers

A key capability of Azure VNETs is the ability to define DNS servers for your cloud-hosted VMs and Azure resources. These DNS servers are defined in name and IP address pairs, separate from the VNETs, but each DNS server can be bound to multiple VNETs. The capability is very similar to the implementation of DNS properties on IP Pools in System Center 2012/2012 SP1/2012 R2 Virtual Machine Manager (VMM), where VMs deployed using that particular IP Pool have their DNS server address properties already set. There are essentially two types of DNS servers that you can add to an Azure VNET:

- **Azure DNS** - If you do not specify a DNS server, name resolution will be provided by Azure. Azure provides name resolution for VMs within the same virtual network, based on the Fully Qualified Domain Name (FQDN). If you need name resolution between VNETs, you will need to provide your own DNS server.

  Azure DNS is an Azure service that can host your domains in Azure. You can then manage your DNS domains with the same credentials and support as with other Azure services. Microsoft Azure includes a large global network of DNS servers for high availability and good performance. If you configure DNS as a service, also you can use it as the DNS service for your virtual networks. A benefit of using DNS as a service is the possibility to easily add and modify DNS settings when working with other resources in Azure. For an example, when you deploy a new set of VMs, you can easily configure DNS entries for these VMs.

- **Custom DNS**. If you configure a list of custom DNS servers, make sure to verify they are working properly. If the first DNS server on the custom DNS list is reachable, the VMs will use that DNS server regardless of whether the DNS server is functioning properly or not. To change the DNS server order for your virtual network, remove the DNS servers from the list and add them back in the correct order. If you change DNS

settings, VMs need to restart to pick up the new DNS Server settings. Custom DNS servers can be provided in multiple ways.

- **Cloud-based DNS**. In this scenario, you provision a VM with DNS services in the network, and add a DNS server entry to the VNET, using the internal IP address of that VM. Other Azure VMs will then use this Azure VM (or VMs) to perform name resolution. It is recommended to configure a static IP address for VMs hosting DNS.
- **On-premises DNS**. In this scenario, you have a VPN configured between the VNET and on-premise, and then you add a DNS server entry to the VNET using the on-premises internal IP address of the DNS server. The Azure VMs communicate over the VPN or ExpressRoute connection to the on-premises DNS server to perform their DNS lookups.
- **External DNS**. In this scenario, you add a DNS server entry to the VNET, using the IP address or addresses of externally hosted DNS services, such as OpenDNS. The Azure VMs will then connect to this external DNS service to perform their DNS lookups.

Of course, you can mix these three types of DNS servers and utilize any combination of these, providing a very flexible, robust DNS service to your Azure VMs. There is no strict recommendation to always use one type of DNS or a specific order of DNS servers in a mix, but the best fit depends on the scenario and the environment.

1.1.1.1.5 Users

With the support for role-based access control (RBAC) on VNETs, you can grant appropriate access to the VNET to support your administrative model. There are a number of default roles that can be used. For an example, you can give different teams (HR and Web) access to use the VNET, but not modify it. Administrators of these resource groups will have permissions to connect network interfaces to the VNET, but not to modify the VNET itself.

1.1.1.1.6 Tags

In the Azure Portal, you can organize resources into resource groups. You can use role-based security to control access to these resource groups. To organize resources across resource groups, you can use tags. Tagging resources with name or values to categorize them enables you to then list and report on resources across resource groups by tag. Figure 4 shows tags for a VM. Three tags have been added to the VM in this example; Budget, Function, and Team.

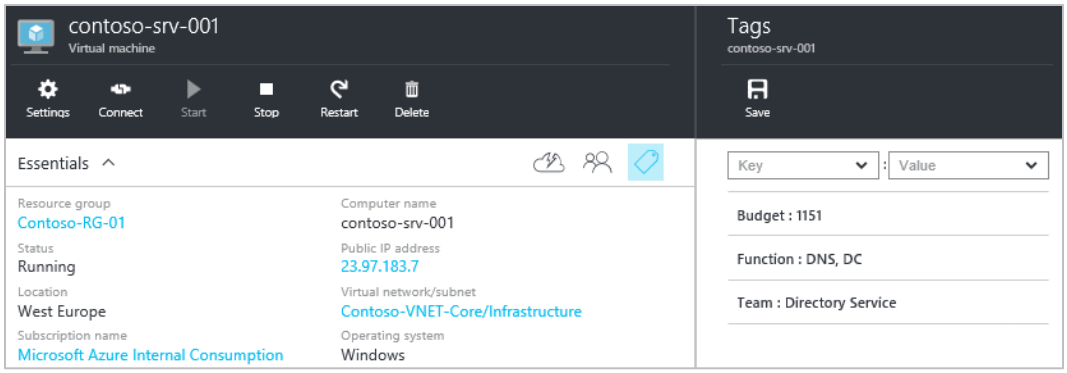


FIGURE 6. TAGS FOR A VIRTUAL MACHINE

Network Interface

Introduced with Azure Resource Manager, the network interface is an independent resource that is referenced (connected) to other resources, such as a VM or a load balancer. Building network interfaces as independent resources bring many administrative benefits. For example, you can configure a network interface with a public IP address and connect it to a VM. You can later move that network interface, including the public IP, to a network load balancer, retaining all of its settings.

Network Interface settings

When setting up a network interface you need to specify several settings. The descriptions of these settings offered here will provide you a good understanding of what to include when planning network interfaces.

1.1.1.1.7 Name and VNET

A name for the network interface is required and ideally, should describe the purpose of the network interface. When creating a new network interface, you should also select which VNET it should connect to as well.

1.1.1.1.8 Attached to and Load Balancers

You can connect a network interface to a VM or to a load balancer. These settings are used to set the reference between a network interface and the attached device, which is the device using the network interface.

1.1.1.1.9 IP Addresses

Each network interface can be configured in two different ways from an IP address point of view. They can be configured to assign a public or static IP address from the VNET. If

the network interface is configured with a static IP address, you can configure which static IP address to use. If you configure the network adapter with a dynamic IP address, the VNET will assign an address to the network adapter when its associated resource is booted or becomes active.

The network adapter can also be configured with a public IP address. The Public IP address is an independent resource in Azure, which is referenced to the network adapter. When configuring a network adapter with a public IP address, you can choose either to get a new public IP address or use a public IP address that you already have created in your Azure subscription.

1.1.1.1.10 DNS Servers

By default, a network adapter uses the DNS settings specified on the VNET. However, it is also possible to specify custom DNS settings per network adapter.

1.1.1.1.11 Users and Tags

Users and tags are used the same way as described earlier in the chapter under VNET settings.

Connecting to on-premise environment

Connecting VNETs in Azure to on-premise data centers is possible through a couple of different ways. You can configure a VNET gateway after the VNET is created. The gateway will be used to connect either the express route connection or a VPN connection.

Point-to-site VPN

Point-to-site VPNs can be used for remote workers, for an example remote field engineers, to securely connect to Azure networks from their laptops.

For more information, see section "5.2 Point-to-Site VPN" in *Chapter 5: Connecting Azure to Your Data Center*

Site-to-Site VPN and ExpressRoute

With a site-to-site VPN, you can extend your local data center network to include connectivity to the networks in Azure. With a fully routed site-to-site VPN including DNS name resolution, the network in Azure will become part of your corporate network.

In an Azure VNET, the gateway type defines how a gateway connects and is a mandatory setting in Azure Resource Management. This is different from the VPN type which defines the routing type of your VPN connection. There are two gateway types, ***VPN*** & ***ExpressRoute,*** and the latter is used explicitly for ExpressRoute connections or ExpressRoute/S2S/P2S coexistence scenarios. On the other hand, if you using the classic deployment model you do not need to specify this but rather select a static routing gateway or a dynamic routing gateway based on your requirements.

When setting up a site-to-site VPN connection in Azure Resource Manager, you can configure a Policy-based or Route-based VPN types. These are referred as static routing and dynamic routing gateways in the classic deployment model. You should configure a Route-based VPN type/dynamic routing gateway if possible if the on-premises hardware supports this features. Route-based VPN type is required for multi-site VPN (connecting multiple on-premises sites to one VNET), VNET-2-VNET (connecting multiple VNETs), Point-to-Site (connecting from a VPN client to an Azure VNET) and forced tunneling. For more information on forced tunneling, see "Forced Tunneling and User Defined Routes" later in this chapter. The concept of Azure VPNs will be discussed at length in *Chapter 5: Connecting Azure to Your Data Center*.

When planning a site-to-site VPN, evaluate the bandwidth requirements. The encryption overhead of the VPN will use around 20% of the bandwidth. For example, if you pay for a 100 Mbps VPN connection, the maximum bandwidth will be around 80 Mbps.

ExpressRoute is another option to connect on-premise networks with networks in Azure. ExpressRoute connections do not use the public Internet. Instead, connectivity is provided through an ExpressRoute-carrier partner directly from your data center to the Azure data center, either with a private link or MPLS network. ExpressRoute connections offer higher security, more reliability, faster speeds and lower latencies than typical connections over the Internet.

A great benefit of using ExpressRoute instead of site-to-site VPN is the low latency. With Site-to-Site VPN, even with one of the high-performance SKUs, there will still be latency on the connection. An ExpressRoute connection can also be used to access other Azure services, such as Office365.

For more information, see sections "5.3 Site-to-Site VPN" and "5.5 ExpressRoute" in *Chapter 5: Connecting Azure to Your Data Center* in this book.

Publish a service to the Internet

VMs within the same VNET can communicate directly with each other. They can also communicate to the Internet, but the traffic must be initiated from the VM. A client on the Internet cannot connect to a service on the VM by default. There are two ways to publish a service to the Internet; either with a public IP address (PIP) on the network interface related to the VM or with a load balancer. A load balancer can be configured with a public IP address and then you can use NAT and load balancing rules to forward and control incoming traffic. For example, a load balancer can be used to translate incoming traffic on port 4050 to port 3389 for remote desktop (RDP) connectivity.

With a public IP address (PIP), traffic goes directly to the VM and is not routed through the Azure Load Balancer. Traffic to the Internet from a VM with a PIP configured is sent over the PIP instead of a VIP. Therefore, there is no need to configure a load balancer when using a PIP. When connecting an Azure VM directly to the Internet, firewall protection is a requirement. When assigning a public IP address to a network interface, you should review and configure the local firewall settings to protect against external threats.

Azure Load Balancer

Internal Load Balancer

Figure 5 shows traffic to and from two VMs. One VM has a network interface configured with a PIP (a public IP address). The second VM is behind a load balancer, with the public IP address assigned to the network load balancer. The second VM network interface is configured as part of the address pool for the load balancer.

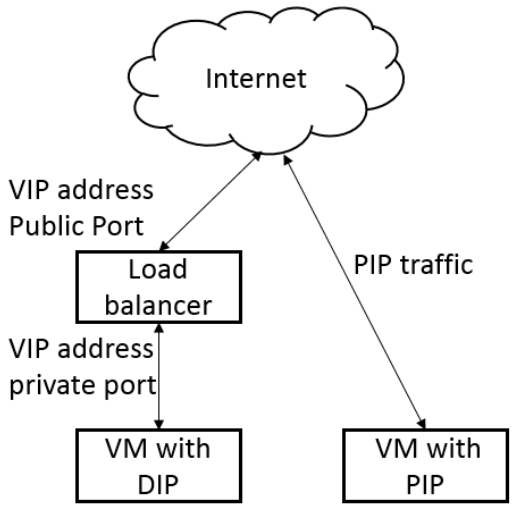


FIGURE 7. TRAFFIC TO VIRTUAL MACHINES (VM)

Internet Facing Load Balancer

Figure 6 illustrates another example, with a load balancer configured to balance incoming traffic on port 80 to three VMs.

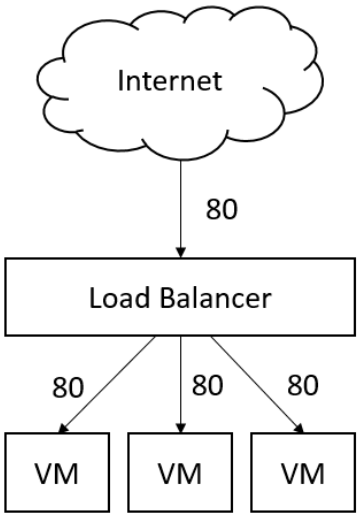


FIGURE 8. LOAD BALANCING OF INCOMING TRAFFIC

The following parameters need to be configured when creating a load balancer:

- **Front end IP configuration -** the public IP address on the front of the load balancer to accept incoming traffic.
- **Backend address pool -** the network interfaces which will receive the load-balanced traffic.
- **Load balancing rules -** source and local port configuration for the load balancer.
- **Probes -** configures the health status probe for VMs.
- **Inbound NAT rules -** configures port translation between public IP address and a VM.

When should you create a load balancer instead of static IP?

Publishing a service by using a PIP address is a quick solution. In that case traffic from Internet connects directly to the VM. However, if you need to publish a service from multiple VMs, you need to create a load balancer in front of them. All the VMs can then share the public IP address. You can also use the load balancer to configure network address translation (NAT) to forward different ports to different VMs.

Network Security Groups

A Network Security Group (NSG) is an object that can control traffic (i.e. deny or allow) to VM instances or to a subnet within a VNET. When an NSG is applied to a VM, all traffic that is sent (outbound) and received (inbound) by the VM instance is controlled by the NSG settings. When an NSG is applied to a subnet in a VNET, NSG rules are applied to all traffic sent and received by ALL the VMs in that subnet.

A VM or a subnet can be affected by only one NSG, but an NSG can contain 200 rules. A supported configuration is to associate one NSG on the VM and one NSG on the subnet. The result will be that the VM gets two layers of protection. The NSG is an additional security layer that, together with a firewall (such as the Windows Firewall on Windows VMs), Resource Groups, and VNET isolation ensure you can build a secure network infrastructure in Azure.

One important thing to keep in mind is that NSGs are stateful. What this means is that an NSG tracks the operating state and characteristics of network connections traversing it and distinguish legitimate packets for different types of connections. Also, only packets matching a known active connection are allowed to pass through.

When configuring NSG rules, the following settings need to be specified:

- **Name**. A name that describes the rule.
- **Type**. Specify if the rule is for outbound or inbound network traffic.
- **Priority**. You can specify a priority between 100 and 4096. All rules are applied in the order of priority, a rule with lower priority number (For an example 10) is processed before rules with higher priority number (For an example 31).
- **Source IP address.** Source IP address range.
- **Source port range.** Specify a port or a range of ports between 0 and 65000.
- **Destination IP address.** Destination IP address range.
- **Destination port range.** Specify a port or a range of ports between 0 and 65000.
- **Protocol**. Specify TCP, UDP or * (all).
- **Access**. Access can be either Allow or Deny.

There are two types of NSG rules, inbound and outbound. The following diagram explains how inbound and outbound rules are processed within an NSG. If you are familiar with ***iptables*** in Linux processing mechanism shown in figure 7 below should be familiar to you.

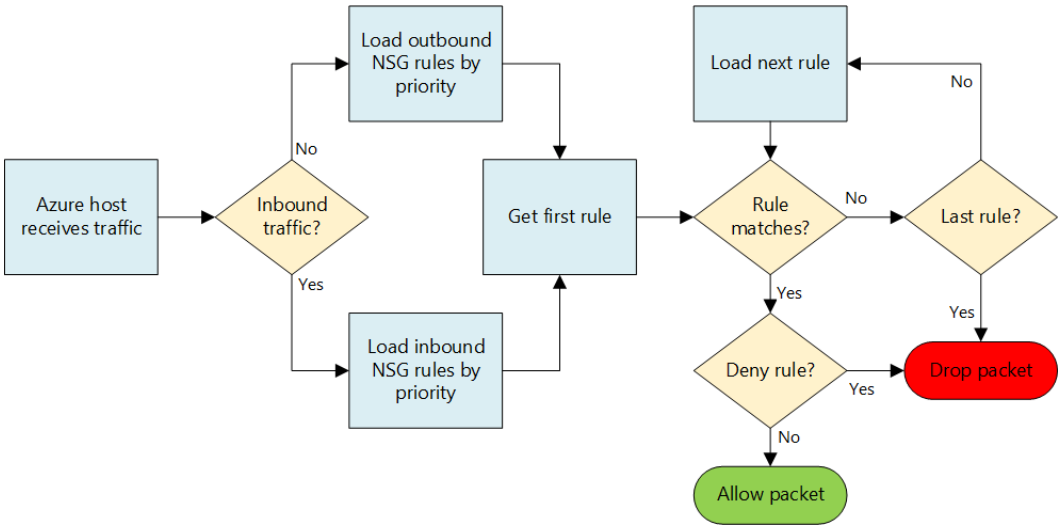


FIGURE 9. PROCESS FLOW FOR NETWORK SECURITY GROUPS

If you have a VM with multiple network interfaces, an NSG can be configured on each network interface.

How should VMs connected to the Internet be secured?

As with on-premise data centers, having multiple layers of security is the best approach. While you can connect an Azure VM directly to the Internet, you should not do so without configuring NSG, antivirus, and local firewall settings to protect against external threats.

Azure Traffic Manager

Traffic manager is a network service that you can use to apply an intelligent policy engine to DNS queries. It may help to think of Traffic Manager as an intelligent DNS used to facilitate high availability for your applications across multiple Azure data centers. Traffic Manager simply requires you to have your application or service deployed in two or more Azure data centers. The process is illustrated in figure 8 and described in the steps below.

1. A user on the Internet requests a web page, for example, http://www.contoso.se. The URL is resolved by a DNS server and points to a traffic manager domain name. This is achieved by using a CNAME resource record that maps the company domain name, in this case, **www.contoso.se**, to the Traffic Manager domain name, For an example **contoso.trafficmanager.net**.
2. The user on the Internet sends a new DNS query to the Traffic Manager domain name, contoso.trafficmanager.net, which is received by Traffic Manager. Traffic Manager uses the specified load balancing profile to determine which Azure endpoint (or other endpoint) should service the request.
3. Traffic Manager sends back the CNAME record for the endpoint to the user. The user's DNS server resolves the record to its IP address.
4. The user accesses the endpoint directly using its IP address. The user continues to use the endpoint as the resolved IP address is cached on the client machine. The user will use the same endpoint until the local DNS cache entry expires.

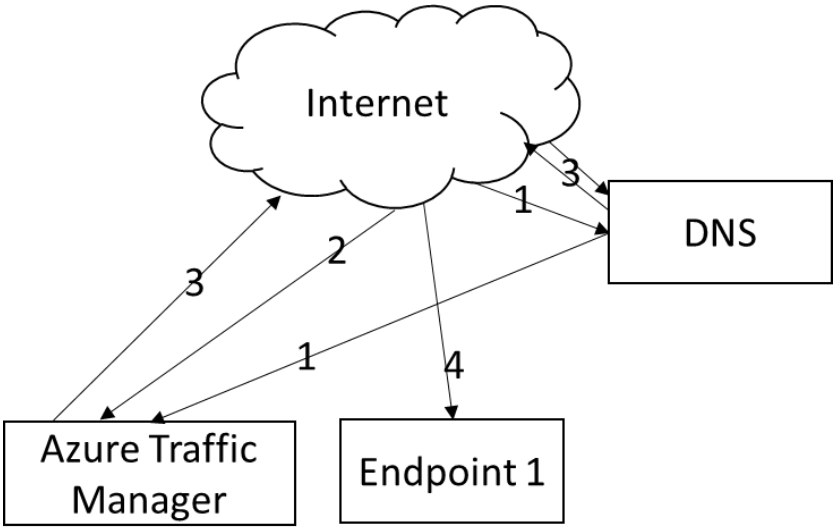


FIGURE 10. TRAFFIC MANAGER LOOKUP AND CONNECTION SEQUENCE

It is possible to nest up to 10 levels of traffic manager, and each profile can be configured with a different method. Traffic Manager offers three load balancing methods,

- **Failover**. Use this profile when you have a primary endpoint you want to use for all traffic, but if that endpoint is not available failover to a backup endpoint.
- **Round Robin**. Use this profile if you want to distribute clients over a set of endpoints in the same data center or in different data centers.
- **Performance**. This profile is recommended when you have endpoints in different geographic locations and you want the client to use the closest one to minimize latency.

Forced Tunneling and User Defined Routes

By default, VMs in Azure always have direct Internet access. With forced tunneling, you can configure VNET subnet(s) to route Internet-bound traffic back to your on-premise data center, through either a site-to-site VPN or ExpressRoute connection. This behavior is configured per-subnet in a VNET and is a critical security requirement for many organizations to control all traffic bound for the Internet. Forced tunneling gives IT administrators an option to inspect and audit Internet-bound traffic.

When forced tunneling is used, remember it affects all outbound traffic for a complete subnet. One thing to remember is that on a subnet that uses forced tunneling, a request

from an Internet client to a VM on the subnet will result in the request being sent back through the corporate firewall. Therefore, it is better to publish the VM service through the corporate firewall instead of directly to the VM in Azure.

Forced tunneling requires a Route-based VPN type gateway, which for site-to-site VPNs means you will need a VPN device that supports dynamic routing. For a list of supported devices and their support for dynamic routing, see "Known Compatible VPN Devices" on the Microsoft website at https://msdn.microsoft.com/en-us/library/azure/jj156075.aspx#bkmk_VPN_Devics.

> **Note**: If forced tunneling is of interest to your organization, you can find step-by-step guidance in "Configure forced tunneling" on the Microsoft site at https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-forced-tunneling/
>
> Forced tunneling can only be achieved through Azure PowerShell for both ASM & ARM modes of deployment.

More information on connecting your corporate networks to Azure is available in *Chapter 5: Connecting Azure to Your Data Center* in this book.

User Defined Routes

User defined routes are yet another powerful feature in Azure Resource Manager. By default, VMs can communicate according to a number of pre-defined system routes in Azure fabric:

- VMs within the same subnet
- VMs on different subnets within the same VNET
- From a VM to the Internet
- From a VNET to another VNET through a VPN gateway
- From a VNET to local data center through a VPN gateway

For most scenarios, these routes work, but there are scenarios where traffic needs to be routed in a different way. One good example for this is when a virtual appliance is in use, (a VM that runs a pre-installed application used to proxy or inspect network traffic in some way, such as a firewall). For each virtual appliance, you need to enable IP forwarding for the VM. IP forwarding is enabled in the Azure Portal. You can configure multiple routing tables and a routing table can be used on multiple subnets, but a subnet can only

be associated with one routing table. Subnets rely on default system routes until a custom routing table is associated to the subnet. Another example would be force tunneling to the Internet via your on-premise network as described in the previous section.

> **Note**: A step-by-step guide on configuring user-defined routing and IP forwarding can be found at https://azure.microsoft.com/en-us/documentation/articles/virtual-network-create-udr-arm-ps/ and https://azure.microsoft.com/en-us/documentation/articles/virtual-network-create-udr-classic-ps/

Networking Planning and Deployment

Planning your networking strategy in Azure is a key task which you should complete before you begin to deploy resources to Azure. Failure to do so will result in substantial wasted effort and repeated work. You can identify the optimal network design by answering the following key questions and reviewing the key considerations:

1. **Start by reviewing capacity limitations for Azure**. These limitations and capacity limits are frequently updated and it is recommended that you review them before each design. More information is available in "Azure Subscription and Service Limits, Quotas and Constraints" at http://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#networking-limits
2. **Reviewing network features in Azure.** Microsoft is rapidly adding features to Azure with each release cycles. You should take the time to review the latest Azure network features before each design.
3. **How will you handle billing?** If your organization requires an invoice per team or application, it is easier to use multiple Azure subscriptions, tags or multiple resource groups. Each team or application will then receive their own invoice.
4. **Identify the number of VNETs that will be needed.** The main question for VNETs is if you need to separate resources within a location. By default, resources on a VNET cannot communicate with resources on another VNET. If there are no requirements for separate resources there is no requirement for multiple VNETs. VNETs are per location, so if you planning to use multiple regions you will also need multiple VNETs. A VNET in one location can be connected to VNETs in other locations or the same location, with a VNET-to-VNET VPN.

5. **Which VNETs will require connection to corporate networks or mobile users?** Understanding which networks will host resources accessed directly by users will help determine routing and security requirements.
6. **How many subnets are needed?** Depending on the number of resources you will deploy, you may need multiple subnets, each with a range of IP addresses. However, there are also other considerations when designing subnets, such as whether you will use NSGs and forced tunneling. Both forced tunneling and NSGs are applied per subnet. Therefore you need to think about which server roles to place together in each subnet. Resources with static IP addresses should be placed on a separate subnet, as discussed earlier in this chapter.
7. Once regions, VNETs, and subnets are designed, you can think about use cases for load balancing and/or Traffic Manager.

Network Design Example

A common network design for Azure networking is shown in figure 9. It is one Azure subscription that contains multiple resource groups (HR Resource Group and Infrastructure Resource Group). The Infrastructure resource group stores a VNET with four subnets. Three subnets for VMs and one gateway subnet. Each application or development team has its own resource group where they can create resources and connect to the central VNET. Each team can login to the same Subscription in the Azure Portal without the risk of affecting another team's resources.

The central VNET is connected to an on-premises data center with a VPN or ExpressRoute connection. The backend subnet is protected by an NSG.

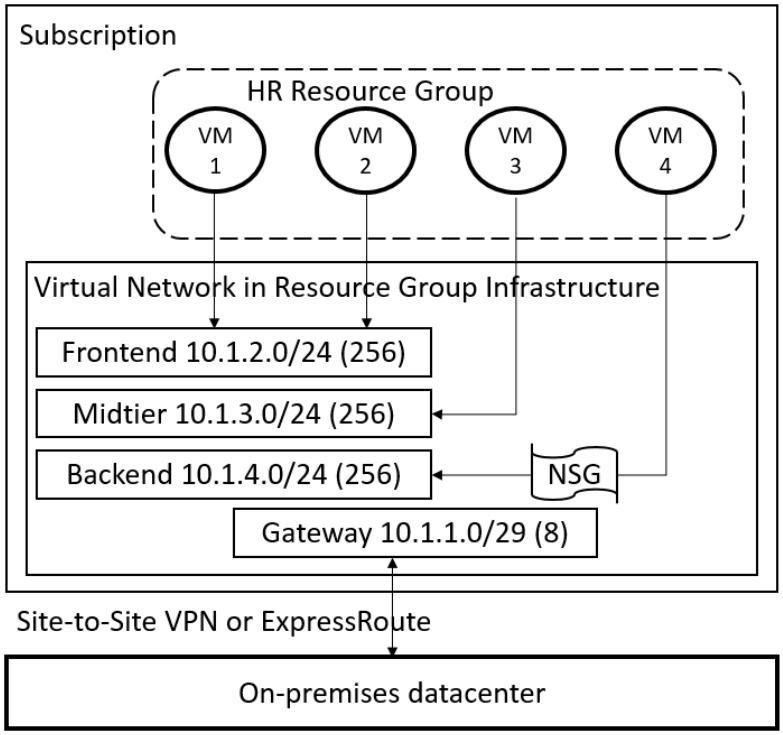


FIGURE 11. AZURE NETWORK DESIGN

Figure 10 shows an example of subnet structure. Even if some subnets may not initially be in use, they are included for future use. The purpose of each subnet in the sample network design is described here. In some scenarios, the different subnets are named after server roles, for an example database, web and system management. While naming strategies may vary, the key point is that the names should be descriptive of the subnets intended use.

- The Gateway subnet is for VPN connectivity, handled by the Azure fabric.
- The Frontend, Midtier, and Backend subnets are for different components of a service. Separating these components into different subnets allows the use of NSGs to increase security.
- The Reserved subnet is for resources with reserved IP addresses.
- The Test subnet is for temporary resources that are being used during test and development.

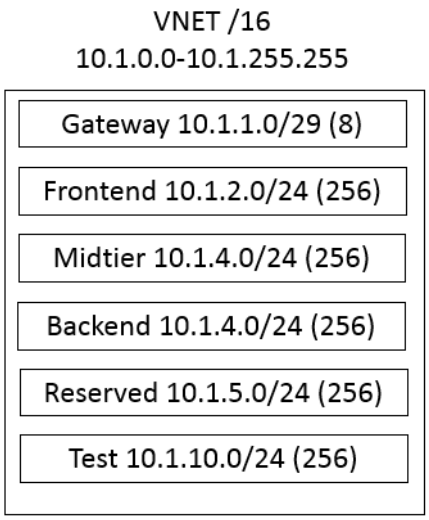


FIGURE 12. SUBNETS PER VNET

> **IMPORTANT:** Even if you created a subnet with 256 available IP addresses, the Azure fabric will use first 3 IP addresses for each subnet. The first IP address you can use starts with number 4 - For an example, 10.1.4.4 in the Backend subnet.

There are three ways to build a VNET in ARM: The Azure Portal, PowerShell and with an ARM JSON template. In the sections that follow, we will deploy the sample 3-subnet VNET described above using each of these methods. You can download the sample code for the PowerShell and JSON methods at the URLs provided in each example.

Deploying a VNET (in the Azure Portal)

Follow these steps to build one of the VNETs with the subnets shown in figure 10 in the Azure Portal.

1. Browse to the Azure Portal, https://portal.azure.com.
2. In the Azure Portal, click **NEW → Networking → Virtual Network**.

3. In the VNET blade, make sure the deployment model is set to **Resource Manager** and then click **Create**. The deployment model switch is shown in figure 11.

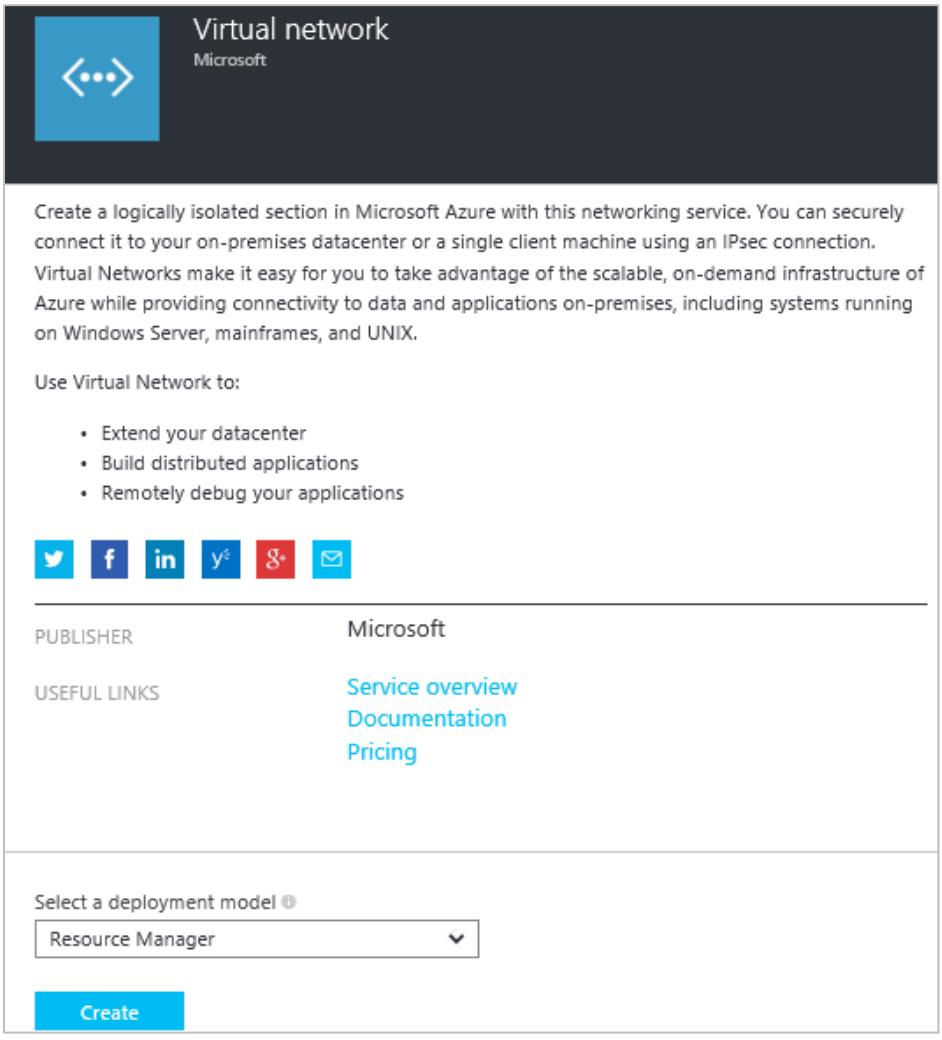


FIGURE 13. DEPLOYMENT OF NEW VNET

4. In the Create VNET blade, type in the following settings and click **Create**.
Name: Contoso-VNET01
Address space: 10.0.0.0/8
Subnet name: Frontend

Subnet address range: 10.1.2.0/24
Subscription: Choose a suitable subscription if you have more than one
Resource Group: For example, create a new resource group, CONTOSO-Infrastructure.
Location: Choose your closest location or choose a location based on design requirements

5. Once the new VNET is completed, (shown in figure 12), browse to the new VNET in the Azure Portal.

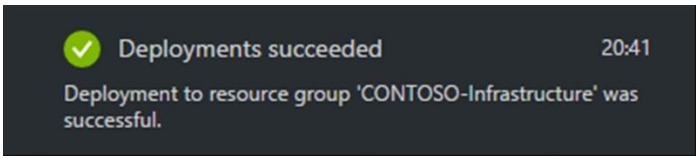


FIGURE 14. NEW VNET DEPLOYED SUCCESSFULLY

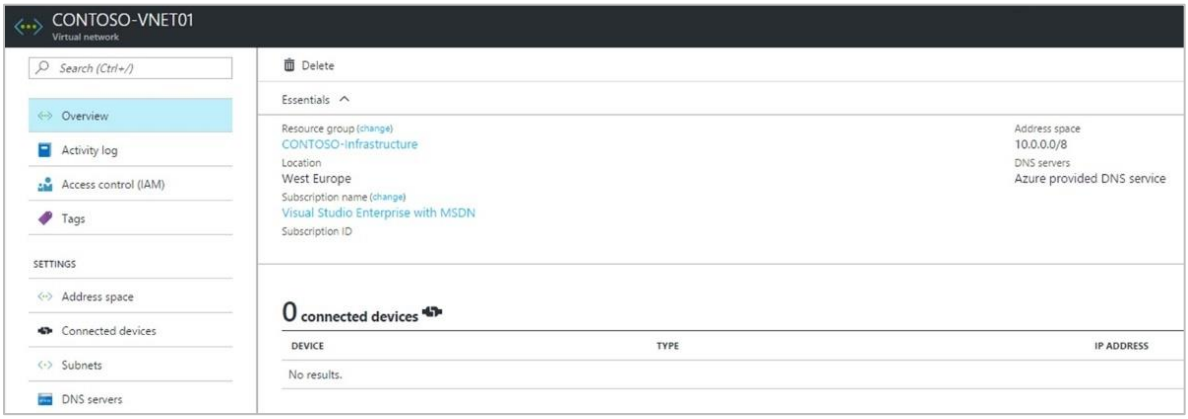


FIGURE 15. NEW VNET ON THE NEW VNET BLADE, ALL SETTINGS

6. On the Settings blade, click **Subnets**.
7. On the Subnets blade, click **Add** and add the other subnets shown in figure 10.

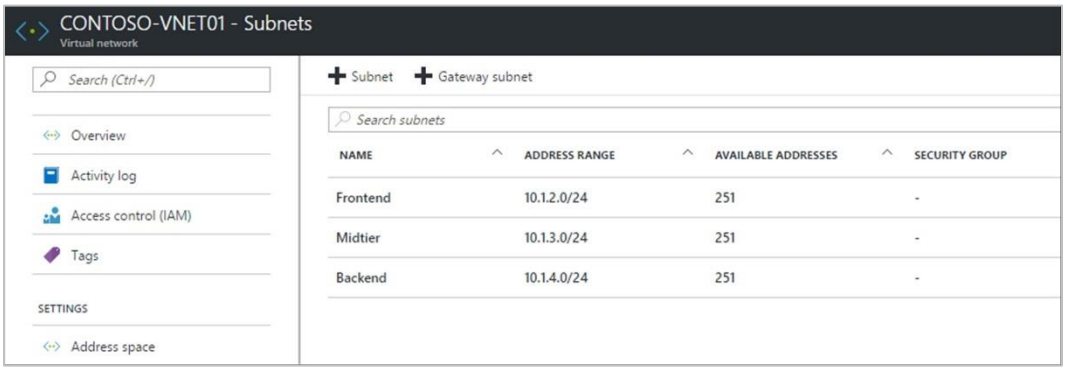


FIGURE 16. SUBNETS IN VNET

The VNET is now configured and subnets are added. The next step is to add a network security group for the backend subnet.

Deploying a VNET (with Azure PowerShell)

Rather than creating VNETs in the Azure Portal, you can deploy a VNET more quickly and consistently using Azure PowerShell with Azure Resource Manager. You can download the full script from the GitHub repository provided in the link at the end of this section.

Step 1: Connect and Authenticate

You will begin by connecting and authenticating to your Azure subscription. This process is explained in much greater detail in "Chapter 2: Azure PowerShell", but is also included here to provide a complete example. This sample will also prompt you for which Azure subscription you wish to use, easing the process for users working with multiple subscriptions.

In this example, we are using the latest Azure PowerShell module available at https://azure.microsoft.com/en-us/downloads/ to build and execute the below script.

```
# Sign-in with Azure account credentials
Login-AzureRmAccount

# Register the latest ARM Providers
Register-AzureRmResourceProvider `
    -ProviderNamespace Microsoft.Compute `
```

```
Register-AzureRmResourceProvider `
    -ProviderNamespace Microsoft.Storage `

Register-AzureRMResourceProvider `
    -ProviderNamespace Microsoft.Network `

# Confirm registered ARM Providers
Get-AzureRmResourceProvider |
     Select-Object `
        -Property ProviderNamespace `
        -ExpandProperty ResourceTypes

# Select Azure Subscription
$subscriptionId =
    (Get-AzureRmSubscription |
     Out-GridView `
        -Title "Select an Azure Subscription …" `
        -PassThru).SubscriptionId

Select-AzureRmSubscription `
    -SubscriptionId $subscriptionId
```

Step 2: Create Resource Group

If the resource group does not already exist, you will need to create a resource group.

```
# Create Resource Group
  New-AzureRmResourceGroup `
    -Name 'Contoso-Infrastructure' `
    -Location "West US"
```

Step 3: Define Subnets

Next, we will define the subnets that will exist within the VNET.

```
# Define Subnets
$frontendSubnet = New-AzureRmVirtualNetworkSubnetConfig `
    -Name Frontend -AddressPrefix 10.1.2.0/24

$midtierSubnet = New-AzureRmVirtualNetworkSubnetConfig `
    -Name Midtier -AddressPrefix 10.1.3.0/24

$backendSubnet = New-AzureRmVirtualNetworkSubnetConfig `
    -Name Backend -AddressPrefix 10.1.4.0/24
```

Step 4: Deploy VNET

In the last step, we will deploy the VNET with the three subnets defined in step 3 to a new resource group.

```
#Deploy VNET and subnets
$vnet = New-AzureRmvirtualNetwork `
   -Name Contoso-VNET01 `
   -ResourceGroupName Contoso-Infrastructure `
   -Location "West US" `
   -AddressPrefix 10.0.0.0/8 `
```

```
    -Subnet $frontendSubnet,$midtierSubnet,$backendSubnet
```

To check the results after executing the script, go to the Azure Portal, expand out the left blade, select → **Resource groups** → **Contoso-Infrastructure**.

Download the Code
You can download the full script from GitHub at https://github.com/insidemscloud/AzureIaasBook, in the **\Chapter 3** directory. The file is named **Create3SubnetVNET.ps1**.

Deploying a VNET (with JSON template)

Instead of creating each resource separately in the Azure Portal, you can use an Azure Resource Manager JSON template. The benefit with Azure Resource Manager templates is that Resource Manager will do all the heavy lifting. In a JSON-based template, you define a "state", all resources and settings you want to deploy, including dependencies between resources to control the order in which they are deployed. When compared to using a PowerShell script for deployment, deploying with Resource Manager is often the better option, particularly when deploying complex scenarios where the order of operations is critical. The benefits of ARM JSON deployment templates and the template authoring experience will be covered in greater depth in other chapters later in this book. However, a sample is provided here so you may see firsthand the power and ease of deployment that ARM JSON templates deliver.

The following example deploys a VNET and three subnets, with the same settings used earlier in the portal example. For more information about JSON templates please see *Chapter 7 - Virtual Machines*.

```
{
    "$schema":
"https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.JSON#",
    "contentVersion": "1.0.0.0",
    "parameters" : {
      "location": {
        "type": "string",
        "allowedValues": ["East US", "West US", "West
Europe", "East Asia", "South East Asia"],
        "metadata" : {
          "Description" : "Deployment location"
```

```
      }
    },
    "addressPrefix":{
      "type" : "string",
      "defaultValue" : "10.0.0.0/8",
      "metadata" : {
        "Description" : "Address prefix"
      }

    },
    "DCsubnetPrefix" : {
      "type" : "string",
      "defaultValue" : "10.1.2.0/24",
      "metadata" : {
        "Description" : "Frontend Subnet Prefix"
      }
    },
    "SCsubnetPrefix" : {
      "type" : "string",
      "defaultValue" : "10.1.3.0/24",
      "metadata" : {
        "Description" : "Midtier Subnet Prefix"
      }
    },
    "SQLsubnetPrefix" : {
      "type" : "string",
      "defaultValue" : "10.1.4.0/24",
      "metadata" : {
        "Description" : "Backend Subnet Prefix"
      }
    }
  },
  "resources": [
  {
    "apiVersion": "2015-05-01-preview",
    "type": "Microsoft.Network/virtualNetworks",
    "name": "Contoso-VNET01",
    "location": "[parameters('location')]",
    "properties": {
      "addressSpace": {
        "addressPrefixes": [
          "[parameters('addressPrefix')]"
        ]
      },
```

```
        "subnets": [
          {
            "name": "FrontEnd",
            "properties" : {
              "addressPrefix":
"[parameters('FrontendsubnetPrefix')]"
            }
          },
            {
                "name": "Midtier",
                "properties": {
                    "addressPrefix":
"[parameters('MidtiersubnetPrefix')]"
                }
            },
          {
            "name": "Backend",
            "properties" : {
              "addressPrefix":
"[parameters('BackendsubnetPrefix')]"
            }
          }
        ]
      }
    }
    ]
}
```

To deploy the VNET using the provided JSON template, perform the following steps. You will first need to download the JSON template from the GitHub repository described below.

Download the Code
You can download the JSON template from GitHub at https://github.com/insidemscloud/AzureIaasBook, in the **\Chapter 3** directory. The file name is **VNET-3subnets-azuredeploy.JSON**.

Step 1: Select Template Deployment

You will begin by selecting the 'Template Deployment' option, as detailed in the steps below.

1. Browse to the Azure Portal, https://portal.azure.com.

2. On the Azure Portal homepage, click **New**.
3. In the search box provided, type '**template deployment'** (without quotes). Search will return the template deployment option, as shown in figure 15.
4. Select template deployment and click **Create**.

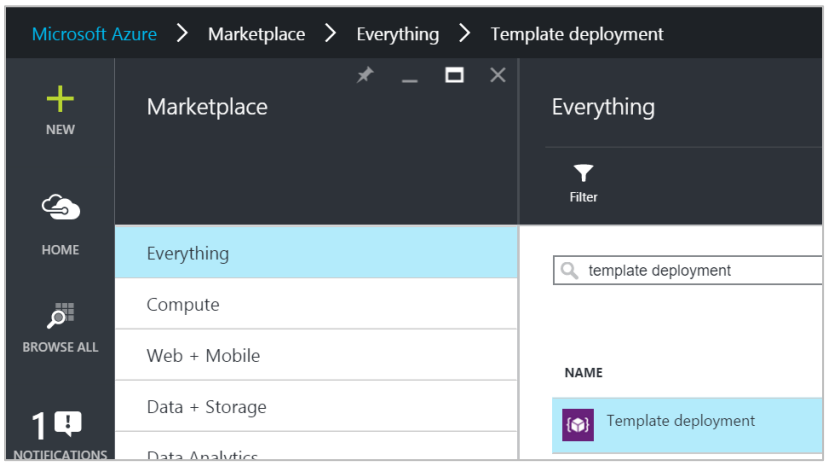


FIGURE 17. TEMPLATE DEPLOYMENT IN AZURE MARKETPLACE

Step 2: Paste JSON template into 'Edit template' window

Next, you will paste the JSON template into the space provided.

1. Select **Edit Template** (shown in figure 16) and paste the sample JSON template you downloaded into the window provided, replacing any sample JSON that may be present.
2. Click **Save** to save your changes.

Step 3: Create Resource Group

The resource group is the container for deployment.

1. If you have not already created a resource group, you can either create a new Resource Group or use an existing one, as shown in figure 16.
2. For this example, create a resource group named '**Contoso-Infrastructure'**.

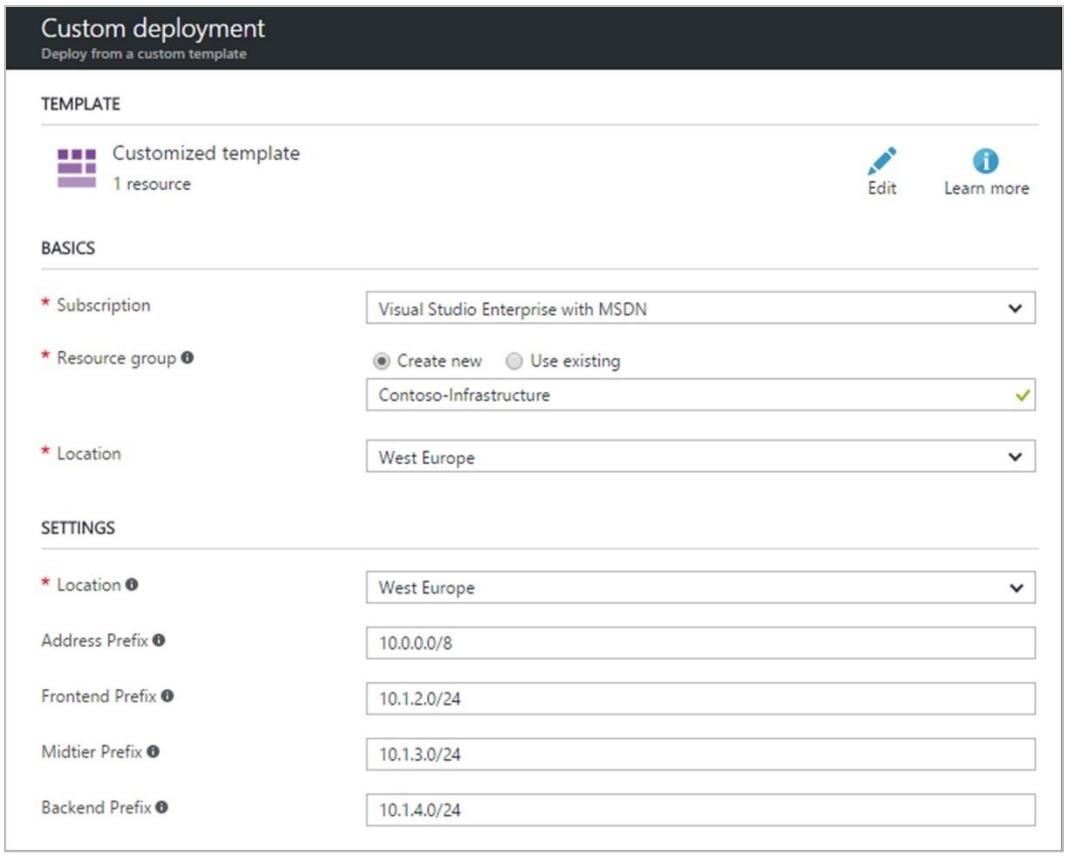


FIGURE 18. SELECT EXISTING OR CREATE NEW RESOURCE GROUP

Step 4: Fill in Parameters

Since JSON templates often have parameters, you will need to fill in any required values.

1. In the Settings section, there is only one value that is not pre-populated. Type in **West US** under location.

Step 5: Accept Terms and Deploy

Before deploying, you must review and accept the terms of the Azure Marketplace.

1. Review the **Azure Marketplace terms** section shown in figure 17 and review the terms of the template.

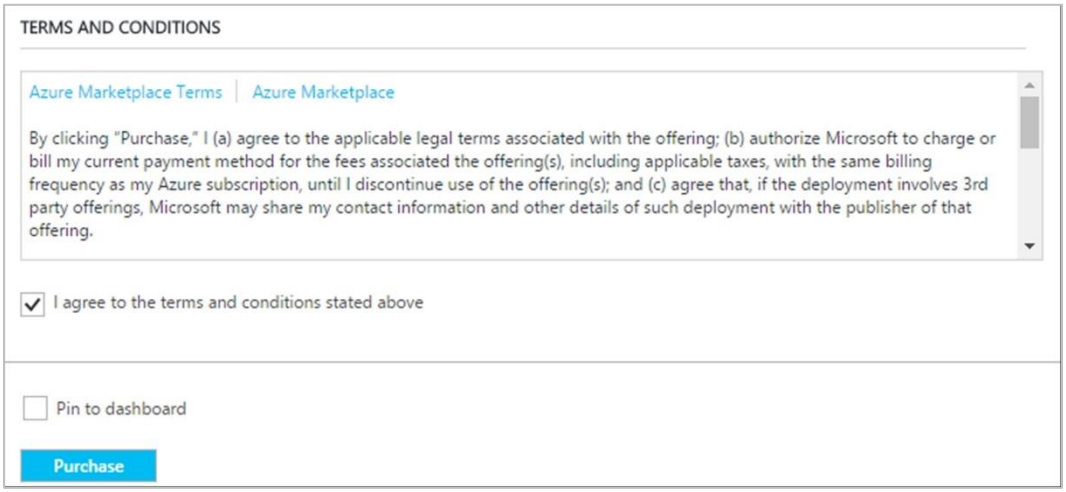


FIGURE 19. TERMS AND CONDITIONS

2. To accept, put a check mark next to **I agree to the terms and conditions stated above**. You are not actually buying anything in this case (VNETs have no cost), but once you add VMs, you do pay for compute resources by the minute.
3. Click **Create** to deploy the VNET defined in the sample JSON template. Deployment will typically complete within 5 minutes.
4. To check the results after the deployment is reported as complete in the **Notifications** section, go to the Azure Portal and select **Browse All → Resource groups → Contoso-Infrastructure**.

Configure Network Security Groups

Once the VNET and its subnets are deployed, the next step is to create appropriate NSGs. Follow these steps to configure an NSG to only allow traffic on port 1433 from the mid-tier subnet to the backend subnet (data tier).

NSGs can only be created with Azure PowerShell and REST API. Before setting up NSGs with Azure PowerShell, you need to configure a connection to your Azure subscription. Visit the following Microsoft web page for more on how to configure Azure PowerShell http://azure.microsoft.com/en-us/documentation/articles/powershell-install-configure/

1. Start **Azure PowerShell**.

2. Login to your Azure subscription.

```
Login-AzureRmAccount
```

3. Create the new network security group for the backend subnet.

```
New-AzureRmNetworkSecurityGroup -Name "Backend" `
-ResourceGroupName "Contoso-Infrastructure" `
-Location "West US"
```

4. Run the following cmdlet to list all different locations if needed.

```
Get-AzureRmLocation | FT Location
```

5. The network security group is now created. We can now manage it from the Azure Portal.

The network security group contains a number of default rules, shown in table 1. These rules cannot be deleted; they can only be superseded. All default rules are created with the lowest priority (highest number) and can be superseded by rules with higher priority (lower number).

Name	Protocol	Source Port	Dest Port	Source Address	Dest Address	Access	Priority	Direction
AllowVnetInBound	*	*	*	VNET	VNET	Allow	65000	Inbound
AllowAzureLoad BalancerInBound	*	*	*	AzureLoad Balancer	*	Allow	65001	Inbound
DenyAllInBound	*	*	*	*	*	Deny	65500	Inbound
AllowVnetOut Bound	*	*	*	VNET	VNET	Allow	65000	Outbound
AllowInternetOut Bound	*	*	*	*	Internet	Allow	65001	Outbound
DenyAllOutBound	*	*	*	*	*	Deny	65500	Outbound

TABLE 1. DEFAULT NSG RULES

Associate NSG to subnet and configure rules

To associate the network security group to the backend subnet and create a rule to allow port 1433 follow these steps.

1. Browse to the new **network security group** in the Azure Portal.
2. Click **Settings** and then **Subnets**.
3. On the Subnet associations blade, click **Associate** and select the **CONTOSO-VNET01** and the **Backend subnet,** click **OK**.
4. On the network security group settings blade, click **Inbound security rules**.
5. On the Inbound security rules blade, click **Add**.
6. Configure the new network security group rule with the settings displayed in figure 18. In order to display all options, make sure to click on the **Advanced** button up top. Once you are finished, click **OK**.

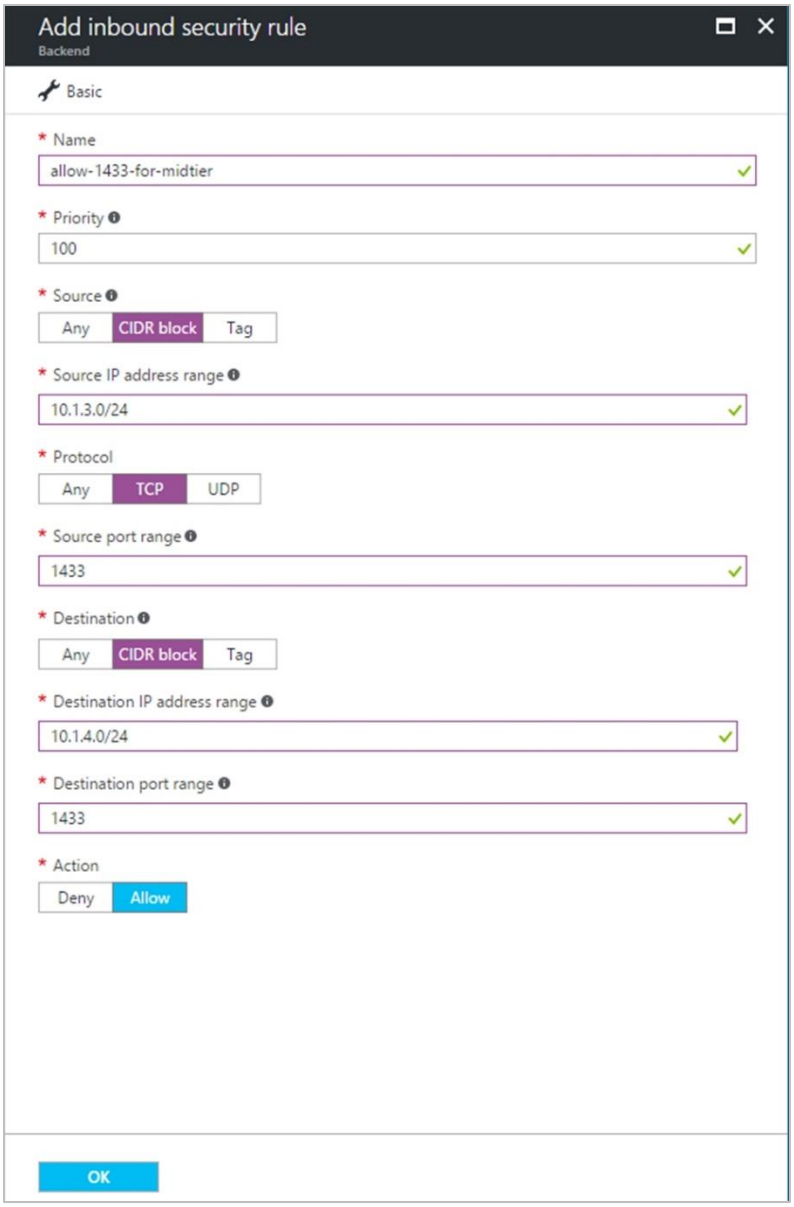


FIGURE 20. NEW NETWORK SECURITY GROUP

The new VNET is now configured, including subnets and a network security group.

Summary

Networking in Azure is a key component in every hybrid and public cloud computing infrastructure scenario. It is important to design and plan the network before starting to deploy resources and maintain strong security when extending the local network to Azure data centers.

In this chapter, we have discussed the components of Azure networking in the context of common networking scenarios, as well as we reviewed the best practices and recommendations for design, deployment, and security. We hope you have used the hands-on examples provided in this chapter to practice, as the knowledge and experience you gain here will be useful in later chapters in this book, as well as in your career as an Azure administrator.

Chapter 4: Azure Storage

Azure Storage is Microsoft's cloud storage solution for hosting IaaS and PaaS application workloads that require availability and performance to meet the needs of their customers. We will cover a number of concepts related to Azure Storage, including:

- Storage types
- Kinds of data can you store with the Azure Storage services
- How to manage access to your data
- Redundancy and replication in Azure Storage data
- How to perform a wide variety of storage-related tasks in Azure PowerShell

What is Azure Storage?

Cloud computing enables new scenarios for applications requiring scalable, durable, and highly available storage. Azure Storage provides support for a variety of application workloads, as well as the storage foundation for Azure Virtual Machines (VMs). Azure Storage is massively scalable, so you can store and process hundreds of terabytes of data to support big data scenarios, but also provides affordable, pay-for-what-you-use options for small PaaS and IaaS workloads.

Azure Premium Storage delivers high-performance, low-latency disk support for I/O intensive workloads running on Azure Virtual Machines. With Azure Premium Storage, you can attach multiple persistent data disks to a virtual machine and configure them to meet your performance requirements. Each data disk is backed by an SSD disk in Azure Premium Storage for maximum I/O performance.

Azure Storage is elastic, so you can scale your application and IaaS VM workloads as needed, both in terms of the amount of data stored and the number of requests made against it. With the new 'managed disks' feature for Azure VMs, managed disks can be scaled up on demand to support your workloads.

While Azure Storage offers a variety of storage options and capabilities for PaaS workloads, in keeping with the primary theme of this book, we will put a bit more focus on storage capabilities for IaaS workloads, specifically Azure VMs.

We will talk more about when and how to use Azure premium storage with your Azure VMs in "Chapter 6 – Azure Virtual Machines".

Azure Storage services

Azure storage provides the following four services: Blob storage, Table storage, Queue storage, and File storage.

- **Blob Storage** stores unstructured object data. A blob can be any type of text or binary data, such as a document, media file, or application installer. Blob storage is also referred to as Object storage.
- **Table Storage** stores structured datasets. Table storage is a NoSQL key-attribute data store, which allows for rapid development and fast access to large quantities of data.
- **Queue Storage** provides reliable messaging for workflow processing and for communication between components of cloud services.
- **File Storage** offers shared storage for legacy applications using the standard SMB protocol. Azure virtual machines and cloud services can share file data across application components via mounted shares, and on-premise applications can access file data in a share via the File service REST API.

An Azure **storage account** is a secure account that gives you access to services in Azure Storage. Your storage account provides a unique namespace for your storage resources, made accessible through a REST API. The image below shows the relationships between the Azure storage resources in a storage account:

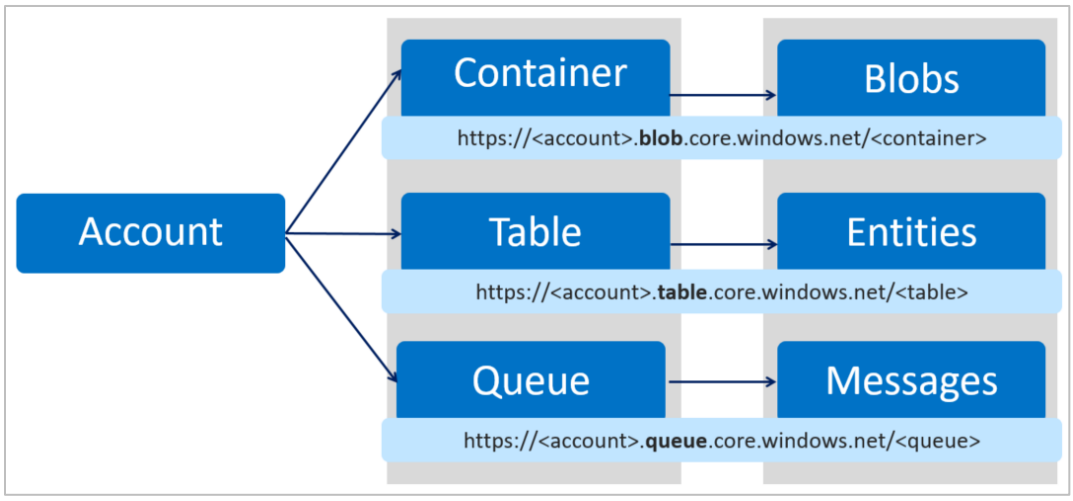


FIGURE 21. AZURE STORAGE LOGICAL ARCHITECTURE

There are two types of storage accounts:

- General-purpose storage accounts
- Blob storage accounts

We will look at each of these accounts in more detail.

General-purpose Storage Accounts

A general-purpose storage account gives you access to Azure Storage services such as tables, queues, files, blobs and Azure VM disks under a single account. This type of storage account has two performance tiers:

- A standard storage performance tier which allows you to store Tables, Queues, Files, Blobs and Azure virtual machine disks.
- A premium storage performance tier which currently only supports Azure virtual machine disks. See Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads for an in-depth overview of Premium storage.

Blob Storage Accounts

A Blob storage account is a specialized storage account for storing your unstructured data as blobs (objects) in Azure Storage. Blob storage accounts are similar to your existing general-purpose storage accounts and share all the durability, availability,

scalability, and performance features for block blobs and append blobs. For applications requiring only block or append blob storage, we recommend using Blob storage accounts.

> **Note**: Blob storage accounts support only block and append blobs, and not page blobs.

Blob storage accounts expose the **Access Tier** attribute which can be specified during account creation and modified later as needed. There are two types of access tiers that can be specified based on your data access pattern:

- A **Hot** access tier which indicates that the objects in the storage account will be more frequently accessed.
- A **Cool** access tier which indicates that the objects in the storage account will be less frequently accessed. This allows you to store data at a lower data storage cost.

If there is a change in the usage pattern of your data, you can also switch between these access tiers at any time. Changing the access tier may result in additional charges. For more information on Hot and cool storage tiers, please visit the Microsoft article "Azure Blob Storage: Hot and cool storage tiers" located at https://docs.microsoft.com/en-us/azure/storage/storage-blob-storage-tiers.

Before you can create a storage account, you must have an Azure subscription. You can create a storage account in the Azure portal, using step-by-step instructions in the "Create a storage account" section on the Microsoft website: https://docs.microsoft.com/en-us/azure/storage/storage-create-storage-account#create-a-storage-account. You can create up to 200 uniquely named storage accounts with a single subscription and up to 500 TB of data per storage account.

Storage Service Versions

The Azure Storage services are regularly updated with support for new features. The Azure Storage services REST API reference describes each supported version and its features. We recommend that you use the latest version whenever possible. For information on the latest version of the Azure Storage services, as well as information on previous versions, see "Versioning for the Azure Storage Services" at https://msdn.microsoft.com/library/azure/dd894041.aspx.

Blob storage

For users with large amounts of unstructured object data to store in the cloud, Blob storage offers a cost-effective and scalable solution. You can use Blob storage to store content such as:

- Documents
- Social data such as photos, videos, music, and blogs
- Backups of files, computers, databases, and devices
- Images and text for web applications
- Configuration data for cloud applications
- Big data, such as logs and other large datasets

Every blob is organized into a container. Containers also provide a useful way to assign security policies to groups of objects. A storage account can contain any number of containers, and a container can contain any number of blobs, up to the 500 TB capacity limit of the storage account.

Blob storage offers three types of blobs, block blobs, append blobs, and page blobs (disks).

- **Block blobs** are optimized for streaming and storing cloud objects and are a good choice for storing documents, media files, backups etc.
- **Append blobs** are similar to block blobs but are optimized for append operations. An append blob can be updated only by adding a new block to the end. Append blobs are a good choice for scenarios such as logging, where new data needs to be written only to the end of the blob.
- **Page blobs** are optimized for representing IaaS disks and supporting random writes and may be up to 1 TB in size. An Azure VM network attached IaaS disk is a VHD stored as a page blob.

For very large datasets where network constraints make uploading or downloading data to Blob storage over the wire unrealistic, you can ship a hard drive to Microsoft to import or export data directly from the data center. Read, "Use the Microsoft Azure Import/Export Service to Transfer Data to Blob Storage" for further details: https://docs.microsoft.com/en-us/azure/storage/storage-import-export-service.

Table storage

Modern applications often demand data stores with greater scalability and flexibility than previous generations of software required. Table storage offers highly available, massively scalable storage so that your application can automatically scale to meet user demand. Table storage is Microsoft's NoSQL key/attribute store, with a schemaless design, making it different from traditional relational databases. Table storage is often significantly lower in cost than traditional SQL for similar volumes of data.

Table storage is a key-attribute store, meaning that every value in a table is stored with a typed property name. The property name can be used for filtering and specifying selection criteria. A collection of properties and their values comprise an entity. Since Table storage is schemaless, two entities in the same table can contain different collections of properties, and those properties can be of different types.

You can use Table storage to store flexible datasets, such as user data for web applications, address books, device information, and any other type of metadata that your service requires. You can store any number of entities in a table, and a storage account may contain any number of tables, up to the capacity limit of the storage account.

Like other Azure storage types, you can manage and access Table storage using standard REST protocols. However, Table storage also supports a subset of the OData protocol, simplifying advanced querying capabilities and enabling both JSON and AtomPub (XML based) formats.

Queue storage

In designing applications for scale, application components are often decoupled, so that they can scale independently. Queue storage provides a reliable messaging solution for asynchronous communication between application components, whether they are running in the cloud, on the desktop, on an on-premises server, or on a mobile device. Queue storage also supports managing asynchronous tasks and building process workflows.

A storage account can contain any number of queues. A queue can contain any number of messages, up to the capacity limit of the storage account. Individual messages may be up to 64 KB in size. Review the "Get started with Azure Queue storage using .NET" article at https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-how-to-use-queues

for further information. At the top of this article, there is a drop-down menu allowing you to view the same documentation for several other programming languages.

File storage

Azure File storage offers cloud-based SMB file shares so that you can migrate legacy applications that rely on file shares to Azure quickly and without rewriting the app. With Azure File storage, applications running on Azure VMs, cloud services, or on-premise can mount a file share in the cloud, just as a desktop application mounts a typical SMB share. Any number of application components can then mount and access the File storage share simultaneously.

Since a File storage share is a standard SMB file share, applications running in Azure can access data in the share via file system I/O APIs. How you manage Files and shares depends on your skillset. Developers can leverage their existing code and skills to migrate existing applications. IT Pros can use PowerShell cmdlets to create, mount, and manage File storage shares as part of the administration of Azure applications.

Like the other Azure storage services, File storage exposes a REST API for accessing data in a share. On-premises applications can call the File storage REST API to access data in a file share, enabling legacy apps hosted on-premises to access an Azure File share.

> **Note**: Mounting a file share in Azure to an on-premise Windows computer requires that TCP port 445 outbound traffic is allowed. Depending on your environment, port 445 may be blocked by default by your I.T. Department or ISP and require you to work with them to enable it.

Distributed applications can also use File storage to store and share application data and development and testing tools. For example, an application may store configuration files and diagnostic data such as logs, metrics, and crash dumps in a File storage share so that they are available to multiple virtual machines or roles. Developers and administrators can store utilities that they need to build or manage an application in a File storage share that is available to all components, rather than installing them on every virtual machine or role instance.

Access to Blob, Table, Queue, and File resources

By default, only the storage account owner can access resources in the storage account. For the security of your data, every request made against resources in your account must be authenticated. Authentication relies on a Shared Key model. Blobs can also be configured to support anonymous authentication.

Your storage account is assigned two private access keys on creation that are used for authentication. Having two keys ensures that your application remains available when you regularly regenerate the keys as a common security key management practice.

If you do need to allow users controlled access to your storage resources, then you can create a shared access signature. A shared access signature (SAS) is a token that can be appended to a URL that enables delegated access to a storage resource. Anyone who possesses the token can access the resource it points to with the permissions it specifies, for the period of time that it is valid. Beginning with version 2015-04-05, Azure Storage supports two kinds of shared access signatures: service SAS and account SAS.

- A **service SAS** delegates access to a resource in just one of the storage services: the Blob, Queue, Table, or File service.
- An **account SAS** delegates access to resources in one or more of the storage services. You can delegate access to service-level operations that are not available with a service SAS. You can also delegate access to read, write, and delete operations on blob containers, tables, queues, and file shares that are not permitted with a service SAS.

Using SAS for Azure storage access with PowerShell is demonstrated in "Create an Azure storage context" later in this chapter.

Finally, you can specify that a container and its blobs, or a specific blob, are available for public access. When you indicate that a container or blob is public, anyone can read it anonymously; no authentication is required. Public containers and blobs are useful for exposing resources such as media and documents that are hosted on websites. For example, to decrease network latency for a global audience, you can cache blob data used by websites with the Azure CDN.

As SAS access is not a core function related to Azure VMs, we will not talk about it more here. However,

See the article "Using Shared Access Signatures (SAS)" at https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-shared-access-signature-part-1 for more information on shared access signatures. See the article "Manage anonymous read access to containers and blobs" at https://docs.microsoft.com/en-us/azure/storage/storage-manage-access-to-resources and the article "Authentication for the Azure Storage Services" at https://msdn.microsoft.com/library/azure/dd179428.aspx for more information on secure access to your storage account.

Replication for durability and high availability

The data in your Microsoft Azure storage account is always replicated to ensure durability and high availability. Replication copies your data, either within the same data center or to a second data center, depending on which replication option you choose. Replication protects your data and preserves your application up-time in the event of transient hardware failures. If your data is replicated to a second data center, that also protects your data against a catastrophic failure in the primary location. Replication ensures that your storage account meets the "Service-Level Agreement (SLA) for Storage" even in the event of failures, which you can read about here: https://azure.microsoft.com/support/legal/sla/storage/

When you create a storage account, you can select one of the following replication options:

- **Locally redundant storage (LRS).** Locally redundant storage maintains three copies of your data. LRS is replicated three times within a single data center in a single region. LRS protects your data from normal hardware failures, but not from the failure of a single data center.

  LRS is offered at a discount. For maximum durability, Microsoft recommends that you use geo-redundant storage, described below.

- **Zone-redundant storage (ZRS).** Zone-redundant storage maintains three copies of your data. ZRS is replicated three times across two to three facilities, either within a single region or across two regions, providing higher durability than LRS. ZRS ensures that your data is durable within a single region.

> **Note**: ZRS is currently available only for block blobs, and is only supported for versions 2014-02-14 and later. Once you have created your storage account and selected ZRS, you cannot convert it to use to any other type of replication or vice versa.

- **Geo-redundant storage (GRS)**. GRS maintains six copies of your data. With GRS, your data is replicated three times within the primary region and is also replicated three times in a secondary region hundreds of miles away from the primary region, providing the highest level of durability. In the event of a failure at the primary region, Azure Storage will failover to the secondary region. GRS ensures that your data is durable in two separate regions.
- **Read-access geo-redundant storage (RA-GRS)**. Read-access geo-redundant storage replicates your data to a secondary geographic location and also provides read access to your data in the secondary location. Read-access geo-redundant storage allows you to access your data from either the primary or the secondary location, in the event that one location becomes unavailable. Read-access geo-redundant storage is the default option for your storage account by default when you create it.

> **Important**: You can change how your data is replicated after your storage account has been created unless you specified ZRS when you created the account. However, note that you may incur an additional one-time data transfer cost if you switch from LRS to GRS or RA-GRS.

For architectural details about durability with Azure Storage, see the article "SSOP Paper – Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency" located at http://blogs.msdn.com/b/windowsazurestorage/archive/2011/11/20/windows-azure-storage-a-highly-available-cloud-storage-service-with-strong-consistency.aspx.

Pricing

You are billed for Azure Storage usage based on your storage account. Storage costs are based on the following factors: region/location, account type, storage capacity, replication scheme, storage transactions, and data egress.

- Region refers to the geographical region in which your account is based.

- Account type refers to whether you are using a general-purpose storage account or a Blob storage account. With a Blob storage account, the access tier also determines the billing model for the account.
- Storage capacity refers to how much of your storage account allotment you are using to store data.
- Replication determines how many copies of your data are maintained at one time, and in what locations.
- Transactions refer to all read and write operations to Azure Storage.
- Data egress refers to data transferred out of an Azure region. When the data in your storage account is accessed by an application that is not running in the same region, you are charged for data egress. (For Azure services, you can take steps to group your data and services in the same location to reduce or eliminate data egress charges.)

Make sure you are familiar with Azure storage costs, both in terms of data storage and data egress.

- The "Azure Storage Pricing" page provides detailed pricing information based on account type, storage capacity, replication, and transactions and is located at https://azure.microsoft.com/pricing/details/storage/.
- The "Data Transfers Pricing Details" page provides detailed pricing information for data egress and is located at https://azure.microsoft.com/pricing/details/data-transfers/ .

You can use the "Azure Storage Pricing Calculator" to help estimate your costs: https://azure.microsoft.com/pricing/calculator/?scenario=data-management.

Managing Azure storage with PowerShell

Azure Storage resources can be accessed by any language that can make HTTP/HTTPS requests. Additionally, Azure Storage offers programming libraries for several popular languages. These libraries simplify many aspects of working with Azure Storage by handling details such as synchronous and asynchronous invocation, batching of operations, exception management, automatic retries, operational behavior and so forth. Libraries are currently available for the following languages and platforms:

- .NET
- Java

- Node.js
- C++
- Python
- PHP
- Ruby
- iOS
- Xamarin

The following code samples demonstrating programmatic access will focus on access via Azure PowerShell.

Prerequisites for using Azure PowerShell with Azure Storage

Azure PowerShell is a module that provides cmdlets to manage Azure through Windows PowerShell. For information on installing and setting up Azure PowerShell, see "How to install and configure Azure PowerShell" at https://docs.microsoft.com/en-us/powershell/azureps-cmdlets-docs. The authors recommend that you download and install or upgrade to the latest Azure PowerShell module before proceeding.

> **Note**: You need access to an Azure subscription and an associated account to run the PowerShell cmdlets shown in the following examples.

You can run the cmdlets in the standard Windows PowerShell console or the Windows PowerShell Integrated Scripting Environment (ISE), either way, it is recommended that you run either environment as Administrator.

Once you have an elevated Powershell prompt open, to see all the available PowerShell cmdlets for Azure Storage in ARM, run:

```
Get-Command -Module AzureRm.Storage
```

Authenticate and select storage account

In an elevated Powershell prompt, type the following command to add your Azure account to the local PowerShell environment:

```
Add-AzureRmAccount
```

In the “Sign into your account” window, type the email address and password associated with your Azure Subscription. Azure authenticates and saves the credential information, and then closes the window.

Next, run the following command to view the Azure Subscriptions currently associated with the credentials you used to log in.

```
Get-AzureRmSubscription
```

Next, run the following command to associate to the Azure Subscription you want to work with.

```
Select-AzureRmSubscription -SubscriptionId "<SUBSCRIPTION_ID>"
```

Create a new Azure storage account

Run the Get-AzureRmLocation cmdlet to find all the available data center locations:

```
Get-AzureRmLocation | Select Location, DisplayName
```

Run the following command to create a new Resource Group where the Azure storage account will reside.

```
New-AzureRmResourceGroup -Name my-store-account -Location "westus"
```

Next, run the New-AzureRmStorageAccount cmdlet to create a new storage account. The following example creates a new storage account in the "West US" datacenter.

```
New-AzureRmStorageAccount `
    -ResourceGroupName my-store-account `
    -Name "mystracctchap4" `
    -Location "westus" `
    -SkuName Standard_LRS
```

The storage account should finish being deployed in about a minute.

> **Important**: The name of your storage account must be unique within Azure and must be lowercase. For naming conventions and restrictions, see the “About Azure Storage Accounts” article located at https://docs.microsoft.com/en-us/azure/storage/storage-create-storage-account.
>
> Additional Storage related naming guidelines can be found in the “Naming and Referencing Containers, Blobs, and Metadata” article located at https://docs.microsoft.com/en-

us/rest/api/storageservices/fileservices/Naming-and-Referencing-Containers--Blobs--and-Metadata.

List all Azure storage accounts in a subscription

To list all the Azure storage accounts in your subscription, run the following command:

```
Get-AzureRmStorageAccount | `
Format-Table StorageAccountName, ResourceGroupName, PrimaryLocation, `
StatusOfPrimary, StatusOfSecondary
```

Create an Azure storage context

An Azure storage context is an object in PowerShell that encapsulates the storage credentials. Using a storage context while running any subsequent cmdlet enables you to authenticate your request without specifying the storage account and its access key explicitly. You can create a storage context by running the following command

```
$Context = (Get-AzureRmStorageAccount `
    -ResourceGroupName "my-store-account" `
    -Name "mystracctchap4").Context
```

Retrieve and generate Azure storage keys

An Azure Storage account comes with two account keys. You can use the following Get-AzureRmStorageAccountKey cmdlet to retrieve your keys and store them in a variable.

```
$StorageAccountKeys = Get-AzureRmStorageAccountKey `
-ResourceGroupName "my-store-account"
-Name "mystracctchap4"
```

As the Get-AzureRmStorageAccountKey cmdlet returns the two account keys in a List Object, you can reference the value of the primary and secondary keys as shown below.

```
$StorageAccountKeys[0].value
$StorageAccountKeys[1].value
```

If you would like to regenerate your keys, use the New-AzureRmStorageAccountKey cmdlet. Valid values for -KeyName are "key1" and "key2"

```
New-AzureRmStorageAccountKey `
    -ResourceGroupName "my-store-account" `
    -Name "mystracctchap4" `
    -KeyName key1
```

Manage Azure blobs

Azure Blob storage is a service for storing large amounts of unstructured data, such as text or binary data, that can be accessed from anywhere in the world via HTTP or HTTPS. This section assumes that you are already familiar with the Azure Blob Storage Service concepts. For detailed information, see the "Get started with Blob storage using .NET" article located at https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-how-to-use-blobs and the "Blob Service Concepts" article located at https://docs.microsoft.com/de-de/rest/api/storageservices/fileservices/Blob-Service-Concepts.

Create a container

Every blob in Azure storage must be in a container. You can create a private container using the New-AzureStorageContainer cmdlet:

```
New-AzureStorageContainer `
    -Name "mycontentschap4" `
    -Permission Off `
    -Context $Context
```

There are three levels of anonymous read access: **Off**, **Blob**, and **Container**. To prevent anonymous access to blobs, set the Permission parameter to **Off**. By default, the new container is private and can be accessed only by the account owner. To allow anonymous public read access to blob resources, but not to container metadata or to the list of blobs in the container, set the Permission parameter to **Blob**. To allow full public read access to blob resources, container metadata, and the list of blobs in the container, set the Permission parameter to **Container**.

Upload a blob into a container

To upload blobs into a container, you can use the Set-AzureStorageBlobContent cmdlet. By default, this command uploads the local files to a block blob. To specify the type for the blob, you can use the -BlobType parameter. Note that the **-Path** value here could be something else like **C:\Windows\Temp*.**

```
Get-ChildItem -Path C:\Screenshots\* `
    | Set-AzureStorageBlobContent `
    -Context $Context `
    -Container "mycontentschap4"
```

Copy blobs between containers

You can copy blobs across storage accounts and regions asynchronously. The following example demonstrates how to copy blobs from one storage container to another in two different storage accounts.

Start off by creating a new Resource Group to store the secondary storage account.

```
New-AzureRmResourceGroup -Name my-backup-store-account -Location "eastus"
```

Next, create the secondary storage account.

```
New-AzureRmStorageAccount `
    -ResourceGroupName my-backup-store-account `
    -Name "mysecondstrchap4" `
    -Location "eastus" `
    -SkuName Standard_LRS
```

Next, retrieve the context of the secondary storage account.

```
$SecondaryContext = (Get-AzureRmStorageAccount `
    -ResourceGroupName "my-backup-store-account" `
    -Name "mysecondstrchap4").Context
```

Next, create a container in the secondary storage account.

```
New-AzureStorageContainer `
    -Name "mysecondarychap4" `
    -Permission Off `
    -Context $SecondaryContext
```

Next, retrieve the blob contents of the container of the first storage account.

```
$Blob = Get-AzureStorageBlob `
    -Context $Context
    -Container "mycontentschap4"
```

Finally, start the copy of the contents of the main storage account to the secondary storage account.

```
$Blob | Start-AzureStorageBlobCopy `
    -DestContainer "mysecondarychap4" `
    -DestContext $SecondaryContext
```

> **Note**: This example performs an asynchronous copy. You can monitor the status of each copy by using the **`Get-AzureStorageBlobCopyState`** cmdlet.

Delete a blob

To delete the blob we just uploaded, first retrieve the blob from the container using the Get-AzureStorageBlob cmdlet.

```
$Blob = Get-AzureStorageBlob `
    -Context $Context `
    -Container "mycontentschap4"
```

Next, use the Remove-AzureStorageBlob cmdlet to remove the blob.

```
Remove-AzureStorageBlob `
    -Context $Context `
    -Blob $Blob.Name `
    -Container "mycontentschap4"
```

On order to remove multiple entities from a container, you need to encapsulate the Remove-AzureStorageBlob cmdlet in a ForEach loop and then filter accordingly. In the example below, we are removing all entries based on the name of each Blob.

```
ForEach ($Item in $Blob)
{
    Remove-AzureStorageBlob `
        -Context $Context `
        -Blob $Item.Name `
        -Container "mycontentschap4"
}
```

How to manage Azure blob snapshots

Azure lets you create a snapshot of a blob. A snapshot is a read-only version of a blob that's taken 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. For more information, see "Creating a Snapshot of a Blob" located at http://msdn.microsoft.com/library/azure/hh488361.aspx.

How to create a blob snapshot

To create a snapshot of a blob, first get a blob reference and then call the `ICloudBlob.CreateSnapshot` method on it. Since we already have the context and container information available to work with from the original storage account that

was created, we can retrieve a single blob entry from the container. In the example below, replace **Screenshot001.jpg** with the name of the file you want to create a snapshot of.

```
$Blob = Get-AzureStorageBlob `
    -Context $Context
    -Container "mycontentschap4" `
    -Blob "ScreenShot001.jpg"
```

Next, use the **ICloudBlob.CreateShapshot** method to create a snapshot.

```
$Blob.ICloudBlob.CreateSnapshot()
```

List a blob's snapshots

You can create as many snapshots as you want for a blob. You can list the snapshots associated with your blob to track your current snapshots. The following example uses the Get-AzureStorageBlob cmdlet and filters on the name of the blob and if it is a snapshot from the original storage account created earlier.

```
Get-AzureStorageBlob `
    -Context $Context `
    -Prefix "ScreenShot001.jpg" `
    -Container "mycontentschap4" `
    | Where-Object {($_.ICloudBlob.IsSnapshot) -and `
    ($_.Name -eq "ScreenShot001.jpg")}
```

Copy a snapshot of a blob

It is possible to copy a snapshot of a blob to a secondary location.

Start by retrieving a blob from the original storage account we created earlier,

```
$Blob = Get-AzureStorageBlob `
    -Context $Context
    -Container "mycontentschap4" `
    -Blob "ScreenShot001.jpg"
```

Next, create a new snapshot.

```
$Snapshot = $Blob.ICloudBlob.CreateSnapshot()
```

Finally, copy the snapshot from the primary storage account to the secondary storage account that was created earlier.

```
Start-AzureStorageBlobCopy `
    -Context $Context `
    -ICloudBlob $Snapshot `
    -DestBlob "ScreenShot001-Secondary.jpg" `
    -DestContainer "mysecondarychap4" `
    -DestContext $SecondaryContext
```

Azure File Storage

Azure File storage is a service that offers file shares in the cloud using the standard Server Message Block (SMB) Protocol. Both SMB 2.1 and SMB 3.0 are supported. With Azure File storage, you can migrate legacy applications that rely on file shares to Azure.

Since a File storage share is a standard SMB file share, applications running in Azure can access data in the share via file system I/O APIs. You can use PowerShell cmdlets to create, mount, and manage File storage shares as part of the administration of Azure applications.

You can create Azure file shares using Azure Portal, the Azure Storage PowerShell cmdlets, the Azure Storage client libraries, or the Azure Storage REST API. Additionally, because these file shares are SMB shares, you can access them via standard and familiar file system APIs.

Common uses of File storage include:

- Migrating on-premises applications that rely on file shares to run on Azure virtual machines or cloud services, without expensive rewrites
- Storing shared application settings, for example in configuration files
- Storing diagnostic data such as logs, metrics, and crash dumps in a shared location
- Storing tools and utilities needed for developing or administering Azure virtual machines or cloud services

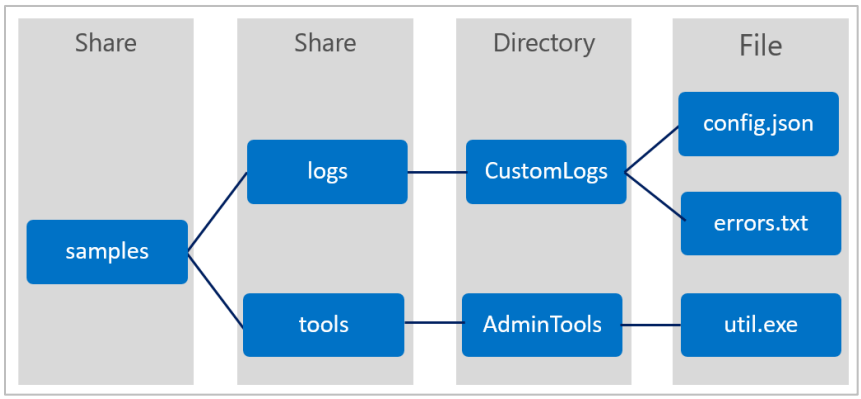


FIGURE 22. AZURE FILE STORAGE COMPONENTS

- **Storage Account:** All access to Azure Storage is brokered through a storage account. At the time this was written, each storage account can store up to 500TB. For updated figures, check "Azure Storage Scalability and Performance Targets" on the Microsoft website at https://docs.microsoft.com/en-us/azure/storage/storage-scalability-targets
- **Share:** A File storage share is an SMB file share in Azure. All directories and files must be created in a parent share, which is hosted in a storage account. A single Azure storage account can contain an unlimited number of shares, and a share can store an unlimited number of files, up to the capacity of the file share, which at the time of this writing, is 5TB.
- **Directory:** Optionally, you can create a hierarchy of directories to organize your files
- **File:** A file is a file in the share. A single file may be up to 1TB in size.
- **URL format:** Files are addressable using the following URL format:

```
https://<storage
account>.file.core.windows.net/<share>/<directory/directories>/<file>
```

The following example URL could be used to address one of the files in the diagram above:

```
http://samples.file.core.windows.net/logs/CustomLogs/config.json
```

Naming file storage entities

There are some rules to guide your file and share naming conventions. For more info, see "Naming and Referencing Shares, Directories, Files and Metadata" on the Microsoft website at http://msdn.microsoft.com/library/azure/dn167011.aspx

Securing and Sharing Azure Storage

Shared access signatures are an important part of the security model for any application using Azure Storage. They are useful for providing limited permissions to your storage account to clients that should not have the account key. By default, only the owner of the storage account may access blobs, tables, and queues within that account. If your service or application needs to make these resources available to other clients without sharing your access key, you have three options:

- Set a container's permissions to permit anonymous read access to the container and its blobs. This is not allowed for tables or queues.
- Use a shared access signature that grants restricted access rights to containers, blobs, queues, and tables for a specific time interval.
- Use a stored access policy to obtain an additional level of control over shared access signatures for a container or its blobs, for a queue, or for a table. The stored access policy allows you to change the start time, expiry time, or permissions for a signature, or to revoke it after it has been issued.

A shared access signature can be in one of two forms:

- **Ad hoc SAS**: When you create an ad hoc SAS, the start time, expiry time, and permissions for the SAS are all specified on the SAS URI. This type of SAS may be created in a container, blob, table, or queue and it is non-revocable.
- **SAS with stored access policy**: A stored access policy is defined on a resource container a blob container, table, or queue - and you can use it to manage constraints for one or more shared access signatures. When you associate a SAS with a stored access policy, the SAS inherits the constraints - the start time, expiry time, and permissions - defined for the stored access policy. This type of SAS is revocable.

Create a Shared Access Signature token for a container

Run the following command to create a shared access signature token to use to access the container from the original storage account.

```
New-AzureStorageContainerSASToken `
    -Context $Context `
    -Name "mycontentschap4" `
    -Permission rwdl `
    -FullUri
```

The Full URI of the SAS token will appear and look something like what is shown below:

```
https://mystracctchap4.blob.core.windows.net/mycontentschap4?sv=2015-04-
05&sr=c&sig=b30bqV62VUHJqKtDg7zwadAVYeN3qoUwuR4UCMxxdFA%3D&se=2017-03-
29T20%3A28%3A26Z&sp=rwdl
```

In order to use this token to access blobs (files) in the **mycontentschap4** container, type in the name of the blob with a "**/**" after the name of the container and before the "**?sv=**" in the URI. An example is shown below for a blob called **Screenshot001.jpg**.

```
https://mystracctchap4.blob.core.windows.net/mycontentschap4/ScreenShot001.jpg?sv=
2015-04-05&sr=c&sig=b30bqV62VUHJqKtDg7zwadAVYeN3qoUwuR4UCMxxdFA%3D&se=2017-03-
29T20%3A28%3A26Z&sp=rwdl
```

Note that you can also easily create your own SAS Tokens for Blobs, Files, Tables, and Queues directly in the Azure Portal in the Storage Account you want to create them for, under the Shared access signature blade.

For additional guidance on SAS Tokens, review the "Using Shared Access Signatures (SAS)" at https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-shared-access-signature-part-1. For additional information on managing SAS Tokens using policies, please review the "Establishing a Stored Access Policy" at https://docs.microsoft.com/en-us/rest/api/storageservices/fileservices/establishing-a-stored-access-policy.

Summary

Azure storage is a foundational component not only in your Azure IaaS VM strategy, but can also be useful in delivering familiar services you use on-premises today, such as SMB file shares. With the new Premium storage tier backed by SSD drives, you can run workloads with disk-intensive workloads.

In this chapter, we have discussed the components of Azure storage in the context of Azure IaaS workloads, as well as automating common storage-specific operations that may augment your Azure IaaS VM hosting strategy. We hope you give the code samples a try, as the knowledge and experience you gain here will be useful in your career as an Azure administrator.

Chapter 5: Connecting Azure to Your Data Center

Microsoft Azure Infrastructure-as-a-Service (IaaS) allows the creation of publicly hosted, cloud-based virtual machines. With Microsoft Azure virtual network capabilities, you can implement hybrid networked environments for the purposes of development, user acceptance testing (UAT), quality assurance (QA) and production services with nearly limitless capacity. In recent months, Microsoft has made it even easier to connect your corporate datacenter to Microsoft Azure regional data centers in a cost-effective fashion, enabling a variety of hybrid cloud scenarios.

In this chapter, we will focus on explaining options available to connect your corporate datacenters to your Azure virtual networks and public services, which options are best suited to particular scenarios, as well as pricing models for each option.

At the conclusion of this chapter, you will have a good understanding of the hybrid connectivity options between your corporate datacenter and Microsoft Azure, capabilities of each, as well as some of the common use cases where each option fits. Your most important takeaways from this chapter are a clear understanding of your hybrid connectivity options and the decision criteria that will enable you to choose the best option(s) for your organization.

As the step-by-step processes for various connectivity options will vary based on your situation and chosen hybrid connectivity option, we will provide high-level guidance along with links to the best step-by-step configuration guidance available at the time of this writing to ensure you are successful in your efforts to implementing hybrid connectivity to Microsoft Azure for your organization.

5.1 Hybrid Connectivity Options

Microsoft Azure offers multiple hybrid connectivity options to connect your corporate data center and applications to your Azure subscriptions. The most effective option (in terms of cost and performance) for your organization depends on the use case. Hybrid connectivity options to connect your Azure IaaS environment to your on-premises datacenter and applications include:

- Point-to-Site VPN (P2S)

- Site-to-Site VPN (S2S)
- ExpressRoute

The target customers and use cases for each of these options are pictured in figure 1. We will explore these options in greater detail in the sections of this chapter that follow.

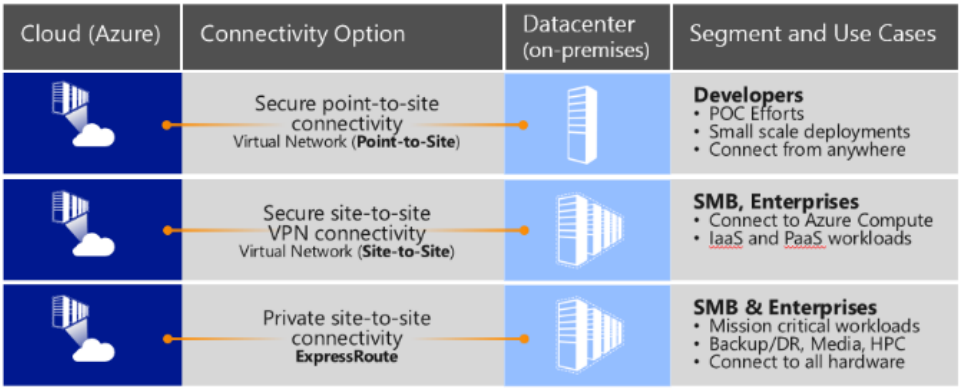


FIGURE 23. AZURE HYBRID CONNECTIVITY OPTIONS, CUSTOMER, AND WORKLOADS

Connectivity between Microsoft Azure VMs and your corporate datacenter begins with an Azure virtual network. Azure virtual networks enable you to create a logically isolated network in Azure and securely connect it to your on-premises datacenter (via a site-to-site VPN or ExpressRoute) or a single client machine using an SSTP connection (via a point-to-site VPN).

You can learn about Azure virtual networks in depth in "Chapter 3 – Azure Virtual Networking" in this book. Now, we will explore Azure hybrid connectivity options in depth.

> Microsoft Azure Virtual Networks provides User Defined Routing and Forced Tunneling capabilities for hybrid network connections. Please refer to Chapter 3 - Azure Virtual Networking for more information.

Point-to-Site VPN

In some cases, it might not be necessary to have the entire corporate network be connected to Azure, but simply client-based remote access via VPN to Microsoft Azure virtual networks for the purposes of providing remote administration and support, small

branch server connectivity, or perhaps for a developer that requires access only to a single, isolated virtual machine running in Microsoft Azure. For these scenarios, Microsoft Azure supports point-to-site VPN.

The point-to-site VPN option utilizes secure socket tunneling protocol (SSTP), which means that VPN connectivity will typically work through firewalls and NAT devices without the need for any additional configuration. (The SSTP protocol utilizes SSL over TCP port 443.) This eliminates the need for dedicated or public IP addresses for clients connecting via the P2S VPN option, though you do need to provide an IP address range from which clients may connect. The P2S and S2S VPN connections can coexist without issue, as illustrated in figure 2.

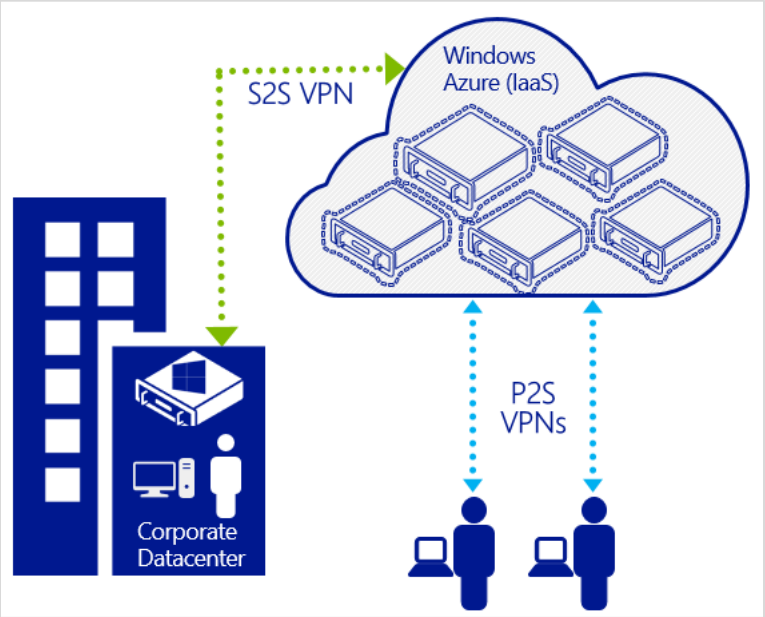


FIGURE 24. CLIENT CONNECTIVITY TO AZURE WITH POINT-TO-SITE VPN

P2S VPN connectivity requires no software installation. Client connectivity is established through a simple SSTP VPN connection using the native Windows client. P2S VPN connectivity is designed for client connectivity to Azure Compute resources only.

What you will need

There are four areas of planning needed before you begin to configure a P2S connection:

- IP address space for the virtual network and subnets (in Azure)
- One or more DNS servers
- Client-side address space (to be assigned to clients connecting via P2S VPN)
- Certificates to support client authentication

When provisioning a new Microsoft Azure virtual machine, make sure to select the correct virtual network and virtual subnet defined in your P2S configuration. Microsoft Azure virtual machines deployed on this virtual network will now be accessible using point-to-site remote access VPN.

Microsoft Azure point-to-site VPN networking relies on client certificates for authentication. There are two options to generate these certificates; generate a self-signed certificate or use a certificate issued from your internal Public Key Infrastructure (PKI). There are some distinct advantages to using an internal PKI to issue certificates for Microsoft Azure, but having a PKI simply to issue certificates for a few users who need access to Azure VMs may seem like an overkill.

Viewing active P2S VPN connections

Active P2S VPN Connections are viewable in the Azure Portal in the virtual network gateway blade as shown in figure 3 below.

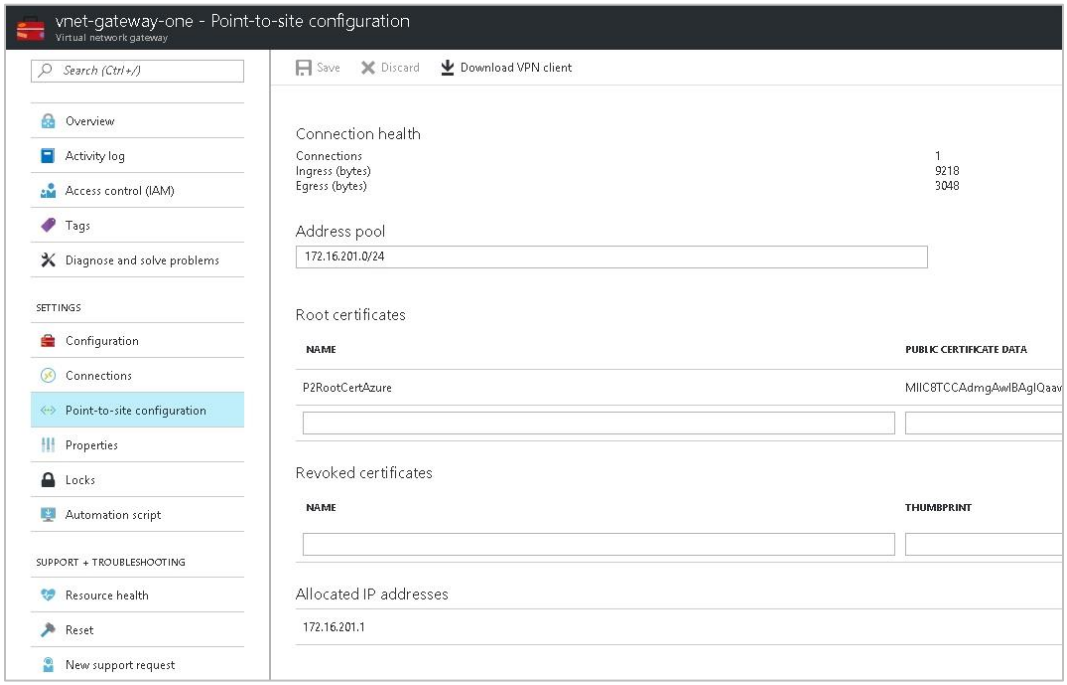


FIGURE 25. POINT-TO-SITE VPN SETTINGS

Step-by-Step

The high-level steps for configuring an Azure P2S VPN include:

1. Configure a virtual network and dynamic routing gateway in Azure
2. Create your certificates
3. Configure your VPN client

Step-by-step instructions for configuring a point-to-site VPN connection between an Azure datacenter and your corporate datacenter is available on the Microsoft website in *Configure a Point-to-Site connection to a VNet using the Azure Portal* at https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-point-to-site-resource-manager-portal. Additionally, when you visit this link, you'll be presented with two additional deployment at the top of the article in a drop-down menu: **Resource Manager - PowerShell** and **Classic - Azure Portal**. You should only use the Classic deployment option for legacy deployments in Azure and use Resource Manager for new Azure deployments.

Site-to-Site VPN

A secure site-to-site VPN connection can be used to create a branch office solution or whenever you want a secure connection between your on-premises network and your Azure virtual network. Site-to-site connections require a public-facing IPv4 IP address and a compatible VPN device or Routing and Remote Access Services (RRAS) running on Windows Server 2012R2/2016.

The Azure site-to-site VPN option, shown in figure 4, is an IPsec VPN, so unlike the point-to-site VPN option, Azure site-to-site VPNs cannot be established through firewalls or NAT devices.

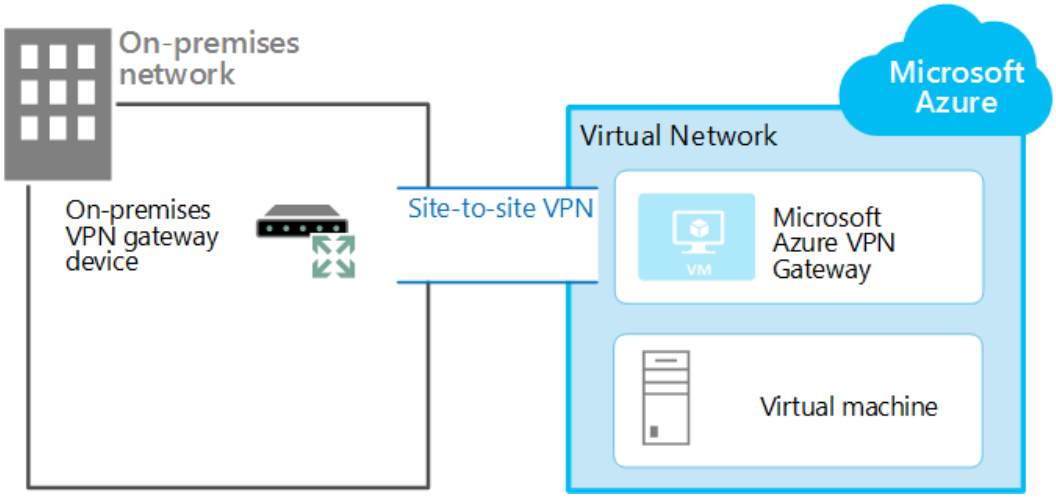


FIGURE 26. CONNECTING CORPORATE NETWORKS TO AZURE WITH SITE-TO-SITE VPN

The site-to-site option is intended for more permanent connections, as opposed to the often ad hoc or temporary client connectivity scenarios common with the point-to-site option. Same as a P2S VPN, S2S VPN connectivity is designed for client connectivity to Azure Compute resources only.

What you will need

Before you attempt to configure a S2S VPN connection, there are a couple of things you need to address:

- First, verify that the VPN device that you want to use meets the requirements necessary to create a cross-premises virtual network connection. Azure supports a broad range of device manufacturers, including Cisco, Juniper, and several others. See the *About VPN devices and IPsec/IKE parameters for Site-to-Site VPN Gateway connections* page on the Microsoft Azure Documentation at https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/ for a detailed list of supported devices.

- Next, you will need an externally facing IPv4 IP for your VPN device. This IP address is required for a S2S VPN configuration and is used for your VPN device, which cannot be located behind a NAT.

- Same as the P2S VPN, the S2S VPN also requires certificates for authentication.

You have two options for creating and configuring a virtual network, using the Azure Portal or Azure PowerShell.

> If you need to use the classic deployment model (ASM) for creating a Site-to-Site VPN connection, you may be required to export the network configuration file to retrieve the values of the Local Network Site Name and Virtual Network Site Name required to complete the connection. Additional instructions can be found in Microsoft's official documentation: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-classic-portal

Step-by-Step

The high-level steps for configuring an Azure site-to-site 2S VPN will vary slightly based on your VPN device.

Step-by-step instructions for configuring a site-to-site VPN connection between an Azure datacenter and your corporate datacenter is available on the Microsoft website in *Create a VNet with a Site-to-Site VPN connection using PowerShell* at https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-create-site-to-site-rm-powershell/

> The Above step-by-step guide covers the Azure Resource Manager model of deployment for S2S VPN. If you're deploying a S2S VPN in the Azure Service Management Portal, please refer to *Create a VNet with a Site-to-Site connection using the classic portal* at https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-site-to-site-create/

Multi-Site VPN

You can create an Azure multi-site VPN in order to connect multiple on-premises sites to a single virtual network gateway. Creating a multi-site VPN connection is very similar to creating other S2S connections. In fact, you can use an existing S2S connection, provided you have dynamic routing configured for your virtual network gateway. If your gateway is static routing (Policy Based), you can always change the gateway type without needing to rebuild the virtual network to accommodate multi-site, although you will also need to ensure your on-premises VPN gateway supports dynamic routing VPN. You will then just

add configuration settings to the network configuration file, and create multiple VPN connections from your virtual network to additional sites. An example of multi-site VPN connectivity to azure is shown in figure 5.

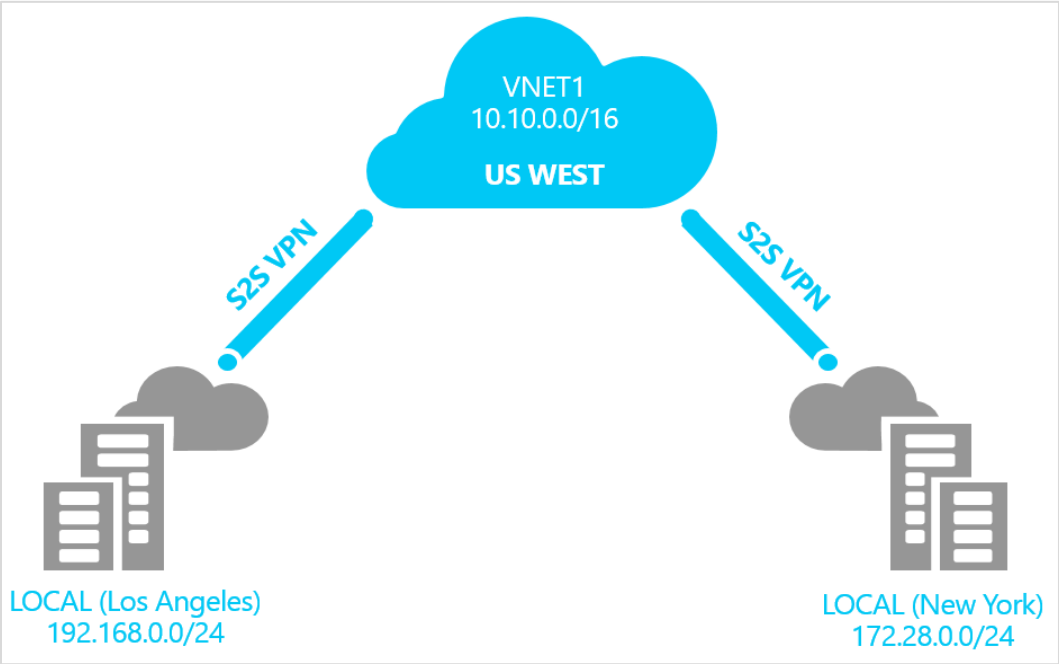


FIGURE 27. CONNECTING MULTIPLE CORPORATE NETWORKS TO AZURE (MULTI-SITE VPN)

The primary scenario where Azure multi-site VPN comes into play is in the enterprise with multiple physical sites with workloads running in Azure VMs (IaaS) that must be accessible to users and/or applications from multiple locations.

> If you are required to create a multi Site-to-Site VPN using the classic model (ASM), you cannot use the management portal to make changes to this virtual network. For this scenario, you must make changes to the network configuration file instead of using the management portal. If you make changes in the management portal, you will overwrite your multi-site reference settings for this virtual network. While the configuration file method can seem foreign at first, but you will feel comfortable using the network configuration file by the time you have completed your first multi-site configuration. Additionally, if you have multiple people working on the network configuration, make sure everyone is aware of this

limitation. Additional documentation can be found at Microsoft's official documentation: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-multi-site

What you will need

There are a few items that you will need to take care of before you attempt to configure your multi-site VPN.

- Compatible VPN hardware for each on-premises location. See the *About VPN devices and IPsec/IKE parameters for Site-to-Site VPN Gateway connections* page on the Microsoft Azure Documentation at https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/ for a detailed list of supported devices.

- An externally facing public IPv4 IP address for each VPN device. Multi-site VPN is really just multiple site-to-site VPNs, which is based on IPsec, so this is a requirement. The IP address cannot be located behind a NAT device.

- Windows PowerShell, updated with the latest Azure modules.

- Someone who knows how to configure your VPN device. This means you will need to have a good understanding of how to configure your VPN device, good instructions or a network engineer to assist with this part of the process.

- The IP address ranges that you want to use for your virtual network. If you haven't yet created your virtual network, you'll need to be aware of the rules related to IP addresses and address ranges in Azure virtual networks. See the Virtual Network Address Spaces page for more information. Review *Chapter 3 - Azure Virtual Networking* if you have not already.

You will need to ensure the IP address ranges for each of the local network sites that you want to connect to Azure do not overlap. Otherwise, the Azure Portal or the REST API (if you are using the Azure PowerShell cmdlets) will reject the configuration.

To prevent routing issues, Azure will not allow you to configure overlapping IP address ranges. For example, if you have two local network sites both containing the IP address range 10.1.2.0/24 and you have a payload with a destination address 10.1.2.4, Azure

would not know which site the payload is intended for because the address ranges are overlapping. Thus, Azure will prevent you from making this mistake.

Step-by-Step

The high-level steps for configuring an Azure multi-site 2S VPN include:

1. Create a site-to-site VPN with a dynamic routing gateway.
2. Add a new site-to-site connection
3. Add a new local network gateway
4. Add the Shared Key to the VPN Device and site-to-site connection
5. Verify the status of the connection.

Step-by-step instructions for configuring a multi-site VPN connection between Azure regional datacenters and your corporate data centers is available on the Microsoft website in *Add a Site-to-Site connection to a VNet with an existing VPN gateway connection* at https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-multi-site-to-site-resource-manager-portal.

VNET-to-VNET Connectivity

Connecting an Azure virtual network (VNET) to another Azure virtual network is very similar to connecting a virtual network to an on-premises site location. Both types of connectivity use a virtual network gateway to provide a secure tunnel using IPsec. The VNETs you connect can be in different regions or even in different subscriptions. You can also combine VNET-to-VNET communication with multi-site configurations to establish network topologies that combine cross-premises connectivity with inter-virtual network connectivity. This enables you to support a hybrid cloud strategy. An example of VNET-to-VNET connectivity is shown in figure 6 below.

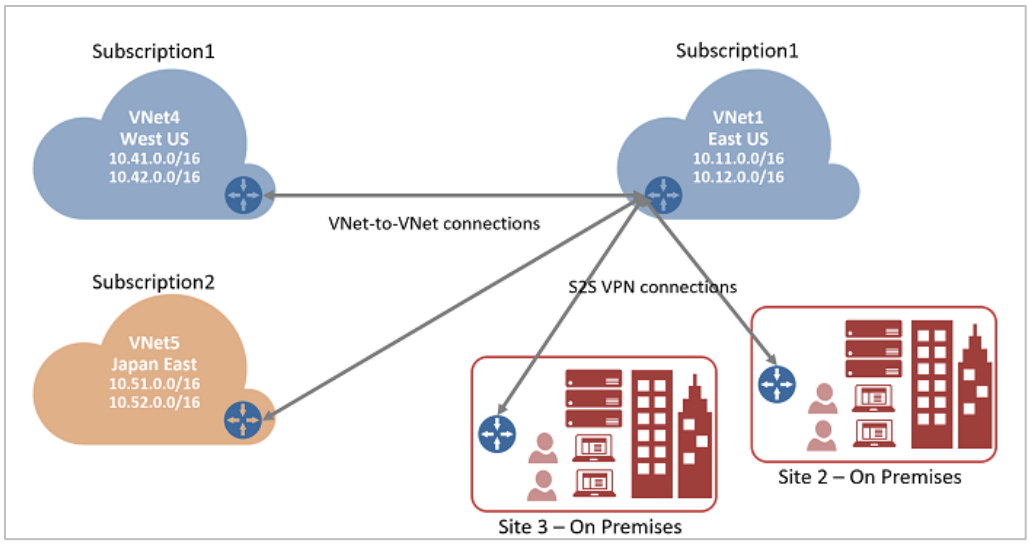


FIGURE 28. CONNECTING MULTIPLE AZURE VIRTUAL NETWORKS (VNET-TO-VNET)

There are several common use cases for VNET-to-VNET connectivity, including:

- **Cross region geo-redundancy and geo-presence.** You can set up your own geo-replication or synchronization with secure connectivity without going over internet-facing endpoints. With Azure Load Balancer and Microsoft or third-party clustering technology, you can setup highly available workloads with geo-redundancy across multiple Azure regions. One important example is to setup SQL Always On with Availability Groups spreading across multiple Azure regions.

- **Regional multi-tier applications with strong isolation boundary**. Within the same region, you can setup multi-tier applications with multiple virtual networks connected together with strong isolation and secure inter-tier communication.

- **Cross subscription, inter-organization communication in Azure**. If you have multiple Azure subscriptions, you can now connect workloads from different subscriptions together securely between virtual networks. For enterprises or service providers, it is now possible to enable cross-organization communication with secure VPN technology within Azure.

Step-by-Step

The high-level steps for configuring VNET-to-VNET include:

1. Plan your IP address ranges
2. Create your virtual networks
3. Add local networks
4. Create the dynamic routing gateways for each VNet.
5. Connect the VPN gateways

Step-by-step instructions for configuring VNET-to-VNET connection between Azure regional data centers is available on the Microsoft website in *Configure a VNet-to-VNet connection using the Azure portal* at https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-vnet-vnet-resource-manager-portal. Additionally, when you visit this link, you'll be presented with two additional deployment at the top of the article in a drop-down menu: **Resource Manager - PowerShell** and **Classic - Azure Portal**. You should only use the Classic deployment option for legacy deployments in Azure and use Resource Manager for new Azure deployments.

Connecting ASM VNets with ARM VNets

If you have been using Azure for some time now you may know that there are two management models in Azure called Azure Service Manager (referred to as classic), and Azure Resource Manager (ARM). Most organizations are leveraging classic mode which was the earlier method of deploying resources in Azure while Microsoft now recommends using the ARM-based deployments. In this case, you may probably have some VMs running in classic VNets where the newer VMs are running on ARM-based VMs.

Such scenarios require you to create a VPN between classic & ARM VNets to enable communication between resources in ASM & ARM. It is possible to create a secure tunnel connectivity between the different versions of VNets.

For a complete reference you can refer *Connect virtual networks from different deployment models using the portal* at https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-connect-different-deployment-models-portal. Additionally, when you visit this link, you'll be presented with a secondary deployment at the top of the article in a drop-down menu: **PowerShell**.

ExpressRoute

Azure ExpressRoute enables you to create private connections between Azure datacenters and infrastructure that are on your premises or in a colocation environment, addressing some of the concerns associated with a site-to-site VPN across the Internet. ExpressRoute connections do not go over the public Internet and offer more reliability, faster speeds, lower latencies and higher security than typical connections over the Internet. In some cases, using ExpressRoute connections to transfer data between on-premises and Azure can also yield significant cost benefits.

With ExpressRoute, you can establish connections to Azure at an ExpressRoute location through three different methods: direct layer 3 through an *exchange provider* facility (Co-located at a cloud exchange) (figure 7 left) or Point-to-point Ethernet connections (figure 7 middle) or directly connect to Azure from your existing WAN network, such as a MPLS VPN, (figure 7 right) provided by a *network service provider* using layer 3.

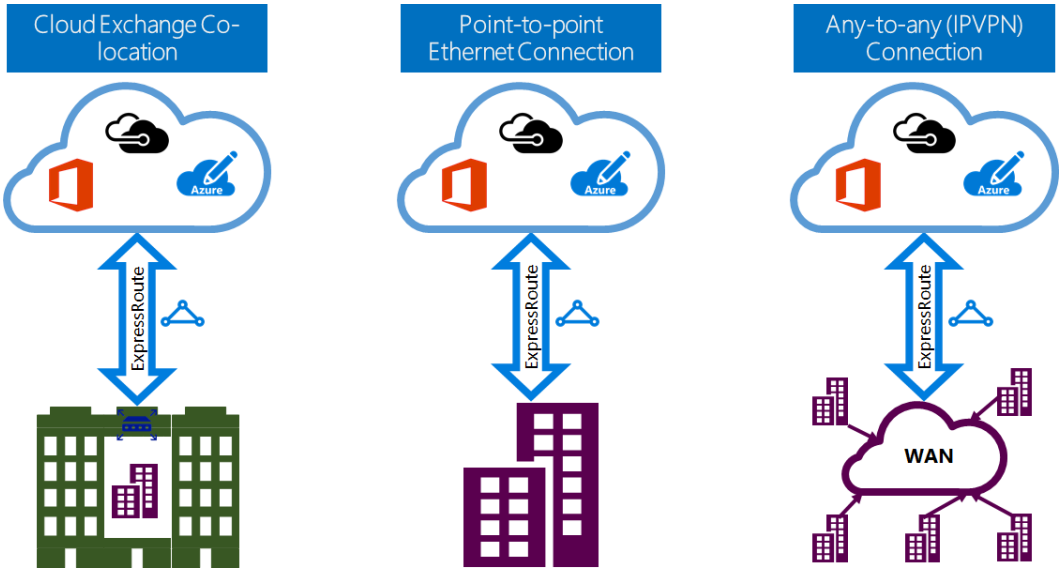


FIGURE 29. AZURE EXPRESSROUTE CONNECTIVITY OPTIONS

You can opt to enable one or both types of connectivity through your circuit to your Azure subscription. However, you will be able to connect to all supported Azure services through the circuit only if you configure both direct layer 3 and layer 3 connectivity.

The diagram in figure 8 shows a logical representation of connectivity between your corporate network and Azure. In the diagram, a circuit represents a redundant pair of logical cross connections between your network and Azure configured in Active-Active configuration. The circuit is partitioned to 3 sub-circuits to isolate the Azure Computer (IaaS) traffic from Azure Public Services (PaaS) traffic and Office 365 Services Traffic.

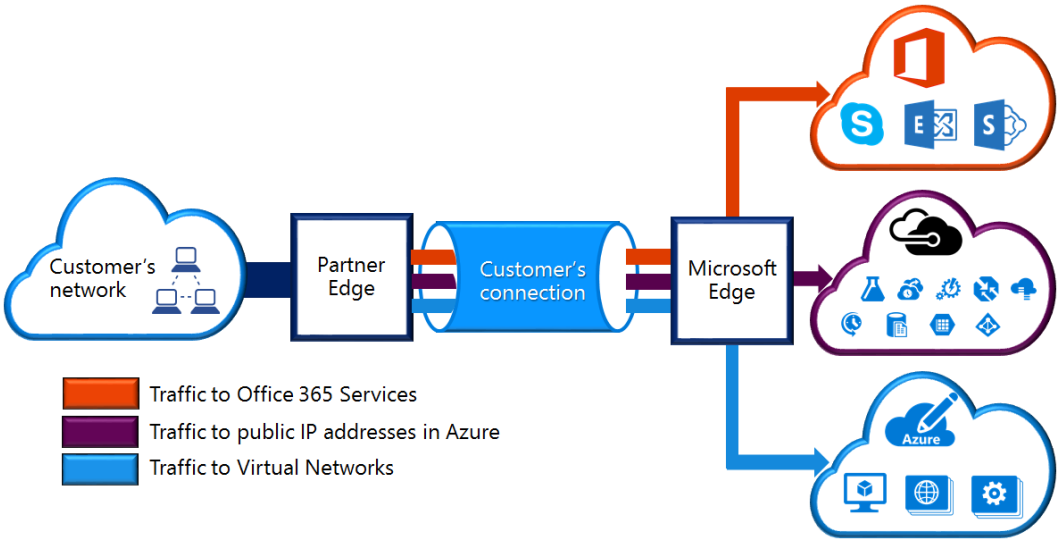


FIGURE 30. EXPRESSROUTE CONNECTIVITY TO AZURE RESOURCES

ExpressRoute supports most Azure services with a few exceptions that are listed in the *Supported services* section of the *ExpressRoute FAQ* page at https://azure.microsoft.com/en-us/documentation/articles/expressroute-faqs/.

ExpressRoute – Co-located at a cloud exchange Option

There is a cost difference between point-to-point connectivity from your datacenter to Azure versus a connection from a datacenter which has **ExpressRoute On Net**. The term **On Net** used by telecom & data center providers means that there is no additional cost needed to provide the connectivity, as the service provider's data center has already been connected to Azure. Several Azure Datacenters have been **On Net**-enabled to provide ExpressRoute connectivity.

ExpressRoute – Point-to-point Ethernet connection

You have the option to connect your on-premise data centers to Microsoft Azure through point-to-point Ethernet links. Point-to-point Ethernet providers offer Layer 2 or managed Layer 3 connections between your data center and Azure.

If you are connecting to Azure through a point-to-point Ethernet connection, you will need a pair of physical cross-connections and to configure a pair of BGP sessions per physical cross connection, one public peering and one for private peering, in order to have a highly available link. The cross-connects go into the exchange provider's cloud connectivity infrastructure. They do not connect directly to Azure routers.

The advantage of this ExpressRoute option includes greater flexibility to leverage existing investments (e.g. – reuse existing hardware, use the co-location facility you are already using), control of network routing and greater network speeds. On the downside, data transfer is not unlimited with this option. This flexibility comes at the cost of additional considerations (and possibly effort) in the onboarding process.

ExpressRoute – Any-to-any (IPVPN) networks Option

If you connect to Azure through a network service provider, the network service provider takes care of configuring routes to all the services. Work with your network service provider to have routes configured appropriately.

The Network Service Provider option is great for organizations already consuming managed WAN services like MPLS. You can connect to Azure through your existing MPLS provider. Coupled with the fact that the network service provider takes care of configuring the routing, onboarding is a relatively easy process. In the end, your organization benefits from private connections to Azure data centers from any of your connected corporate sites.

ExpressRoute Service Tiers, Pricing, and Providers

The service tiers and pricing of the different ExpressRoute options differ in ways that may also make the best option (from a cost perspective) immediately obvious, at least in some cases. Customers can select either **Metered Data** plan or **Unlimited Data** plan. ExpressRoute circuits allow for bursting up your subscribed bandwidth at no additional charge, subject to your service provider's ability to support the additional capacity. For instance, a customer whose Azure services are located in the US can send or retrieve data to/from any Azure region in the US without the need to pay an additional fee on top of their existing plan charges.

Pricing
You can find the latest ExpressRoute pricing details on the *ExpressRoute Pricing* page at http://azure.microsoft.com/en-us/pricing/details/expressroute/. Be sure to select the appropriate Azure Datacenter location on the page to see pricing appropriate to your situation.

Providers
ExpressRoute requires that your organization work with a service provider on Microsoft Azure's partner list. For more information, see the *ExpressRoute connectivity providers* section of the *ExpressRoute partners and peering locations* page on the Microsoft website at https://azure.microsoft.com/en-us/documentation/articles/expressroute-locations/.

Chapter Summary

In this chapter, we have covered the hybrid network connectivity options available with Microsoft Azure, including point-to-site VPN, site-to-site VPN, multi-site VPN, VNET-to-VNET connectivity and ExpressRoute. We also discussed the use cases where each option fits in a hybrid connectivity strategy. Finally, we examined high-level configuration steps and where to find the most current step-by-step guidance for a successful configuration.

You may have noticed several references to in "Chapter 3 – Azure Virtual Networking" in this book, which is a critical chapter you should read before continuing further.

You should now have a base of knowledge on Azure hybrid connectivity options that will enable you to talk to key IT decision makers and business stakeholders to consider Microsoft Azure for your hybrid datacenter workloads.

Chapter 6 Azure Virtual Machines

Azure Virtual Machines (VM) are the primary touch point many new customers will use from their very first day working with Microsoft Azure (Azure). In principal, this is where people feel naturally safe as the concept of an Azure VM is in many ways like a traditional on-premise VM. However, in Azure, there are further concepts that we need to understand in order to design and build enterprise-grade workloads in Azure Virtual Machines. Additionally, we need to explore topics that show us how we can use the power and flexibility of the cloud to ease management of complex environments.

In this chapter, you will learn about Azure VM concepts, as well as basic and advanced deployment options through hands-on examples, including the Azure Portal, Azure PowerShell, as well as with ARM Templates. You will learn how you can create ARM templates in Visual Studio 2015, which offers a bit of GUI authoring help for developers and IT Pros alike.

Once you have deployed VMs and other resources, you will learn how to perform VM configurations using Azure VM Extensions, as well as PowerShell DSC integrated with Azure.

A couple of important notes:

As in previous chapters, we will focus on Azure Resource Manager (ARM) functionality or Azure v2. Before we get started, there are a couple of important points of clarification:

- Where you see the tag *Classic* in the Azure Portal (ARM Portal, or simply portal) this refers to the original Azure Service Management Portal or service management API located at https://manage.windowsazure.com.

- Examine the components for their Availability in the new Azure Portal located https://portal.azure.com.

Everything in this chapter leverages ARM capabilities and the Azure Portal at https://portal.azure.com unless otherwise specified.

Azure Resource Manager

In this section, we will discuss and introduce some of the most important topics related to ARM features and functionality.

Overview of Azure Resource Manager

ARM, which is accessible at https://portal.azure.com, was designed to be flexible and agile for the ever-changing Azure Platform and the requirements of Microsoft's most demanding customers.

In Figure 1, you will see a screenshot of the Azure Resource Manager Portal.

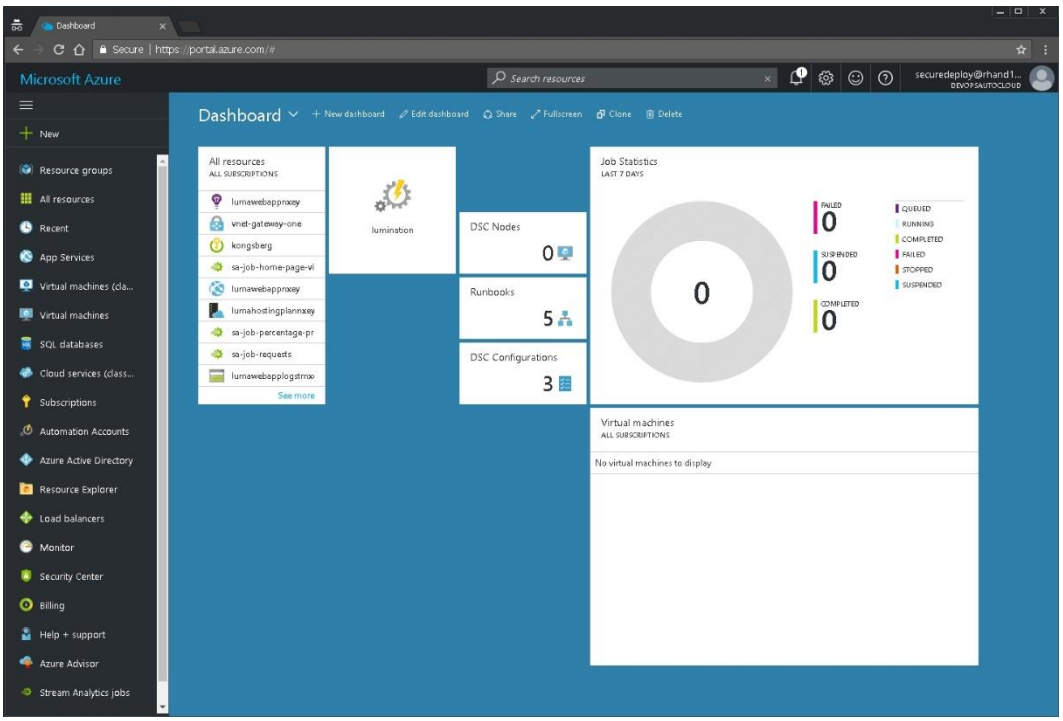


FIGURE 31. AZURE RESOURCE MANAGER PORTAL

You might ask what was wrong with the original platform that is accessible at https://manage.windowsazure.com (also referred to as the Service Management Portal, or Azure v1). The answer lies in some key limitations related to deployment and management. ARM is not only designed to make the deployment of VMs easier, but also to make entire environments (VMs, VNETs and applications) efficient, faster and more reliable.

The Azure Portal is a completely new framework and API model. This means that if you have legacy resources in the Classic Portal, will likely be migrating services into ARM to take advantage of its rich management capabilities.

The ARM API and framework allows you take advantage of the following features within Azure for tenants such as:

- Template Deployment
- Role Based Access Control
- Tagging
- Azure Resource Groups

These components will be covered in the following pages. The key takeaway is that ARM is the foundation of Microsoft's strategy to provide a consistent API and framework, which spans public and private cloud offerings.

Azure Resource Groups

In Azure v1, the *cloud service* was the logical container for management when you deployed a VM into Azure. The cloud service allowed you to store multiple VMs in the same container for management or load balancing purposes or a variety of other different reasons we will not delve into here.

However, it was an inflexible model and did not allow association with already constructed management domains within a customer's environment. For example, many companies today have a networking team, whose job is to look after the switches, the IP network, routing, firewalls etc. They do not know about the servers or services except their switch port connections and IP assignments.

If they needed to access Azure to manage the virtual networks and gateways, previously via the Service Management Portal, we would have to grant them co-administrator privileges for the Azure Subscription. This would, of course, allow them to see all the resources in Azure, not just the networking components, limiting our ability to apply the rule of least privilege. This security model opened the door for accidents in which novice administrators might unintentionally cause problems.

As described in previous chapters, the new boundary for management is the *resource group*. Azure resource groups allow you to combine resources together to manage them as you wish. Let us revisit the previous example if you wanted to grant the network team rights to manage only virtual networking components in Azure. With Azure resource

groups, you can create a resource group, which will contain all the networking resources in an Azure subscription. Subsequently, you can assign permissions to this resource group so that only the network team can perform operations on resources in this group. In figure 2, we show you a logical model of two resource groups separating out the management domains for storage and networking.

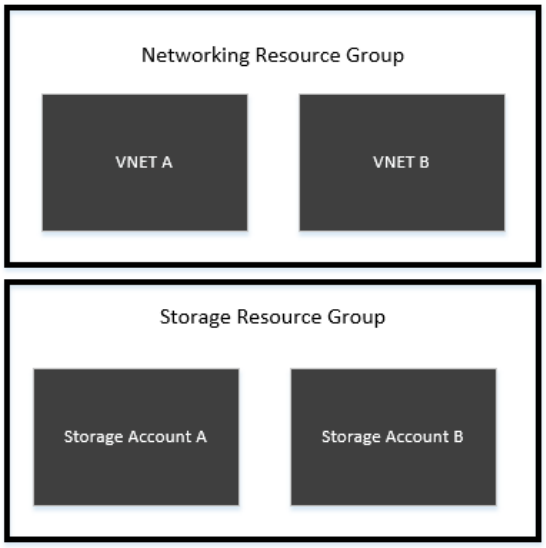


FIGURE 32. AZURE RESOURCE GROUPS FOR STORAGE AND NETWORKING

The choice on how resource groups are constructed and what resources they contain is an important part of the design process. Implementing services in Azure requires thorough planning because you need to design your resource group structure based on the types of services your organization wants to deploy and how administrative privileges should be delegated to them.

> **IMPORTANT:** Resources (i.e. VM's, VHD's, and VNETs, etc.) can only be a member of one resource group and every resource in ARM is required to be in a resource group.

In the Azure Portal by default, there are multiple default resource groups to help get you started. Figure 3 shows you the Default-Networking resource group; notice that the resources and their names in your Subscription will be different than shown in the image below.

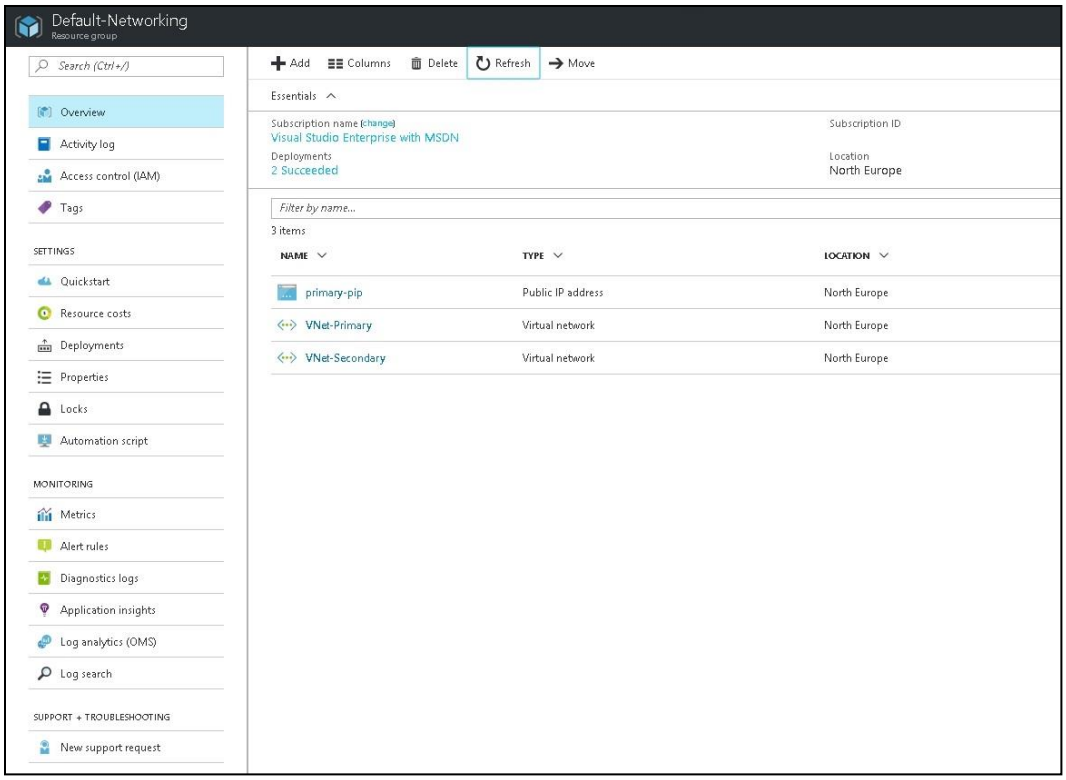


FIGURE 33. DEFAULT-NETWORKING RESOURCE GROUP

In Figure 4, you can see that the Resource costs blade provides basic billing information about the resources within the Resource Group.

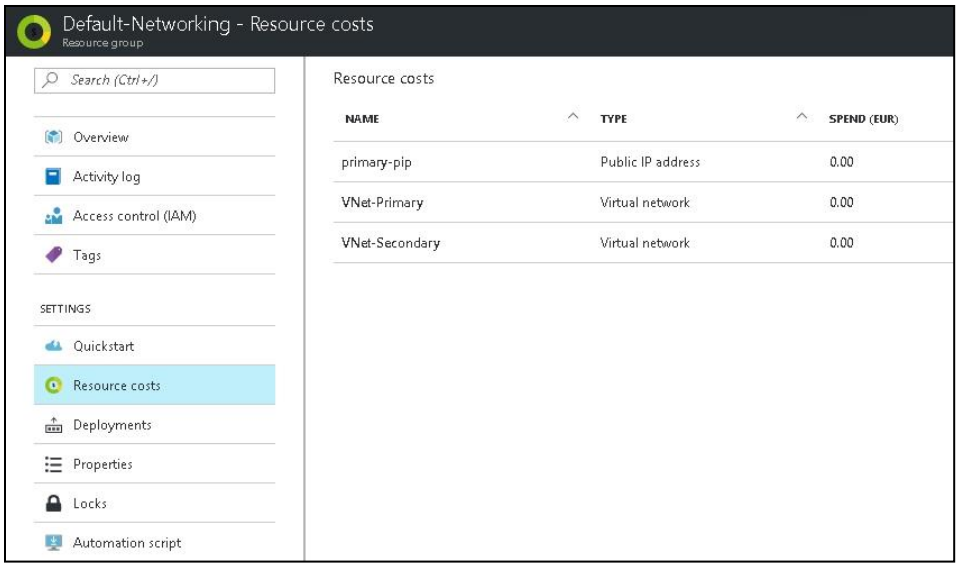


FIGURE 34. RESOURCE COSTS BLADE

In Figure 5, you can see monitoring and diagnostic features that are available to you under the *Monitoring* section of the resource group blade.

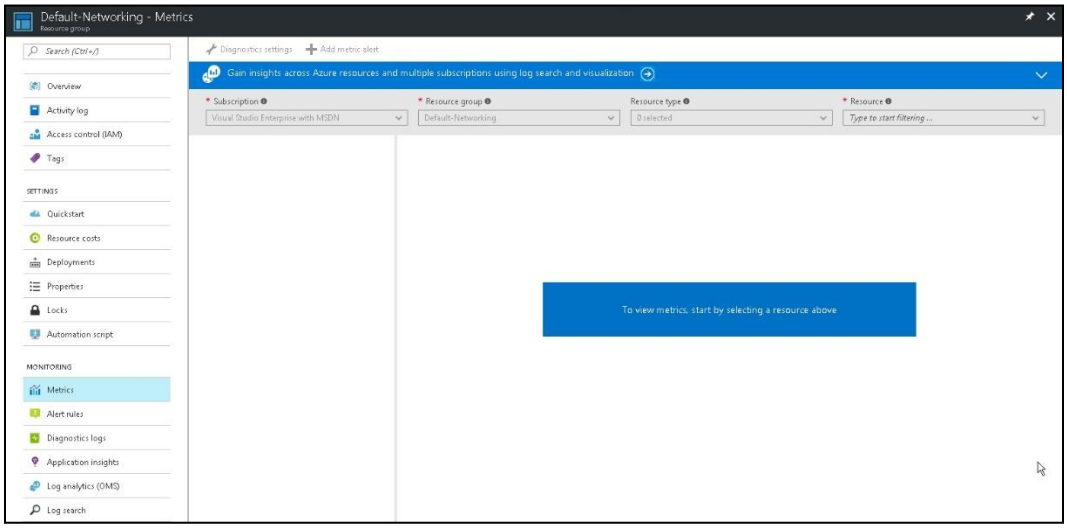


FIGURE 35. MONITORING SECTION UNDER THE RESOURCE GROUP BLADE

Template Deployment

ARM allows you to deploy IaaS and PaaS environments into Azure in a programmatic and consistent manner using templates. These JSON based templates consist of expressions that you can use to construct the values and configuration of your deployment. This is essential when you are moving an application between environments and wish to ensure consistency. For example, moving an app from Development to a Test environment and finally, to the Production environment. Each stage of the move requires pre-requisite software installations and potentially multiple post-deployment steps. The installation requirements of the pre-requisite software and the post deployment steps may be relatively simple or extremely intricate.

Technically, you could perform all of your deployment just using Azure PowerShell cmdlets, but your deployment would only be as reliable as your PowerShell authoring skills, error checking and restart routines, etc. More importantly, you would be missing out on some of the key benefits of ARM. A few of the key features of ARM that make it the go-to option for complex deployments include:

- **Declarative**. Instead of deploying resources individually, ARM Templates allow you to deploy an entire environment programmatically to a single Resource Group. You can define dependencies between resources to ensure all elements of your deployment are deployed in the correct order. For example, if a VM tried to join an Active Directory domain before the Active Directory domain was fully deployed, the domain join operation (and all steps after) would fail. The ARM JSON template capabilities allow you to describe these dependencies to deliver reliable results every time.

- **Idempotent.** If a JSON based deployment in ARM fails, you can restart the deployment and it will verify all of the previous elements of the deployment were successfully deployed and then attempt to deploy the elements that were not.

> **Note**: It is important that we not overstate idempotence here. For example, if your deployment fails during the deployment of a PowerShell script, you will likely have to do some cleanup to the script before you attempt to restart the deployment. In many cases, it is often faster to delete the deployment and re-deploy.

- **Reusable.** Once you have created and tested a template, you can share the template and related resources (PowerShell scripts, DSC modules, etc.) with anyone who can then use the template to deploy their own environment!

- **Fast Cleanup**. Not only does this allow you to deploy several resources very quickly in Azure, you are also able to delete all the resources you deployed by simply deleting the Resource Group they were deployed to. Having to worry about dependencies from using the Service Management Model in Azure are no longer an issue.

This is not to say deploying with ARM via Azure PowerShell cmdlets does not have its place. There are many deployment and configuration operations where this is going to be faster, easier and a better fit than ARM JSON templates. Scenarios where you just want to deploy a VM to an existing environment, or just do some bulk configuration or administration, PowerShell is still a great fit. Simply match the right methodology to your situation using what we have shared here as guidelines.

In Figure 5 we show the basic outline of a JSON template, for more information around the schema and language please refer to *Authoring Azure Resource Manager Templates* on the Microsoft website at https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/

```
{
"$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
"contentVersion": "",
"parameters": { },
"variables": { },
"resources": [ ],
"outputs": { }
}
```

FIGURE 36. JSON TEMPLATE OUTLINE

A common example of how you can use an ARM template is a web application consisting of a database server and a web server. These components require resources, such a storage account, virtual networks and public IP addresses. Under the resources section in the ARM template, you can specify each resource as well as the dependencies of that resource. For example, a VM will not be deployed without a storage account (or managed disk) and a virtual network being created. ARM templates can also leverage PowerShell Desired State Configuration (DSC), enabling additional system configuration and application deployment capabilities post-VM deployment.

The paradigm of ARM allows IT operations and application development teams to update their ARM templates as their applications evolve. As the application moves from development to test to production, the dependencies and order-of-operations are already written into the application deployment, helping to ensure consistent application deployment during the lifecycle process.

Tags

Earlier we described resource groups and how they are used to group and manage resources together as a single entity. However, this might not give us the full picture of all the applications that are being managed.

For example, you may have a finance department that uses multiple applications, as well as a sales department that also uses a number of applications, spread across multiple resource groups. Since these applications are unique for their respective department they have different cost centers and require different administrative and delegation requirements.

In this case, we have good visualization of the application by department, but no single view of what other resource groups these departments interact with on a regular basis. To organize resources across multiple resource groups, you can use *tags*. Tags allow you to categorize all resources in ARM using whatever logical group is best suited for your needs. Each tag consists of a key-value pair that can be modified or removed as needed. Each resource or resource group can have a maximum of 15 tags, each tag name can be up to 512 characters in length and each tag value can be up to 256 characters in length. In the image shown below in Figure 7, the Tags Blade has been opened up and the Department:Sales tag has been used to filter out the resources and resource groups that belong to the sales department, also shown in Figure 7.

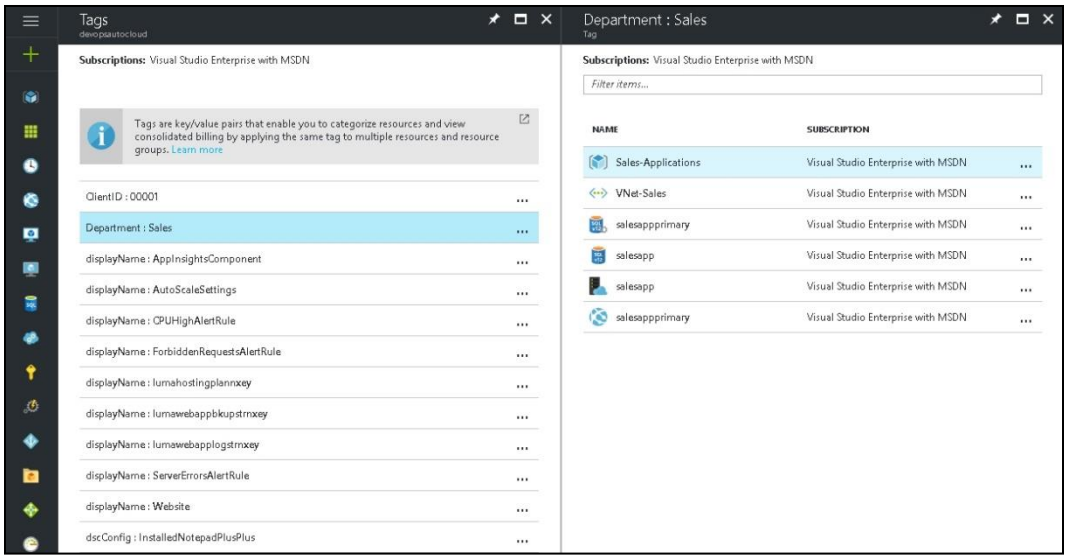


FIGURE 37. RESOURCES WITH THE DEPARTMENT:SALES TAG ARE SHOWN

Role Base Access Control

One of the most important additions in the Azure Portal is role-based access control (RBAC). Simply put, this gives companies the ability to delegate administration, assigning appropriate rights for resources to suit their management model.

For example, in a scenario where a company has a networking team for managing all network elements and a storage team for managing the storage systems, with RBAC we can assign the network and storage teams the permissions they need to achieve their administration tasks in Azure without giving them permissions to areas they do not need. In the Azure Portal, there are a wide range of default roles, as shown in Figure 8.

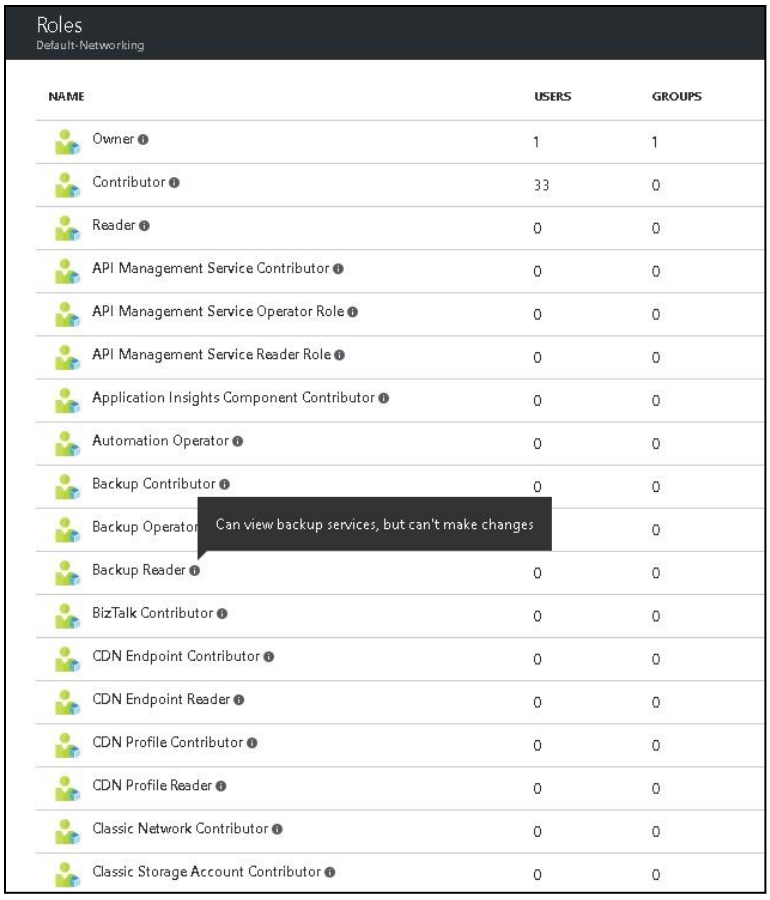


FIGURE 38. DEFAULT ROLES

As you can see from Figure 8 if you click the ***i*** icon next to the name of the role, you will get a description of what tasks the role allows a user to perform. You can assign users to multiple roles, as necessary.

Additionally, you can assign permissions directly at the resource level. As shown in Figure 9, the Public IP address has the Access control (IAM) blade available so that you can assign permissions.

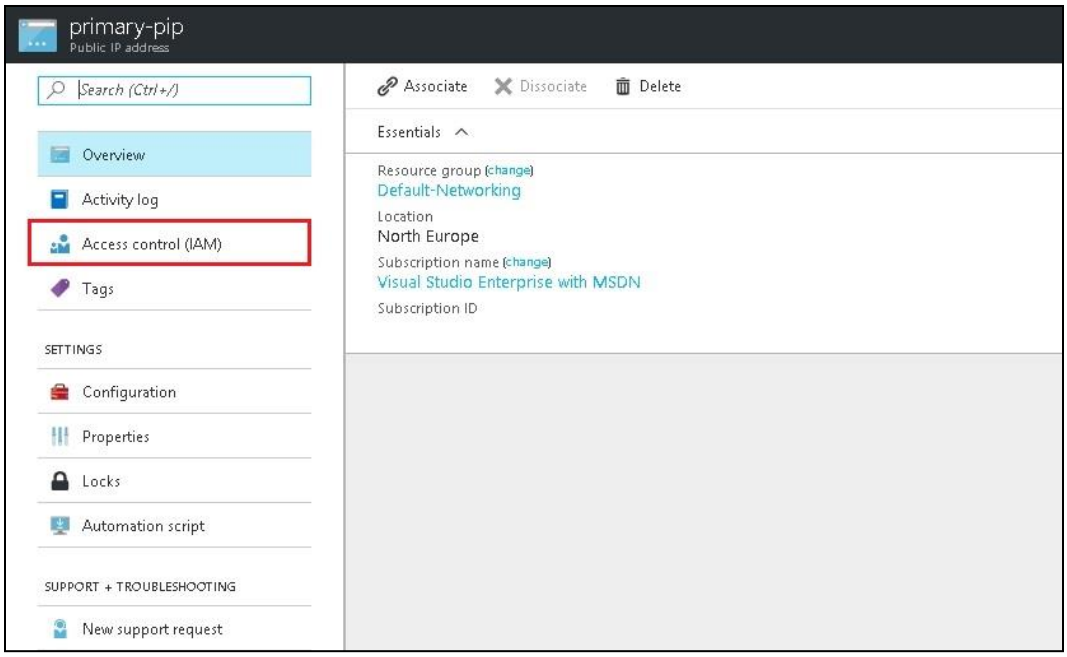


FIGURE 39. RBAC ON A PUBLIC IP ADDRESS RESOURCE

Azure Virtual Machine

In this section, we are going to cover Azure virtual machines in detail. Azure virtual machines are very similar in conceptually to the traditional VM you might have used on-premise in Hyper-V. In many cases, you can move a VM from on-premise to Azure, enabling a company to move an application and host it on Azure without requiring them to redevelop the application to be *cloud aware*. Azure currently supports generation 1 Hyper-V VMs with virtual disks in VHD format. This is important for many companies as the investment to move these workloads to be *cloud aware* is significant and often the benefits do not justify the development costs.

> Azure Site Recovery (ASR) is the recommended service to use to migrate workloads running on Hyper-V Generation 2 VMs to Azure since ASR will do the conversion and migration for you.

VM Architecture

The first area we need to understand is the architecture of the VM. In Figure 10, we show you the basic layout of VM architecture

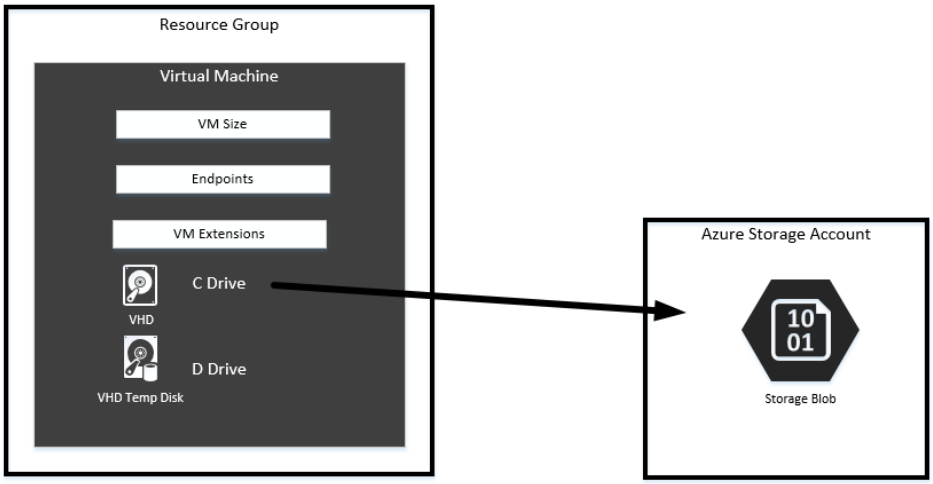


FIGURE 40. VIRTUAL MACHINE BASIC ARCHITECTURE

The VM, much like in Hyper-V, is a configuration container that references the resources it requires in order to operate. In this section, we will discuss each component in detail.

> The basic architecture diagram is for a VM deployed with an unmanaged disk. You also have the option to deploy VMs in Azure on managed disks. Managed disks are a storage resource that is fully managed by Azure and requires you to only define the type and size of the disk you need.

VM Size

Azure VMs have a variety of different choices when it comes to the size of the VM you will select and deploy. First, you need to choose your pricing tier for the VM; as of March 2017, there are two choices: Basic and Standard.

Basic tier VMs are best suited for environments (i.e. - test and development) that do not require high availability, auto-scaling or memory intensive workloads. Additionally, Basic

tier VMs restricts the number of virtual disks you can attach to the VM and limit the IOPS of the disk to approximately 300 IOPS per disk.

Standard tier VMs supports all available virtual machine configurations and features including high availability and auto-scaling; also, there are no restrictions on the size of the VMs you choose to deploy. Virtual disk performance is also better, supporting up to 500 IOPS per disk. When choosing a standard tier VM, you have a choice of machines rated by Series.

All VMs include a temporary disk (D: by default), which is designed to be used as a working area to store non-persistent (temporary data).

> **Note:** When designing services in Azure, it is very important to understand that the limit is considered an "up to" limit not a guarantee or SLA.

The VM series available in Azure currently include the following:

- **A Series** - A Series range from an A0 to A11 with CPU cores that range from 1 to 16. Memory ranges from 768MB to 112GB. The latest generation of the A Series virtual machines, called Av2, include similar CPU performance and faster disks. In this series, there are no solid-state drive (SSD) options available.

- **D Series** - D Series range from D1 to D14 with CPU cores that range from 1 to 16. Memory ranges from 3.5GB to 112GB. D Series also have a DS and DSv2 range of machines, they have similar ranges as the D Series but allow you to achieve higher performance and disk IOPS.

- **F Series** - F Series range from F1 to F16 with CPU Cores that range from 1 to 16. Memory ranges from 2GB to 32GB. F Series VMs come with 2GB and 16GB of local SSD per CPU core and are effective for compute intensive workloads. F Series also has an FS variant that supports Premium Storage.

- **G Series** - G Series range from G1 to G5 with CPU Cores that range from 2 to 32. Memory ranges from 28GB to 448GB. The Temporary disk is based on SSD ranging in size from 384GB to 6144GB. G Series also have a GS range of machines, they have similar ranges as the G Series but allow you to achieve

higher performance and disk IOPS with support for Premium Storage. G Series VMs are effective for computational performance and large database workloads.

- **H Series** - H Series range from H8 to H16 with CPU Cores that range from 8 to 16. Memory ranges from 56GB to 224GB. H Series VMs are designed to handle high performance computing workloads such as financial risk modeling, molecular modeling, and genomic research.

- **L Series** - L Series range from L4 to L32 with CPU Cores that range from 4 to 32. Memory ranges from 32GB to 448GB. The Temporary disk is based on SSD ranging in size from 768GB to 5630GB. L Series are storage optimized VMs built for low latency workloads such as NoSQL databases.

- **N Series** – N Series range from N6 to N24 with CPU Cores that range from 6 to 24. Memory ranges from 56GB to 224GB. N Series VMs come in two type, NC and NV. N Series VMs are ideal for compute and graphics intensive workloads and are effective for remote visualization, deep learning, and predictive analysis.

For a complete list and exact details of all the VM options currently available, refer to *Sizes for virtual machines* in Azure on the Microsoft website at https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-size-specs/.

Generally speaking, the higher the series of VM you choose the better performance you will get. For example, if you need a VM for a CPU intensive workload, like a large SQL Server instance, you may select a G Series VM. The underlying physical hardware the virtual machine would be deployed on is specially designed for CPU intensive workloads.

Storage

A VM will always have at least two disks by default, the operating system disk, and a temporary disk.

The operating system disk is where the operating system lives and the VM boot disk. This disk has caching turned on by default and you cannot turn it off, the options you have are read, read/write.

The temporary disk is simply a scratch disk; this scratch disk comes from the underlying physical host. The temporary disk should not be used as a persistent data store. The size and performance of the temporary disk are linked to the series and size of the VM you select.

VMs can also give you the option of high performance storage underpinning the workload. DS and FS Services VMs can deliver high-performance with up to 51,200 IOPS, while GS Series VMs can deliver up to 80,000 IOPS. The temporary disks on these S-Series VMs reside on SSD hosted by the underlying physical hardware to which the VM is deployed.

Finally, all storage that a VM accesses is via a network connection (through a RESTful API), this introduces an imposed limit that needs to be taken into account. It is possible to achieve a throughput of up to around 3Gbps when you select a VM image size that enables remote direct memory access (RDMA). The A, H, and N Series VMs support access to RDMA.

> As of March 2017, Windows Server 2016 does not support RDMA connectivity in Azure.

You can read more about Azure storage, including storage architecture and disk cache options in *Chapter 4 – Microsoft Azure Storage*.

Network connectivity

In ARM, the network interface of the VM is an independently manageable resource. This is a very important change from the Azure Service Management model where you were required to associate services directly to the VM; in ARM you associate network interfaces directly to other services. An Azure VM can have up to 8 network interfaces assigned to it depending on the series and size of the VM. In Figure 11, you can see the network interfaces listed as individual resources within resource groups.

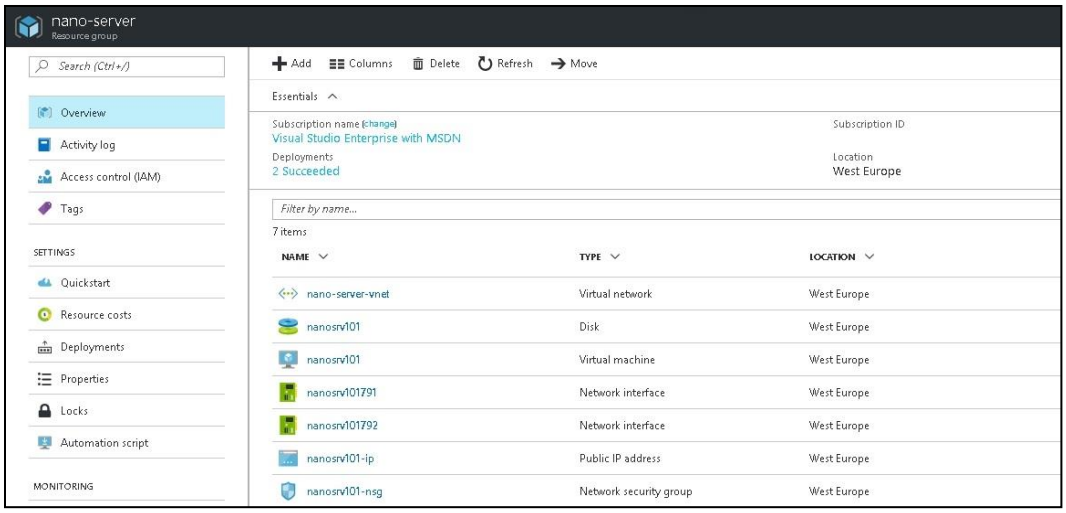


FIGURE 41. NETWORK CARDS AZURE RESOURCES

Figure 12 displays one of the network interfaces in the sample environment from Figure 11. As you can see, the some of the network cards in this sample environment, which can be assigned (to VMs or load balancers). As you can see, the network interface is currently assigned to a public IP address and a network security group (NSG). This is essential as there is a public IP address assigned to the VM in this sample environment. For more details on NSGs, refer to *Chapter 3 – Azure Virtual Networking*.

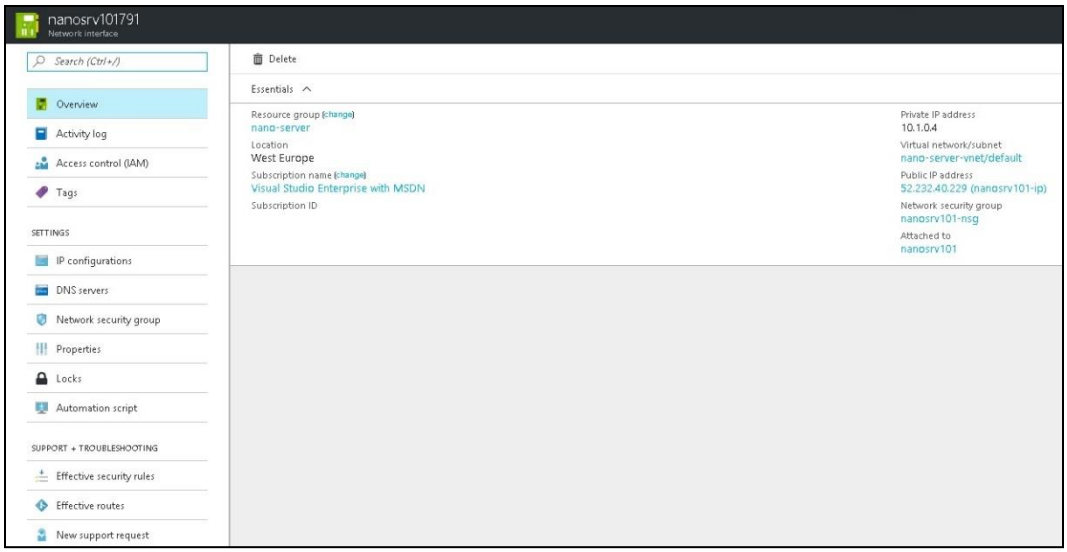


FIGURE 42. NETWORK INTERFACE PROPERTIES

The public IP shown from figure 11 is shown below in figure 13 and is associated with a virtual machine and a network interface. Public IP addresses have the same type of flexibility as network interfaces in that they can be reassigned to other resources as required.

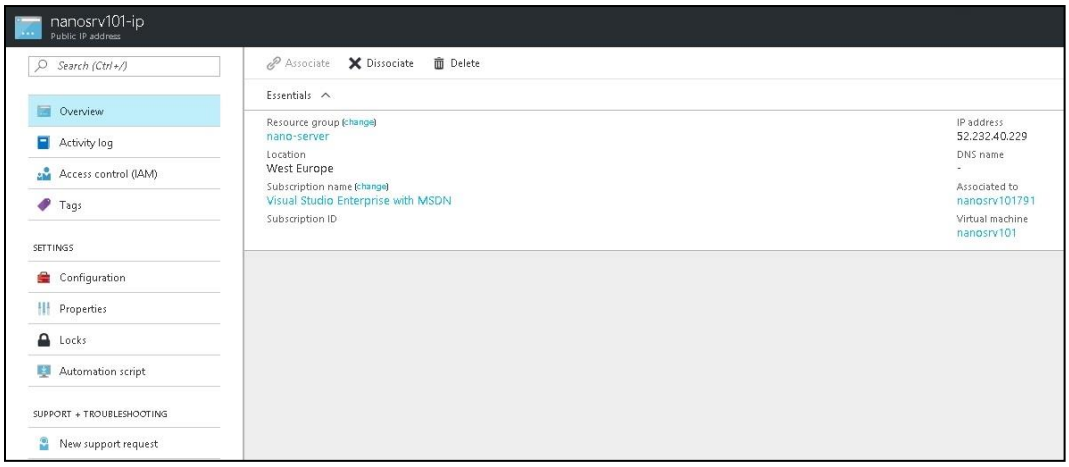


FIGURE 43. PUBLIC IP ADDRESS RESOURCE

If you require endpoints opened to a VM without assigning it a public IP directly, then you will need to configure a network address translation (NAT) rule in an Azure load

balancer and associate it to a network interface. Endpoints still exist in ARM, but are controlled by an Azure load balancer.

VM security

Figure 7.2.5 illustrates the various layers of security you can implement to protect your VM. Administrators can implement multiple layers of NSGs, utilize the Windows firewall (enabled by default in Azure VMs), as well as install Microsoft Antimalware as part of the VM deployment process.

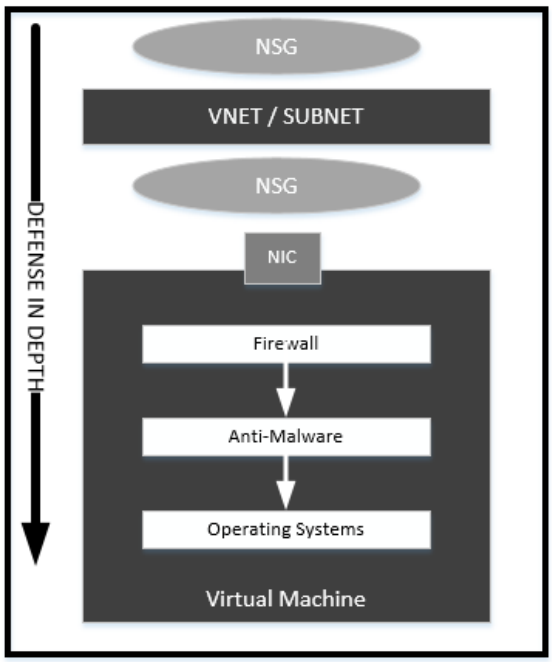


FIGURE 44. DEFENSE IN DEPTH FOR A VIRTUAL MACHINE

This ensures that from the moment you provision the VM into Azure you can enforce protection on any workload on that system.

NSG's can be used to isolate a VMs traffic from other VMs, the internet or an IP source. NSG's can also control outbound traffic flow if necessary, thus allowing you to enforce protection before traffic leaves or arrives at a VM.

Marketplace

The Azure Marketplace is a single repository of prebuilt VMs ready for deployment into Azure. These VMs are designed to be a click and deploy system for rapid deployment of services. Some of the VMs have software licenses built into the runtime cost of the VM (such as with Windows Server or SQL Server VMs), while other VMs require you to purchase a license after deployment. The Azure Marketplace homepage is shown in figure 15.

In the Azure Marketplace, available VMs are presented by category. A search function is also available, enabling you to locate the VM you require through text-based search.

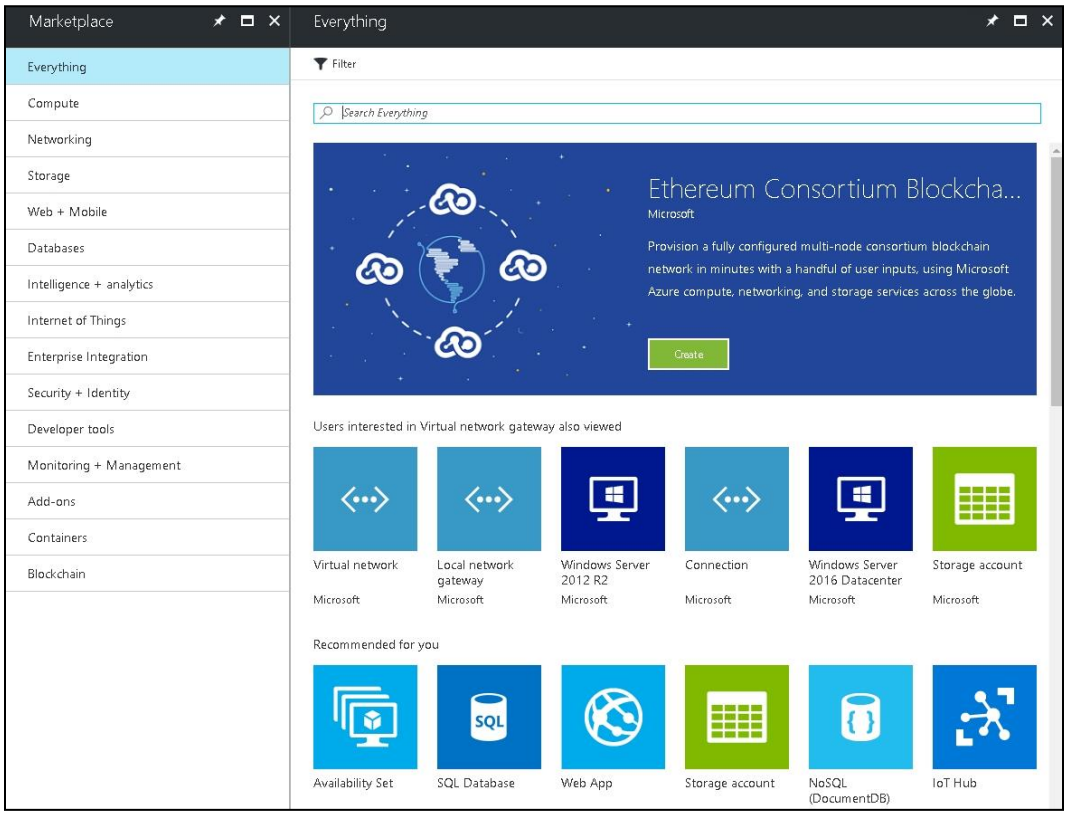


FIGURE 45. MARKETPLACE

Microsoft updates the Azure Marketplace regularly (and the VM images made available through the Marketplace) with the latest patches and releases available from vendors.

Deploying and Configuring Virtual Machines

In the previous two sections, we have described the options available to you for Azure VMs in ARM. In this section, we will dive deeper and show examples on how to deploy a VM, its associated components and some of the available deployment options.

Deploying Virtual Machines

All the examples in the following section utilized Azure PowerShell module version 3.6.0 and all Azure Portal examples leverage the UI located at https://portal.azure.com.

Next, open up a PowerShell Prompt and run the following command to verify that the Azure Resource Manager Cmdlets are available:

```
Get-Module -ListAvailable -Name AzureRm.Resources
```

You should get back the following response, shown in figure 16:

```
ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   3.6.0      AzureRM.Resources                   {Get-AzureRmProviderOperation,
```

FIGURE 46. AZURE POWERSHELL MODULE VERSION

To log in to the Azure Portal, do the following

1. Open a browser and navigate to https://portal.azure.com
2. Type in your credentials for your work, school or personal Microsoft account. show in figure 17.

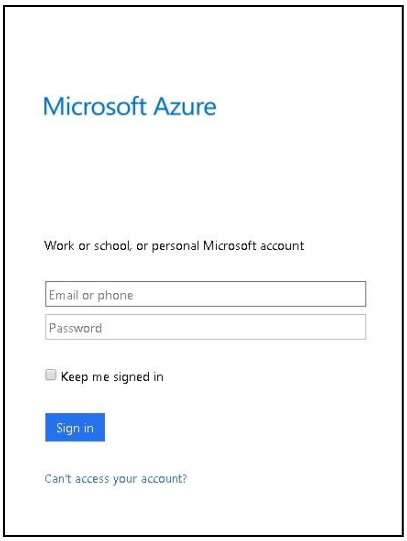


FIGURE 47. LOGGING INTO PORTAL

3. If you are prompted, select the account type as shown in figure 18

FIGURE 48. SELECT ACCOUNT TYPE

4. Enter your password and **click sign in**

5. This should bring you to the Azure Portal Dashboard as shown in figure 19.

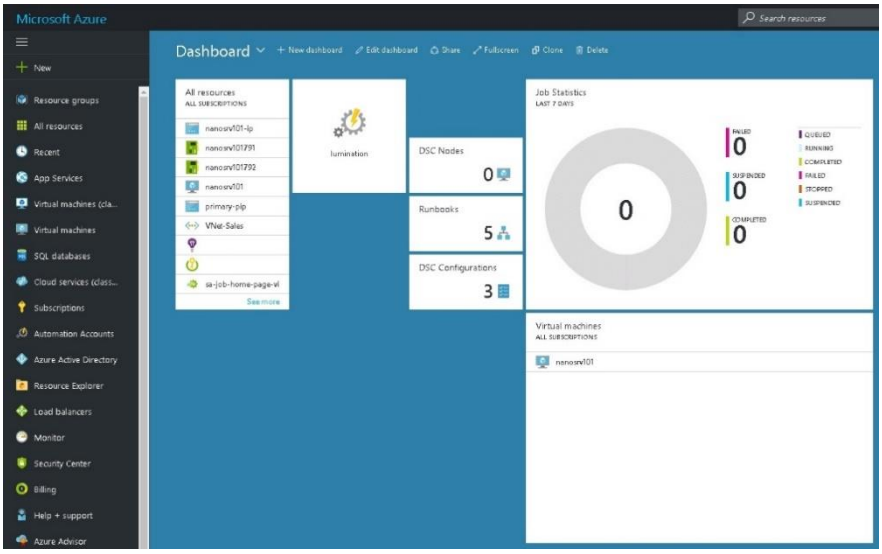


FIGURE 49. AZURE PORTAL AZURE DASHBOARD

To connect to ARM using Powershell with a Microsoft Live account:

1. Open up an elevated PowerShell Prompt.

   Type in the following command and hit Enter.

   ```
   Add-AzureRmAccount
   ```

   The first time you run this command, you will be prompted as to whether you would like to participate in Azure PowerShell Data Collection for usability improvements, type in either **N**[No] or **Y**[Yes] and hit Enter.

2. Next, type in your Credentials for your work, school or personal Microsoft account as shown in figure 20.

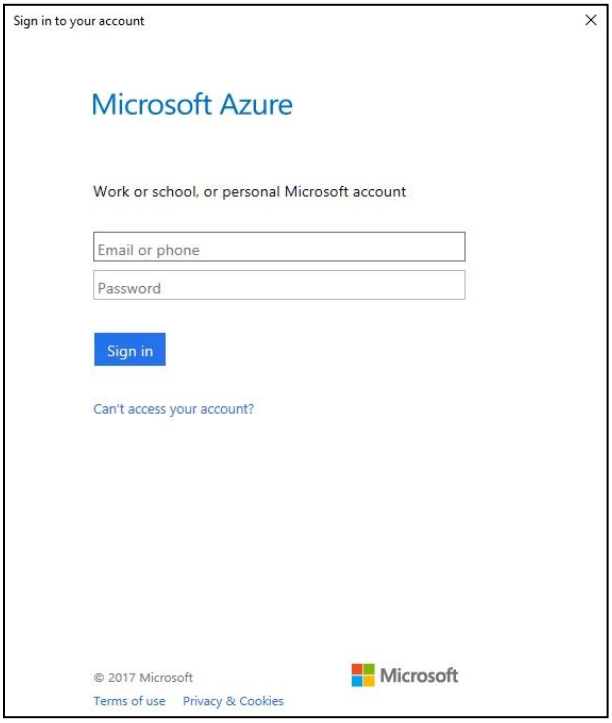


FIGURE 50. SIGN IN PROMPT

If you are prompted, select the account type as shown in figure 21.

FIGURE 51. SIGN IN PROMPT

3. Enter your password and **click sign in**.
4. Once you have successfully logged in, you should see output similar to figure 22 in the PowerShell Console.

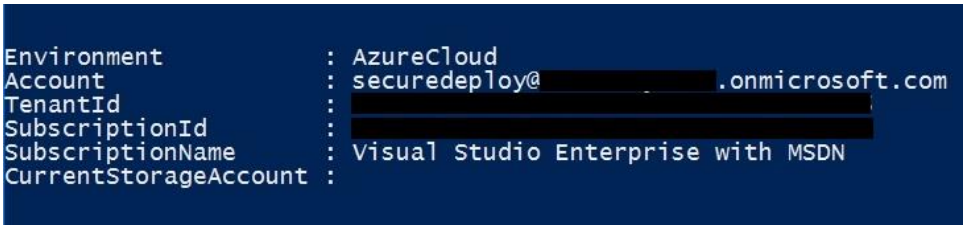


FIGURE 52. AZURE LOGON RESULTS

Deploying a Virtual Machine from the Portal (Windows and Linux)

For your first VM deployment, you will use the Azure Portal. This will help you gain some basic familiarity with the Azure Portal and the Azure Marketplace.

To create a VM in the Azure Portal, complete the following steps:

1. From the Azure Resource Manager Portal located at https://portal.azure.com **Click + NEW** as shown in Figure 23.

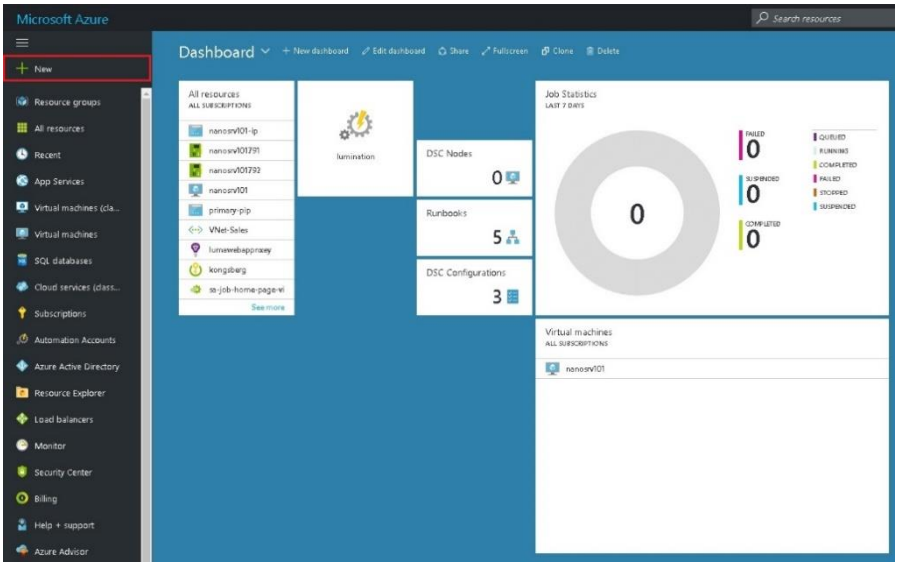


FIGURE 53. NEW ON AZURE PORTAL DASHBOARD

2. From the **New** blade, all the options for deployment are listed in groups. For a VM we need to select **Compute**, as shown in figure 24.

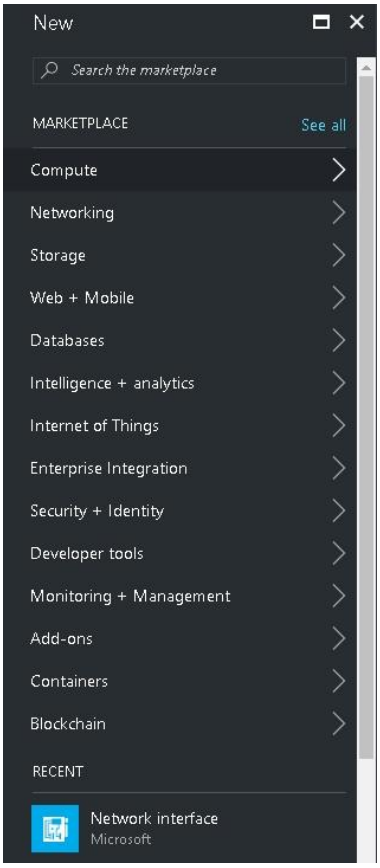


FIGURE 54. SELECT COMPUTE UNDER MARKETPLACE

3. The **Compute** blade presents the most recently used and most commonly requested images. Marketplace provides an area where all the images available (Microsoft and 3rd Party published) in Azure can be selected for Installation. **Click Windows Server 2012 R2 Datacenter** as shown in Figure 25.

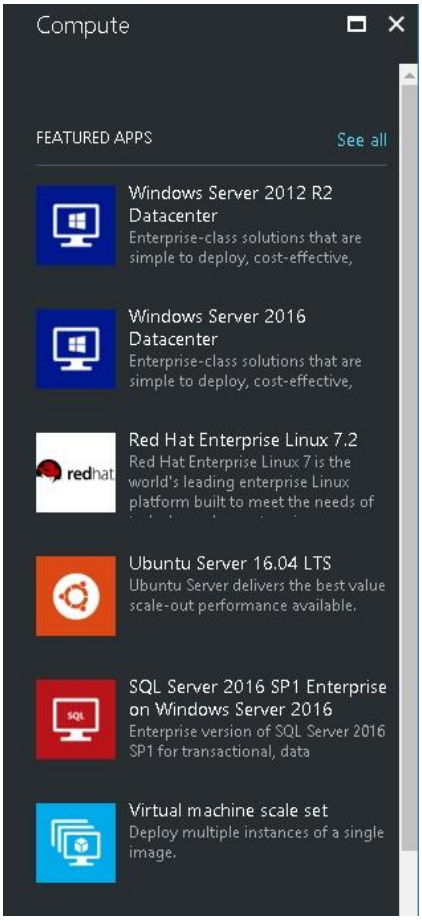


FIGURE 55. WINDOWS SERVER 2012 R2 DATACENTER UNDER COMPUTE

4. After clicking on the Windows Server 2012 R2 Datacenter image, it will open up a summary blade describing the image and ask you to confirm which deployment model you wish to use. As a reminder, *Classic* is the Azure v1 Service Management Portal and *Resource Manager* is ARM as described in this chapter. Select **Resource Manager** and click **Create**, as shown in Figure 26.

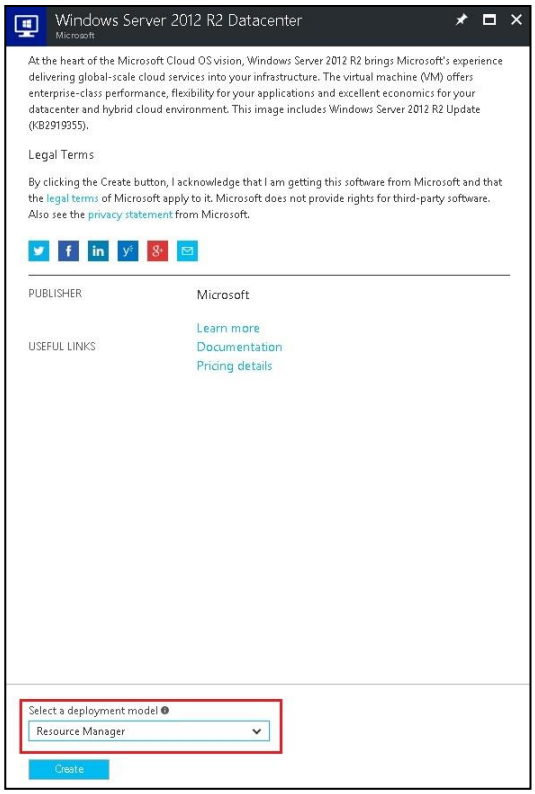


FIGURE 56. CHOOSING DEPLOYMENT MODEL

5. In the **Create Virtual Machine** blade, the **Basics** blade will automatically open, as shown in Figure 27.

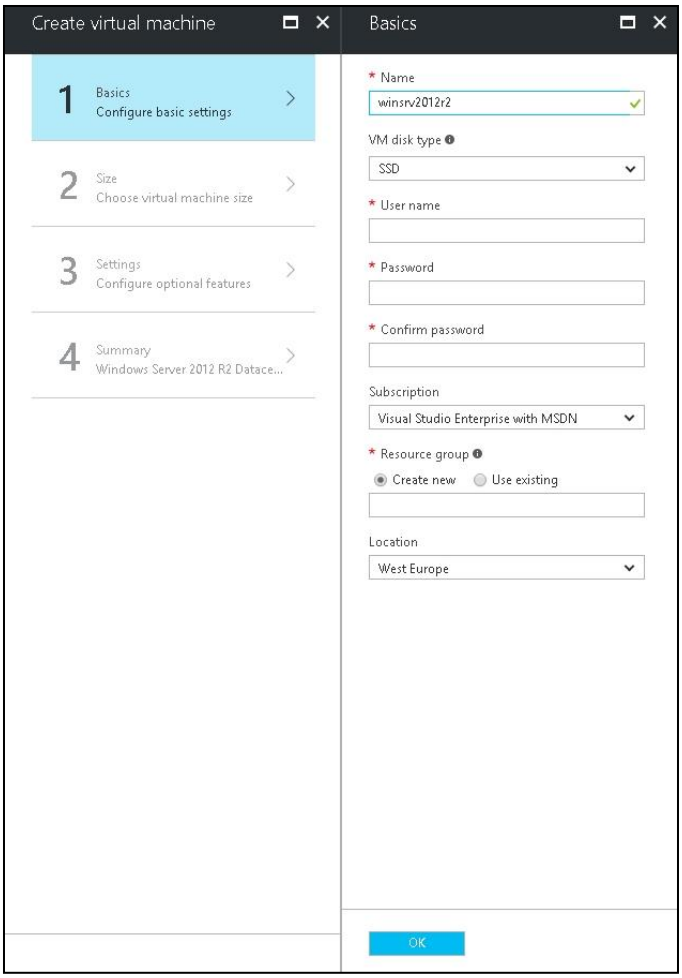


FIGURE 57. BASIC SETTINGS FOR VM

Enter values for the following fields shown in the **Basic** blade:

- **Name** (which is the name of the VM)
- **VM Disk Type** (**HDD** for Standard Disk, **SSD** for Premium Disk)
- **Username** (administrator is not allowed)
- **Password** (complex rules apply)
- **Confirm Password** (complex rules apply)
- For **Resource Group** the field is by default setup to create a new resource group, if you enter a new name, it will create a new resource group and all the VM components will be associated with it. Alternatively, you can click **select existing** and select a

resource group that has already been deployed. For this example, type **deploy-vm** in the **resource group** field.

- For **Location**, select the location where you want to deploy to, in this example choose **West Europe**.

When you are finished, click **OK.**

6. Next, the **Choose a Size** blade will open, which by default will have recommended options for the type of image you have chosen. In Figure 28, you can see we have two recommended options listed. If you would like to view all of the possible options available to you, click on **View all**. Within each option, you should also see the estimated monthly cost of running the VM. For this example, click on **DS1_V2 Standard** and then click **Select**.

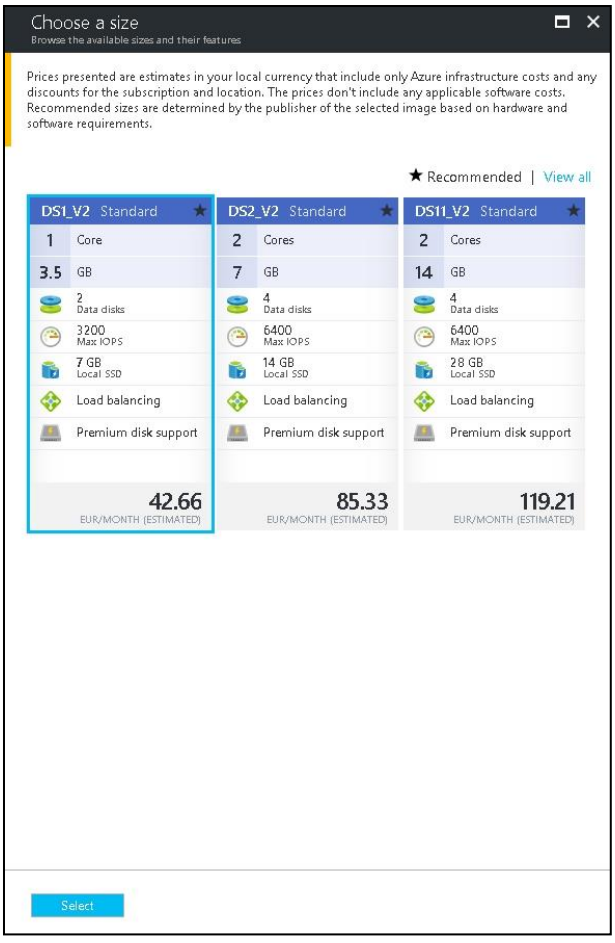


FIGURE 58. VM SIZE SELECTION

7. Next, we will configure VM settings in the **Settings** blade as shown in figure 29.

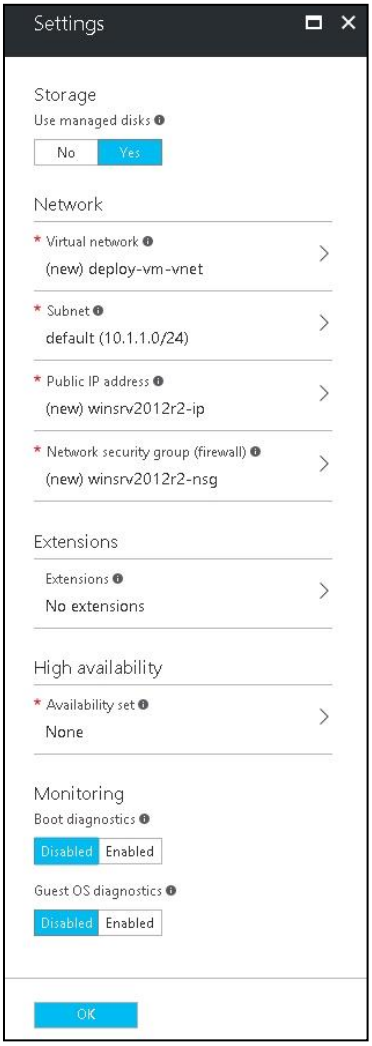


FIGURE 59. VM SETTINGS

In this example, we use the following settings

- Set **Use managed disks** to Yes. This is the current recommended way of deploying disks for an Azure VM as the management and availability of the disk is managed by Azure.
- **Virtual network**, leave the default VNet that is created, noting that you do not have to use it and can create your own instead.
- **Subnet**, leave the default Subnet that is created, noting that you do not have to use it and can create your own instead.

- **Public IP address**, leave the default public IP address that is created, noting that you do not have to use it and can create your own instead.

> **Note:** For most installations, you will not assign a public IP to a VM. You will assign the IP address to the Azure Load balancer and create a NAT rule.

- **Network Security Group (firewall)**, leave the default network security group that is created, noting that you do not have to use it and can create your own instead.

> **Note:** Network security groups when assigning public IP addresses are essential as they filter the incoming public traffic and stop malicious connections. If this was an internal only deployment, you may choose to turn off NSG for the VM and apply it to the subnet on which the VM will be located

- **Extensions**, leave this blank.
- **High Availability**, leave this blank.
- **Monitoring**, set the Boot diagnostics and the Guest OS diagnostics to **disabled**.
- When you are finished, click **OK**.

8. In the **Summary** blade, as shown in Figure 30, wait for the validation to complete, confirm the settings and then click **OK**.

> Note that next to the **OK** button, you can click on **Download template and parameters** to save the deployment as an ARM Template.

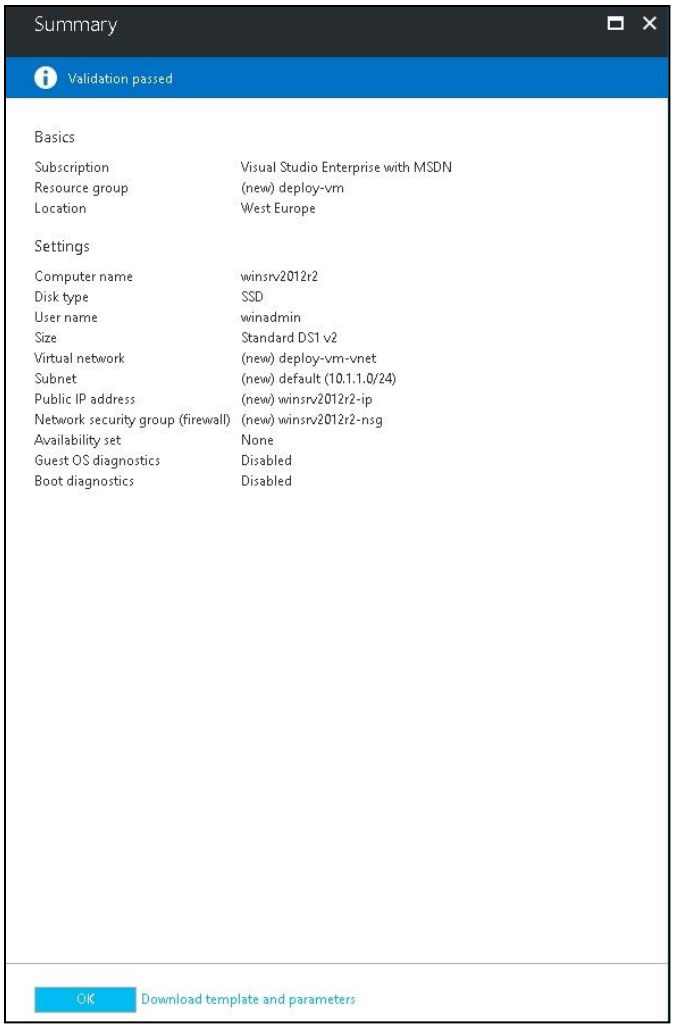


FIGURE 60. VIRTUAL MACHINE SUMMARY OF DEPLOYMENT

The VM will now deploy. Deployment time varies from 15 minutes to 45 minutes depending on the workload and deployment options selected. Watch the **Notifications** area of the Azure Portal for a message indicating whether your deployment was successful as shown in Figure 31.

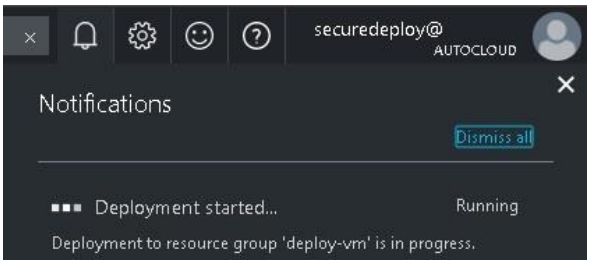


FIGURE 61. AZURE NOTIFICATIONS - VM DEPLOYMENT

To create a Linux VM from the Portal, perform the following steps:

1. From the ARM Portal located at https://portal.azure.com **Click + NEW** as shown in Figure 32.

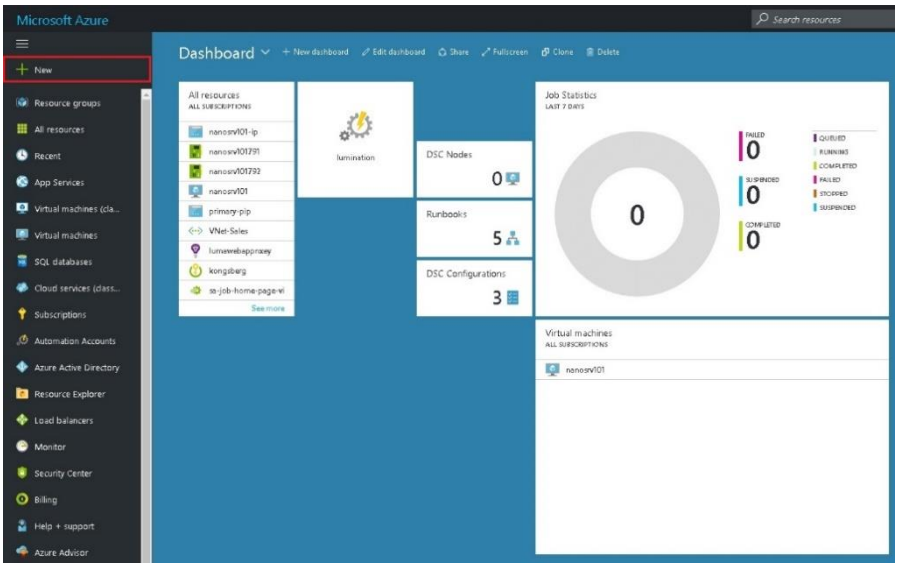


FIGURE 62. NEW

2. From the **New** blade all the options for deployment are listed, for a VM we need to select **Compute** as shown in figure 33.

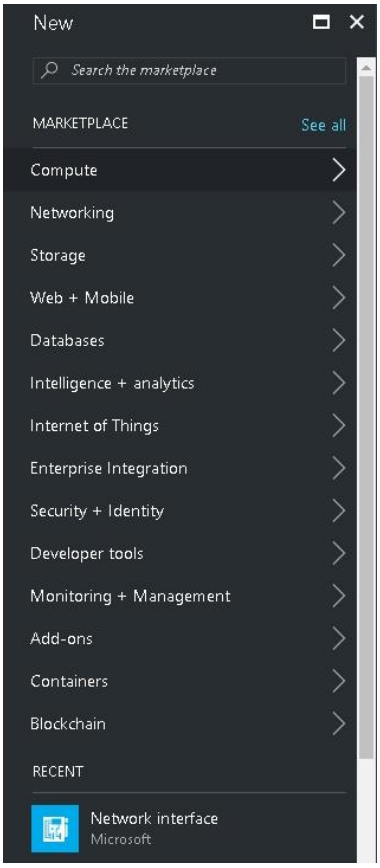


FIGURE 63. SELECT COMPUTE UNDER MARKETPLACE

3. In the **Compute** blade, the most popular images are presented for use. The Azure Marketplace gives an area where all the images available (Microsoft and 3rd Party published) in Azure can be selected for Installation. Click **Ubuntu Server 16.04 LTS** as shown in Figure 34.

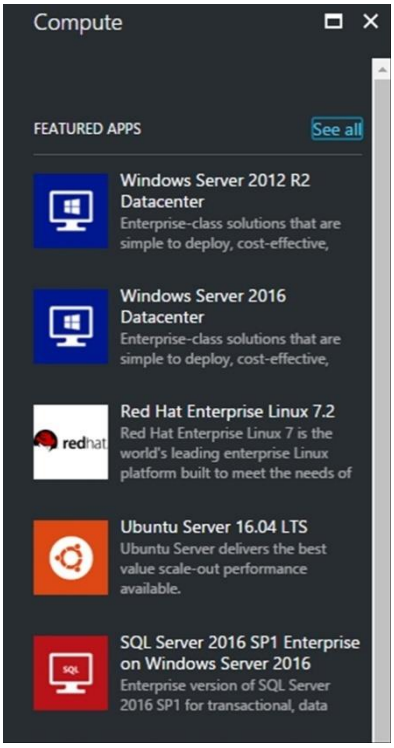


FIGURE 64. UBUNTU SERVER 16.04 LTS

4. After clicking on the Ubuntu Server 16.04 LTS image, it will open up a summary blade describing the image and ask you to confirm which deployment model you wish to use; Classic being the Azure Service Management and Resource Manager being ARM. **Select Resource Manager** and **Click Create** as shown in Figure 35.

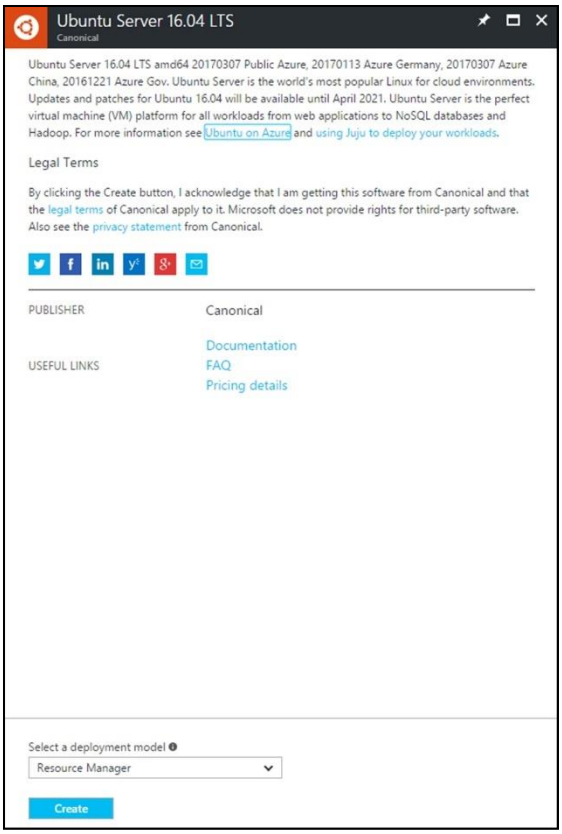


FIGURE 65. DEPLOYMENT MODEL

5. In the **Create Virtual Machine** blade, the basics blade will automatically open as shown in Figure 36.

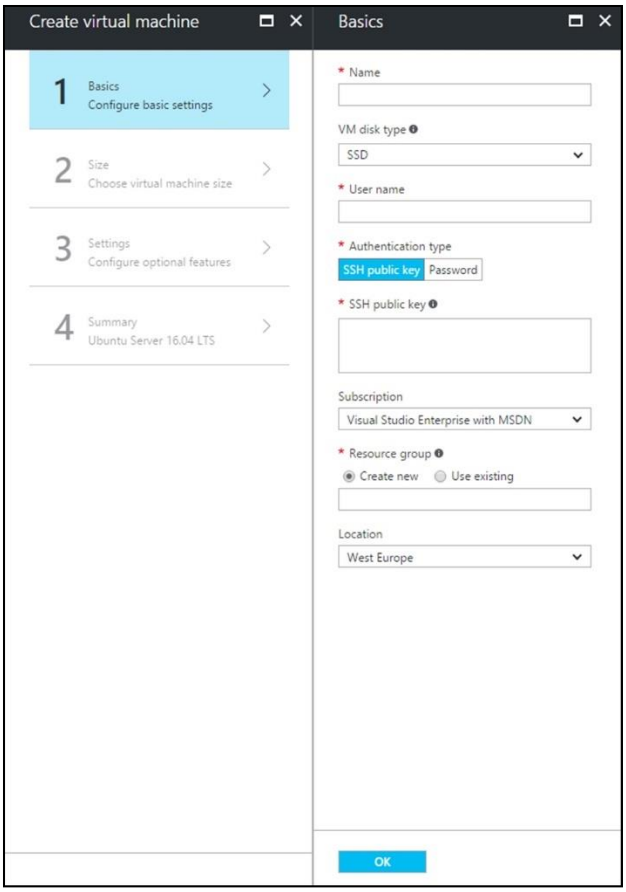


FIGURE 66. CREATE VM

Enter values for the following fields shown in the **Basic** blade:

- **Name** (which is the name of the VM)
- **VM Disk Type** (**HDD** for Standard Disk, **SSD** for Premium Disk)
- **Username** (administrator and root are not allowed)
- **Authentication Type** is set to **SSH public key** by default. However, for this demo, set the authentication type to **Password**. You can select SSH Public Key as shown in Figure 37 and enter the public key you have to authentication.

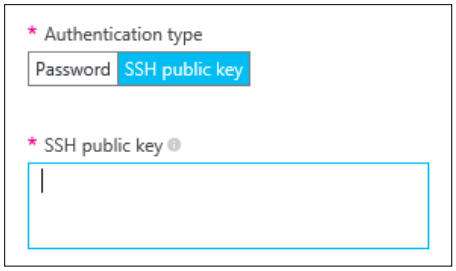


FIGURE 67. SSH PUBLIC KEY

- **Password** (complex rules apply)
- **Confirm Password** (complex rules apply)
- For **Resource Group** the field is by default setup to create a new resource group, if you enter a new name, it will create a new resource group and all the VM components will be associated with it. Alternatively, you can click **select existing** and select a resource group that has already been deployed. For this example, type in **deploy-linux-vm** in the **resource group** field.
- For **Location**, select the location where you want to deploy to, in this example choose **West Europe**.

When you are finished, click **OK.**

9. Next, the **Choose a Size** blade will open, which by default will have recommended options for the type of image you have chosen. In Figure 38, you can see we have two recommended options listed. If you would like to view all of the possible options available to you, click on **View all**. Within each option, you should also see the estimated monthly cost of running the VM. For this example, click on **DS1_V2 Standard** and then click **Select**.

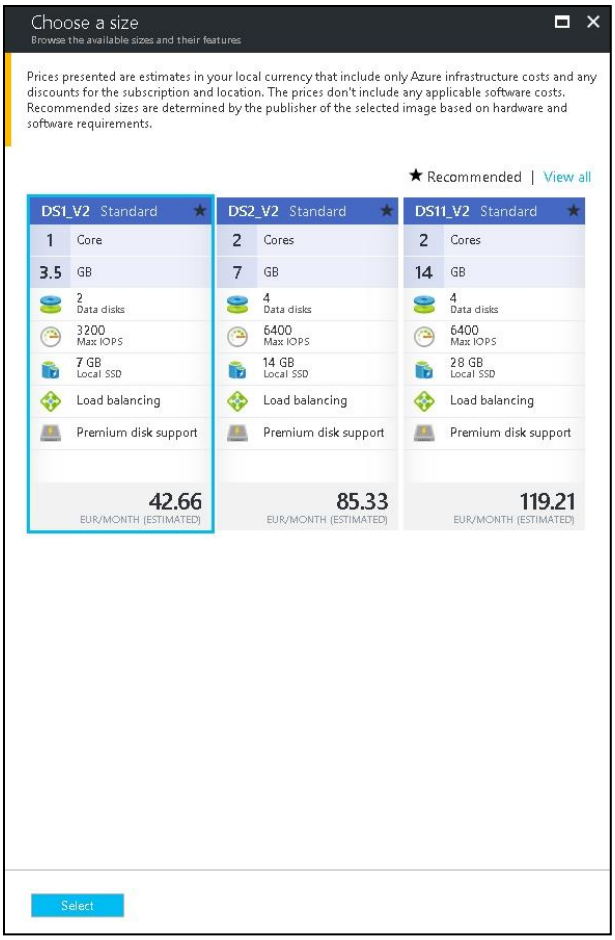


FIGURE 68. VM SIZE SELECTION

6. Next, we will configure VM settings in the **Settings** blade as shown in figure 39.

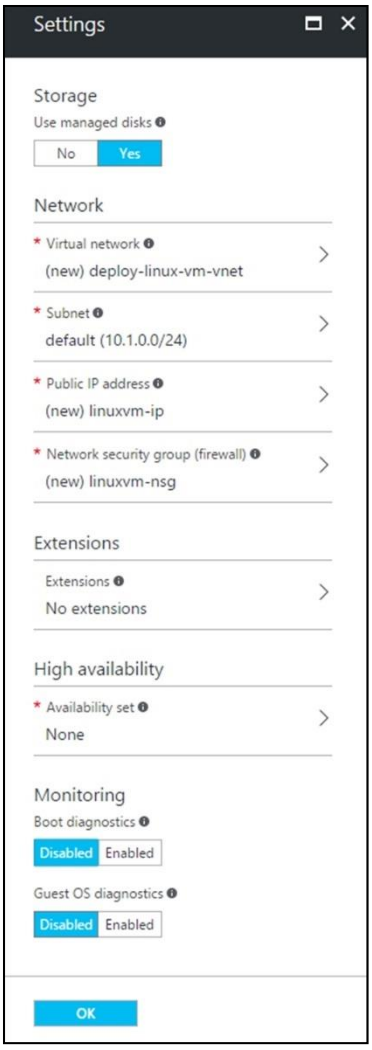


FIGURE 69. VM SETTINGS

In this example, we use the following settings

- Set **Use managed disks** to Yes. This is the current recommended way of deploying disks for an Azure VM as the management and availability of the disk is managed by Azure.
- **Virtual network**, leave the default VNet that is created, noting that you do not have to use it and can create your own instead.
- **Subnet**, leave the default Subnet that is created, noting that you do not have to use it and can create your own instead.

- **Public IP address**, leave the default public IP address that is created, noting that you do not have to use it and can create your own instead.

> **Note:** For most installations, you will not assign a public IP to a VM. You will assign the IP address to the Azure Load balancer and create a NAT rule.

- **Network Security Group (firewall)**, leave the default network security group that is created, noting that you do not have to use it and can create your own instead.

> **Note:** Network security groups when assigning public IP addresses are essential as they filter the incoming public traffic and stop malicious connections. If this was an internal only deployment, you may choose to turn off NSG for the VM and apply it to the subnet on which the VM will be located

- **Extensions**, leave this blank.
- **High Availability**, leave this blank.
- **Monitoring**, set the Boot diagnostics and the Guest OS diagnostics to **disabled**.
- When you are finished, click **OK**.

7. In the **Summary** blade, as shown in Figure 40, wait for the validation to complete, confirm the settings and then click **OK**.

> Note that next to the **OK** button, you can click on **Download template and parameters** to save the deployment as an ARM Template.

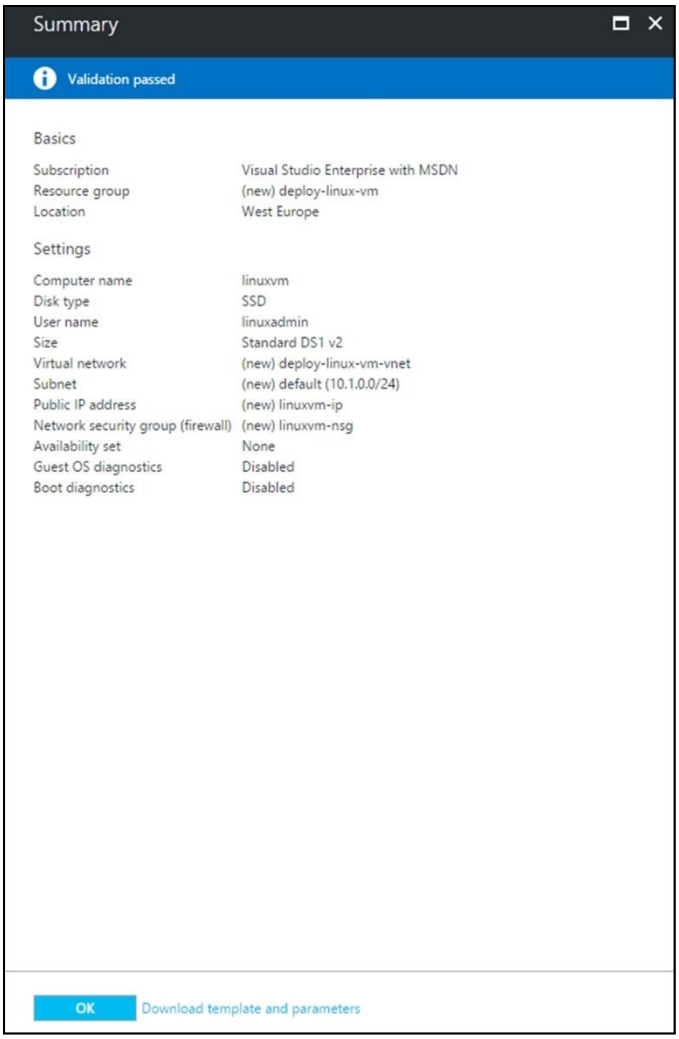


FIGURE 70. VIRTUAL MACHINE SUMMARY OF LINUX VM DEPLOYMENT

The VM will now deploy. Deployment time varies from 15 minutes to 45 minutes depending on the workload and deployment options selected. Watch the **Notifications** area of the Azure Portal for a message indicating whether your deployment was successful as shown in Figure 41.

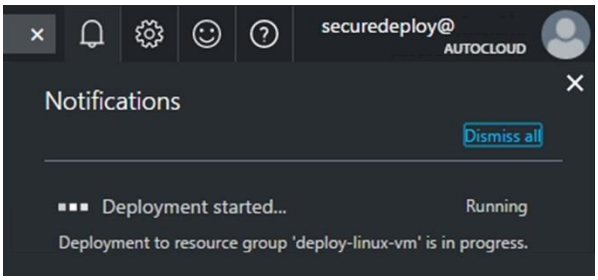


FIGURE 71. AZURE NOTIFICATIONS - VM DEPLOYMENT

> **Note:** You can verify that Linux VM is accessible, post deployment, using an SSH client like Putty to connect to it using it's public IP address. You can download the Putty SSH client free at http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

Deploying a Virtual Machine from PowerShell (Windows and Linux)

Now that you have completed a couple of VM deployments in the Azure Portal, you are ready to try your hands at VM deployment through PowerShell. You will deploy both a Windows VM and a Linux VM.

> **Note:** You can download the sample PowerShell scripts for this section (and entire chapter) using the link provided under the "Download the Code" heading at the end of this section.

Example 1: Create a Windows VM from Azure PowerShell

In this section, we will go through deploying a Windows Server 2012 R2 VM in Azure Resource Manager deployed to an unmanaged disk using Azure PowerShell.

> **Note**: managed disks in Azure do not require an additional storage account to be deployed whereas an unmanaged disk does. The full benefits of using managed vs unmanaged disks is outside the scope of this chapter; however, the supplemental scripts for this chapter included options to deploy to both types for thoroughness. For more information about the benefits of managed disks, please

visit https://docs.microsoft.com/en-us/azure/storage/storage-managed-disks-overview.

After you have logged into a PowerShell Session and selected the Azure subscription to work with, you can begin provisioning resources into Azure. The following PowerShell commands illustrate how you can provision a new VM into Azure along with its required prerequisites.

Note: You will notice a lot of backtick characters (`) in the following code. Do not forget to copy them as they are responsible for allowing the longer commands to wrap into the next line, making it easier to read in the format this book is provided in.

1. First, start off by configuring the following variables. Change the **location** variable to your particular region you are deploying to, i.e. - **northeurope** for North Europe, **eastus** for East Coast (USA).

    - virtual machine name
    - virtual machine size
    - resource group name
    - storage account name
    - storage account type
    - local administrator username
    - local administrator password
    - Location

```
$VMName             = "iaas-vm"
$VMSize             = "Standard_DS1_v2"
$ResourceGroupName  = "iaas-vm-demo"
$StorageAccountName = "iaasvmdemo"
$StorageAccountType = "Standard_LRS"
$LocalAdminUsername = "localadmin"
$LocalAdminPassword = "LetMeInNow1!"
$Location           = "westeurope"
```

Note: It is recommended that you append the **VMName** and **StorageAccountName** variables with additional numbers and characters to ensure they are unique within Azure. For example:

VMName = "iaas-vmt4r5"
StorageAccountName = "iaasvmdemot4r5"

The reason the VMName needs to be unique in this scenario is that the Public IP Address that is being deployed is based on the Name of the VM.

Additionally, in production environments, it is never recommended to have your credentials passed in clear text, but as this is to illustrate concepts, we have left them as such.

2. Next, we'll deploy a new Resource Group for the Virtual Machine and its related resources.

```
New-AzureRmResourceGroup `
    -Name $ResourceGroupName `
    -Location $Location
```

3. Next, we need to deploy a Storage Account to store the VM OS Disk. This will take a minute or so to finish deploying.

```
$VMStorageAccount = New-AzureRmStorageAccount `
    -ResourceGroupName $ResourceGroupName `
    -AccountName $StorageAccountName `
    -Location $Location `
    -Type $StorageAccountType
```

4. Next, we need to retrieve the latest VM Image version of Windows Server 2012 R2 Datacenter available in the location you are deploying to.

```
$VMImage = Get-AzureRmVMImage `
    -Location $Location `
    -PublisherName MicrosoftWindowsServer `
    -Offer WindowsServer `
    -Skus 2012-R2-Datacenter `
    | Sort-Object Version -Descending `
    | Select-Object -First 1
```

5. Next, create a new VM Configuration.

```
$VMConfig = New-AzureRMVMConfig `
    -VMName $VMName `
    -VMSize $VMSize
```

6. Next, create a Subnet.

```
$Subnet = New-AzureRmVirtualNetworkSubnetConfig `
    -Name Subnet-$ResourceGroupName `
    -AddressPrefix "10.0.0.0/24"
```

7. Next, create a VNet. Ignore any warning output from the command.

```
$VNet = New-AzureRmVirtualNetwork `
    -Name VNet-$ResourceGroupName `
    -AddressPrefix $Subnet.AddressPrefix `
    -Subnet $Subnet `
    -ResourceGroupName $ResourceGroupName `
    -Location $Location
```

8. Next, create a Public IP Address which will be used for RDP Access to the VM. Ignore any warning output from the command.

```
$PublicIP = New-AzureRmPublicIpAddress `
    -Name pip-$VMName `
    -ResourceGroupName $ResourceGroupName `
    -AllocationMethod Dynamic `
    -DomainNameLabel "$VMName-pip" `
    -Location $Location
```

9. Next, create a NIC Card. Ignore any warning output from the command.

```
$NIC = New-AzureRmNetworkInterface `
    -Name "$VMName-NIC" `
    -ResourceGroupName $ResourceGroupName `
    -SubnetId $VNet.Subnets[0].id `
    -PublicIpAddressId $PublicIP.Id `
    -Location $Location
```

10. Next, associate the NIC Card to the VM.

```
$VM = Add-AzureRmVMNetworkInterface `
    -VM $VMConfig `
    -Id $NIC.Id
```

11. Next, configure the OS Disk Settings.

```
$OSDiskName    = "$VMName" + "_OS_Disk"
$OSDiskCaching = "ReadWrite"
$OSDiskVhdUri  =
"https://$StorageAccountName.blob.core.windows.net/vhds/$OSDiskName.vhd"
```

12. Next, add the local administrator credentials to a PSCredential Object.

```
$SecureLocalAdminPassword = ConvertTo-SecureString $LocalAdminPassword -
AsPlainText -Force
$Credentials = New-Object System.Management.Automation.PSCredential
($LocalAdminUsername, $SecureLocalAdminPassword)
```

13. Next, set the VM Operating System Settings.

```
$VM = Set-AzureRmVMOperatingSystem `
    -VM $VM `
    -Windows `
    -ComputerName $VMName `
    -Credential $Credentials
```

14. Next, setting the VM Source Image for the VM.

```
$VM = Set-AzureRmVMSourceImage `
    -VM $VM `
    -PublisherName $VMImage.PublisherName `
    -Offer $VMImage.Offer `
    -Skus $VMImage.Skus `
    -Version $VMImage.Version
```

15. Next, setting the Operating System Disk settings for the VM

```
$VM = Set-AzureRmVMOSDisk `
    -VM $VM `
    -VhdUri $OSDiskVhdUri `
    -Name $OSDiskName `
    -CreateOption FromImage `
    -Caching $OSDiskCaching
```

16. Finally, deploying the Azure VM.

```
New-AzureRmVM `
    -ResourceGroupName $ResourceGroupName `
    -Location $Location `
    -VM $VM `
```

17. Once the VM has finished being deployed, you should get the following response back as shown in Figure 42.

```
RequestId IsSuccessStatusCode StatusCode ReasonPhrase
--------- ------------------- ---------- ------------
                         True         OK OK
```

FIGURE 72. WINDOWS VM DEPLOYMENT COMPLETED

At this point, you have the option to use the DNS Name of the Public IP Address you deployed earlier to RDP into the VM using the credentials from in the initial variables.

Once you have finished exploring the deployed resources, it is recommended that you delete the Resource Group they were deployed to.

> **Note**: You can find this script in the GitHub Repository https://github.com/insidemscloud/AzureIaaSBook in the **Chapter 6** directory. The file is called **deploy-unique-windows-vm-to-azure-unmanaged-disk.ps1**.

The difference between the script you'll find on GitHub and the walkthrough you just completed is that the script on GitHub is fully parameterized, contains detailed information on how to use the script and includes extensive error handling to help you troubleshoot the script if it fails.

Additionally, this script ensures that the resources that are deployed into Azure are unique by appending 4 alphanumeric characters to the end of each resource from a GUID that is generated at runtime.

Example 2: Create a Linux VM from Azure PowerShell

In this section, we will go through deploying an Ubuntu Server 16.04 LTS VM in Azure Resource Manager deployed to an unmanaged disk using Azure PowerShell.

> **Note**: managed disks in Azure do not require an additional storage account to be deployed whereas an unmanaged disk does. The full benefits of using managed vs unmanaged disks is outside the scope of this chapter; however, the supplemental scripts for this chapter included options to deploy to both types for thoroughness. For more information about the benefits of managed disks, please visit: https://docs.microsoft.com/en-us/azure/storage/storage-managed-disks-overview

After you have logged into Azure from an elevated PowerShell Prompt and selected the Azure subscription to work with, you can begin provisioning resources into Azure. The following PowerShell commands illustrate how you can provision a new VM into Azure along with its required prerequisites.

Note: You will notice a lot of backtick characters (`) in the following code. Do not forget to copy them as they are responsible for allowing the longer commands to wrap into the next line, making it easier to read in the format this book is provided in.

1. First, start off by configuring the following variables. Change the **location** variable to your particular region you are deploying to, i.e. - **northeurope** for North Europe, **eastus** for East Coast (USA).

    - virtual machine name
    - virtual machine size
    - resource group name
    - storage account name
    - storage account type
    - local administrator username
    - local administrator password
    - Location

```
$VMName              = "iaas-linux-vm"
$VMSize              = "Standard_DS1_v2"
$ResourceGroupName   = "iaas-linux-vm-demo"
$StorageAccountName  = "iaasvmlinuxdemo"
$StorageAccountType  = "Standard_LRS"
$LocalAdminUsername  = "linuxadmin"
$LocalAdminPassword  = "LetMeInNow1!"
$Location            = "westeurope"
```

Note: It is recommended that you append the **VMName** and **StorageAccountName** variables with additional numbers and characters to ensure they are unique within Azure. For example:

VMName = "iaas-linux-vmt4r5"
StorageAccountName = "iaaslinuxvmdemot4r5"

The reason the VMName needs to be unique in this scenario is that the Public IP Address that is being deployed is based on the Name of the VM.

Additionally, in production environments, it is never recommended

to have your credentials passed in clear text, but as this is to illustrate concepts, we have left them as such.

2. Next, we'll deploy a new Resource Group for the Virtual Machine and its related resources.

```
New-AzureRmResourceGroup `
    -Name $ResourceGroupName `
    -Location $Location
```

3. Next, we need to deploy a Storage Account to store the VM OS Disk. This will take a minute or so to finish deploying.

```
$VMStorageAccount = New-AzureRmStorageAccount `
    -ResourceGroupName $ResourceGroupName `
    -AccountName $StorageAccountName `
    -Location $Location `
    -Type $StorageAccountType
```

4. Next, we need to retrieve the latest VM Image version of Ubuntu Server 16.04 LTS available in the location you are deploying to.

```
$VMImage = Get-AzureRmVMImage `
    -Location $Location `
    -PublisherName Canonical `
    -Offer UbuntuServer `
    -Skus 16.04-LTS `
    | Sort-Object Version -Descending `
    | Select-Object -First 1
```

5. Next, create a new VM Configuration.

```
$VMConfig = New-AzureRMVMConfig `
    -VMName $VMName `
    -VMSize $VMSize
```

6. Next, create a Subnet.

```
$Subnet = New-AzureRmVirtualNetworkSubnetConfig `
    -Name Subnet-$ResourceGroupName `
    -AddressPrefix "10.0.0.0/24"
```

7. Next, create a VNet. Ignore any warning output from the command.

```
$VNet = New-AzureRmVirtualNetwork `
    -Name VNet-$ResourceGroupName `
```

```
    -AddressPrefix $Subnet.AddressPrefix `
    -Subnet $Subnet `
    -ResourceGroupName $ResourceGroupName `
    -Location $Location
```

8. Next, create a Public IP Address which will be used for RDP Access to the VM. Ignore any warning output from the command.

```
$PublicIP = New-AzureRmPublicIpAddress `
    -Name pip-$VMName `
    -ResourceGroupName $ResourceGroupName `
    -AllocationMethod Dynamic `
    -DomainNameLabel "$VMName-pip" `
    -Location $Location
```

9. Next, create a NIC Card. Ignore any warning output from the command.

```
$NIC = New-AzureRmNetworkInterface `
    -Name "$VMName-NIC" `
    -ResourceGroupName $ResourceGroupName `
    -SubnetId $VNet.Subnets[0].id `
    -PublicIpAddressId $PublicIP.Id `
    -Location $Location
```

10. Next, associate the NIC Card to the VM.

```
$VM = Add-AzureRmVMNetworkInterface `
    -VM $VMConfig `
    -Id $NIC.Id
```

11. Next, configure the OS Disk Settings.

```
$OSDiskName     = "$VMName" + "_OS_Disk"
$OSDiskCaching = "ReadWrite"
$OSDiskVhdUri  =
"https://$StorageAccountName.blob.core.windows.net/vhds/$OSDiskName.vhd"
```

12. Next, add the local administrator credentials to a PSCredential Object.

```
$SecureLocalAdminPassword = ConvertTo-SecureString $LocalAdminPassword -
AsPlainText -Force
$Credentials = New-Object System.Management.Automation.PSCredential
($LocalAdminUsername, $SecureLocalAdminPassword)
```

13. Next, set the VM Operating System Settings.

```
$VM = Set-AzureRmVMOperatingSystem `
    -VM $VM `
    -Linux `
    -ComputerName $VMName `
```

```
    -Credential $Credentials
```

14. Next, setting the VM Source Image for the VM.

```
$VM = Set-AzureRmVMSourceImage `
    -VM $VM `
    -PublisherName $VMImage.PublisherName `
    -Offer $VMImage.Offer `
    -Skus $VMImage.Skus `
    -Version $VMImage.Version
```

15. Next, setting the Operating System Disk settings for the VM

```
$VM = Set-AzureRmVMOSDisk `
    -VM $VM `
    -VhdUri $OSDiskVhdUri `
    -Name $OSDiskName `
    -CreateOption FromImage `
    -Caching $OSDiskCaching
```

16. Finally, deploying the Azure VM.

```
New-AzureRmVM `
    -ResourceGroupName $ResourceGroupName `
    -Location $Location `
    -VM $VM
```

17. Once the VM has finished being deployed, you should get the following response back as shown in Figure 43.

```
RequestId IsSuccessStatusCode StatusCode ReasonPhrase
--------- ------------------- ---------- ------------
                         True         OK OK
```

FIGURE 73. LINUX VM DEPLOYMENT COMPLETED

At this point, you have the option to use the DNS Name of the Public IP Address you deployed earlier to SSH into the VM using the credentials from in the initial variables.

Once you have finished exploring the deployed resources, it is recommended that you delete the Resource Group they were deployed to.

> **Note**: You can find this script in the GitHub Repository https://github.com/insidemscloud/AzureIaaSBook in the **Chapter 6**

directory. The file is called **deploy-unique-linux-vm-to-azure-unmanaged-disk.ps1**.

The difference between the script you'll find on GitHub and the walkthrough you just completed is that the script on GitHub is fully parameterized, contains detailed information on how to use the script and includes extensive error handling to help you troubleshoot the script if it fails.

Additionally, this script ensures that the resources that are deployed into Azure are unique by appending 4 alphanumeric characters to the end of each resource from a GUID that is generated at runtime.

Deploying additional disks

In this section, you will add an additional unmanaged disk to the Windows you deployed in the previous section through the Azure Portal. Note that this process is identical for a Linux VM.

To add an additional disk to the Windows VM you deployed in the previous section through the Azure Portal, perform the following steps:

1. Open a browser and navigate to https://portal.azure.com
2. In the left-hand menu, click **Virtual Machines** as shown in Figure 44.

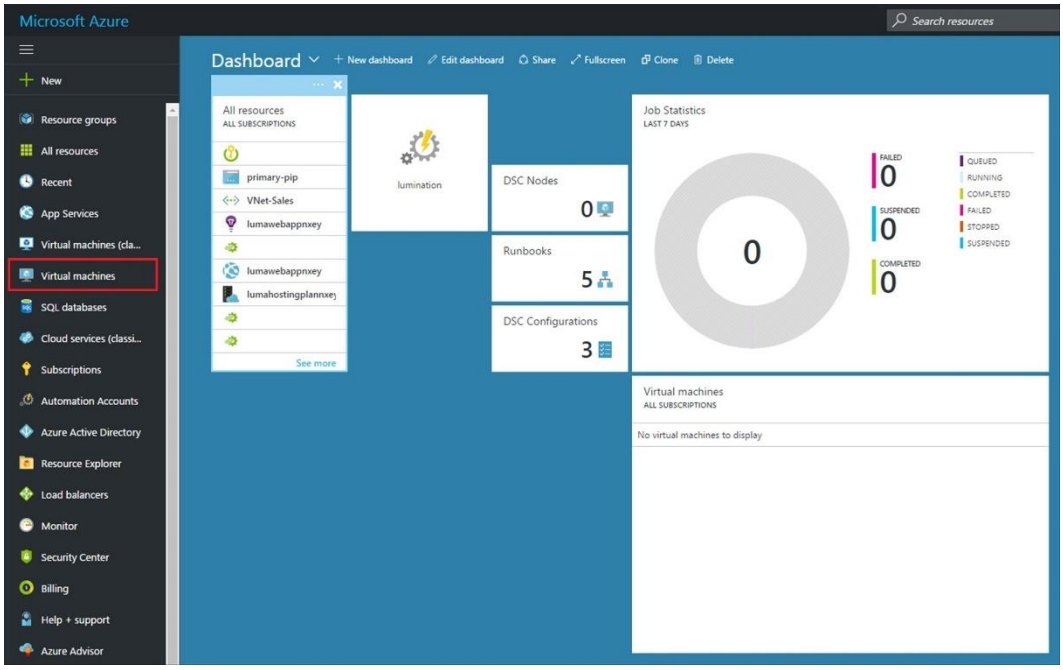


FIGURE 74. BROWSING TO VIRTUAL MACHINES

3. In the **Virtual Machines** Blade click on **iaas-vm** as shown in Figure 45

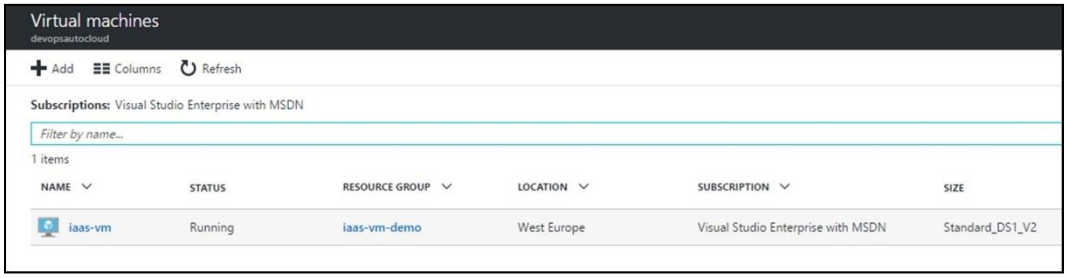


FIGURE 75. SELECT IAAS-VM VIRTUAL MACHINE

4. This will open the **iaas-vm Virtual Machine** Blade, Click **Disks** as shown in Figure 46.

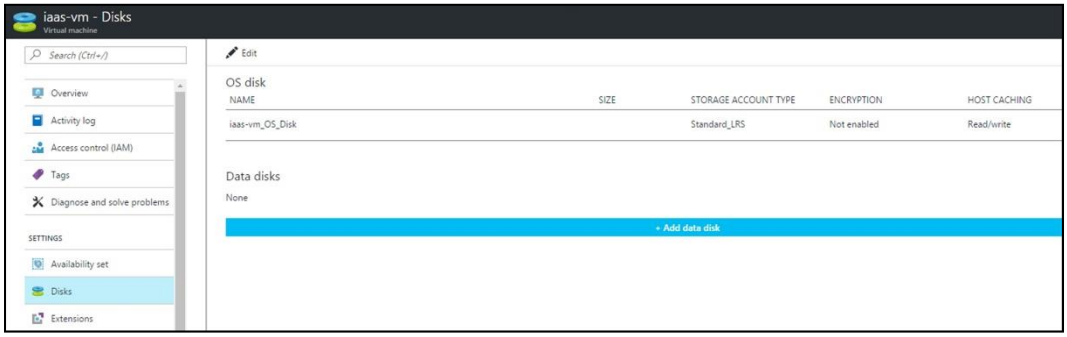


FIGURE 76. DISKS BLADE

5. In the **Disks** blade, click **+ Add data disk** and the **Attach unmanaged disk** blade will open up as shown in figure 47.

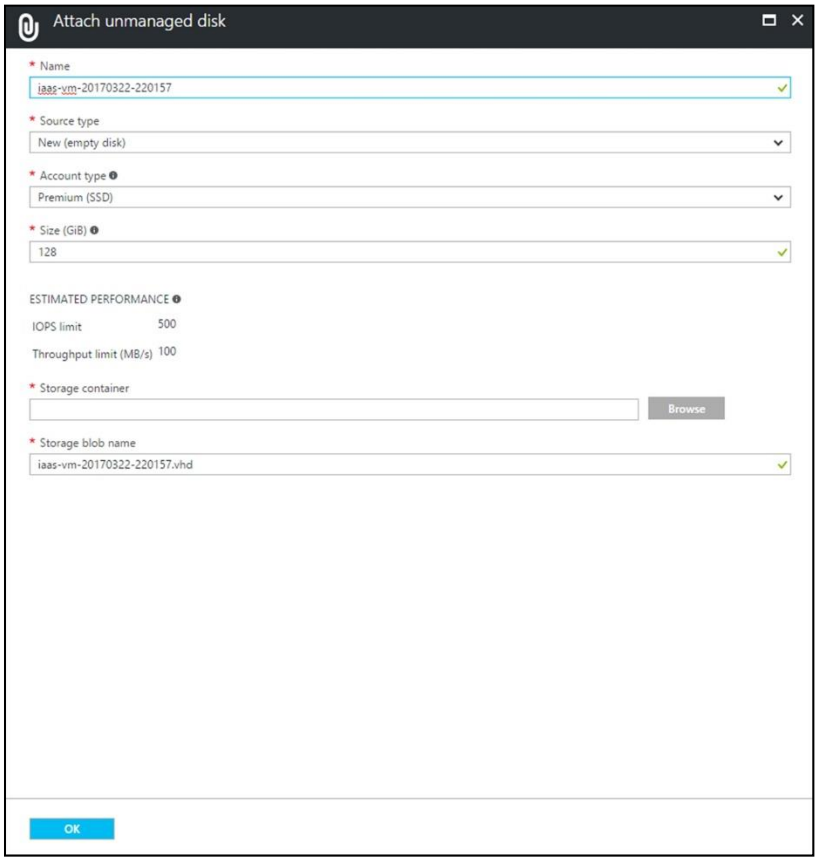


FIGURE 77. ATTACH UNMANAGED DISK BLADE

Use the following settings for the new unmanaged disk:

- **Name**, leave the default value that appears for you. It will be different each time you add a new disk. You can use your own naming convention as well.
- **Source type**, leave the default value of **New (empty disk)**.
- **Account Type**, you can either choose Premium (SSD) or Standard (HDD). For this demo use **Standard (HDD)**.
- **Size (GiB)**, the size of the disk in Gigabytes from 1 to 1024. The recommended value here for this demo is **128**.
- **Storage container**, click on Browse and choose **iaasvmdemo** as shown in Figure 48 below.

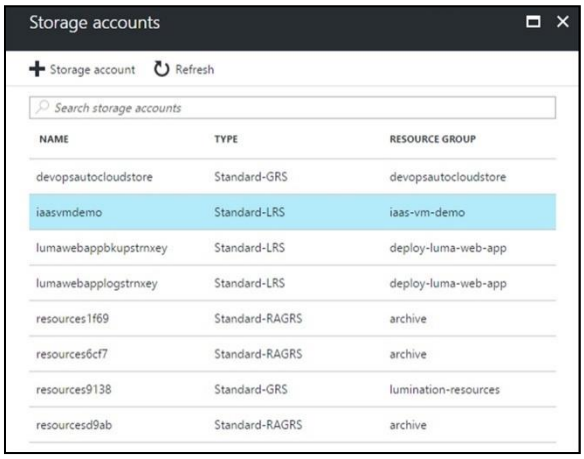


FIGURE 78. IAASVMDEMO STORAGE ACCOUNT

- Next, in **Containers**, click on **vhds** and then click on the **Select** button as shown in Figure 49.

FIGURE 79. ATTACHING NEW DISKS

- **Storage blob name**, leave the default value as is. You can use your own naming convention as well.

6. When you are finished filling out the values, click **OK**.

Back on the **Disks** Blade, click **Save** as shown in Figure 50.

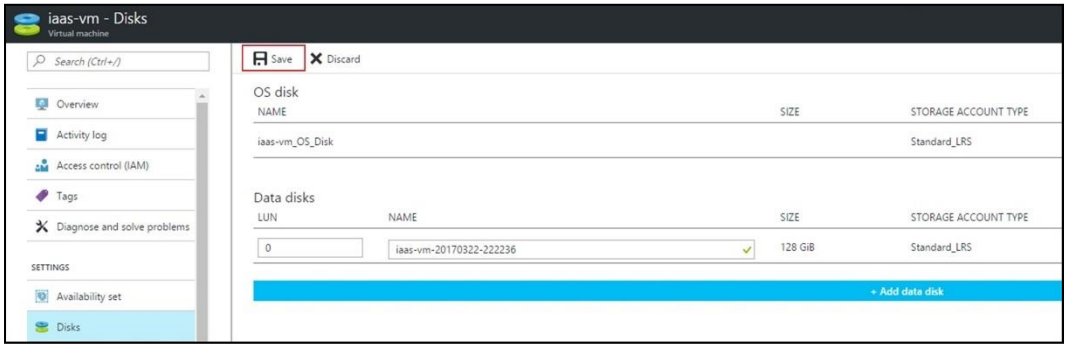


FIGURE 80. SAVING THE DISK CONFIGURATION

7. Under the Notifications area, you'll see the virtual machine disks being updated as shown in figure 7.3.36. The process should complete in a minute or so.

FIGURE 81. ATTACHING NEW DISKS

> **Note:** If you have existing virtual machines in your environment that are using unmanaged disks, you can convert them to be able to use managed disks using the following steps located in Microsoft's documentation here: https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-windows-convert-unmanaged-to-managed-disks.

Creating a NAT Rule for an existing Virtual Machine

There are many deployments where you will not assign a public IP to your VM directly. Previously, in the Service Management Portal, you would achieve this functionality using endpoints and cloud services. These created a NAT rule to allow you to connect to your VM. In ARM, you create all endpoints as part of the Azure load balancer. We will walk through the steps to create a simple rule for traffic on Port 80 (HTTP) into a VM. Note that

these steps are written to work with the Windows VM that was previously deployed; however, they can be modified to work with the Linux VM as well.

As with previous examples, a sample script is available for download as detailed in the "Download the Code" at the end of this tutorial.

To create an Azure load balancer, a NAT rule, and associate to an existing Windows VM, utilize the following steps and PowerShell sample:

1. First, create a new public IP address.

```
$VIP = New-AzureRmPublicIPaddress `
        -ResourceGroupName "iaas-vm-demo" `
        -Name "vm-nat-pub-ip" `
        -Location "West Europe" `
        -AllocationMethod Dynamic
```

2. Next, create a front end IP configuration for the load balancer and use the public IP address to bind to it.

```
$FrontEndIPConfig = New-AzureRmLoadBalancerFrontEndIPConfig `
    -Name "LBFrontEndIP" `
    -PublicIpAddress $VIP
```

3. Next, create the Inbound NAT Rule that will translate secure web traffic from port 443 to Port 80.

```
$HttpNATRule = New-AzureRmLoadBalancerInboundNatRuleConfig `
    -Name "http-nat-rule" `
    -FrontEndIPConfiguration $FrontEndIPConfig `
    -Protocol TCP `
    -FrontEndPort 443 `
    -BackendPort 80
```

4. Next, give the load balancer the backend subnet where the Windows VM will reside on.

```
$LBBEPool = New-AzureRmLoadBalancerBackEndAddressPoolConfig `
    -Name "BEPool01"
```

5. Next, create the load balancer Rule.

```
$LBRule = New-AzureRmLoadBalancerRuleConfig `
    -Name "http-lb-rule" `
    -FrontEndIPConfiguration $FrontEndIPConfig `
    -BackEndAddresspool $LBBEPool `
    -Protocol TCP `
    -FrontEndPort 80 `
    -BackEndPort 80
```

6. Next, create a health probe rule to check the availability of the Windows VM instance in the back-end address pool.

```
$HealthProbe = New-AzureRmLoadBalancerProbeConfig `
    -Name HealthProbe `
    -Protocol Tcp `
    -Port 80 `
    -IntervalInSeconds 15 `
    -ProbeCount 1
```

7. Next, create the load balancer itself and use the rules and items we configure as base items.

```
$AzureLB = New-AzureRmLoadBalancer `
    -ResourceGroupName "iaas-vm-demo" `
    -Name "iaas-vm-lb" `
    -Location "west Europe" `
    -FrontendIpConfiguration $FrontEndIPConfig `
    -InboundNatRule $HttpNATRule `
    -LoadBalancingRule $LBRule `
    -BackendAddressPool $LBBEPool `
    -Probe $HealthProbe
```

8. Next, associate the Windows VM network interface to the load balancer rule using the following code snippets.

```
$VNet = Get-AzureRmVirtualNetwork `
    -Name “VNet-iaas-vm-demo” `
    -ResourceGroupName “iaas-vm-demo”

$Subnet = Get-AzureRmVirtualNetworkSubnetConfig `
    -VirtualNetwork $VNet `
    -Name “Subnet-iaas-vm-demo”

$NIC = Get-AzureRmNetworkInterface `
    -Name “iaas-vm-NIC” `
    -ResourceGroupName “iaas-vm-demo”

$NIC.ipconfigurations[0].LoadBalancerBackendAddressPools.Add($A
zureLB.backendaddresspools[0])

$NIC | Set-AzureRmNetworkInterface
```

9. After the association has been set, you can verify it by checking the **Backend pools** blade under the **iaas-vm-lb** load balancer resource as shown in Figure 52.

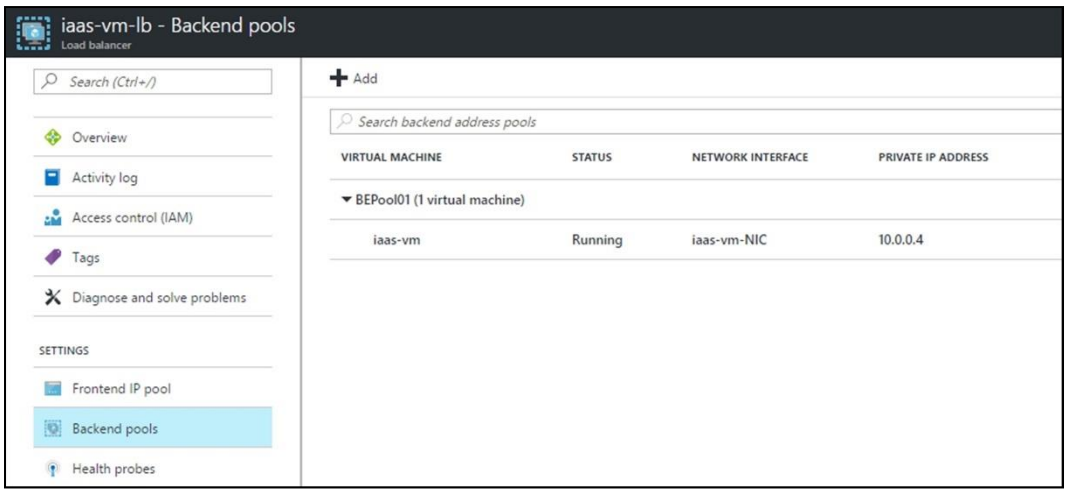


FIGURE 82. BACKEND POOLS AND WINDOWS VM NETWORK INTERFACE ASSOCIATION

Download the Code

You can download the full script from GitHub at https://github.com/insidemscloud/AzureIaasBook, in the **Chapter 6** directory. The file name is **add-nat-rule-to-vm.ps1**.

ARM Deployment Templates

As mentioned in the first section of this chapter when you create a template you can define the resources you require in azure, the order of installation and even call VM Extensions inside the VM for additional configuration and deployment of an application. Deployment templates are designed for repeatable and consistent deployment of applications/workloads to Microsoft cloud infrastructures.

It will help your understanding of the sections that follow if we first describe the ARM template schema.

The schema of the JSON file required is displayed below.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [],
  "outputs": {}
}
```

To give you a better understanding of the schema, a description of each section of the JSON file is outlined below.

- **$schema** - $schema is a required section of the JSON file and it points to the location of the JSON schema file. This file will describe the version of the template language.

- **contentVersion** - contentVersion is a required section of the JSON file. The version number is in the format of X.X.X.X ; For example, 1.0.0.0. The version number allows you to ensure that you can select the right template.

- **parameters** - parameters are not a required section of the JSON file. Parameters are useful elements as they allow you to customize a template before deployment. For example, if you want to have a template which allows you to give a VM a specific naming convention, you can set a parameter for the VM Name.

  The Format of the parameters section is below.

  ```
  "parameters": {
    "<parameterName>": {
      "type": "<type-of-parameter-value>",
      "defaultValue": ":<optional-default-value-of-parameter>"
      "allowedValues": [ "<optional-array-of-allowed-values>" ]
    }
  }
  ```

The **`parameterName`** and **`type`** are required.

`Type` can be set to one of the following values

- string or secureString
- int
- bool
- object or secureObject
- array

A sample parameters section for setting the deployment location is below.

```
"parameters": {
  "Location": {
    "type": "String",
    "defaultValue": "North Europe"
    "allowedValues": [
      "North Europe",
      "West Europe",
      "North US",
      "West US"
    ],
  }
}
```

- **variables** - Variables are not a required section of the JSON file. However, JSON templates can get quite complex and if you want to simplify the references in the template, variables become very useful. For example, if you consistently want to reference a virtual network name, creating it as a variable allows you to reference it easily throughout the JSON template.

  The format of the variables is a combined in a key pair value, as shown here:

  ```
  "variables": {
      "key": "value"
  },
  ```

  The following example is used to set a VM Name, the size of a VM and the Name of a Vnet.

  ```
  "variables": {
      "vmName": "MyWindowsVM",
      "vmSize": "Standard_A2",
      "virtualNetworkName": "MyVNET",
  },
  ```

- **resources** - Resources are a required section of the JSON file. Resources allow you to specify the specific Azure resources what you want to be deployed in Azure from the template, i.e. - a VM, a network card, a storage account, etc…

  The format of the resources sections is as follows:

```
"resources": [
  {
    "apiVersion": "<api-version-of-resource>",
    "type": "<resource-provider-namespace/resource-type-name>",
    "name": "<name-of-the-resource>",
    "tags": "<name-value-pairs-for-resource-tagging>",
    "dependsOn": [
      "<array-of-related-resource-names>"
    ],
    "properties": "<settings-for-the-resources>",
    "resources": [
      "<array-of-dependent-resources>"
    ]
  }
 ]
```

The apiVersion, type, and name are required elements of the resources section. The location, tags, dependsOn, properties, resources are all optional elements. The apiVersion for the schemas available can be located at the following URL: https://github.com/Azure/azure-resource-manager-schemas

TIP: The **DependsOn** section of the JSON template is the key to controlling deployment order. For a good example of effective use of the DependsOn section, see the "Create a new Windows VM and create a new AD Forest, Domain and DC" deployment template at https://github.com/Azure/azure-quickstart-templates/tree/master/active-directory-new-domain.

DependsOn is used in 5 places in the azuredeploy.json template. Perform a search on "DependsOn" and review all matching instances to get a feel for how this option is used.

The **type** field varies depending on the resource you are deploying, the following is a sample of the type values of some common resources and how you reference them in a template.

- Microsoft.Computer/virtualMachines
- Microsoft.Web/serverfarms
- Microsoft.Web/sites
- Extensions
- Microsoft.Network/virtualNetworks
- Microsoft.Network/networkInterfaces

A Sample resources for deploying a virtual network and a storage account is below.

```
"resources": [

  {
    "type": "Microsoft.Storage/storageAccounts",
    "name": "[parameters('newStorageAccountName')]",
    "apiVersion": "2015-05-01-preview",
    "location": "[variables('location')]",
    "tags": {
      "displayName": "StorageAccount"
    },
    "properties": {
      "accountType": "[variables('storageAccountType')]"
    }
  },
  {
    "apiVersion": "2015-05-01-preview",
    "type": "Microsoft.Network/virtualNetworks",
    "name": "[variables('virtualNetworkName')]",
    "location": "[variables('location')]",
    "tags": {
      "displayName": "VirtualNetwork"
    },
    "properties": {
      "addressSpace": {
        "addressPrefixes": [
          "[variables('addressPrefix')]"
        ]
      },
      "subnets": [
        {
          "name": "[variables('subnetName')]",
          "properties": {
            "addressPrefix": "[variables('subnetPrefix')]"
          }
        }
      ]
    }
  }
```

```
]
```

Outputs are not a required section of the JSON file. If you need the deployment to return data after the deployment, you can specify the output in this section. This might include a simple success/failure value, or perhaps something more dynamic, like a value constructed (concatenated) from multiple parameters submitted to this template deployment time.

The format of the output section is as follows

```
"outputs": {
  "<outputName": {
    "type": "<type-of-output-value>",
    "value": "<output-value-expression>",
  }
}
```

The outputName, type and value fields are required. The type value allows the same input types.

A sample output section is shown below.

```
"outputs": {
  "operationResult": {
    "type": "string",
    "value": "[parameters('location')]",
  }
}
```

Using Preconfigured Templates

To get started quickly with ARM Templates, Microsoft has published a variety of templates that are freely available on GitHub in the azure-quickstart-templates repository located at the following url: https://github.com/Azure/azure-quickstart-templates.

The templates are designed to help you get started quickly.

To deploy a pre-configured SharePoint ARM template in the Azure Portal using the "Deploy to Azure" button, complete the following steps:

1. Open a browser and navigate to http://azure.microsoft.com/en-us/documentation/templates/

2. The Azure Quickstart Template page, shown in Figure 53, displays current popular templates and also provides a search function to find a template for your needs.

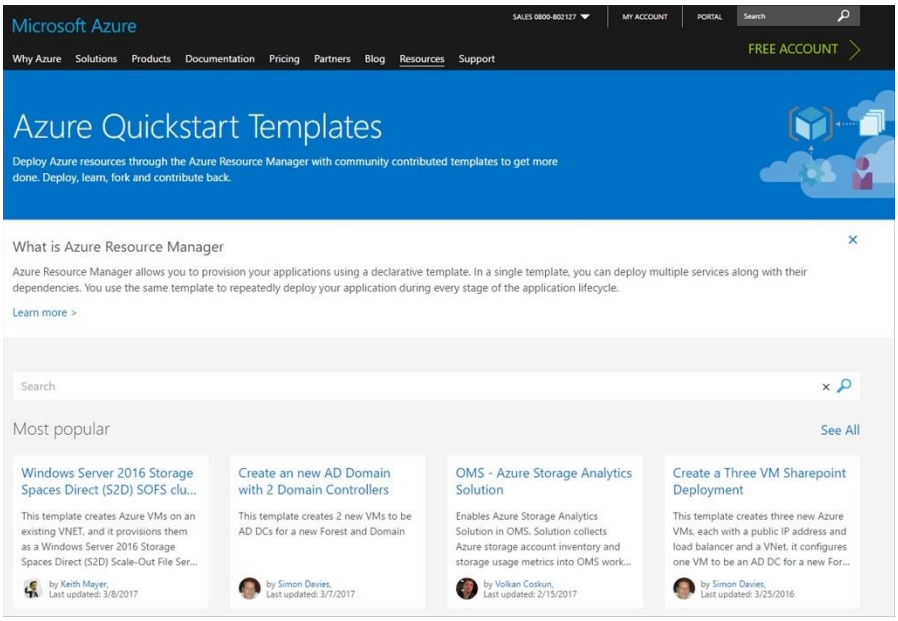


FIGURE 53. AZURE QUICKSTART

3. In the Search field type "SharePoint" and click the search magnifying glass; this will return 3 templates as shown in figure 54.

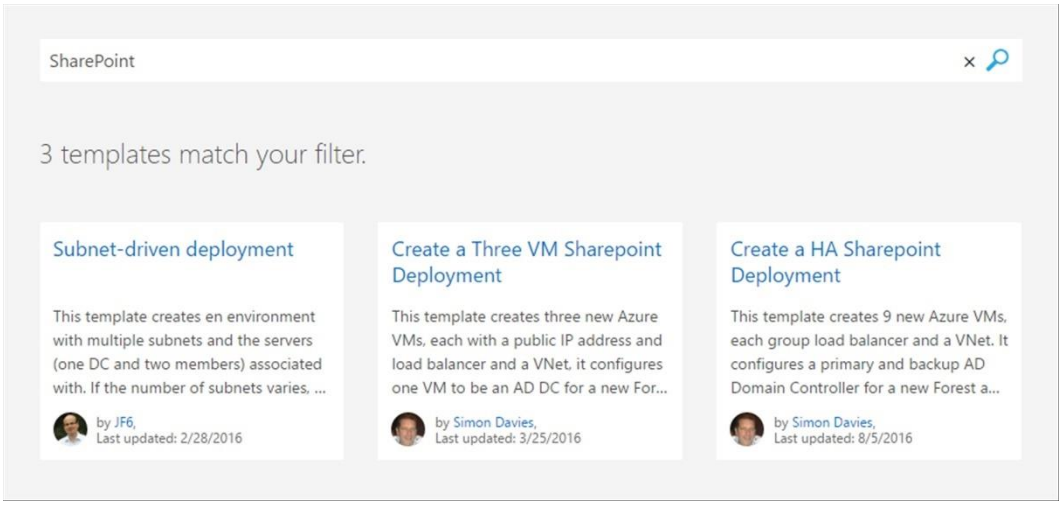


FIGURE 54. SHAREPOINT QUICKSTART TEMPLATES

4. Click the "Create an HA SharePoint Deployment"
5. Review the template and the parameters of the template as shown in Figure 55 and finally Click the Deploy to Azure button.

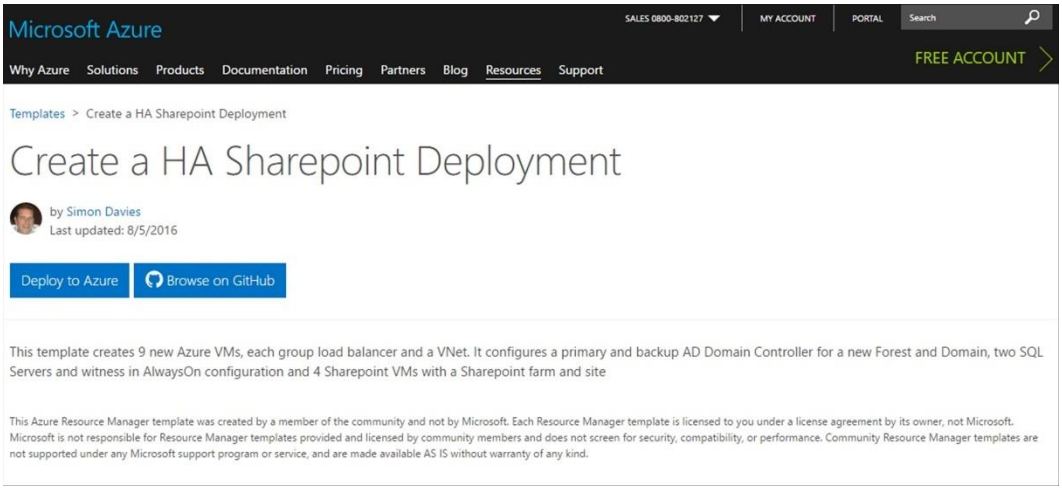


FIGURE 55. PARAMETERS AND DEPLOY TO AZURE

6. After you click Deploy to Azure, this will re-direct you to the Azure portal and open up a Custom Deployment blade as shown in Figure 56

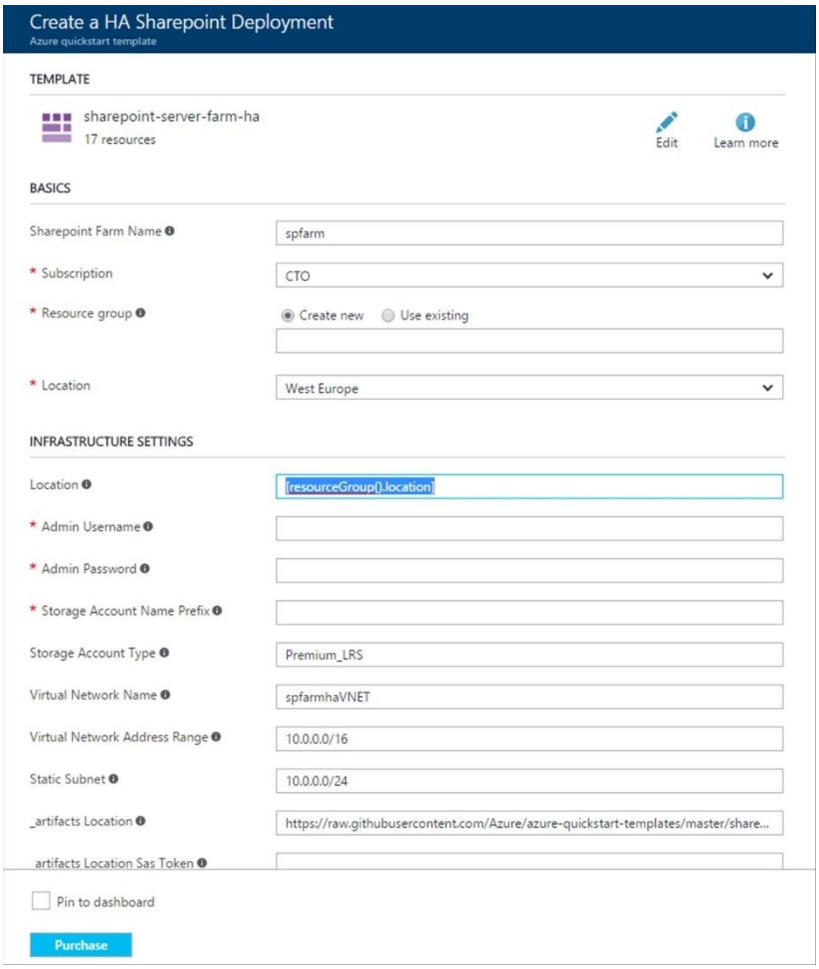


FIGURE 0. TEMPLATE DEPLOYMENT PARAMETER SETTINGS

7. Next, review and populate all of the required parameters in the template. Once you are finished filling out the template parameters, review the terms and conditions statement and put a checkmark in the **I agree to the terms and conditions as stated** above box and then click **Purchase** as shown in Figure 57.

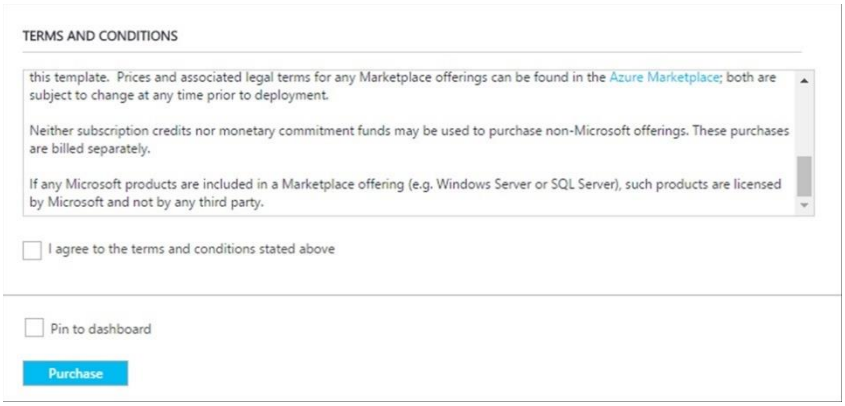


FIGURE 57. TERMS AND CONDITIONS AND PURCHASE

Authoring ARM Templates

QuickStart templates are a good way to get started, and for some people, may be all that is required. However, most organizations will have a need to create custom templates tailored to their specific deployment needs.

For this, we need to author a Template to cover the specific needs. There are several authoring options depending on your needs and experience. Currently, you can author in Visual Studio, the Azure Portal, and in a text editor like Notepad++.

Authoring and deploying an ARM Template in the Azure Portal

In this section, we will quickly demonstrate how you can author and deploy some basic resources using an ARM template in the Azure Portal. While this demonstration can be extended to deploy several resources into Azure, deploying complex environments that require post-deployment tasks (i.e. – Custom Scripts, PowerShell, DSC, etc…) is not recommended.

1. Open a browser and navigate to https://portal.azure.com.
2. Sign-in to Azure with your credentials.
3. Click **+NEW** from the left-hand blade menu.
4. Under the text box, type in Template and then click on **Template deployment** as shown in Figure 58.

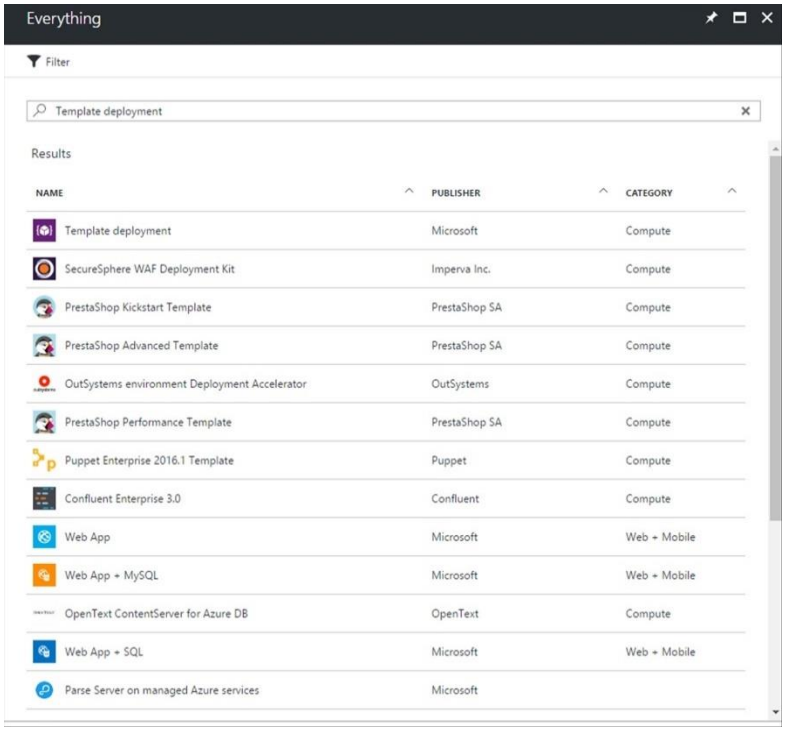


FIGURE 58. SELECTING TEMPLATE DEPLOYMENT

5. Click **Create** as shown in Figure 59.

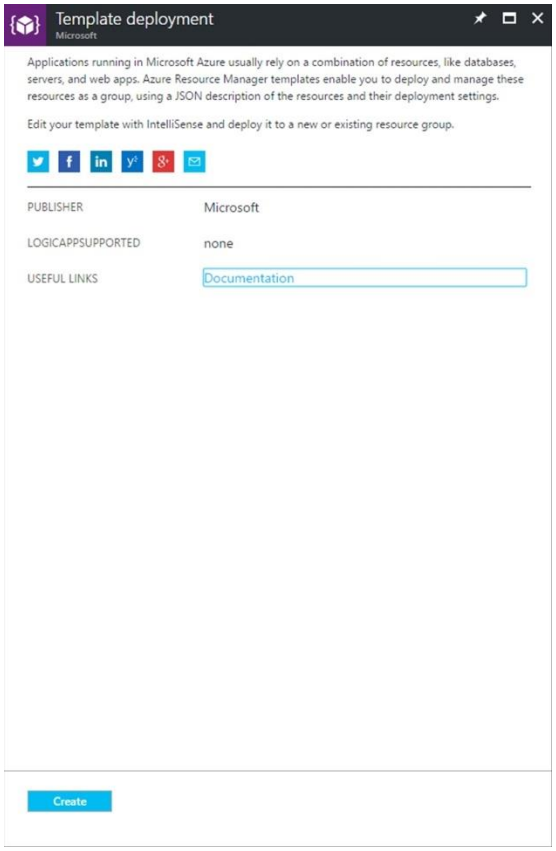


FIGURE 59. TEMPLATE DEPLOYMENT - CREATE

6. In the **Custom deployment** blade, click **Edit** as shown in Figure 60.

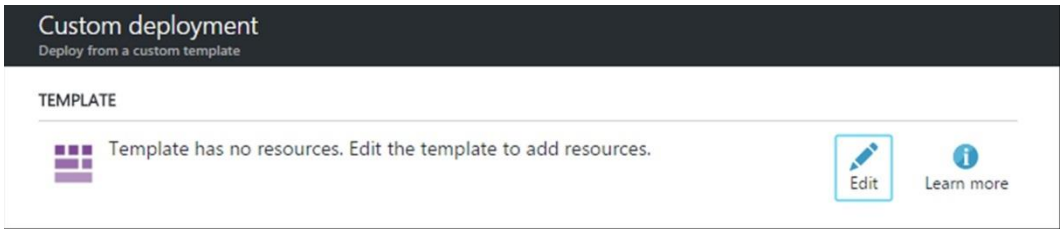


FIGURE 60. EDIT TEMPLATE

7. This drops you into a template editor, which presents you with the outline of the JSON file schema and allows you to add your resources, parameters, and variables in as required. Click on **+ Add resource** as shown in 61.

FIGURE 61. TEMPLATE EDITOR

8. Next, under the 'Select a resource' drop-down menu, select **Windows virtual machine**. Three additional text fields will appear for the **Name** of the VM, **Storage account**, and **Virtual network** as shown in Figure 62. Populate these fields with the following values:

   - **Name** set to template-vm
   - **Storage account** set to templatevm
   - **Virtual network** set to template-VNet

   When you are finished, click on **OK**.

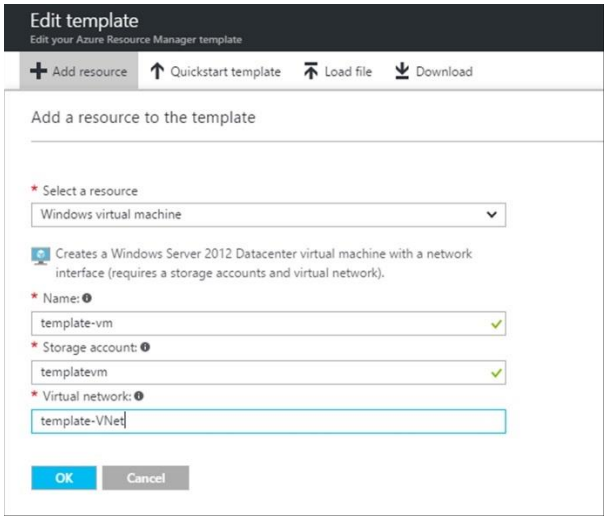


FIGURE 62. ADDING A WINDOWS VIRTUAL MACHINE TO THE TEMPLATE

9. Next, review the contents of the template and then click the **Save** button as shown in Figure 63.

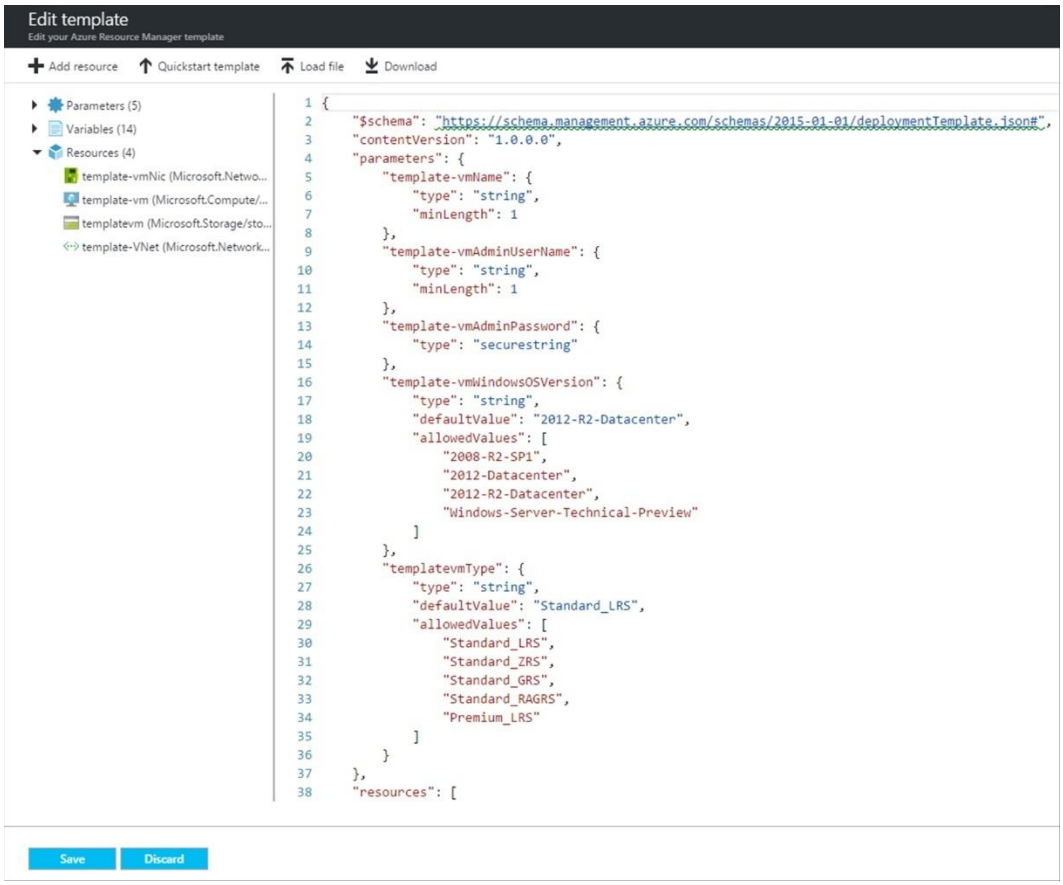


FIGURE 63. REVIEWING THE TEMPLATE

10. Next, the **Custom deploy** blade will appear. Fill out the settings in the ARM template with the following values as shown in Figure 64.

- **Resource Group**, choose **Create new** and set to deploy-template-vm
- **Template-vm Name** set to template-vm
- **Template-vm Admin User Name** set to winadmin
- **Template-vm Admin Password** set to LetMeInThis1!
- **Template-vm Windows OS Version** set to 2012-R2-Datacenter
- **Templatevm Type** set to Standard_LRS

When you are finished, put a checkmark next to the **I agree to the terms and conditions stated above** checkbox and click on **Purchase**.

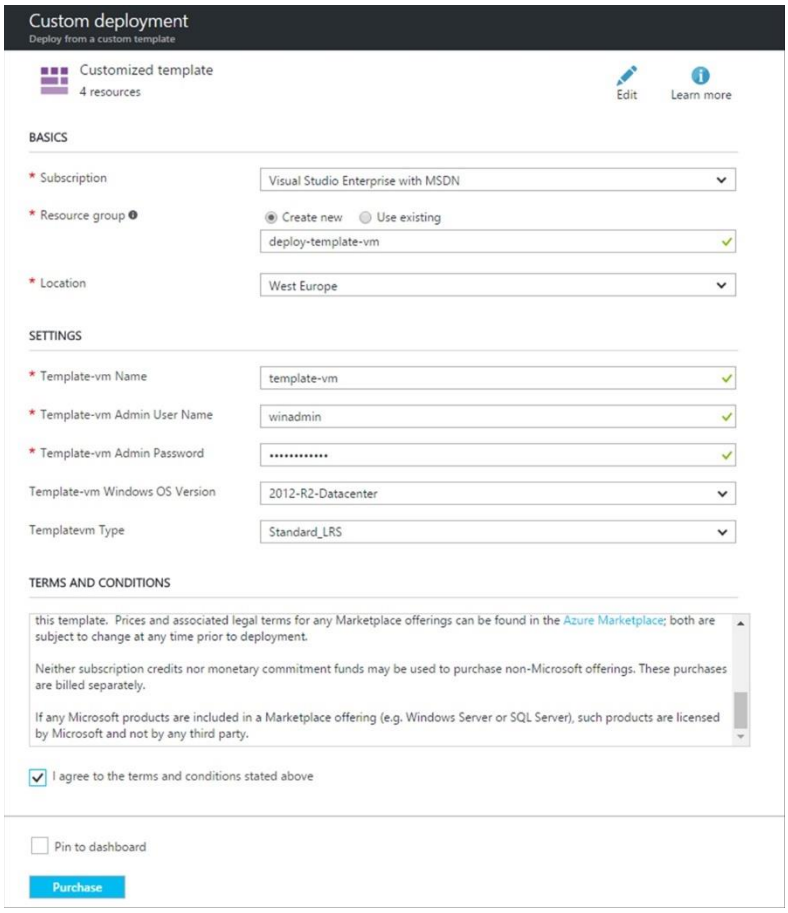


FIGURE 64. FILLING OUT THE CUSTOM DEPLOYMENT OF THE WINDOWS VM

11. Once the deployment has completed successfully, you can open up the **deploy-template-vm** resource group to see the resources that were deployed as shown in Figure 65.

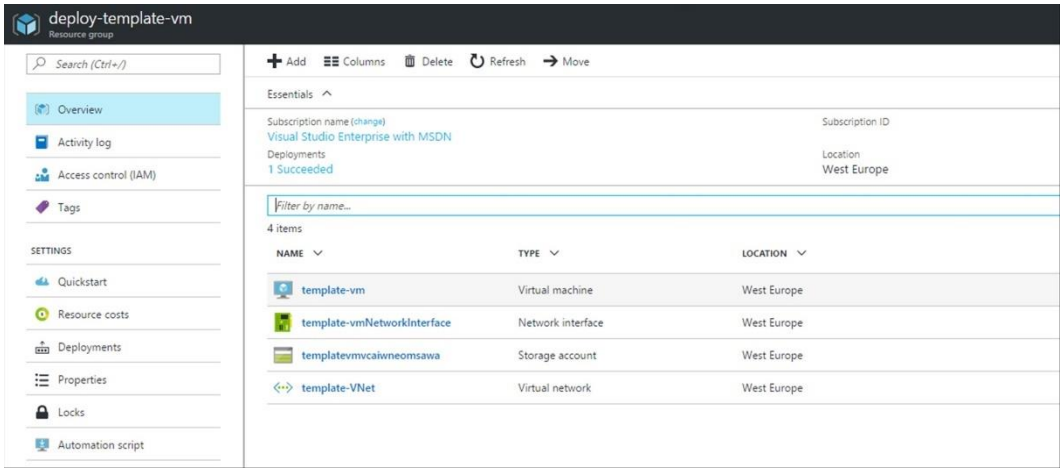


FIGURE 65. DEPLOY-TEMPLATE-VM RESOURCE GROUP AND RESOURCES

Authoring and deploying an ARM Template in Visual Studio

In this section, we will author and deploy a simple Azure Web App in Visual Studio 2015. Regardless of which of these Visual Studio versions you choose, template authoring in Visual Studio requires that you install the Azure SDK 2.5 or above. The Web Platform Installer will allow you to install the latest version of the Azure SDK for the visual studio version you have installed. You can download the Web Platform Installer at the following link: http://www.microsoft.com/web/downloads/platform.aspx.

The following example is based on Visual Studio 2015 Community Edition Update 3 with Azure SDK 2.9.6.

Step 1: Create a New Project

1. Start **Visual Studio 2015**.
2. Click **File**, Click **New** and click **Project**
3. Expand **Templates**, Click **Cloud**, Select **Azure Resource Group** as shown in Figure 66.

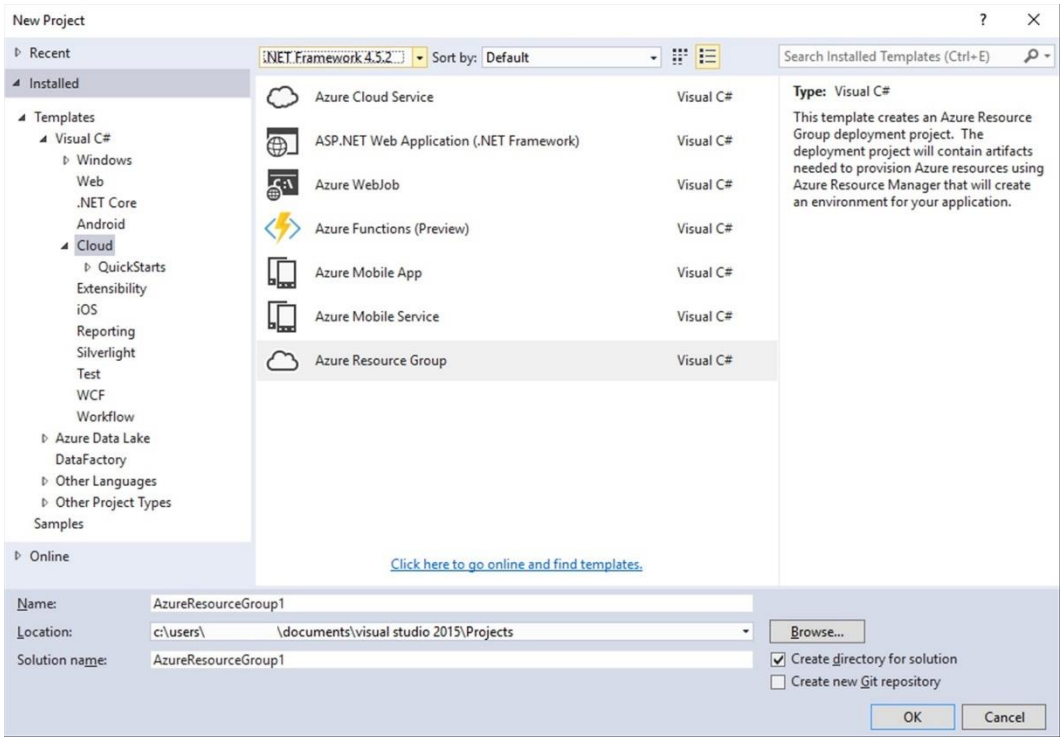


FIGURE 66. CREATING AZURE RESOURCE GROUP

4. Enter a Name for the Project and Click **OK**.
5. In the **Select Azure Template** window locate **Blank Template** and click **OK** as shown in Figure 67. Note that you have the option to load Templates from Azure Quickstart Templates on GitHub in the drop-down menu.

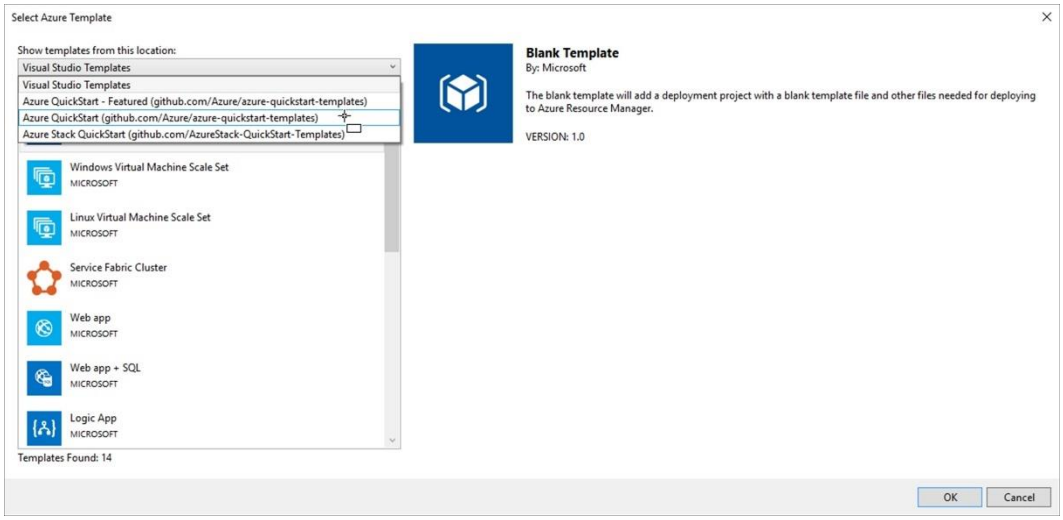


FIGURE 67. SELECT AZURE TEMPLATE

6. The project is broken down as follows, in the left-hand window you have the **JSON Outline**, in the center you have the JSON file and in the right-hand window you have the solution explorer for the project you are authoring as shown in Figure 68.

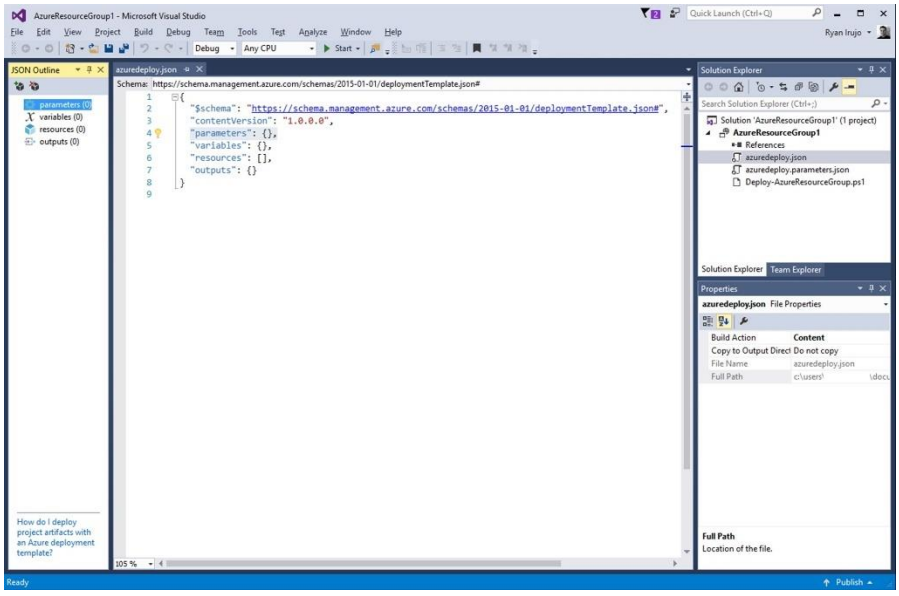


FIGURE 68. VISUAL STUDIO PROJECT

Step 2: Add resources to your JSON template

Next, we will use the visual JSON authoring feature in Visual Studio to add resources to an ARM template.

1. In the **JSON Outline** menu, right click on **resources** and Click **Add New Resource**
2. In the **Add Resource** window, select **Virtual Network** and type a **Name** and click **Add** as shown in Figure 69. In this example, we gave our VNET name 'TestVNet'.

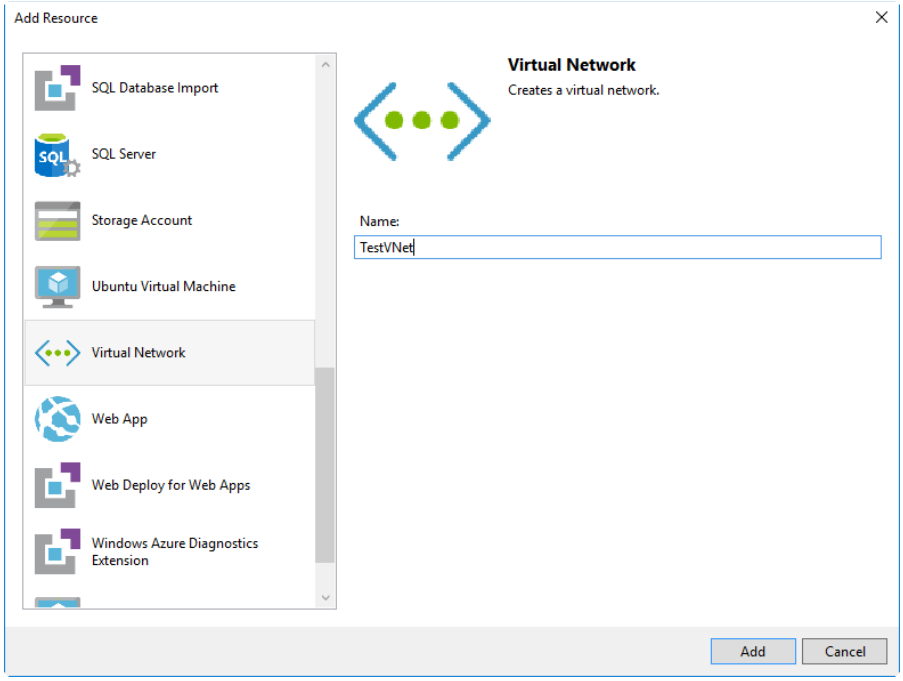


FIGURE 69. ADD RESOURCE

3. Review the code added to the **azuredeploy.json** file as shown in Figure 70.

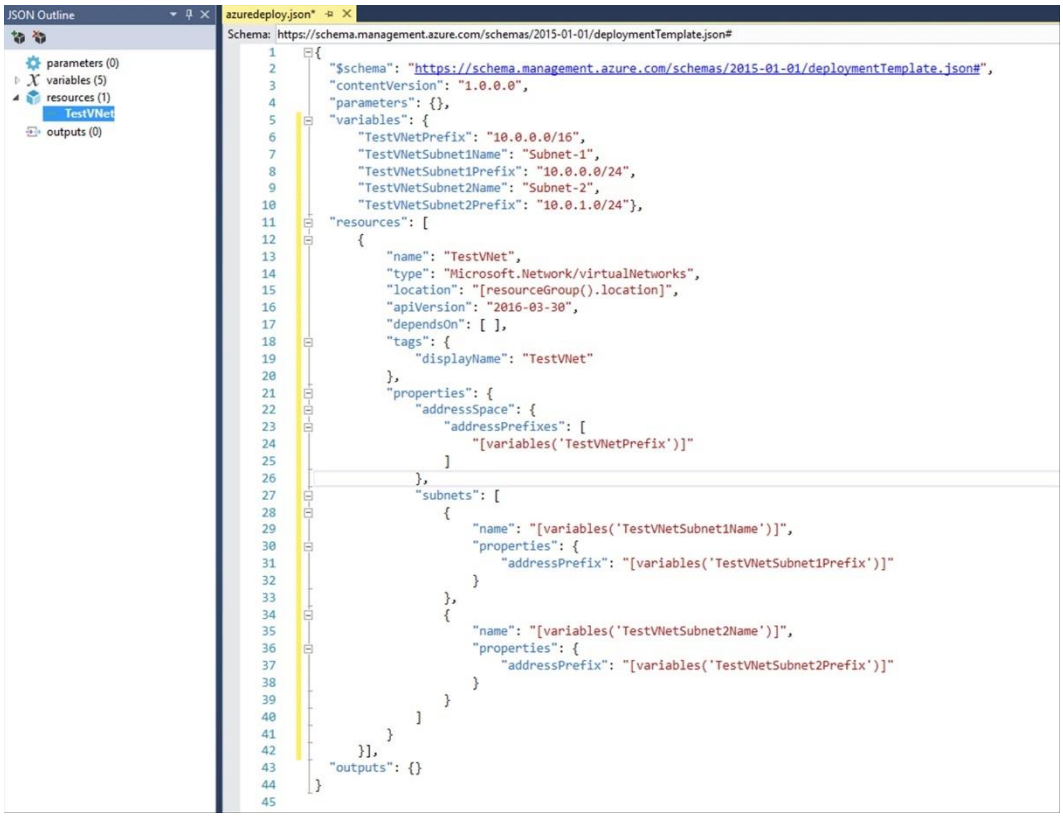


FIGURE 70. JSON TEMPLATE WITH VIRTUAL NETWORK ADDED

4. In the **JSON Outline** menu, right click on **resources** and Click **Add New Resource**
5. In the **Add Resource** Window Select **Storage Account** and Type a **Name** and Click **Add**.

> **NOTE:** Make sure the name of the storage account is only between 3 and 6 characters long or else the deployment will fail. This is because the full name of the storage account is appended with additional characters to make it unique before deployment and if the full name is longer than 24 characters long, Azure will reject it.

6. Review the code added to the **azuredeploy.json** and the **JSON Outline** for the addition of the storage account
7. In the **JSON Outline** menu, right click on **resources** and Click **Add New Resource**
8. In the **Add Resource** window, select **Windows Virtual Machine**.
    a. For the **Name** field, type a name for the VM in the space provided.
    b. For the **Storage account** field, click on the drop-down menu and select the storage account you just created.
    c. For the **Virtual network/subnet** field, click on the drop-down menu and select one of the Subnets that was created earlier from the VNet. Once you are finished, click **Add** as shown in Figure 71.

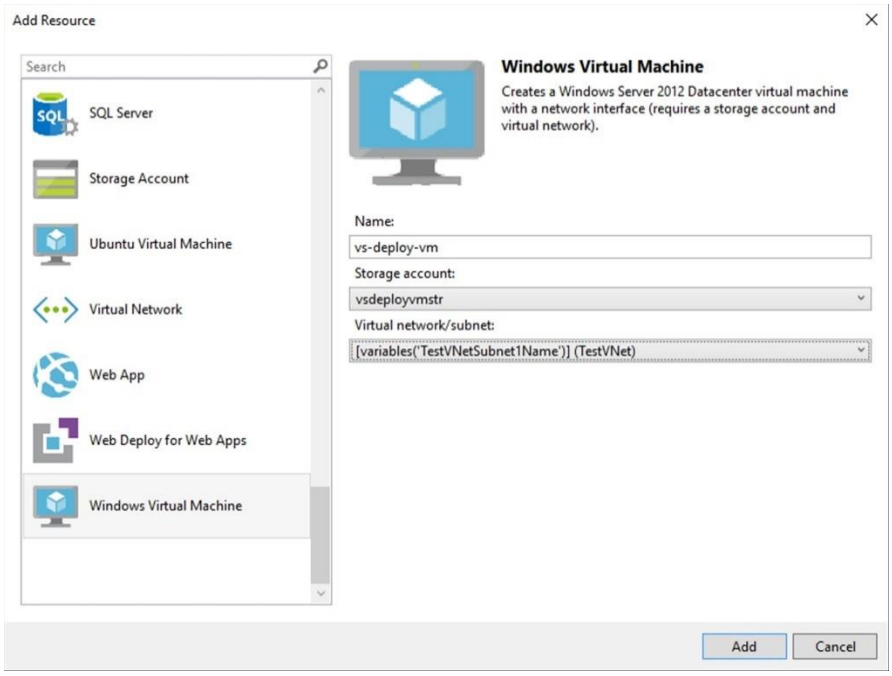


FIGURE 71. CONFIGURING THE WINDOWS VIRTUAL MACHINE RESOURCE

9. In the **azuredeploy.json** file as shown in figure 72, you will see the updated code added for the Windows VM.

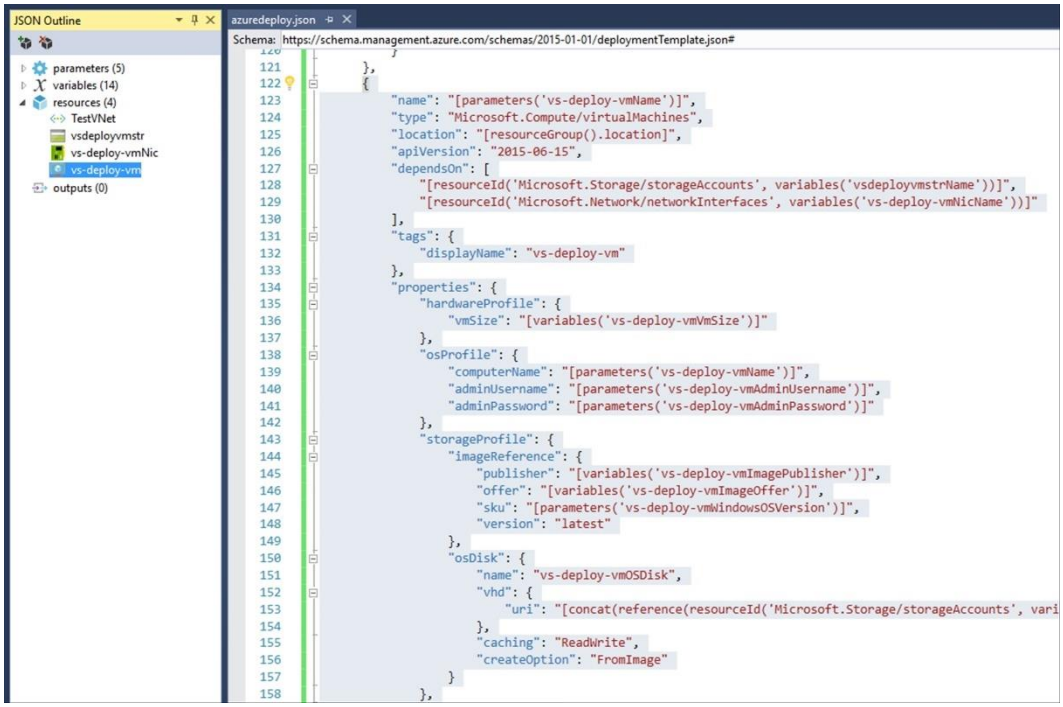


FIGURE 72. VIRTUAL MACHINE RESOURCE SECTION

Step 3: Deploy the JSON template from Visual Studio

With authoring complete, you can now deploy your template directly from Visual Studio.

1. In the main toolbar, click on **Project**, **Deploy**, **New...** as shown in Figure 73 below.

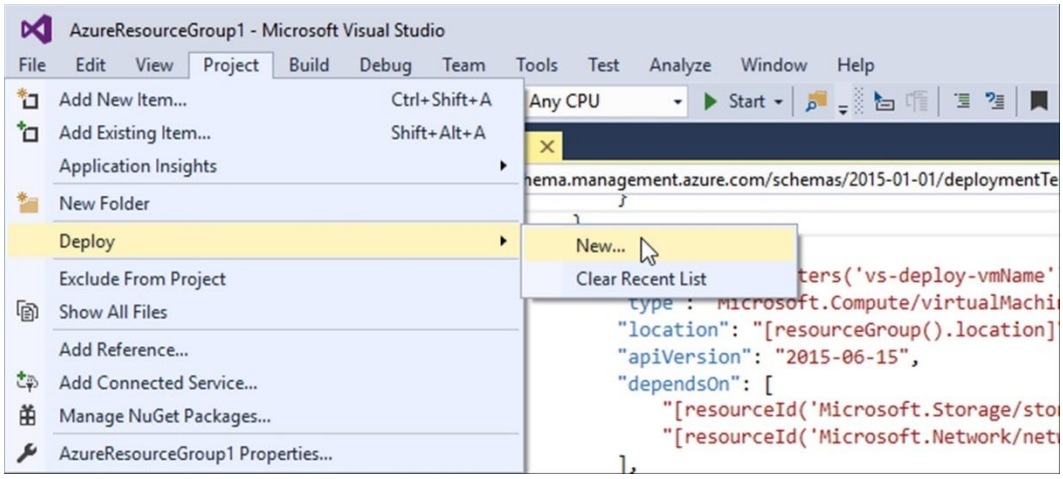


FIGURE 73. GETTING READY TO DEPLOY THE ARM TEMPLATE

2. In the **Deploy to Resource Group** Window**,** click **Sign In**, as shown in Figure 74.

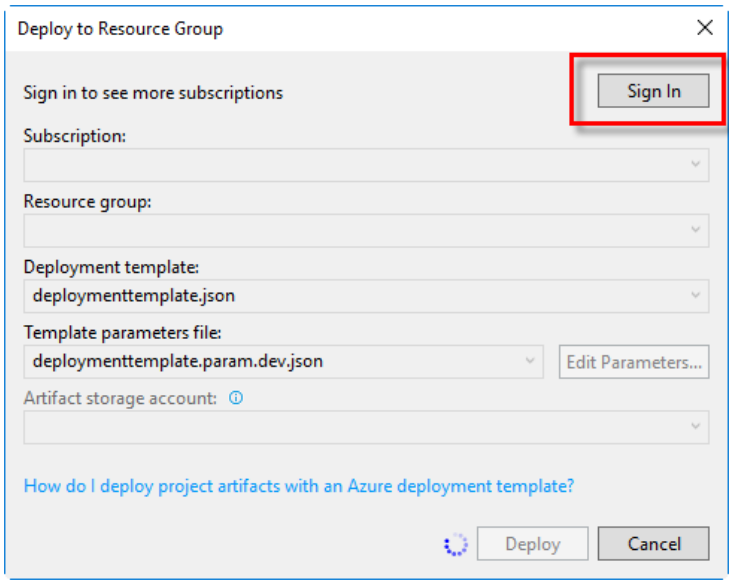


FIGURE 74. DEPLOY TO RESOURCE GROUP

3. Sign into Azure and follow the prompts.
4. When sign in has completed, in the **Deploy to Resource Group** window you can select the **Subscription** to which you want to deploy, create or select the

Resource Group you want to deploy to, as shown in Figure 75. Once you are finished, click **Edit Parameters**.

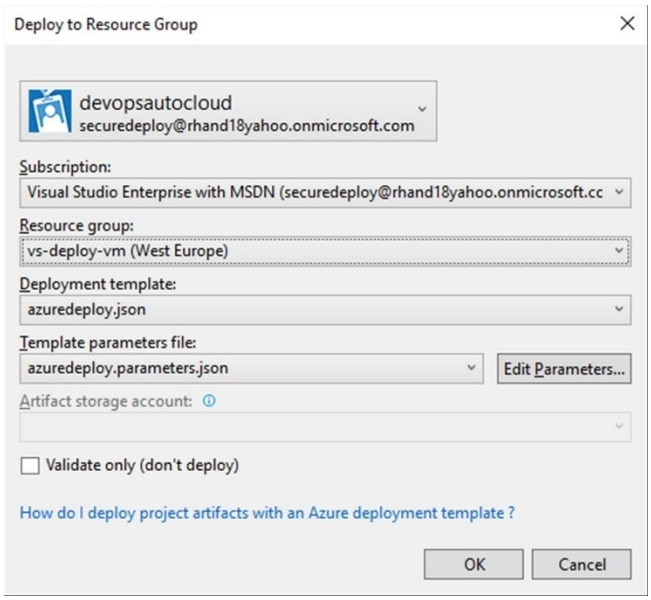


FIGURE 75. SUBSCRIPTION AND RESOURCE GROUP SELECTION

5. In the **Edit Parameters** window, populate the fields and select the appropriate items for the Storage Account, Windows login information and the Windows OS as shown in Figure 76. After you are finished, click **Save**.

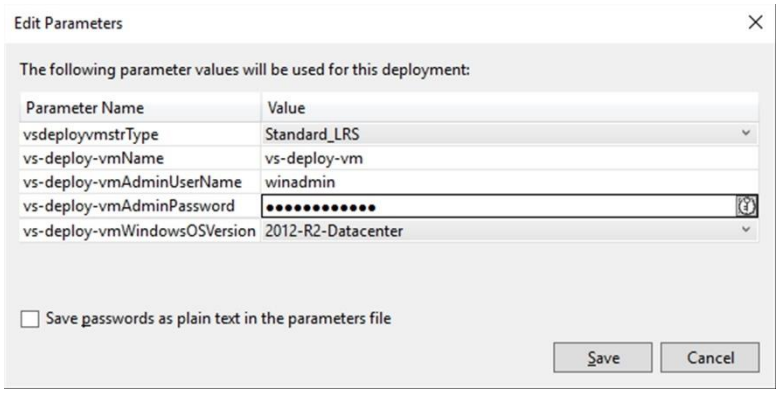


FIGURE 76. EDIT PARAMETERS

6. Back in the **Deploy to Resource Group** window, click **OK** to deploy the ARM Template.
7. When prompted, enter a password for the Windows admin account and click on **OK** as shown in figure 77.

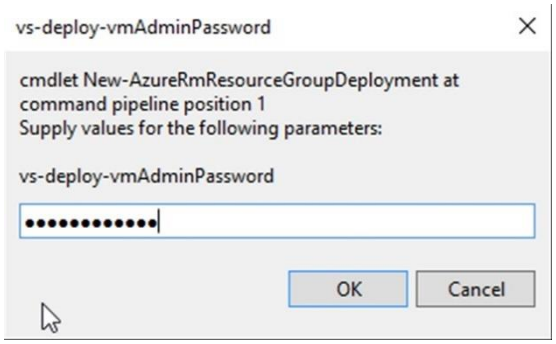


FIGURE 0. WINDOWS ADMIN ACCOUNT PASSWORD PROMPT

8. Observe the **Output** Window as shown in Figure 78. to track the status of the deployment.

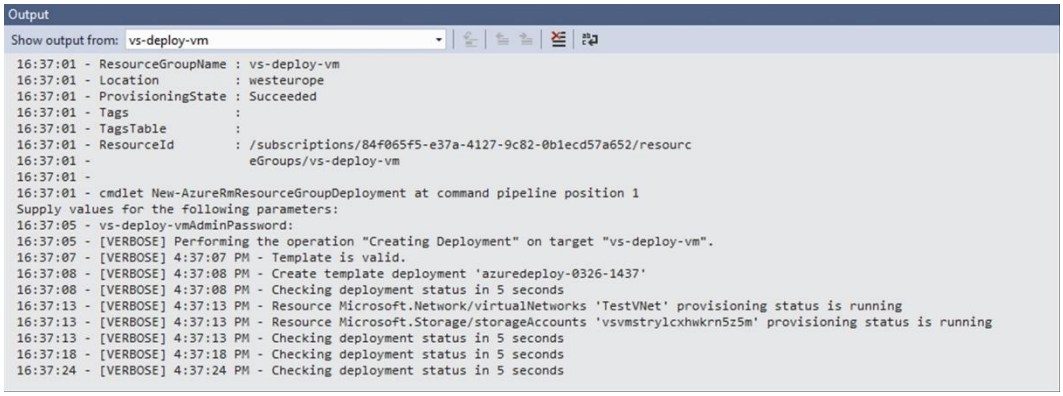


FIGURE 78. OUTPUT WINDOW

9. Once the deployment is complete, verify all of the resources have been deployed successfully in the Azure Portal.

Deploying an ARM Template from PowerShell

Once you have created a JSON template file you can save it and use it to deploy the authored or obtain ARM template. When you deploy a JSON template with PowerShell, you will be prompted to supply values for the parameters defined in the file.

The `New-AzureRmResourceGroupDeployment` cmdlet is used to deploy the template. You can deploy from a PowerShell prompt or the PowerShell ISE.

Syntax and Sample

This working sample will deploy a relatively simple JSON template from the Azure Quickstart Templates repository on Github. This particular template will deploy an Ubuntu Linux VM to a new VNET and is accessible via SSH from a public IP address.

1. Login to Azure from an elevated PowerShell prompt and create new Resource Group for the Linux VM.

```
New-AzureRmResourceGroup `
    -Name deploy-linux-vm `
    -Location "West Europe"
```

2. Create the URI variable that points to the **azuredeploy.json** file for deploying the Linux VM in the Azure Quickstart Template Github Repository.

```
$URI = "https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/101-vm-simple-linux/azuredeploy.json"
```

3. Define the parameter values to be used in the template deployment. Yes, passwords in clear text are bad and are only used here for demonstration purposes!

```
$Parameters =
@{adminUsername="linuxadmin";adminPassword="PasswordInClrTxtIsBad1
!";dnsLabelPrefix="linuxvmwe"}
```

4. Deploy the ARM template.

```
New-AzureRmResourceGroupDeployment `
    -Name LinuxDeployment `
    -ResourceGroupName deploy-linux-vm `
    -TemplateURI $URI `
    -TemplateParameterObject $Parameters
```

5. Once the deployment is complete, you should get something similar to the response below as shown in Figure 79.

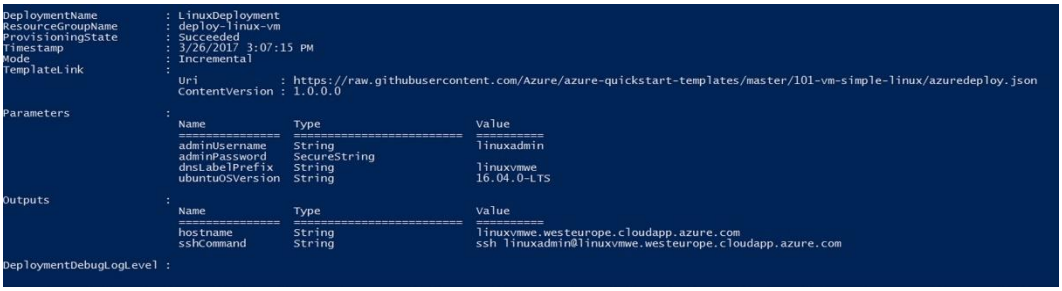


FIGURE 79. OUTPUT WINDOW

Configuring Azure VMs

In this chapter, we have covered several aspects of deploying virtual machines in Azure and configuring various settings related to the VM; however, what if we need to install or configure a component inside the VM itself?

For example, if we want to deploy an application post VM deployment, or reset the local administrator password. In this section, we will cover how we can leverage VM Extensions and PowerShell DSC to achieve these scenarios and many more!

VM Extensions

VM Extensions enable further customization of a virtual machine post deployment. For example, if you want to run a custom script or leverage a 3rd party utility like Chef or Puppet, to deploy an application, VM Extensions can be enabled on a VM using the Azure Portal, Azure PowerShell, the Azure CLI or from an ARM template.

The list of VM Extensions available for Azure is constantly being updated and includes extensions for Chef, Puppet, DSC, DSC for Linux, along with anti-malware and monitoring agents from Microsoft and 3rd party vendors.

For up-to-date use cases and samples of VM extensions and features for Windows, visit the Microsoft website at https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-windows-extensions-features.

For up-to-date use cases and samples of available VM extensions and features for Linux, visit the Microsoft website at https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-linux-extensions-features

PowerShell DSC

PowerShell DSC is a declarative model for deployment and configuration of Windows resources. PowerShell DSC enables the creation of file-based configurations to enable a wide variety of Windows and Linux roles, features and application. For example, if you wanted to install the Windows Active Directory domain controller role on a VM and promote it to a domain controller in a new or existing domain, you could do this with PowerShell DSC. PowerShell DSC allows you to download and install software from a share or the internet.

The high-level process is as follows:

- Download or create a PowerShell DSC configuration file
- PowerShell DSC takes a configuration file and packages it with the associated modules into a Zip file.
- Upload the Zip file to an artifact store, publicly accessible file hosting service or version control repository, or an azure storage account.
- The Zip file is referenced in an ARM Template for use during a Virtual Machine deployment and its associated Virtual Machine Extension and deploys the DSC configuration post-VM deployment.

Covering Powershell DSC in full detail is outside the scope of this book, as an entire book could be written specifically about DSC. For more introductory material on using DSC in Azure, please review the Microsoft documentation at https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-windows-extensions-dsc-overview.

Summary

In this chapter, we discussed Azure Resource Manager (ARM) and the new deployment model for VM and application deployment. We discussed VM sizing and configuration options of VM images in the Azure Marketplace. We explored ARM deployment options for Windows and Linux VMs in depth, including via the Azure Portal, Azure PowerShell, and ARM templates. With a few deployments completed, we touched on the additional configuration capabilities of ARM, including VM Extensions and PowerShell DSC.

In chapter 7, we will explore options for migration from on-premises data centers to Microsoft Azure from planning to implementation, including tips and tricks to ensure your migration to Azure is a success.

Chapter 7 – Migration to Microsoft Azure

This chapter provides an analysis of how to migrate servers to Microsoft Azure. The chapter will walk through the pricing model for virtual machines (VM) in Microsoft Azure, how to evaluate which servers to migrate and how to migrate them. The chapter will also look into how you can optimize the cost and how System Center can support you. In this chapter, we will cover all parameters that can affect the total cost of a virtual machine.

Evaluate Servers to Migrate to Azure

The Microsoft Assessment and Planning (MAP) toolkit is an agentless tool that can help you investigate your current environment and generate suitable reports before starting a migration to Microsoft Azure. These reports will help you estimate the needed capacity to run your servers on Microsoft Azure platform. It will also rank applications based on migration suitability and generate a TCO-ROI analysis.

Note that Microsoft Assessment and Planning (MAP) Toolkit can't collect and analyze all parameters. For example, if you have a client console that requests data from its application servers and database servers, and you move the application servers or database servers to Microsoft Azure, each request from a client will cost extra, because there is network bandwidth being consumed by data traffic leaving the servers hosted in Microsoft Azure. The MAP tool cannot analyze this cost. In this scenario with the application and/or database servers in Microsoft Azure, publishing the console through Azure RemoteApp might be a path worth exploring.

Other things that the MAP Toolkit can't always analyze include:

- **Unsupported roles and features.** Most of the Windows Server roles and features are supported in Microsoft Azure, or they have an equivalent service. For example, the Windows Network Load Balancing feature is not supported on a VM running in Microsoft Azure. Instead, you can use Azure Virtual Networking to enable similar functionality. DHCP, Hyper-V, Remote Access and SNMP are some of the features and roles that are not supported in Microsoft Azure. For more information about what roles and features that are supported see "Microsoft

server software support for Microsoft Azure VMs" at http://support.microsoft.com/kb/2721672.

- Microsoft recommends all volumes to be formatted as NTFS
- 32-bit applications are supported, but the server must be an X64 server with Windows Server 2008 R2 or newer operating system.
- Even if the operating system is supported in Microsoft Azure, the server software running on top of the operating system might not be supported. For more information about which server software is supported in Microsoft Azure VMs, see the following Microsoft support KB article - https://support.microsoft.com/kb/2721672.
- If your server is configured to boot from a SCSI controller today, you need to update the server so that the system volume resides on an IDE-based VHD. Currently, all VMs in Microsoft Azure must boot from an IDE controller. If you are using the Microsoft Virtual Machine Converter, it will help you reconfigure the boot disk to IDE.
- If the server hosting applications that have license connected to current hardware profile it can be a risk that the software is not working after moving to new hardware in Azure.
- The MAP Toolkit cannot see connectivity between servers or bandwidth requirements.
- D is a restricted drive letter in Microsoft Azure VMs. All Microsoft Azure VMs utilize a temporary, non-persistent disk that is mounted as the D drive.
- A data disk can be a maximum of 1 TB. If you need larger disks than 1 TB you will need to use spanned volumes over multiple disks, with for example storage spaces.
- The disk for the operating system can be a maximum of 1023 GB.

The MAP Toolkit is free to download from the Microsoft Download Center. To inventory and generate a server report, follow these steps:

1. Download the MAP Toolkit from Microsoft Download Center and install it on a server with SQL Server database engine installed.
2. Start the MAP Toolkit from the start screen.
3. In MAP Toolkit, open the File menu and select Create/Select Database.
4. In the dialog box input a database name, for example, ContosoIT, as shown in figure 1. Click OK.

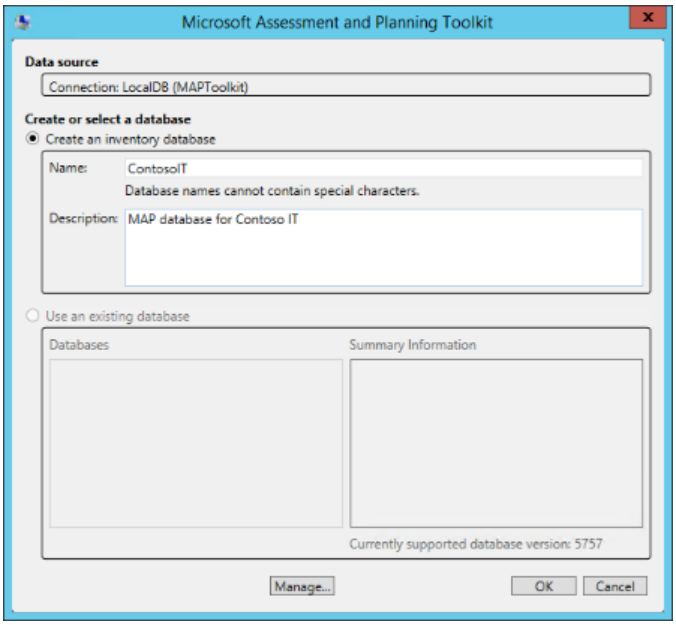


FIGURE 83. CONFIGURATION OF DATABASE FOR THE MAP TOOLKIT

1. In MAP Toolkit, click **Cloud** on the left side.
2. In MAP Toolkit, on the Cloud page, click **Collect inventory data** under Azure VM Readiness.
3. In Inventory and Assessment Wizard - Inventory Scenarios, select the scenario. In this example, we will select Windows computers as we are investigating which Windows based servers are candidates to move to Microsoft Azure. Figure 2 as shown the wizard. Click **Next**.

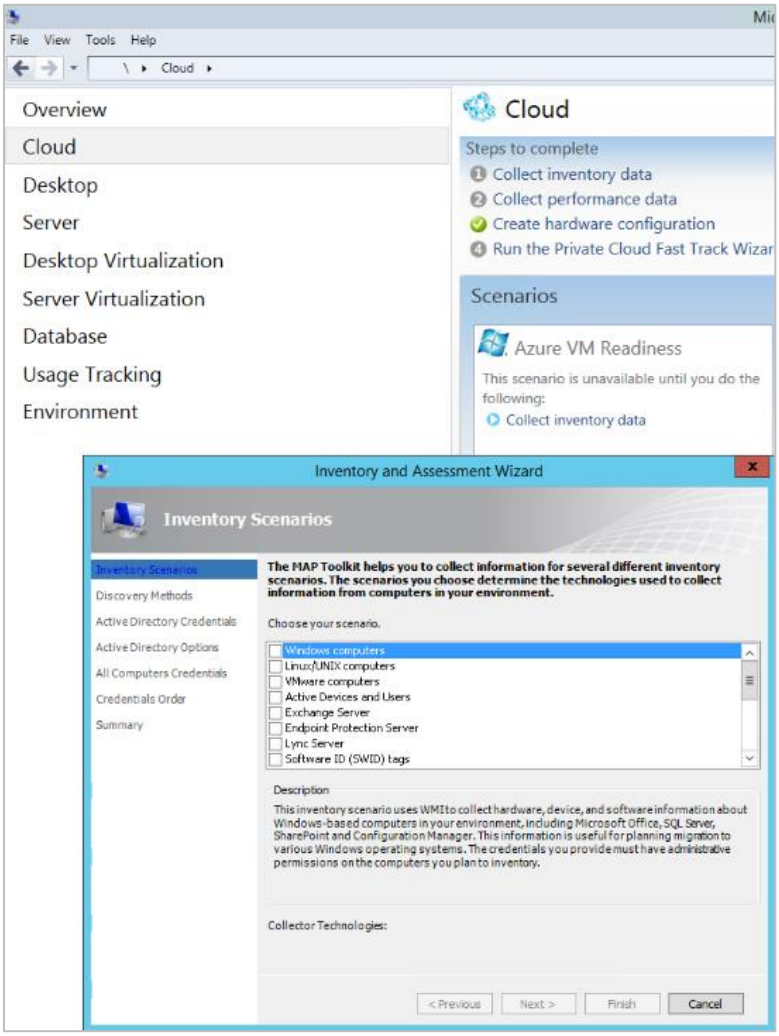


FIGURE 84. SELECT INVENTORY SCENARIO IN THE WIZARD

4. In Inventory and Assessment Wizard - Discovery Methods, select Use Active Directory Domain Servers (AD DN). You can use a number of sources to discover computers. In this example, we will use Advice Directory Domain Service only. Click Next.
5. In Inventory and Assessment Wizard - Active Directory Credentials, input domain and account credentials for an account that have permissions to access Active Directory. Click Next.

6. In Inventory and Assessment Wizard - Active Directory Options, select one of the two options to scope the query. Click Next.
7. In Inventory and Assessment Wizard - All Computers Credentials, input domain and account credentials for an account that have permissions to connect via WMI to each computer. Click Next.
8. In Inventory and Assessment Wizard - Credentials Order, verify the correct account is selected for WMI. Click Next.
9. In Inventory and Assessment Wizard - Summary, verify all settings and click Finish.

When you close the Inventory and Assessment Wizard a new window will pop up - the Inventory and Assessment page. Figure 3 shows the dialog box. This window will show the progress of machines discovered and inventoried. Let the machine inventory and collection processes complete.

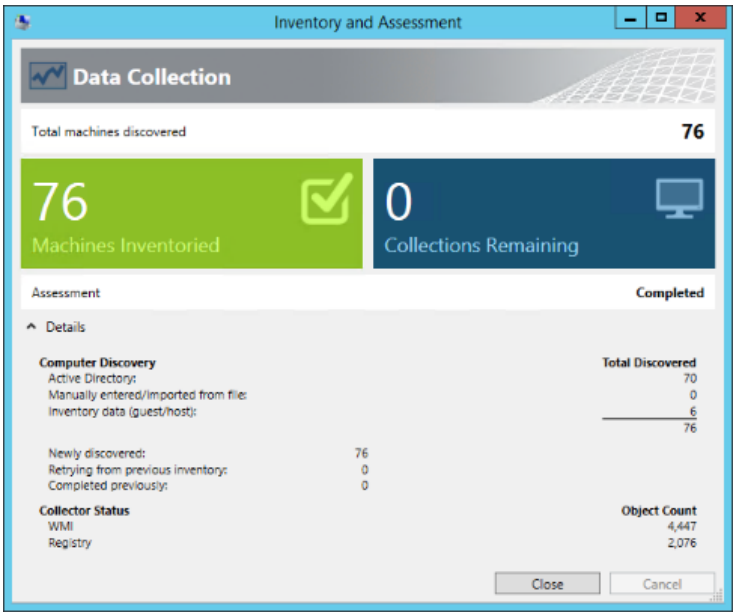


FIGURE 85. DIALOG BOX SHOWING THAT DATA COLLECTION FINISHED

We have now discovered a number of servers and collected basic information about them. The next step is to collect performance data for all discovered servers. To collect performance data, follow these steps:

1. In MAP Toolkit, on the Cloud page, click **Collect performance data** under Azure VM Readiness.
2. In Performance Metrics Wizard - Collection Configuration, configure a suitable performance collection duration. The default setting is one hour. It is important to collect performance data during normal operations to get as accurate data as possible. Click Next.
3. In Performance Metrics Wizard - Choose Computers, you can make a selection to either pick from the computers discovered earlier or provide a text file. In this example, we will leave the default setting and click **Next**.
4. In Performance Metrics Wizard - Computer List, shown in figure 4, select the computers from which to collect performance data. Computers that have type equals Insufficient Data were not contacted successfully during discovery. For example, if the computer was offline. Click **Next**.

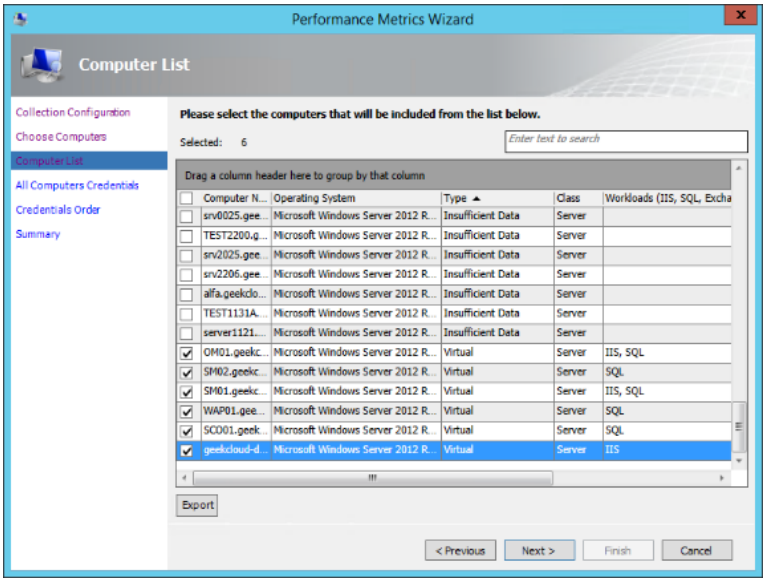


FIGURE 86. THE PERFORMANCE MATRIX WIZARD

5. In Performance Metrics Wizard - All Computers Credentials, verify the account and click **Next**. This account will be used to connect to each computer and collect performance data.
6. In Performance Metrics Wizard - Credentials Order, verify credentials and click **Next**.
7. In Performance Metrics Wizard - Summary, verify all settings and click **Finish**.

When you close the Performance Metrics Wizard, a new window will pop up, shown in figure 5. This window will show the progress of the performance data collection. When the collection of performance data is completed, click Close.

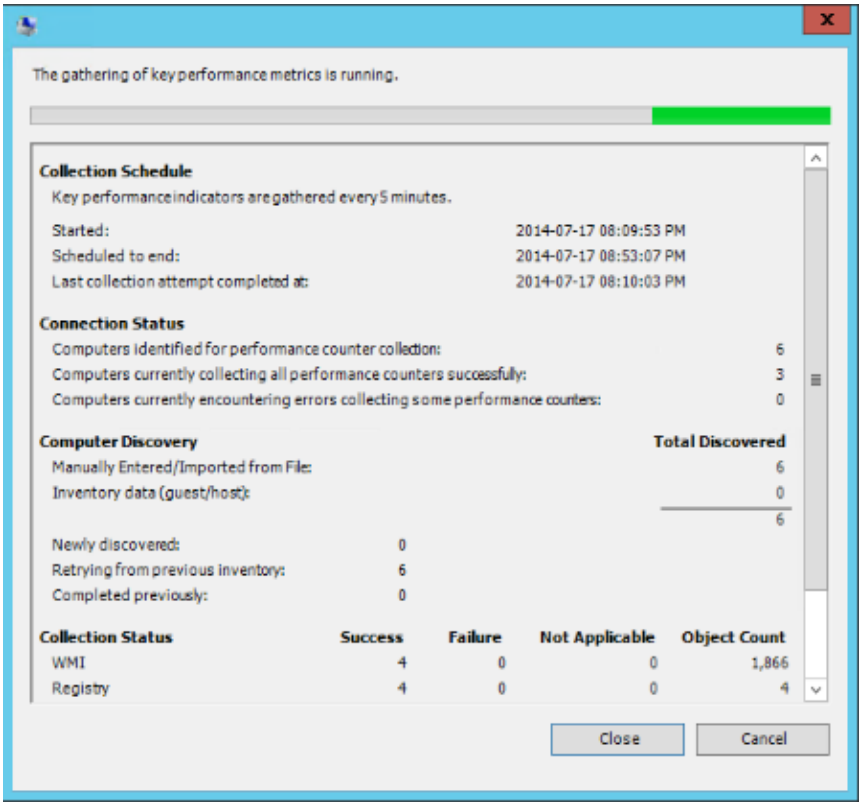


FIGURE 87. WIZARD SHOWING PROGRESS OF PERFORMANCE DATA COLLECTION

We now have collected inventory data and performance data. The next step is to generate the Windows Azure VM Capacity Report:

1. In the Cloud dashboard, click the Azure VM Capacity scenario.
2. The Windows Azure Virtual Machine Capacity dashboard gives you an overview of the machines we have collected data from and how they would fit in Microsoft

Azure. You can see the number of servers and the recommended VM sizing in Microsoft Azure. Figure 6 shows the Windows Azure Capacity Dashboard.

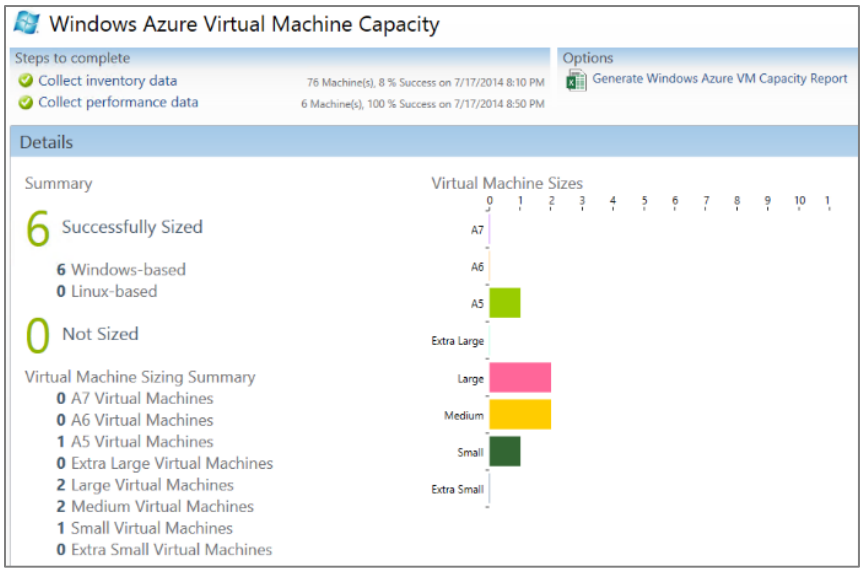


FIGURE 88. DASHBOARD SHOWING MACHINES NEEDED IN AZURE

3. To generate a report, click on Generate Windows Azure VM Capacity Report in the top right of the Windows Azure Capacity Dashboard page.
4. A dialog box, Report Generation Status, is displayed. When the requested report has been successfully prepared, check Open reports folder after Close, and click Close.
5. In the folder, open the report (an Excel file) that was generated in the previous step. Figure 7 and figure 8 show examples of the virtual machine sizing report. In the Machine Name column, you can see the name of the VM, and in the Azure VM Size column, you can see recommended size of the VM for migration to Microsoft Azure.

Virtual Machine Sizing Results

This report provides machine level sizing information and estimated monthly resource usage for each machine in your environment after it has been migrated to a Windows Azure Virtual Machine.

Machine Name	Operating System	Machine Type	Azure VM Size
geekcloud-dc01.geekcloud.local	Microsoft Windows Server 2012 R2 Datacenter	Virtual	Small
OM01.geekcloud.local	Microsoft Windows Server 2012 R2 Datacenter	Virtual	Large
SCO01.geekcloud.local	Microsoft Windows Server 2012 R2 Datacenter	Virtual	Medium
SM01.geekcloud.local	Microsoft Windows Server 2012 R2 Datacenter	Virtual	Large
SM02.geekcloud.local	Microsoft Windows Server 2012 R2 Datacenter	Virtual	A5
WAP01.geekcloud.local	Microsoft Windows Server 2012 R2 Datacenter	Virtual	Medium

FIGURE 89. EXAMPLE OF VIRTUAL MACHINE RESULT REPORT, SHOWING INVENTORIED MACHINES AND RECOMMENDED MICROSOFT AZURE VIRTUAL MACHINE SIZES

Machine Name	Azure VM Size	Est. Monthly Small Compute Hours	Est. Monthly Network Use-Outgoing (GB)	Est. Monthly Storage Use (GB)
geekcloud-dc01.geekcloud.local	Small	720	14.7	32.32
OM01.geekcloud.local	Large	2880	3.76	31.49
SCO01.geekcloud.local	Medium	1440	17.45	40.25
SM01.geekcloud.local	Large	2880	6.12	53.47
SM02.geekcloud.local	A5	1440	19.29	50.48
WAP01.geekcloud.local	Medium	1440	3.37	38.28

FIGURE 90. ANOTHER EXAMPLE OF THE REPORT SHOWN IN FIGURE 7

The Azure VM Capacity report includes a number of sheets:

- **Summary** - This sheet shows a summary of estimated resource usage and sizing requirements for the computers that you selected during data collection earlier in the MAP Toolkit process.
- **SizingResults** - This sheet shows information for each computer. For example, recommended VM size in Microsoft Azure and estimated storage utilization is shown.
- **VirtualMachineProfiles** - This sheet shows settings for each VM size in Microsoft Azure.
- **CurrentUtilization** - This sheet shows the current utilization on your computers. For example, disk size and CPU utilization.
- **MachineNotSized** - This sheet shows a list of machines that could not be sized.

If you are planning on migrating SQL Server, SharePoint Server or Active Directory servers, you can also use the Microsoft Azure Virtual Machine Readiness Assessment tool

to inspect your on-premises environment. The Virtual Machines Readiness Assessment tool will inspect your on-premises environment, both physical and virtualized servers, and generate a detailed report for these technologies on steps you need to take to migrate your environment to the cloud. The Microsoft Azure Virtual Machine Readiness Assessment tool will look into settings for the specific technology. For example, in a SQL Server scenario, the tools report will look into the following areas:

- Benefits and scenario
- Identity provider
- Supported roles and features
- Hardware needs
- Network
- Storage
- Disaster recovery
- Security
- Optimize configuration
- Move data
- Monitor data
- Support

We have now talked about how to investigate which servers are candidates to migrate to Microsoft Azure. We have looked at the MAP Toolkit that can help us investigate what we have and how it would fit into Microsoft Azure. Now that we identified our migration candidates, the next step is to determine the cost of running our servers in Microsoft Azure. The next section will cover Microsoft Azure IaaS pricing.

Microsoft Azure IaaS Pricing Overview

Cloud computing has five characteristics, according to National Institute of Standards and Technology (NIST). The five characteristics are:

- **On-demand self-service -** The cloud solution can automatically provision computing capabilities as needed without requiring human interaction.
- **Broad network access -** The data and services that your cloud provider needs to be available for access from a wide range of devices. For example, tablets, PCs, and phones.
- **Resource pooling -** Compute resources are pooled to serve multiple consumers dynamically. Based on customer or service demands compute resources are re-allocated.

- **Rapid elasticity -** The fabric of the cloud must be built in a way such that it can scale out and in easily, based on demand. For example, maybe you need 50 servers at the end of the month, but rest of the month you only need 5 servers. You want to scale up to 50 fast and only pay for them when using them. Then scale down to 5 servers again and pay only for them.
- **Measured Service -** Resource usage in a Cloud environment is "pay as you go", meaning you only pay for the services and resources you are using. The cloud provides usage data for this kind of chargeback.

In this part of the chapter, we will focus on two of the five characteristics, on-demand self-service, and rapid elasticity, from an economic perspective. As Microsoft Azure supports these two characteristics of a cloud computing, we will use them to optimize the cost of our cloud services. One thing to remember is that cloud computing is a very competitive space and we often see changes in the pricing model. The only constants we can see is that the price of cloud services keeps falling and the number of available features is ever increasing.

When using Microsoft Azure virtual machines, it is important to understand what you are paying for. The total price of a VM is built upon a number of factors listed below.

- **VM size** - You are billed on an hourly basis, but on a prorated minute. If your VM is only running of 10 minutes you will pay for 10 minutes. If your VM runs for 10 minutes one week and 10 minutes the next week you will pay for 20 minutes. Microsoft Azure doesn't round up to the nearest hour. (Microsoft Azure did previously round to the nearest hour, but the practice was discontinued in 2013.) Microsoft Azure also offers a large number of different VMs sizes, from an extra small machine with a shared core and 768 MB RAM to an A11 machine with 16 CPU codes and 112 GB RAM.
- **VM type** - VMs in Microsoft Azure come in two types (tiers) - *Basic* and *Standard*.
    - **Basic** - This type of virtual machine is a cheaper than Standard type, but don't include auto-scaling or load balancing. These virtual machines can be used for applications that only run in one instance and don't need load balancing or auto-scale.
    - **Standard** - This type of virtual machine includes auto-scaling, load balancing, and internal load balancing capabilities (ability to load balance Azure VMs with private IP addresses, current in preview state). Standard tier machines also include more hardware options and VM sizes than Basic tier machines.
- **Region** - Depending on the region where the virtual machine is deployed, there can be a price difference. For example, during the writing of this book, a medium

basic tier machine cost $0.148 per hour in East USA, but $0.19 per hour in East Japan.

- **Storage** - You pay for everything stored on your storage accounts. Virtual machine hard disks are stored as VHD files on the Microsoft Azure Storage Blob service, on your storage account. The temporary disc (D:) is included in the virtual machine price but the temporary disc is stored on the physical host running the virtual machine and not on your storage account. The cost for storage can also depend on the redundant type you select for your storage. The operating system disk is charged at the regular rate for storage.
- **Tax** - When looking at Microsoft Azure Virtual Machine Overview on Microsoft TechNet, all prices don't include tax which will be added separately.
- **Bandwidth between Microsoft Azure regions -** If you are running virtual machines in two different regions, you will pay for the bandwidth when the two machines communicate.
- **Static IP address** – if you enable static IP addresses on a network adapter you are billed for the reservation, even if now data is transported through the network adapter.
- **Bandwidth between Microsoft Azure data center and "the world" -** Data transfers into Microsoft Azure data centers are free. Data transfers out of Microsoft Azure data center are billed. The price will depend on the number of gigabytes transferred.
- **Discount** - 6 or 12-month plans offer discounts, and if you pre-pay, you can get additional discounts.
- **Storage transactions** - For both read and write operations there is a fee, which affects all kind of storage.
- **Support** - Microsoft offers a number of different support offerings for Microsoft Azure. Depending if you are a developer doing some tests or an enterprise running business critical applications you can select a different level of support, with different prices.
- **Extensions** - When deploying a virtual machine from the Gallery in Microsoft Azure, you can select from a number of extensions. For example, antivirus and different agents. These extensions can add additional costs to your virtual machine.
- **VPN Gateway** - Configuration of virtual networks is free, but if you want to create a point-to-site or site-to-site VPN, you have to pay for each hour the VPN Gateway is online and available. Data going out of Azure data centers also cost extra.

A virtual machine can be in five different states. Depending on the VM state, resource allocation and usage will be billed:

- **Starting** - The VM is starting, going through the boot cycle. This period is billed as the virtual machine is running.
- **Running (Started)** - The running state of a virtual machine. In this state, the virtual machine is billed.
- **Stopped** - If the virtual machine is stopped from within the operating system the virtual cores are still billed. This is because the resources are still allocated to your virtual machine. This state can be beneficial in some situations. For example, when software updates are being applied, but you don't want to release the resources because this could result in losing a public mapped IP address on the virtual machine.
- **Stopped (Deallocated) -** The virtual machine is stopped from the Microsoft Azure Portal and de-mounted from the physical host. *In this state, the virtual machine is not billed.*
- **Deleted** - The virtual machine is deleted and no longer using any resources. No billing.

What about your Windows Server license when moving the virtual machine to Microsoft Azure? The Windows Server license is covered in the cost for the virtual machine in Microsoft Azure, meaning that you don't need an extra license for the virtual machine when running in Microsoft Azure. If you have applications running on the virtual machine that you upload to Microsoft Azure, you need to review Licensing Mobility rules for that application. For example, if you upload a virtual machine running SQL server it can be covered by license mobility benefits under Software Assurance. SQL Server 2012 is licensed per core, which is considered the equivalent to virtual cores in Azure virtual machine instances. If your SQL Server is not covered under Software Assurance, you can deploy a new virtual machine running SQL Server based on a template from the Gallery, and move your SQL databases to the new virtual machine. Then both Windows Server and SQL Server licenses are covered in the virtual machine per minute price.

If you are currently monitoring your servers with System Center Operations Manager, you may have System Center licenses based on your host servers, covering all virtual machines running on the hosts. System Center licenses can be translated when moving virtual machines to Microsoft Azure. A System Center Standard license can be used to manage two virtual machine instances. A System Center Datacenter license can be used to manage eight virtual machine instances.

Now that we have a good understanding of all the elements that will affect the cost of our Microsoft Azure virtual machines, we can begin to look at the migration process. The next section will cover migrating virtual machines from your on-premises datacenter to Microsoft Azure.

Migrate Machines to Microsoft Azure

This section of the chapter will talk about how you can migrate machines to Microsoft Azure. We will start with discussing how to migrate virtual machines running on Hyper-V. We will then cover migration of virtual machines running in VMware and end this section by talking about migrating physical servers to Microsoft Azure.

Migrate Hyper-V Machines to Microsoft Azure

If you are running Hyper-V in your on-premises datacenter, the migration between the on-premises datacenter and Microsoft Azure is not complicated. You can upload and download supported (Windows Server 2008 R2 or Windows Server 2012 all versions) versions of Windows Server easily. The migration process includes multiple steps:

1. Convert virtual machine hard disks to a supported format.
2. Upload virtual hard disk to a storage account in Microsoft Azure.
3. Create a disk in Microsoft Azure based on the virtual disk you uploaded.
4. Create a new virtual machine in Microsoft Azure and configure it to use the new disk.

Microsoft Azure only supports VHD files and not VHDX files. If your virtual machines using disks in the VHDX format today, you will need to convert them to the VHD format before uploading them to Microsoft Azure. To convert a VHDX file to VHD, you can use PowerShell, as described in the following steps.

> More information about the Convert-VHD cmdlet can be found in the TechNet library at http://technet.microsoft.com/en-us/library/hh848454.aspx.

1. Verify that the virtual machine that owns the VHDX file is not running.
2. On the Hyper-V server running the VHDX file you need to convert, open Windows PowerShell as Administrator.

3. In Windows PowerShell run,

```
Convert-VHD -Path C:\VM\server001.vhdx -DestinationPath
C:\VM\server001-new.vhd
```

Remember that your server might use multiple virtual hard disks. If this is the case, then all of the VHDX virtual hard disks will need to be converted. When the all virtual hard disks are in the VHD format, you can upload the VHD files to Microsoft Azure. There are multiple ways to upload VHD files to a storage account in Microsoft Azure. The three options are:

- **Windows Azure PowerShell** - Scripting can be a good solution if you want to do more than upload a VHD file. With Windows Azure PowerShell you can upload the VHD file, create the disk and then build and configure a new virtual machine.
- **Azure Storage Explorer** - Azure Storage Explorer is a graphical tool to work with the different storage types in Microsoft Azure. Azure Storage Explorer can be downloaded from CodePlex at http://azurestorageexplorer.codeplex.com/.
- **CloudBerry Azure Storage Explorer** - CloudBerry Lab offers a free version of the tool that can be used to manage files in Microsoft Azure and can be downloaded at http://www.cloudberrylab.com/free-microsoft-azure-explorer.aspx.

In this example, we will use Windows Azure PowerShell to upload a VHD file to Microsoft Azure. The **Add-AzureVHD** cmdlet will change the disk from dynamically sized to fixed size before upload. The **Add-AzureVHD** cmdlet only copies bytes that contain data and skips the empty ones, which will save some time during upload. It also verifies the upload with a checksum to make sure the file is correct.

1. Before starting to upload a VHD, make sure the server is configured to use DHCP and that Remote Desktop is enabled. Otherwise, you will not be able to connect to the server in Microsoft Azure. If you are uploading a Linux server, make sure SSH is configured and working.
2. First, download and configure Windows Azure PowerShell. Information how to download, install and configure Windows Azure PowerShell can be found at http://azure.microsoft.com/sv-se/documentation/articles/install-configure-powershell/
3. The following script uploads a VHD file to Microsoft Azure.

```
$source = "C:\SRV1010.vhd"
```

```
$destination =
https://portalvhd123.blob.core.windows.net/vhds/srv1010.VHD

Add-AzureVHD -LocalFilePath $Source -Destination
$Destination -NumberOfUploaderThreads 3
```

The first line of the script configures the source variable with the path to the file you want to upload. In this example, C:\SRV1010.VHD. The second line configures the destination variable for the file. The storage URL can be found easily in the Microsoft Azure Portal. Click on one of your storage accounts and then on the Dashboard tab. You will see the URL for the blobs storage. The last line of the script starts the upload. The upload will include three steps - create an MD5 hash, create a new page blob, and uploading. Uploading is the step that takes most of the time. Figure 9 and figure 10 show two of the three upload steps. Once the file is uploaded to Microsoft Azure, shown in figure 9, you can create a disk from the VHD file.

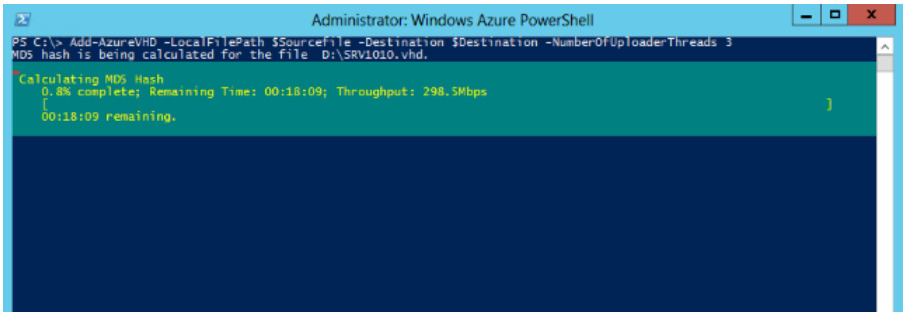


FIGURE 91. CALCULATING MD5 HASH FOR THE VHD FILE ABOUT TO UPLOAD

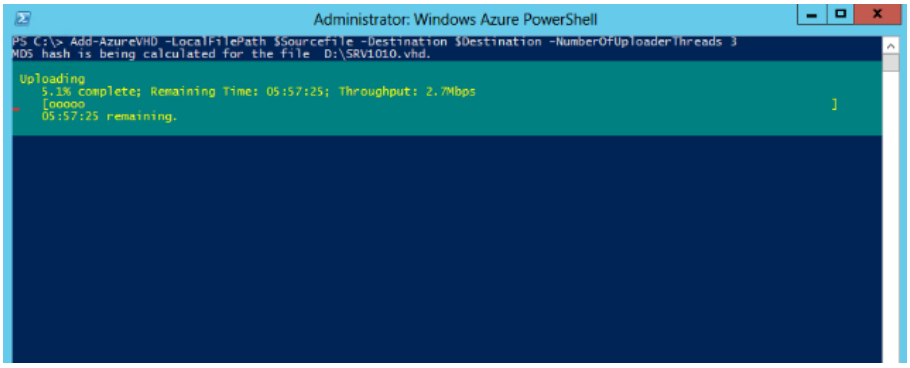


FIGURE 92. UPLOADING VHD FILE TO MICROSOFT AZURE

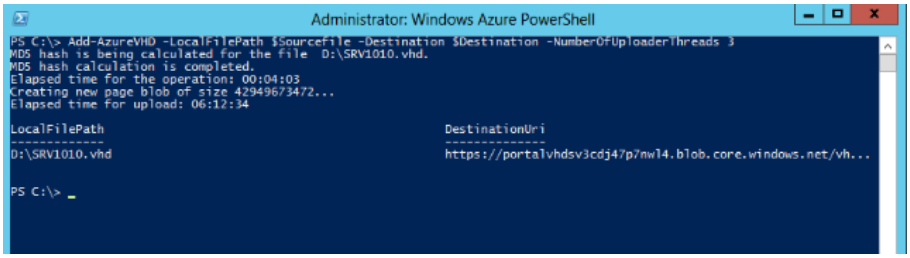


FIGURE 93. UPLOAD OF VHD FILE COMPLETED

To create a disk based on the VHD file you just uploaded, you need to use PowerShell. It is not possible to create an ARM based virtual machine from the Azure portal today. Use this script to create a new ARM-based Virtual Machine based on the VHD you just uploaded. Before you run the script you will need to update a number of variables:

- Change **SubscriptionID** to your own subscription ID
- Change **$location** to the location where you want to deploy the new virtual machine
- Change **$storageaccount** to the name the storage account that stores the VHD file
- Change **$rgname** to the name of your resource group where the storage account exist and the new virtual machine will be deployed
- Change **$VMname** to the name of the new virtual machine
- Change **$vmSize** to the size the new virtual machine should have
- Change **$osDiskUri** to the URL of the existing VHD file and change **$osDiskName** to the name of the VHD file
- Change **$VNET** and **$Subnet** to your network

```
# Logon to Azure and select subscription
Add-AzureRmAccount -SubscriptionId hb2f17a0-bgfs-4a1c-sd43-46172653455

# Set location for the new virtual machine
$location = "West Europe"

# Set storage account name for the storage account with the VHD file
$storageAccountName = "contoso00001"

# Configure Resource Group name
$rgName = "website"

# Get and set the storage account
$storageAcount = Get-AzureRMStorageAccount `
-StorageAccountName $storageAccountName -ResourceGroupName $rgName
Set-AzureRMCurrentStorageAccount -ResourceGroupName $rgName `
-StorageAccountName $storageAccountName

# Set new VM name
$VMname = "srv0001"
```

```
# Set new NIC name
$nicName = $VMName + "-" + "NIC01"

# Set size for new VM (Get-AzureRMVMSize -Location $location)
$vmSize = "Standard_D4"

# existing OS disk
$osDiskUri =
"https://contoso00001.blob.core.windows.net/vhds/srv0001.vhd"
$osDiskName = "srv0001.vhd"

# Get VNET and subnet for the new virtual machine
$VNET = Get-AzureRmVirtualNetwork -Name GeekCloudVNET `
-ResourceGroupName Geekcloud
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -Name subnet1 `
-VirtualNetwork $VNET

# Create NIC for the new virtual machine
$nic = New-AzureRMNetworkInterface -Name $nicName `
-ResourceGroupName $rgName `
-Location $location -Subnet $subnet

# Setup VM configuration for the new virtual machine
$vm = New-AzureRMVMConfig -VMName $vmName -VMSize $vmSize
$vm = Set-AzureRMVMOSDisk -VM $vm -Name $osDiskName -VhdUri $osDiskUri `
-CreateOption attach -Windows -Caching ReadWrite
$vm = Add-AzureRMVMNetworkInterface -VM $vm -Id $nic.Id

# Create the new virtual machine
New-AzureRMVM -ResourceGroupName $rgName -Location $location -VM $vm
```

The result of the script should be three resources in the resource group, the virtual machine, the storage account and the network interface, shown in figure 12.

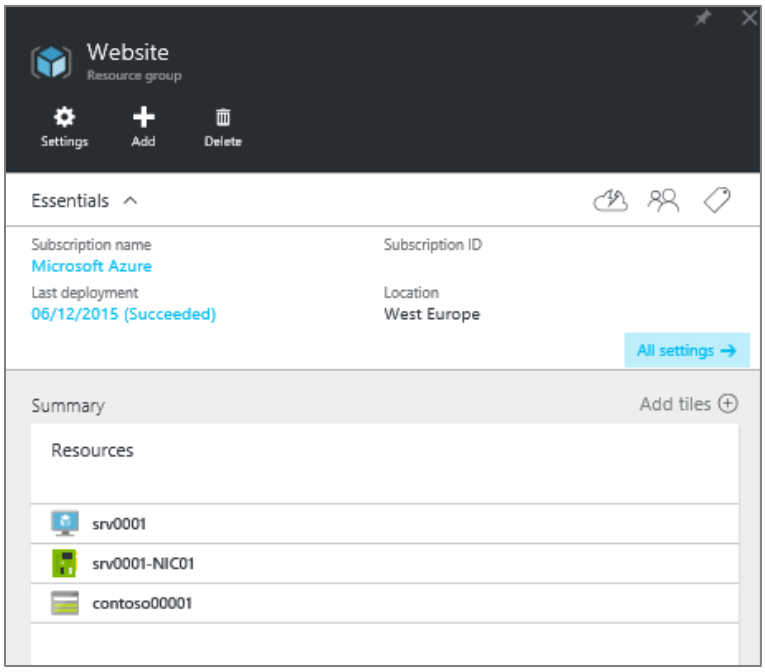


FIGURE 94. NEW VIRTUAL MACHINE, STORAGE ACCOUNT AND NETWORK INTERFACE IN THE RESOURCE GROUP

The new virtual machine is now deployed and ready to be used. If you remembered to enable remote desktop you can now connect to the server.

The new server that has been deployed uses the C: drive size configured in your Hyper-V environment. A standard server in Microsoft Azure have a C: drive size of 127 GB. The cost of the C: drive will be added on top of compute (VM Size) and will affect the total cost of the server. You are billed for the number of GB the disk is using, if your disk size is 127 GB but you only use 40 GB, you will only pay for 40 GB.

> **TIPS:** Vision Solutions, a Microsoft partner, sells a product that can migrate virtual and physical machines with near zero downtime. If you are planning to migrate a large number of virtual machines, it might be a good idea to evaluate this tool. More info can be found at the Vision Solutions website - http://www.visionsolutions.com.

Migrate VMware virtual machines to Microsoft Azure

What if you are running VMware virtual machines today but want to migrate to Microsoft Azure? Until now we have only talked about migrate virtual machines from Hyper-V to

Microsoft Azure. The Microsoft Virtual Machine Converter is a tool that can be used to migrate from VMware to Microsoft Azure. The Microsoft Virtual Machine Converter can migrate a virtual machine from its VMware host to Microsoft Azure, all in one wizard. The Microsoft Virtual Machine Converter has a fully scriptable command-line interface, which means it can be used inside data center automation workflows such as System Center 2012 Orchestrator runbooks or Azure Automation. It can also be invoked through Windows PowerShell. Figure 13 show the Microsoft Virtual Machine Converter wizard.

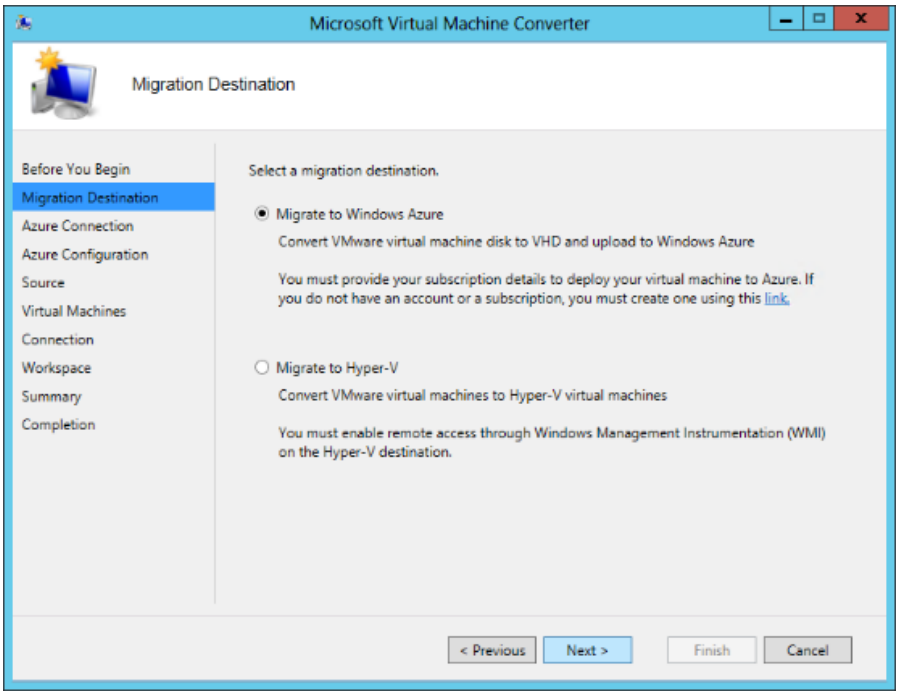


FIGURE 95. MICROSOFT VIRTUAL MACHINE CONVERTER WIZARD

If you have a large number of virtual machines running in VMware that you will convert into Microsoft Azure virtual machines, you may want to take a look the Migration Automation Toolkit. The Migration Automation Toolkit is a solution to run Microsoft Virtual Machine Converter in large scale. You can set up one central server, a "control server", which coordinates a number of workers, "helper node" servers. The Migration Automation Toolkit is a series of PowerShell scripts that use the powerful Microsoft Virtual Machine Manager Conversion engine and scale it to multiple conversions at the same time. Because the Migration

Automation Toolkit is built on PowerShell, the community can rebuild the scripts and enable new features on the toolkit.

Migrate Physical Machine to Microsoft Azure?

Until now we have only talked about virtual machines hosted in Hyper-V or VMware, but what about physical servers? If you have physical servers that you need to migrate to Microsoft Azure, you first need to perform a physical-to-virtual (P2V) conversion. If you are running Microsoft System Center 2012 Virtual Machine Manager SP1 (VMM), you can use built-in features for physical to virtual. System Center 2012 Virtual Machine Manager 2012 R2 don't support physical to virtual conversion. If you are not running Virtual Machine Manager, you can use standalone tools Microsoft Virtual Machine Converter.

Microsoft Virtual Machine Converter includes PowerShell cmdlets to run physical to virtual conversions. Together with other tools covered in this chapter you can automate the process of migrating physical servers to Microsoft Azure. One thing to keep in mind when converting physical machines in a highly virtual environment is that there is often a good reason why these machines have not already been converted to VMs. Make sure to verify machine roles and requirements before starting the migration.

Migration with Azure Site Recovery

Azure Site Recovery you can replicate virtual machines between on-premises and Azure. If on-premises goes offline virtual machines in Azure can start and take over the workload. Azure Site Recovery support protection of VMware, Hyper-V, and physical servers. Even if the primary scenario for Azure Site Recovery is disaster recovery it can be used in a migration scenario too. In a migration scenario, this service is free for 31 days, which gives you plenty of time to replicate the VHD file and failover to Azure. A great benefit compared with other migration tools discussed in this chapter is that Azure Site Recovery can migrate with near-zero downtime, it is also possible to test the migration before the final switch. The high-level steps are:

1. Setup the Azure Site Recovery agent (migrating from Hyper-V) or the Azure Site Recovery Role servers (migrating from VMware). Setup everything as a disaster recovery protection scenario
2. Make sure communication from VM hosts to the Azure service works
3. Setup replication for virtual machines to Azure

4. Once replication is in sync, all data synchronized and kept in sync while virtual machine still running on-premises, failover to Azure. The virtual machine on-premises will stop and a virtual machine will start in Azure. During the failover, there will be a near-zero downtime window
5. When the virtual machine is running in Azure, with an updated replica of the disk from the virtual machine on-premises, stop the protection
6. You now have a stopped virtual machine on-premises and a running virtual machine in Azure delivering the service

For more information about migration with Azure Site Recovery see Microsoft website, https://azure.microsoft.com/en-us/blog/azure-site-recovery-ga-move-vmware-aws-hyper-v-and-physical-servers-to-azure/.

Chapter Summary

This chapter started with talking about how to evaluate which servers to migrate to Microsoft Azure. We looked into how to collect inventory data and how to analyze it. We talked about pricing in Microsoft Azure for VMs. We talked about which servers and workloads that are suitable to move and which are not. You learned about all the different parameters and settings that affect the price for a VM in Microsoft Azure. Later the chapter discussed in depth how to migrate a VM from Hyper-V, and we looked at different tools and different processes. The chapter also covered how to migrate VMs from VMware to Microsoft Azure, and how to migrate physical servers to Microsoft Azure.

Chapter 8: Backup & Disaster Recovery

In this chapter, we will explore the backup and recovery options for both on-premises and Azure-based workloads that leverage Microsoft Azure. Before we delve into your Azure-integrated backup options, we will cover some terminology and concepts related to data and application backup and recovery. In this chapter, you will learn about the various options available, including:

- Onboarding in OMS
- Technology and Concepts
- System Center Data Protection Manager (DPM)
- Microsoft Azure Backup Server (MABS)
- Azure Backup
    - Backing Up Azure VMs
    - Backing Up On-Premises VMs
- Azure Site Recovery (ASR)
- Backing Up SQL Workloads to Azure Blob Storage
- Automated SQL Backup in Azure VMs
- 3rd Party Azure-Integrated Backup

Objectives for this chapter

The capabilities of Azure Backup and ASR are huge features and are always evolving. Rather than repeating step-by-step documentation that becomes quickly outdated, this chapter will focus on the following objectives:

- Provide a quick study on feature capabilities
- Explain how to identify which option(s) suit your use cases
- Point to current step-by-step documentation
- Offer recommendations and best practices for each feature

We will start with a look at OMS integration with Azure Backup and ASR.

Onboarding in OMS

At present, the integration between OMS and these two features is pretty light. OMS really only serves as a single pane of glass for navigating to Azure Backup and ASR in the Azure portal, as well as a licensing vehicle. You can license Azure Backup and ASR together through OMS Protection and Recovery for a single monthly per-node fee (currently $30 USD/node/month). Pricing details are available at http://bit.ly/2lU1j6k.

> **Note**: Technically, if you only require Azure Backup or ASR, you can buy them directly in the Azure portal. When purchased standalone, Azure Backup is billed based on the amount of storage consumed. ASR is priced per node. Note that the two services, when purchased separately standalone are more expensive than purchasing through OMS.

To enable both the Azure Backup and Azure Site Recovery features from the OMS Portal, perform the following steps:

1. In the OMS Portal, go to the **Solutions Gallery**.
2. Click **Protection & Recovery**.
3. Click **Add**.

At this point, both the Backup and Site Recovery solutions will reflect "Solution Requires Additional Configuration". The next step for both the Backup and Azure Site is to create a new Azure vault for the given feature.

For example, if you click on the Azure Site Recovery solution in the OMS Portal, you will see a link labeled "Create a new Azure Site Recovery Vault", as shown in figure 1.

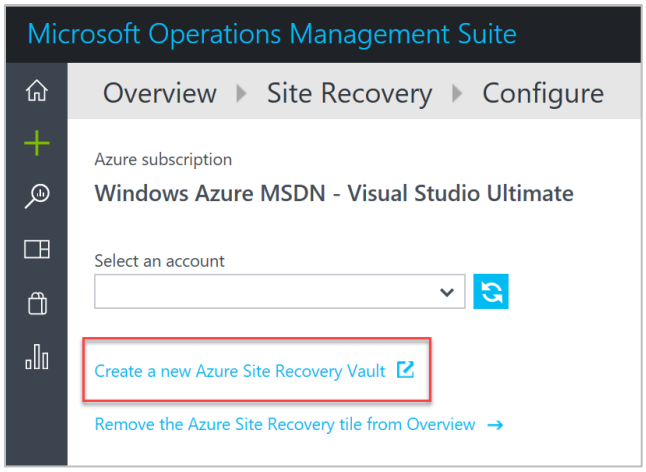


FIGURE 96. AZURE SITE RECOVERY SETTINGS

Likewise, both Backup and ASR will present a "Select an account" drop-down where you can select an existing vault if you have already created one. Once you select a vault, statistics will be presented in the tile in the OMS Portal.

When you click on the "Create a new…" link to create a vault for either solution, you will be taken from the OMS Portal to a new tab on the appropriate screen in the Azure Portal. While one can expect there will be deeper integration in the future, this is the extent of the integration between OMS and the Backup and ASR solutions today.

Terminology and Concepts

When discussing recovery services, it is important to distinguish backup services from disaster recovery (DR). Each of them is crucially important to a business, but they are very different in their goals and approaches. Backups are designed as safeguards for data loss while disaster recovery processes are designed to ensure business continuity.

Before we get into the Azure Backup and ASR solutions in greater depth, we will briefly touch on some concepts related to backup and disaster recovery that will influence your configuration decisions.

Backup

Backup procedures are designed to serve one purpose: prevent data loss in the event of a catastrophic failure. In the event of a failure, data can be retrieved and restored to a separate location from the original source.

Two metrics typically govern backup processes: Recovery Point Objective (RPO), and Recovery Time Objective (RTO). The RPO defines how frequently and how many backups are taken, and the RTO defines how long it takes to restore the data to a fully functional state. For example, an RPO might be defined to require a SQL server to be backed up once a day, and those backups kept for the last 14 days before they expire and space is reclaimed. It might also require sending a full backup offsite once a month. However, the RTO might be defined to require that any data from the last 14 days needs to be restored within 30 minutes to avoid downtime, while data that needs to be restored from two months ago, needs to be restored within 24 hours.

Disaster Recovery

Disaster Recovery is designed to serve a larger goal: which is to ensure the business is able to continue functioning in the event of a disaster. For this reason, disaster recovery tends to be tightly intertwined with Business Continuity Planning (BCP), as businesses today are unable to function without the technology systems as they have come to rely on in place and functioning.

When it comes to disaster recovery, there is one metric that is critical: the RTO. Since the business needs to continue functioning, the RTO needs to be as short as possible. For this reason, services deemed critical to the business's success are often architected from the ground up to be highly available (HA). However, solutions that are architected to be highly available are typically geographically constrained. That is, they are typically built within the same data center due to latency or throughput requirements. This is less than ideal in a disaster, where the entire data center might be entirely offline in the event of a catastrophic failure (perhaps even including the backup systems!).

Availability is typically defined in '9s'. Three '9s' availability requires 99.9% uptime, while five '9s' availability requires 99.999% uptime. There is a corollary to this, however: the more '9s' are required, the more expensive a solution becomes. There is a delicate trade-off in which disaster recovery RTO is balanced against budget constraints.

In BCP, selecting a failover site location that is far enough away from the primary site is very important. While there is no single correct answer, whether it is winter storms, hurricanes, power outages, or terror threats, you need to make sure that it is very unlikely

that a single event could take down both sites. Fortunately, with regional data centers around the world, it is easy to achieve the necessary distance when leveraged as a disaster recovery site.

Which Azure Backup components should I use?

If you aren't sure which Azure Backup component works for your needs, it may help to weigh your options based on a few simple questions:

- What data or application do I need to protect?
- How often do I need to take backups?
- What and how much data do I need to recover?
    - How often? And how quickly?
- What are my backup retention requirements?

To help sort through your options, the two tables below outline the capabilities and limitations of four different protection options:

- **Azure Backup (MARS) agent**. An agent you deploy to Windows systems that enables backup files and folders directly to an Azure Backup Vault. The agent is already present on VMs created from the Azure gallery. *This is going to be important for recovering individual files and folder*.
- **System Center 2016 Data Protection Manager (DPM)**. This is the enterprise solution that has been around since 2006, with features that support on-premises and hybrid recovery strategies, with support for disk-to-disk, disk-to-tape backup, as well as disk-to-disk-to-cloud. *You cannot implement disk-to-tape backup without DPM*.
- **Azure Backup Server (MABS)**. A lightweight DPM equivalent version for Azure customers, but does not offer disk-to-tape backup, as it is cloud-focused.
- **Azure IaaS VM Backup**. Azure Backup includes an agentless backup option of Azure VMs, using a backup extension, *designed for VM-level backups*.

What is protected? Where are backups stored?

What you have to protect and where backups will be stored, are two good opening questions. Your RPO and RTO requirements will factor strongly in determining which of these options is best. The answers for each option are shown in table 1.

Component	What is protected?	Where are backups stored?
Azure Backup (MARS) agent	• Files, • Folders	• Azure Backup vault
System Center DPM	• Files, • Folders, • Volumes, • VMs, • Applications, • Workloads	• Azure Backup vault, • Locally attached disk, • Tape (on-premises only)
Azure Backup Server	• Files, • Folders, • Volumes, • VMs, • Applications, • Workloads	• Azure Backup vault, • Locally attached disk
Azure IaaS VM Backup	• VMs, • All disks (using PowerShell)	• Azure Backup vault

TABLE 2. PROTECTION AND BACKUP STORAGE

Benefits & Limitations

As you can see in table 1, some of our options have overlapping capabilities, so we need to ask some additional question to determine which option is best for our circumstances.

Component	Benefits	Limits
Azure Backup (MARS) agent	• Back up files and folders on physical or virtual Windows OS (VMs can be on-premises or in Azure) • No separate backup server required.	• Backup 3x per day • Not application aware; file, folder, and volume-level restore only, • No support for Linux.
System Center DPM	• Application-aware snapshots (VSS) • Full flexibility for when to take backups • Recovery granularity (all) • Can use Azure Backup vault • Linux support on Hyper-V and VMware VMs • Backup and restore VMware VMs using DPM 2012 R2	Cannot back up Oracle workloads.
Azure Backup Server	• App-aware snapshots (VSS) • Full flexibility for when to take backups • Recovery granularity (all) • Can use Azure Backup vault	• Cannot back up Oracle workloads. • Always requires live Azure subscription • No support for tape backup

Component	Benefits	Limits
	• Linux support on Hyper-V and VMware VMs • Backup and restore VMware VMs • Does not require a System Center license	
Azure IaaS VM Backup	• Native backups for Windows/Linux • No specific agent installation required • Fabric-level backup with no backup infrastructure needed	• Back up VMs once-a-day • Restore VMs only at disk level • Cannot back up on-premises

TABLE 3. BENEFITS AND LIMITATIONS

Do I need DPM or MABS?

Before we get into the pure Azure scenarios like Azure IaaS VM protection and recovery, it is helpful to be clear about the capabilities and limitations of DPM and MABS, so you know when and where one or the other may be required.

DPM Capabilities

DPM is important for some on-premises or multi-site scenarios, including:

Broad support for Microsoft workloads

DPM was originally designed to protect Microsoft enterprise workloads from Windows Server, SQL, SharePoint, Exchange, and Hyper-V with support for Windows clients thrown in for good measure. In recent years, Microsoft has added support for Linux VMs hosted in Hyper-V, albeit only with file system-level consistency.

Additional Reading: Backup consistency level, including crash-consistent, file system consistent, and application-consistent are described in detail in a table in "Plan your VM backup infrastructure in Azure" on the Microsoft website at http://bit.ly/2m5AnAf.

For detailed workload support information, see the DPM support matrix in "Table 3. DPM Supported Scenarios" later in this chapter.

Disk-to-tape backup

None of the other solutions covered in this chapter cover backup to tape.

Site-to-site replication

One of the features of DPM is that it allows replication of one DPM server to another. This capability allows the DPM server itself to be redundant so that in the event a DPM backup server fails, it can be restored from the other, preferably a DPM server in another physical site. It is a 'backup for your backup', enabling transport of backups offsite, providing a disaster recovery capability.

System Center integration

DPM is the only solution covered here that offers any integration with Microsoft System Center components, such as integrated monitoring and management in System Center Operations Manager (SCOM).

DPM integration with Azure Backup

Not every business is large enough to have multiple data centers to which they can replicate their backup data through DPM. In some cases, businesses are moving to a cloud-first strategy, with no on-premises data center to replicate their cloud backups. This poses a disaster recovery challenge for these customers. Fortunately, Azure Backup offers a backup target for DPM. By installing a backup agent on the DPM server, protected sources can be replicated to Azure, ensuring that should the DPM server be lost in a catastrophic event, the backups are still available.

Backups to Azure are secured through encryption. A management certificate (generated outside of Azure) is used to encrypt API communications to Azure Backup, and the backup data itself is encrypted with a passphrase (that is also generated on-premises).

This ensures that the communication channel is always protected and that the data at rest is always protected and are only accessible by you.

> **STEP-BY-STEP**: For step-by-step configuration guidance, see "Configure Azure Backup to quickly and easily backup Windows Server" on the Microsoft website at https://msdn.microsoft.com/en-us/library/azure/dn337332.aspx

Microsoft Azure Backup Server (MABS)

Recently Microsoft came up with an initiative called Microsoft Azure Backup Server which is a DPM equivalent version for Azure customers. Azure Backup Server is functionally equivalent to DPM, with several notable exceptions, most importantly:

- MABS cannot be integrated with System Center components.
- MABS doesn't support Tape backup.
- MABS requires an active Azure subscription.

Like DPM, MABS also support application workload backups of Microsoft SQL Server, Hyper-V VMs, SharePoint Server, Microsoft Exchange and Windows Clients. This solution is ideal for customers who do not own System Center licenses to implement DPM, but owns an Azure subscription and/or planning to leverage Azure for data backups.

MABS is supported in both on-premises and in Azure IaaS VMs. If you deploy MABS on-premises and your backup strategy includes disk-to-disk backups only, MABS is totally free. MABS is a free product and you are only charged for the storage usage of the Azure Backup vault consumed for the backups if you are planning to implement a disk-to-disk backup on-premises, in addition to a disk-to-cloud backup strategy for long-term retention. Basically, you need MABS at a minimum to implement a disk-to-disk strategy on premises.

> **STEP-BY-STEP**: For a step-by-step guide on implementing Microsoft Azure Backup Server, see "Preparing to backup workloads to Azure with DPM" on the Microsoft website at https://azure.microsoft.com/en-us/documentation/articles/backup-azure-dpm-introduction/

Limitations of DPM with Azure VMs

While most of the application tier workloads have no change as they move to Azure Infrastructure-as-a-Service (IaaS), some of the sources that DPM can back up on-premises by their very nature cannot be backed up from Azure. For an example, Azure VMs are running on hypervisors, but tenant VMs are abstracted away from the hypervisor and DPM cannot access them at a hypervisor level. Azure SQL is a distributed, highly available SQL service that is also by its very design abstracted away so that tenants are unable to access the underlying resources (as Azure SQL is Platform-as-a-Service (PaaS) offering). This prevents DPM from backing up Azure VMs (at the VM level) or Azure SQL Tables, but note that you can back up an Azure VM at the OS level, as well as applications hosted in the VM.

Table 3 below outlines the Azure IaaS resources that are (and are not) supported by the latest version of DPM or MABS for recent versions Windows and enterprise workloads.

> Need info on legacy versions of Windows and server applications (older than 2 versions)? See the full support matrix at https://technet.microsoft.com/en-us/library/jj860400(v=sc.12).aspx.

Version	DPM installation	DPM 2016	Protection and recovery
System Center VMM			
VMM 2016, VMM 2012, SP1, R2	Physical server Hyper-V VM	Y	All deployment scenarios: Database
Client computers			
Windows 10 (64-bit and 32-bit)	Physical server Hyper-V VM	Y	Files Protected volumes must be NTFS. FAT and FAT32 aren't supported.

Version	DPM installation	DPM 2016	Protection and recovery
	VMware VM		Volumes must be at least 1 GB. DPM uses Volume Shadow Copy Service (VSS) to take the data snapshot and the snapshot only works if the volume is at least 1 GB.
Windows 8.1 (64-bit and 32-bit)	Physical server Hyper-V VM	Y	Files Protected volumes must be NTFS. FAT and FAT32 aren't supported. Volumes must be at least 1 GB. DPM uses Volume Shadow Copy Service (VSS) to take the data snapshot and the snapshot only works if the volume is at least 1 GB.
Windows 8.1 (64-bit and 32-bit)	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	Files Protected volumes must be NTFS and at least 1 GB.
Servers (32-bit and 64-bit)			
Windows Server 2016	Azure VM (when workload is running as Azure VM) Windows VM in VMWare (protects workloads running in Windows VM in	Y Not Nano server	Volume, share, folder, file, system state/bare metal), deduped volumes

Version	DPM installation	DPM 2016	Protection and recovery
	VMWare) Physical server On-premises Hyper-V VM		
Windows Server 2012 R2 - Datacenter and Standard	Azure VM (when workload is running as Azure VM)	Y	Volume, share, folder, file DPM must be running on at least Windows Server 2012 R2 to protect Windows Server 2012 deduped volumes.
Windows Server 2012 R2 - Datacenter and Standard	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	Volume, share, folder, file, system state/bare metal) DPM must be running on Windows Server 2012 or 2012 R2 to protect Windows Server 2012 deduped volumes.
Windows Server 2012/2012 with SP1 - Datacenter and Standard	Physical server On-premises Hyper-V VM	Y	Volume, share, folder, file, system state/bare metal DPM must be running on at least Windows Server 2012 R2 to protect Windows Server 2012 deduped volumes.
Windows Server 2012/2012 with SP1 - Datacenter and Standard	Azure VM (when workload is running as Azure VM)	Y	Volume, share, folder, file DPM must be running on at least Windows Server 2012 R2 to protect Windows Server 2012 deduped volumes.

Version	DPM installation	DPM 2016	Protection and recovery
Windows Server 2012/2012 with SP1 - Datacenter and Standard	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	Volume, share, folder, file, system state/bare metal DPM must be running on at least Windows Server 2012 R2 to protect Windows Server 2012 deduped volumes.
SQL Server			
SQL Server 2016	Physical server On-premises Hyper-V VM Azure VM Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y (UR2 Onwards)	All deployment scenarios: database
SQL Server 2014	Azure VM (when workload is running as Azure VM)	Y	All deployment scenarios: database
SQL Server 2014	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	All deployment scenarios: database
SQL Server 2012 with SP2	Physical server	Y	All deployment scenarios: database

Version	DPM installation	DPM 2016	Protection and recovery
	On-premises Hyper-V VM		
SQL Server 2012 with SP2	Azure VM (when workload is running as Azure VM)	Y	All deployment scenarios: database
SQL Server 2012 with SP2	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	All deployment scenarios: database
SQL Server 2012, SQL Server 2012 with SP1	Physical server On-premises Hyper-V VM	Y	All deployment scenarios: database
SQL Server 2012, SQL Server 2012 with SP1	Azure VM (when workload is running as Azure VM)	Y	All deployment scenarios: database
SQL Server 2012, SQL Server 2012 with SP1	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	All deployment scenarios: database
Exchange			
Exchange 2016	Physical server On-premises Hyper-V VM	Y	Protect (all deployment scenarios): Standalone Exchange server, database under a database availability group (DAG)

Version	DPM installation	DPM 2016	Protection and recovery
			Recover (all deployment scenarios): Mailbox, mailbox databases under a DAG
Exchange 2016	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	Protect (all deployment scenarios): Standalone Exchange server, database under a database availability group (DAG) Recover (all deployment scenarios): Mailbox, mailbox databases under a DAG
Exchange 2013	Physical server On-premises Hyper-V VM	Y	Protect (all deployment scenarios): Standalone Exchange server, database under a database availability group (DAG) Recover (all deployment scenarios): Mailbox, mailbox databases under a DAG
Exchange 2013	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	Protect (all deployment scenarios): Standalone Exchange server, database under a database availability group (DAG) Recover (all deployment scenarios): Mailbox, mailbox databases under a DAG
SharePoint			

Version	DPM installation	DPM 2016	Protection and recovery
SharePoint 2016	Physical server On-premises Hyper-V VM	Y (UR2 Onwards)	Protect (all deployment scenarios): Farm, frontend web server content Recover (all deployment scenarios): Farm, database, web application, file or list item, SharePoint search, front-end web server Note that protecting a SharePoint farm that's using the SQL Server 2012 AlwaysOn feature for the content databases isn't supported.
SharePoint 2013	Physical server On-premises Hyper-V VM	Y	Protect (all deployment scenarios): Farm, frontend web server content Recover (all deployment scenarios): Farm, database, web application, file or list item, SharePoint search, front-end web server Note that protecting a SharePoint farm that's using the SQL Server 2012 AlwaysOn feature for the content databases isn't supported.
SharePoint 2013	Azure VM (when workload is running as Azure VM) - DPM 2012 R2 Update Rollup 3 onwards	Y	Protect (all deployment scenarios): Farm, SharePoint search, front-end web server content Recover (all deployment scenarios): Farm, database, web application, file or list item, SharePoint search, front-end web server

Version	DPM installation	DPM 2016	Protection and recovery
			Note that protecting a SharePoint farm that's using the SQL Server 2012 AlwaysOn feature for the content databases isn't supported.
SharePoint 2013	Windows VM in VMWare (protects workloads running in Windows VM in VMWare)	Y	Protect (all deployment scenarios): Farm, SharePoint search, front-end web server content **Recover**: Farm, database, web application, file or list item, SharePoint search, front-end web server Note that protecting a SharePoint farm that's using the SQL Server 2012 AlwaysOn feature for the content databases isn't supported.
Hyper-V host - DPM protection agent on Hyper-V host server, cluster, or VM			
Windows Server 2016	Physical server On-premises Hyper-V VM	Y	**Protect**: Hyper-V computers, cluster shared volumes (CSVs) **Recover**: VM, Item-level recovery of files and folder, volumes, virtual hard drives
Windows Server 2012 R2 - Datacenter and Standard	Physical server On-premises Hyper-V VM	Y	**Protect**: Hyper-V computers, cluster shared volumes (CSVs) **Recover**: VM, Item-level recovery of files and folder, volumes, virtual hard drives

Version	DPM installation	DPM 2016	Protection and recovery
Windows Server 2012 - Datacenter and Standard	Physical server On-premises Hyper-V VM	Y	**Protect**: Hyper-V computers, cluster shared volumes (CSVs) **Recover**: VM, Item-level recovery of files and folder, volumes, virtual hard drives
Linux			
Linux running as Hyper-V guest	On-premises Hyper-V VM	Y	Hyper-V must be running on Windows Server 2012 R2 or Windows Server 2016. Protect: Entire VM **Recover**: Entire VM

TABLE 4. DPM SUPPORTED SCENARIOS

In the next section, we will look at the native data protection capabilities of Azure in the Azure Backup feature.

Azure Backup

Of course, not everyone has DPM in place for backups. Azure Backup is a feature that enables protection of both Azure IaaS VMs, as well as physical and virtual systems in your corporate data center. As you will see next, the process for protecting Azure VMs is streamlined versus the steps for on-premises VMs.

Features and Limitations

Today, there are a few capabilities and limitations in Azure Backup to be aware of:

- Supports client operating systems Windows 7 and later.

- Supports server operating systems Windows 2008 R2 and later.
- Supports VM backup with application-level consistency, but only supports VM, file and folder backups.
- Supports backup and recovery of Azure IaaS VMs running supported versions of Linux, but only with file system-level consistency.
- Cannot protect removable media, read-only volumes, network shares, BitLocker protected volumes and non-NTFS volumes.
- Protects volumes up to 1 TB.

Backup Azure VMs

Azure Backup for Azure VMs is a feature that makes protecting your Azure VMs a relatively simple task. Azure Backup for Azure VMs provides application-consistent backups for both Windows and Linux VMs with no downtime.

High-level Steps

The high-level steps to configure Azure Backup to protect Azure VMs are:

- Create an Azure backup vault (classic) or recovery services vault (IaaS v2) in the same Azure regional data center as the VMs you wish to backup.
- Run a VM discovery (which identifies unprotected VMs in the same Azure regional data center as your backup vault)
- Register the VMs that will be protected (backed up), including the backup policy
- Set the backup and retention policy

Because this all runs as-a-service, you can perform all the configuration in the Azure portal with no backup infrastructure required.

> **STEP-BY-STEP**: For step-by-step instructions for configuring protection of IaaS v2 VMs, see "Backup Azure virtual machines to a recovery services vault" at https://docs.microsoft.com/en-us/azure/backup/backup-azure-arm-vms.

How it works

We should start with a brief description of how the service works "under the hood". To back up an Azure VM, first, you need a point-in-time snapshot of the data. The Azure Backup service initiates the backup job at the scheduled time and triggers the backup extension to take a snapshot. The backup extension coordinates with the Microsoft VSS service in the Azure VM to achieve consistency (Windows VMs only). Once consistency is reached, the backup extension invokes the blob snapshot API of the Azure Storage service to get a consistent snapshot of the disks of the virtual machine (VM), without having to shut it down.

After the snapshot has been taken, the data is transferred by the Azure Backup service into the backup vault. The service handles the job of by identifying and transferring only the blocks that have changed from the last backup – making the backups storage very efficient. When the data transfer is completed, the snapshot is removed and a recovery point is created. You can view this recovery point in the Azure Portal.

Prerequisites

The primary prerequisite to configuring backups is creating a **backup vault** or **recovery services vault**. The Azure Backup service has two types of vaults - the Backup vault and the Recovery Services vault. The Backup vault came first. Then the Recovery Services vault came along to support the expanded Resource Manager deployments. Microsoft recommends using Resource Manager deployments unless you specifically require a Classic deployment.

Deployment	Portal	Vault
Classic	Classic	Backup
Resource Manager	Azure	Recovery Services

TABLE 5. AZURE VAULT TYPES

Create a Backup Vault

You can use the "Quick Create" option to create an Azure backup vault in few clicks, as you no longer have to create and upload an x.509 v3 certificate. In the Azure Portal, click **New → Recovery Services → Backup Vault → Quick Create**, as pictured in figure 2.

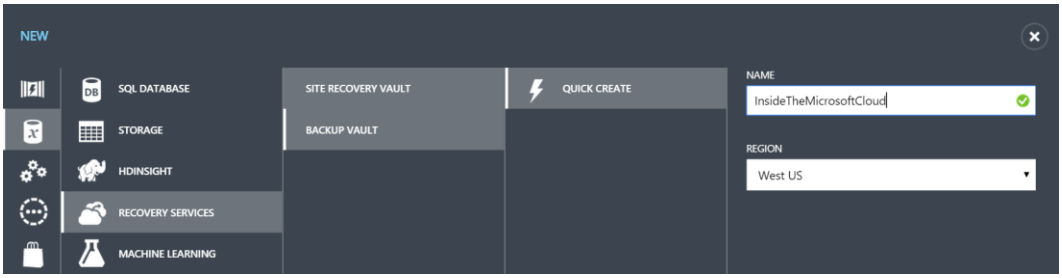


FIGURE 97. AZURE BACKUP VAULT "QUICK CREATE" OPTION

In case you need it, the step-by-step process for configuring a backup vault in the Azure management portal is available as a video guide on the Microsoft website in "Getting Started with Azure Backup 1 of 3 - Set up a backup vault on Azure" at https://azure.microsoft.com/en-us/documentation/videos/getting-started-with-azure-backup-1-of-3-set-up-a-backup-vault-on-azure/

Calculating data and cost protected instances

Azure VMs that are backed up using Azure Backup will be subject to Azure Backup pricing. The Protected Instances calculation is based on the actual size of the VM, which is the sum of all the data in the VM, excluding the resource (OS) disk. You are not billed based on the maximum size supported for each data disk attached to the VM but on the actual data stored on the data disk. Similarly, charges for the backup storage are also based on the amount of data stored with Azure Backup, which is the sum of the actual data in each recovery point.

The billing does not start until the first successful backup is completed. At this point, the billing for both storage and protected instances will begin.

For a complete reference on Azure backup pricing, you can refer "Backup Pricing" in Microsoft Azure documentation at https://azure.microsoft.com/en-us/pricing/details/backup/

Back up Azure IaaS v2 VMs

The high-level steps for backing up Azure IaaS v2 VMs are:

- Configure the backup vault or recovery services vault
- Configure the backup job (from the VM management blade in the Azure portal)
- Set backup goal, policy, and items to protect

The backup extension is installed by the Backup service whether the VM is running. A running VM provides the greatest chance of getting an *application-consistent* recovery point. However, the Azure Backup service continues to back up the VM even if it is turned off, and the extension could not be installed. This is known as Offline VM. In this case, the recovery point will be *crash consistent*.

> **STEP-BY-STEP**: For step-by-step instructions for configuring protection of IaaS v2 VMs, see "First look: Protect Azure VMs with a recovery services vault" at https://docs.microsoft.com/en-us/azure/backup/backup-azure-vms-first-look-arm.

Back up Classic Azure IaaS VMs

The high-level steps for backing up Classic Azure IaaS VMs are:

1. Create a backup vault or identify an existing backup vault.
2. Use the Azure Classic portal to discover and register the virtual machines.
3. Install the VM Agent.
4. Create the policy for protecting the virtual machines.
5. Run the backup.

> **STEP-BY-STEP**: For step-by-step instructions for configuring protection of Classic Azure IaaS VMs, see "First look: Backing up Azure virtual machines" at https://docs.microsoft.com/en-us/azure/backup/backup-azure-vms-first-look.

Security Features for Hybrid Backups

To protect data against cyber security issues like intrusion, ransomware and data exfiltration, Microsoft has added a number of security features to protect data in hybrid backup scenarios. The security features are designed to support three pillars

- **Prevention** - An additional layer of authentication is added whenever a critical operation like Change Passphrase is performed.
- **Alerting** - Email notification is sent to subscription admin whenever a critical operation like Delete Backup data is performed, ensuring administrators are aware of operations affecting access or recoverability.
- **Recovery** - Deleted backup data is retained for additional 14 days from the date of delete. This ensures recoverability of the data within given time period so there

is no data loss even if an attack happens. Also, the number of minimum recovery points are maintained to guard against corrupt data.

Security Features

There are three primary security features. Once enabled, you get these Security Features for all the Azure Recovery Services Agent (MARS) machines, Azure Backup Servers and DPM servers registered with the vault.

- **Retention of deleted backup data**. As a security measure, Azure Backup retains deleted backup data for additional 14 days (by default) and does not delete it immediately if Stop backup with delete backup data operation is performed. Microsoft a minimum retention range check based on the type of backup (daily, weekly, monthly or yearly), ranging from a minimum of one day for daily, up to one-year retention for deleted yearly backups.
- **Alerts and notifications**. Whenever some critical operations are performed, the subscription admin will be sent an email notification with details about the operation. However, you can configure additional IDs in the Azure portal.
- **Multiple layers of security**. As part of adding an extra layer of authentication for critical operations, you would be prompted to enter Security PIN when performing Stop Protection with Delete data and Change Passphrase operations.

> **Note**: These features are enabled by default for newly created recovery services vaults. Enabling this setting is a one-time action and you cannot disable these features after enabling them.

Security Feature Prerequisites & Limitations

Before you enable these features, there are some prerequisites and limitations to be aware of:

- **Agent & server versions**. These security features only support the following agents and server minimum version MABS agent (2.0.9052), Azure Backup Server (update 1), and DPM 2012 R2 (UR12) or DPM 2016 (UR2).
- **Vault support.** Only recovery services vaults are supported. These features are not available for backup vaults.
- **IaaS VM backup**. These security features are not yet available for IaaS VM backup. Enabling these features on IaaS backups will have no effect.

> **STEP-BY-STEP**: The steps to enable these features are simple and already well-documented on the Microsoft website in "Security features for protecting hybrid backups using Azure Backup" at http://bit.ly/2lqz2Ft.

Protecting Shielded VMs

Shielded VMs in Windows Server 2016 help protect sensitive VMs from inspection, tampering, and data theft (exfiltration) by malware and malicious administrators. DPM 2016 backups retain the protections provided by shielded VMs to ensure they can be recovered seamlessly and securely.

> **Note**: Currently, neither Azure Backup nor MABS support shielded VMs yet.

Trusted Platform Modules (TPM) are a chip in the motherboard of computers that help integrate cryptographic keys. These keys are used by BitLocker to protect the computer, even if it is stolen. Virtual TPM (vTPM) is a feature in Windows Server 2016, that allows you to use BitLocker and a virtual TPM chip to encrypt a VM with Bitlocker, protecting the VM. These VMs, called Shielded VMs, can only be run on healthy and approved hosts in the fabric.

DPM 2016 supports backup and recovery of Shielded VMs that have their VHDs/VHDXs protected with a virtual TPM.

> **Note**: Item Level Recovery (ILR) and Alternate Location Recovery (ALR) to a location outside the guarded fabric is not available for this scenario.

Recovering Individual Files and Folders

Recovery of individual files and folders is a new feature in Azure Backup, released in Preview in early 2017. Azure Backup will do an online backup by default for the first backup, but some customers might have a large amount of data (terabytes typically) that they want to ship by secure disk instead, which is possible via the offline backup workflow in Azure Backup. Once the first backup is completed, Azure Backup switches to a "changes only" backup.

A backup schedule is created when you complete the wizard, but a backup is not performed until the first scheduled time. You can wait for the first backup to take place, or you can trigger a manual backup by clicking **Back Up Now** in the Actions pane of the Azure Backup console.

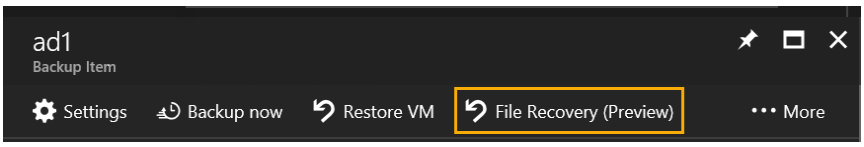
Instant File Recovery

The compelling value proposition of Instant File Recovery is how you restore individual files and folders using the VM-level backup, eliminating the need to backup files with the MARS agent, which doubles your effort and expense. With Instant File Recovery, you recover files and folders through a recovery point mounted using iSCSI, which points to your VM backup.

> **Note**: You will still need to install the January 2017 (version 2.0.9062.0) or later of the MARS agent to use Instant File Recovery. However, you do not need to create another backup beyond that created by the VM-level backup of your Azure VM.

To recover a file using the Instant File Recovery feature, perform the following steps:

1. Select your VM level backup in the Azure portal, by browsing to **< Recovery vault> > Backup Items > Virtual Machine > <VM backup item>**.
2. Next, click the **File Recovery (Preview)** menu item.



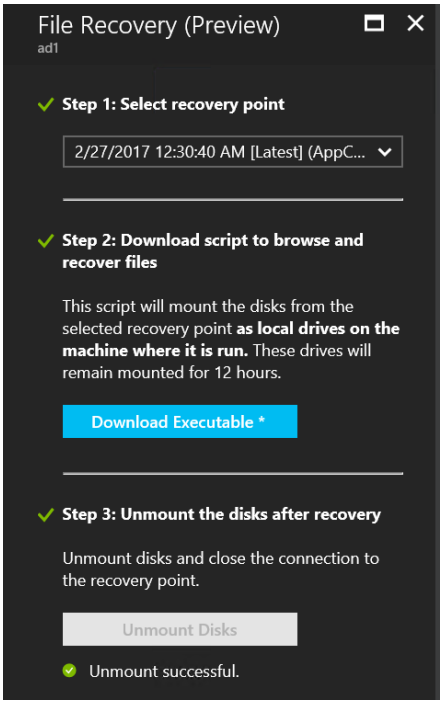


FIGURE 98. INSTANT FILE RECOVERY INTERFACE

3. Click the **Download Executable** button to download the script that will mount the recovery point.

   Be sure to select **Save**, not **Run**!

4. Once the executable download completes, you will need to save, right click and select **Properties**, then click the **Unblock** button present on all files downloaded from the Internet.

5. Then, run the executable as Administrator. If it runs successfully, you will see a result similar to that shown in figure 4.

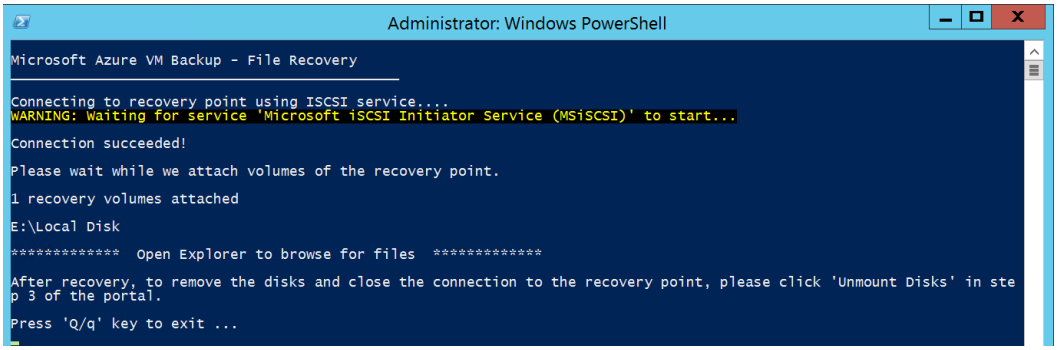


FIGURE 99. INSTANT FILE RECOVERY SCRIPT RESULT

IMPORTANT: if you do not save the File Recovery script and click the 'Unblock' button in the script, the script will return an error when you attempt to run it.

6. Basically, the on-premises computer (physical or virtual) mounts a recovery point in the recovery services vault in Azure, which appears as a usable volume in File Manager.

   At this point, you can browse through the file structure on the mounted drive, open files and folders to verify the content before you restore the content on the mounted local drive ('E:\Local Disk' in this example, as shown in figure 4). You simply copy the file from the mounted recovery point and paste to an alternate location, just as you would with Windows Previous Versions.

7. When you are done, you simply click the **Unmount Disks** button shown in figure 3 or let the mount point expire and auto-unmount in 12 hours.

> **Note**: At the time this chapter was written, this feature is only available in Azure Backup. It is not yet available with DPM or MABS.

Alerting and Monitoring on Backups

If you are an existing Azure backup customer using recovery services vault, update to the latest azure backup agent to use this feature. If you have configured email notifications before enrolling, turn off email notifications, enroll the subscription, and then configure notifications.

A short 5-minute video on alerting and monitoring capability for Azure Backup is available on the Microsoft website at http://bit.ly/2lTTa1P.

If you want to integrate the alerts into OMS, you could do this via the HTTP Data Collector API. For an example of the Data Collector API, see the chapter on Custom Solutions later in this book or visit "Send data to Log Analytics with the HTTP Data Collector API" on the Microsoft website at http://bit.ly/2lnHhy0.

VMware Backup with MABS

In late 2016, Microsoft announced MABS support for VMware VM backup to disk and to cloud for an offsite copy or long term retention.

- **Agentless Backup**. Azure Backup Server uses VMware's VADP API to protect VMware VMs remotely without installing agents on vCenter or ESXi servers, freeing administrators from the effort and hassle of managing agents for VMware VM backup.
- **Discoverability and Auto-Protection**. You can now discover and protect VMware VMs residing on external storage targets, such as NFS and cluster storage. Since VMs are discovered and protected at folder level automatically, managing large VMware environments much easier. Any future VMs added to a protected folder are backed up automatically.
- **Integrated Hybrid backup**. You can back implement both disk-to-disk on-premises for faster operational recovery, coupled with a disk-to-cloud copy for long term retention.

Note: You must download and install Update 1 to leverage the VMware backup feature, which is available on the Microsoft website at http://bit.ly/2lZLCL2.

STEP-BY-STEP: Step-by-step instructions for configuring the new VMware protection feature are available in "Four simple steps to backup VMware VMs with MABS" on the Microsoft website at http://bit.ly/2mkp9cd.

Azure Site Recovery

A key piece of any business continuity plan is a solid disaster recovery procedure. Azure provides a robust disaster recovery engine via Azure Site Recovery, which is capable of orchestrating seamless failover between your data centers. What's more, Azure Site Recovery also provides easy-to-use tools for testing your disaster recovery plan, ensuring that in the event of a disaster, your business continuity plan is already well tested, and your plan executes as expected.

Overview

With Windows Server 2012, Microsoft introduced Hyper-V Replica. This technology is built into the Hyper-V hypervisor role, enabling encrypted replication of the VM data and configuration from one Hyper-V host to another. Replication occurs at intervals of 30

seconds, 5 minutes, 15 minutes or 30 minutes. This helps to ensure that 'standby' virtual machines are always recent copies of mission-critical servers, shortening the recovery time in the event of a disaster. Replica VMs can be configured to utilize the same IP addresses as the source, or alternatively, use a different IP address space. Additionally, recovery points can be created, enabling recovery to an earlier point in the day (up to the last 24 hours). This functionality is all built-in free of charge in Windows Server 2012 and later.

Of course, in the event of a disaster, someone needs to make the decision to switch over to the replicas and initiate the process. Often, applications must be failed over in a specific order to account for dependencies (e.g. – SQL services require Active Directory to authenticate service and user accounts), opening up the possibility of human error in the failover process. In the event of a disaster, the human factor can become a bottleneck to a speedy recovery.

Microsoft Azure's Disaster Recovery as a Service (DRaaS) solution, Azure Site Recovery (ASR) drastically simplifies the failover process, enabling administrators to create groups of VMs that failover together, enabling single-click orchestrated failover in the event of a data center going offline. ASR leverages System Center Virtual Machine Manager (VMM) & Hyper-V Replica, by monitoring the VMM servers, and replicating configurations, snapshots, and data from one location to the destination. ASR can replicate the on-premises data center VMs from Hyper-V to Azure IaaS, enabling cost savings by eliminating the need for a second physical data center. This makes ASR a comprehensive, automated, and highly capable disaster recovery tool. Unlike the competitive solutions for ASR, you only pay for IaaS in the case of an actual failover while paying the storage cost only during idle times.

As of today, ASR has a few unsupported scenarios you should be aware of

- Unified Extensible Firmware Interface (UEFI)/Extensible Firmware Interface
- (EFI) boot is not supported.
- BitLocker encrypted volumes are not supported.
- Clustered servers are also not supported.
- Volumes larger than 1023 MB cannot be protected.

Let's take a look at the supported scenarios that ASR provide for creating a BCP solution.

1. **Hyper-V to Azure (no VMM)**

ASR acts as the replication broker in this scenario. This is ideal for SMEs with 1 – 5 individual Hyper-V hosts that aren't managed by VMM.

2. **Hyper-V to Azure (with VMM)**

This is for organizations with a larger number of Hyper-V hosts managed by either single or multiple VMM servers. ASR pulls the configuration data from VMM servers and you can leverage ASR recovery plans to automate an orchestrated recovery schedule.

3. **VMware to Azure**

This is a more popular scenario as there are many organizations that have invested in VMware for their datacenter workloads and looking for a cloud-based cost effective BCP solution rather than investing for hardware and VMware Site Recovery.

4. **Physical servers to Azure**

Similar to VMWare scenario. A physical site can be EC2 instances hosted in Amazon Web Services as well thereby providing a competitive advantage for your BCP solution.

5. **Hyper-V to a secondary Hyper-V site (with VMM)**

If you want to leverage on-premises Hyper-V Replica technology with VMM while letting ASR handle the orchestrated recovery for your critical workloads, then this is the solution for you.

6. **Hyper-V to a secondary Hyper-V site with SAN replication (with VMM)**

Leverages SAN replication technology provided by SAN vendors to provide near real-time replication. Following is the list of SAN vendors and respective SAN products that support the ASR recovery scenarios.

Supported array	Replication groups	Planned failover	Unplanned failover	Test failover	Reverse replication
NetApp Clustered Data ONTAP 8.2	Yes	Yes	Yes	Yes	Yes
HP 3PAR	Yes	Yes	Yes	Yes	Yes
EMC VMAX Series	Yes	Yes	Yes	Yes	Yes

TABLE 6. SAN SUPPORT FOR ASR RECOVERY SCENARIOS

7. **VMware VMs to a secondary VMWare site**

Traditionally you would use VMWare Site Recovery Manager to achieve this functionality. InMage Scout included in Azure Site Recovery provides makes this scenario possible thereby saving you the cost of SRM licensing if you already an Azure customer.

8. **Physical servers to a secondary site**

This scenario utilizes the InMage Scout technology, a company which Microsoft acquired in 2014. Again, you can use this to configure a BCP solution between two AWS datacenter locations.

> **STEP-BY-STEP**: Step-by-step guidance on configuring ASR, as well as a list of FAQs, is available on the Microsoft site at https://azure.microsoft.com/en-us/documentation/services/site-recovery/ or get the PDF version at https://docs.microsoft.com/pdfstore/en-us/Azure.azure-documents/live/site-recovery.pdf.

Between two Hyper-V Sites managed by SCVMM

The simplest and easiest implementation of ASR would be to use Hyper-V Replica. Using the native, out of box functionality in Windows Server requires the least amount of effort to configure. At a high level, the configuration process is as follows:

1) **Create an Azure Site Recovery Vault.** The Site Recovery Vault is essentially a grouping of assets and logical entities relating to your Site Recovery solution. It includes things like recovery plans, which define automated recovery actions, and how/which VMs will be failed over when the plan is executed.
2) **Install the Azure Site Recovery Provider on your VMM servers.** The Site Recovery Provider is essentially a plugin for VMM, enabling the VMM installations in each data center to plug into ASR. The provider is used to synchronize information about the Hyper-V servers, VMM clouds, snapshots, and VMs to Azure.
3) **Install the Microsoft Azure Recovery Services Agent on your Hyper-V hosts.** The Recovery Services serves to replicate data from Hyper-V to Azure. You can specify specific proxy connections for the agent to use if you do not want it to utilize the default internet settings on your Hyper-V host. The agent can be pushed out through standard methods, such as Group Policy or System Center Configuration Manager (SCCM).
4) **Configure protection settings for VMM clouds in the VMM console**. You can specify which VMM clouds you want to show up in Azure Site Recovery Manager,

minimizing the amount of visual noise in the Azure console. You can also define how the initial replications are performed, whether it be offline or online. Settings, such as the frequency of replication and number of recovery points in the last 24 hours can also be set, as well as the target cloud in the disaster recovery data center. Data compression can be turned on in this step as well, enabling smaller amounts of data to be transferred over the WAN, at the expense of additional CPU utilization on the source Hyper-V host. The authentication mechanism and port can also be chosen, enabling you to define more granular throttling policies at the network level. This ensures that WAN bandwidth is not completely consumed during replications.

5) **Configure network and storage mappings.** Network mappings allow you to define which VM Network a VM will be joined to when it fails over to the replica in the DR location, ensuring that it receives appropriate connectivity during failover. Storage mappings allow you to define storage classification mappings, ensuring that appropriate disk subsystems are available on the replicated VMs.
6) **Enable protection for virtual machines.** In the Azure Site Recovery Manager, you turn on protection for VMs on a VM by VM basis, ensuring that only necessary VMs are replicated from one data center to another.
7) **Configure recovery plans.** The recovery plan defines how VMs will be failed over, which VMs will be failed over, and in which order they will failover. Scripts and manual actions can be added to the plan, enabling automation of failover actions, as well as a pause for necessary manual actions. The manual actions must be marked as complete before the recovery plan continues with further actions.

How does failover work for Hyper-V VMs?

Below diagram provides a better understanding of how failover works in ASR for Hyper-V workloads. Some of the important steps are briefly described to provide an insight into what's happening under the hood in ASR.

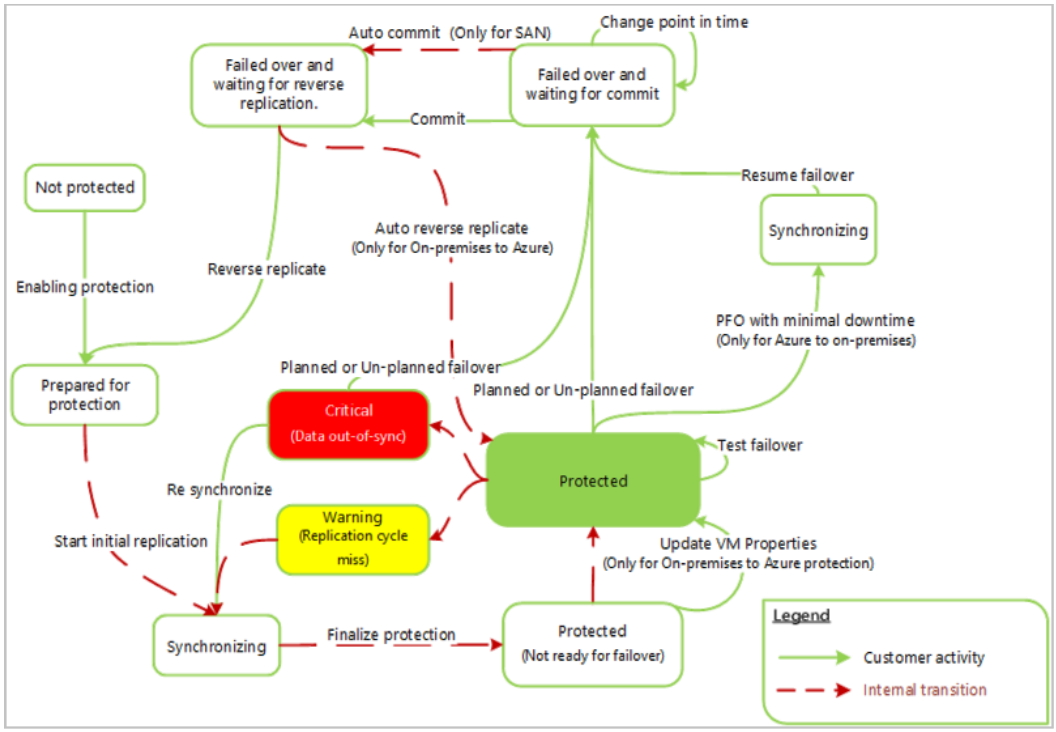


FIGURE 100. PROTECTION WORKFLOW FOR HYPER-V VMS

Enable protection: When you enable protection for VMs ASR will create and trigger an *Enable Protection* job which validates that the VM complies with prerequisites. At this point, the Hyper-V host will initialize a full VM replication to the ASR vault.

Initial replication: During this process, a snapshot of the VM is taken and VHDs are replicated one by one to Azure or to the secondary datacenter. One of the important facts to remember is that the duration to complete a full replication depends on the size of the VHDs, network bandwidth and the initial replication method you've chosen. Once the initial replication finishes the VM snapshot is deleted and the delta changes in the log are then synchronized and merged.

Finalize protection: This is the final step of enabling the protection where a **Finalize protection** job configures network and other post-replication settings for the VM. After this, you'll be able to change the settings for replica VM in Azure (i.e. instance size) so that you will not be overconsuming your Azure resources.

Replication Issues:

- **Replication failure**: This happens when delta replication fails and a full replication. This means that the data is out of sync and to avoid a full replication resynchronization occurs. A resynchronization operation computes checksums of the source and target VMs and sends only the delta changes to minimize the amount of data transfer. Once resync finishes delta replications resume. Although by default resynchronization is scheduled to run automatically outside office hours, you can resynchronize a VM manually whenever you require.
- **Replication error**: If you are experience network communication issues there is a high chance for the replication schedule to be missed. ASR will retry to continue the next cycle. For non-recoverable errors, such as If an authentication or authorization error, or a replica VM in an invalid state there will be no retry but for recoverable errors such as network errors, or low disk space/memory retries occur with increasing intervals between retries.

Planned/unplanned failovers: Keep in mind that failovers should be initiated only when a need arises and the type of a failover should match your BCP situation. For example, if your goal is to test your DR solution you should initiate a test failover, not an unplanned failover. In a planned failover source VMs are shut down and once replica VMs are created they're in a commit pending state. Unless you are utilizing SAN replication you should manually commit to complete the failover. Once you are through with a failover you can failback to the primary site where the reverse replication is automatic if your target site is Azure or otherwise you will have to manually trigger the reverse replication action.

> For a complete reference on failover scenarios, you can refer the Azure Site Recovery documentation at https://azure.microsoft.com/en-us/documentation/articles/site-recovery-failover/

VMware to Azure

In 2014, Microsoft acquired a company called InMage Systems. InMage offered a product that would replicate physical and VMware servers across the WAN between data centers. This acquisition led to integration and development of InMage functionality into the ASR

feature, resulting in an updated ASR feature with support for disaster recovery of VMware VMs.

Key features of this offering are:

- Heterogeneous workload support for multiple Windows and Linux editions with replication to and recovery in Azure.
- Automated discovery of VMware vCenter Server managed VMs for replication to and recovery in Azure.
- Continuous data protection with software-based replication to provide near-zero Recovery Point Objectives (RPO).
- On-the-fly conversion of source VMware Virtual Machine Disk (VMDK) files to bootable target Azure Virtual Hard Disk (VHD) files, ensuring low Recovery Time Objectives (RTO).
- Multi-VM consistency using ASR Protection Groups to ensure that all tiers of an n-tier application replicate consistently and fail over at the same time.
- Failback to VMware infrastructure from Azure when on-premises data center returns to service post-disaster.
- Active-passive replication that does not require running target Azure VMs at the time of replication, unlike other competitive disaster recovery products, thereby reducing the Total Cost of Ownership (TCO).
- Single-Click Failovers with ASR Recovery Plans to provide end-to-end workload-aware disaster recovery and orchestration at low Recovery Time Objectives (RTO).
- Health monitoring for replication, failover, and failback, with events and e-mail notifications.

Recently Microsoft has introduced an enhanced model for VMware/Physical to Azure scenario where the complication of the setup procedure has been much simplified. Previously you had to deploy Azure IaaS VMs to handle to incoming replication to an ASR vault but with this new model, VM data replicates directly to an Azure storage account. Some of the notable additions in the enhanced deployment scenario are listed below.

- **Recovery points**: Support for crash-consistent and application-consistent recovery points for Windows and Linux environments, and supports both single VM and multi-VM consistent configurations.
- **Test failover**: Support for non-disruptive test failover to Azure, without impacting production or pausing replication.

- **Unplanned failover**: Support for unplanned failover to Azure with an enhanced option to shut down VMs automatically before failover.
- **Failback**: Integrated failback that replicates only delta changes back to the on-premises site.
- **vSphere 6.0**: Limited support for VMware VSphere 6.0 deployments.

> **STEP-BY-STEP**: For step-by-step documentation on configuring VMware protection with ASR, see "Replicate VMware virtual machines and physical servers to Azure with Azure Site Recovery" on the Microsoft website at https://azure.microsoft.com/en-us/documentation/articles/site-recovery-vmware-to-azure-classic/

ASR & ARM

Up to this point, all the scenarios that we've discussed on ASR are compatible with the Azure Service Management (ASM) model. Azure Resource Manager (ARM) model supports most of the services offered by Azure while continuously adding and improving existing services to the new fabric.

Currently, the support for ARM is limited to:

- VMware VMs or physical servers to Azure
- Hyper-V VMs in VMM clouds to Azure
- Hyper-V VMs (without VMM) to Azure
- Hyper-V VMs in VMM clouds to a secondary site

This is achieved through ARM cmdlets for ASR and it is not an option in the new Azure portal. If you are planning to utilize ASR for the rest of the scenarios described in this chapter, you should keep in mind that you still need to leverage the ASM model.

> **STEP-BY-STEP**: A step-by-step guidance on configuring ASR with ARM cmdlets is available at https://azure.microsoft.com/en-us/documentation/articles/site-recovery-deploy-with-powershell-resource-manager/

Backing Up SQL Workloads to Azure

Since SQL Server is one of the most common (and often most critical) workloads hosted in Azure, coverage of options for protecting SQL server and databases instances in Azure VMs is warranted. Aside from the Azure Backup Server and Azure Backup service, there are multiple ways you can leverage Azure as part of your SQL backup and recovery strategy, including:

- SQL Database Backup to Azure Blob Storage
- Automated SQL Backup in Azure VMs

Both of these options are described in detail in this section, including a step-by-step tutorial on how to configure SQL database backups directly to Azure Blob Storage. The latter is an easy-to-configure and economical option.

SQL Database Backup to Azure Blob Storage

You can actually perform backups of SQL databases directly to a blob in Azure storage, even without a service like Azure Backup. This option will work for SQL Server instances running in on-premises VMs, as well as SQL instances running in Azure VMs.

Benefits

You can back up your databases to Microsoft Azure without having to manage devices and manage storage capacity on those devices. You can back up the databases to Microsoft Azure based on the activity of your databases. You can take a SQL log backup only when needed, such as when space used in database logs is running low. There are several benefits to this approach, including:

- Off-site, highly redundant storage to meet compliance regulations and industry standards.
- Lower management and operating costs, no hardware management, and low-cost storage.
- More time to focus on other tasks and no need to spend time managing storage for backups.

Considerations

There are a few things to bear in mind as you prepare to leverage SQL backup to Azure:

- A backup file can be a maximum of 1 TB in size.

- Backup and restore times will vary based on the bandwidth and latency of your network connection.

To backup databases from versions of SQL Server older than SQL Server 2014, you must download and install the "Microsoft SQL Server Backup to Microsoft Azure Tool". This tool enables backup to Azure from SQL Server 2005, 2008 and 2008 R2 databases with encryption capabilities. You can download this tool at http://www.microsoft.com/en-us/download/details.aspx?id=40740.

Step-by-Step: Performing a SQL Database Backup to Azure

In order to configure SQL database backups to Azure blobs, there are few things you that will need in Azure Storage:

- **Storage Account:** The storage account is the starting point for all storage services. To access the Windows Azure Blob Storage service, begin by creating an Azure storage account. The **storage account name** and its **access key** properties are required to authenticate to the Azure Blob Storage service and its components.
- **Container:** A container provides a grouping of a set of Blobs, and can store an unlimited number of Blobs. To write a SQL Server backup to the Azure Blob service, you must have at least the root container created.
- **URL:** A URL specifies a Uniform Resource Identifier (URI) to a unique backup file. The URL is used to provide the location and name of the SQL Server backup file. In this implementation, the only valid URL is one that points to a page Blob in an Azure storage account. The URL must point to an actual Blob, not just a container. If the Blob does not exist, it is created. If an existing Blob is specified, the backup fails, unless the "WITH FORMAT" option is specified.
- **Blob:** A file of any type and size. There are two types of blobs that can be stored in the Azure Blob storage service: block and page blobs. SQL Server backup uses page blobs as the blob type. Blobs are addressable using the following URL format: https://<storage account>.blob.core.windows.net/<container>/<blob>.

IMPORTANT: If you choose to copy and upload a backup file to the Microsoft Azure Blob storage service, use page blob as your storage option. Restores from Block Blobs are not supported. RESTORE from a block blob fails with an error.

- **Credential:** A SQL Server credential is an object that is used to store authentication information required to connect to a resource outside of SQL Server. SQL Server backup and restore processes use the credential to authenticate to the Azure Blob storage service. The Credential stores the name of the storage account and the storage account **access key** values.

At a high level, following are the steps for creating SQL workload backups:

- Create Azure Storage Account
- Create Azure Storage Container
- Create Credential
- Backup Using the Backup Task (in SSMS)
- Backup Using T-SQL BACKUP DATABASE Command

Once you have a created a backup, you may be interested in how to recover a database using a backup hosted in Azure storage. The restore process will be covered in the "Restoring Data from an Azure Blob" section later in this chapter.

Create Azure Storage Account:

If you are not already logged into the Azure Portal, open a supported web browser and browse to https://portal.azure.com then sign in using your credentials.

1. Click **Browse** & Search for **Storage Accounts** from the navigation pane on the left, as shown in figure 6. Here we are creating a v2 storage account, not a classic storage account.

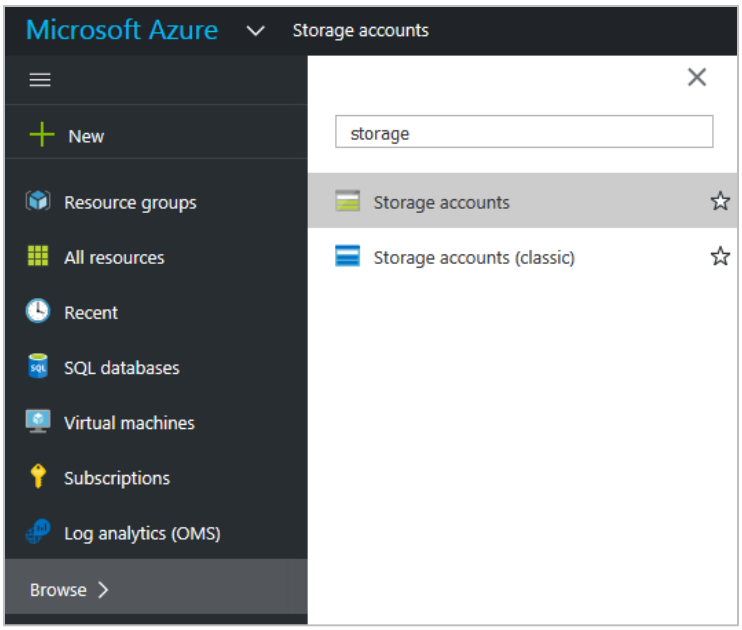


FIGURE 101. STORAGE BLADE IN THE AZURE PORTAL

2. Click +**Add in the Storage Blade**

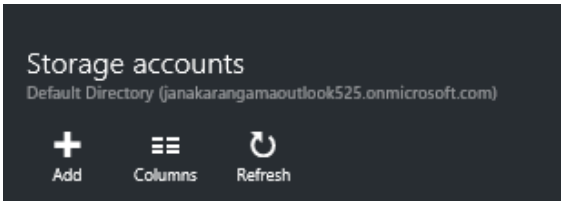


FIGURE 102. STORAGE CONTAINER QUICK CREATE

3. Provide a name for the new storage account, select the storage account type and select an existing or create a new resource group for the storage account as in figure 8.

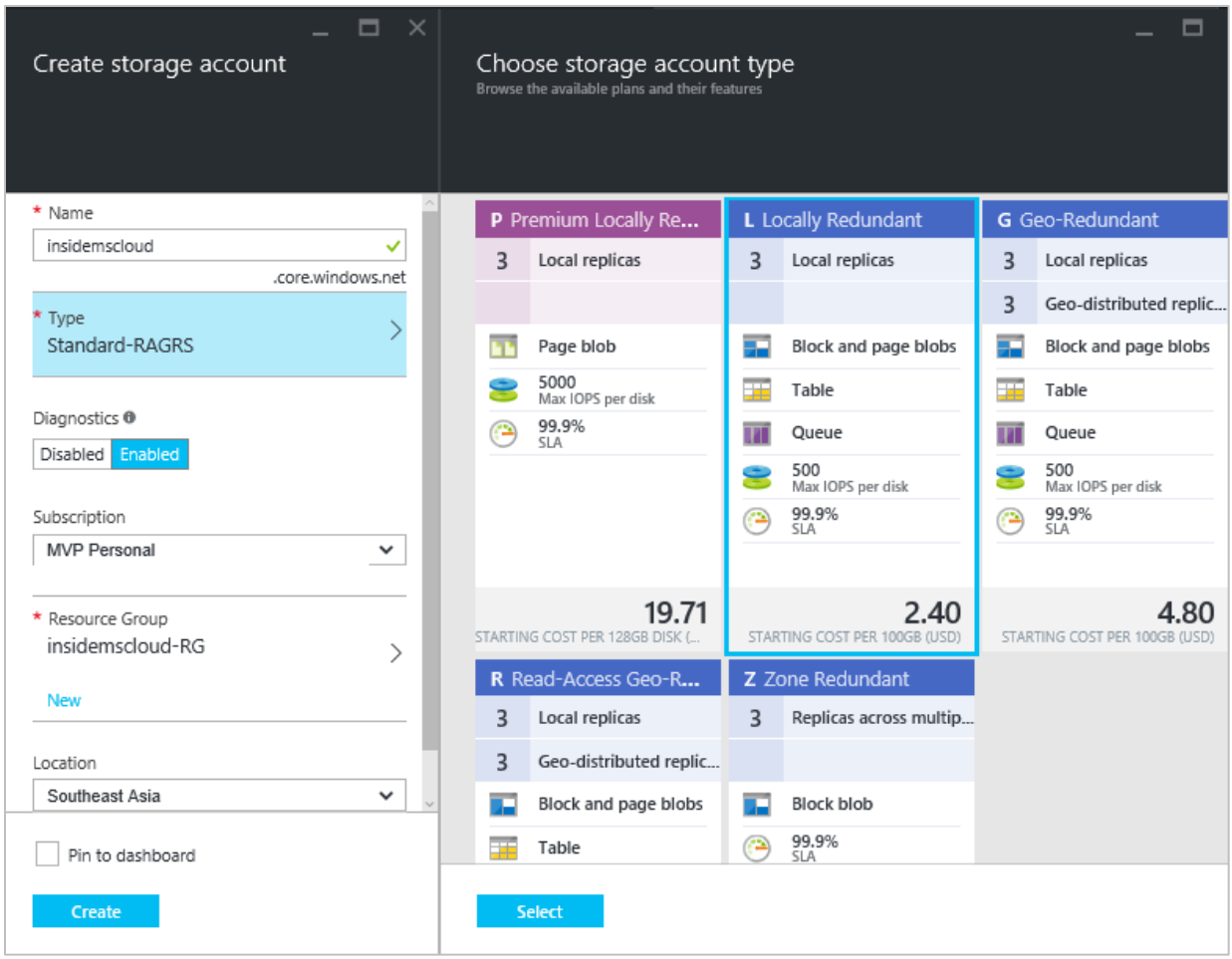


FIGURE 103. STORAGE ACCOUNT CREATION IN NEW AZURE PORTAL

4. In **Location**, select the geographic region for the storage account.
5. Click **Create** to create the new storage account.

Create Azure Storage Container

To create a storage container in Azure, perform the following steps:

1. In the Azure Portal, select the storage account you previously created, then select the **Blobs** Tile, as shown in figure 9.

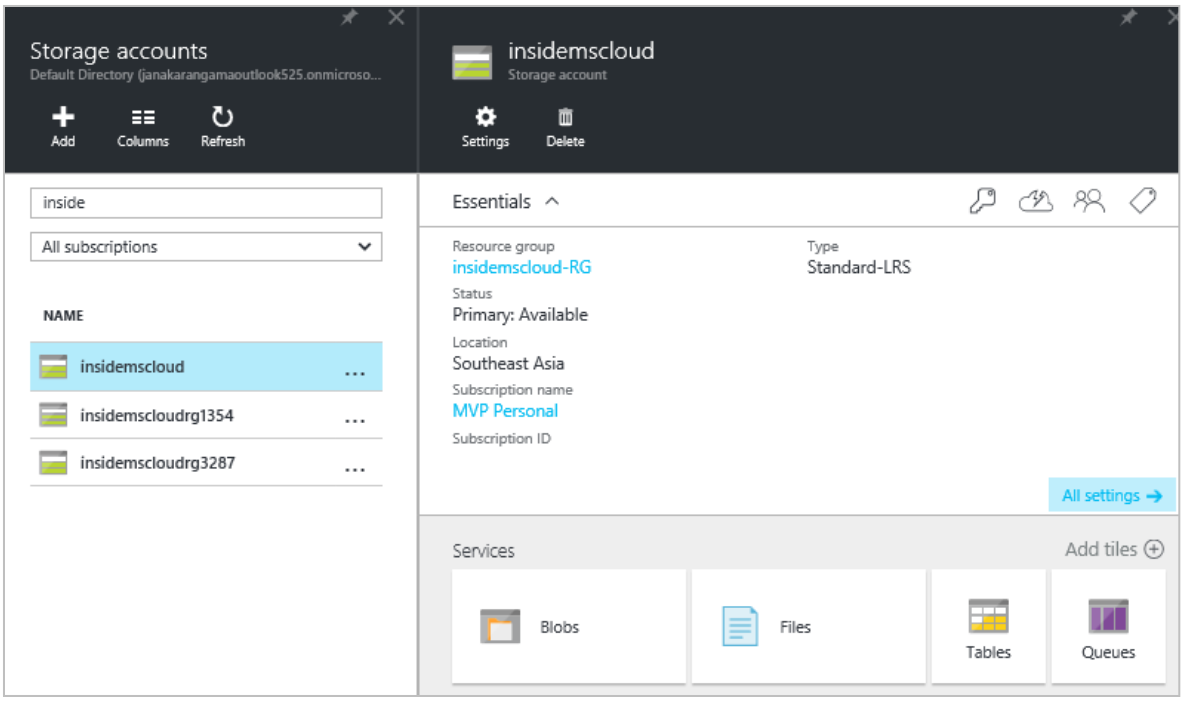


FIGURE 104. STORAGE BLOBS TILE

2. Click **+Container** button to open the **New container** dialogue.

FIGURE 105.BLOB SERVICE DIALOG BOX

3. Enter a unique value in the **Name** field, and set a value for **Access** by selecting **Private** and click **Create** as shown in Figure 11 below.

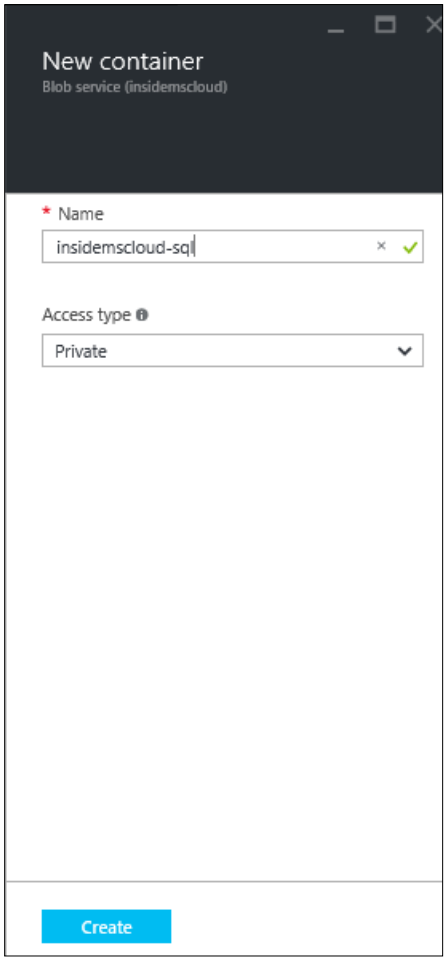


FIGURE 106. NEW CONTAINER DIALOG BOX

Create Credential

To create a credential for accessing and writing backups to blobs in Azure storage, perform the following steps:

1. Launch **SQL Server Management Studio (**SSMS).
2. Expand the **Security** node.
3. Right click the **Credentials** node and select **New Credential** from the context menu, as shown in figure 12. This opens a **New Credential** window.

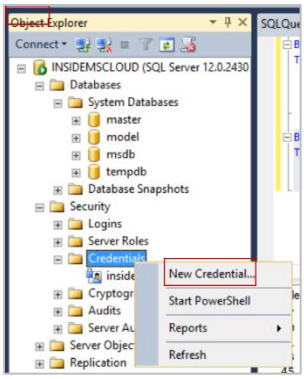


FIGURE 107. NEW CREDENTIAL MENU IN SSMS

4. In the **New Credential** window (shown in figure 12), specify the following values for ease of use:
    - **Credential Name** – Use the name of the storage container
    - **Identity** – Use the name of the storage account
    - **Password** – The access key from the storage container. You can find this value by selecting the storage container in the Azure portal, and at the **Settings** Blade, selecting **Access Keys**. Copy the primary access key(KEY1), as shown in figure 13.

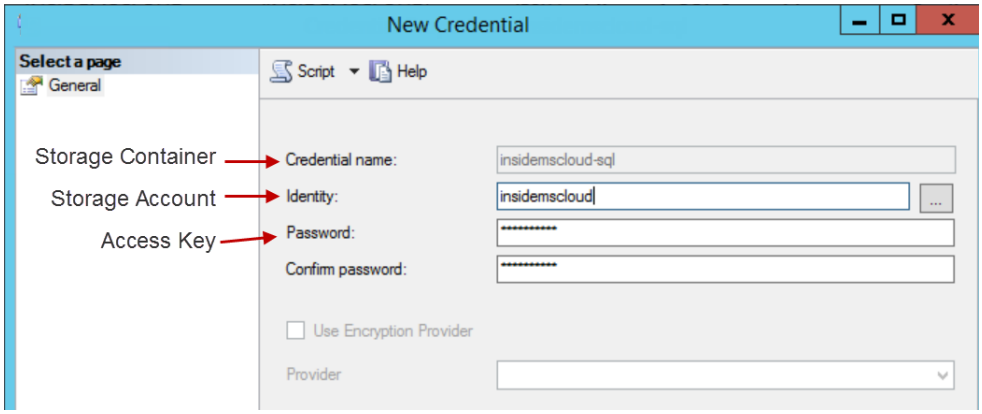


FIGURE 108. NEW CREDENTIAL DIALOGUE

5. When you have filled in the values above, click **OK**.

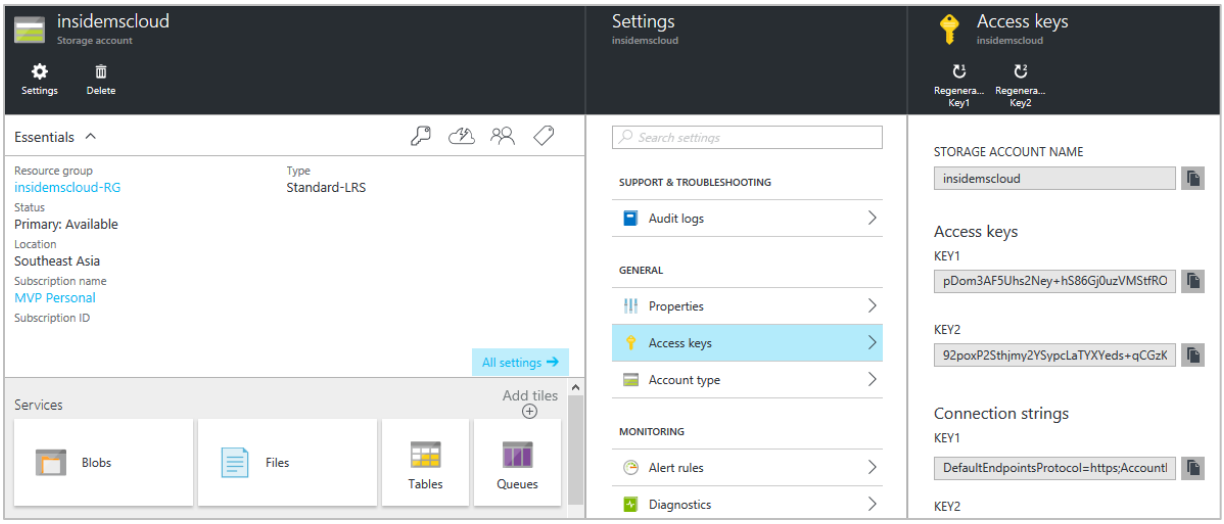


FIGURE 109. MANAGE ACCESS KEYS BLADE IN THE AZURE PORTAL

You are now ready to perform a SQL backup to Azure storage.

Backup Using the Backup Task in SSMS

The Backup task in SQL Server Management Studio has been enhanced to include URLs as one of the destination options, and other supporting objects required to backup to Windows Azure storage, such as the SQL Credential.

To backup a SQL database to Azure storage, perform the following steps:

1. Start SQL Server Management Studio and connect to the SQL Server instance. Select a database you want to backup, right click on **Tasks**, and select **Backup...**, as shown in figure 15. This opens the **Backup Database** dialog box.

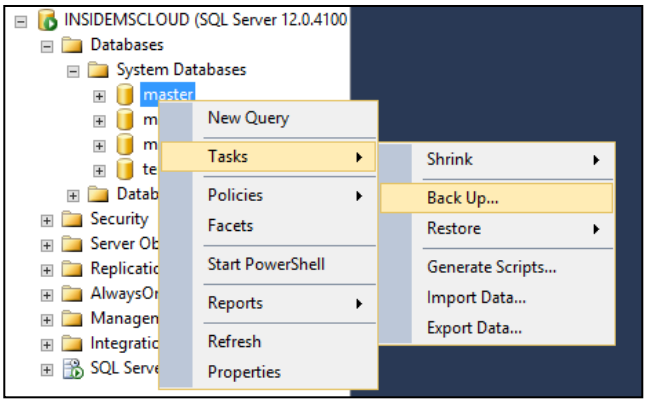


FIGURE 110. SQL BACKUP WIZARD IN SSMS

2. On the general page, select the **URL** option to create a backup to Azure storage, as shown in figure 16. When you select this option, you see other options enabled on this page:
    a. **File Name:** Name of the backup file.
    b. **SQL Credential:** You can either specify an existing SQL Server Credential or create a new one by clicking on the **Create** next to the SQL Credential box.

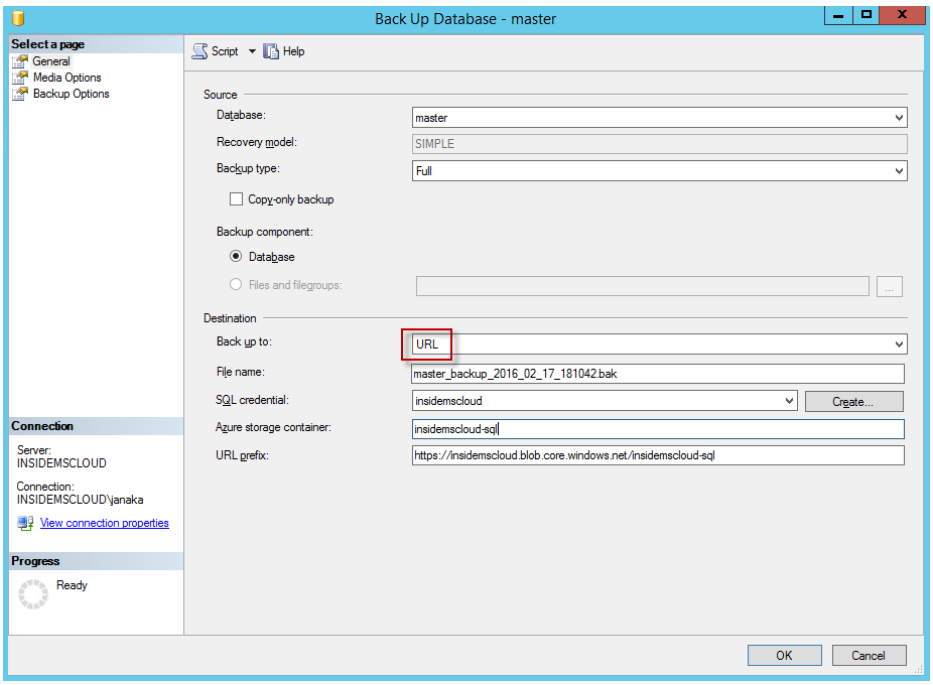


FIGURE 111. BACKUP DATABASE DIALOGUE

NOTE: The dialog that opens when you click **Create** requires a management certificate or the publishing profile for the subscription. SQL Server currently supports publishing profile version 2.0. It is easier to simply create the credential as documented in the previous step.

c. **Azure storage container:** The name of the Windows Azure storage container to store the backup files.
d. **URL prefix:** This is built automatically using the information specified in the fields described in the previous steps. If you do edit this value manually, make sure it matches with the other information you provided previously. For example, if you modify the storage URL, make sure the SQL Credential is set to authenticate to the same storage account.

Backup Using T-SQL BACKUP DATABASE Command

To backup a database to Azure storage using a T-SQL query, perform the following steps:

1. In SQL Server Management Studio, right-click the database you want to backup and select **New Query**.
2. Enter the following T-SQL command into the query window, replacing the database name, URL, and Credential with the values from your environment. Specify a file name on the end of the URL, which will be created at runtime. There are two examples below,

   Backup (default compression, which is no compression)

```
BACKUP DATABASE Master
TO URL =
'https://insidemscloud.blob.core.windows.net/insidemscloud-
sql/MasterNoCompress.bak'
      WITH CREDENTIAL = 'insidemscloud-sql'
     ,STATS = 1
```

   To turn compression on, simply add the COMPRESSION option as shown below

```
BACKUP DATABASE Master
TO URL =
'https://insidemscloud.blob.core.windows.net/insidemscloud-
sql/MasterCompress.bak'
      WITH CREDENTIAL = 'insidemscloud-sql'
     ,COMPRESSION
     ,STATS = 1
```

When finished, check the target Azure storage container to verify the backup is present. In figure 17, you will notice that the backup file created with COMPRESSION specified is much smaller than when compression is not specified.

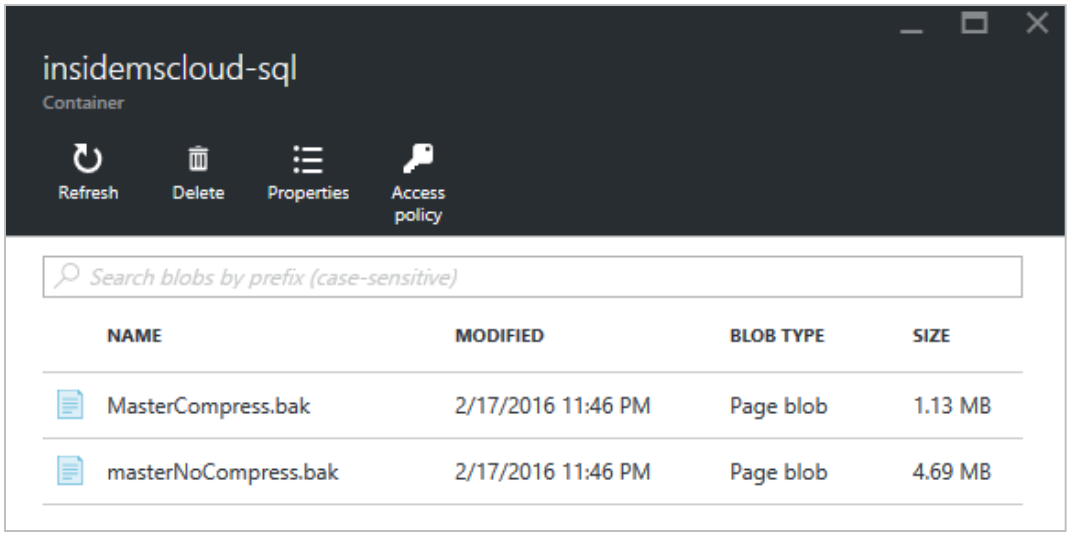


FIGURE 112. SQL BACKUP FILES IN AZURE STORAGE

> **TIP**: Backing up with the COMPRESSION option means lower storage usage and thus, lower costs to store your SQL database in Azure. Notice the size difference with and without compression shown in figure 17.

Now that you have created a database backup, you can attempt to restore a backup from Azure storage.

Restoring from Microsoft Azure Storage

If you are restoring a database, **URL** is included as the device to restore from. The following steps describe the changes in the **Restore** task to allow restoring from Microsoft Azure storage:

1. Right-click on the database you want to restore and select **Tasks → Restore → Database**, as shown in figure 18.

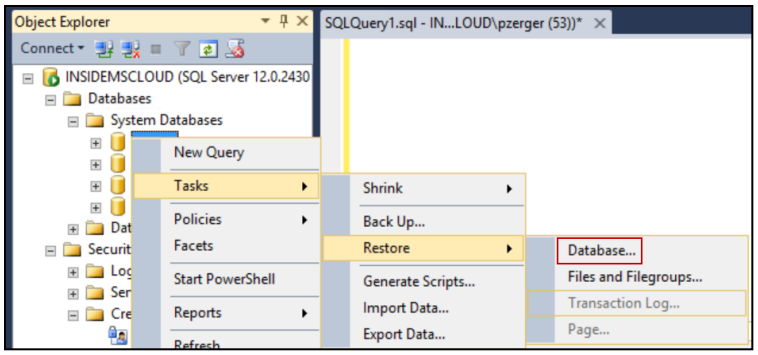


FIGURE 113. LAUNCHING THE RESTORE DATABASE DIALOGUE

2. When you select **Devices** in the **General** page of the Restore task in SQL Server Management Studio, this takes you to the **Select backup devices** dialog box, which includes **URL** as a backup media type.
3. Select **URL** and click **Add**. This opens the **Connect to Azure storage** dialog. Specify the SQL Credential information to authenticate to Azure storage, as shown in figure 19.

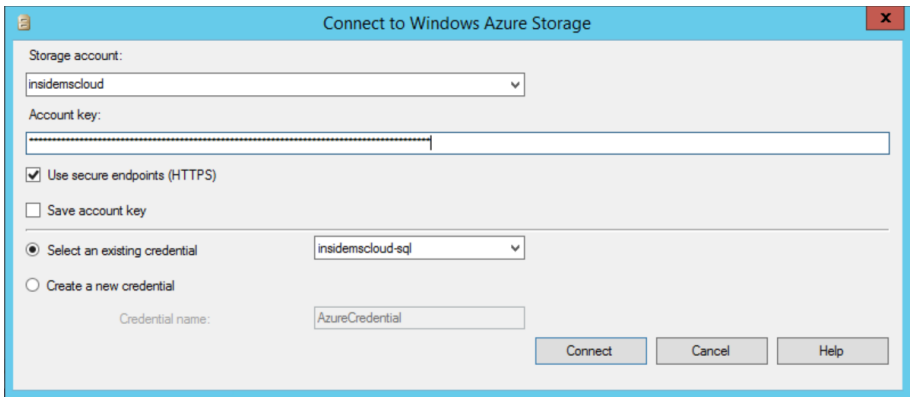


FIGURE 114. CONNECT TO WINDOWS AZURE STORAGE DIALOGUE

4. SQL Server then connects to Azure storage using the SQL Credential information you provided and opens the **Locate Backup File in Windows Azure** dialog. The backup files residing in the storage are displayed on this page. Select the file you want to use to restore and click **OK**.
   This takes you back to the **Select Backup Devices** dialog, and Clicking **OK** on this dialog takes you back to the main **Restore** dialog (shown in figure 20) where you will be able complete the restore.

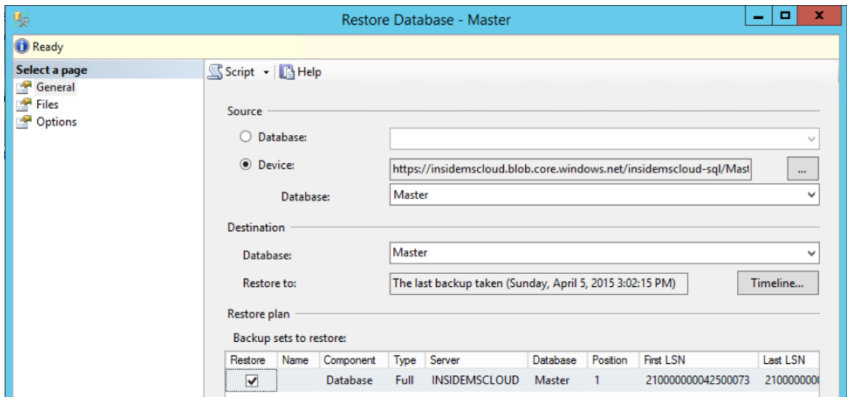


FIGURE 115. RESTORE DATABASE DIALOGUE

5. You can use the **Script** menu to create a T-SQL restore script and save to file, clipboard or open in a new query editor window, as shown in figure 21. You can also select the **Agent Job** option, which will bring your selections in the Restore Database wizard into the **Schedule Job** interface.

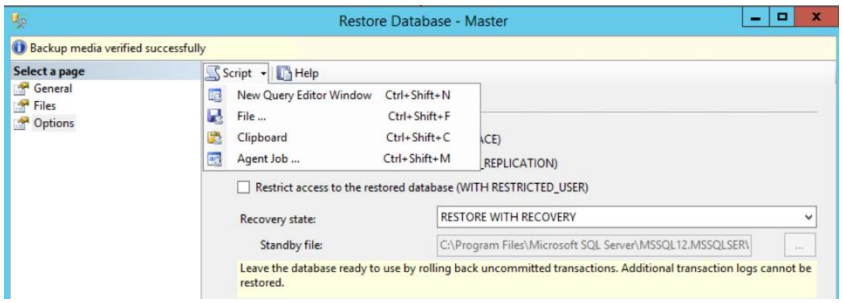


FIGURE 116. RESTORE DATABASE DIALOGUE

As you can see, backing up and recovering SQL databases from backups stored in Azure is a straightforward process.

Automated SQL Backup in Azure VMs

A key piece of any business continuity plan is a solid disaster recovery procedure. Azure provides a storage platform that enables organizations to store their SQL backup offsite. The following products are compatible with the SQL Server IaaS Agent features.

Automated Backup:

- Windows Server 2012
- Windows Server 2012 R2
- SQL Server 2014 Standard
- SQL Server 2014 Enterprise

Automated Patching:

- Windows Server 2012
- Windows Server 2012 R2
- SQL Server 2012
- SQL Server 2014

> **NOTE:** You can automate patching of SQL 2012 and 2014 VMs, but the automated backup feature is only available for SQL 2014.

Automated Backup automatically configures Managed Backup to Microsoft Azure for all existing and new databases on an Azure VM running SQL Server 2014 Standard or Enterprise. This enables you to configure regular database backups that utilize durable Azure blob storage. The actual configuration steps vary depending on whether you use the Azure Portal (pictured in figure 22) or Azure Windows PowerShell commands.

Azure Service Management (ASM) Option

If you want to protect classic Azure VMs following will help to configure automated SQL server backup.

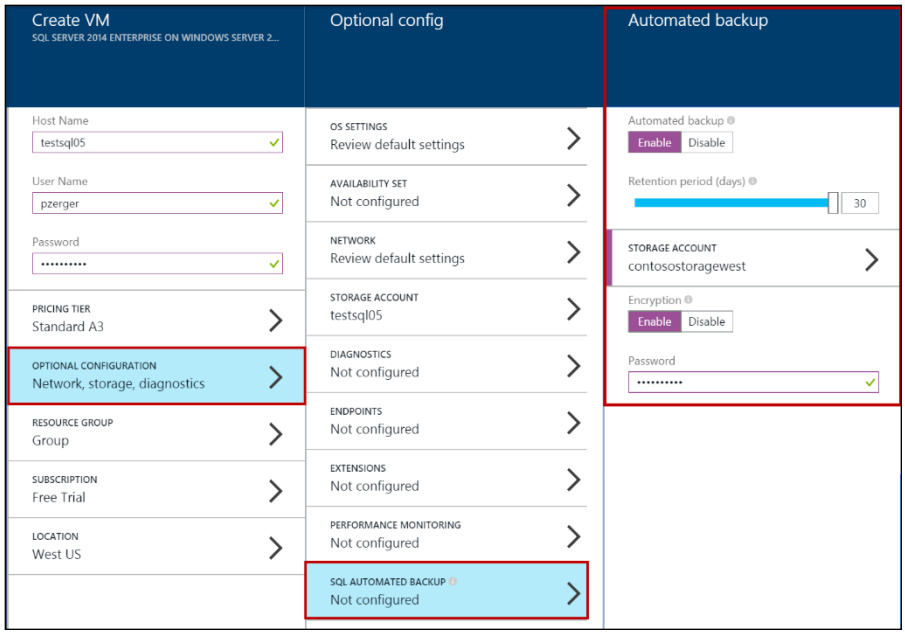


FIGURE 117. AUTOMATED SQL BACKUP OPTION IN AZURE CLASSIC VMS

To enable this feature on a VM that is already deployed, you will have to use Azure PowerShell to complete the configuration. A sample script is included below. Be sure update the values in brackets <> with values applicable to your environment.

```
$storageaccount = "<storageaccountname>"
$storageaccountkey = (Get-AzureStorageKey `
-StorageAccountName $storageaccount).Primary

$storagecontext = New-AzureStorageContext `
-StorageAccountName $storageaccount -StorageAccountKey $storageaccountkey

$autobackupconfig = New-AzureVMSqlServerAutoBackupConfig `
-StorageContext $storagecontext -Enable -RetentionPeriod 10

Get-AzureVM -ServiceName <vmservicename> -Name <vmname> | `
Set-AzureVMSqlServerExtension -AutoBackupSettings `
$autobackupconfig | Update-AzureVM
```

It could take several minutes to install and configure the SQL Server IaaS Agent. View the status of the VM in the Azure Portal. It should indicate that it is installing extensions. The extensions area should also report that the Microsoft.SqlServer.Management.SqlIaaSAgent is being enabled. In PowerShell, you can

test that the extension has completely installed and configured by using the following command:

```
(Get-AzureVM -ServiceName <vmservicename> | `
Get-AzureVMSqlServerExtension).AutoBackupSettings
```

To disable automatic backup, run the same script without the **-Enable** parameter to the **New-AzureVMSqlServerAutoBackupConfig**. As with installation, it can take several minutes to disable Automated Backup.

Azure Resource Manager Option

Follow below steps in your VMs are v2 ARM.

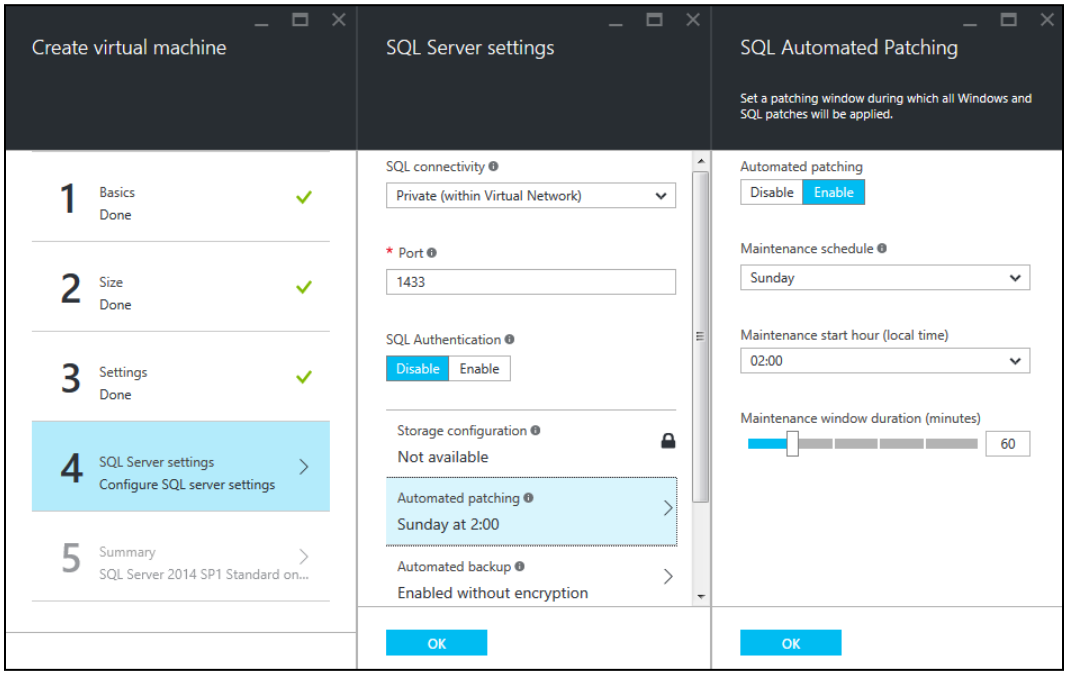


FIGURE 118. AUTOMATED SQL BACKUP OPTION IN V2 AZURE IAAS VMS

Like in previous ASM option you can use below PowerShell snippet to enable automated SQL backup for an existing ARM VM.

```
$vmname = "vmname"
$resourcegroupname = "resourcegroupname"
```

```
$password = "P@ssw0rd"
$encryptionpassword = $password | ConvertTo-SecureString -
AsPlainText -Force

$autobackupconfig = AzureRM.Compute\New-
AzureVMSqlServerAutoBackupConfig -Enable -RetentionPeriod 10 -
EnableEncryption -CertificatePassword $encryptionpassword -
ResourceGroupName $resourcegroupname

Set-AzureRmVMSqlServerExtension -AutoBackupSettings
$autobackupconfig -VMName $vmname -ResourceGroupName
$resourcegroupname
```

As it takes several minutes to configure the SQL Server IaaS Agent when you enable Automated Backup for the first time, the Azure portal might not show that Automated Backup is configured. Once the agent is installed and configured these changes will be reflected in the portal.

To disable automatic backup, run the same script without the **-Enable** parameter to the **AzureRM.Compute\New-AzureVMSqlServerAutoBackupConfig** command. One thing to keep in mind is that removing the SQL Server IaaS Agent does not remove the previously configured Automated Backup settings and therefore you need to disable Automated Backup before disabling or uninstalling the SQL Server IaaS Agent.

3rd Party Azure-Integrated Backup

A popular independent software vendor (ISV) that provides Azure-integrated backup solutions is Veeam Software. Their core enterprise backup solution is called Veeam Backup & Replication.

Veeam Backup & Replication is a solution specifically built for virtual environments. It operates at the virtualization layer and uses an image-based approach for VM backup. To retrieve VM data, no agent software needs to be installed inside the guest OS. Instead, Veeam Backup & Replication leverages VSS snapshot capabilities. When a new backup session starts, a VSS snapshot is taken to create a cohesive point-in-time copy of a VM including its configuration, OS, applications, associated data, system state and so on. Veeam Backup & Replication uses this point-in-time copy to retrieve VM data. Image-based backups can be used for different types of recovery, including Instant VM Recovery, full VM recovery, VM file recovery and file-level recovery.

The latest version of Veeam Backup and Replication (v9) supports the following operating systems, file systems, and applications.

- vSphere 4.1, 5.x, 6.0
- ESX(i) 4.1, ESXi 5.x, ESXi 6.0
- Hyper-V on Windows Server 2008 R2 SP1, 2012, 2012 R2, Microsoft Hyper-V Server 2008 R2 SP1, 2012, 2012 R2
- All operating systems supported by above Hyper-V and VMware editions
- Any application
- Any file system

You can protect both VMware and Hyper-V hosts directly. Having vCenter/vCloud Director or SCVMM deployed is optional.

When using Microsoft Azure for offsite backups with Veeam, there are three implementation options.

- Veeam Integration with StorSimple
- Veeam Cloud Connect for Service Providers
- Veeam Cloud Connect for Enterprise

Each option is described briefly in the sections that follow.

Veeam Integration with StorSimple

The StorSimple hybrid cloud storage solution adds dynamic storage tiering across SSD, SAS and Microsoft Azure to ensure data growth requirements are always met. StorSimple uses Azure as an extension of its on-premises storage arrays and automatically tiers data across on-premises storage and cloud storage.

StorSimple, make data protection and archiving to Azure easy and efficient. To Veeam, StorSimple looks like any another connected, on-premises data repository. However, StorSimple is more than just storage—it automatically manages the movement of data to and from Azure for efficient availability.

Veeam Backup & Replication™ provides agentless image-level backup to help meet stringent RPOs and RTOs while allowing more recovery options than you ever thought possible:

- Recovery of a failed VM in as little as two minutes
- Near-continuous data protection with built-in replication
- Fast, agentless item recovery and e-discovery for Microsoft Exchange, SharePoint and Active Directory, along with transaction-level recovery of SQL databases
- Automatic recoverability testing of every backup and every replica, every time

Veeam and StorSimple work in unison to mitigate the cost and management of data growth while providing secure backup and recovery. Veeam ensures that backups are initially stored on a traditional primary storage for short-term recovery, and depending on your availability needs, Veeam archives older versions of backups to StorSimple for long-term compliance. StorSimple will, in turn, ensure that backups are moved into Azure via cloud snapshot.

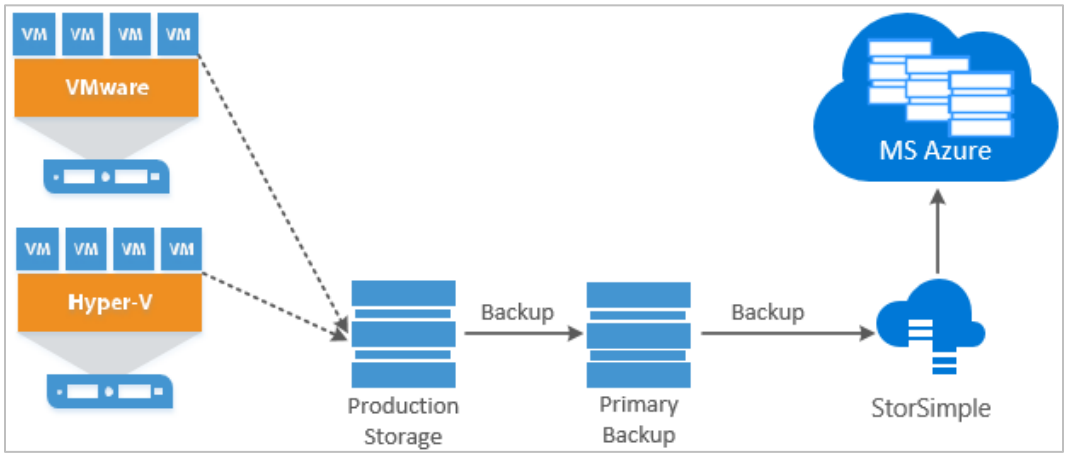


FIGURE 119. VEEAM BACKUP & REPLICATION COUPLED WITH STORSIMPLE

Veeam Cloud Connect for Service Providers

Veeam Cloud Connect™ for Service Providers extends the functionality of Veeam Backup & Replication, enabling service providers to provide storage and recovery of backup data in Microsoft Azure. The Cloud Connect solution architecture is very simple, enabling service providers to provide customers (tenants) a hosted backup repository in Microsoft Azure. The solution architecture is quite simple, consisting of the following roles:

- **Cloud Gateway** - The Cloud Gateway is a network service running on a Windows server that resides on the service provider side and acts as a communication

point in the cloud. It routes commands and traffic between the service provider, tenants, and the cloud repository.

- **WAN Accelerator (optional)** - WAN accelerators are optional components in the Veeam Cloud Connect infrastructure. Tenants may use WAN accelerators for Backup Copy jobs targeted at the cloud repository. WAN accelerators deployed in the cloud run the same services and perform the same role as WAN accelerators in an on-premises backup infrastructure. When configuring Veeam Backup Copy jobs, tenants can choose to exchange data over a direct channel or communicate with the cloud repository via a pair of WAN accelerators. To pass VM data via WAN accelerators, the service provider and tenants must each configure WAN accelerator, with the source WAN accelerator located on tenant side (in the tenant data center), the target WAN accelerator is configured on the service provider side.
- **Tenant Veeam Backup Server** - To connect to the cloud and use the cloud repository service provided by the service provider, tenants utilize Veeam backup servers deployed on their side.

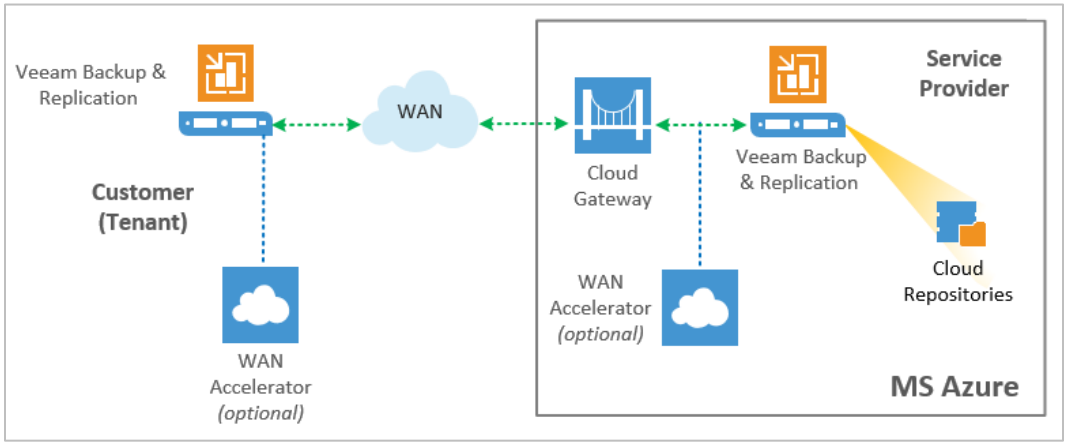


FIGURE 120. VEEAM CLOUD CONNECT – POC ARCHITECTURE

Expanding the capacity of a single VM setup is easy to do by leveraging the distributed model of Veeam Backup & Replication, along with the rapid resource provisioning capabilities in Azure. The diagram below illustrates a distributed model.

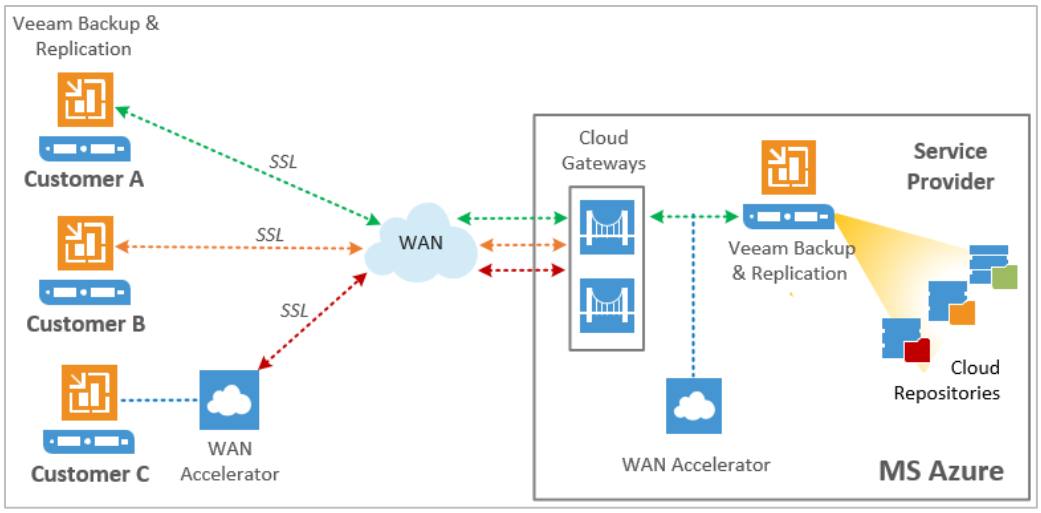


FIGURE 121. VEEAM CLOUD CONNECT – PRODUCTION ARCHITECTURE

For systems integrators trying to build a successful business in the Microsoft cloud, Veeam Cloud Connect offers the ability to create a recurring revenue stream.

As a Veeam Cloud Connect Service Provider, you deploy the appliance from the Azure Marketplace, where you will find the Veeam Cloud Connect VM offering. If your company is not yet enrolled in the Veeam Cloud Provider Program, you can click the link provided in the Azure Marketplace offering and receive a 30-day trial.

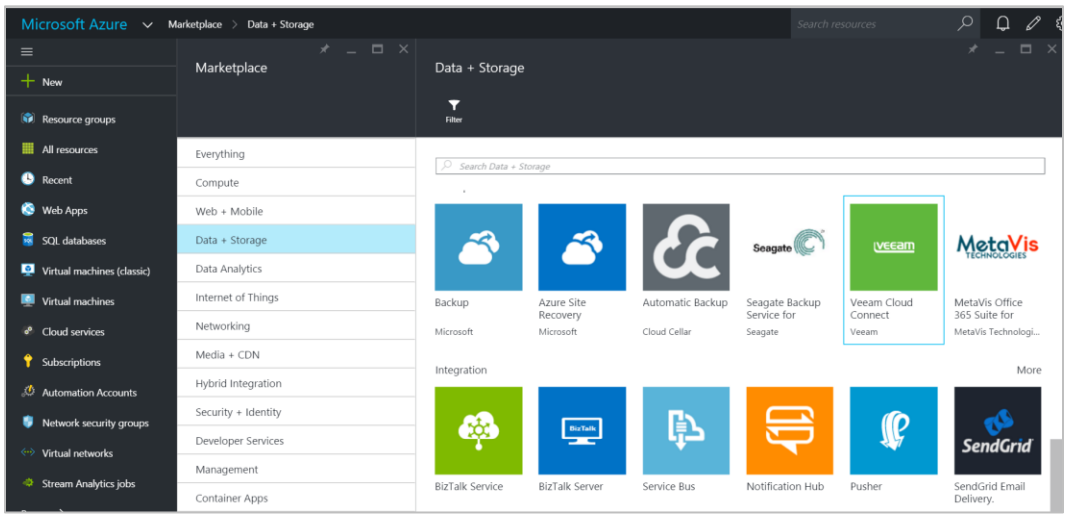


FIGURE 122. VEEAM CLOUD CONNECT IN THE AZURE MARKETPLACE

You will find multiple resources on the Veeam website to get familiarized yourself with the solution, including the following whitepapers, which cover everything from reference architecture to comprehensive hands-on deployment guidance.

Solution Brief: Veeam, StorSimple, and Microsoft Azure

https://www.veeam.com/wp-veeam-storsimple-microsoft-azure.html

Veeam Backup & Replication v8: Cloud Connect Reference Architecture

http://www.veeam.com/wp-cloud-connect-reference-architecture-veeam-backup-replication-v8.html

Veeam Cloud Connect: Manual configuration guide for Microsoft Azure

http://www.veeam.com/wp-build-services-business-veeam-microsoft-azure.html

Veeam Cloud Connect: Pre-configured VM deployment from the Microsoft Azure Marketplace

http://www.veeam.com/wp-build-services-business-veeam-microsoft-azure-marketplace.html

These resources and a 30-day trial license make getting up to speed on how Veeam can help you to design your cloud availability strategy a manageable task.

Veeam Cloud Connect for Enterprise

Veeam Cloud Connect™ for Enterprise, a new offering from Veeam, is a Cloud Connect option for customers who would prefer to manage their own hybrid disaster recovery strategy. You will find the Veeam Cloud for Enterprise offering in the Azure Marketplace as well. The VM is basically the same as the Service Provider edition and the difference is the license that enables the Enterprise edition.

Summary

Microsoft offers a number of workload protection options in Azure, enabling organizations to ensure their data is protected in the event of a disaster, even if they don't have a secondary data center. With the Azure Backup service, Microsoft has enabled

a comprehensive, cloud-based offsite backup solution. Azure backup vaults provide secure, encrypted backup targets for customers of all sizes. Regardless of the fact that an organization has a more complex backup and recovery configurations utilizing System Center Data Protection Manager, or whether they have a simpler, more basic need for offsite backup, Azure Backup enables offsite backup for everyone. The Instant File Recovery feature adds tremendous value in reducing the need to deploy the MARS agent to facilitate recovery of individual files.

With Azure Site Recovery, Microsoft has enabled cloud-based disaster recovery orchestration. Perhaps one of the simplest, most easy-to-use implementations of disaster recovery solutions ever, Azure Site Recovery enables effective disaster recovery for organizations of all sizes. In Azure Site Recovery, disaster recovery plans can easily be tested, ensuring business continuity in the event of an actual disaster.

With multiple protection options for Microsoft SQL Server, organizations have greater flexibility in planning and implementing high availability and disaster recovery strategies for one of the most common (and often, most critical) workloads in a variety of scenarios.

For partners and customers looking to leverage capabilities of Azure as part of a hybrid disaster recovery strategy for heterogeneous environments, Veeam offers options to suit a variety of needs.

Conclusion

We hope that you found the material in the book informative and helpful. If you have any feedback or suggestions, please feel free to e-mail me directly at ryan.irujo@gmail.com or contact us on twitter @insidemscloud.