Microsoft

Fundamentals of Azure

Second Edition

Microsoft Azure Essentials

Michael Collier
Robin Shahan

PUBLISHED BY

Microsoft Press
A division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

ISBN: 978-1-5093-0296-3

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Support at mspinput@microsoft.com. Please tell us what you think of this book at http://aka.ms/tellpress.

This book is provided "as-is" and expresses the author's views and opinions. The views, opinions and information expressed in this book, including URL and other Internet website references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

Acquisitions Editor: Devon Musgrave
Developmental Editor: Carol Dillingham
Editorial Production: Cohesion
Copyeditor: Ann Weaver
Cover: Twist Creative • Seattle

To my wife, Sonja, and sons, Aidan and Logan; I love you more than words can express. I could not have written this book without your immense support and patience.

—Michael S. Collier

I dedicate this book to the many people who helped make this the best book possible by reviewing, discussing, and sharing their technical wisdom. I especially want to mention Neil Mackenzie, who is always willing to share his encyclopedic knowledge of Azure with me, and whose tech reviews were incredibly helpful. I'd also like to mention Jennelle Crothers, without whom networking would be a complete mystery to me.

—Robin E. Shahan

Contents

Introduction

Microsoft Azure is Microsoft's cloud computing platform, providing a wide variety of services you can use without purchasing and provisioning your own hardware. Azure enables the rapid development of solutions and provides the resources to accomplish tasks that may not be feasible in an on-premises environment. Azure's compute, storage, network, and application services allow you to focus on building great solutions without the need to worry about how the physical infrastructure is assembled.

This book covers the fundamentals of Azure you need to start developing solutions right away. It concentrates on the features of the Azure platform that you are most likely to need to know rather than on every feature and service available on the platform. This book also provides several walkthroughs you can follow to learn how to create VMs and virtual networks, websites and storage accounts, and so on. In many cases, real-world tips are included to help you get the most out of your Azure experience.

In addition to its coverage of core Azure services, the book discusses common tools useful in creating and managing Azure-based solutions. The book wraps up by providing details on a few common business scenarios where Azure can provide compelling and valuable solutions, as well as a chapter providing overviews of some of the commonly used services not covered in the book.

Who should read this book

This book focuses on providing essential information about the key services of Azure for developers and IT professionals who are new to cloud computing. Detailed, step-by-step demonstrations are included to help the reader understand how to get started with each of the key services. This material is useful not only for those who have no prior experience with Azure, but also for those who need a refresher and those who may be familiar with one area but not others. Each chapter is standalone; there is no requirement that you perform the hands-on demonstrations from previous chapters to understand any particular chapter.

Assumptions

We expect that you have at least a minimal understanding of virtualized environments and virtual machines. There are no specific skills required overall for this book, but having some knowledge of the topic of each chapter will help you gain a deeper understanding. For example, the chapter on virtual networks will make more sense if you have some understanding of networking, and the chapter on databases will be more useful if you understand what a database is and why you might use one. Web development skills will provide a good background for understanding Azure Web Apps, and some understanding of identity will be helpful when studying the chapter on Active Directory.

This book might not be for you if…

This book might not be for you if you are looking for an in-depth developer or architecture-focused discussion on a wide range of Azure features, or if you are looking for details on other public or private cloud platforms.

Organization of this book

This book explores six foundational features of the Microsoft Azure platform, along with insights on getting started with Azure, management tools, and common business scenarios. This book also includes a chapter with overviews of some of the more commonly used services, such as HDInsight (Azure's Hadoop service) and Service Bus, but there are many services in the Azure platform that are not in the scope of this book, such as Azure Batch, Data Lake Analytics, and Azure DNS, just to mention a few. To learn about all of the services available in the Azure platform, start your journey at http://azure.microsoft.com. Also, there is a web application that shows the many services of Azure and allows you to drill down to learn move. See http://aka.ms/azposterapp.

The topics explored in this book include:

- **Getting started with Azure**: Understand what cloud computing is, learn about Azure Resource Manager and Role-Based Access Control, visit the management portals, learn about billing, find out how you can contribute to the Azure documentation and code samples.
- **Azure App Service and Web Apps**: Learn about the Azure App Service, consisting of Web Apps, Logic Apps, Mobile Apps, API Apps, and Function Apps. We will focus on Web Apps and how they work with the App Service and App Service plans, covering the topic from deployment to monitoring and scaling.
- **Virtual Machines**: Explore the basic features of Azure Virtual Machines, including how to create, configure, and manage them.
- **Storage**: Read about the basics of Azure Storage, including blobs, tables, queues, and file shares, as well as some of the options available such as Premium Storage and Cool Storage.
- **Virtual Networks**: Learn the basics of virtual networks, including how to create one, and why a virtual network might be necessary. This also covers site-to-site and point-to-site networking, as well as ExpressRoute.
- **Databases**: Explore two relational database options available in Azure: Azure SQL Database and SQL Server in Azure Virtual Machines.
- **Azure Active Directory**: Explore basic features of Azure AD, including creating a directory, users and groups, and using the application gallery.
- **Management Tools**: Explore three common tools for working with Azure: Visual Studio 2015 and the Azure SDK, Azure PowerShell cmdlets, and the Cross-Platform Command-Line Interface
- **Additional Azure services**: Get an overview about Azure services not covered in the book that may be fundamental to you now or in the future, such as Azure Service Fabric and Azure Container Service.
- **Business Scenarios**: Explore five common scenarios for utilizing Azure features: development and test, hybrid, application and infrastructure modernization, and Azure Mobile Apps, and Machine Learning.

Conventions and features in this book

This book presents information using conventions designed to make the information readable and easy to follow:

- To create specific Azure resources, follow the numbered steps listing each action you must take to complete the exercise.
- There are currently two management portals for Azure: the Azure portal at https://portal.azure.com and the Azure classic portal at http://manage.windowsazure.com. In most cases, the book uses the Azure portal, but the Azure classic portal may be used for those features that have not been migrated to the newer portal yet, such as Azure Active Directory.
- Boxed elements with labels such as “Note” or "See Also" provide additional information.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, “Press Alt+Tab” means that you hold down the Alt key while you press Tab.
- A right angle bracket between two or more menu items (e.g., File Browse > Virtual Machines) means that you should select the first menu or menu item, then the next, and so on.

System requirements

For many of the examples in this book, you need only Internet access and a browser (Internet Explorer 10 or higher) to access the Azure portals.

Chapter 2, "Azure App Service and Web Apps," and Chapter 4, "Azure Storage," use Visual Studio to show concepts used in developing applications for Azure. For these examples, you will need Visual Studio. The system requirements are:

- Windows 7 Service Pack 1, Windows 8, Windows 8.1, Windows 10, Windows Server 2008 R2 SP1, Windows Server 2012, or Windows Server 2012 R2
- Computer that has a 1.6GHz or faster processor (2GHz recommended)
- 1 GB (32 Bit) or 2 GB (64 Bit) RAM (Add 512 MB if running in a virtual machine)
- 4 GB of available hard disk space
- 5400 RPM hard disk drive
- DirectX 9 capable video card running at 1024 x 768 or higher-resolution display
- DVD-ROM drive (if installing Visual Studio from DVD)
- Internet connection

After installing Visual Studio, you must also install the Azure Tools and SDK for the language of your choice from https://azure.microsoft.com/tools/.

The system requirements for the Azure SDK that are not included in the Visual Studio system requirements are as follows:

- IIS7 with ASP.NET and WCF HTTP Activation, Static Content, IIS Management Console, and HTTP Redirection
- Web Deployment Tools 2.1 or up

- Internet Explorer 10 or higher

Depending on your Windows configuration, you might require Local Administrator rights to install or configure Visual Studio 2015.

Downloads

Some of the chapters in this book include exercises that let you interactively try out new material learned in the main text. Chapter 4, "Azure Storage," has PowerShell scripts; Chapter 5, "Virtual Networks," has PowerShell scripts and a Resource Manager template. These can be downloaded from the following page:

https://aka.ms/FundAzure2e/downloads

Follow the instructions on the target page to download the code sample files.

> **Note** To use the PowerShell scripts, you need to have Azure PowerShell installed. This article explains how to install and configure Azure PowerShell: https://azure.microsoft.com/documentation/articles/powershell-install-configure/.

Using the code samples

The code samples are stored within a unique .ZIP file, "FundAzure2E.ZIP," which can be downloaded to your computer and unzipped so that you can use them with the exercises in this book.

- Samples for Chapter 4, "Azure Storage," are in the Chapter4_PowerShellScripts folder in the ZIP file. This includes the PowerShell scripts for both Blob storage and File Storage. You can open, edit, and run these using the PowerShell ISE.

- Samples for Chapter 5, "Azure Virtual Networks," are in the folder "Chapter5_PowerShellScripts_And_Templates." This includes both the Resource Manager templates used to create and modify a virtual network and the PowerShell script used to create a point-to-site VPN Network. To use the Resource Manager templates, please follow the instructions provided in the chapter. You can open, edit, and run the PowerShell script with PowerShell ISE.

Acknowledgments

The Azure community is made up of many people bound together by this one technology. We are honored to be members of this community, and we thank you for your help and support. We would like to especially thank Neil Mackenzie, Mike Wood, and Mike Martin, as well as Byron Tardif, Ashwin Kamath, and Rajesh Ramabathiran from the Azure App Service team for their detailed technical reviews and feedback. All of them provided additional insights that greatly enhanced the overall quality and value of this book.

Special thanks to the team at Microsoft Press for their unwavering support and guidance on this journey. It was a pleasure to work with our editors, Devon Musgrave and Carol Dillingham. Thanks to Chris Norton for helping us through the final edit cycles.

Most importantly, we are profoundly grateful to our families and friends for their love, encouragement, and patience. Many nights and weekends were sacrificed in the writing of this book.

Errata, updates, & support

We've made every effort to ensure the accuracy of this book. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

http://aka.ms/FundAzure2e/errata

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at mspinput@microsoft.com.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to http://support.microsoft.com.

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

http://aka.ms/mspressfree

Check back often to see what is new!

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: http://twitter.com/MicrosoftPress

CHAPTER 1

Getting started with Microsoft Azure

The purpose of this ebook is to help you understand the fundamentals of Microsoft Azure so you can hit the ground running when you start using it.

With an Azure account, you can work through the demos in this book and use them as hands-on labs. If you don't have an Azure account, you can sign up for a free trial at azure.microsoft.com. If you have an MSDN subscription, you can activate the included Azure benefits and use the associated monthly credit. You can also check out Purchase Options at https://azure.microsoft.com/pricing/purchase-options/ and Member Offers at https://azure.microsoft.com/pricing/member-offers/ (for members of MSDN, the Microsoft Partner Network, BizSpark, and other Microsoft programs).

What is Azure?

The following will give an overview of Azure, which is Microsoft's cloud computing platform.

Overview of cloud computing

Cloud computing provides a modern alternative to the traditional on-premises datacenter. A public cloud vendor is completely responsible for hardware purchase and maintenance and provides a wide variety of platform services that you can use. You lease whatever hardware and software services you require on an as-needed basis, thereby converting what had been a capital expense for hardware purchase into an operational expense. It also allows you to lease access to hardware and software resources that would be too expensive to purchase. Although you are limited to the hardware provided by the cloud vendor, you only have to pay for it when you use it.

Cloud environments provide an online portal experience, making it easy for users to manage compute, storage, network, and application resources. For example, in the Azure portal, a user can create a virtual machine (VM) configuration specifying the following: the VM size (with regard to CPU, RAM, and local disks), the operating system, any predeployed software, the network configuration, and the location of the VM. The user then can deploy the VM based on that configuration and within a few minutes access the deployed VM. This quick deployment compares favorably with the previous mechanism for deploying a physical machine, which could take weeks just for the procurement cycle.

In addition to the public cloud just described, there are private and hybrid clouds. In a private cloud, you create a cloud environment in your own datacenter and provide self-service access to compute resources to users in your organization. This offers a simulation of a public cloud to your users, but you remain completely responsible for the purchase and maintenance of the hardware and software services you provide. A hybrid cloud integrates public and private clouds, allowing you to host workloads in the most appropriate location. For example, you could host a high-scale website in the public cloud and link it to a highly secure database hosted in your private cloud (or on-premises datacenter).

Microsoft provides support for public, private, and hybrid clouds. Microsoft Azure, the focus of this book, is a public cloud. Microsoft Azure Stack is an add-on to Windows Server 2016 that allows you to deploy many core Azure services in your own datacenter and provides a self-service portal experience to your users. You can integrate these into a hybrid cloud through the use of a virtual private network.

Comparison of on-premises versus Azure

With an on-premises infrastructure, you have complete control over the hardware and software that you deploy. Historically, this has led to hardware procurement decisions focused on scaling up; that is, purchasing a server with more cores to satisfy a performance need. With Azure, you can deploy only the hardware provided by Microsoft. This leads to a focus on scale-out through the deployment of additional compute nodes to satisfy a performance need. Although this has consequences for the design of an appropriate software architecture, there is now ample proof that the scale-out of commodity hardware is significantly more cost-effective than scale-up through expensive hardware.

Microsoft has deployed Azure datacenters in over 22 regions around the globe from Melbourne to Amsterdam and Sao Paulo to Singapore. Additionally, Microsoft has an arrangement with 21Vianet, making Azure available in two regions in China. Microsoft has also announced the deployment of Azure to another eight regions. Only the largest global enterprises are able to deploy datacenters in this manner, so using Azure makes it easy for enterprises of any size to deploy their services close to their customers, wherever they are in the world. And you can do that without ever leaving your office.

For startups, Azure allows you to start with very low cost and scale rapidly as you gain customers. You would not face a large up-front capital investment to create a new VM—or even several new VMs. The use of cloud computing fits well with the scale fast, fail fast model of startup growth.

Azure provides the flexibility to set up development and test configurations quickly. These deployments can be scripted, giving you the ability to spin up a development or test environment, do the testing, and spin it back down. This keeps the cost very low, and maintenance is almost nonexistent.

Another advantage of Azure is that you can try new versions of software without having to upgrade on-premises equipment. For example, if you want to see the ramifications of running your application against Microsoft SQL Server 2016 instead of Microsoft SQL Server 2014, you can create a SQL Server 2016 instance and run a copy of your services against the new database, all without having to allocate hardware and run wires. Or you can run on a VM with Microsoft Windows Server 2012 R2 instead of Microsoft Windows Server 2008 R2.

Cloud offering

Cloud computing usually is classified in three categories: SaaS, PaaS, and IaaS. However, as the cloud matures, the distinction among these is being eroded.

SaaS: Software as a service

SaaS is software that is centrally hosted and managed for the end customer. It usually is based on a multitenant architecture—a single version of the application is used for all customers. It can be scaled out to multiple instances to ensure the best performance in all locations. SaaS software typically is licensed through a monthly or annual subscription.

Microsoft Office 365 is a prototypical model of a SaaS offering. Subscribers pay a monthly or annual subscription fee, and they get Exchange as a Service (online and/or desktop Outlook), Storage as a Service (OneDrive), and the rest of the Microsoft Office Suite (online, the desktop version, or both). Subscribers are always provided the most recent version. This essentially allows you to have a Microsoft Exchange server without having to purchase a server and install and support Exchange—the Exchange server is managed for you, including software patches and updates. Compared to installing and upgrading Office every year, this is much less expensive and requires much less effort to keep updated.

Other examples of SaaS include Dropbox, WordPress, and Amazon Kindle.

PaaS: Platform as a service

With PaaS, you deploy your application into an application-hosting environment provided by the cloud service vendor. The developer provides the application, and the PaaS vendor provides the ability to deploy and run it. This frees developers from infrastructure management, allowing them to focus strictly on development.

Azure provides several PaaS compute offerings, including the Web Apps feature in Azure App Service and Azure Cloud Services (web and worker roles). In either case, developers have multiple ways to deploy their application without knowing anything about the nuts and bolts supporting it. Developers don’t have to create VMs, use Remote Desktop Protocol (RDP) to log into each one, and install the application. They just hit a button (or pretty close to it), and the tools provided by Microsoft provision the VMs and then deploy and install the application on them.

IaaS: Infrastructure as a service

An IaaS cloud vendor runs and manages server farms running virtualization software, enabling you to create VMs that run on the vendor’s infrastructure. Depending on the vendor, you can create a VM running Windows or Linux and install anything you want on it. Azure provides the ability to set up virtual networks, load balancers, and storage and to use many other services that run on its infrastructure. You don't have control over the hardware or virtualization software, but you do have control over almost everything else. In fact, unlike PaaS, you are completely responsible for it.

Azure Virtual Machines, the Azure IaaS offering, is a popular choice when migrating services to Azure because it enables the "lift and shift" model for migration. You can configure a VM similar to the infrastructure currently running your services in your datacenter and migrate your software to the new VM. You might need to make tweaks, such as URLs to other services or storage, but many applications can be migrated in this manner.

Azure VM Scale Sets (VMSS) is built on top of Azure Virtual Machines and provides an easy way to deploy clusters of identical VMs. VMSS also supports autoscaling so that new VMs can be deployed automatically when required. This makes VMSS an ideal platform to host higher-level microservice compute clusters such as for Azure Service Fabric and the Azure Container Service.

Azure services

Azure includes many services in its cloud computing platform. Let's talk about a few of them.

- **Compute services** This includes the Azure Virtual Machines—both Linux and Windows, Cloud Services, App Services (Web Apps, Mobile Apps, Logic Apps, API Apps, and Function Apps), Batch (for large-scale parallel and batch compute jobs), RemoteApp, Service Fabric, and the Azure Container Service.
- **Data services** This includes Microsoft Azure Storage (comprised of the Blob, Queue, Table, and Azure Files services), Azure SQL Database, DocumentDB, StorSimple, and the Redis Cache.
- **Application services** This includes services that you can use to help build and operate your applications, such as Azure Active Directory (Azure AD), Service Bus for connecting distributed systems, HDInsight for processing big data, Azure Scheduler, and Azure Media Services.
- **Network services** This includes Azure features such as Virtual Networks, ExpressRoute, Azure DNS, Azure Traffic Manager, and the Azure Content Delivery Network.

When migrating an application, it is worthwhile to have some understanding of the different services available in Azure because you might be able to use them to simplify the migration of your application and improve its robustness. It is impossible for us to cover everything in this book, but there are some services we felt you should know about. Chapter 9, "Additional Azure services," provides a list of these services and a brief description of each of them.

The new world: Azure Resource Manager

The Azure Resource Manager is the new methodology for deploying resources.

What is it?

Since it went into public preview, the Azure Service Management (ASM) deployment model has been used to deploy services. In the Azure portal, services managed with ASM are referred to as *classic*. In 2015, Microsoft introduced the Resource Manager deployment model as a modern, more functional replacement for ASM. The Resource Manager deployment model is recommended for all new Azure workloads.

These deployment models are often referred to as *control planes* because they are used to control services, not just to deploy them. This is different from a data plane, which manages the data used by a service.

Typically, your running Azure infrastructure will contain many resources, but some of the resources will be related to one another in some way, such as all being the component services required to run a web application. For example, you might have two VMs running the web application, using a database to store data, and residing in the same virtual network. With Resource Manager, you deploy these assets into the same resource group and manage and monitor them together. You can deploy, update, or delete all of the resources in a resource group in one operation.

In this example, the resource group would contain the following:

- VM1
- VM2
- Virtual network
- Storage account
- Azure SQL Database

You can also create a template that precisely defines all the Resource Manager resources in a deployment. You can then deploy this Resource Manager template into a resource group as a single control-plane operation, with Resource Manager in Azure ensuring that resources are deployed correctly. After deployment, Resource Manager provides security, auditing, and tagging features to help you manage your resources.

Why use Resource Manager?

There are several advantages to using Resource Manager. The deployment is faster because resources can be deployed in parallel rather than sequentially as they are in ASM. The Resource Manager model enables each service to have its own service provider, and they can update it as needed independently of the other services. Azure Storage has its own service provider, VMs have their own service provider, and so on. With the ASM model, all services had to be updated at one time, so if one service was finished and the rest were not, the one that was ready had to wait on the others before it could be released. Here are some of the other major advantages to the Resource Manager model:

- Deployment using templates
    - You can create a reusable (JSON) template that can be used to deploy all of the resources for a specific solution in one fell swoop. You no longer have to create a VM in the portal, wait for it to finish, then create the next VM, and so on.
    - You can use the template to redeploy the same resources repeatedly. For example, you may set up the resources in a test environment and find that it doesn't fit your needs. You can delete the resource group, which removes all of the resources for you, then tweak your template and try again. If you only want to make changes to the resources deployed, you can just change the template and deploy it again, and Resource Manager will change the resources to conform to the new template.
    - You can take that template and easily re-create multiple versions of your infrastructure, such as staging and production. You can parameterize fields such as the VM name, network name, storage account name, etc., and load the template repeatedly, using different parameters.

- Resource Manager can identify dependencies in a template but allows you to specify additional dependencies if necessary. For example, you wouldn't want to deploy a virtual machine before creating the storage account for the VHD files that are used for the OS and data disks.
- Security
    - You can use the new Role-Based Access Control (RBAC) to control access to the resources in the group. For example, you can assign the Owner role to a user, giving that user full administrative privileges to those resources in the group but not to other resources in the subscription. Other roles include Reader (you can read anything except secrets) and Contributor (you can do most anything except add or revoke access).
- Billing
    - To help organize all of the resources in a subscription for billing purposes, you can assign tags to each resource and then retrieve all of the billing information for a specific tag.

      For example, if one department owns a web application and several related components, you can assign the same tag to all of those resources. Then, you can retrieve the billing for that department by retrieving the billing for that tag.

Note If you apply a tag to a resource group, the resources in the group do not inherit that tag. You have to apply the tag to each individual resource.

Maximize the benefits of using Resource Manager

Microsoft has several suggestions to help you maximize the use of the Resource Manager model when working with your applications and components.

- Use templates rather than using scripting like PowerShell or the Azure Command-Line Interface (CLI). Using a template allows resources to be deployed in parallel, making it much faster than using a script executed sequentially.
- Automate as much as possible by leveraging templates. You can include configurations for various extensions like PowerShell DSC and Web Deploy. This way, you don't need any manual steps to create and configure the resources.
- Use PowerShell or the Azure CLI to manage the resources, such as to start or stop a virtual machine or application.
- Put resources with the same lifecycle in the same resource group. In our example above, what if the database is used by multiple applications? If that's true, or if the database is going to live on even after the application is retired or removed, you don't want to re-create the database every time you redeploy the application and its components. In that case, put the database in its own resource group.

Resource group tips

You can decide how to allocate your resources to resource groups based on what makes sense for you and your organization. A resource group is a logical container to hold related resources for an application or group of applications. These tips should be considered when making decisions about your resource group:

- As noted before, all of the resources in a group should have the same lifecycle.
- A resource can only be assigned to one group at a time.

- A resource can be added to or removed from a resource group at any time. Note that every resource must belong to a resource group, so if you remove it from one group, you have to add it to another.
- Most types of resource can be moved to a different resource group at any time.
- The resources in a resource group can be in different regions.
- You can use a resource group to control access for the resources therein.

Tips for using Resource Manager templates

Resource Manager templates define the deployment and configuration of your application. They are used to deploy an application and all of its component resources repeatedly.

You can divide the deployments in a set of templates and create a master template that links in all of the required templates.

Templates can be modified and redeployed with updates. For example, you can add a new resource or update configuration information about a resource in a template. When deployed again, Resource Manager will create any new resources it finds and perform updates for any that have been changed. You will see this in Chapter 5, “Azure Virtual Networks,” where you deploy a template defining a VNet with two subnets. Then, you add a third subnet and redeploy the template, and you can see the third subnet appear in the Azure portal.

Templates can be parameterized to allow you more flexibility in deployment. This is what allows you to use the same template repeatedly but with different values, such as VM name, virtual network name, storage account name, region, and so on.

You can export the current state of the resources in a resource group to a template. This can then be used as a pattern for other deployments, or it can be edited and redeployed to make changes and additions to the current resource group’s resources.

Here is an example of a JSON template. Deploying this template will create a storage account in West US called mystorage. This is parameterized; you can include a parameter file that provides the values for newStorageAccountName and location. Otherwise, it will use the defaults.

```
{
   "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
   "contentVersion": "1.0.0.0",
   "parameters": {
    "newStorageAccountName": {
      "type": "string",
      "defaultValue": "mystorage",
      "metadata": {
        "description": "Unique DNS Name for the Storage Account where the Virtual Machine's disks
will be placed."
      }
    },

    "location": {
      "type": "string",
      "defaultValue": "West US",
      "allowedValues": [
        "West US",
        "East US"
      ],
      "metadata": {
        "description": "Restricts choices to where premium storage is located in the US."
      }
    }
   },

   "resources": [
     {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[parameters('newStorageAccountName')]",
      "apiVersion": "2015-06-15",
      "location": "[parameters('location')]",
      "properties": {
        "accountType": "Standard_LRS"
      }
     }
  ]
}
```

The classic deployment model

Let’s talk a bit about what came before Resource Manager. These resources are now referred to as *classic*. For example, you can have storage accounts, virtual machines, and virtual networks that use the classic deployment model. The classic and Resource Manager models are not compatible with each other. The classic resources cannot be seen by the Resource Manager resources, and vice versa. For example, the PaaS Cloud Services feature of Azure is a classic feature, so you can only use it with storage accounts that are classic storage accounts. The exception to that rule is that you can use classic storage accounts to host Resource Manager VMs. This will make it easier to migrate your VMs from the classic deployment model to the Resource Manager deployment model.

Note that this means you may log into the classic Azure portal and see classic resources but not see Resource Manager resources, and vice versa.

Note There are two versions of the portal. The production portal is the Azure portal at https://portal.azure.com. Most features have been moved to the Azure portal, with some exceptions such as Azure Active Directory (Azure AD). The previous portal is called the classic Azure portal (https://manage.windowsazure.com), and it can still be used to manage Azure AD and to configure and scale classic resources such as Cloud Services.

You can migrate your assets from the classic to the Resource Manager deployment model.

- For storage accounts, you can use AzCopy to copy blobs, files, and tables to a new Resource Manager storage account. Note that tables must be exported from the classic account and then imported into the Resource Manager account.
- For virtual machines, you can shut them down and copy their VHD file to a new Resource Manager storage account and then use the VHD file to re-create the VM.
- For virtual networks, you can re-create them as Resource Manager VNets.
- There is also a migration service that is in public preview. Microsoft recommends using this only for nonproduction workloads at this time. For more information, check out this article: https://azure.microsoft.com/documentation/articles/virtual-machines-windows-migration-classic-resource-manager/

PowerShell changes for the Resource Manager and classic deployment models

Chapter 8, "Management tools," talks about some of the tools available to use with Azure, including the Azure PowerShell cmdlets and the Azure CLI.

One of the other changes made when the Azure team created the Resource Manager model was to create PowerShell cmdlets that work just for the Resource Manager model. They did this by appending "Rm" to "Azure" in the name of the cmdlets. For example, to create a classic storage account, you would use the *New-AzureStorageAccount* cmdlet. To create a Resource Manager storage account, you would use the *New-AzureRmStorageAccount* cmdlet.

Microsoft did this so you could easily tell which kind of resource you were creating. Also, this ensures that scripts that are currently being used will continue to work. Each time you deploy a Resource Manager resource, you have to specify the resource group into which it should be placed. Also, some of the cmdlets for Resource Manager (such as creating a VM) have more details than their counterparts in the classic model.

One last note: for storage accounts, the only PowerShell cmdlets impacted are on the control plane, such as those for creating a storage account, listing storage accounts, removing a storage account, and so on. All of the PowerShell cmdlets used to access the actual objects in storage—blobs, tables, queues, and files—remain unchanged. So once you are pointed to the right storage account, you're good to go.

Role-Based Access Control

In this section, we'll take a look at Role-Based Access Control (RBAC) to understand how you can use it to manage the security for your Resource Manager resources.

What is it?

In addition to the Resource Manager deployment model that allows you to group and manage your related resources, Microsoft introduced RBAC, providing fine-grained control over the operations and scope with which a user can perform a control-plant action. The previous methodology (classic) only allows you to grant either full administrative privileges to everything in a subscription or no access at all.

With Resource Manager, you can grant permissions at a specified scope: subscription, resource group, or resource. This means you can deploy a set of resources into a resource group and then grant permissions to one or more specific users, groups, or service principal. Those users will only have the permissions granted to those resources in that resource group. This access does not allow them to modify resources in other resource groups. You can also give a user permission to manage a single VM, and that's all that user will be able to access and administer.

In addition to users, Azure RBAC also supports service principals that formally are identities representing applications, but informally are used by RBAC to allow automated processes to manage Resource Manager resources. To grant access, you assign a role to the user, group, or service principal. There are many predefined roles, and you can also define your own custom roles.

Roles

Each role has a list of Actions and Not Actions. The Actions are allowed, and the Not Actions are excluded. See https://azure.microsoft.com/documentation/articles/role-based-access-built-in-roles/ for the full list of roles and their Actions and Not Actions.

For example, there is a role called Contributor. With this role, a user can manage everything except access. This role has the following Actions and Not Actions:

- Actions: * → Can create and manage resources of all types
- Not Action: Microsoft.Authorization/*/Write → Can't create roles or assign roles
- Not Action: Microsoft.Authorization/*/Delete → Can't delete roles or role assignments

Let's take a look at some of the most common roles.

- **Owner** A user with this role can manage everything, including access. This role has no Not Actions. This is synonymous with Co-Administrator in the classic deployment model.
- **Reader** A user with this role can read resources of all types (except secrets) but can't make changes. This role will allow someone to look at the properties of a storage account, but it won't let that person retrieve the access keys.
- **SQL DB Contributor** A user with this role can manage SQL databases but not their security-related policies.
- **SQL Security Manager** A user with this role can manage the security-related policies of SQL Servers and databases.
- **Storage Account Contributor** A user with this role can manage storage accounts but cannot manage access to the storage accounts. This means the user with this role can't assign any roles to any users for the storage account. Note that the user with this role *can* retrieve the access keys for the storage account, which means they have full access to the data in the storage account.
- **Virtual Machine Contributor** A user with this role can manage virtual machines but can't manage the VNet to which they are connected or the storage account where the VHD file resides. Note that this role *does* include access to the storage account keys, which is needed to create the container for the VHD files as well as the VHD files themselves.

These are only a few of the many roles that can be assigned to a user, a group of users, or an application.

Custom roles

If none of the built-in roles and no combination of the built-in roles provides exactly what you need, you can create a custom role. You can do this using PowerShell, the Azure CLI, or the REST APIs. Once you create a custom role, you can assign it to a user, group, or application for a subscription, resource group, or resource. Custom roles are stored in the Azure AD and can be shared across all subscriptions that use the same Active Directory.

For example, you could create a custom role for monitoring and restarting virtual machines. Here are the Actions you would assign to that role:

- Microsoft.Storage/*/read
- Microsoft.Network/*/read
- Microsoft.Compute/*/read
- Microsoft.Compute/virtualMachines;/start/action
- Microsoft.Compute/virtualMachines/restart/action
- Microsoft.Authorization/*/read
- Microsoft.Resources/subscriptions/resourceGroups/read
- Microsoft.Insights/alertRules/*
- Microsoft.Insights/diagnosticSettings/*
- Microsoft.Support/*

Note that as requested, this role can only start and restart virtual machines. It can't create them or delete them.

A convenient way to create a custom role is to download the definition of an existing role and use that as a starting point. When you create a custom role, you also need to specify in which subscriptions it can be used—at least one must be specified.

In the next section, we'll see how to assign roles to users for a resource group and how to give full administrative privileges for a subscription to a user.

The Azure portal

An online management portal provides the easiest way to manage the resources you deploy into Azure. You can use this to create virtual networks, set up Web Apps, create VMs, define storage accounts, and so on, as listed in the previous section.

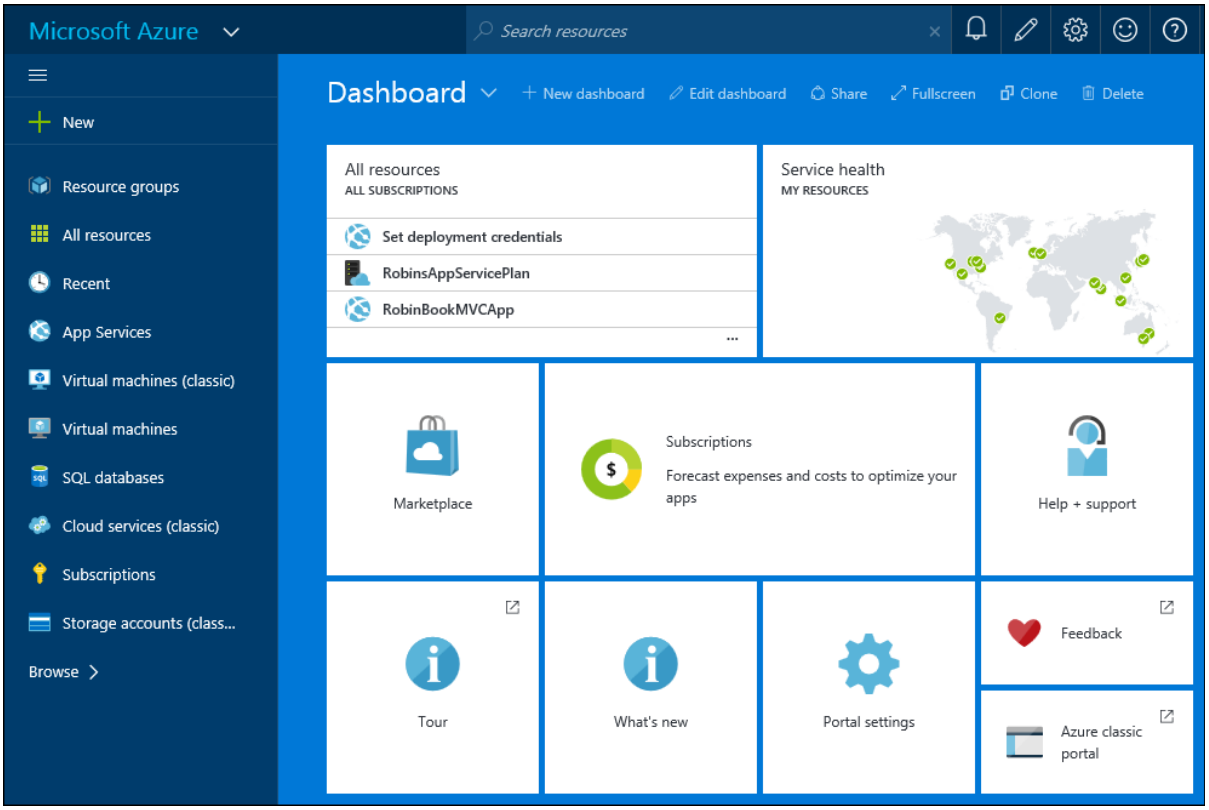
As noted earlier in this chapter, there are currently two versions of the portal. The production portal is the Azure portal at https://portal.azure.com. Most features have been moved to the Azure portal, with some exceptions such as Azure AD. The previous portal is called the classic Azure portal (https://manage.windowsazure.com), and it can still be used to manage Azure AD and to configure and scale classic resources such as Cloud Services.

In most cases, you will be using the Azure portal, so that's what we're going to focus on in this book. All of the resources that use the Resource Manager deployment model can only be accessed in the Azure portal.

Let's take a look at the Azure portal and how you navigate through it.

Dashboard and hub

The Azure portal is located at https://portal.azure.com. When you open this the first time, it will look similar to Figure 1-1.



Forecast expenses and costs to optimize your apps

Figure 1-1 Azure portal.

This is called your Dashboard. The column on the left is called a hub; it shows you a core set of options such as Resource Groups, All Resources, and Recent. The other items on this hub are resources you have selected and/or used before. For example, I have recently created some App Services and VMs. You can click any of these, and it will show the resources you have for that type. For example, if you click SQL Databases, it will show a list of your SQL Databases.

You can customize the list of resources that show up in that left hub. If you click Browse, you will see a selection screen showing all of the options, and you can select which ones you want to appear, as displayed in Figure 1-2.

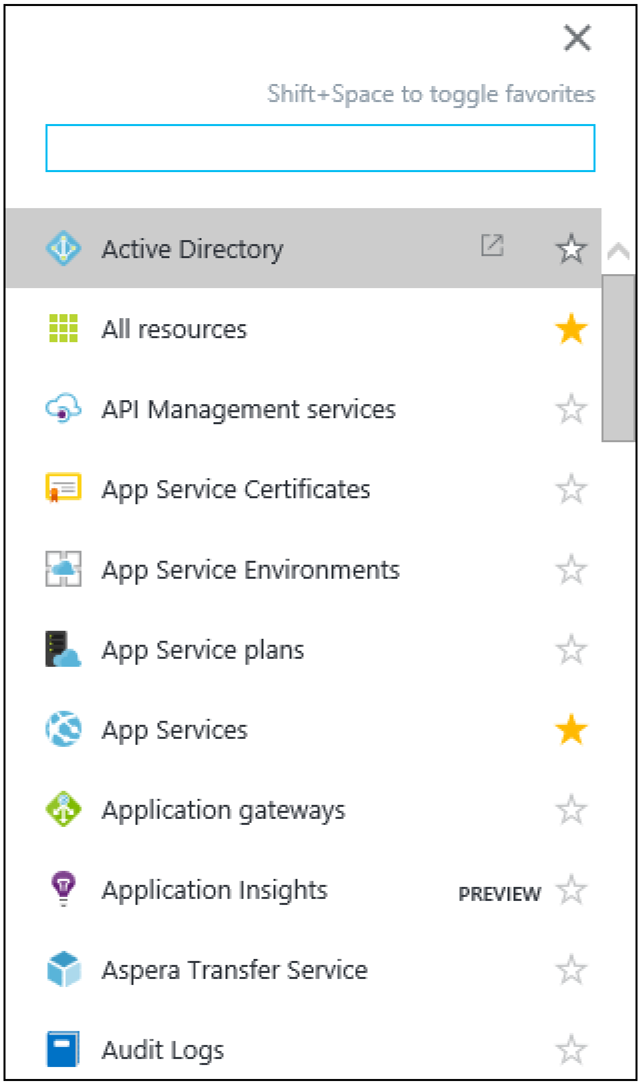


Figure 1-2 Configure default hub in the Azure portal.

The area on the right with the tiles is called your Dashboard. You can customize this by adding tiles, removing tiles, resizing tiles, and so on by selecting Edit Dashboard, as shown in Figure 1-3.

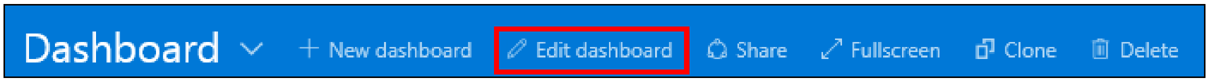


Figure 1-3 How to edit the Dashboard in the Azure portal.

As you create resources, you can choose to pin them to the Dashboard, and it will add them to this section.

There are a couple of default tiles on the Dashboard that are of interest.

- **All Resources** Clicking this will bring up a list of all of your resources.
- **Service Health** This shows the health of the regions around the world. If you click this, it will show a list of the regions, and you can select one to get more detailed information.
- **Marketplace** This will take you directly to the Marketplace blade where you can search for and add resources.

- **Subscriptions** This shows the subscriptions that can be managed by the account you are using. You can select a subscription and see the billing information for the current month. If you have a starting credit, this will show the amount of credit left. Accounts having starting credit include MSDN accounts and BizSpark accounts.
- **Help + Support** This takes you to the blade where you can submit a new support request and manage the requests you have already put in. It also provides links to the MSDN forums and StackOverflow where you can post questions.

Now, let's look at the icons in the upper-right corner of the Azure portal, as shown in Figure 1-4.

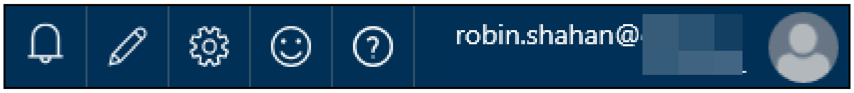


Figure 1-4 Notifications, settings, etc. in the Azure portal.

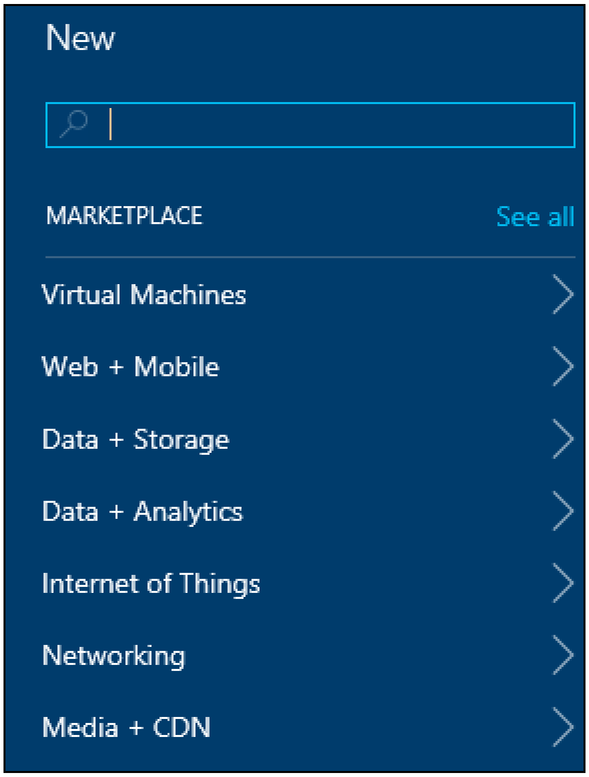
From left to right, here's what these icons mean:

- Clicking the bell shows notifications from this session. For example, if you create a new VM, when it's finished, it will put a notification here.
- Clicking the pencil puts the Dashboard into edit mode, just like clicking Edit Dashboard above.
- Clicking the gear icon brings up the Settings screen for the portal, where you can do things like enable or disable toast notifications, set the default language, and so on.
- Clicking the smiley face will show a dialog you can use to send feedback to the portal team.
- Clicking the question mark will show a drop-down menu allowing you to create a new support request, view your current support requests, and so on.
- The last field shows the account you have used to log into the portal. If you administer more than one subscription, this will show the list of Azure ADs to which the user belongs. You can click this to sign out, change your password, or submit an idea.

Creating and viewing resources

As you make selections, the portal scrolls to the right. The separate sections that get opened are called blades.

Click New in the main hub. You see a categorized list of the resources available, as shown in Figure 1-5. This is a new blade.



New

MARKETPLACE

Figure 1-5 Creating a new resource in the Azure portal.

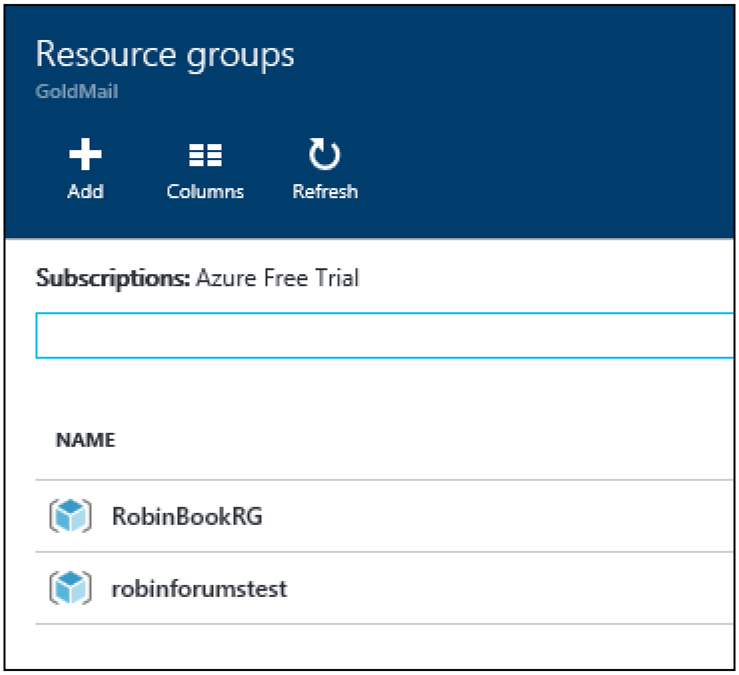
If you click See All, it will take you to the Azure Marketplace. The Marketplace contains all of the resources that you can use in Azure. This includes everything from VM images, which are certified before being made available, all of the SQL Server options, and Web Apps. It also includes applications such as Drupal and WordPress. To add any resource, you can search for it, then select it to add it to your Azure subscription.

You can also select a category on this blade. It will show the list of resources valid for that category, and you can then select which one you want to create. For example, to create a VM, you would click the Virtual Machines category; to create a storage account or a SQL Server, you would click Data + Storage.

Once you have created some resources, there are several ways to view them. Let's look back in the main hub (Figure 1-1), which has two helpful options—Resource Groups and All Resources.

View by resource group

Use this option to see all of your resources by resource group. Click Resource Groups, and you see a blade like Figure 1-6 showing all of your resource groups.

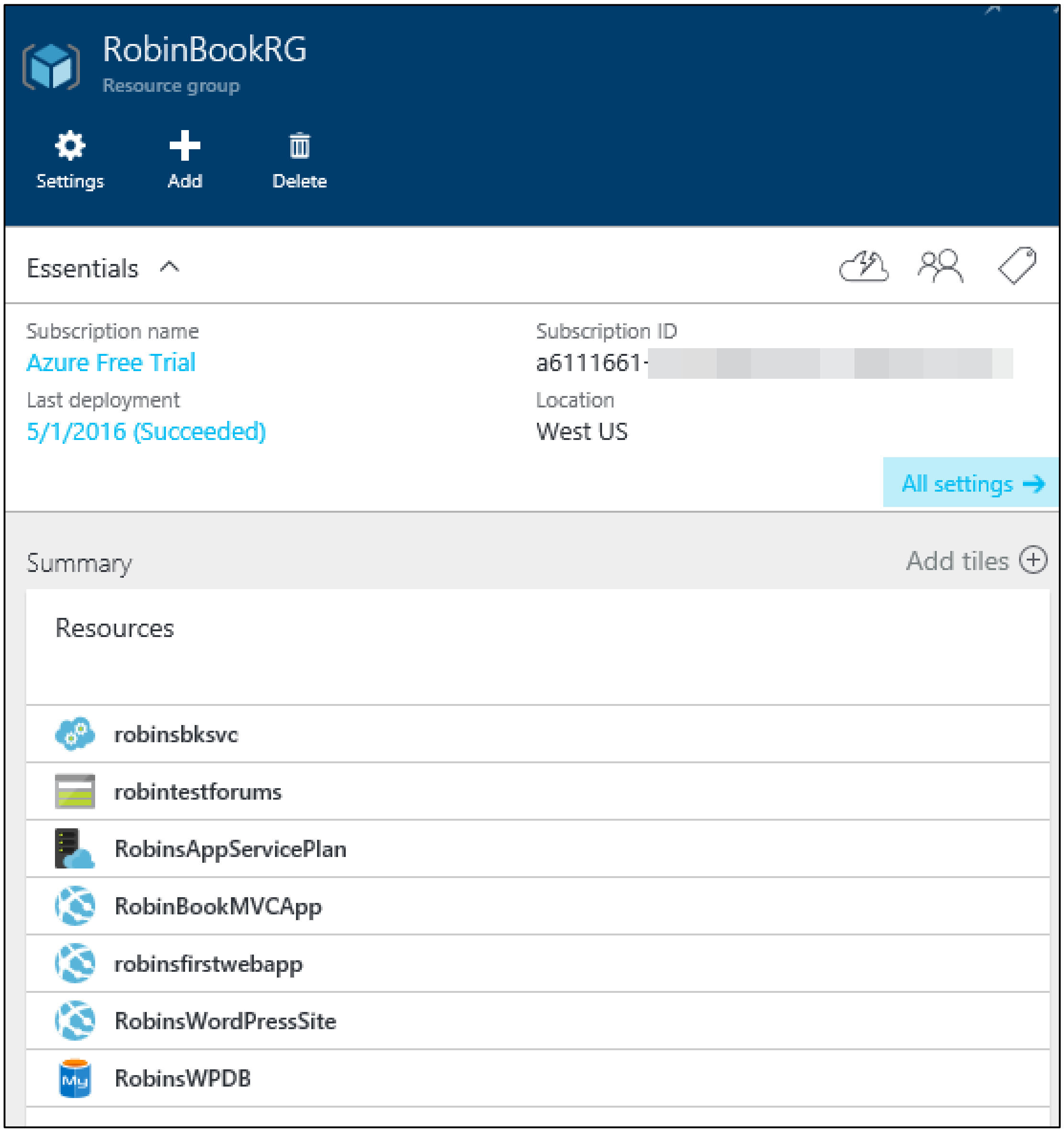


Subscriptions: Azure Free Trial

NAME
RobinBookRG
robinforumstest

Figure 1-6 Screenshot showing all of your resource groups in the Azure portal.

Next, select one of the resource groups, and it shows all of the resources deployed to that group (Figure 1-7).



RobinBookRG
Resource group

Essentials

Subscription name
Azure Free Trial

Subscription ID
a6111661-[redacted]

Last deployment
5/1/2016 (Succeeded)

Location
West US

Summary

Resources

- robinsbksvc
- robintestforums
- RobinsAppServicePlan
- RobinBookMVCApp
- robinsfirstwebapp
- RobinsWordPressSite
- RobinsWPDB

Figure 1-7 List of resources in the selected resource group.

You can click any of the resources here, and they will be displayed in a new blade.

Click All Settings to show the Settings blade (Figure 1-8). From there, you can look at the costs by resource, view the deployment history of the resources, set tags and locks, and manage what users have access to this resource group.

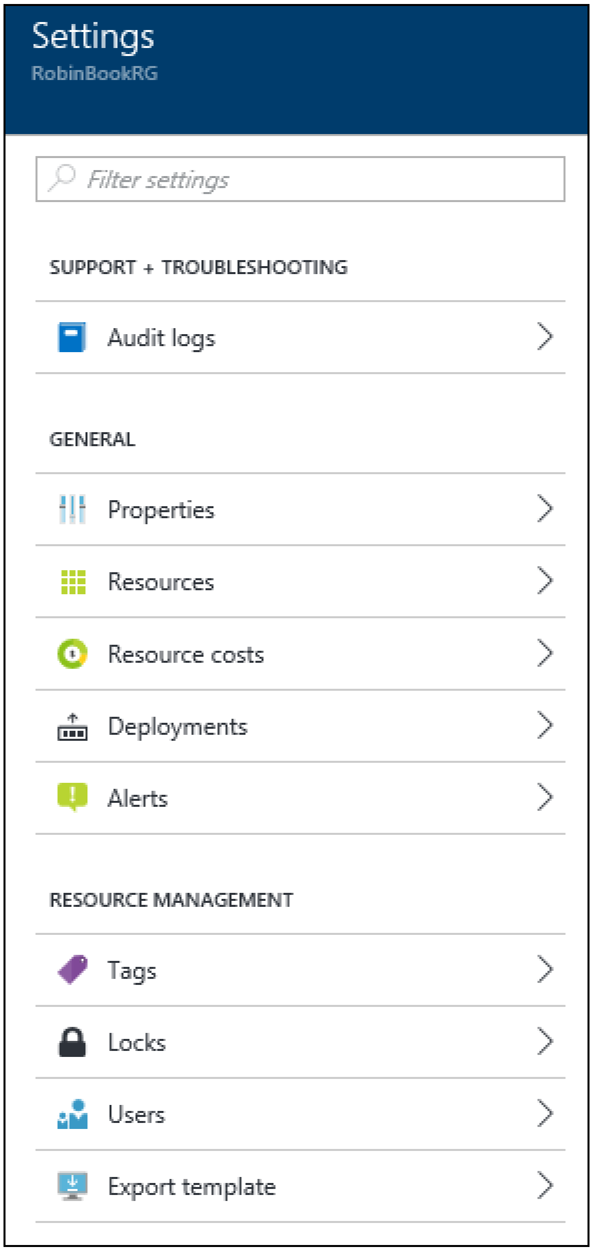
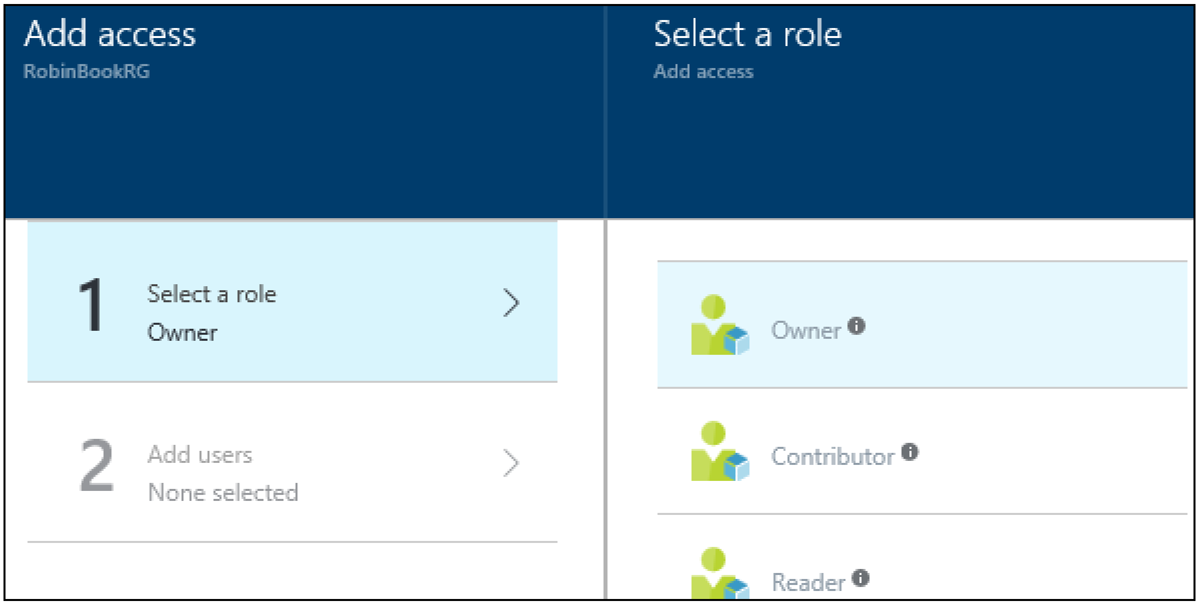


Figure 1-8 Settings blade when looking at resources in a resource group.

This is where you can use RBAC to control access to all of the resources in the same resource group at one time by assigning roles to users. The user has to be set up in the Azure AD, which is done in the classic Azure portal (https://manage.windowsazure.com).

Let's give VM Contributor access to another user account. This is granting the ability to manage the VMs but not the ability to manage the access to the VMs. So this new user could not grant access to anybody else. If you want someone to have full administrative privileges of all the resources in the resource group, you can grant that user the Owner role.

In the Users blade, click Add. You are prompted to select the role you want the user to have (Figure 1-9).



Add access

RobinBookRG

1 Select a role
Owner

2 Add users
None selected

Select a role

Add access

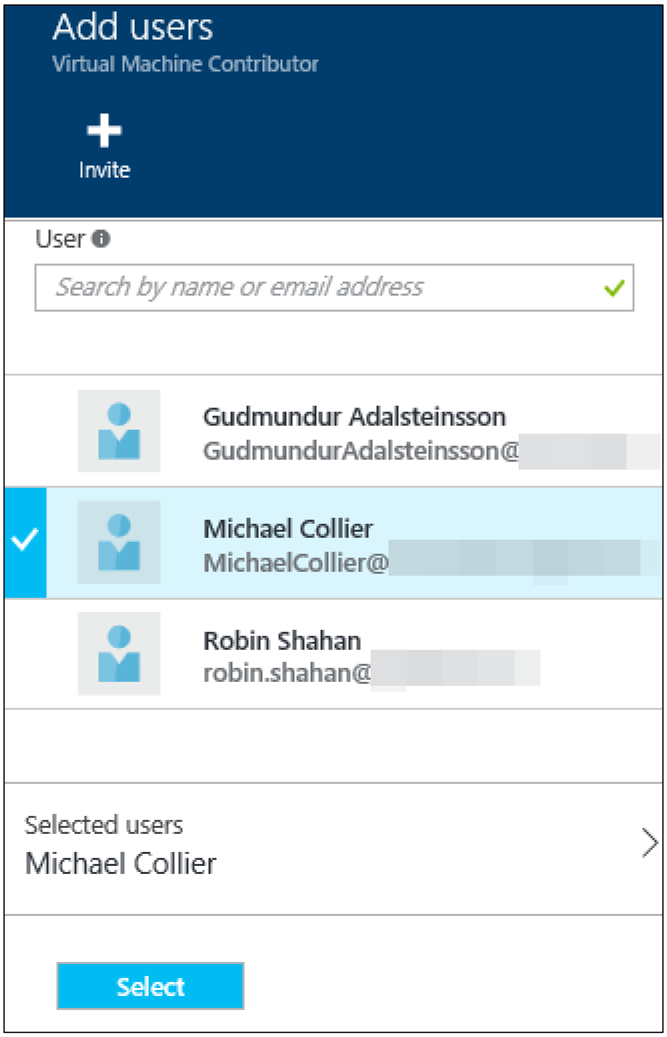
Owner

Contributor

Reader

Figure 1-9 Select a role to assign to a new user.

Look through the list and find the Virtual Machine Contributor role and select it. The Add Access blade highlights Add Users and shows a list of users to the right from which to select (Figure 1-10). Select an account and then click Select at the bottom of the blade.



Add users

Virtual Machine Contributor

Invite

User

Search by name or email address

Gudmundur Adalsteinsson

GudmundurAdalsteinsson@[redacted]

✓ Michael Collier

MichaelCollier@[redacted]

Robin Shahan

robin.shahan@[redacted]

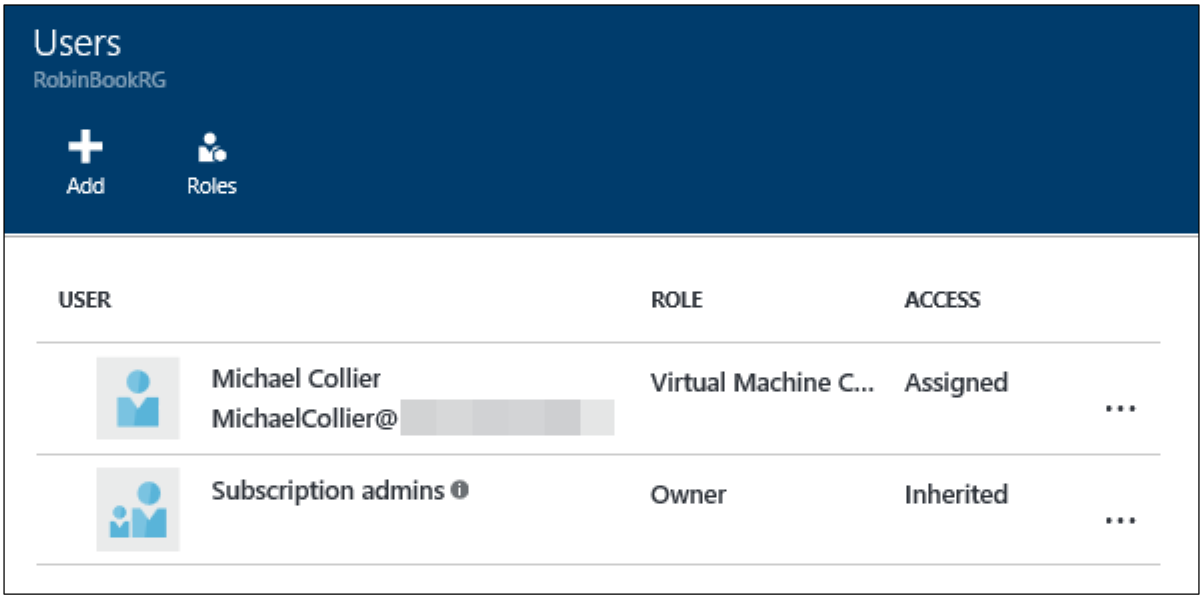
Selected users

Michael Collier

Select

Figure 1-10 Select a user to add.

Next, click OK on the Add Access blade. It returns to the Users screen, which now reflects the user(s) added and their roles (Figure 1-11).



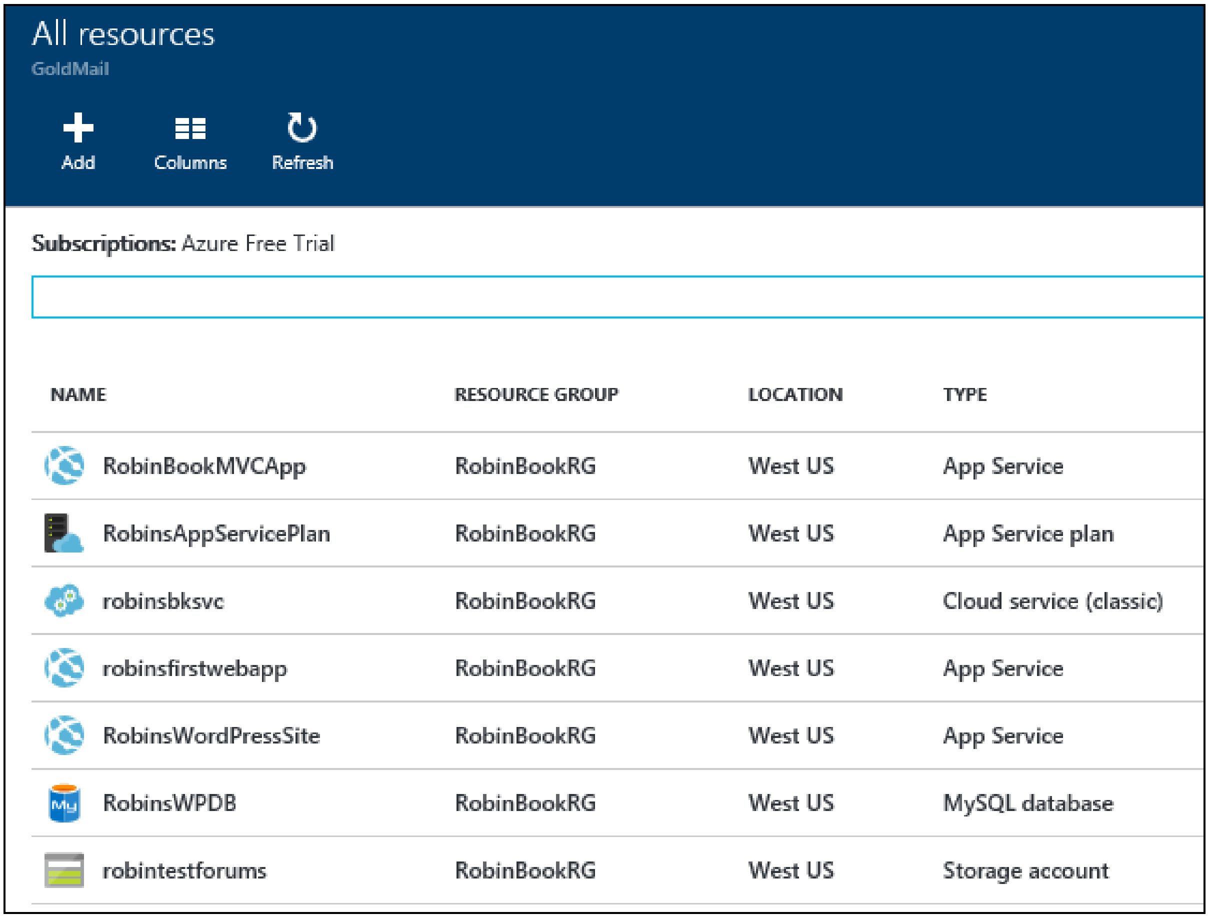
USER	ROLE	ACCESS
Michael Collier MichaelCollier@[redacted]	Virtual Machine C...	Assigned
Subscription admins 🛈	Owner	Inherited

Figure 1-11 List of users and their assigned roles.

I added the Virtual Machine Contributor role for Michael Collier. This means that Michael Collier now has the ability to manage the VMs in that resource group.

View by resource

Back in the main hub (Figure 1-1), let's look at the other view of our resources. Click All Resources. This shows exactly what you expect—a list of all your resources (Figure 1-12). You can edit the columns by selecting Columns. I've added the Type column because I can never remember what all of the icons mean.



All resources

GoldMail

Subscriptions: Azure Free Trial

NAME	RESOURCE GROUP	LOCATION	TYPE
RobinBookMVCApp	RobinBookRG	West US	App Service
RobinsAppServicePlan	RobinBookRG	West US	App Service plan
robinsbksvc	RobinBookRG	West US	Cloud service (classic)
robinsfirstwebapp	RobinBookRG	West US	App Service
RobinsWordPressSite	RobinBookRG	West US	App Service
RobinsWPDB	RobinBookRG	West US	MySQL database
robintestforums	RobinBookRG	West US	Storage account

Figure 1-12 List of resources in the subscription.

Clicking any resource brings up a blade for that specific resource.

Subscription management and billing

In this section, we'll look at the subscription types available and how to manage access to your subscription, as well as how to check your current billing balance.

Available subscriptions

There are several different kinds of subscriptions providing access to Azure services. You must have a Microsoft account (created by you for personal use) or a work or school account (issued by an administrator for business or academic use) to access these subscriptions.

Let's take a look at the most common subscriptions:

- **Free accounts** The link to sign up for a free account is on the front page of azure.com. This gives you a $200 credit over the course of 30 days to try out any combination of resources in Azure. If you exceed your credit amount, your account will be suspended. At the end of the trial, your services will be decommissioned and will no longer work. You can upgrade this to a pay-as-you-go subscription at any time.

- **MSDN subscriptions** If you have an MSDN subscription, you get a specific amount in Azure credit each month. For example, if you have a Visual Studio Enterprise with MSDN subscription, you get $150 per month in Azure credit.

  If you exceed the credit amount, your service will be disabled until the next month starts. You can turn off the spending limit and add a credit card to be used for the additional costs. Some of these costs are discounted for MSDN accounts. For example, you pay the Linux price for VMs running Windows Server, and there is no additional charge for Microsoft Servers such as Microsoft SQL Server. This makes MSDN accounts ideal for development and test scenarios.

  For more information and to see the available MSDN subscription tiers, check out http://azure.microsoft.com/pricing/member-offers/msdn-benefits-details/. Note that these subscriptions are to be used for development and testing, not for production.

- **BizSpark accounts** The BizSpark program provides a lot of benefits to startups, not the least of which is access to all of Microsoft's software for development and test environments for up to five MSDN accounts. In addition to these benefits, you get $150 in Azure credit for each of those five MSDN accounts, and you pay reduced rates for several of the Azure services, such as Windows Virtual Machines.

  For more information, check out http://azure.microsoft.com/offers/ms-azr-0064p/.

- **Pay-as-you-go** With this subscription, you pay for what you use by attaching a credit card or debit card to the account. If you are an organization, you can also be approved for invoicing.

  For more information, check out http://azure.microsoft.com/offers/ms-azr-0003p/.

- **Enterprise agreements** With an enterprise agreement, you commit to using a certain amount of services in Azure over the next year, and you pay that amount ahead of time. The commitment that you make is consumed throughout the year. If you exceed the commitment amount, you can pay the overage in arrears. Depending on the amount of the commitment, you get a discount on the services in Azure.

  For more information, check out http://azure.microsoft.com/pricing/enterprise-agreement/.

Share administrative privileges for your Azure subscription

Once you have signed up for an Azure subscription, you can give administrative access to additional Microsoft accounts. This is done differently depending on whether you are using the classic Azure portal or the Azure portal. If you want the new account to be able to administer the subscription in both portals, you must make sure it has been given access in each portal. You want to do this if you need someone to administer the Azure AD for the subscription or if the subscription contains classic resources.

As we discussed previously, the Azure portal uses RBAC, and the classic Azure portal does not. This means in the classic Azure portal, you can *only* grant full administrative (co-admin) access to an account.

Add administrative privileges in the Azure portal

We just saw how to grant administrative privileges to a resource group in the Azure portal. Granting administrative privileges is almost the same process, except instead of selecting a resource group, you select the subscription.

Go to the hub (the selector on the far left) and select Subscriptions, then select the Subscription to which you want to add an administrator. Click Settings to go to the Settings blade, and then select Users.

From the Users blade, you can use the same process we used before. Click Add, select the Owner role this time, select the user to whom you want to grant this role, and click OK to add the user to the RBAC settings for the subscription. They will show up in the Users blade with the user's new permission.

If you want to grant access to one specific resource, you can select the resource from the All Resources blade, go to Settings > Users, and add a user and role exactly the same way.

Granting administrative privileges in the classic Azure portal

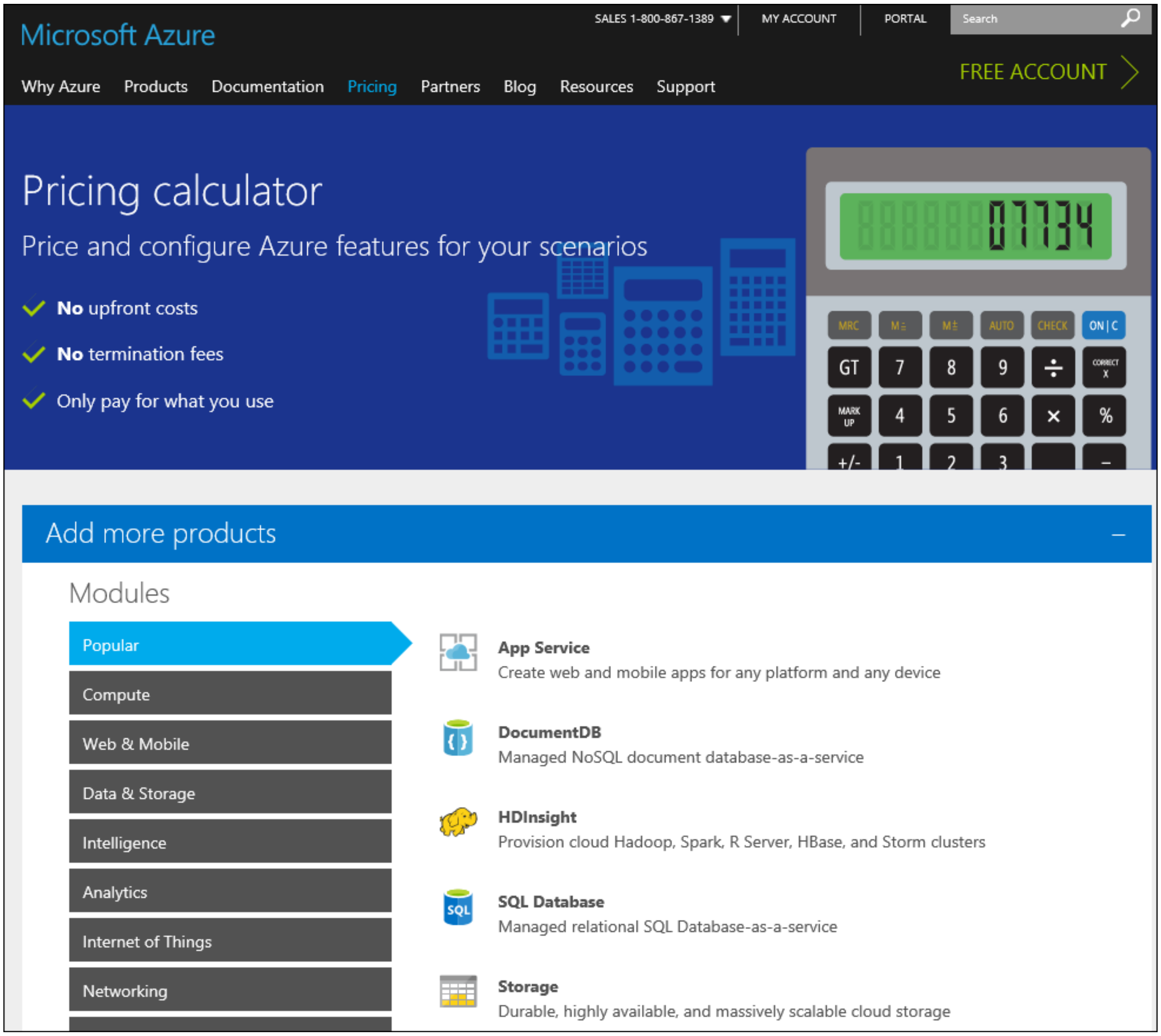
To grant administrative access to an account in the classic Azure portal, add the user's account as a co-administrator to the subscription. This account will have all of the same privileges as the owner of the original subscription, but it does not allow the user to change the service administrator or to add and remove other co-administrators.

By using the classic Azure portal with administrative access, the user can access and maintain classic resources, such as classic storage accounts. There are also some Resource Manager resources that the account can impact, such as Web Apps. However, this user can't see storage accounts and virtual machines created with the Resource Manager deployment model.

Note that co-administrators are automatically added to the Subscription Admin RBAC role.

Pricing calculator

Pricing for your Azure infrastructure can be estimated by using the pricing calculator found at http://azure.microsoft.com/pricing/calculator/ (Figure 1-13).



Pricing calculator

Price and configure Azure features for your scenarios

- ✓ **No** upfront costs
- ✓ **No** termination fees
- ✓ Only pay for what you use

Add more products –

Modules

- Popular
- Compute
- Web & Mobile
- Data & Storage
- Intelligence
- Analytics
- Internet of Things
- Networking

App Service
Create web and mobile apps for any platform and any device

DocumentDB
Managed NoSQL document database-as-a-service

HDInsight
Provision cloud Hadoop, Spark, R Server, HBase, and Storm clusters

SQL Database
Managed relational SQL Database-as-a-service

Storage
Durable, highly available, and massively scalable cloud storage

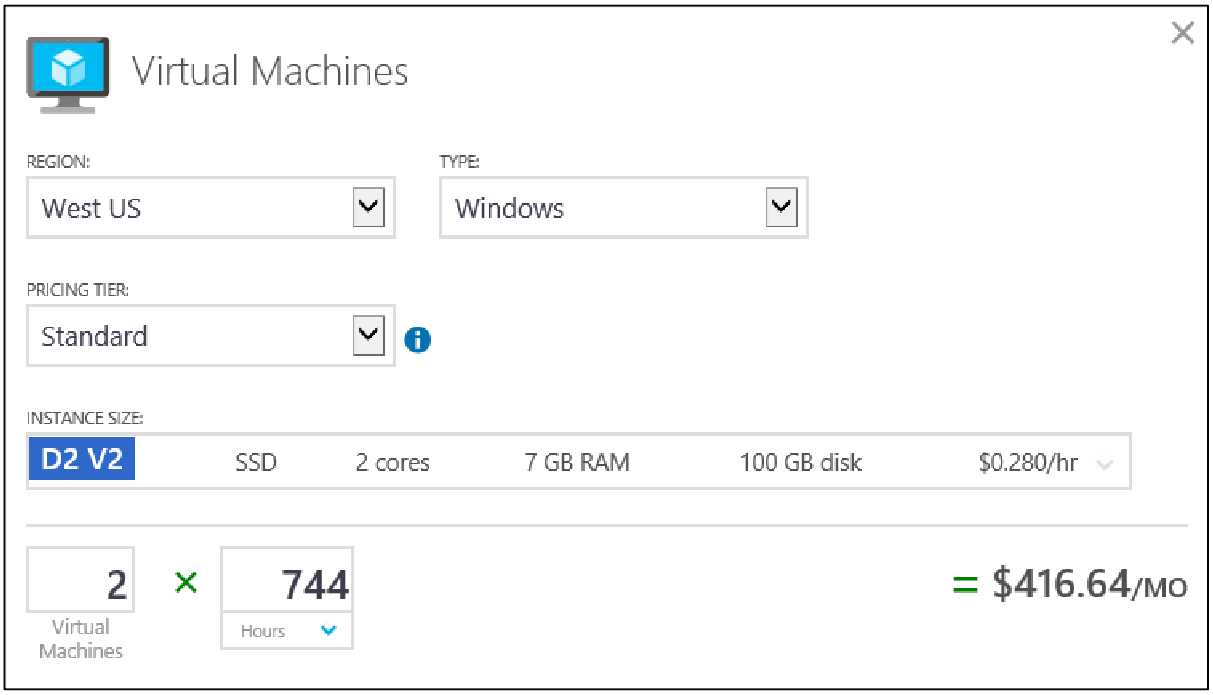
Figure 1-13 The pricing calculator.

The pricing for each service in Azure is different. Many Azure services provide Basic, Standard, and Premium tiers, usually with several price and performance levels in each tier, allowing you to select an appropriate performance level for your use of the service. As you change the selections, the pricing estimate is provided on the right side of the page. You can look at each feature separately or select several resources to estimate multiple features together.

Let’s create a pricing example for two virtual machines and a storage account with 500 GB of data.

1. Click Compute > Virtual Machines. A message appears saying it has been added.
2. Click Data & Storage > Storage. A message appears saying it has been added.
3. Now, scroll to the bottom of the page, and you see it has added Virtual Machines and Storage. It also shows the total for all the resources you’ve specified.
4. On the Virtual Machines tile, set the Region to the one closest to you and set Type to Windows (other options include Linux). Next, set the Pricing Tier to Standard. Then, check the drop-down list on instance size and select a D2 V2. If we set the storage to Premium storage, this will also work for DS2 V2 VMs because the pricing is identical for D2 and DS2 VMs. D2 VMs use Standard storage; DS2 VMs use Premium storage.

Next, set the number of virtual machines to 2 (Figure 1-14). This shows an estimated cost for having those two virtual machines.



Virtual Machines

REGION: West US

TYPE: Windows

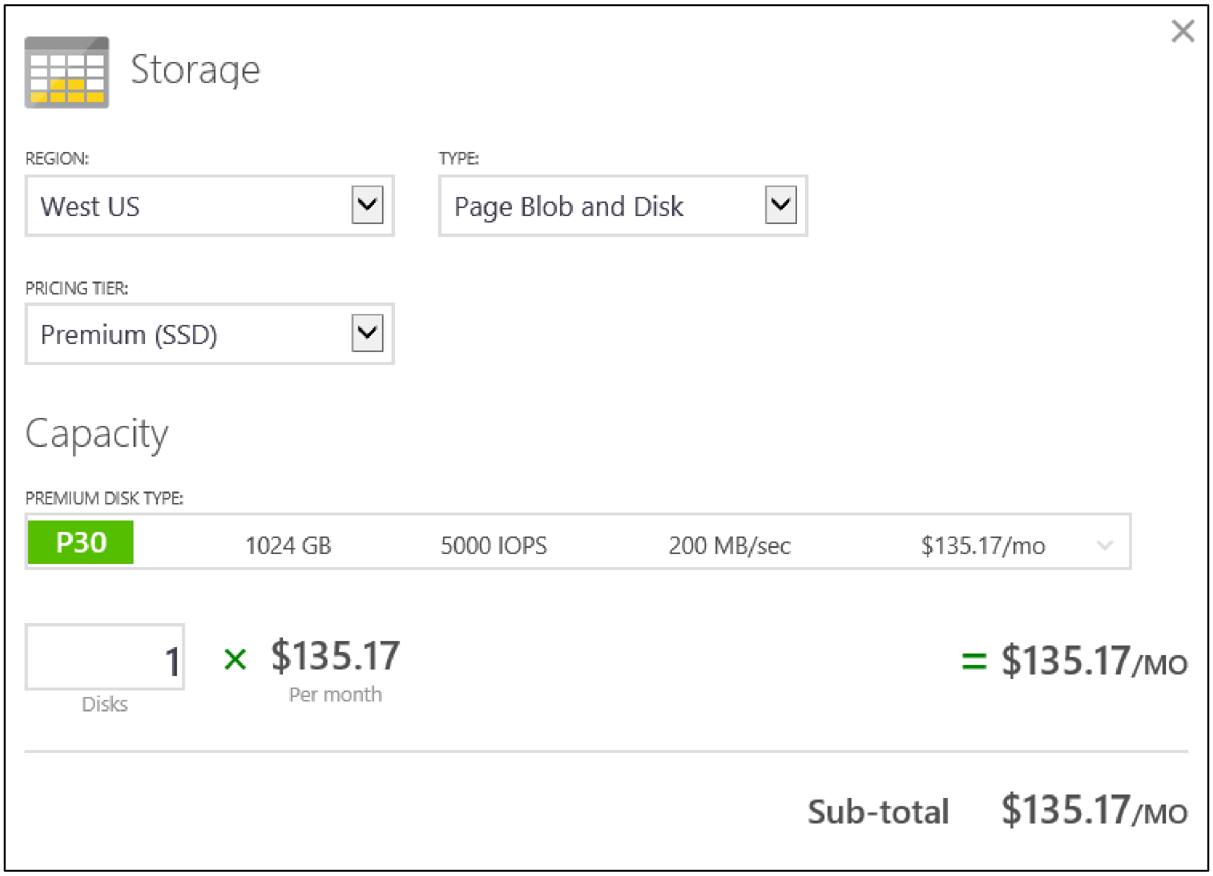
PRICING TIER: Standard

INSTANCE SIZE: D2 V2 SSD 2 cores 7 GB RAM 100 GB disk $0.280/hr

2 Virtual Machines × 744 Hours = $416.64/MO

Figure 1-14 Calculating pricing on two virtual machines.

5. On the Storage tile, set the Region. Set Type to Page Blob and Disk, indicating that we are going to use this storage account to store the VHD files for our virtual machines. Set the Pricing Tier to Premium (SSD). Select the P30 disk. If you are deploying VMs, you want to use Premium storage for the best reliability and speed; Premium storage only uses SSDs. This will give an estimated cost for that configuration (Figure 1-15).



Storage

REGION: West US

TYPE: Page Blob and Disk

PRICING TIER: Premium (SSD)

Capacity

PREMIUM DISK TYPE:

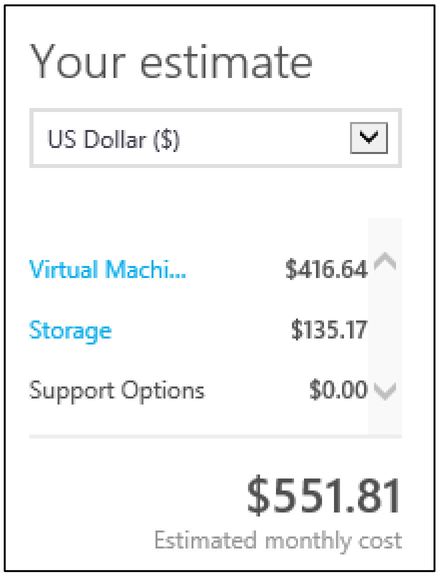
P30	1024 GB	5000 IOPS	200 MB/sec	$135.17/mo

1 Disks × $135.17 Per month = $135.17/MO

Sub-total $135.17/MO

Figure 1-15 Calculating pricing on storage.

6. Now if you look at the total section, it gives a total estimated cost for the two virtual machines and the storage (Figure 1-16).



Your estimate

US Dollar ($)

Virtual Machi...	$416.64
Storage	$135.17
Support Options	$0.00

$551.81
Estimated monthly cost

Figure 1-16 Calculating total cost of selected resources.

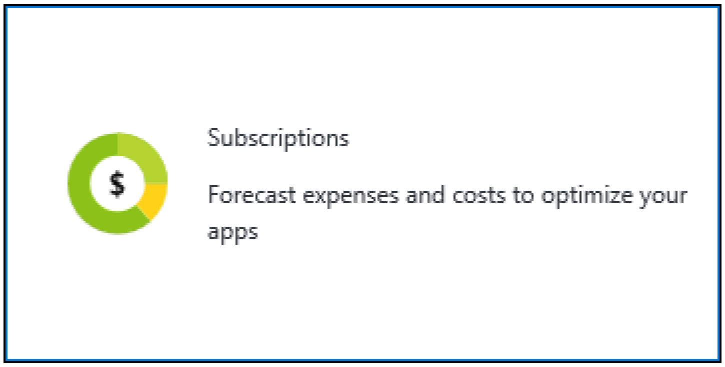
7. If you click Export Estimate, it will export all of the data to an Excel spreadsheet.

The pricing calculator can be helpful in estimating your Azure costs for new projects you want to add or for an entire infrastructure design.

> **Note** The overall pricing plan page does not include variations by region, but you can find those if you go to the individual service pricing pages at http://azure.microsoft.com/pricing/ and select the service in which you're interested. At that point, you can also select the specific region.

Viewing billing in the Azure portal

An important component of using Azure is being able to view your billing information. If you have an account that allows you a certain amount of credit, it's nice to know how much you have left and to view where the costs are accumulating. To see your current usage, click the Subscriptions tile in the Dashboard of the Azure portal (Figure 1-17).

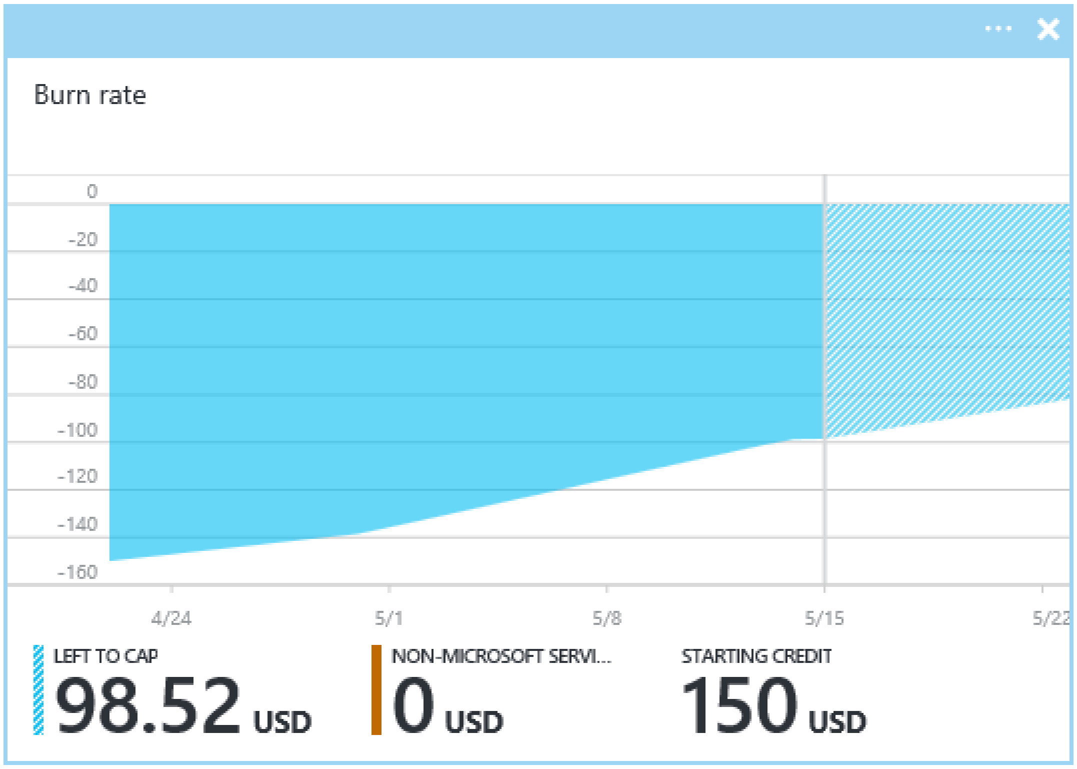


Subscriptions

Forecast expenses and costs to optimize your apps

Figure 1-17 The Subscriptions tile on the Dashboard of the Azure portal.

Click this tile to go to the Subscriptions blade, then select the subscription you want to examine. The Subscriptions blade is displayed. On the bottom of that blade is a tile showing the amount left before you hit the cap, what the starting credit was, and the burn rate (Figure 1-18).



LEFT TO CAP 98.52 USD

NON-MICROSOFT SERVI... 0 USD

STARTING CREDIT 150 USD

Figure 1-18 The overall cost information for the selected subscription.

We can see that for the account displayed above, the cap is $150 (starting credit), and $98.52 of that has been used so far. Underneath this graphic is the cost by resource. This account is taken up by the small web app that is running, but if there are VMs, storage accounts, and so on, the total cost of each resource would be displayed here (Figure 1-19).

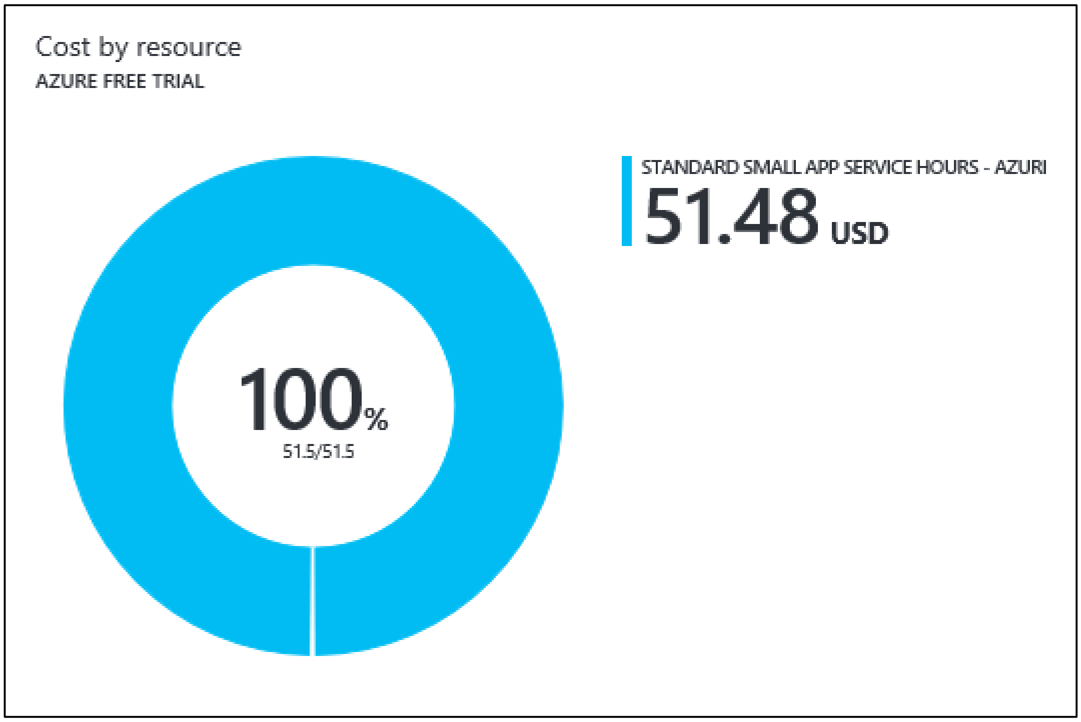
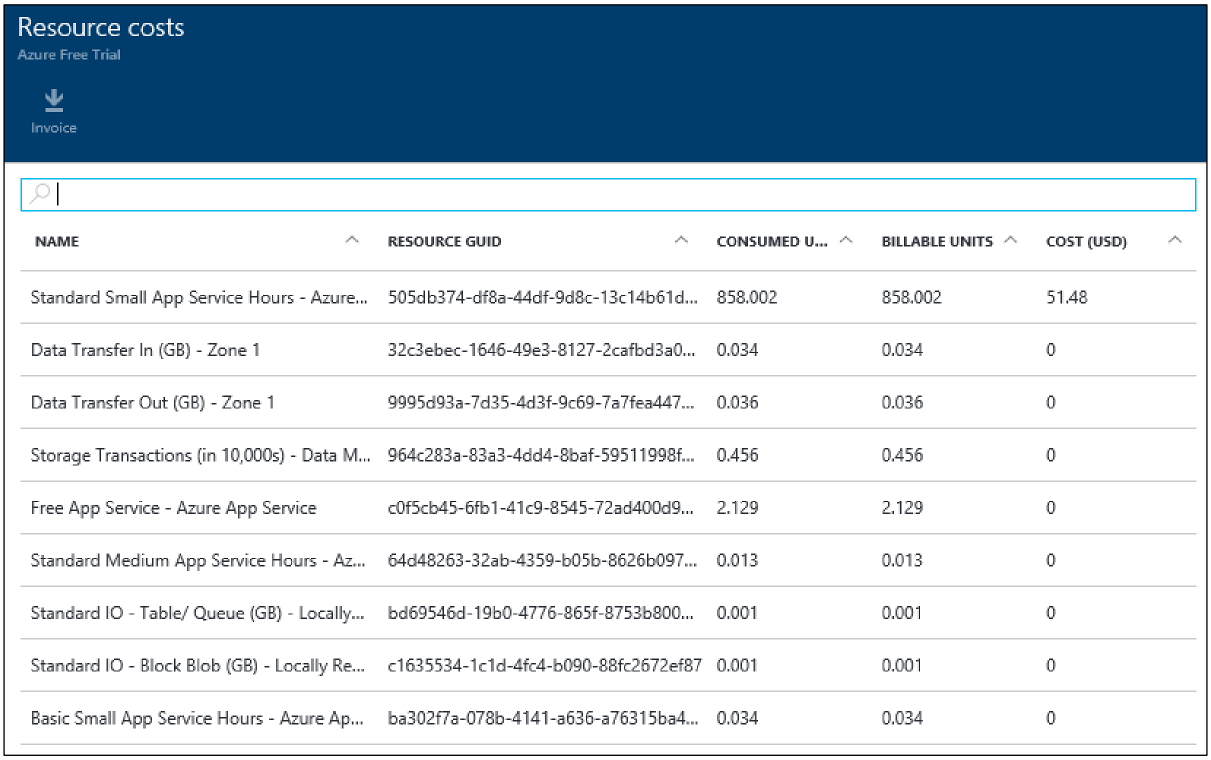


Figure 1-19 The cost by resource for the selected subscription.

If you click the graphic, it will show the resource costs by resource in a new blade (Figure 1-20).



NAME	RESOURCE GUID	CONSUMED U...	BILLABLE UNITS	COST (USD)
Standard Small App Service Hours - Azure...	505db374-df8a-44df-9d8c-13c14b61d...	858.002	858.002	51.48
Data Transfer In (GB) - Zone 1	32c3ebec-1646-49e3-8127-2cafbd3a0...	0.034	0.034	0
Data Transfer Out (GB) - Zone 1	9995d93a-7d35-4d3f-9c69-7a7fea447...	0.036	0.036	0
Storage Transactions (in 10,000s) - Data M...	964c283a-83a3-4dd4-8baf-59511998f...	0.456	0.456	0
Free App Service - Azure App Service	c0f5cb45-6fb1-41c9-8545-72ad400d9...	2.129	2.129	0
Standard Medium App Service Hours - Az...	64d48263-32ab-4359-b05b-8626b097...	0.013	0.013	0
Standard IO - Table/ Queue (GB) - Locally...	bd69546d-19b0-4776-865f-8753b800...	0.001	0.001	0
Standard IO - Block Blob (GB) - Locally Re...	c1635534-1c1d-4fc4-b090-88fc2672ef87	0.001	0.001	0
Basic Small App Service Hours - Azure Ap...	ba302f7a-078b-4141-a636-a76315ba4...	0.034	0.034	0

Figure 1-20 The details of the cost by resource for the selected subscription.

The ability to view the billing information on a regular basis is helpful when managing the costs for your Azure subscription. If you have a subscription with a monthly credit, you can tell when you're getting close to the cap. You can also tell where your costs are accumulating. Also, if you provision some VMs and forget they're out there, you'll be able to see them because they will have billing associated with them.

Azure Billing APIs

In addition to viewing the billing in the portal, you can access the billing information programmatically through the Azure Billing REST APIs for a specific subscription. There are two APIs that you can use.

- The Azure Usage API enables you to retrieve your usage data. You can fine-tune the billing usage information retrieved to be grouped by resource if you have used the resource tags that can be set through most of the Settings screens. For example, you can tag each of the resources in a resource group with a department name or project name, then track the costs specifically for that one tag.
- The Azure RateCard API enables you to list all of the resources that you can use, along with the metadata and pricing information about each of those resources.

To get you started, there are Billing API code samples on GitHub that you can download and try out. They are located here: https://github.com/Azure/BillingCodeSamples.

Azure documentation and samples

In this section, we'll talk about the Azure documentation and samples, including where you can find them and how you can contribute bug fixes, changes, or even entirely new articles and samples to the Azure community.

Documentation

The Azure documentation can be found at http://azure.microsoft.com. This is the conceptual documentation, which explains the services, how they work, how to use them, and so on. The reference documentation is on MSDN (http://msdn.microsoft.com). For example, the documentation for the REST APIs is on MSDN, and it shows every command and all of their options.

All of the conceptual documentation at azure.microsoft.com resides on GitHub. You can contribute to the documentation by adding articles or modifying articles to include information you believe will be helpful to others. To view the contributor guide and the current documentation, please go to https://github.com/Azure/azure-content.

Samples

In addition to the documentation, there are many Azure samples to help you get started with Azure, also stored in GitHub. For example, Azure Storage has getting-started samples for .NET and Java for Blob storage, Table storage, Queue storage, and File storage. You can use these samples to help you, and you can also contribute to this repository. These samples can be found here: http://github.com/azure-samples.

For the Resource Manager resources, there is a repository of quick start templates available here: https://github.com/Azure/azure-quickstart-templates. This has templates for creating many resources such as the Azure Content Delivery Network, Azure Key Vault, virtual machines, virtual networks, and storage accounts.

CHAPTER 2

Azure App Service and Web Apps

In this chapter, we take a look at the Azure App Service, consisting of Web Apps, Logic Apps, Mobile Apps, API Apps, and Function Apps. We focus on Web Apps and how they work together with the App Service. We create a web app and publish it to Azure. We also look at the options for prebuilt web apps offered by Azure.

App Service and App Service plans

Before we talk about Web Apps, let's talk about App Service and the App Service plans.

What is an App Service?

The App Service is a service that hosts one of five kinds of applications:

- Web Apps
- Mobile Apps
- Logic Apps
- API Apps
- Function Apps

Each app runs in its own app service. When you look in the Azure portal to see your website, you will look for the app service in which it is running. It conveniently has the same name as the app it's hosting.

So what is an App Service plan?

An App Service plan defines the capacity and resources to be shared among one or more app services that are assigned to that plan.

The following are some of the criteria you can define when creating an App Service plan.

- Location (such as West US)
- Instance count
- Pricing tier (such as Free, Standard, or Premium) providing distinct settings for a variety of performance and service capabilities:
    - Number of cores or instance size
    - Amount of memory
    - Amount of storage
    - Maximum number of instances
    - Autoscaling options (depends on tier—automatic, manual, or none)

When you deploy your app service for the first time, you specify which App Service plan you want to use. At deployment time, you can select an App Service plan you have created or create a new App Service plan.

How does this help you?

With infrastructure as a service (IaaS), you can create your own virtual machines (VMs), deploy your apps to them, and deal with the IIS setup and application pools and so on. Then, every time you change an app, you have to deploy it to all the VMs again. If you scale it out, and you have four VMs or eight VMs, it just becomes more onerous. With IaaS, you are responsible for the continuing management of your service. Using App Service plans enables you to run multiple applications on one set of VMs, even if each of the applications is deployed separately.

For example, let's say you have five websites and three mobile apps that you want to host. You could run each one on its own VM, which would require 8 VMs. If you wanted redundancy (recommended), that would require 16 VMs. Even if you select small instances, the cost adds up really quickly. Plus, you have to scale each set of VMs separately.

If you could run those eight applications on the same set of two VMs, it would be more cost-effective and easier to manage. This is what using App Service plans does for you. You set up one App Service plan with a specific VM size, number of instances, etc. Then, you deploy the eight applications, specifying the same App Service plan for each one. This results in all eight applications running on that same set of two VMs. You can deploy and update each application as needed—you don't have to update them all at the same time.

When you create your App Service plan, the resources you requested are allocated for you. When you deploy an app to that App Service plan, it simply deploys the applications to those allocated resources.

If you decide you want to have four VMs instead of two, you simply go to the Azure portal and modify the App Service plan, changing the number of instances from two to four. It will create two more VMs and deploy your apps to them for you. If you are using small VMs and want to scale up to medium VMs, you can simply modify the Pricing Tier in the App Service plan, and it will scale up.

With web apps running in an app service using an App Service plan, the management is handled for you, and you can easily scale up and out just by changing the settings of the App Service plan.

How to create an App Service plan in the Azure portal

Now, I'll show you how to create an App Service plan using the Azure portal. Later, I'll show you how to create a web app and deploy it to an app service using that App Service plan.

1. Log in to the Azure portal.
2. Click New, then click See All, as displayed in Figure 2-1.

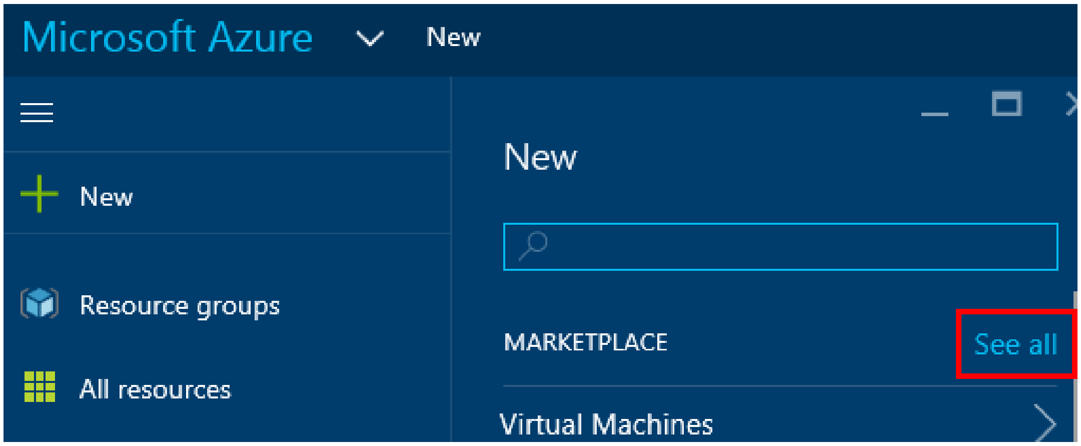


Figure 2-1 Go to the Marketplace to search for a resource to add.

3. It opens the search screen for the Marketplace (Figure 2-2). Type **app service plan** in the search box and press Enter.

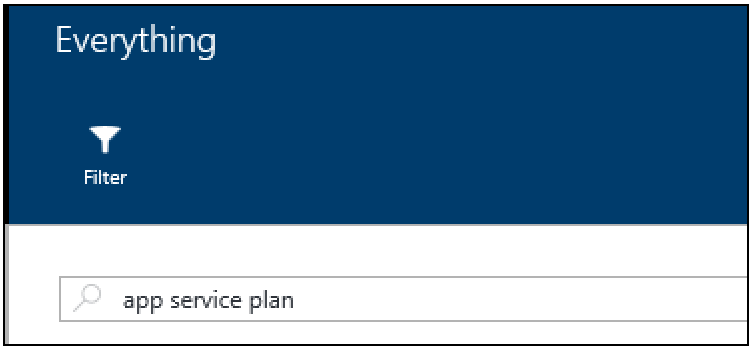
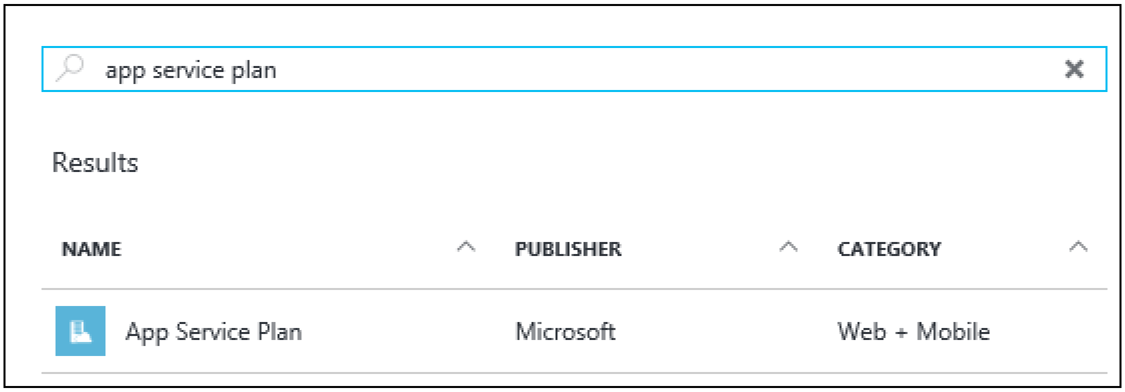


Figure 2-2 The input screen for searching the Marketplace.

4. Select App Service Plan in the search results, as shown in Figure 2-3.



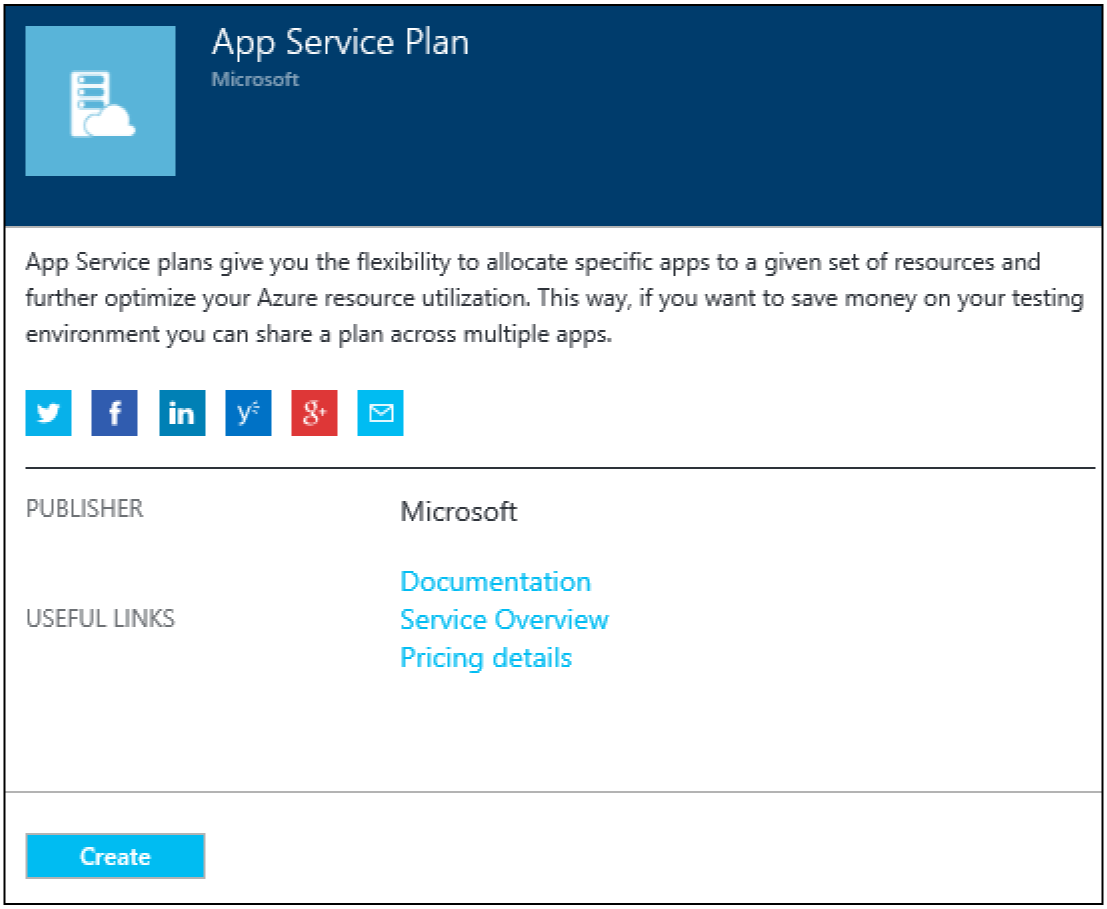
app service plan

Results

NAME	PUBLISHER	CATEGORY
App Service Plan	Microsoft	Web + Mobile

Figure 2-3 The search results for App Service plan.

5. Click Create on the App Service Plan blade displayed in Figure 2-4.



App Service Plan

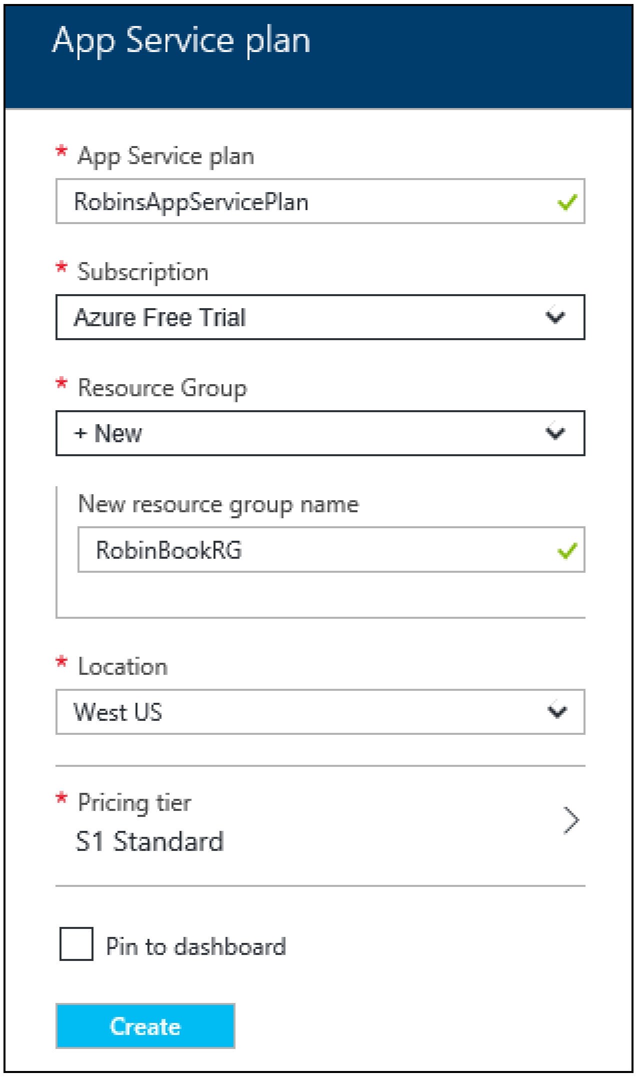
Microsoft

App Service plans give you the flexibility to allocate specific apps to a given set of resources and further optimize your Azure resource utilization. This way, if you want to save money on your testing environment you can share a plan across multiple apps.

PUBLISHER	Microsoft
USEFUL LINKS	Documentation Service Overview Pricing details

Figure 2-4 Click Create to create a new App Service plan.

6. After you see something similar to the App Service Plan blade displayed in Figure 2-5, you can define the parameters for your App Service plan.



App Service plan

* App Service plan

RobinsAppServicePlan ✓

* Subscription

Azure Free Trial

* Resource Group

+ New

New resource group name

RobinBookRG ✓

* Location

West US

* Pricing tier

S1 Standard

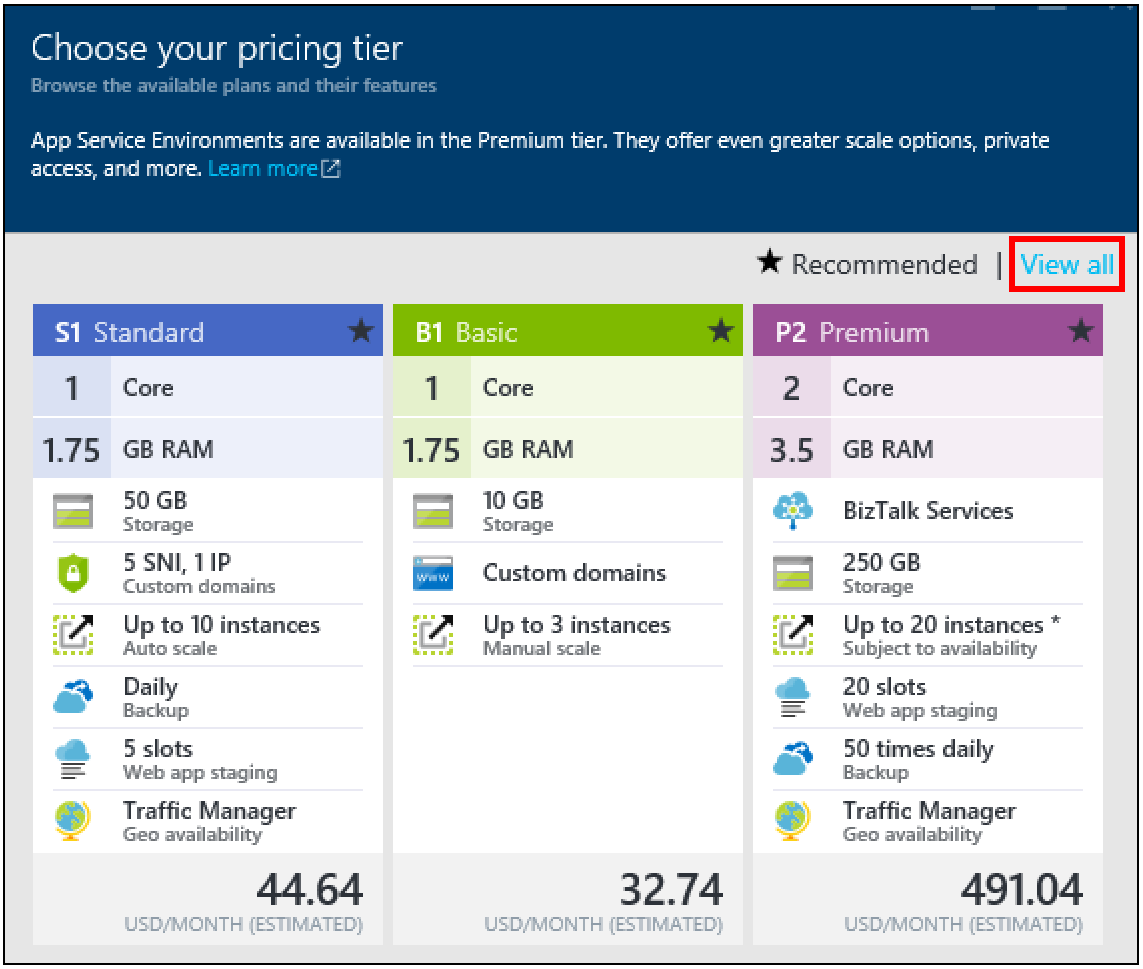
□ Pin to dashboard

Figure 2-5 The fields to be filled in for your new App Service plan.

- **App Service Plan** This is what you would like to name your App Service plan. Make this something you can recognize when you want to use the plan later.
- **Subscription** If you have multiple Azure subscriptions administered by this account, this will have a drop-down list of subscriptions, and you can select which one to use.
- **Resource Group** Resource groups provide a logical container for a related set of resources. For example, you could put all of the resources you create for this book in the same resource group. When you're finished, you can delete the resource group, and it will deallocate and remove all of those resources for you. Let's create a new resource group for our App Service plan; later in this chapter, we will create a web app and assign it to our App Service plan. Leave the value as +New and specify the name of your new resource group. It's recommended that you specify something that indicates what the resources are used for.
- **Location** This is the Azure region where the resource group will be hosted. This includes metadata such as audit logs, where each resource in the group resides. This can be different from the resources themselves; this is important for those who care about where data is hosted—for example, those in countries with data sovereignty laws. Also, Resource Manager operations are

sourced through this region, so you typically want it to be the same as most of the resources in the group. For our example, select the region closest to you.

- **Pricing Tier** Click this field to see your choices. The new blade (displayed in Figure 2-6) shows the recommended pricing plans. This is a subset of all of the available pricing tiers. If you want to see all of the plans, click View All on this blade. The pricing plan lets you specify the amount of storage, scalability, backup choices, and so on.



Choose your pricing tier

Browse the available plans and their features

App Service Environments are available in the Premium tier. They offer even greater scale options, private access, and more. Learn more

★ Recommended | View all

S1 Standard ★	B1 Basic ★	P2 Premium ★
1 Core	1 Core	2 Core
1.75 GB RAM	1.75 GB RAM	3.5 GB RAM
50 GB Storage	10 GB Storage	BizTalk Services
5 SNI, 1 IP Custom domains	Custom domains	250 GB Storage
Up to 10 instances Auto scale	Up to 3 instances Manual scale	Up to 20 instances * Subject to availability
Daily Backup		20 slots Web app staging
5 slots Web app staging		50 times daily Backup
Traffic Manager Geo availability		Traffic Manager Geo availability
44.64 USD/MONTH (ESTIMATED)	32.74 USD/MONTH (ESTIMATED)	491.04 USD/MONTH (ESTIMATED)

Figure 2-6 The Pricing Tier blade.

Select the S1 Standard pricing plan and then click Select at the bottom of the blade. Now, your App Service Plan blade should display the pricing plan you selected.

7. Select the check box on the bottom of the App Service Plan blade that says Pin To Dashboard. This will pin a tile to the Dashboard showing your App Service plan, providing easy access to it. Now, click Create. It creates the plan and adds a tile to your Dashboard.

8. After the App Service plan is created, you can click the tile on the Dashboard and modify it. You can also see what apps are using that plan. After the web app is created and deployed, I'll show you how to scale the apps by scaling the App Service plan.

At this point, you can create one or more app services, such as a web app, and assign them to that App Service plan. They will all run on the same VMs.

Creating and deploying Web Apps

Now that you understand App Services and App Service plans, I'll show you what a Web App is, discuss some of its features, and then talk about the various options you have for creating one. Then, I'll show you how to use a couple of those options to create and deploy a Web App.

What is a Web App?

A Web App is a web application that is hosted in an App Service. The App Service is the managed service in Azure that enables you to deploy a web application and make it available to your customers on the Internet in a very short amount of time. As noted above, you don't directly support the VMs on which your web app runs; they are managed for you. In fact, you don't have access to those underlying VMs.

Supported languages include .NET, Java, PHP, Node.js, and Python. In addition to creating your own web app, there are several web applications available to use as a starting point, such as WordPress, Umbraco, Joomla!, and Drupal.

You can use continuous deployment with Team Foundation Server (TFS), GitHub, TeamCity, Jenkins, or BitBucket so that every time you commit a change, a new version of the web app is deployed.

Scaling is done by scaling the App Service plan to which the web app belongs. You can scale the number of instances in and out on demand. You can configure autoscaling so Azure will scale it in or out for you depending on specific performance measures such as CPU percentage. You can also publish your website to multiple locations and use the Azure Traffic Manager to handle the routing of the traffic to the location nearest to your customer.

For diagnostics, you can gather performance statistics, application logging, web server logging, IIS logs, and IIS Failed Request logs. If you're using Microsoft Visual Studio, you can even remotely debug your application while it is running in the cloud.

In short, there are many features available when using Web Apps to make it easy for you to deploy, manage, and troubleshoot a web application.

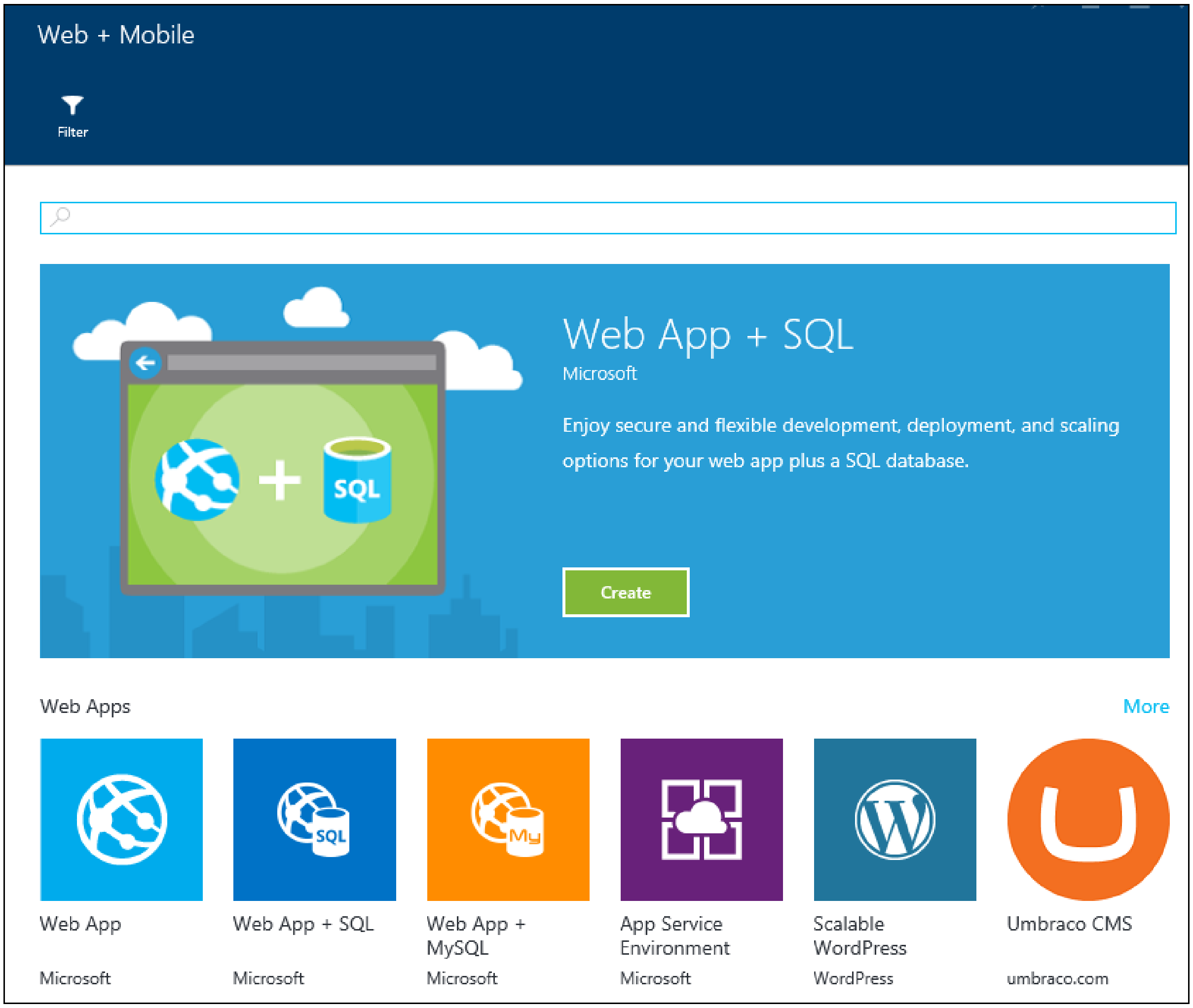
Options for creating Web Apps

There are multiple options for creating a Web App and deploying the content to an app service. Let's look at a few of these, including the following.

- **Azure Marketplace** This contains all of the resources you can deploy in Azure. I'll show you how you can use this to create Web Apps from preexisting templates such as WordPress.
- **Visual Studio Code** This is a free, open source, cross-platform code editor with debugging capabilities.
- **Visual Studio** This is Microsoft's full-featured development IDE.

Marketplace

There are many pre-created websites and templates in the Azure Marketplace that you can use. To see all of the options available, log into the Azure portal and click New > Web + Mobile > See All. This shows the Marketplace blade filtered for Web and Mobile apps, as displayed in Figure 2-7.


Web + Mobile

Filter

Web App + SQL

Microsoft

Enjoy secure and flexible development, deployment, and scaling options for your web app plus a SQL database.

Figure 2-7 Options in the Azure Marketplace for Web and Mobile apps.

If you scroll down on the page, you can see the categories. At the end of any row, clicking More will show additional options in that category. Here are just a few of the choices available:

- **Web Apps** Web App, Web App + SQL, Web App + MySQL, WordPress, and Umbraco CMS
- **Blogs + CMSs** Joomla!, Drupal, DNN, Orchard CMS, Umbraco CMS, and MonoX
- **Starter Web Apps** ASP.NET, HTML5, Node.js, PHP, Apache Tomcat, and some examples like the Bakery web app and the Java Coffee Shop web app

Visual Studio Code

Visual Studio Code (VS Code) is a free, open source code editor with support for development operations such as debugging, task running, and version control. It runs on Windows, OS X, and Linux.

VS Code makes debugging easier, providing IntelliSense code completion and easy code refactoring. It integrates with Git and also package managers, repositories, and various build tools.

VS Code has built-in support for Node.js, JavaScript, and TypeScript. Using extensions, you can use VS Code to debug languages such as C#, C++, Python, Ruby, and PowerShell. There is also tooling for web technologies such as HTML, CSS, JSON, and Markdown.

Using the Azure portal, you can set your web app to get the source code from OneDrive, Dropbox, or a local code repository such as GitHub or Visual Studio Team Service. If you enable continuous deployment for your WebApp, updates will be published automatically when changes are made to your source repository.

You can download Visual Studio Code for Windows, Linux, or Mac here: https://code.visualstudio.com/#alt-downloads.

Visual Studio

Visual Studio is a full development environment, giving you the ability to create many different kinds of applications including, but not limited to, ASP.NET MVC applications, .NET client applications, Windows Communication Foundation (WCF) services, Web APIs, and Cloud Services, using languages such as C#, C++, VB, F#, and XAML.

With Visual Studio, you can create a new web application and publish it to an app service in Azure. I'll show you how to do this in an upcoming demo.

Demo: Create a web app by using the Azure Marketplace

Let's take a look at how to create a web app from one of the templates available in the Azure Marketplace.

1. Log into the Azure portal. As seen in Figure 2-8, click New on the left side of the page, then click See All.

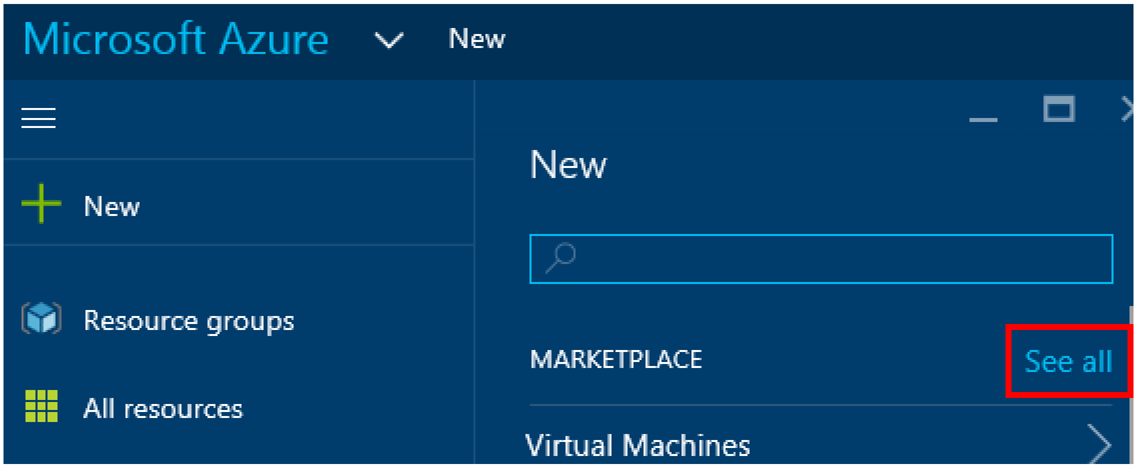
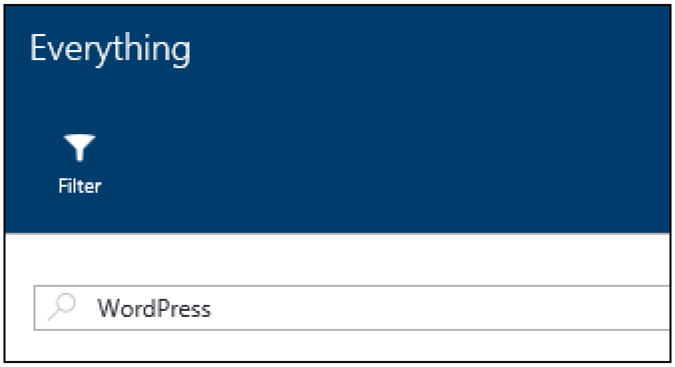


Figure 2-8 Go to the Marketplace Search blade.

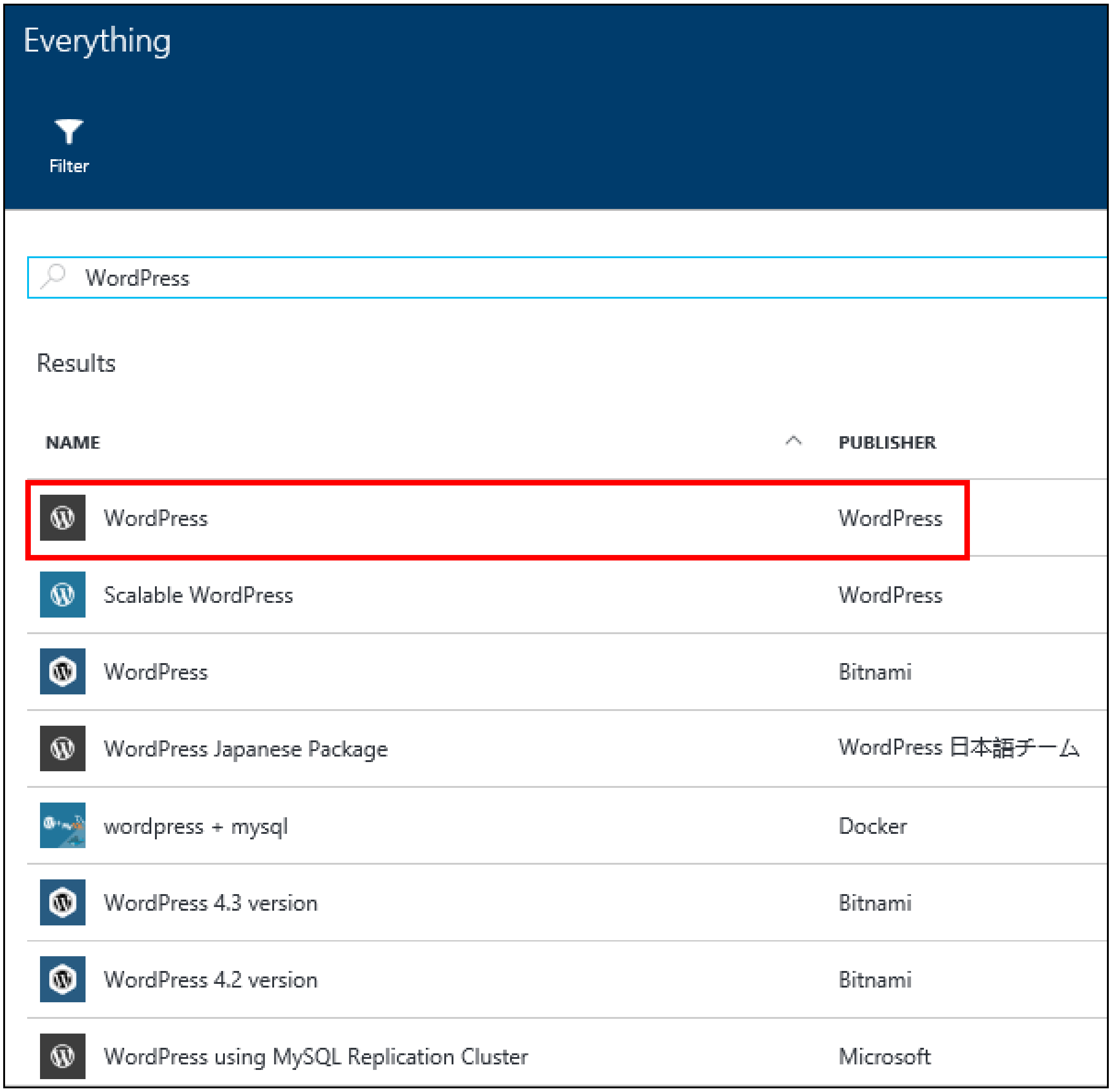
2. This brings up the search screen for the Marketplace. All resources that can be deployed to Azure are listed in the Marketplace, including virtual machines, virtual networks, storage accounts, web apps, and so on. As shown in Figure 2-9, type in **WordPress** and press Enter to perform the search.



WordPress

Figure 2-9 Search for WordPress.

3. You see a list of matches, as displayed in Figure 2-10.



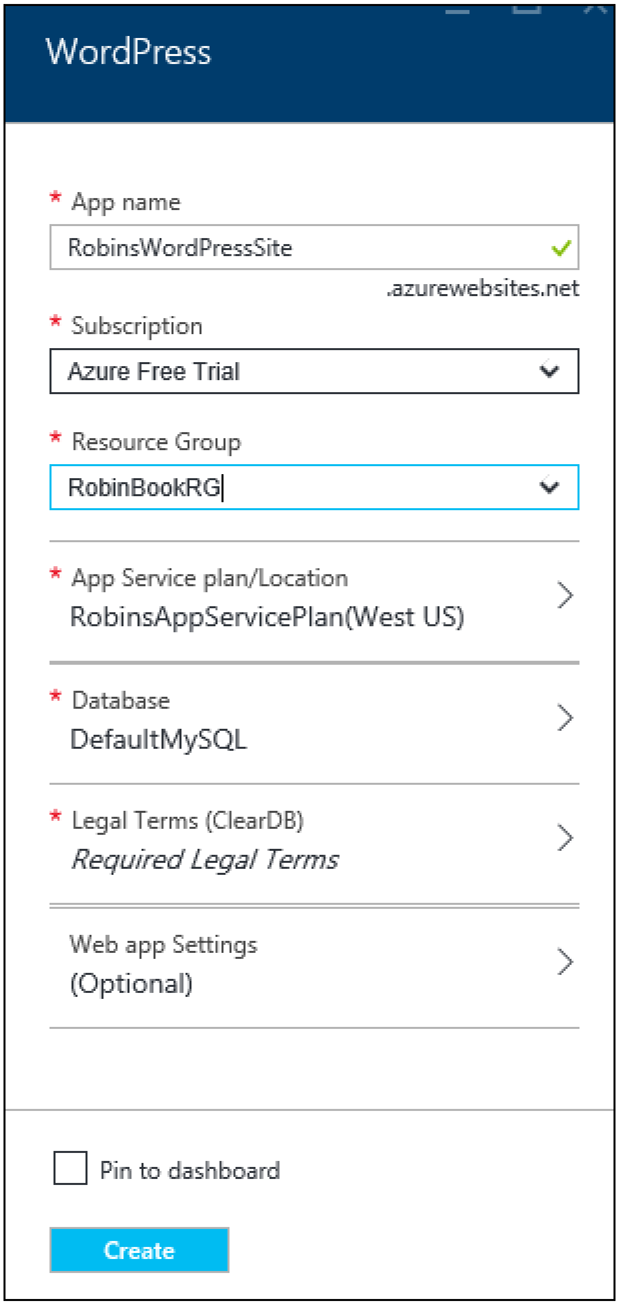
WordPress

Results

NAME	PUBLISHER
WordPress	WordPress
Scalable WordPress	WordPress
WordPress	Bitnami
WordPress Japanese Package	WordPress 日本語チーム
wordpress + mysql	Docker
WordPress 4.3 version	Bitnami
WordPress 4.2 version	Bitnami
WordPress using MySQL Replication Cluster	Microsoft

Figure 2-10 The search results for WordPress.

4. Select the row with WordPress from publisher WordPress. This shows you the blade for WordPress; click Create at the bottom to create a WordPress site. You now see a blade where you can start configuring your WordPress site, as displayed in Figure 2-11.



WordPress

* App name

RobinsWordPressSite ✓

.azurewebsites.net

* Subscription

Azure Free Trial

* Resource Group

RobinBookRG

* App Service plan/Location

RobinsAppServicePlan(West US)

* Database

DefaultMySQL

* Legal Terms (ClearDB)

Required Legal Terms

Web app Settings

(Optional)

□ Pin to dashboard

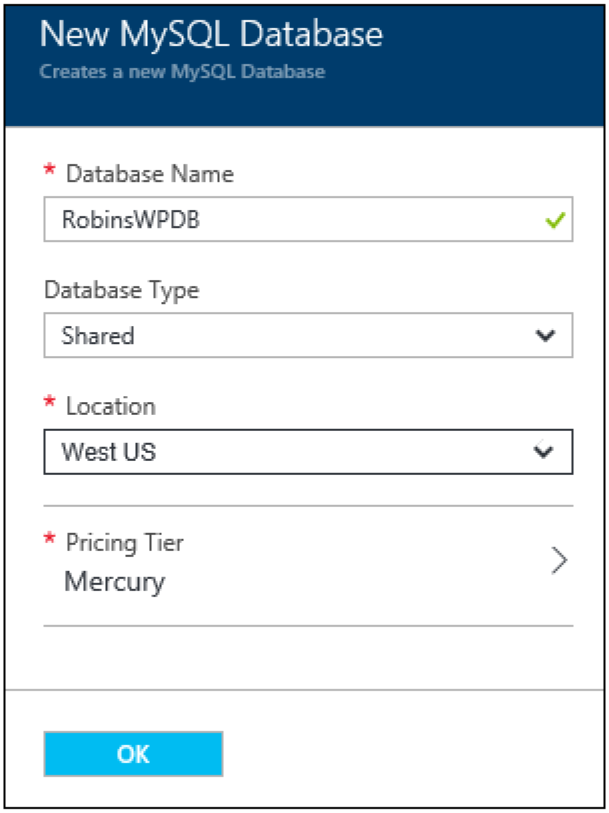
Create

Figure 2-11 Create a WordPress website.

5. Now, start filling in the fields on this blade:

- **App Name** This is used to create the URL to access your web app.
- **Subscription** If the account you are using is associated with multiple subscriptions, select the subscription you want to use.

- **Resource Group** This is a way of grouping multiple resources that are related to one another, such as a web app and a database. Select the resource group you used for the App Service plan you created earlier.
- **App Service Plan** Select the App Service plan you created earlier in this chapter.
- Click Database to see the database settings, as shown in Figure 2-12. WordPress uses MySQL by default. Set your Database Name and Type (Shared or Dedicated). For Location, select the same region in which your app is going to run. Click Pricing Tier and select the least expensive, which at this time is Mercury. Click OK to save the database settings.



New MySQL Database

Creates a new MySQL Database

* Database Name: RobinsWPDB ✓

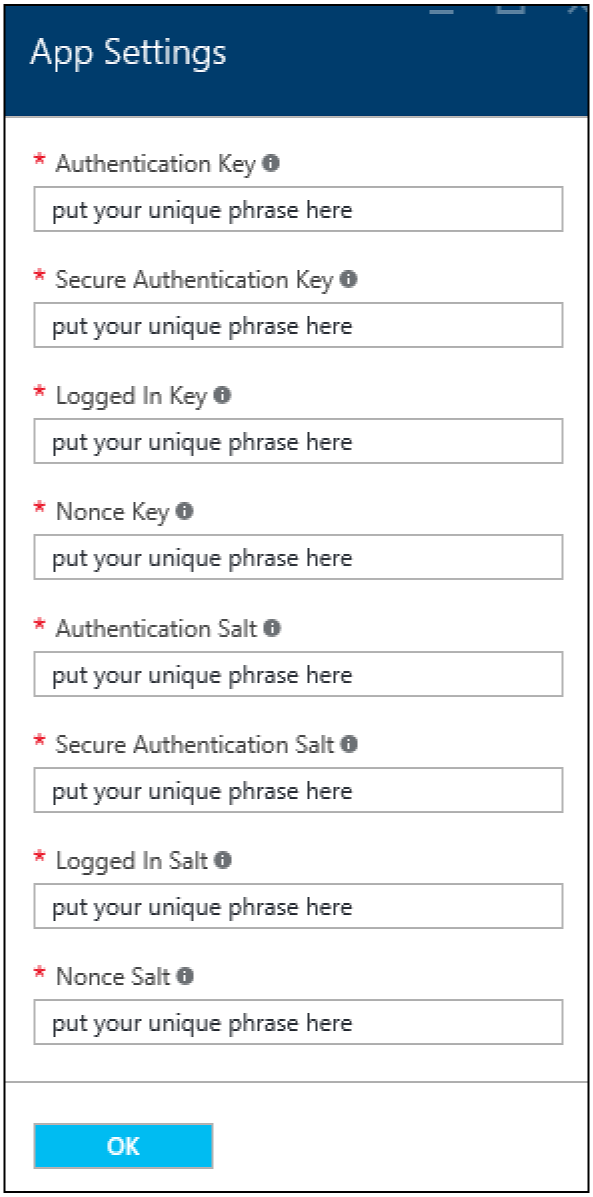
Database Type: Shared

* Location: West US

* Pricing Tier: Mercury

Figure 2-12 Specify database settings.

- Back on the WordPress Settings blade for your new website, click Legal Terms. If you agree with the Legal Terms, click OK at the bottom of that screen, which will set Legal Terms to Accepted.
- You can use Web App Settings (Optional) to set the WordPress-specific settings shown in Figure 2-13; this is optional.



App Settings

* Authentication Key
put your unique phrase here

* Secure Authentication Key
put your unique phrase here

* Logged In Key
put your unique phrase here

* Nonce Key
put your unique phrase here

* Authentication Salt
put your unique phrase here

* Secure Authentication Salt
put your unique phrase here

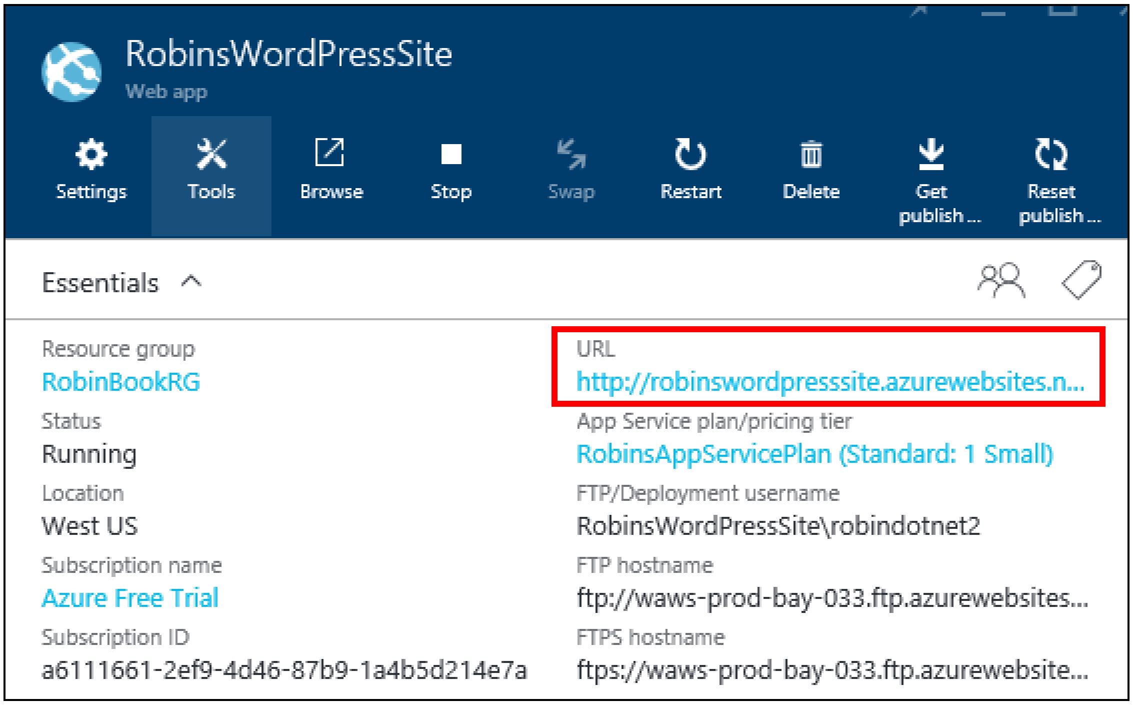
* Logged In Salt
put your unique phrase here

* Nonce Salt
put your unique phrase here

Figure 2-13 Fill in App Settings (optional).

- Back on the WordPress blade, select the check box to pin the web app to your Dashboard, then click Create. Azure will create the WordPress site for you.

6. After Azure has finished publishing the web app, click the tile on your Dashboard to open its properties, as displayed in Figure 2-14. To open the site, click the URL. You are prompted for the rest of the details needed to create your WordPress site, such as language, site title, username, password, and email address. After all the fields are filled in, click the Install WordPress button. After the WordPress installation is finished, you're ready to go.



Resource group
RobinBookRG

Status
Running

Location
West US

Subscription name
Azure Free Trial

Subscription ID
a6111661-2ef9-4d46-87b9-1a4b5d214e7a

URL
http://robinswordpresssite.azurewebsites.n...

App Service plan/pricing tier
RobinsAppServicePlan (Standard: 1 Small)

FTP/Deployment username
RobinsWordPressSite\robindotnet2

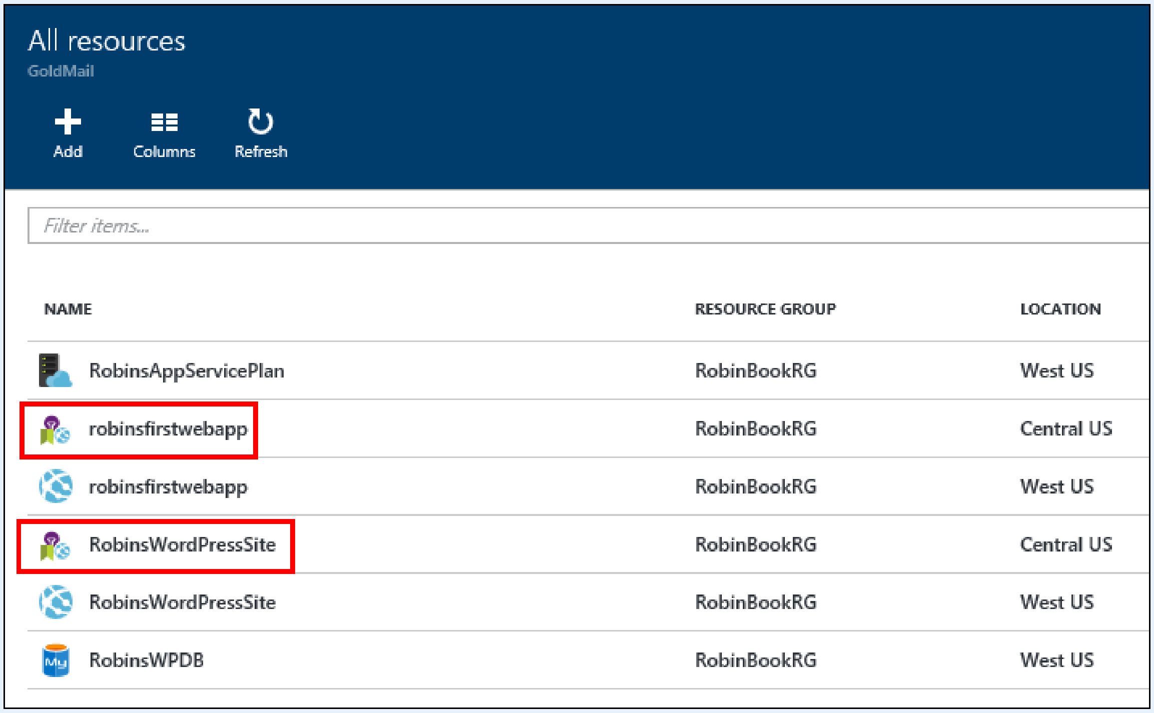
FTP hostname
ftp://waws-prod-bay-033.ftp.azurewebsites...

FTPS hostname
ftps://waws-prod-bay-033.ftp.azurewebsite...

Figure 2-14 Open your new WordPress site by clicking its URL.

Note When your web app is created, Azure also creates an Application Insights instance. Application Insights is an analytics service that monitors your live application. It can help you resolve performance issues and understand how your application is used. Application Insights is outside the scope of this book. For more information, check out the Getting Started article about Application Insights: https://azure.microsoft.com/documentation/articles/app-insights-overview/.

You can see the Application Insights instances in the All Resources blade; it will have the same name as your web app, but it will be a different resource type. My list of resources is displayed in Figure 2-15; the ones with the rectangle around them are the Application Insights instances. Note that they have a different icon from the Web Apps. Simply select those Application Insights resources and delete them. (When you select that resource, it will open a bunch of blades. Just close them until you get back to the first one, and select Delete from that blade.)



NAME	RESOURCE GROUP	LOCATION
RobinsAppServicePlan	RobinBookRG	West US
robinsfirstwebapp	RobinBookRG	Central US
robinsfirstwebapp	RobinBookRG	West US
RobinsWordPressSite	RobinBookRG	Central US
RobinsWordPressSite	RobinBookRG	West US
RobinsWPDB	RobinBookRG	West US

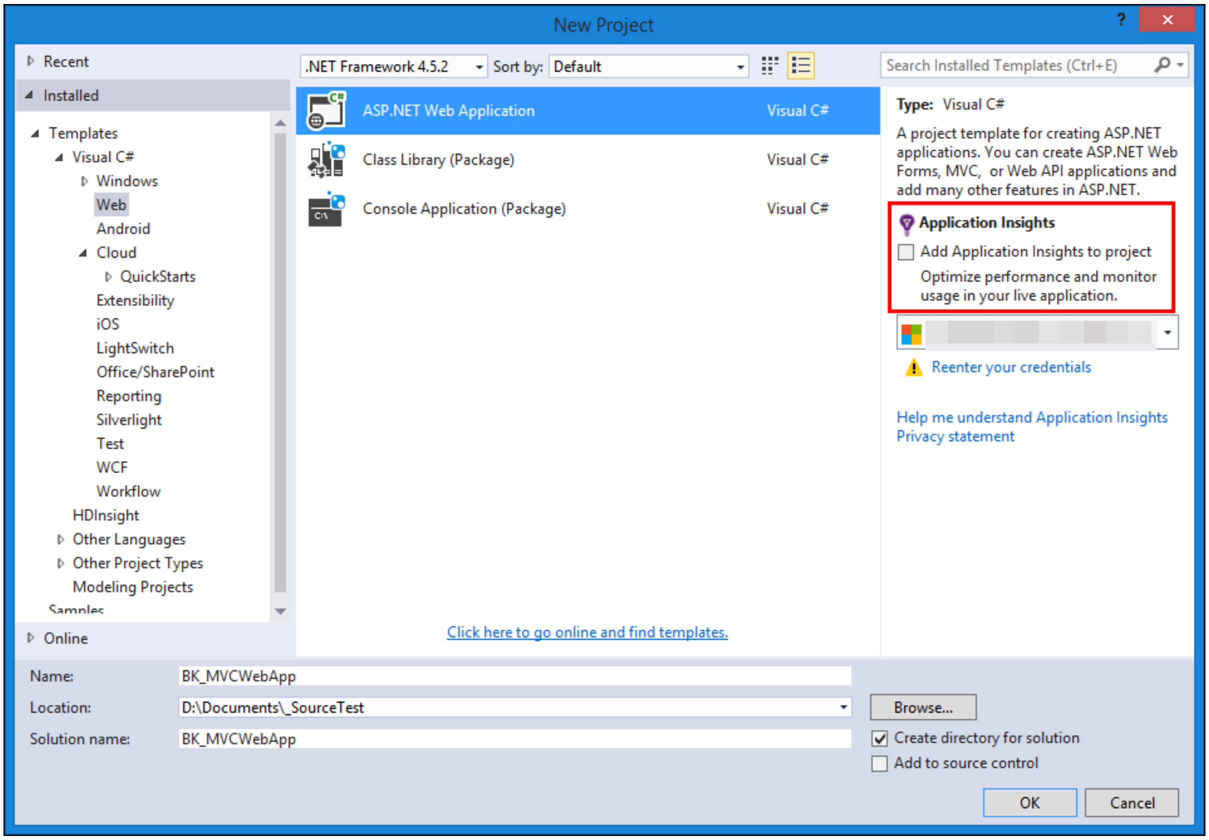
Figure 2-15 The Application Insights instances are created automatically when you create a web app.

Demo: Create an ASP.NET website in Visual Studio and deploy it as a web app

To perform this tutorial, you must have Visual Studio 2013 or Visual Studio 2015 installed and the most recent version of the Azure Tools and SDK.

Create a new web application with Visual Studio by following these steps:

1. Open Visual Studio. Select File > New > Project.
2. Select ASP.NET Web Application; the dialog box for creating a project appears, as shown in Figure 2-16. On the right side of the dialog box, clear the Add Application Insights To Project check box. This will prevent the creation of a separate Application Insights instance for this web application.



.NET Framework 4.5.2

Sort by: Default

Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Application Insights

□ Add Application Insights to project

Optimize performance and monitor usage in your live application.

Name: BK_MVCWebApp

Location: D:\Documents_SourceTest

Solution name: BK_MVCWebApp

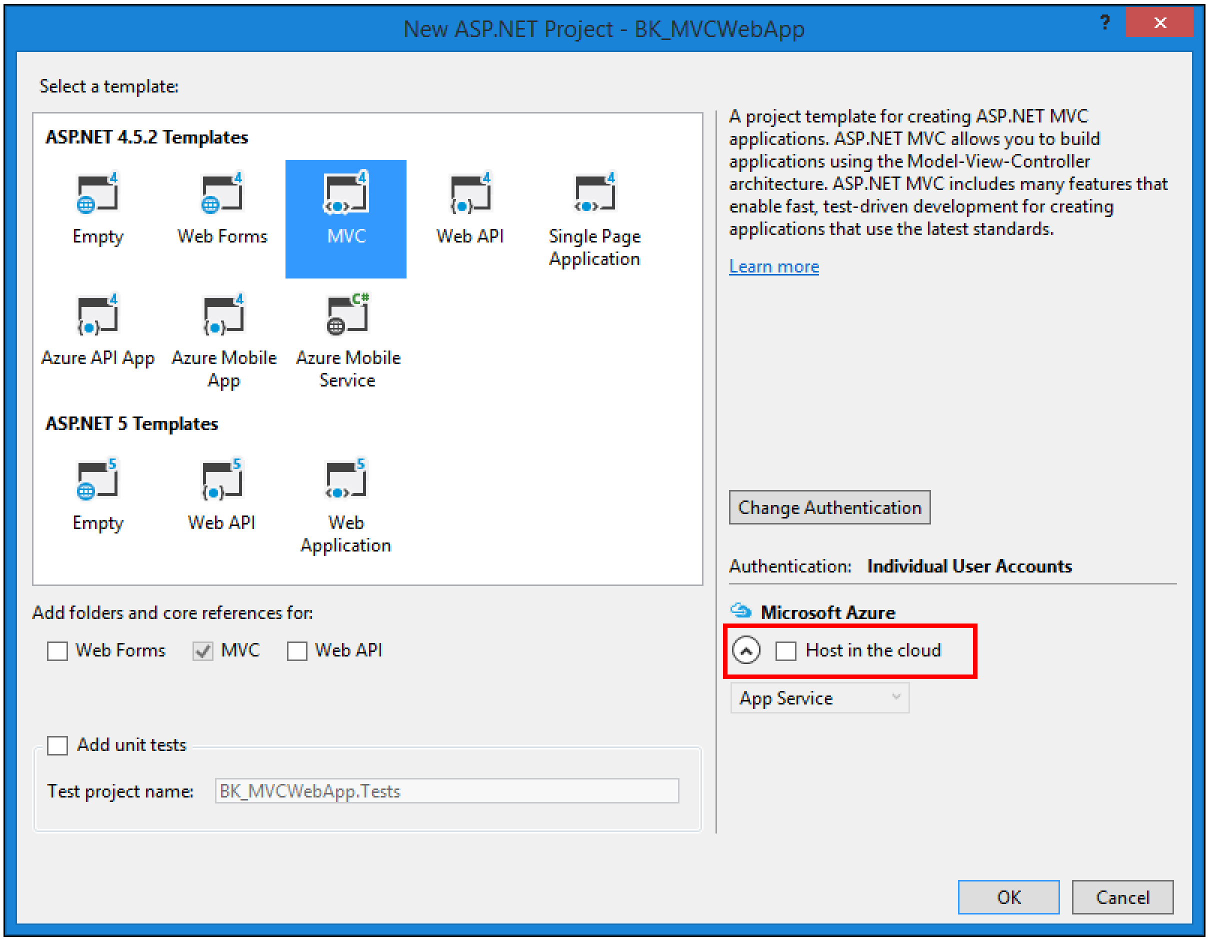
☑ Create directory for solution

□ Add to source control

Figure 2-16 Create an ASP.NET Web Application; deselect Application Insights.

3. Specify the Name of the application and the Location for the solution, then click OK.

4. When prompted to select the type of ASP.NET application to create, select MVC from the list of ASP.NET Templates, as shown in Figure 2-17. Clear the Host In The Cloud check box. You will set that up separately. Click OK to continue.



A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Authentication: **Individual User Accounts**

Add folders and core references for:

□ Web Forms ☑ MVC □ Web API

□ Add unit tests

Test project name: BK_MVCWebApp.Tests

□ Host in the cloud

Figure 2-17 Select an MVC application and clear the Host In The Cloud check box.

5. Visual Studio will create a basic ASP.NET MVC application that runs "as is." You can modify it later to make it your own.

6. Now, publish this web application to an App Service in Azure and assign it to the App Service plan created earlier in this chapter. You will create the App Service when you publish the web app the first time. Right-click the website and select Publish (Figure 2-18).

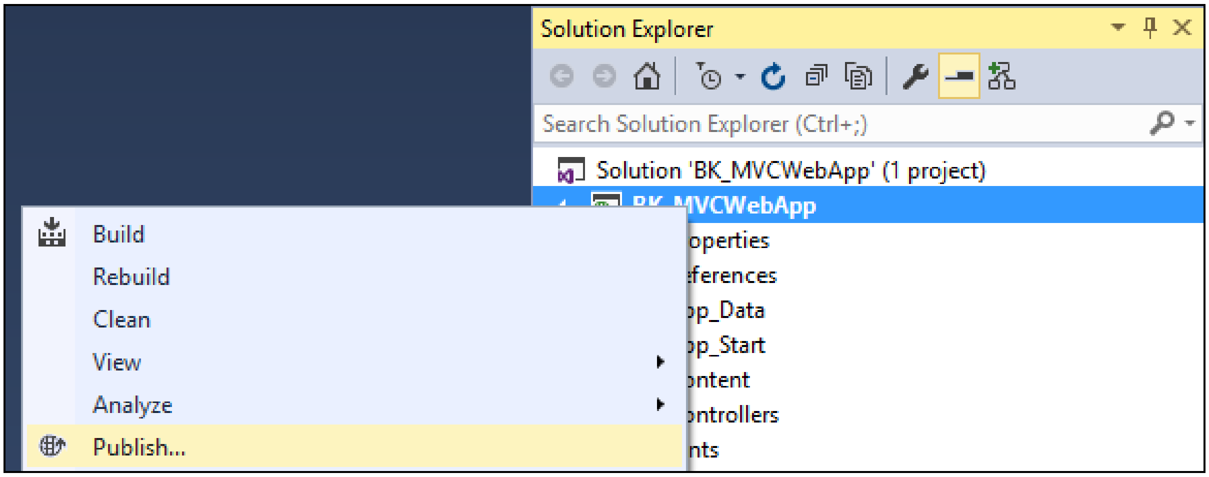


Figure 2-18 Step 1 for publishing the web application.

7. The Publish Web dialog box will be displayed. Select the Microsoft Azure App Service (Figure 2-19).

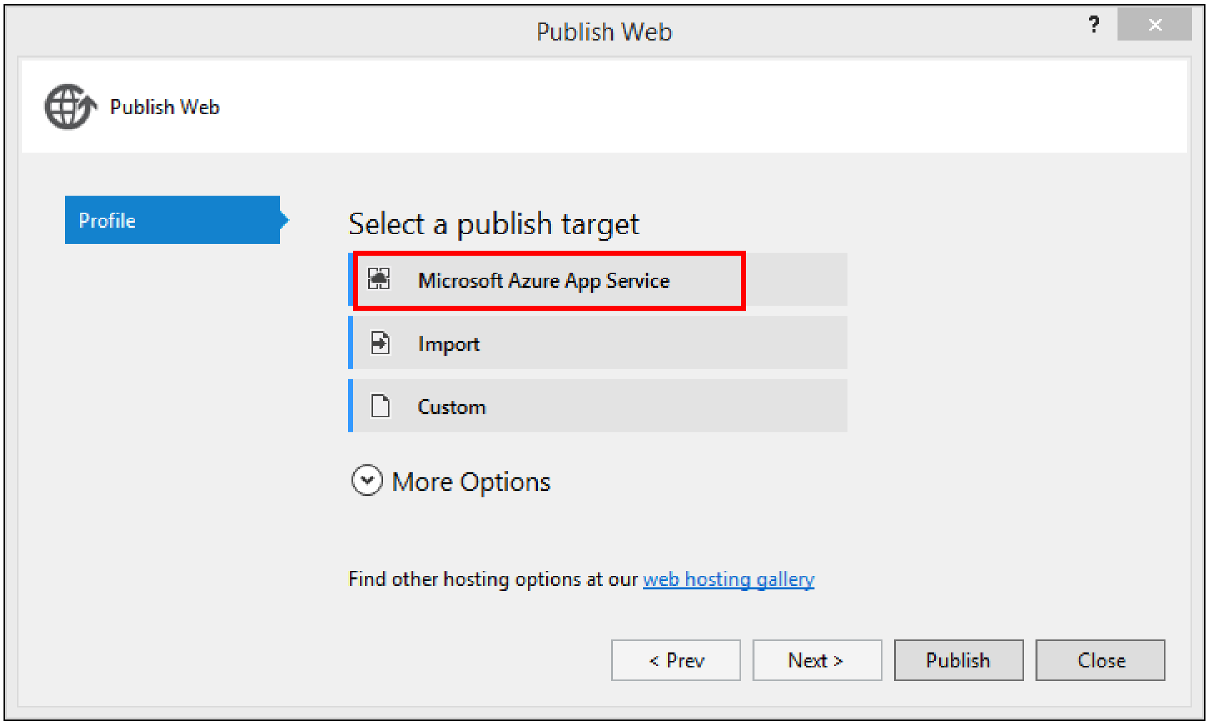
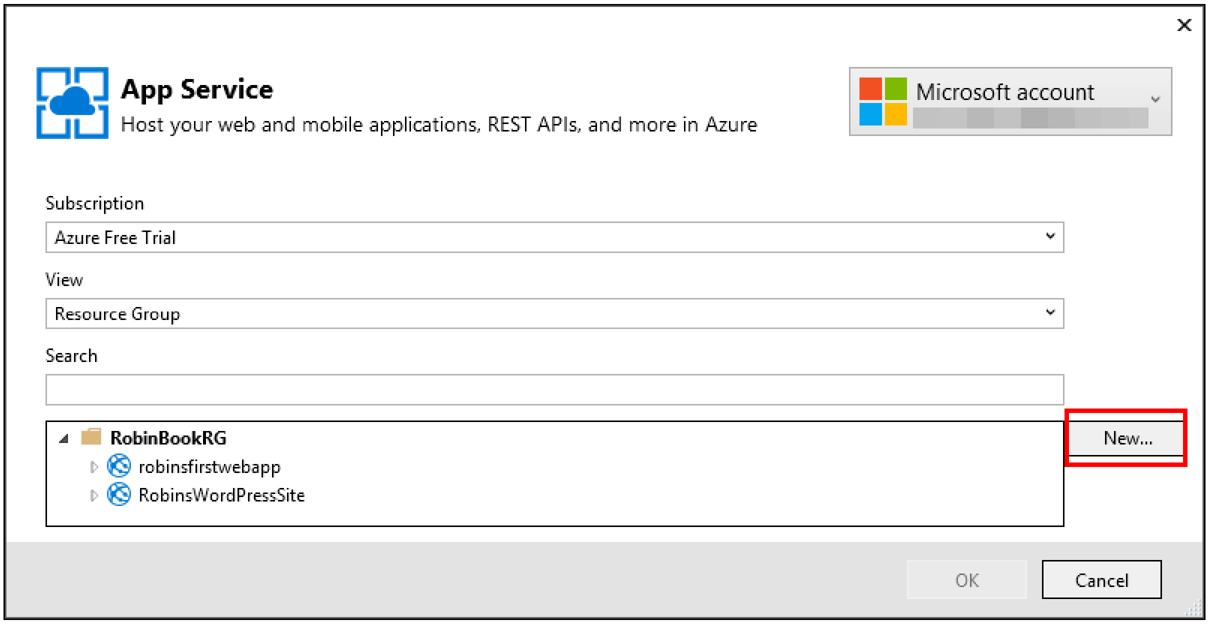


Figure 2-19 Select the Microsoft Azure App Service for the publish target.

8. You will be prompted for your subscription name. You may be prompted again to enter the credentials for your Azure subscription. If the correct account is not displayed, click it to show a drop-down list and add an account if necessary. When the correct account is selected, select the Subscription and be sure the View is set to Resource Group. Open the Resource Group, and you will see the resources that have been set up already. In Figure 2-20, you can see the web apps that I have already created. To publish this application to a new web app, click New.



App Service

Host your web and mobile applications, REST APIs, and more in Azure

Microsoft account

Subscription: Azure Free Trial

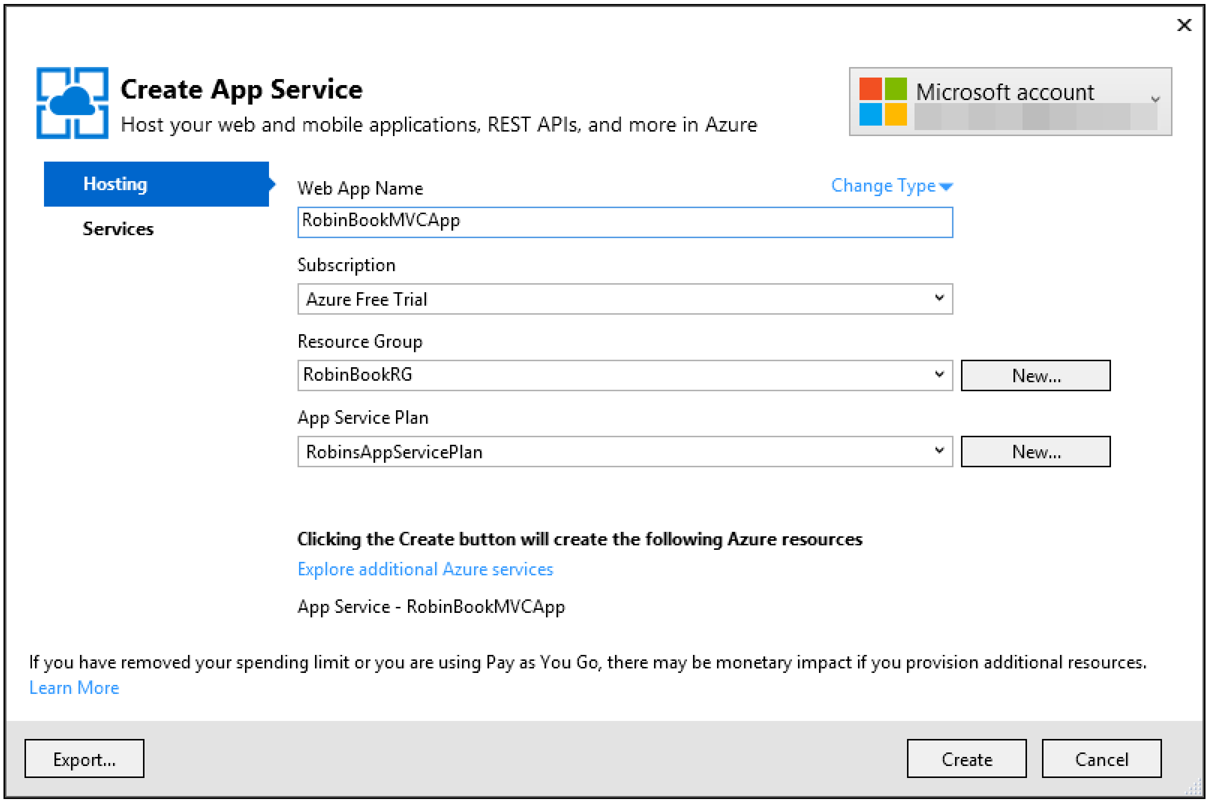
View: Resource Group

Search:

- RobinBookRG
  - robinsfirstwebapp
  - RobinsWordPressSite

Figure 2-20 Make sure the right account and subscription are selected; show the resources by group.

9. The Create App Service dialog box (Figure 2-21) appears next. Remember that an App Service is simply the host for a Web App, Mobile App, Logic App, API App, or Function App. You'll create a new App Service to host your MVC web application here.



Create App Service

Host your web and mobile applications, REST APIs, and more in Azure

Microsoft account

Web App Name: RobinBookMVCApp

Subscription: Azure Free Trial

Resource Group: RobinBookRG

App Service Plan: RobinsAppServicePlan

Clicking the Create button will create the following Azure resources

Explore additional Azure services

App Service - RobinBookMVCApp

If you have removed your spending limit or you are using Pay as You Go, there may be monetary impact if you provision additional resources.
Learn More

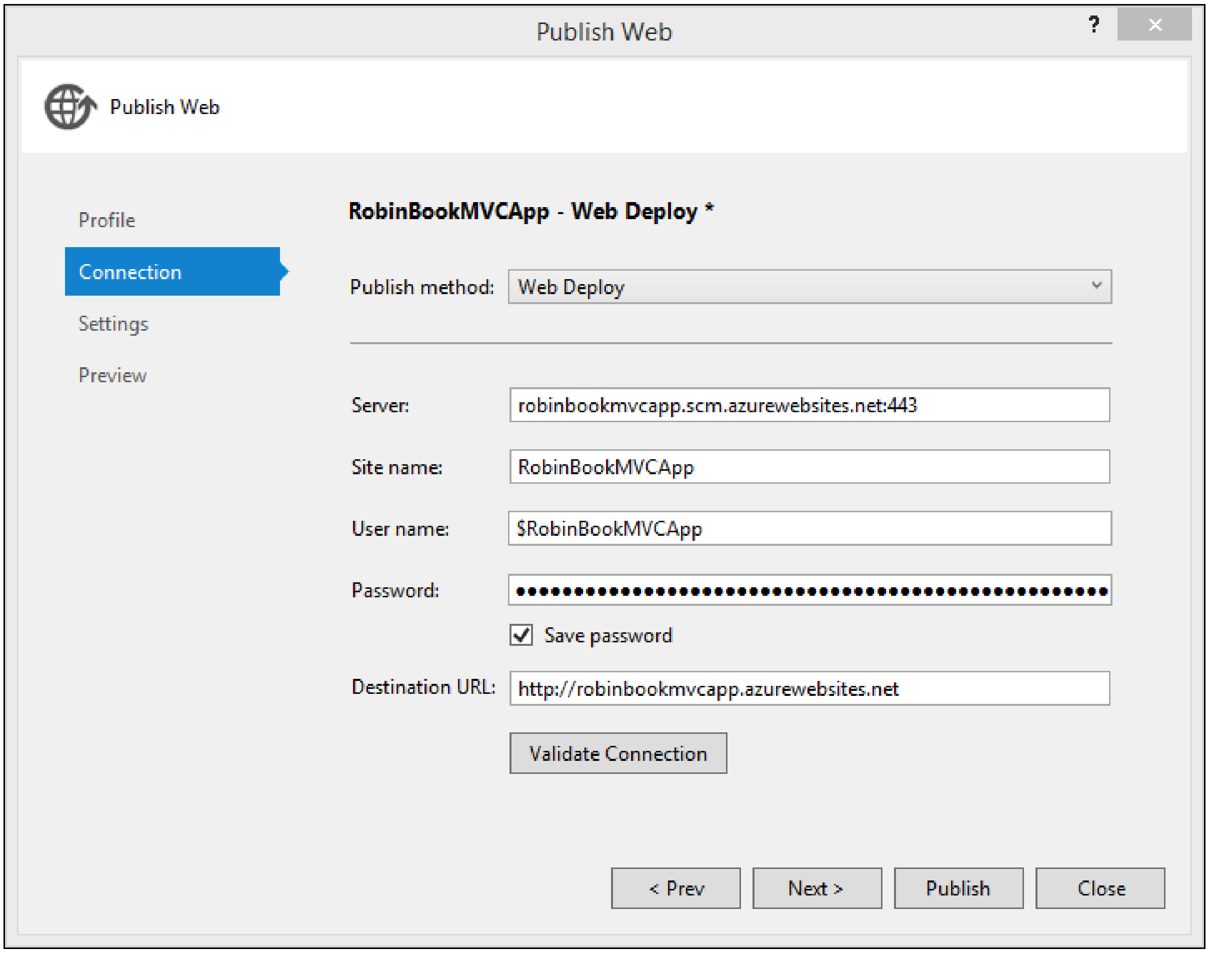
Figure 2-21 Create an App Service to host the MVC application.

- Set the Web App Name. This will be used for the URL for the web app, so select it wisely.
- Select the Subscription.
- Select the Resource Group. If you use the one you created at the beginning of this chapter, then when you're done, you can delete that Resource Group and all of your resources will be removed.
- Last, select the App Service plan that you created earlier in this chapter. This application will be hosted on the same VMs as the other web app(s) you have placed in that plan.

Click Create to create the App Service.

If you look in the Azure portal now, you will see your App Service has been created.

10. Now let's use Web Deploy to publish our web app to our app service. After creating the app service, the Publish Web dialog box will be displayed (Figure 2-22). You can use the default values.



RobinBookMVCApp - Web Deploy *

Publish method: Web Deploy

Server: robinbookmvcapp.scm.azurewebsites.net:443

Site name: RobinBookMVCApp

User name: $RobinBookMVCApp

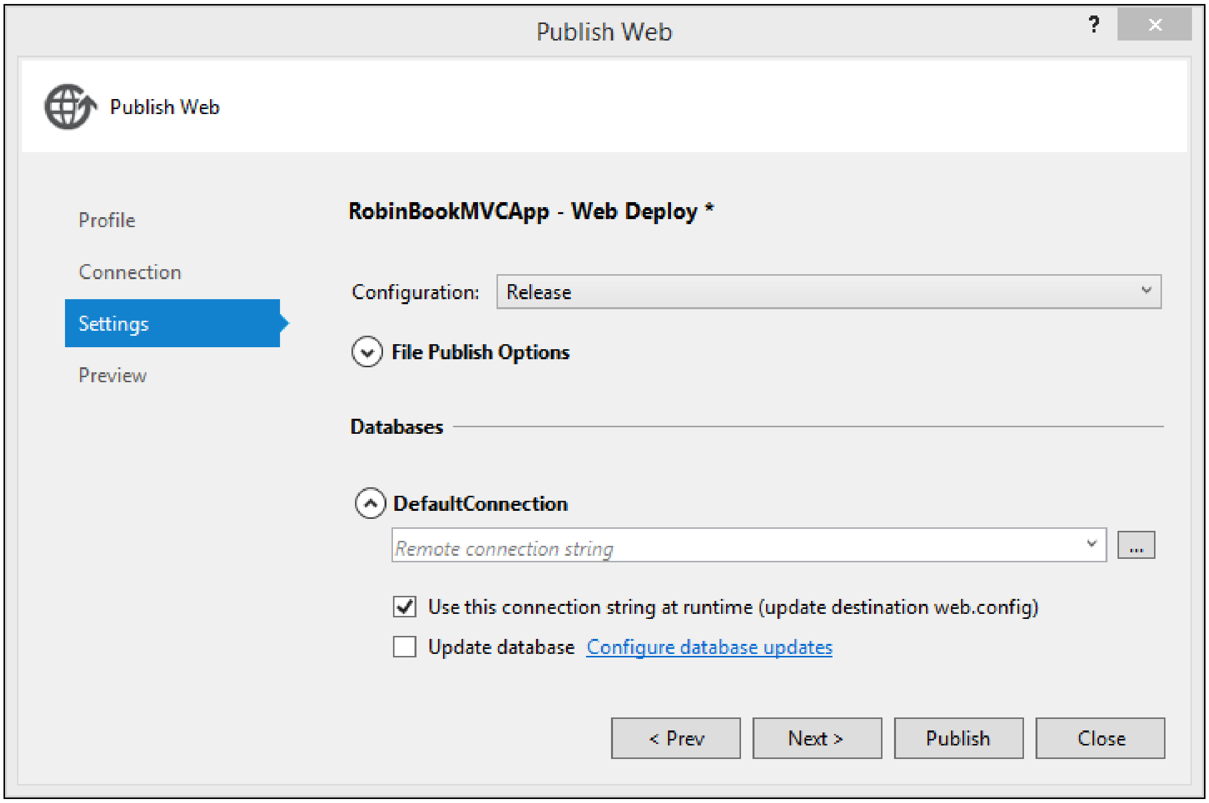
Password: ●●

☑ Save password

Destination URL: http://robinbookmvcapp.azurewebsites.net

Figure 2-22 Publish settings for the MVC application.

11. Click Validate Connection to make sure the information is correct. After it validates, click Next to go to the next dialog box (Figure 2-23).



RobinBookMVCApp - Web Deploy *

Configuration: Release

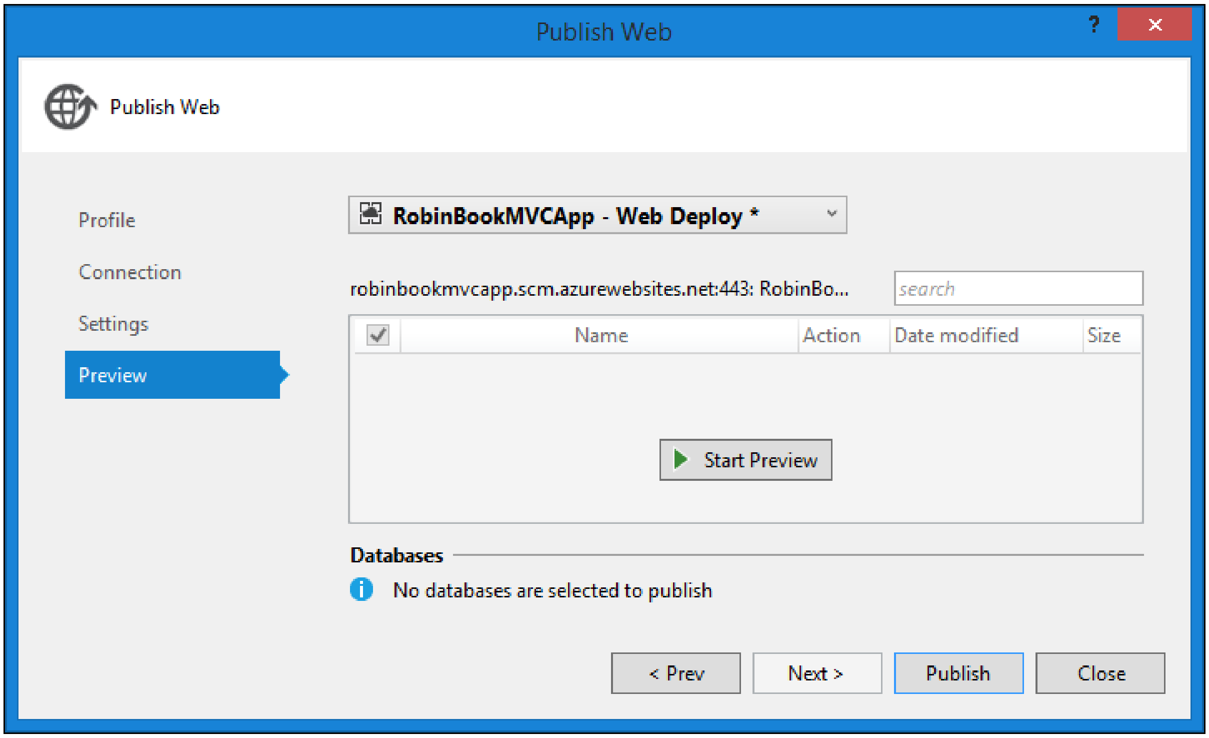
Remote connection string

☑ Use this connection string at runtime (update destination web.config)

□ Update database Configure database updates

Figure 2-23 Settings used when publishing the MVC application.

12. This dialog box lets you set the Configuration to Debug or Release and provide a connection string to a database if needed. Note that if you are going to use remote debugging on your web app, you will want to select the Debug configuration. Click Next to reach the final page (Figure 2-24).



RobinBookMVCApp - Web Deploy *

robinbookmvcapp.scm.azurewebsites.net:443: RobinBo...

search

No databases are selected to publish

Figure 2-24 Publish the MVC application.

13. You can preview your site here. When you're finished, click Publish to deploy the web application to the App Service. It will open your web application in the default browser after it is published.

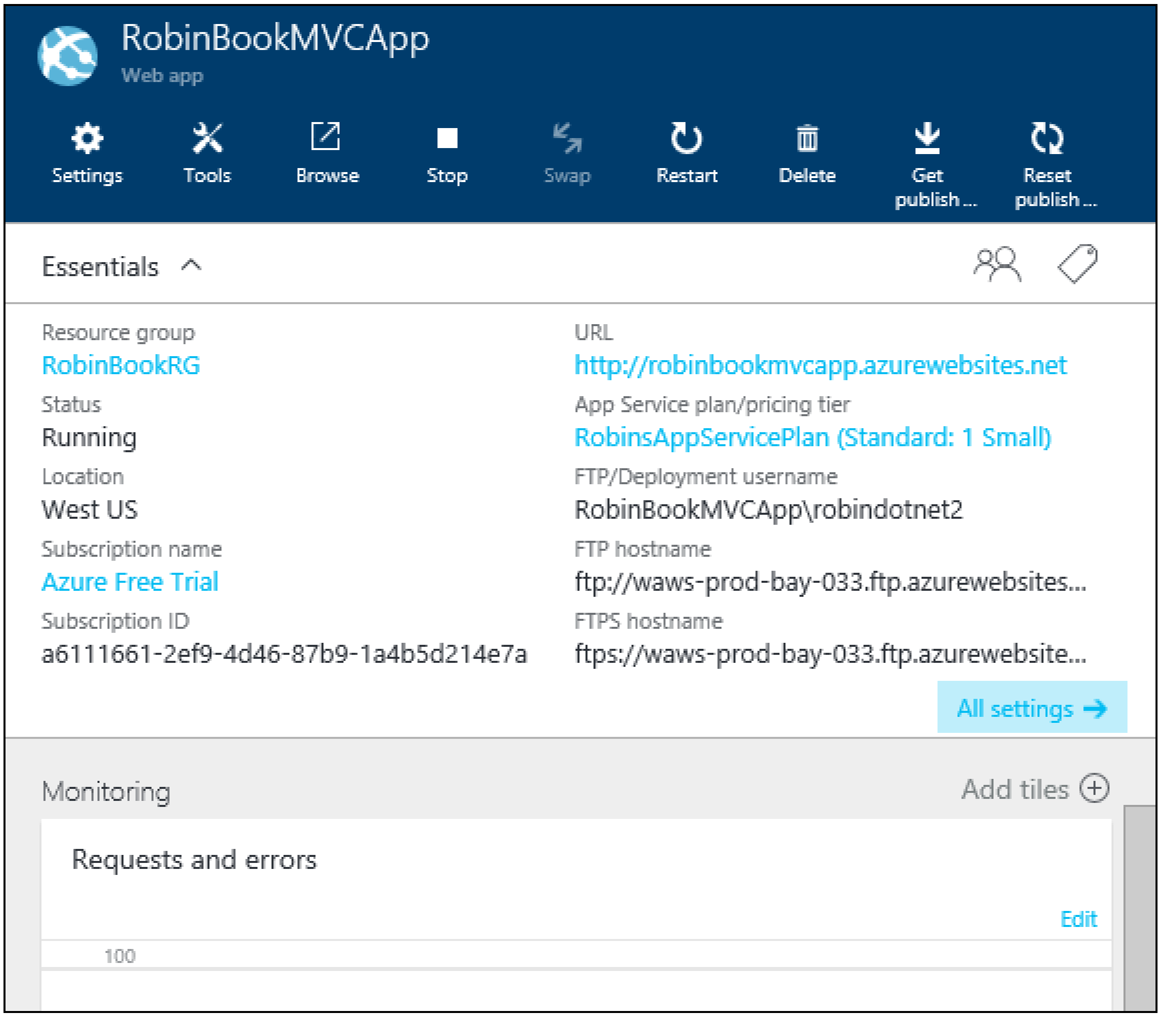
When you make changes to your website, you can go through this same process to publish the website again. Note that it will only publish the files that have been added or modified.

Configuring, scaling, and monitoring Web Apps

Now that you've created a web app, assigned it to an App Service plan, and deployed it, let's take a look at the configuration in the portal and how to scale your web application.

Configuring Web Apps

Log into the Azure portal and go to the web application you created and deployed from Visual Studio earlier. The primary blade should look like Figure 2-25.



Resource group
RobinBookRG

Status
Running

Location
West US

Subscription name
Azure Free Trial

Subscription ID
a6111661-2ef9-4d46-87b9-1a4b5d214e7a

URL
http://robinbookmvcapp.azurewebsites.net

App Service plan/pricing tier
RobinsAppServicePlan (Standard: 1 Small)

FTP/Deployment username
RobinBookMVCApp\robindotnet2

FTP hostname
ftp://waws-prod-bay-033.ftp.azurewebsites...

FTPS hostname
ftps://waws-prod-bay-033.ftp.azurewebsite...

Figure 2-25 Web App blade.

The Essentials section

Let's start with the icons across the top of the Web App blade and look at what they are used for.

- **Settings** This opens a new blade called Settings. This displays by default when you first open the Web App blade, and is the same blade you see when you click All Settings.
- **Tools** This opens the Tools blade, which provides access to Performance testing, Process Explorer, Performance monitoring, and so on. It also provides access to the Kudu console, which is helpful for troubleshooting and analysis.
- **Browse** This opens your web app in your default browser.
- **Stop/Start** This option starts and stops the web app.
- **Swap** This option swaps the versions deployed to two different deployment slots. For example, if you have a production slot and a staging slot, you can publish your web app to staging and test it. When you're satisfied with it, you can promote it to production by using the Swap option. When you're sure everything is working okay, you can remove the staging version.
- **Restart** This restarts your web app.

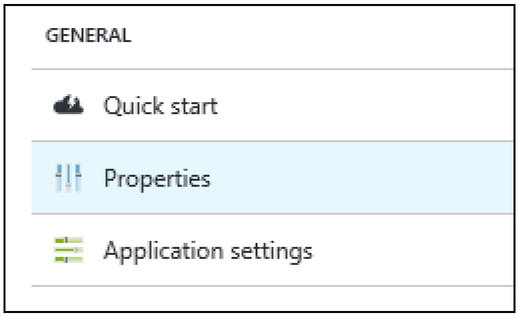
- **Delete** This removes the web app.
- **Get Publish Profile** This retrieves the information needed to publish a web app from Visual Studio.
- **Reset Publish Profile** This resets the publishing credentials and invalidates the old credentials. These credentials are used for FTP and Git access.

In the Essentials area, it shows the settings provided when creating the web app: the Resource Group, Location, Azure Subscription ID, the URL of the website, and the name of the App Service plan being used. It also shows the credentials for FTP'ing into the web app in case you want to deploy new files via FTP.

Click Settings to open the Settings blade. Let's take a closer look at some of the options on this blade.

The Settings blade: General

Figure 2-26 shows the General section of the Settings blade.



GENERAL

Quick start

Properties

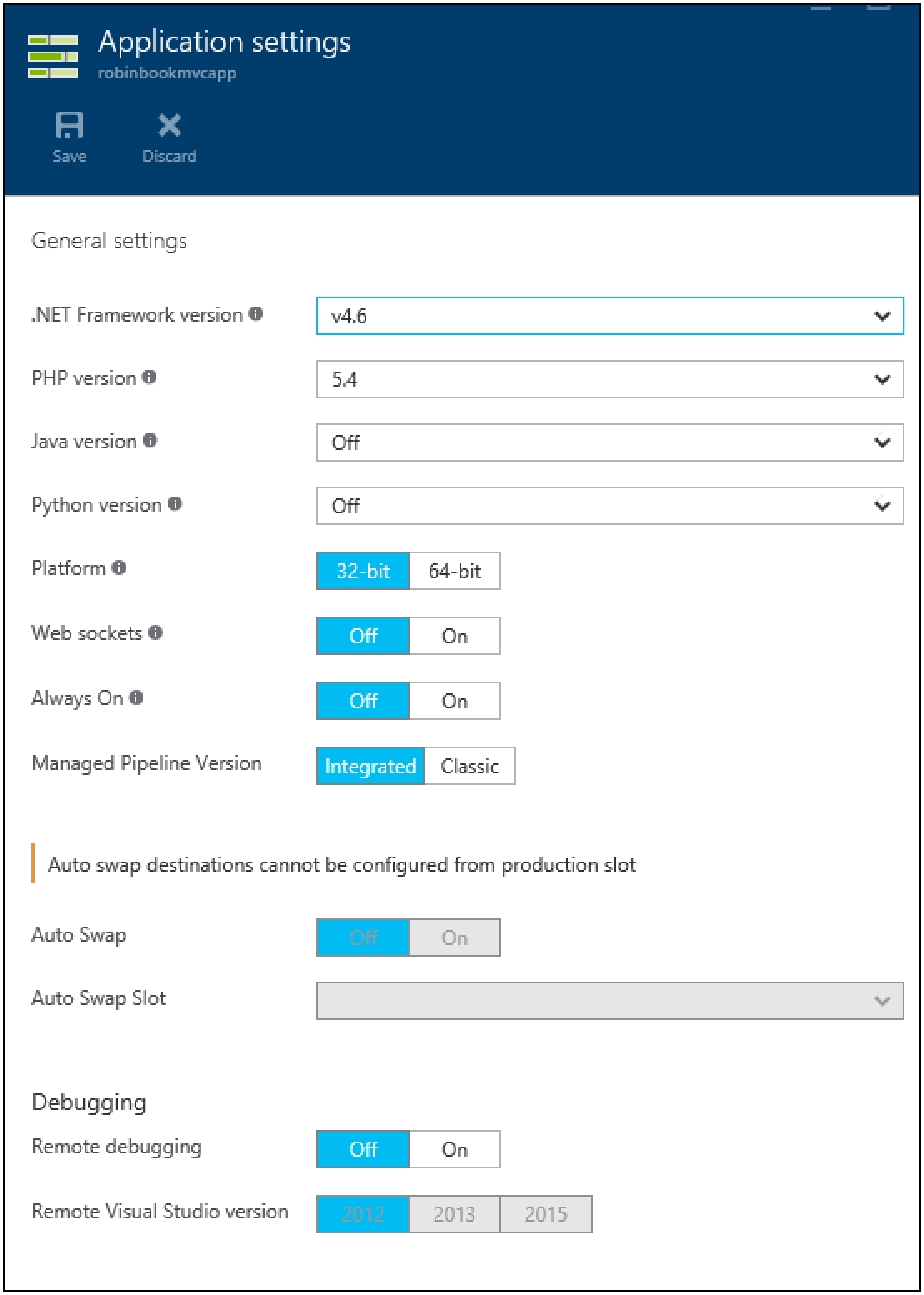
Application settings

Figure 2-26 General section on the web app's Settings blade.

Let's take a look at the General settings we can configure on this blade.

- **Quick Start** This brings up some resources you can use to learn more about Web Apps. There are links to install Visual Studio and the Microsoft Azure SDK, links to reset your deployment credentials, and links to tutorials, forums, samples, etc.
- **Properties** This shows some of the same values that are in the Essentials blade: the URL, the mode (Standard), the outbound IP addresses, the FTP settings, and so on.
- **Application Settings** These are values that apply to your web app.

The top of the Application Settings blade shown in Figure 2-27 lets you set things like the .NET Framework version, PHP version, etc.



Application settings

robinbookmvcapp

General settings

.NET Framework version: v4.6

PHP version: 5.4

Java version: Off

Python version: Off

Platform: [✓ 32-bit] 64-bit

Web sockets: [✓ Off] On

Always On: [✓ Off] On

Managed Pipeline Version: [✓ Integrated] Classic

Auto swap destinations cannot be configured from production slot

Auto Swap: [✓ Off] On

Auto Swap Slot:

Debugging

Remote debugging: [✓ Off] On

Remote Visual Studio version: [✓ 2012] 2013 2015

Figure 2-27 Application Settings blade for the web app.

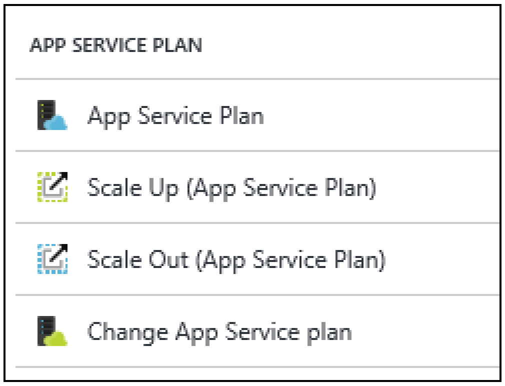
Let's look at what some of these settings are used for:

- **.NET Framework Version** If your web app is a .NET application, this will denote the major version being used. Values available are 3.5 and 4.6.
- **PHP, Java, and Python Versions** If using one of these technologies, this allows you to set the version to be run for the App Service. PHP 5.4, 5.5, 5.6, and 7.0 are supported. Java 7 and 8 are supported. For Python, versions 2.7 and 3.4 are supported.
- **Platform** This indicates whether your web app runs on a 32-bit platform or a 64-bit platform. Note that you cannot select 32-bit for Free websites.
- **Always On** By default, webpages are unloaded after being idle for a certain amount of time. If you need your webpage to be live and active all of the time, set this to On.
- **Debugging** These settings allow you enable and disable remote debugging. If set to On, you can then select which version of Visual Studio you want to use to perform the debugging. Be sure to specify the Debug configuration when you publish your web app if you want to perform remote debugging.

Other settings farther down this blade include the list of default documents, handler mappings, and virtual applications and directories.

The Settings blade: App Service plan

This is the App Service Plan section of the Settings blade (Figure 2-28).



APP SERVICE PLAN

App Service Plan

Scale Up (App Service Plan)

Scale Out (App Service Plan)

Change App Service plan

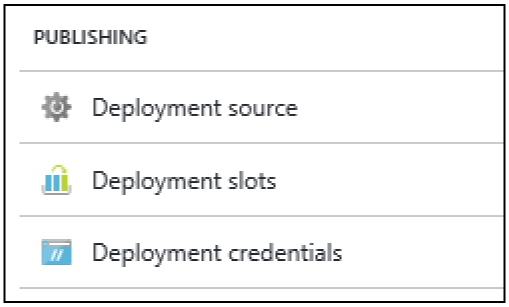
Figure 2-28 App Service Plan section on the web app's Settings blade.

These are the App Service plan settings you can configure on this blade.

- **App Service Plan** This shows which App Service plan is used by the web app. This will show the settings for that App Service plan, which are the same values you see if you choose your App Service plan from All Resources on the main menu of the Azure portal.
- **Scale Up (App Service Plan)** This lets you change the pricing tier for the plan. Each pricing tier provides different values for the number of cores, amount of memory, amount of storage, and so on.
- **Scale Out (App Service Plan)** This is where you can set up autoscaling for your App Service plan and all of its app services. For example, you can ask it to increase the number of VMs if your CPU percentage reaches 90 percent and stays there for X number of minutes. We'll take a closer look at this in the "Scaling Web Apps" section later in this chapter.
- **Change App Service Plan** This enables you to select a different App Service plan or create a new one.

The Settings blade: Publishing

Figure 2-29 shows the Publishing section of the Settings blade for a web app.



PUBLISHING

Deployment source

Deployment slots

Deployment credentials

Figure 2-29 Publishing section on the web app's Settings blade.

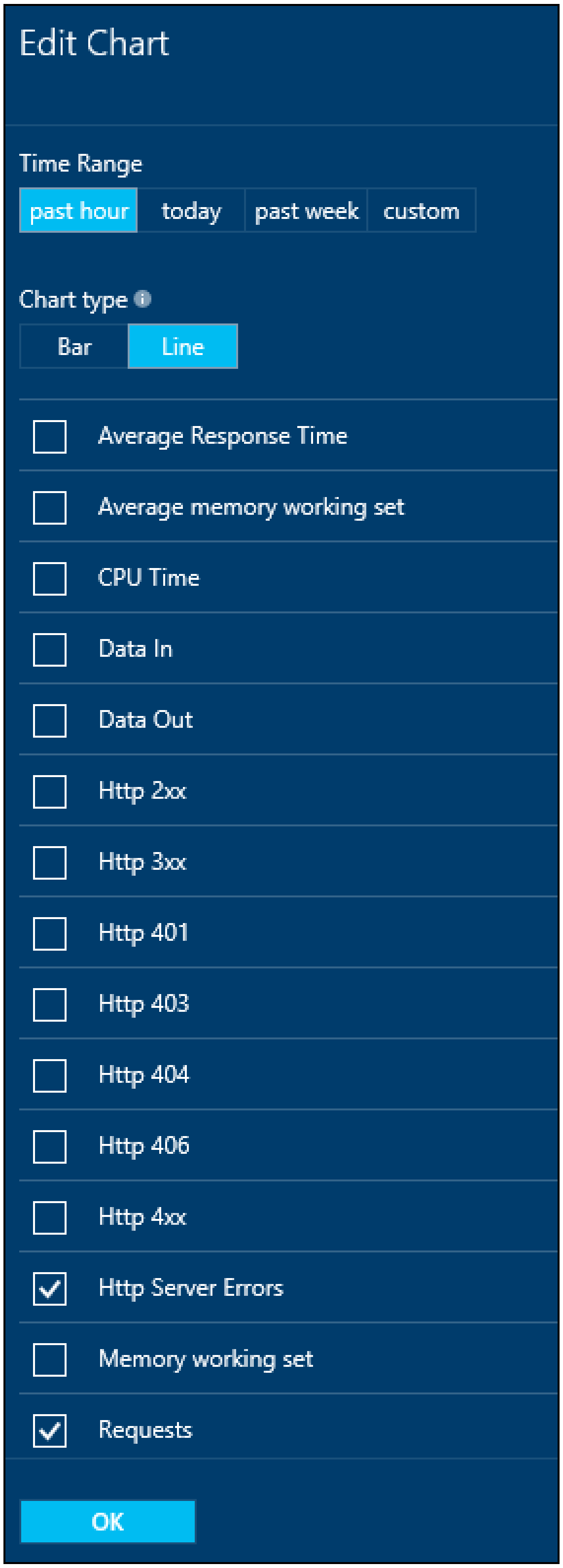
Here is what each of the Publishing settings is for:

- **Deployment Source** This is where you can choose a source such as Git, GitHub, OneDrive, Bitbucket, Dropbox, or Visual Studio Team Services to be used for continuous deployment.
- **Deployment Slots** This lets you publish multiple versions of your web app to different URLs. For example, you can set one up and call it *staging*, then publish interim changes to it. After you've tested the new version thoroughly, you can put the new version in production by swapping the deployment slot called *staging* with production.
- **Deployment Credentials** This lets you set the user name and password for use with Git and FTP deployment.

There are additional sections for Mobile Apps, WebJobs, and Routing, and a section that enables you to set up a custom domain and SSL bindings.

Monitoring Web Apps

Let's take a look at the many ways you can monitor your application. If you're not already there, log into the Azure portal and go to the blade for your web application. Below the properties of the web app is a pane showing the default Monitoring. You can click Edit in that pane to see all of the metrics you can add to that chart and set the time range to be displayed and the type of chart (Figure 2-30).



Edit Chart

Time Range

[✓ past hour] today past week custom

Chart type

Bar [✓ Line]

□ Average Response Time

□ Average memory working set

□ CPU Time

□ Data In

□ Data Out

□ Http 2xx

□ Http 3xx

□ Http 401

□ Http 403

□ Http 404

□ Http 406

□ Http 4xx

☑ Http Server Errors

□ Memory working set

☑ Requests

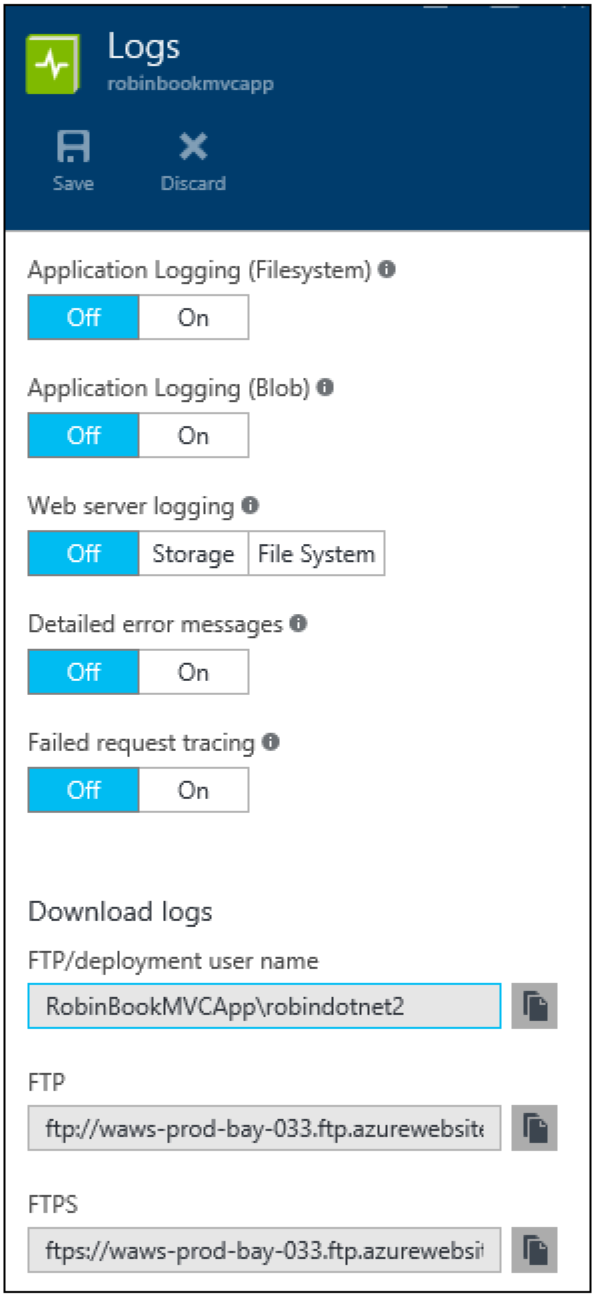
Figure 2-30 Specify the metrics to display on the chart.

In the Settings blade, you can check out your diagnostics in the Site Metrics Per Instance option. This shows overall metrics for your web app as well as metrics for each instance that is running. You can ask to see the last 24 hours, the last hour, or the last 5 days. This is graphed for you.

You can also see the metrics for all of the apps running in your App Service plan by selecting App Service plan Metrics Per Instance. This has the same settings as the option for your site (24 hours, etc.), but the numbers are combined metrics for all of the apps running.

Another option is Live HTTP Traffic, which will show what's going on currently with the web app, showing Request count, HTTP 5xx responses, and HTTP 4xx responses.

Using the Diagnostics Logs setting, you can enable and disable the different kinds of diagnostics logging for your web app, as shown in Figure 2-31. This includes any logging that the application may do, as well as IIS requests and Failed requests. You can FTP into the site to check the logs; the FTP information is also displayed on that blade.



Application Logging (Filesystem)
[✓ Off] On

Application Logging (Blob)
[✓ Off] On

Web server logging
[✓ Off] Storage File System

Detailed error messages
[✓ Off] On

Failed request tracing
[✓ Off] On

Download logs

FTP/deployment user name
RobinBookMVCApp\robindotnet2

FTP
ftp://waws-prod-bay-033.ftp.azurewebsit

FTPS
ftps://waws-prod-bay-033.ftp.azurewebsi

Figure 2-31 Enable or disable the logging.

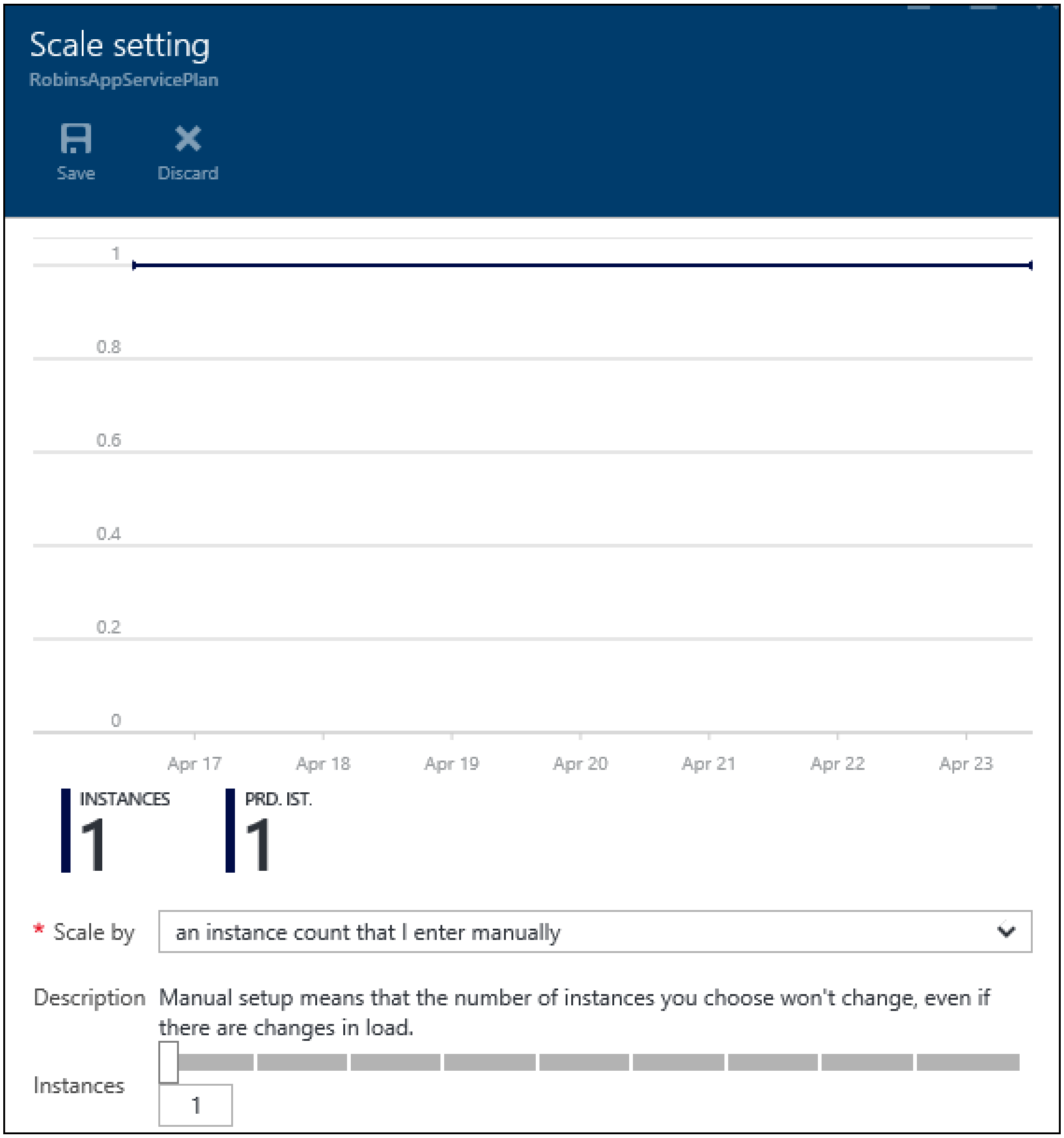
Scaling Web Apps

Let’s go to the Settings blade and look at the scaling options.

> **Note** You don’t scale the web app specifically; you scale the App Service plan, which scales all of the apps running in app services that use that plan.

Scale Up will allow you to select a different pricing tier. This lets you increase the VM size, providing a different amount of memory, storage, etc. that we saw when we originally set up the App Service plan.

Let's take a closer look at scaling out your App Service plan. Figure 2-32 shows the Scale Setting blade that you see when you click Scale Out.



Scale setting
RobinsAppServicePlan

* Scale by: an instance count that I enter manually

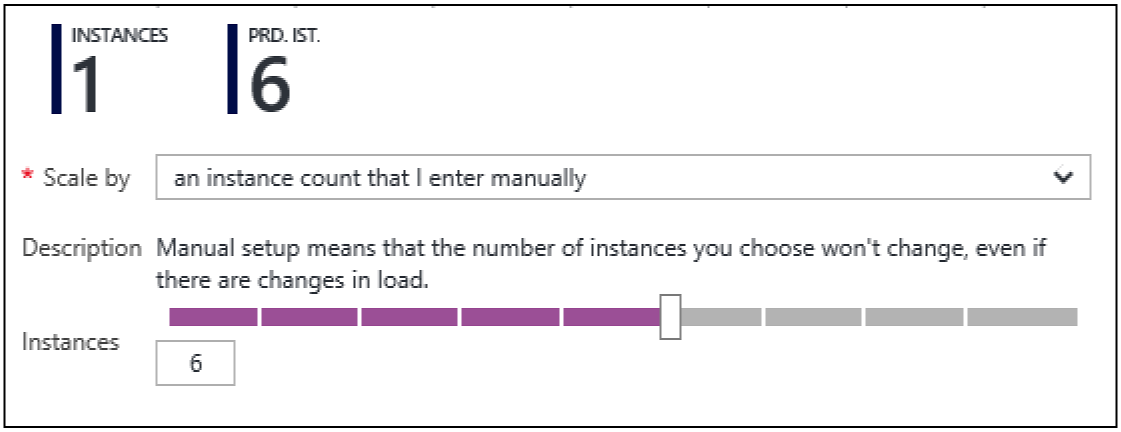
Description: Manual setup means that the number of instances you choose won't change, even if there are changes in load.

Instances: 1

Figure 2-32 Scale Setting blade used for scaling out.

Scaling out manually

On the blade displayed in Figure 2-32, you can specify the number of instances that you want to run by either editing the text box with the number in it or dragging the slider over to the right. Figure 2-33 shows an example requesting that the App Service plan should be scaled out to six instances. This means all apps running in app services that are assigned to that plan will now have six instances.


INSTANCES 1

PRD. IST. 6

* Scale by: an instance count that I enter manually

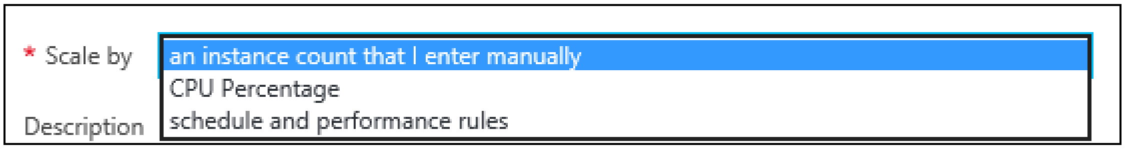
Description: Manual setup means that the number of instances you choose won't change, even if there are changes in load.

Instances: 6

Figure 2-33 Manually scaling out to six instances.

Scaling manually isn't practical unless you're sure your apps will run consistently all of the time. What if you have an application used by a small company, and usage is only high from 8 AM to 5 PM? Do you set it to handle the usage during the day and let it run at that size through the night? It would make more sense to scale down the apps in the evening when there's less usage.

What if you just want to use a simple plan of increasing the instance size when your CPU percentage reaches a specific value, and decrease when the CPU percentage goes back down? You can monitor the app service for these conditions and scale it manually, but wouldn't it be better if you could set it to scale up and down automatically? Figure 2-34 shows the options for Scale By in a drop-down list.


* Scale by

Description

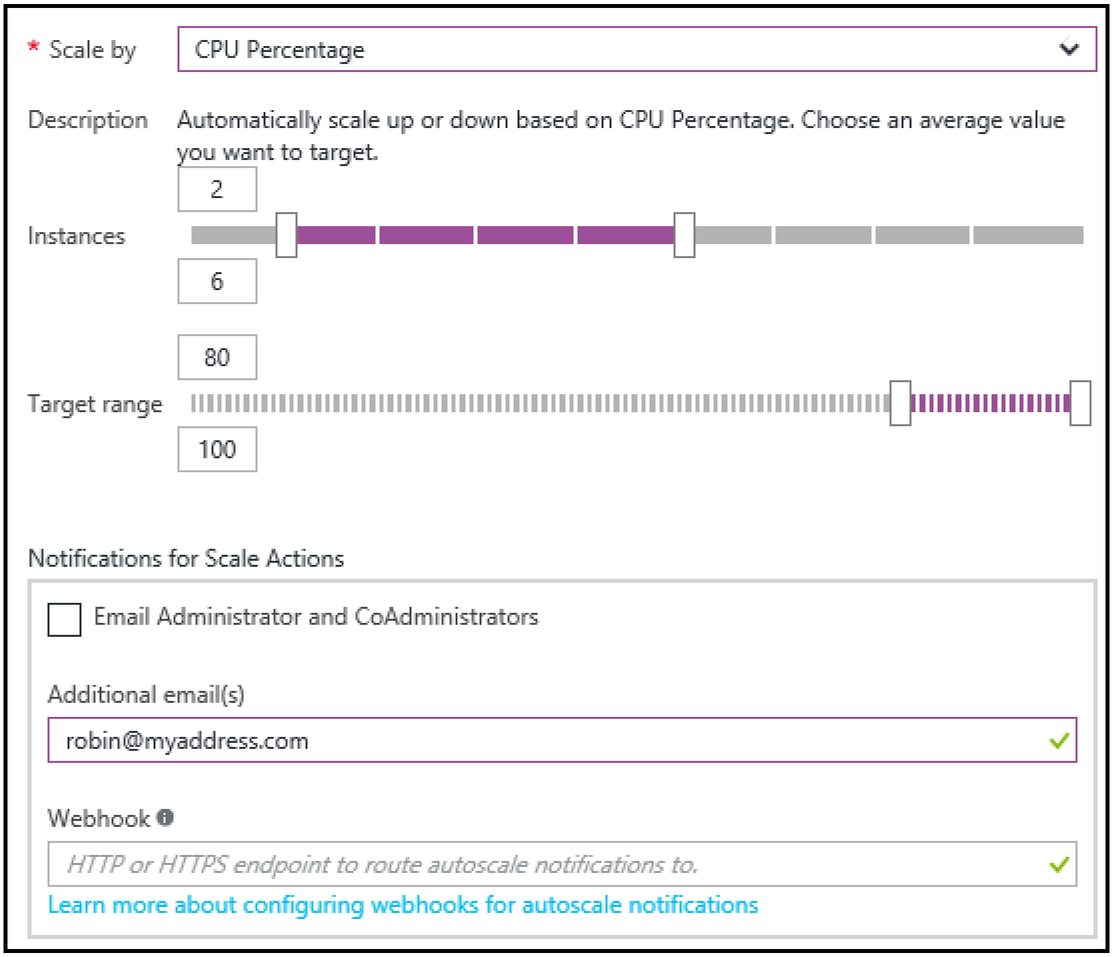
an instance count that I enter manually
CPU Percentage
schedule and performance rules

Figure 2-34 Options for scaling out.

You can see that you now have two more options. One is for scaling by CPU Percentage, and other option lets you put in specific rules for scaling.

Scaling by CPU Percentage

Let's take a look at the CPU Percentage scale settings, shown in Figure 2-35.



* Scale by: CPU Percentage

Description: Automatically scale up or down based on CPU Percentage. Choose an average value you want to target.

Instances: 2 – 6

Target range: 80 – 100

Notifications for Scale Actions

□ Email Administrator and CoAdministrators

Additional email(s): robin@myaddress.com ✓

Webhook

HTTP or HTTPS endpoint to route autoscale notifications to. ✓

Learn more about configuring webhooks for autoscale notifications

Figure 2-35 Scale by CPU Percentage.

This will allow you to scale up or down depending on the CPU Percentage. You can set the lowest number of instances and the highest number of instances as well as the CPU Percentage value where you want the autoscaling to occur. In the case displayed in Figure 2-35, the web app will run on a minimum of two instances and a maximum of six instances. The autoscaling uses standard Microsoft Insights autoscaling, creating an upper and lower bound rule that you can view using the Resource Explorer in the Microsoft.Insights resource for the App Service. It waits 10 minutes between each scaling action.

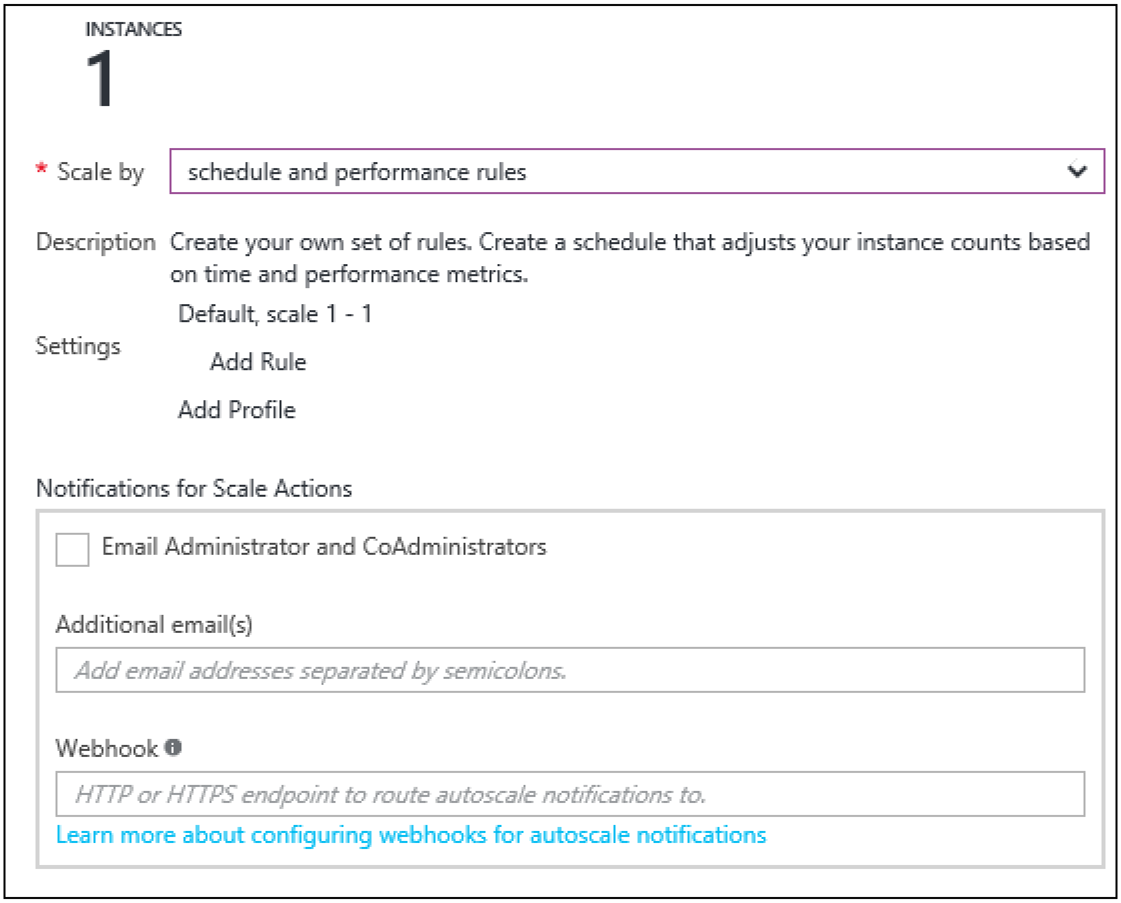
In our case here, when the CPU Percentage reaches 80 percent and stays there for at least 10 minutes, it will start scaling up the instances until the CPU Percentage is below the limit or it reaches the maximum number of instances. When the CPU Percentage is below the limit, it will scale down until it reaches the minimum number of instances.

> **Note** When talking about autoscaling, the average CPU percentage used to scale up or down is the average across all of the VMs running in that App Service plan. This is also true for the other metrics you can use.

You can also set up notifications so it will email you when it starts scaling up and configure a webhook to be run. Webhooks allow you to route the notification to other systems. For example, you could have a service that sends you an SMS message when the scaling begins.

Scaling by schedule and performance rules

The third option allows you to set your own rules. You can set a schedule telling when to scale out and in, and you can even combine that with a performance metric. This is very useful when you want to be really specific about how your app scales out and in. For example, let's say that rather than accepting the default amount of time a value is exceeded before a scaling operation starts, you want to set it to a specific value like 20 minutes, or you want to scale using a different performance metric. You can do this by using this third setting, as shown in Figure 2-36.



INSTANCES
1

* Scale by: schedule and performance rules

Description: Create your own set of rules. Create a schedule that adjusts your instance counts based on time and performance metrics.

Settings:
Default, scale 1 - 1
Add Rule
Add Profile

Notifications for Scale Actions

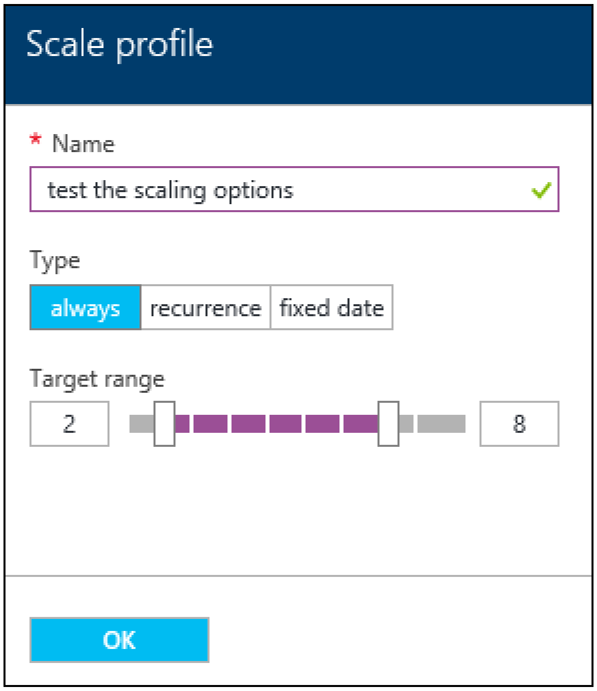
□ Email Administrator and CoAdministrators

Additional email(s)
Add email addresses separated by semicolons.

Webhook
HTTP or HTTPS endpoint to route autoscale notifications to.
Learn more about configuring webhooks for autoscale notifications

Figure 2-36 Custom scaling rules.

This comes with a default profile called Default, Scale 1-1. Let's edit that profile and then define a rule that will specify that the App Service plan should scale out when the average CPU Percentage is greater than 80 percent for more than 17 minutes, and scale in when it averages less than 50 percent for 12 minutes. (I'm using odd numbers rather than the defaults here so you can pick out the numbers on the screen.) Click Default, Scale 1-1 to change the default profile as displayed in Figure 2-37. After setting up the profile, you'll add the rules.



Scale profile

* Name

test the scaling options ✓

Type

[✓ always] recurrence fixed date

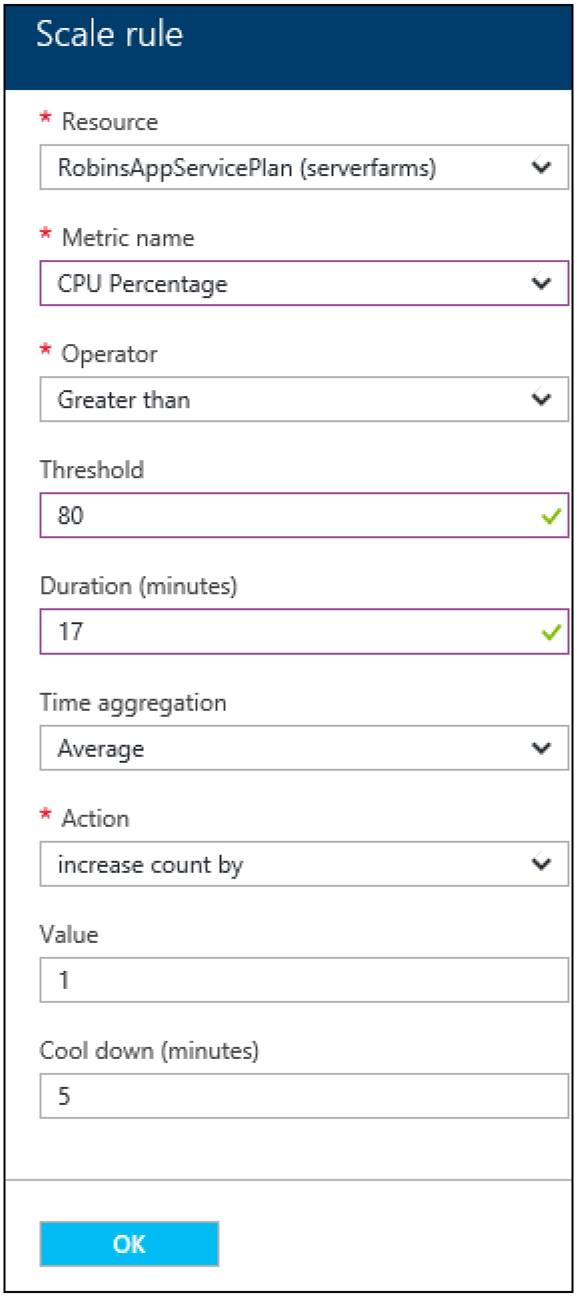
Target range

2 8

Figure 2-37 Set up a profile for scaling.

This profile is called Test The Scaling Options, and it will apply all of the time. The minimum number of instances is two; the maximum number of instances is eight. Set the fields and click OK to add the profile.

Next, click Add Rule under the profile you just edited. This will bring up a blade similar to Figure 2-38.



Scale rule

* Resource
RobinsAppServicePlan (serverfarms)

* Metric name
CPU Percentage

* Operator
Greater than

Threshold
80

Duration (minutes)
17

Time aggregation
Average

* Action
increase count by

Value
1

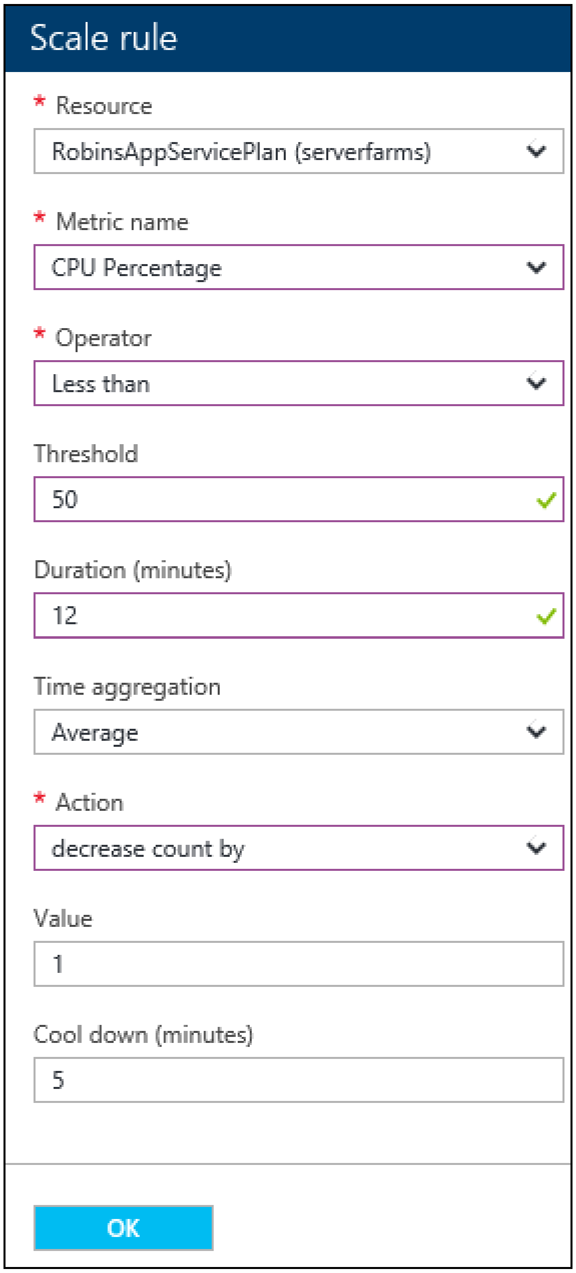
Cool down (minutes)
5

Figure 2-38 Set up a rule for scaling out.

If you click the drop-down list for Metric Name, you'll see several different metrics that you can use to autoscale, such as Memory Percentage, Disk Queue Length, HTTP Queue Length, Data In, and Data Out. For example, if your web application lets people upload data, you might want to autoscale it if they are uploading a ton of data and send a notification to someone who can check and make sure it's legitimate and not a hacker. Set this to autoscale based on CPU Percentage.

In this case, when the average CPU Percentage is over 80 percent for more than 17 minutes, it will scale up by one instance. The Cool Down (Minutes) is the amount of time before another scaling action will take place. So after 5 minutes, if the average CPU Percentage is still over 80 percent, it will add another instance. It will continue to do this until it reaches the maximum number of instances you set on the profile, which was eight. Fill in the fields and click OK to save the rule.

Now we need a rule that says if the average CPU Percentage is less than 50 percent for more than 12 minutes, decrease the instance count. It will keep decreasing the instance count until it reaches the minimum number of instances, which is two in our case. Figure 2-39 shows how to set up this rule.

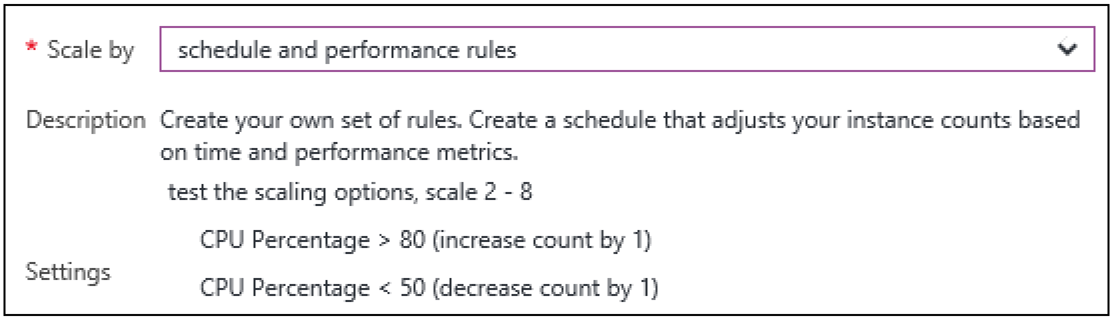


Scale rule

* Resource	RobinsAppServicePlan (serverfarms)
* Metric name	CPU Percentage
* Operator	Less than
Threshold	50
Duration (minutes)	12
Time aggregation	Average
* Action	decrease count by
Value	1
Cool down (minutes)	5

Figure 2-39 Set up a rule for scaling in.

After filling in the fields, click OK to save the rule. Now, the Scale Setting blade should look similar to Figure 2-40.



* Scale by: schedule and performance rules

Description: Create your own set of rules. Create a schedule that adjusts your instance counts based on time and performance metrics.

Settings: test the scaling options, scale 2 - 8

CPU Percentage > 80 (increase count by 1)

CPU Percentage < 50 (decrease count by 1)

Figure 2-40 A screenshot showing the specified profile and its rules.

Click Save at the top of the Scale Setting blade to save the scaling settings. Now, the App Service plan will scale using these rules. Note that all Web Apps, Mobile Apps, etc. that use that App Service plan will be scaled as the App Service plan scales.

There are two other options for the profile—one is *Recurrence* and the other is *Fixed Date*. Fixed Date allows you to provide a specific date/time range with scaling information. For example, you may want it to scale up faster on opening day for your web application.

Recurrence allows you to specify a start time and which days of the week apply. This is what you would use if you had a web application used by a company from 8 AM to 5 PM but not much after that. You would add a profile to start at 8 AM on Monday through Friday to scale out, and then add one to start at 5 PM on Monday through Friday to scale back in for the evenings and weekends.

If you have multiple profiles, there is an order of precedence in which they are handled. When processed by the Autoscale service, the profiles are always checked in the following order:

1. Fixed Date
2. Recurrence profile
3. Default (Always) profile

In the example above, you would set up a Recurrence profile for Monday through Friday, 8 AM to 5 PM, and then set up a Fixed Date profile for a specific holiday, and the Fixed Date profile will take precedence on that one date.

For more information about autoscaling, including best practices, please check out this article: https://azure.microsoft.com/documentation/articles/insights-autoscale-best-practices/.

CHAPTER 3

Azure Virtual Machines

Platform as a service (PaaS) is an attractive option for a certain category of workloads. However, not every solution can, or should, fit within the PaaS model. Some workloads require near-total control over the infrastructure: operating system configuration, disk persistence, the ability to install and configure traditional server software, and so on. This is where infrastructure as a service (IaaS) and Azure Virtual Machines come into the picture.

What is Azure Virtual Machines?

Azure Virtual Machines is one of the central features of Azure's IaaS capabilities, together with Azure Virtual Networks. Azure Virtual Machines supports the deployment of Windows or Linux virtual machines (VMs) in a Microsoft Azure datacenter. You have total control over the configuration of the VM. You are responsible for all server software installation, configuration, and maintenance and for operating system patches.

> **Note** The terminology used to describe the Azure Virtual Machines feature and a virtual machine instance can be a little confusing. Therefore, throughout this chapter, *Azure Virtual Machines* will refer to the feature, while *virtual machine* or *VM* will refer to an instance of an actual compute node.

There are two primary differences between Azure's PaaS and IaaS compute features: persistence and control. As discussed in Chapter 2, "Azure App Service and Web Apps," PaaS features such as Cloud Services (that is, web and worker roles) and App Services are managed primarily by the Azure platform, allowing you to focus on creating the application and not managing the server infrastructure. With an Azure Virtual Machines VM, you are responsible for nearly all aspects of the VM.

Azure Virtual Machines supports two types of durable (or persistent) disks: OS disks and data disks. An OS disk is required, and data disks are optional. The durability for the disks is provided by Azure Storage. More details on these disks will be provided later in this chapter, but for now understand the OS disk is where the operating system resides (Windows or Linux), and the data disk is where you can store other things, such as application data, images, and so on. By contrast, Azure PaaS cloud services use ephemeral disks attached to the physical host—the data on which can be lost in the event of failure of the physical host.

Because of the level of control afforded to the user and the use of durable disks, VMs are ideal for a wide range of server workloads that do not fit into a PaaS model. Server workloads such as database servers (SQL Server, Oracle, MongoDB, and so on), Windows Server Active Directory, Microsoft SharePoint, and many more become possible to run on the Microsoft Azure platform. If desired, users can move such workloads from an on-premises datacenter to one or more Azure regions, a process often called *lift and shift*.

Billing

Azure Virtual Machines is priced on a per-hour basis, but it is billed on a per-minute basis. For example, you are only changed for 23 minutes of usage if the VM is deployed for 23 minutes. The cost for a VM includes the charge for the Windows operating system. Linux-based instances are slightly cheaper because there is no operating system license charge. The cost, and the appropriate licensing, for any additional software you install is your responsibility. Some VM images, such as Microsoft SQL Server, you acquire from the Azure Marketplace may include an additional license cost (on top of the base cost of the VM).

There is a direct relationship between the VM’s status and billing:

- **Running** The VM is on and running normally (billable).
- **Stopped** The VM is stopped but still deployed to a physical host (billable)
- **Stopped (Deallocated)** The VM is not deployed to a physical host (not billable).

You are charged separately for the durable storage the VM uses. The status of the VM has no relation to the storage charges that will be incurred; even if the VM is stopped/deallocated and you aren’t billed for the running VM, you will be charged for the storage used by the disks.

By default, stopping a VM in the Azure portal puts the VM into a Stopped (Deallocated) state. If you want to stop the VM but keep it allocated, you will need to use a PowerShell cmdlet or Azure command-line interface (CLI) command.

Stopping an Azure VM

To stop a VM but keep it provisioned, you would need to use the *Stop-AzureRmVM* PowerShell cmdlet such as in the following example:

```
Stop-AzureRmVM -Name "AzEssentialDev3" -ResourceGroup "AzureEssentials" -StayProvisioned
```

For classic VMs, a similar cmdlet, *Stop-AzureVM*, would be used.

When using the Azure CLI, there are two commands used to control the stopped state of a VM: *azure vm stop* and *azure vm deallocate*.

Shutting down the VM from the operating system of the VM will also stop the VM but will not deallocate the VM.

Note The Azure Hybrid Use Benefit program may offer additional savings by allowing you bring your on-premises Windows Server licenses to Azure. For more information, please see https://azure.microsoft.com/pricing/hybrid-use-benefit/.

Service level agreement

As of the time of this writing, Microsoft offers a 99.95 percent connectivity service level agreement (SLA) for multiple-instance VMs deployed in an availability set. That means that for the SLA to apply, there must be at least two instances of the VM deployed within an availability set. Additional details pertaining to availability sets for Azure Virtual Machines are discussed later in this chapter.

See Also See the SLA at http://azure.microsoft.com/support/legal/sla/ for full details.

Virtual machine models

As you may recall from earlier in this book, there are two models for working with many Azure resources: Azure Resource Manager (ARM) and Azure Service Management (often referred to as the classic model or ASM). Please see Chapter 1, "Getting started with Microsoft Azure," for a more detailed overview. It is recommended that you use the Resource Manager model for new deployments. The classic model is still supported; however, the newest innovations will be made available only for the Resource Manager model.

For the purposes of this chapter, both models are covered, but the emphasis is on the Resource Manager model.

There are significant and fundamental differences in working with Azure Virtual Machines in these models.

Azure Resource Manager model

When working with the Resource Manager model, you have explicit and fine-grained control over nearly all aspects of the Azure VM. You will explicitly add components such as a network interface card (NIC), public IP address, data disks, load balancer, and much more.

You may recall that Resource Manager uses various resource providers to enable access to and management of Azure resources. There are three main resource providers used when working with Azure Virtual Machines: Network, Storage, and Compute.

- The Network resource provider (Microsoft.Network) handles all aspects of network connectivity such as IP addresses, load balancers, NICs, and so on.
- The Storage resource provider (Microsoft.Storage) handles the storage of the disks for a VM (in the context of Azure Virtual Machines).
- The Compute resource provider (Microsoft.Compute) handles details related to the VM itself, such as naming, operating system details, and configuration (size, number of disks, and so on).

In addition to explicit control over the virtual machine's components, you have the ability to take advantage of other Resource Manager features, such as:

- Deployment and management of related resources as part of a resource group
- Tags to logically organize and identify resources
- Role Based Access Control (RBAC) to apply necessary security and control policies
- Declarative template files
- Deployment policies to enforce specific organizational rules
- Consistent, orchestrated deployment process

This ability affords you a great deal of control in configuring the environment to your exact needs.

Classic/Azure Service Management model

In the classic deployment model, VM deployments are always in the context of an Azure cloud service—a container for VMs. The container provides several key features, including a DNS endpoint, network connectivity (including from the public Internet if desired), security, and a unit of management. While you get these things for free—because they're inherited from the cloud service model—you have limited control over them.

Use of the classic model also excludes the use of the additional value adding features available via Azure Resource Manager (tags, template files, and so on).

Virtual machine components

Like a car, there are many components that make up a virtual machine. Also like a car, there are multiple configuration options available to suit the specific functional needs and desires of the owner. The sections that follow describe several of the critical components of Azure Virtual Machines. Additionally, more advanced configuration options will be discussed later in the chapter. But first, the base model needs to be established.

Virtual machine

It is sometimes helpful to think of an Azure VM as a logical construct. A virtual machine can be defined as having a status, a specific configuration (operating system, CPU cores, memory, disks, IP address, and so on), and state. That logical definition can be instantiated by Azure, and the appropriate resources can be allocated to bring that VM to life.

Disks

Azure VMs use attached VHDs to provide durable storage. There are two types of VHDs used in Azure Virtual Machines:

- **Image** A VHD that is a template for the creation of a new Azure VM. As a template, it does not have settings such as a machine name, administrative user, and so on. More information on creating and using images is provided later in this chapter.
- **Disk** A possibly bootable VHD that can be used as a mountable disk for a VM. There are two types of disks: an OS disk and a data disk.

All durable disks (the OS disk and data disks) are backed by page blobs in Azure Storage. Therefore, the disks inherit the benefits of blob storage: high availability, durability, and geo-redundancy options. Blob storage provides a mechanism by which data can be stored safely for use by the VM. The disks can be mounted as drives on the VM. The Azure platform will hold an infinite lease on the page blob to prevent accidental deletion of the page blob containing the VHD, the related container, or the storage account.

Standard and Premium Storage

The disk files (.vhd files) can be backed by either Standard or Premium Storage accounts in Azure. Azure Premium Storage leverages solid-state disks (SSDs) to enable high performance and low latency for VMs running I/O-intensive workloads. Standard storage is available for all VM sizes, while Premium storage is available for DS, DSv2, F, and GS-series VMs only. Standard storage can also be used with DS, DSv2, F, and GS-series VMs, in which case only the local, ephemeral drive runs on an SSD.

In general, it is recommended to use Azure Premium Storage for production workloads, especially those that are sensitive to performance variations or are I/O intensive. For development or test workloads, which are often not sensitive to performance variations and are not I/O intensive, Azure Standard Storage is generally recommended.

For a thorough review of Azure Premium Storage and implications for Azure VMs, please see Chapter 4, "Azure Storage," and reference https://azure.microsoft.com/documentation/articles/storage-premium-storage/.

An OS disk is used precisely as the name suggests: for the operating system. For a Windows VM, the OS disk is the typical C drive; this is where Windows places its data. For a Linux VM, it hosts the /dev/sda1 partition used for the root directory. The maximum size for an OS disk is currently 1,023 GB.

The other type of disk used in Azure Virtual Machines is a data disk. The data disk is also used precisely as the name would suggest: for storing a wide range of data. The maximum size for a data disk is also 1,023 GB. Multiple data disks can be attached to an Azure VM, although the maximum number varies by VM size—typically two disks per CPU. The data disks are often used for storing application data, such as data belonging to your custom application, or server software, such as Microsoft SQL Server and the related data and log files. Multiple data disks can be made into a disk array using Storage Spaces on Windows or mdadm on Linux.

See Also For a breakdown of the various VM sizes, including CPU cores, memory, maximum data disks, and so on, please see https://azure.microsoft.com/documentation/articles/virtual-machines-windows-sizes/.

Azure Virtual Machines also include a temporary disk on the physical host that is not persisted to Azure Storage. The temporary disk is a physical disk located within the chassis of the server. Depending on the type of VM created, the temporary disk may be either a traditional HDD platter or an SSD. The temporary disk should be used only for temporary (or replicated) data because the data will be lost in the event of a failure of the physical host or when the VM is stopped/deallocated. Figure 3-1 shows the various disk types.

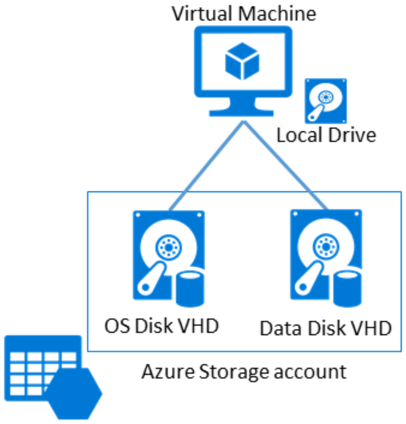


Figure 3-1 Disk types in Azure Virtual Machines.

Virtual Network

In an on-premises physical infrastructure, you may have many components that all allow you to operate your virtual machines in a scalable and secure manner. These components could include equipment such as separate network spaces for Internet-facing and backend servers, load balancers, firewalls, and more. Many of these components can logically be deployed in an Azure Virtual Network (often referred to as VNET). Azure Virtual Network provides many similar features, such as the following:

- **Subnet** A subnet is a range of IP addresses within a virtual network. A VM must be placed in a subnet within the VNET. VMs placed in one subnet of a VNET can freely communicate with VMs in another subnet of the same virtual network. However, you can use network security groups (NSGs) and user-defined routes to control such communication.
- **IP address** An IP address can be either public or private. Public IP addresses allow communication from the Internet to the VM. A public IP address can be allocated dynamically—that is, created only when the associated resource (such as a VM or load balancer) is started and released when said resource is stopped—or statically, in which case the IP address is assigned immediately and persists until deleted. Private IP addresses are non–Internet routable addresses used for communication with VMs and load balancers in the same VNET.
- **Load balancer** VMs are exposed to the Internet or other VMs in a VNET by using Azure load balancers. There are two types of load balancers:
  - **External load balancer** Used for exposing multiple VMs to the Internet in highly available manner.
  - **Internal load balancer** Used for exposing multiple VMs to other VMs in the same VNET in a highly available manner.
- **Network security group** A NSG allows you to create rules that control (approve or deny) inbound and outbound network traffic to network interface cards (NICs) of a VM or subnets.

When creating a VM in Azure using the Resource Manager model, it is required that the VM be placed within an Azure Virtual Network (VNET). You will decide to use an existing VNET (or create a new one), the subnet to use, the IP address, if there is a load balancer or not, the number of NICs, and how network security is handled, as depicted in Figure 3-2. While it may seem like a lot just to get a VM deployed, these are important aspects to consider for the accessibility and security of the VM.

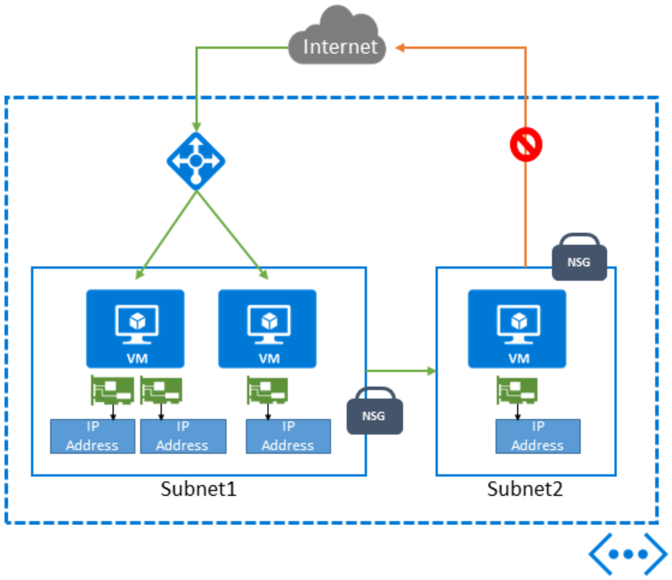


Figure 3-2 VMs in the Resource Manager model have explicit control over related network components.

Classic VMs can also be placed in an Azure Virtual Network. However, this is not a requirement (as it is with VMs in the Resource Manager model).

For more a more detailed discussion of the Azure Virtual Network feature, please see Chapter 5, "Azure Virtual Networks."

IP address

In the Resource Manager model, by default, a VM does not have an IP address. One must be explicitly granted to a VM via an associated NIC. A VM requires an IP address to support communication with other VMs in the virtual network or the public Internet.

Each NIC has an associated private address (often referred to as a DIP, or dynamic IP) used to connect to the virtual network and is optionally associated with a public IP address connected directly to the public Internet. By default, these dynamic IP addresses are lost when the VM is stopped/deallocated, but both may be declared as static to make them persist unchanged throughout the shutdown/deallocation of the VM. This is useful for VMs that need permanent DIPs, such as Microsoft SQL Server, DNS server VMs, or permanent public IP addresses. Multiple NICs, each with their own DIPs, can be attached to a VM if more than one DIP is needed—for example, to multi-home a VM in multiple subnets.

In the classic model, the story is similar except that NICs and public IP addresses can only exist in the context of a VM—that is, they are not independent resources. Furthermore, in the classic model, it is more usual to have Internet connectivity provided by the Azure Load Balancer rather than through a public IP Address.

Azure Load Balancer

As mentioned previously, the Azure Load Balancer is used to provide a relatively even distribution of network traffic across a set of (often similarly configured or related) VMs. Using the load balancer allows you to have multiple VMs work together—for example, as a collection of web servers in a web farm environment. With a load-balanced set (of VMs), incoming requests are distributed across the available VMs instead of being routed to a single VM.

There are two types of load balancers available in Azure: an external load balancer and an internal load balancer, as depicted in Figure 3-3. The external load balancer is used for distributing traffic from the Internet across one or more VMs. This enables you to expose your application in a highly scalable and highly available manner.

The internal load balancer is used to distribute traffic from within a virtual network across a set of VMs. For example, this could be traffic to a web API or database cluster that should be available only to front-end web servers, not to the public Internet.

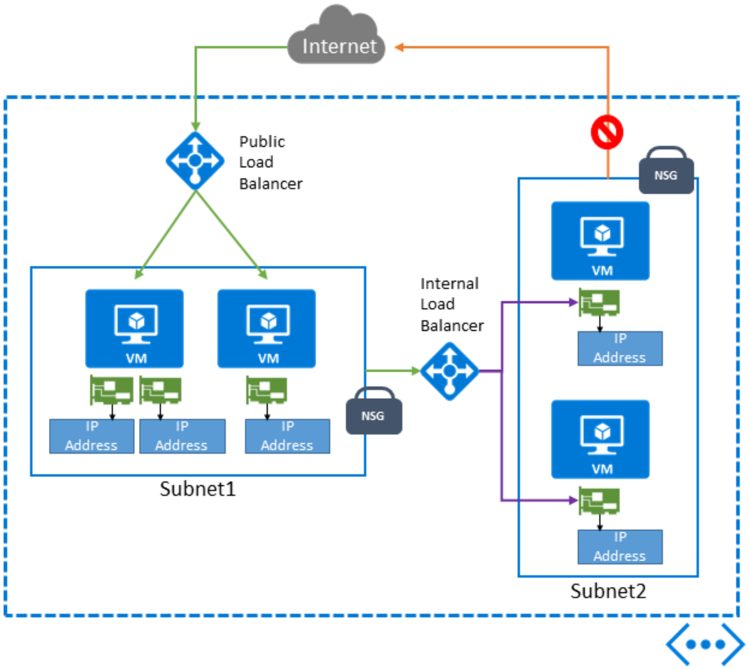


Figure 3-3 Use of both an external and an internal load balancer.

In the Resource Manager model, to use a load balancer, several additional items must first be created:

- Public IP address for the incoming network traffic (for an external load balancer)
- A pool of backend (private) IP addresses associated to NICs for the VMs
- Rules to define the mapping of a public port on the load balancer to a port in the backend pool
- Inbound NAT rules to define the mapping of a public port on the load balancer to a specific VM in the pool
- Health probes to determine if a VM in the pool is healthy

In the classic model, the external load balancer is provided automatically as part of the cloud service model. All VMs in the cloud service are automatically configured to use the load balancer if they expose a public endpoint. Classic VMs can also use an internal load balancer.

See Also For more details on working with a load balancer in Azure, please see the following:

Resource Manager model: https://azure.microsoft.com/documentation/articles/load-balancer-arm/

Classic model: https://azure.microsoft.com/documentation/articles/load-balancer-get-started-internet-classic-portal/.

Classic model (internal load balancer): https://azure.microsoft.com/documentation/articles/load-balancer-get-started-ilb-classic-ps/

Network interface card (NIC)

A network interface card (NIC) provides network access to resources in an Azure virtual network. A NIC is a standalone resource, but it must be associated with a VM to provide network access (a NIC by itself is of little value). The maximum number of NICs attached to a VM is dependent on the size of the selected VM.

There are several important points to be aware of when working with NICs and VMs:

- The IP address for each NIC on a VM must be located in a subnet of the VNET to which the VM belongs.
- If multiple NICs are assigned to a VM, only the primary NIC can be assigned the public IP address. Each NIC will get assigned a private IP address (assuming the NIC is not the primary NIC and has a public IP address). The NICs can be in different subnets of the VNET.
- Any NIC on a VM can be associated with a network security group (NSG).

When working with classic VMs, it is not necessary to worry about the NIC configuration because that is handled automatically as part of the cloud service model and cannot exist outside the context of a VM.

> **See Also** For more information on working with VMs with multiple NICs, please see https://azure.microsoft.com/documentation/articles/virtual-networks-multiple-nics/.

Network security groups

Network security groups (NSGs) allow you to have fine-grained and explicit control over how network traffic flows into or out of Azure VMs and subnets.

NSGs allow you to shape the network traffic flow in and out of your environment. You create rules based on the source IP address and port and the destination IP address and port. The NSG rules can be applied to a VM and/or a subnet. For a VM, the NSG is associated with the NIC attached to the VM.

For more information on network security groups, please see Chapter 5 and also https://azure.microsoft.com/documentation/articles/virtual-networks-nsg/.

Availability set

Azure VMs reside on physical servers hosted within Microsoft’s Azure datacenters. As with most physical devices, there is a chance that there could be a failure. If the physical server fails, the Azure VMs hosted on that server will also fail. Should a failure occur, the Azure platform will migrate the VM to a healthy host server on which to reconstitute the VM. This service-healing process could take several minutes. During that time, the application(s) hosted on that VM will not be available.

Besides hardware failures, the VMs could be affected by periodic updates initiated by the Azure platform itself. Microsoft will periodically upgrade the host operating system on which the guest VMs are running (you’re still responsible for the operating system patching of the guest VM that you create). During these updates, the VM will be rebooted and thus temporarily unavailable.

To avoid a single point of failure, it is recommended to deploy at least two instances of the VM. In fact, Azure provides an SLA only when two or more VMs are deployed into an availability set. This is a logical feature used to ensure that a group of related VMs are deployed so that they are not all subject to a single point of failure and not all upgraded at the same time during a host operating system upgrade in the datacenter. The first two VMs deployed in an availability set are allocated to two different fault domains, ensuring that a single point of failure will not affect them both simultaneously. Similarly, the first five VMs deployed in an availability set are allocated to five different update domains, minimizing the impact when the Azure platform induces host operating system updates one update domain at a time. VMs placed in an availability set should perform an identical set of functionalities.

The number of fault domains and update domains is different depending on the deployment model—Resource Manager or classic. In the Resource Manager model, you can have up to 3 fault domains and 20 upgrade domains. With the classic model, you can have 2 fault domains and 5 upgrade domains.

Create virtual machines

There are two tiers for Azure Virtual Machines, Basic and Standard. VMs in the Basic tier are well suited for workloads that do not require load balancing or the ability to autoscale. VMs in the Standard tier support all Azure Virtual Machines configurations and features. This tier is recommended for most production scenarios.

The Basic tier contains only a subset of the A-series VM sizes, A0–A4. The Standard tier supports all available VM sizes and series: A-Series, D-Series, Dv2-Series, F-Series, and G-Series. There are also variants of the D, Dv2, F, and G-Series sizes, called DS, DSv2, F, and GS, which support Azure Premium Storage.

> **Note** With the introduction of the F-Series VM sizes, Microsoft announced a new naming standard for VM sizes. Starting with the F-Series and applying to any future VM sizes, a numeric value after the family name will match the number of CPU cores. Additional capabilities, such as premium storage, will be designated by a letter following the CPU core count. For example, Standard_F8s will indicate an F-Series VM supporting premium storage with eight CPU cores (the "s" indicates premium storage support). This new naming standard will not be applied to previously introduced VM sizes.

- **A-Series** The "traditional" sizes that have been around since Azure Virtual Machines was introduced. These are your general-purpose VMs.
- **D-Series** Introduced in September 2014, they feature processors that are 60 percent faster than the A-Series, a higher memory-to-core ratio, and an SSD for the temporary physical disk.
- **Dv2-Series** Introduced in October 2015, the Dv2-Series are the next generation of the D-Series instances. They carry the same memory and disk configuration as the D-Series, yet they are on average 35 percent faster than the D-Series (thanks to the 2.4 GHz Intel® Xeon® E5-2673 v3 [Haswell] processor).
- **G-Series** Introduced in January 2015, the G-Series VMs are intended for your most demanding workloads. The G-Series VMs feature two times more memory and four times more storage than D-Series VMs and also include the latest Intel® Xeon® E5 v3 processors. G-Series VMs also use a SSD for the temporary physical disk.
- **F-Series** Introduced in June 2016, the F-Series VMs provide the same CPU performance (the same 2.4 GHz Intel® Xeon® E5-2673 v3 [Haswell] processor) as the Dv2-Series VMs but at a lower per-hour price. The difference with the F-Series is they feature 2 GB of memory per CPU core and less local SSD space. The F-Series can be an excellent choice for workloads that might not benefit from additional memory or local SSD space.
- **N-Series** Announced in September 2015, the N-Series VMs feature GPU capabilities, powered by NVIDIA. At the time of this writing, N-Series VMs are limited to a private preview.

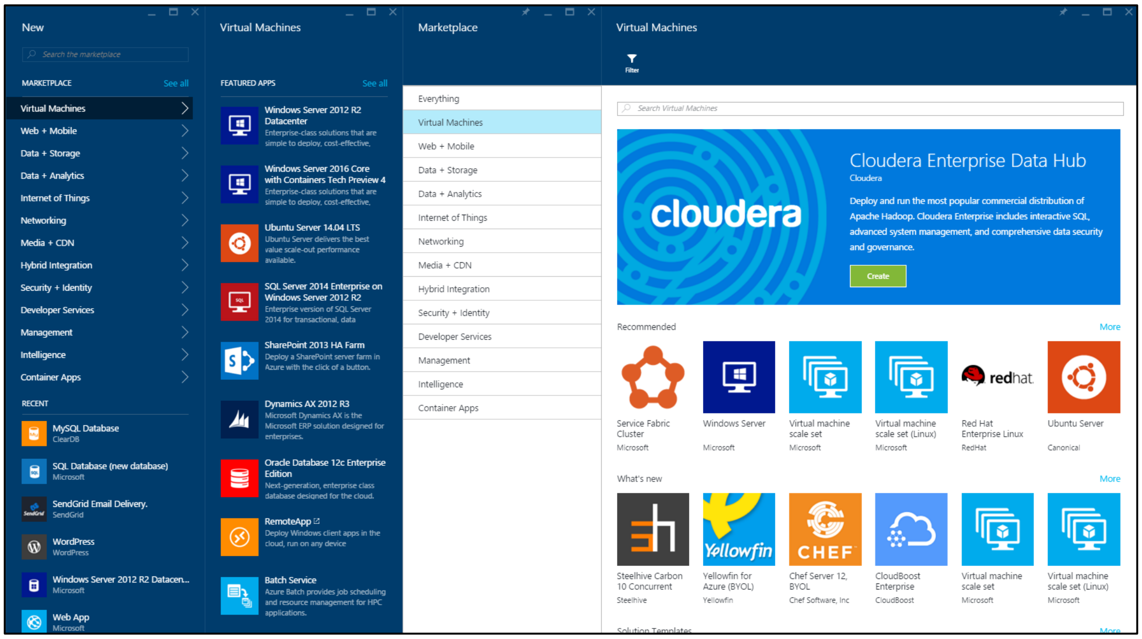
> **See Also** For the most current Azure Virtual Machines configurations, please see https://azure.microsoft.com/documentation/articles/virtual-machines-windows-sizes/.

One of the easiest ways to get started creating Azure VMs is to use the Azure portal.

Create a virtual machine with the Azure portal

If you haven't already done so, log into the Azure portal at http://portal.azure.com. At this point, you will need an Azure subscription. If you don't have one, you can sign up for a free trial at http://azure.microsoft.com.

To get started, click New in the navigation section of the site and then the Virtual Machines option in the Marketplace. As can be seen in Figure 3-4, doing so opens the Virtual Machines Marketplace blade, where you can select from a wide range of VM configurations and preconfigured images from Microsoft, Microsoft partners, and ISVs. The images in the Marketplace include official images from Microsoft for Windows-based deployments such as Window Server 2012, Microsoft SharePoint server farms, and more, and select partners such as Red Hat, Canonical, DataStax, Oracle, and many more.



Windows Server 2012 R2 Datacenter
Enterprise-class solutions that are simple to deploy, cost-effective,

Windows Server 2016 Core with Containers Tech Preview 4
Enterprise-class solutions that are simple to deploy, cost-effective,

Ubuntu Server 14.04 LTS
Ubuntu Server delivers the best value scale-out performance available.

SQL Server 2014 Enterprise on Windows Server 2012 R2
Enterprise version of SQL Server 2014 for transactional, data

SharePoint 2013 HA Farm
Deploy a SharePoint server farm in Azure with the click of a button.

Dynamics AX 2012 R3
Microsoft Dynamics AX is the Microsoft ERP solution designed for enterprises.

Oracle Database 12c Enterprise Edition
Next-generation, enterprise class database designed for the cloud.

RemoteApp
Deploy Windows client apps in the cloud, run on any device

Batch Service
Azure Batch provides job scheduling and resource management for HPC applications.

Cloudera Enterprise Data Hub
Cloudera
Deploy and run the most popular commercial distribution of Apache Hadoop. Cloudera Enterprise includes interactive SQL, advanced system management, and comprehensive data security and governance.

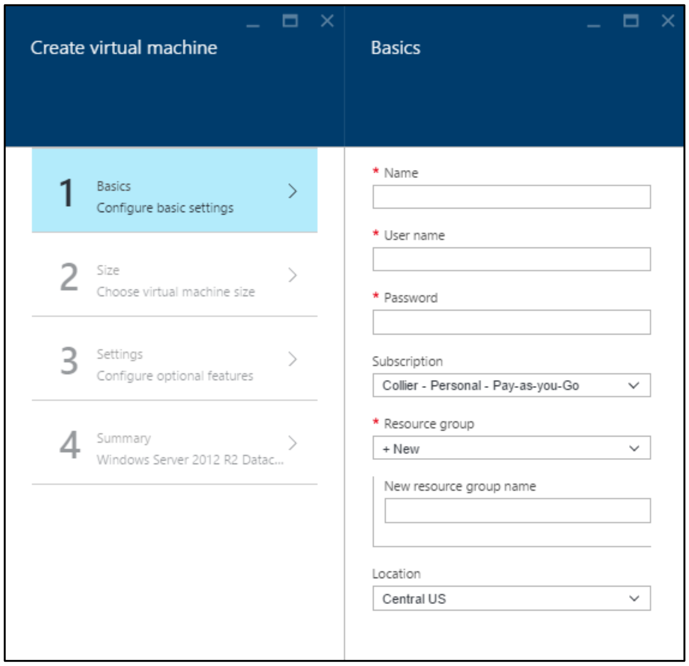
Figure 3-4 The Virtual Machines Marketplace.

For the purposes of this example, select the Windows Server 2012 R2 Datacenter image. If it isn't immediately listed, you can search for the desired image. On the resulting blade, you can read information about the image, including any operating system updates. You will also have the option to choose a deployment model, either Resource Manager or Classic. For the purposes of this example, choose Resource Manager. Click the Create button to proceed with creating your new VM.

> **Note** As Microsoft and its partners transition to the Resource Manager model, an increasing number of images in the Marketplace are only available via the Resource Manager model.

Next, the Create Virtual Machine blade should open and then extend the first blade to configure basic settings. As you can see in Figure 3-5, on this blade you provide several important details about your new VM:

- **Name** The name of the VM
- **User Name** The administrative user name
- **Password** The password for the administrative user
- **Subscription** The Azure subscription to use if you have more than one
- **Resource Group** Provides a logical container for Azure resources (to help manage resources that are often deployed together)
- **Location** The Azure region where the VM should be placed



Create virtual machine

1 Basics
Configure basic settings

2 Size
Choose virtual machine size

3 Settings
Configure optional features

4 Summary
Windows Server 2012 R2 Datac...

Basics

* Name

* User name

* Password

Subscription
Collier - Personal - Pay-as-you-Go

* Resource group
+ New

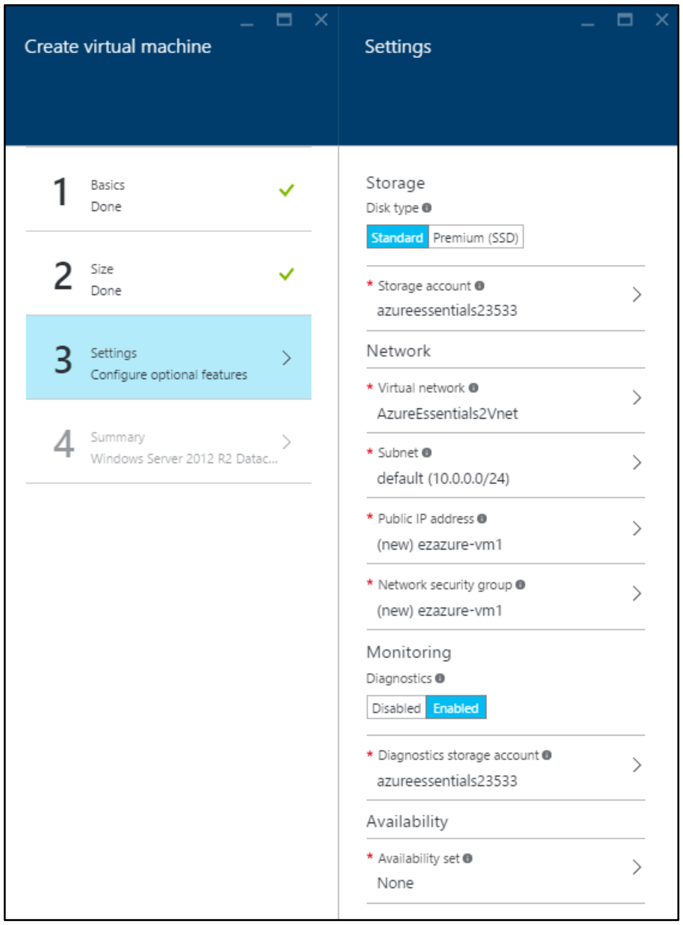
New resource group name

Location
Central US

Figure 3-5 The Create Virtual Machine blade.

When finished with the Basics blade, click the OK button to proceed to the next step to select your VM size. Not all VM sizes are available in all Azure regions. If a size is not available in the selected region, that size option will show as disabled when viewing all the VM sizes.

After selecting the VM size, you'll move to the third configuration blade, as seen in Figure 3-6, to set up features related to storage, networking, monitoring, and availability.



Create virtual machine

1 Basics
Done ✓

2 Size
Done ✓

3 Settings
Configure optional features

4 Summary
Windows Server 2012 R2 Datac...

Settings

Storage

Disk type
Standard | Premium (SSD)

* Storage account
azureessentials23533

Network

* Virtual network
AzureEssentials2Vnet

* Subnet
default (10.0.0.0/24)

* Public IP address
(new) ezazure-vm1

* Network security group
(new) ezazure-vm1

Monitoring

Diagnostics
Disabled | Enabled

* Diagnostics storage account
azureessentials23533

Availability

* Availability set
None

Figure 3-6 Optional configuration settings for a new Azure VM.

Let's walk through several of the important settings in this third blade:

- **Storage** Select the storage medium for the OS disk in the new Azure VM.
  - **Disk Type** Select either a Standard (backed by a traditional magnetic HDD) or Premium (backed by SSD) disk.
  - **Storage Account** Select the Azure Storage account in which to place the OS disk. This can be a new storage account or an existing storage account.
- **Network** All VMs in the Resource Manager model must be placed within a VNET.
  - **Virtual Network** Either select an existing VNET or create a new one. VMs in the same VNET can access one another by default.
  - **Subnet** Select the subnet (range of IP address from the VNET) in which to place the VM.
  - **Public IP Address** Optionally, chose to create a new public (either dynamic or static) IP address, or select None to not have a publicly accessible IP address for the VM.
  - **Network Security Group** Configure a set of inbound and outbound firewall rules that control traffic to and from the VM. Note that the default is set to allow Remote Desktop Protocol (RDP) for Windows and SSH for Linux.

- **Monitoring**
  - **Diagnostics** Choose to enable or disable diagnostic metrics for the VM. This setting enables the Azure Diagnostics extension that by default persists metric data to an Azure Storage account.
  - **Diagnostics Storage Account** Select either an existing Azure Storage account or create a new one to which diagnostic metrics are written.
- **Availability**
  - **Availability Set** Optionally, select the availability set in which to place the VM. This configuration cannot be changed once the VM is created.

> **Note** Diagnostic data (that is, ETW events, performance counters, Windows and application logs, and so on) can optionally be sent to Azure Event Hubs. It is still necessary to enable the Azure Diagnostics extension – new configuration settings are used to optionally send the data to Azure Event Hubs. For more information, please see https://azure.microsoft.com/documentation/articles/event-hubs-streaming-azure-diags-data/.

The fourth, and final, step is a review step. Once some basic platform validation is complete, you will see a summary of the VM to be created. Select the OK button to start the deployment process. It may take several minutes before the VM is fully provisioned and ready for use.

Create a virtual machine with a template

As mentioned in Chapter 1, one of the key features in the Resource Manager model is the ability to deploy full solutions, using many Azure services (or resources), in a consistent and easily repeatable manner by using templates. Azure Resource Manager templates (ARM templates) are JSON-structured files that explicitly state each Azure resource to use, related configuration properties, and any necessary dependencies. ARM templates are a great way to deploy solutions in Azure, especially solutions that include multiple resources.

As a simple example, if you want to create a solution that requires two VMs using a public load balancer, you can do that in the Azure portal. In doing so, you will need to create a storage account (or use an existing one), a virtual network, public IP addresses, an availability set, and a NIC for each VM. If you have to do this in a repeatable or automated manner, using the Azure portal may not be an optimal approach (due to risk of introducing human error into the process, speed of moving through a user interface, and so on). An alternative deployment mechanism would be to use an ARM template. The example below demonstrates using both PowerShell and Azure CLI commands to deploy the same template.

Deploying an ARM template via PowerShell

```
$resourceGroupName = "AzureEssentials2016-VM"

$location ="centralus"

$templateFilePath = "C:\Projects\azure-quickstart-templates\201-2-vms-loadbalancer-
lbrules\azuredeploy.json"

$templateParameterFilePath = "C:\Projects\azure-quickstart-templates\201-2-vms-loadbalancer-
lbrules\azuredeploy.parameters.json"

New-AzureRmResourceGroup -Name $resourceGroupName `
                         -Location $location
```

```
New-AzureRmResourceGroupDeployment -Name "My_2_VMs_with_LB" `
                                   -ResourceGroupName $resourceGroupName `
                                   -TemplateFile $templateFilePath `
                                   -TemplateParameterFile $templateParameterFilePath
```

Deploying an ARM template via the Azure CLI

```
azure resource group create –name AzureEssentials2016-VM2 --location centralus

azure group deployment create AzureEssentials2016-VM3 --template-file "C:\Projects\azure-
quickstart-templates\201-2-vms-loadbalancer-lbrules\azuredeploy.json" --parameters-file
"C:\Projects\azure-quickstart-templates\201-2-vms-loadbalancer-
lbrules\azuredeploy.parameters.json"
```

You can browse a myriad of Microsoft and community-contributed templates at https://azure.microsoft.com/documentation/templates/. The same view of the templates, lacking the integrated search capabilities, is available at https://github.com/Azure/azure-quickstart-templates. The template referenced in the example above can be found at https://github.com/Azure/azure-quickstart-templates/tree/master/201-2-vms-loadbalancer-lbrules.

Connecting to a virtual machine

After creating a new VM, one of the common next steps is to connect to the VM. Connectivity can be done by remotely accessing (for example, logging in remotely to) the VM for an interactive session or by configuring network access to allow other programs or services to communicate with the VM.

Remotely access a virtual machine

When creating a Windows VM using the Azure portal, Remote Desktop is enabled by default. This is enabled via an NSG and the automatic configuration of the appropriate inbound security rule, allowing inbound TCP traffic on port 3389 (the default RDP port). To connect to a Windows VM, select the Connect button from the VM blade, as shown in Figure 3-7.

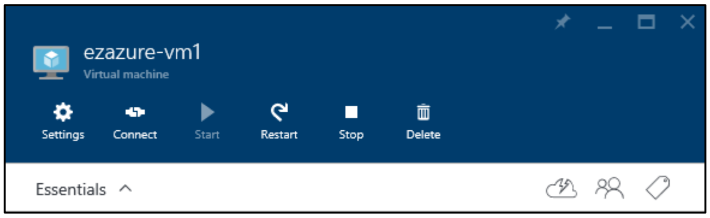


Figure 3-7 Connecting to a VM.

This will initiate a download to your local machine of a preconfigured Remote Desktop (.rdp) file. Open the RDP file and connect to the VM. You will need to provide the administrative user name and password set when initially provisioning the VM.

If a Linux VM was created, the process to connect remotely will be a bit different because you will not connect via Remote Desktop. Instead, you will connect via SSH in the standard way for Linux VMs. If you’re connecting from Windows, you will likely use an SSH client such as PuTTY.

Network connectivity

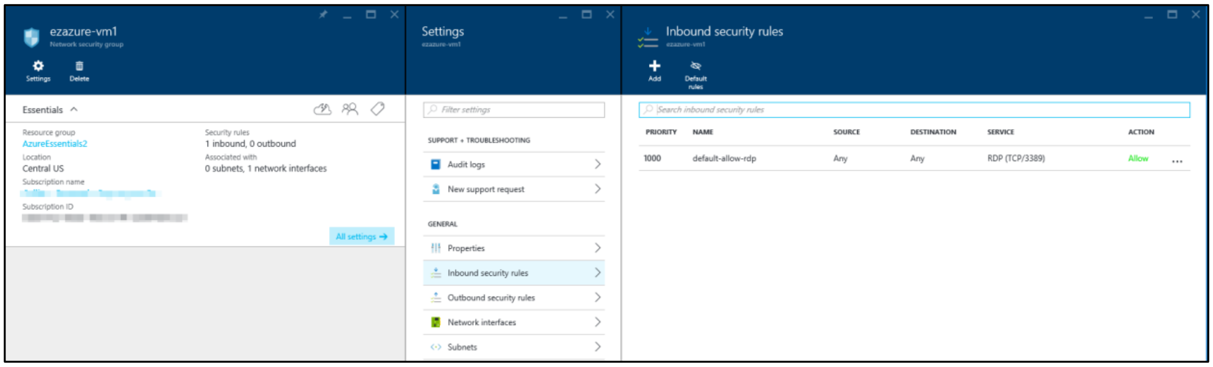
By default, Azure VMs are not able to accept requests from the Internet. To do so, a VM must be configured to permit inbound network traffic.

> **Note** Configuring network connectivity sets rules for how network traffic reaches the VM. It does not have any relation to the firewall (software or similar features) running on the VM itself. You might need to configure the server's firewall to allow traffic on the desired port and protocol.

In the Resource Manager model, a VM has inbound connectivity from the Internet if it either has a public IP address on the associated NIC or is the NAT/load-balanced target of an Azure load balancer. NSGs can further restrain that connectivity. To view the NSG rules for a VM using the Azure portal, you will need to start by examining the network interface in the Settings blade for the VM. From there, you would view the Inbound Security Rules on the NSG. There are several blades to move through when viewing this information in the Azure portal. The path would be as follows:

```
[Your VM] > Settings > Network Interfaces > [Select the NIC] > Settings (for the selected NIC) >
Network Security Group > [Select the Network Security Group] > Settings (for the selected NSG) >
Inbound Security Rules
```

In the end, you should get to a screen that looks like that shown in Figure 3-8.



Resource group
AzureEssentials2

Location
Central US

Subscription name
[redacted]

Subscription ID
[redacted]

Security rules
1 inbound, 0 outbound

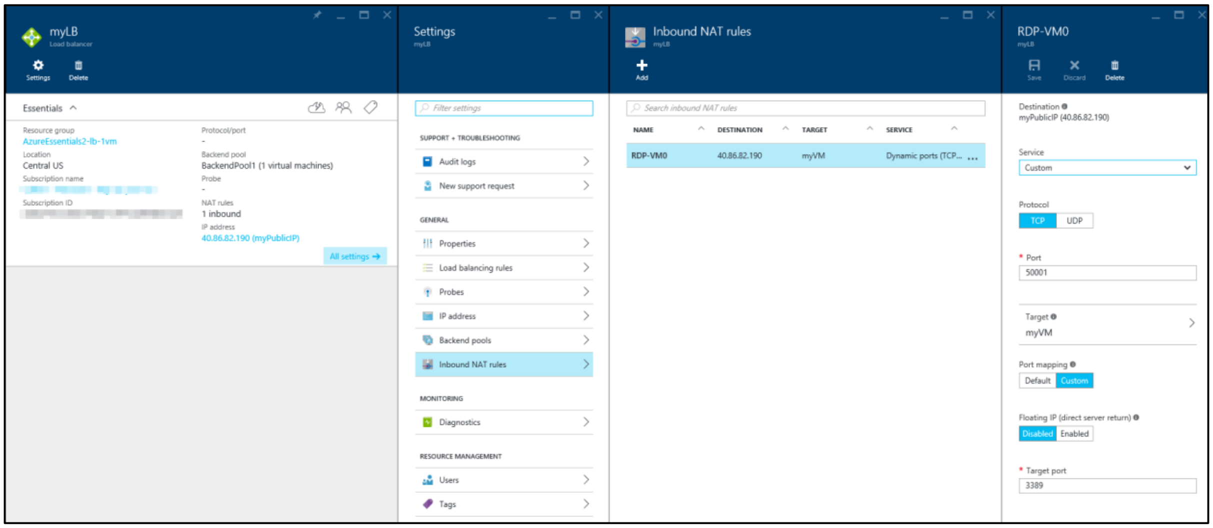
Associated with
0 subnets, 1 network interfaces

PRIORITY	NAME	SOURCE	DESTINATION	SERVICE	ACTION
1000	default-allow-rdp	Any	Any	RDP (TCP/3389)	Allow

Figure 3-8 The Inbound Security Rules for an NSG on a VM.

Another approach to viewing the NSG configuration is to use the *Get-AzureRmNetworkSecurityGroup* PowerShell cmdlet.

When using a load balancer in conjunction with one or more VMs in an availability set, the connectivity from the public Internet to the VM is controlled by inbound NAT rules and load balancing rules, as seen in Figure 3-9. The rules are part of the load balancer resource configuration, not the VM. The load balancer is configured to work with, or target, the specific VM(s).



Resource group: AzureEssentials2-lb-1vm
Location: Central US
Subscription name: [redacted]
Subscription ID: [redacted]
Protocol/port: -
Backend pool: BackendPool1 (1 virtual machines)
Probe: -
NAT rules: 1 inbound
IP address: 40.86.82.190 (myPublicIP)

NAME	DESTINATION	TARGET	SERVICE
RDP-VM0	40.86.82.190	myVM	Dynamic ports (TCP...

Destination: myPublicIP (40.86.82.190)
Service: Custom
Protocol: [✓ TCP] UDP
* Port: 50001
Target: myVM
Port mapping: Default [✓ Custom]
Floating IP (direct server return): [✓ Disabled] Enabled
* Target port: 3389

Figure 3-9 The Inbound NAT Rules for a Load Balancer resource targeting a Resource Manager VM.

For classic Azure VMs, the Azure Load Balancer exposes endpoints for an Azure cloud service. It is the configuration of the Azure Load Balancer that controls how requests from the Internet reach a specific port using a related protocol (such as TCP or UDP) on the VM. This configuration is configuring the Azure Load Balancer to allow traffic from the Internet, creating a mapping between public ports on the Azure Load Balancer and private ports on the VM.

Note NSGs can be applied to both classic VMs and Resource Manager VMs. For the purposes of this scenario on virtual machine connectivity, NSGs are not discussed for classic VMs.

Configuring and managing a virtual machine

Creating an Azure VM is only the beginning. There are several important factors that you should consider to successfully manage the VMs. Factors such as scalability, SLA, disk management, and machine maintenance are all important to consider.

The overall management of the VMs is largely the user's responsibility—you can do pretty much whatever you desire on the VM. Configuration and management of the VM can be done via numerous methods, such as manually via a Remote Desktop connection, remotely using PowerShell or PowerShell DSC (desired state configuration), or VM extensions for popular tools like Chef and Puppet. There is a wide range of choices for configuring the VM—the choice is yours.

See Also Unfortunately, all VM configuration options and approaches cannot be covered in this book. Please reference the Azure Virtual Machines documentation at https://azure.microsoft.com/documentation/services/virtual-machines/ for additional detailed information.

Disks

As mentioned earlier in this chapter, Azure VMs have two types of disks: an OS disk and a data disk. These disks are durable (or persistent) disks backed by page blobs in Azure Storage. You have several options on for configuring and using the disks for your VM.

Azure Storage uses page blobs to store the VHDs. For VMs that use Standard storage, the VHD is stored in a sparse format. This means that Azure Storage charges apply only for data within the VHD that has actually been written. Because of this, it is recommended that you use a quick format when formatting the disks. A quick format will avoid storing large ranges of zeros with the page blob, thus conserving actual storage space and saving you money. However, if the VM uses Premium storage, you are charged for the full disk size. Meaning, if you attach a P20 disk (which has a size of 512 GB) to a VM and allocate 300 GB for the drive, you are charged the full price for the P20 disk (not just the space used or allocated). Therefore, it is usually wise to allocate the full size for the drive because you're charged for it anyway

Disk caching

Azure Virtual Machines has the ability to cache access to OS and data disks. Caching potentially can reduce transactions to Azure Storage and can improve performance for certain workloads. There are three disk cache options: Read/Write, Read Only, and None.

The OS disk has two cache options: Read/Write (default) and Read Only.

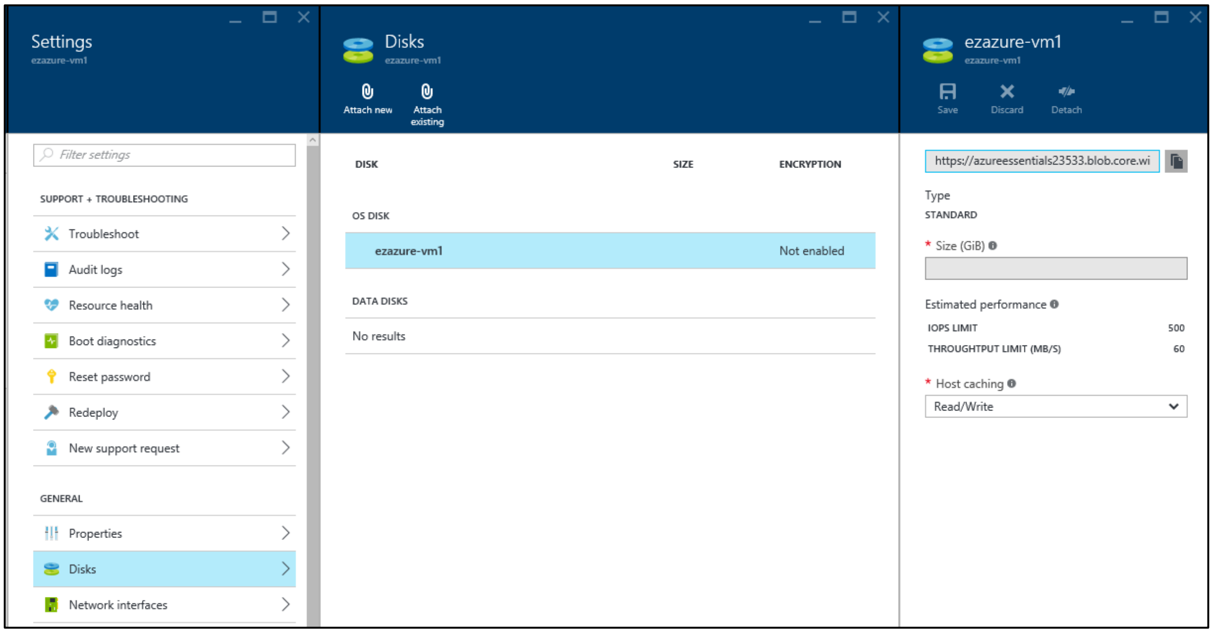
The data disk has three cache options: Read/Write, Read Only, and None (default).

You should thoroughly test the disk caching configuration for your workload to ensure it meets your performance objectives.

Attach a disk

To add a data disk to a VM, you can start with a new, empty disk or upload an existing VHD. Either can be done using the Azure portal (or using Azure PowerShell or the Azure CLI).

By browsing to the Disks options in the Settings menu, as seen in Figure 3-10, you can view all the OS and data disks that are attached to the current VM. This view also allows you to see the disk type (Standard or Premium), size, estimated performance, and cache setting.



DISK	SIZE	ENCRYPTION
OS DISK		
ezazure-vm1		Not enabled
DATA DISKS		
No results		

https://azureessentials23533.blob.core.wi

Type
STANDARD

* Size (GiB)

Estimated performance

IOPS LIMIT	500
THROUGHPUT LIMIT (MB/S)	60

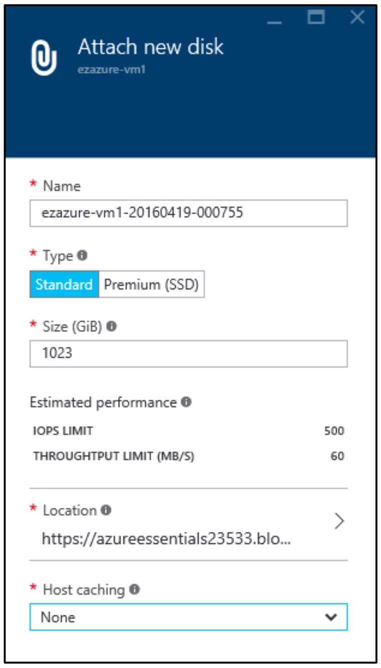
* Host caching
Read/Write

Figure 3-10 Number and size of disks.

To create and attach a new disk, first click the Disks options in the Settings menu to open the Disks blade. On this blade, you will be able to attach a new disk or attach an existing disk.

To attach a new disk, click Attach New. From the resulting Attach New Disk blade, as seen in Figure 3-11, you will be able to provide several key settings:

- **Name** Provide your own or accept the default.
- **Type** A disk backed by either Azure Standard Storage or Azure Premium Storage.
- **Size** The size of the new data disk (VHD).
- **Location** The Azure Storage account and blob container that will store your new data disk. You can either select an existing storage account and container or create a new storage account.
- **Host Caching** The cache option to use for the data disk.



Attach new disk
ezazure-vm1

* Name
ezazure-vm1-20160419-000755

* Type
Standard | Premium (SSD)

* Size (GiB)
1023

Estimated performance

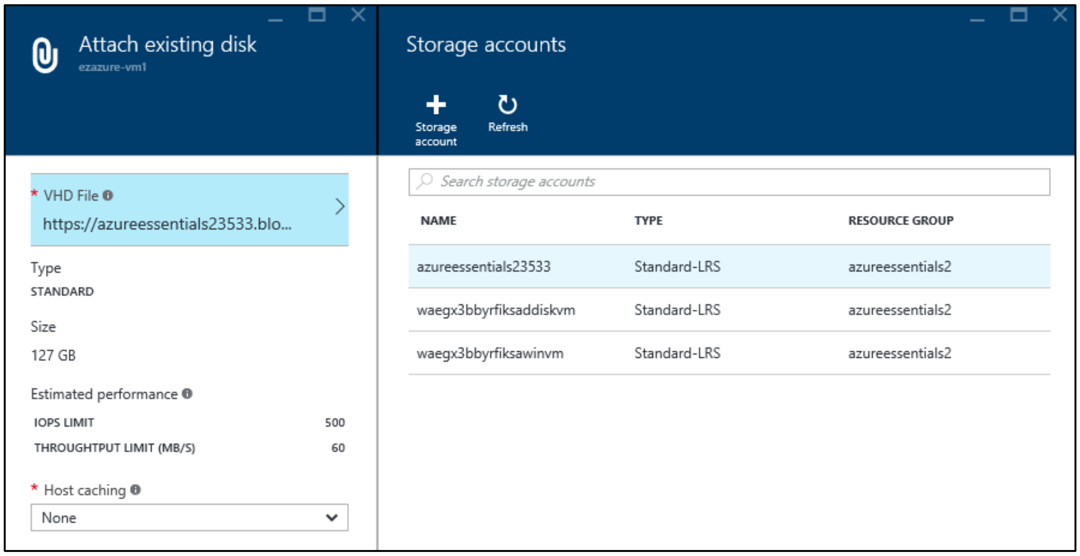
IOPS LIMIT	500
THROUGHPUT LIMIT (MB/S)	60

* Location
https://azureessentials23533.blo...

* Host caching
None

Figure 3-11 Attach a new data disk.

To attach an existing data disk, click Attach Existing on the Disks blade. The resulting Attach Existing Disk blade will present an option to select an existing VHD from your Azure Storage account, as you can see in Figure 3-12. You can use your favorite Azure Storage management tool to upload an existing VHD to a blob container in the desired storage account (be sure that VHD is set as a page blob and not a block blob).



Attach existing disk
ezazure-vm1

* VHD File
https://azureessentials23533.blo...

Type
STANDARD

Size
127 GB

Estimated performance

IOPS LIMIT	500
THROUGHTPUT LIMIT (MB/S)	60

* Host caching
None

Storage accounts

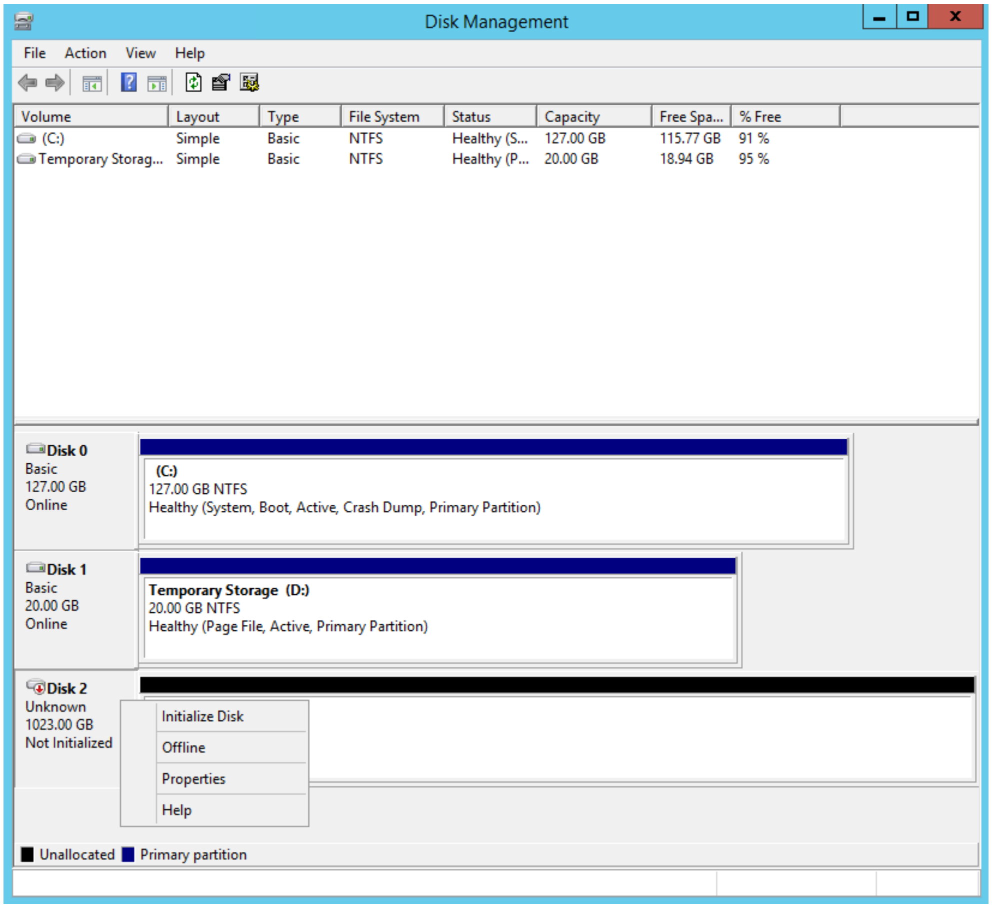
NAME	TYPE	RESOURCE GROUP
azureessentials23533	Standard-LRS	azureessentials2
waegx3bbyrfiksaddiskvm	Standard-LRS	azureessentials2
waegx3bbyrfiksawinvm	Standard-LRS	azureessentials2

Figure 3-12 The Attach Existing Disk blade.

Formatting disks

Once the data disks are attached to the Azure VM, each data disk needs to be formatted (or initialized), just like a disk on a physical server. Because Standard storage disks are billed only for the occupied space, it is recommended that you use a quick format when formatting the disks. A quick format will avoid storing large ranges of zeros with the page blob, thus conserving actual storage space and saving you money.

To format the disk(s), remotely connect to the VM. For a Windows VM, once you are connected and logged into the VM, open Disk Management. Disk Management is a native Windows application that allows you to view the disks and format any unallocated disks. As can be seen in Figure 3-13, proceed by right-clicking the unallocated disk and selecting Initialize Disk.



Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(C:)	Simple	Basic	NTFS	Healthy (S...	127.00 GB	115.77 GB	91 %
Temporary Storag...	Simple	Basic	NTFS	Healthy (P...	20.00 GB	18.94 GB	95 %

Disk 0
Basic
127.00 GB
Online

(C:)
127.00 GB NTFS
Healthy (System, Boot, Active, Crash Dump, Primary Partition)

Disk 1
Basic
20.00 GB
Online

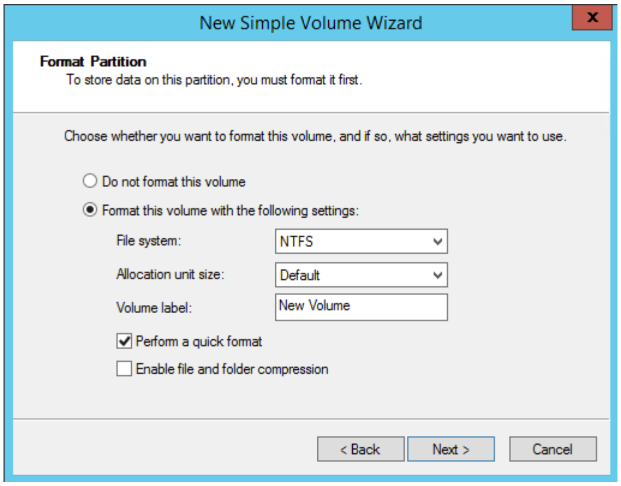
Temporary Storage (D:)
20.00 GB NTFS
Healthy (Page File, Active, Primary Partition)

Disk 2
Unknown
1023.00 GB
Not Initialized

Figure 3-13 Windows Disk Management.

Complete the wizard to initialize the disk. Once the disk has been initialized, you can proceed with formatting the disk.

1. Right-click the disk and select New Simple Volume. The New Simple Volume Wizard should open.
2. Continue through the wizard, selecting the desired volume size and drive letter.
3. When presented with an option to format the volume, be sure to select Perform A Quick Format.



New Simple Volume Wizard

Format Partition
To store data on this partition, you must format it first.

Choose whether you want to format this volume, and if so, what settings you want to use.

○ Do not format this volume

● Format this volume with the following settings:

File system: NTFS

Allocation unit size: Default

Volume label: New Volume

☑ Perform a quick format

□ Enable file and folder compression

4. Finish the steps in the wizard to start formatting the disk.

> **See Also** For a step-by-step walkthrough on initializing data disks for a Linux VM, please see https://azure.microsoft.com/documentation/articles/virtual-machines-linux-classic-attach-disk/#how-to-initialize-a-new-data-disk-in-linux.

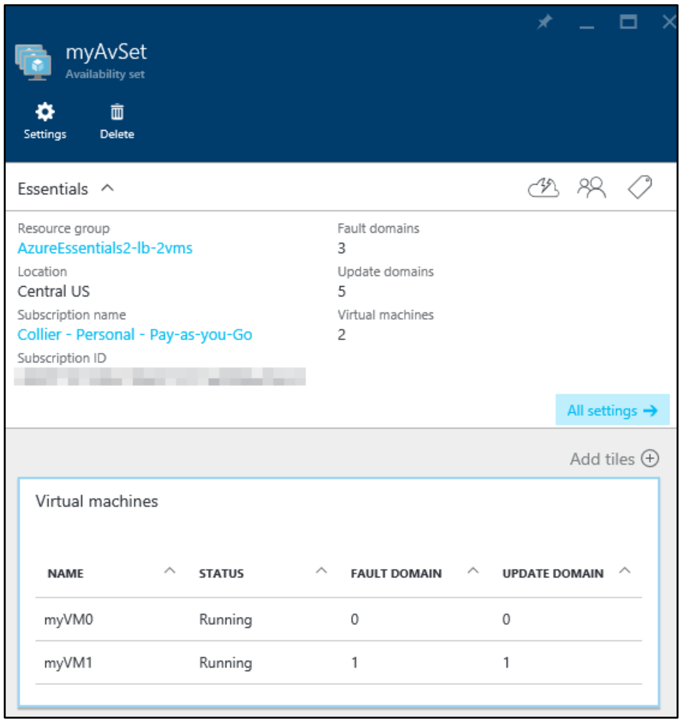
Disk performance

Another factor to be aware of with Azure VM disks is IOPS. At the time of this writing, each data disk backed by Azure Standard Storage has a maximum of 500 IOPS and 60 MB/s (for Standard-tier VMs). For Azure VMs backed by Azure Premium Storage (that is DS, DSv2, F, and GS-series VMs), there is currently a maximum of 5,000 IOPS and 200 MB/s per disk, depending on the specific tier of Azure Premium Storage used. This might or might not be sufficient for the desired workload. You should conduct performance tests to ensure the disk performance is sufficient. If it is not, consider adding disks and creating a disk array via Storage Spaces on Windows or mdadm on Linux. Because Azure Storage keeps three copies of all data, it is only necessary to use RAID 0.

> **See Also** For more information on advanced configuration of Azure VM disks, including striping and Storage Spaces, please review the Microsoft Azure whitepaper available at http://msdn.microsoft.com/library/azure/dn133149.aspx. Although the referenced whitepaper is specific to running SQL Server on an Azure VM, the disk configuration details are common across a multitude of workloads.

Fault domains and update domains

For Resource Manager VMs, you can view the update and fault domains by looking at the Availability Set resource associated with the VMs, as seen in Figure 3-14.



Resource group: AzureEssentials2-lb-2vms
Location: Central US
Subscription name: Collier - Personal - Pay-as-you-Go
Subscription ID: [redacted]
Fault domains: 3
Update domains: 5
Virtual machines: 2

Virtual machines

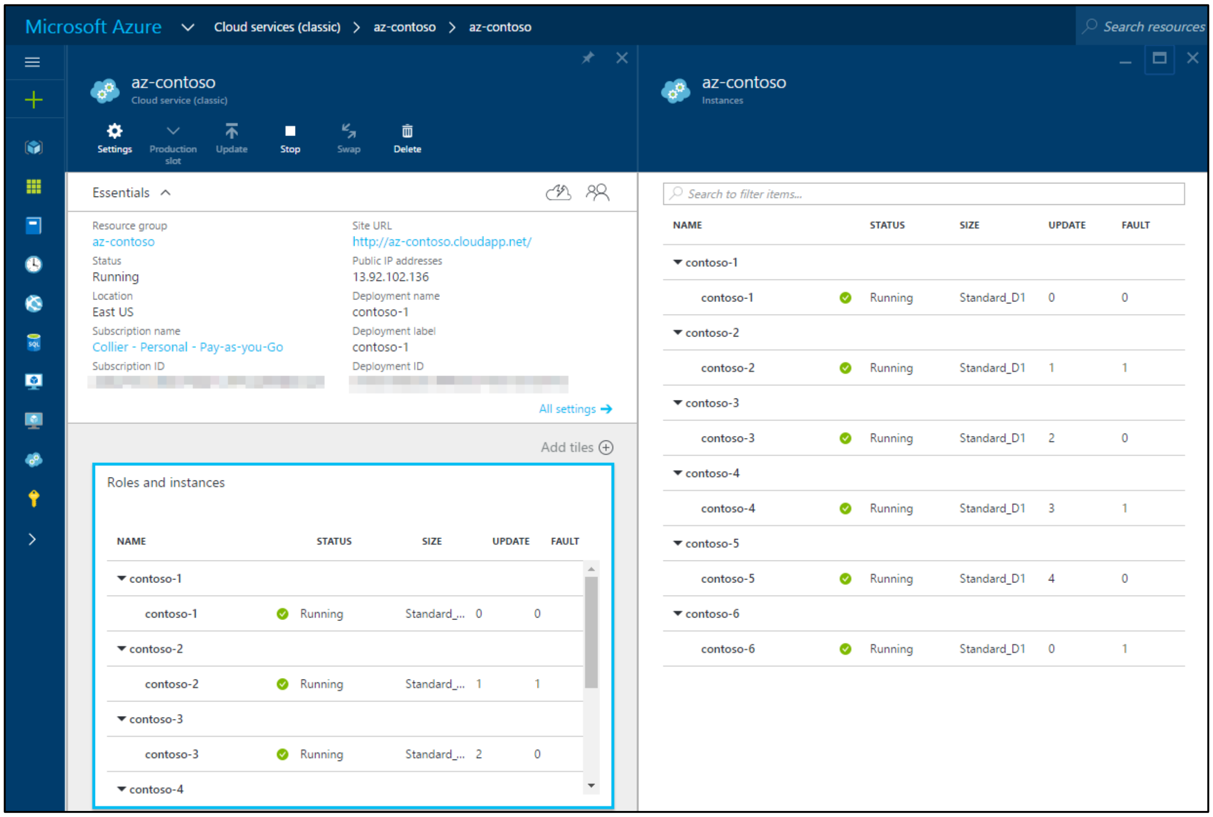
NAME	STATUS	FAULT DOMAIN	UPDATE DOMAIN
myVM0	Running	0	0
myVM1	Running	1	1

Figure 3-14 Update and fault domains for Resource Manager VMs.

If there is an existing availability set, the VM can be placed within the availability set as part of the VM provisioning process. If there is not an existing availability set, one will need to be created.

> **Note** At the time of this writing, for Resource Manager VMs, the VM must be added to the desired availability set at the time the VM is created. The VM cannot be added to the availability set at a later time.

You can view the update and fault domains used for your classic VMs by looking at the related Cloud Service (Classic) in the Azure portal. As seen in Figure 3-15, the first five VMs are each placed in a different update domain, and the sixth VM is placed in update domain 0.



Resource group
az-contoso

Status
Running

Location
East US

Subscription name
Collier - Personal - Pay-as-you-Go

Subscription ID
[redacted]

Site URL
http://az-contoso.cloudapp.net/

Public IP addresses
13.92.102.136

Deployment name
contoso-1

Deployment label
contoso-1

Deployment ID
[redacted]

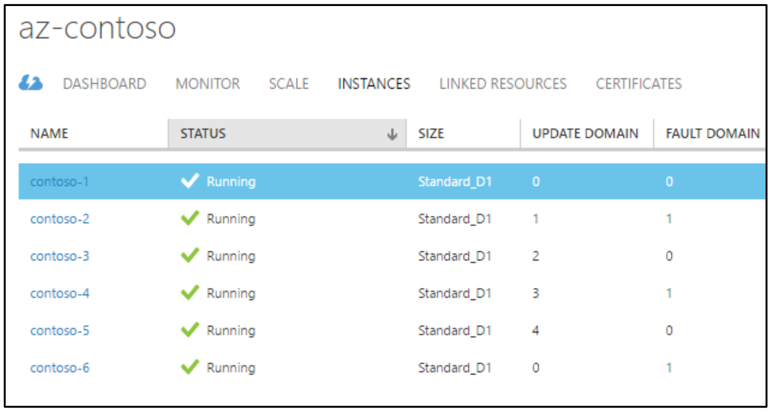
Roles and instances

NAME	STATUS	SIZE	UPDATE	FAULT
▾ contoso-1				
contoso-1	✓ Running	Standard_...	0	0
▾ contoso-2				
contoso-2	✓ Running	Standard_...	1	1
▾ contoso-3				
contoso-3	✓ Running	Standard_...	2	0
▾ contoso-4				

NAME	STATUS	SIZE	UPDATE	FAULT
▾ contoso-1				
contoso-1	✓ Running	Standard_D1	0	0
▾ contoso-2				
contoso-2	✓ Running	Standard_D1	1	1
▾ contoso-3				
contoso-3	✓ Running	Standard_D1	2	0
▾ contoso-4				
contoso-4	✓ Running	Standard_D1	3	1
▾ contoso-5				
contoso-5	✓ Running	Standard_D1	4	0
▾ contoso-6				
contoso-6	✓ Running	Standard_D1	0	1

Figure 3-15 VMs, update domains, and fault domains for classic VMs.

A similar view can be found in the Azure classic portal, as shown in Figure 3-16.



NAME	STATUS	SIZE	UPDATE DOMAIN	FAULT DOMAIN
contoso-1	✓ Running	Standard_D1	0	0
contoso-2	✓ Running	Standard_D1	1	1
contoso-3	✓ Running	Standard_D1	2	0
contoso-4	✓ Running	Standard_D1	3	1
contoso-5	✓ Running	Standard_D1	4	0
contoso-6	✓ Running	Standard_D1	0	1

Figure 3-16 VMs, update domains, and fault domains for classic VMs in the Azure classic portal.

Image capture

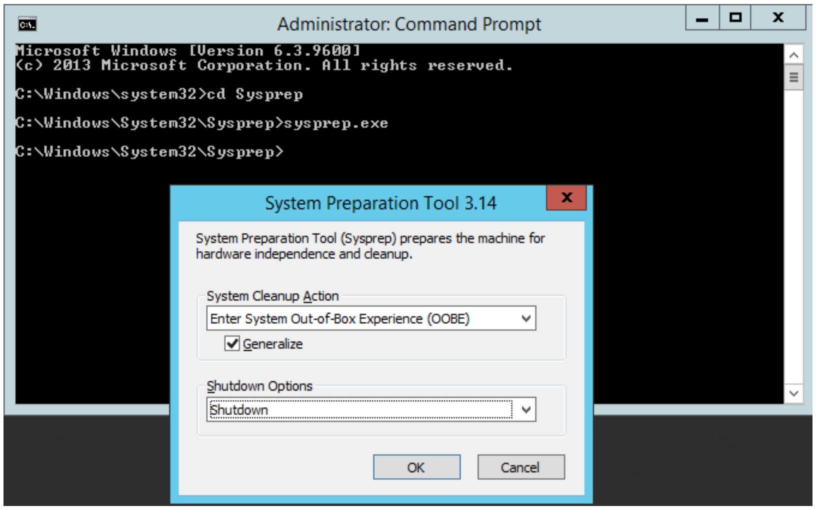
Once you have your new Azure VM configured as you would like it, you might want to create a clone of the VM. For example, you might want to create several more VMs using the one you just created as a template. You do this by capturing the VM and creating a generalized VM image. When you create a VM image, you capture not only the OS disk, but also any attached data disks.

When you capture the VM to use it as a template for future VMs, you will no longer be able to use the original VM (the original source) because it is deleted after the capture is completed. For classic VMs, you will find a template image available for use in your Virtual Machine gallery in the Azure classic portal. As of this writing, there is no view available in the Azure portal for viewing images related to Resource Manager VMs. Instead, you will need look for the image in the same storage account as the original VM (most often the image will be stored at a path similar to *https://[storage_account].blob.core.windows.net/system/Microsoft.Compute/Images/[container_name]/[template_prefix]-osDisk.xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.vhd.*

Capture a Windows VM in the Resource Manager model

To capture a Windows VM in the Resource Manager model, you will use Azure PowerShell, the Azure CLI, or the Azure Resource Explorer tool. Capturing a VM is not yet possible in the Azure portal. To capture a Windows VM, complete the following steps:

1. Connect to the VM using Remote Desktop (as discussed earlier in this chapter).
2. Open a command prompt window as the administrator.
3. Navigate to the *%windir%/system32/sysprep* directory and then run *Sysprep.exe*.
4. In the System Preparation Tool, perform the following actions:
   a. From the System Cleanup Action list, select Enter System Out-Of-Box Experience (OOBE).
   b. Select the Generalize check box.
   c. In the Shutdown Options drop-down list, select Shutdown.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd Sysprep

C:\Windows\System32\Sysprep>sysprep.exe

C:\Windows\System32\Sysprep>
```

System Preparation Tool 3.14

System Preparation Tool (Sysprep) prepares the machine for hardware independence and cleanup.

System Cleanup Action: Enter System Out-of-Box Experience (OOBE)

☑ Generalize

Shutdown Options: Shutdown

5. The VM will run sysprep. If you are still connected to the VM via RDP, you will be disconnected when it begins to shut down. Watch the VM in the Azure portal until it completely shuts down and shows a status of Stopped.
6. Open PowerShell and log into your Azure account using the *Login-AzureRMAccount* cmdlet. Optionally, select the necessary Azure subscription using the *Select-AzureRMSubscription* cmdlet.
7. Stop and deallocate the VM's resources by using the *Stop-AzureRmVM* cmdlet, as seen in the example below. The VM's status will change from Stopped to Stopped (Deallocated).

```
Stop-AzureRmVM -ResourceGroupName AzureEssentials2-vm -Name ezazure3
```

8. Set the status of the VM to Generalized by using the *Set-AzureRmVM* cmdlet, as seen in the example below.

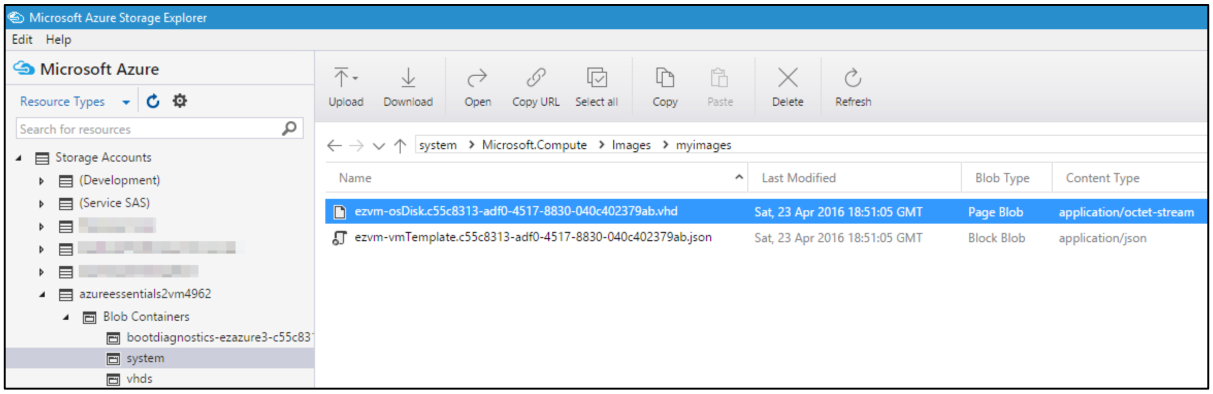
```
Set-AzureRmVM -ResourceGroupName AzureEssentials2-vm -Name ezazure3 -Generalized
```

Tip View the VM status using the *Get-AzureRmVm* cmdlet, as shown below. This will show you a *VM generalized* status when the previous command is complete. The *VM generalized* status will not appear in the Azure portal.

```
(Get-AzureRmVM -ResourceGroupName AzureEssentials2-vm -Name ezazure3 -
Status).Statuses
```

9. Capture the VM, placing the image in an Azure Blob storage container folder, by executing the *Save-AzureRmVMImage* cmdlet, as seen in the example below. Note that the value of the *DestinationContainerName* parameter is not a top-level blob container, but a folder under the System container, as can be seen in Figure 3-17. You can also see the full path to the image file by looking in the saved JSON file under the resource\storageProfile\osDisk\image\uri location.

```
Save-AzureRmVMImage -ResourceGroupName AzureEssentials2-vm -VMName ezazure3 -
DestinationContainerName myimages -VHDNamePrefix ezvm -Path C:\temp\imagetemplate.json
```



Name	Last Modified	Blob Type	Content Type
ezvm-osDisk.c55c8313-adf0-4517-8830-040c402379ab.vhd	Sat, 23 Apr 2016 18:51:05 GMT	Page Blob	application/octet-stream
ezvm-vmTemplate.c55c8313-adf0-4517-8830-040c402379ab.json	Sat, 23 Apr 2016 18:51:05 GMT	Block Blob	application/json

Figure 3-17 The VHD associated with the saved VM image.

Note The saved JSON file is a valid ARM template file that can be used to create a new Azure VM based on the saved image. You will need to add any additional required components, such as an NIC.

With the image safely stored in Azure Storage, you can use this image as the basis for new Azure VMs. To do so, you would use the *Set-AzureRmVMOSDisk* cmdlet, specifying the path to the saved VHD in the *SourceImageUri* parameter. Keep in mind that the image and the OS disk must be in the same storage account. If they are not, you will need to copy the image VHD to the desired storage account. A full example can be seen below (replace with your values as appropriate).

Creating a new Azure VM from a captured VM image

```
$resourceGroupName = "EZAzureVM-2016"
$location = "centralus"
$capturedImageStorageAccount = "azureessentials2vm4962"

$capturedImageUri
=https://azureessentials2vm4962.blob.core.windows.net/system/Microsoft.Compute/Images/myimages/ez
vm-osDisk.c55c8313-adf0-4517-8830-040c402379ab.vhd

$catpuredImageStorageAccountResourceGroup = "AzureEssentials2-vm"

# Create the new resource group.

New-AzureRmResourceGroup -Name $resourceGroupName -Location $location
```

```
# !!!! This example assumes the new VM is in a different resource group and storage account from
the captured VM. !!!!

$srcKey = Get-AzureRmStorageAccountKey -StorageAccountName $capturedImageStorageAccount -
ResourceGroupName $catpuredImageStorageAccountResourceGroup

$srcContext = New-AzureStorageContext -StorageAccountName $capturedImageStorageAccount -
StorageAccountKey $srcKey.Key1

# **** Create the Network Resources ****

$publicIp = New-AzureRmPublicIpAddress -Name "MyPublicIp01" `
                -ResourceGroupName $resourceGroupName `
                -Location $location -AllocationMethod Dynamic

$subnetConfiguration = New-AzureRmVirtualNetworkSubnetConfig -Name "MySubnet" `
                           -AddressPrefix "10.0.0.0/24"

$virtualNetworkConfiguration = New-AzureRmVirtualNetwork -Name "MyVNET" `
                                   -ResourceGroupName $resourceGroupName `
                                   -Location $location `
                                   -AddressPrefix "10.0.0.0/16" `
                                   -Subnet $subnetConfiguration

$nic = New-AzureRmNetworkInterface -Name "MyServerNIC01" `
          -ResourceGroupName $resourceGroupName `
          -Location $location `
          -SubnetId $virtualNetworkConfiguration.Subnets[0].Id `
          -PublicIpAddressId $publicIp.Id


# ****  Create the new Azure VM ****

# Get the admin credentials for the new VM

$adminCredential = Get-Credential

# Get the storage account for the captured VM image
$storageAccount = New-AzureRmStorageAccount -ResourceGroupName $resourceGroupName -Name
"ezazurevm2016" -Location $location -Type Standard_LRS

# Copy the captured image from the source storage account to the destination storage account
$destImageName = $capturedImageUri.Substring($capturedImageUri.LastIndexOf('/') + 1)

New-AzureStorageContainer -Name "images" -Context $storageAccount.Context

Start-AzureStorageBlobCopy -AbsoluteUri $capturedImageUri -DestContainer "images" -DestBlob
$destImageName -DestContext $storageAccount.Context -Context $srcContext -Verbose -Debug

Get-AzureStorageBlobCopyState -Context $storageAccount.Context -Container "images" -Blob
$destImageName -WaitForComplete

# Build the URI for the image in the new storage account
$imageUri = '{0}images/{1}' -f $storageAccount.PrimaryEndpoints.Blob.ToString(), $destImageName




# Set the VM configuration details
$vmConfig = New-AzureRmVMConfig -VMName "ezazurevm10" -VMSize "Standard_D1"

# Set the operating system details
$vm = Set-AzureRmVMOperatingSystem -VM $vmConfig -Windows -ComputerName $vmConfig.Name -
Credential $adminCredential -TimeZone "Eastern Standard Time" -ProvisionVMAgent -EnableAutoUpdate
```

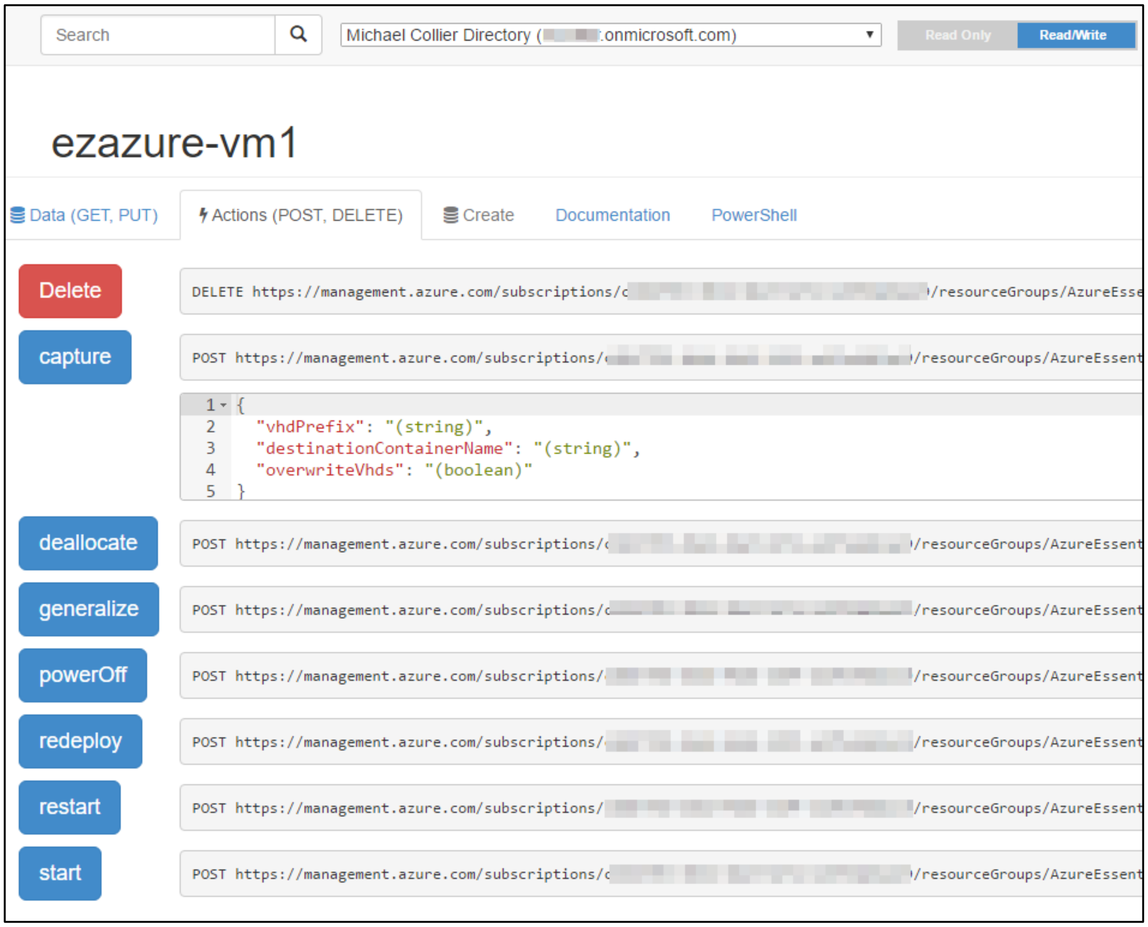
```
# Set the NIC
$vm = Add-AzureRmVMNetworkInterface -VM $vm -Id $nic.Id

# Create the OS disk URI
$osDiskUri = '{0}vhds/{1}_{2}.vhd' -f $storageAccount.PrimaryEndpoints.Blob.ToString(),
$vm.Name.ToLower(), ($vm.Name + "_OSDisk")

# Configure the OS disk to use the previously saved image
$vm = Set-AzureRmVMOSDisk -vm $vm -Name $vm.Name -VhdUri $osDiskUri -CreateOption FromImage -
SourceImageUri $imageUri -Windows

# Create the VM
New-AzureRmVM -ResourceGroupName $resourceGroupName -Location $location -VM $vm
```

For Linux VMs, the capture process is similar. Although you can use PowerShell to capture the VM, a common approach is to use the Azure CLI. You would use three basic Azure CLI commands:

```
azure vm stop -g <resource group name> -n <vm name>
azure vm generalize -g <resource group name> -n <vm name>
azure vm capture <resource group name> <vm name> <vhd prefix> -t <template file name>
```

See Also For a detailed walkthrough of using the Azure CLI to capture a Linux VM, please see https://azure.microsoft.com/documentation/articles/virtual-machines-linux-capture-image/.

As an alternative to using PowerShell or the Azure CLI, you can use the Azure Resource Explorer tool available at https://resources.azure.com. This tool allows you to work against the Azure Resource Manager (ARM) native REST APIs in a user-friendly manner. After signing into your Azure account and setting the tool to Read/Write mode to allow PUT, POST, and DELETE operations (default is Read Only, allowing GET operations), you will need to find the VM you want to capture. Once you've located the VM, go the tab for Actions (POST/DELETE). There, you will find options, as seen in Figure 3-18, to deallocate, generalize, and capture the VM. Capturing the VM will create the VHD for the image and the JSON template file, just as executing the *Save-AzureRmVMImage* cmdlet or *azure vm capture* command would.



```
DELETE https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEsse
POST https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEssent
{
  "vhdPrefix": "(string)",
  "destinationContainerName": "(string)",
  "overwriteVhds": "(boolean)"
}
POST https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEssent
POST https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEssent
POST https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEssent
POST https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEssent
POST https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEssent
POST https://management.azure.com/subscriptions/[redacted]/resourceGroups/AzureEssent
```

Figure 3-18 Capture a VM using the Azure Resource Explorer tool.

For more information, including details on using the Azure Resource Explorer tool, please refer to the tutorial available at https://azure.microsoft.com/documentation/articles/virtual-machines-windows-capture-image/.

Capture a Windows VM in the classic model

Similarly, in the classic model, there are several steps you will need to follow to capture a VM so it is available for use as a template image. The majority of the steps are the same as in the Resource Manager model. Once the VM has executed the sysprep process (or Linux equivalent), you will be able to initiate the capture process from within the Azure classic portal. Once the capture process is complete, the image will appear in your Virtual Machine gallery, under My Images. You can now use this image to create a new VM instance

Scaling Azure Virtual Machines

As with most Azure services, Azure Virtual Machines follow a scale out, not scale up, model. This means it is preferable to deploy more instances of the same configuration than to add larger, more powerful machines. The approach for scaling out VMs varies depending on whether you're working with classic VMs or Resource Manager VMs.

Resource Manager virtual machines

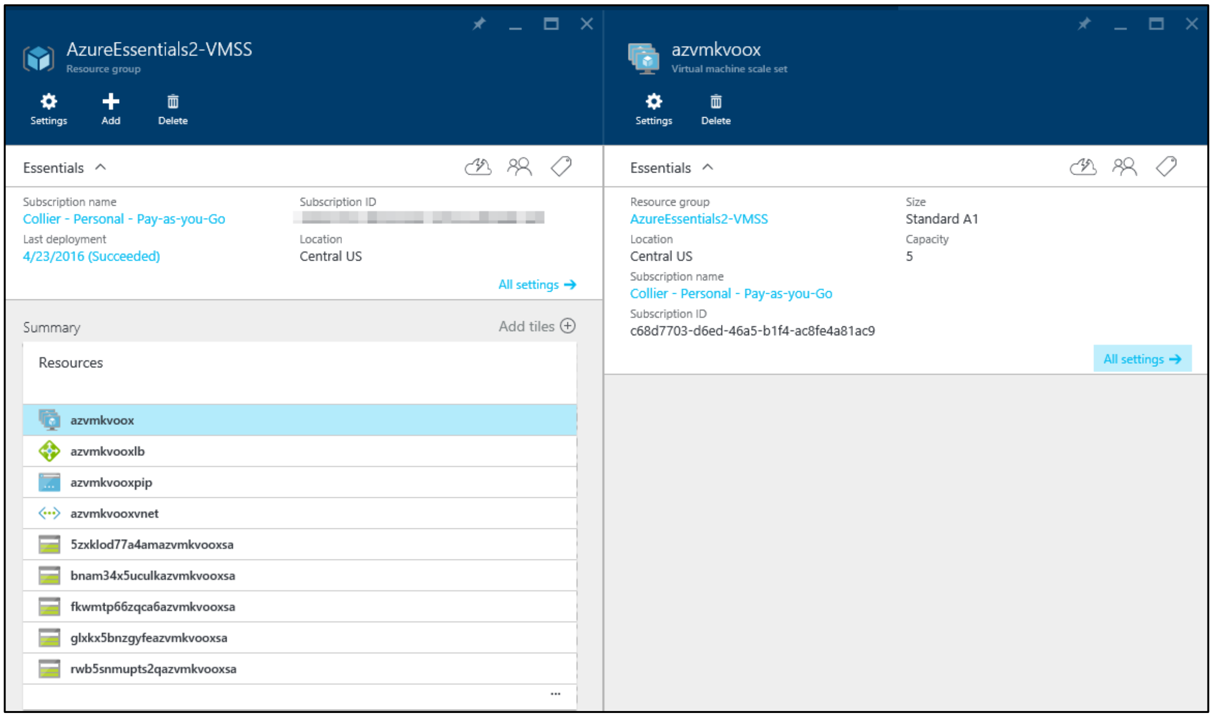
In the Resource Manager model, you don't (typically) scale out VMs in an automated way—at least not how you would with VMs in the classic model. Instead, a different Azure resource construct is used for scaling out VMs: Azure Virtual Machine Scale Sets (often abbreviated as VMSS).

Virtual Machine Scale Sets are a relatively new Azure compute option for deploying and managing a set of identical VMs. You configure all VMs in a scale set in an identical manner. You configure the VM image to be used (operating system configuration, software installed on the VM, and so on) and let Azure provision the desired number of identical VMs (based on the provided image). The VMs in a scale set can run either a Windows or a Linux operating system. Scaling with VMSS does not require the preprovisioning of VMs within an availability set (like autoscale for classic VMs does). At the time of this writing, you can have up to 100 VMs in a VM scale set.

It should be noted that when working with VMSS, there is no data disk available (as you may have with a regular Azure VM). Data should be stored on either the OS disk or an external data store such as Azure Table, File, or Blob storage; Azure SQL Database; Azure DocumentDB, and so on.

VMSS can be provisioned either via the Azure portal or ARM templates. Using the ARM template approach for working with VMSS is likely to be the most common approach because doing so offers many more features than are currently available in the Azure portal. For instance, you can configure autoscale rules relatively easily using the ARM template. Such configuration is not yet available within the Azure portal.

Once the VMSS is created, as can be seen in Figure 3-19, you can see that it contains several familiar constructs, such as a load balancer, virtual network, IP address, and multiple Azure Storage accounts.


AzureEssentials2-VMSS
Resource group

Subscription name
Collier - Personal - Pay-as-you-Go
Last deployment
4/23/2016 (Succeeded)
Subscription ID
[redacted]
Location
Central US

Resources

- azvmkvoox
- azvmkvooxlb
- azvmkvooxpip
- azvmkvooxvnet
- 5zxklod77a4amazvmkvooxsa
- bnam34x5uculkazvmkvooxsa
- fkwmtp66zqca6azvmkvooxsa
- glxkx5bnzgyfeazvmkvooxsa
- rwb5snmupts2qazvmkvooxsa
- ...

azvmkvoox
Virtual machine scale set

Resource group
AzureEssentials2-VMSS
Location
Central US
Subscription name
Collier - Personal - Pay-as-you-Go
Subscription ID
c68d7703-d6ed-46a5-b1f4-ac8fe4a81ac9
Size
Standard A1
Capacity
5

Figure 3-19 A resource group containing assets related to a new VMSS.

VMSS are the preferred way to implement a scale-out compute cluster in Azure. In fact, Azure uses VMSS to host higher-level services such as Azure Batch, Azure Service Fabric, and Azure Container Service.

See Also For more information on Azure VMSS, please refer to the documentation available at https://azure.microsoft.com/documentation/articles/virtual-machine-scale-sets-overview/.

A complete step-by-step tutorial for creating a Windows VMSS is available at https://azure.microsoft.com/documentation/articles/virtual-machine-scale-sets-windows-autoscale/. Additionally, a full sample ARM template can be found at https://github.com/Azure/azure-quickstart-templates/tree/master/201-vmss-windows-autoscale.

Similarly, a tutorial for Linux VMSS can be found at https://azure.microsoft.com/documentation/articles/virtual-machine-scale-sets-linux-autoscale/.

Classic virtual machines

In the classic model, before VMs can be scaled (out or in), the instances must be placed within an availability set. When determining the scale-out approach for VMs, it is important to determine the maximum number of VMs because that maximum number of VMs must be created, configured, and placed into the availability set. When it comes time to scale out, the VMs within the availability set are used to fulfill the scale-out needs. VMs within an availability set should all be the same size to take advantage of Azure's autoscale feature.

CHAPTER 4

Azure Storage

Microsoft Azure Storage is a Microsoft-managed service that provides durable, scalable, and redundant storage. Microsoft takes care of maintenance and handles critical problems for you. An Azure subscription can host up to 100 storage accounts, each of which can hold 500 TB. If you have a business case, you can talk to the Azure Storage team and get approval for up to 250 storage accounts in a subscription.

Azure Storage consists of four data services: Blob storage, File storage, Table storage, and Queue storage. Blob storage supports both standard and premium storage, with premium storage using only SSDs for the fastest performance possible. Another new feature added in 2016 is cool storage, allowing you to store large amounts of rarely accessed data for a lower cost.

In this chapter, we look at the four Azure Storage services. We talk about each one, discuss what they are used for, and show how to create storage accounts and manage the data objects. We'll also touch briefly on securing your applications' use of Azure Storage.

Storage accounts

This reference table shows the various kinds of storage accounts and what objects are used with each.

Type of storage account	General-purpose Standard storage account	General-purpose Premium storage account	Blob storage account, hot and cool access tiers
Services supported	Blob, File, Table, Queue Services	Blob service	Blob service
Types of blobs supported	Block blobs, page blobs, append blobs	Page blobs	Block blobs and append blobs

You can view your data objects using one of a number of storage explorers, each of which has different capabilities. For your convenience, Microsoft has a page listing several of these, including its own: https://azure.microsoft.com/documentation/articles/storage-explorers/. While you can view and update some data in the Azure portal, the customer experience is not complete. For example, you cannot upload blobs or add and view messages in a queue. In this chapter, we use the Azure portal, Visual Studio Cloud Explorer, and PowerShell to access the data.

Note After this chapter was completed, the Microsoft Azure Storage Explorer team released a new version that supports all four types of storage objects—blobs, files, tables, and queues. This is a free multi-platform tool that you can download from here: http://storageexplorer.com/

General-purpose storage accounts

There are two kinds of general-purpose storage accounts.

Standard storage

The most widely used storage accounts are Standard storage accounts, which can be used for all four types of data—blobs, files, tables, and queues. Standard storage accounts use magnetic media to store data.

Premium storage

Premium storage provides high-performance storage for page blobs and specifically virtual hard disks (VHDs). Premium storage accounts use SSD to store data. Microsoft recommends using Premium storage for all of your virtual machines (VMs).

Blob storage accounts

The Blob storage account is a specialized storage account used to store block blobs and append blobs. You can't store page blobs in these account;, therefore, you can't store VHD files. These accounts allow you to set an access tier to Hot or Cool; the tier can be changed at any time.

The hot access tier is used for files that are accessed frequently. For blobs stored in the hot access tier, you pay a higher cost for storing the blobs, but the cost for accessing the blobs is much lower.

The cool access tier is used for files that are accessed infrequently. For blobs stored in the cool access tier, you pay a higher cost for accessing the blobs, but the cost of storage is much lower.

Storage services

Azure Storage supports four kinds of objects that can be stored—blobs, files (on a file share), tables, and queues. Let's take a closer look at each one of these.

Blob storage

The word *blob* is an acronym for binary large object. Blobs are basically files like those that you store on your computer (or tablet, mobile device, etc.). They can be pictures, Microsoft Excel files, HTML files, virtual hard disks (VHDs)—pretty much anything.

The Azure Blob service gives you the ability to store files and access them from anywhere in the world by using URLs, the REST interface, or one of the Azure SDK storage client libraries. Storage client libraries are available for multiple languages, including .NET, Node.js, Java, PHP, Ruby, and Python. To use the Blob service, you have to create a storage account. Once you have a storage account, you can create containers, which are similar to folders, and then put blobs in the containers. You can have an unlimited number of containers in a storage account and an unlimited number of blobs in each container, up to the maximum size of a storage account, which is 500 TB. The Blob service supports only a single-level hierarchy of containers; in other words, containers cannot contain other containers.

Azure Storage supports three kinds of blobs: block blobs, page blobs, and append blobs.

- Block blobs are used to hold ordinary files up to 195 GB in size (4 MB × 50,000 blocks). The primary use case for block blobs is the storage of files that are read from beginning to end, such as media files or image files for websites. They are named *block blobs* because files larger than 64 MB must be uploaded as small blocks, which are then consolidated (or committed) into the final blob.

- Page blobs are used to hold random-access files up to 1 TB in size. Page blobs are used primarily as the backing storage for the VHDs used to provide durable disks for Azure Virtual Machines (Azure VMs), the IaaS feature in Azure Compute. They are named *page blobs* because they provide random read/write access to 512-byte pages.

- Append blobs are made up of blocks like block blobs, but they are optimized for append operations. These are frequently used for logging information from one or more sources into the same blob. For example, you might write all of your trace logging to the same append blob for an application running on multiple VMs. A single append blob can be up to 195 GB.

Blobs are addressable through a URL, which has the following format:

https://[storage account name]/blob.core.windows.net/[container]/[blob name]

The Blob service supports only a single physical level of containers. However, it supports the simulation of a file system with folders within the containers by allowing blob names to contain the '/' character. The client APIs provide support to traverse this simulated file system. For example, if you have a container called *animals* and you want to group the animals within the container, you could add blobs named cats/tuxedo.png, cats/marmalade.png, and so on. The URL would include the entire blob name including the "subfolder," and it would end up looking like this:

https://mystorage.blob.core.windows.net/animals/cats/tuxedo.png

https://mystorage.blob.core.windows.net/animals/cats/marmalade.png

When looking at the list of blobs using a storage explorer tool, you can see either a hierarchical directory tree or a flat listing. The directory tree would show cats as a subfolder under animals and would show the .png files in the subfolder. The flat listing would list the blobs with the original names, cats/tuxedo.png and cats/marmalade.png.

You also can assign a custom domain to the storage account, which changes the root of the URL, so you could have something like this:

http://[storage.companyname.com]/[container]/[blobname]

This eliminates cross-domain issues when accessing files in blob storage from a website because you could use the company domain for both. Blob storage also supports Cross-Origin Resource Sharing (CORS) to help with this type of cross-source usage.

> **Note** At this time, Microsoft does not support using a custom domain name with HTTPS.

File storage

The Azure Files service enables you to set up highly available network file shares that can be accessed by using the standard Server Message Block (SMB) protocol. This means that multiple VMs can share the same files with both read and write access. The files can also be accessed using the REST interface or the storage client libraries. The Files service removes the need for you to host your own file shares in an Azure VM and go through the tricky configuration required to make it highly available.

One thing that's really special about Azure file shares versus file shares on-premises is that you can access the file from anywhere by using a URL that points to the file (similar to the blob storage URL displayed above). To do this, you have to append a shared access signature (SAS). We'll talk more about shared access signatures in the section on Security.

File shares can be used for many common scenarios:

- Many on-premises applications use file shares; this makes it easier to migrate those applications that share data to Azure. If you mount the file share to the same drive letter that the on-premises application uses, the part of your application that accesses the file share should work without any changes.
- Configuration files can be stored on a file share and accessed by multiple VMs.
- Diagnostic logs, metrics, crash dumps, etc. can be saved to a file share to be processed and analyzed later.
- Tools and utilities used by multiple developers in a group can be stored on a file share to ensure that everyone uses the same version and that they are available to everyone in the group.

To make the share visible to a VM, you just mount it as you would any other file share, and then you can access it through the network URL or the drive letter to which it was assigned. The network URL has the format \\*[storage account name].file.core.windows.net**[share name]*. After the share is mounted, you can access it using the standard file system APIs to add, change, delete, and read the directories and files.

To create or view a file share or upload or download files to it from outside Azure, you can use the Azure portal, PowerShell, the Azure Command-Line Interface (CLI), the REST APIs, one of the storage client libraries, or AzCopy, a command-line tool provided by Microsoft. (For more information on AzCopy, check out this link: http://azure.microsoft.com/documentation/articles/storage-use-azcopy/.) There are also several storage explorers you can use, as noted at the beginning of this article.

Here are some of the points about Azure Files that you need to know:

- When using SMB 2.1, the share is available only to VMs within the same region as the storage account. This is because SMB 2.1 does not support encryption.
- When using SMB 3.0, the share can be mounted on VMs in different regions, or even the desktop. Note that to mount an Azure file share on the desktop, port 445 (SMB) must be open, so you may

need to negotiate that with your company. Many ISPs and corporate IT departments block this port. This TechNet wiki shows a list of ISPs reported by Microsoft customers as allowing or disallowing port 445 traffic:
http://social.technet.microsoft.com/wiki/contents/articles/32346.azure-summary-of-isps-that-allow-disallow-access-from-port-445.aspx

- If using a Linux VM, you can only mount shares available within the same region as the storage account. This is because while the Linux SMB client supports SMB 3.0, it does not currently support encryption. The Linux developers responsible for SMB functionality have agreed to implement this, but there is no known time frame.
- If using a Mac, you can't mount Azure File shares because Apple's Mac OS doesn't support encryption on SMB 3.0. Apple has agreed to implement this, but there is no known time frame.
- You can access the data from anywhere by using the REST APIs (rather than SMB).
- The storage emulator does not support Azure Files.
- The file shares can be up to 5 TB.
- Throughput is up to 60 MB/s per share.
- The size limit of the files placed on the share is 1 TB.
- There are up to 1,000 IOPS (of size 8 KB) per share.
- Active Directory–based authentication and access control lists (ACLs) are not currently supported, but it is expected that they will be supported at some time in the future. For now, the Azure Storage account credentials are used to provide authentication for access to the file share. This means anybody with the share mounted will have full read/write access to the share.
- For files that are accessed repeatedly, you can maximize performance by splitting a set of files among multiple shares.

Table storage

Azure Table storage is a scalable NoSQL data store that enables you to store large volumes of semi-structured, nonrelational data. It does not allow you to do complex joins, use foreign keys, or execute stored procedures. Each table has a single clustered index that can be used to query the data quickly. You also can access the data by using LINQ queries and Odata with the WCF Data Service .NET libraries. A common use of table storage is for diagnostics logging.

To use table storage, you have to create a storage account. Once you have a storage account, you can create tables and fill them with data.

A table stores entities (rows), each of which contains a set of key/value pairs. Each entity has three system properties: a partition key, a row key, and a timestamp. The partition key and row key combination must be unique; together they make up the primary key for the table. The PartitionKey property is used to shard (partition) the entities across different storage nodes, allowing for load balancing across storage nodes. All entities with the same PartitionKey are stored on the same storage node. The RowKey is used to provide uniqueness within a given partition.

To get the best performance, you should give a lot of thought to the PrimaryKey and RowKey and how you need to retrieve the data. You don't want all of your data to be in the same partition; nor do you want each entity to be in its own partition.

The Azure Table service provides scalability targets for both storage account and partitions. The Timestamp property is maintained by Azure, and it represents the date and time the entity was last modified. Azure Table service uses this to support optimistic concurrency with Etags.

In addition to the system properties, each entity has a collection of key/value pairs called properties. There is no schema, so the key/value pairs of each entity can contain values of different properties. For example, you could be doing logging, and one entity could contain a payload of {customer id, customer name, request date/time, request} and the next could have {customer id, order id, item count, date-time order filled}. You can store up to 252 key/value pairs in each table entity.

The number of tables is unlimited, up to the size limit of a storage account.

Tables can be managed by using the storage client library. The Table service also supports a REST API that implements the Odata protocol; tables are addressable with the Odata protocol using a URL in the following format:

http://[storage account name]/table.core.windows.net/[table name]

Queue storage

The Azure Queue service is used to store and retrieve messages. Queue messages can be up to 64 KB in size, and a queue can contain millions of messages—up to the maximum size of a storage account. Queues generally are used to create a list of messages to be processed asynchronously. The Queue service supports best-effort first in, first out (FIFO) queues.

For example, you might have a background process (such as a worker role or Azure WebJob) that continuously checks for messages on a queue. When it finds a message, it processes the message and then removes it from the queue. One of the most common examples is image or video processing.

Let’s say you have a web application that allows a customer to upload images into a container in blob storage. Your application needs to create thumbnails for each image. Rather than making the customer wait while this processing is done, you put a message on a queue with the customer ID and container name. Then, you have a background process that retrieves the message and parses it to get the customer ID and the container name. The background process then retrieves each image, creates a thumbnail, and writes the thumbnail back to the same blob storage container as the original image. After all images are processed, the background process removes the message from the queue.

What if you need the message to exceed 64 KB in size? In that case, you could write a file with the information to a blob in blob storage and put the URL to the file in the queue message. The background process could retrieve the message from the queue and then take the URL and read the file from blob storage to do the required processing.

Azure Queues provide at-least-once semantics in which each message may be read one or more times. This makes it important that all processing of the message be idempotent, which means the outcome of the processing must be the same regardless of how many times the message is processed.

When you retrieve a message from a queue, it is not deleted from the queue—you have to delete it when you’re done with it. When the message is read from the queue, it becomes invisible. The Invisibility Timeout is the amount of time to allow for processing the message—if the message is not deleted from the queue within this amount of time, it becomes visible again for processing. In general, you want to set this property to the largest amount of time that would be needed to process a message so that while one instance of a worker role is processing it, another instance doesn’t find it (visible) on the queue and try to process it at the same time.

You don’t want to read the message from the queue, delete it from the queue, and then start processing it. If the receiver fails, that queue entry will never be processed. Leaving the message on the queue (but invisible) until the processing has completed handles the case of the receiving process failing—eventually, the message will become visible again and will be processed by another instance of the receiver.

You can simulate a workflow by using a different queue for each step. A message can be processed from one queue from which it is deleted on completion, and then that processing can place a new message on a different queue to initiate processing for the next step in the workflow. You can also prioritize messages by using queues and processing the messages in them with different priorities.

The Queue service provides poison message support through the dequeue count. The concern is that an invalid message could cause an application handling it to crash, causing the message to become visible on the queue again only to crash the application again the next time the message is processed. Such a message is referred to as a *poison message*. You can prevent this by checking the dequeue count for the message. If this exceeds some level, the processing of the message should be stopped, the message deleted from the queue, and a copy inserted in a separate poison message queue for offline review. You could process those entries periodically and send an email when an entry is placed on the queue, or you could just let them accumulate and check them manually.

If you want to process the queue messages in batches, you can retrieve up to 32 messages in one call and then process them individually. Note, however, that when you retrieve a batch of messages, it sets the Invisibility Timeout for all of the messages to the same time. This means you must be able to process all of them within the time allotted.

Redundancy

What happens if the storage node on which your blobs are stored fails? What happens if the rack holding the storage node fails? Fortunately, Azure supports something called *redundancy*. There are four choices for redundancy; you specify which one to use when you create the storage account. You can change the redundancy settings after they are set up, except in the case of zone redundant storage.

- **Locally Redundant Storage (LRS)** Azure Storage provides high availability by ensuring that three copies of all data are made synchronously before a write is deemed successful. These copies are stored in a single facility in a single region. The replicas reside in separate fault domains and upgrade domains. This means the data is available even if a storage node holding your data fails or is taken offline to be updated.

  When you make a request to update storage, Azure sends the request to all three replicas and waits for successful responses for all of them before responding to you. This means that the copies in the primary region are always in sync.

  LRS is less expensive than GRS, and it also offers higher throughput. If your application stores data that can be easily reconstructed, you may opt for LRS.

- **Geo-Redundant Storage (GRS)** GRS makes three synchronous copies of the data in the primary region for high availability, and then it asynchronously makes three replicas in a paired region for disaster recovery. Each Azure region has a defined paired region within the same geopolitical boundary for GRS. For example, West US is paired with East US. This has a small impact on scalability targets for the storage account. The GRS copies in the paired region are not accessible to you, and GRS is best viewed as disaster recovery for Microsoft rather than for you. In the event of a major failure in the primary region, Microsoft would make the GRS replicas available, but this has never happened to date.

- **Read-Access Geo-Redundant Storage (RA-GRS)** This is GRS plus the ability to read the data in the secondary region, which makes it suitable for partial customer disaster recovery. If there is a problem with the primary region, you can change your application to have read-only access to the paired region. The storage client library supports a fallback mechanism via Microsoft.WindowsAzure.Storage.RetryPolicies.LocationMode to try to read from the secondary copy if the primary copy can't be reached. This feature is built in for you. Your customers might not be able to perform updates, but at least the data is still available for viewing, reporting, etc.

You also can use this if you have an application in which only a few users can write to the data but many people read the data. You can point your application that writes the data to the primary region but have the people only reading the data access the paired region. This is a good way to spread out the performance when accessing a storage account.

- **Zone-Redundant Storage (ZRS)** This option can only be used for block blobs in a standard storage account. It replicates your data across two to three facilities, either within a single region or across two regions. This provides higher durability than LRS, but ZRS accounts do not have metrics or logging capability.

Security and Azure Storage

Azure Storage provides a set of security features that help developers build secure applications. You can secure your storage account by using Role-Based Access Control (RBAC) and Microsoft Azure Active Directory (Azure AD). You can use client-side encryption, HTTPS, or SMB 3.0 to secure your data in transit. You can enable Storage Service Encryption, and the Azure Storage service will encrypt data written to the storage account. OS and Data disks for VMs now have Azure Disk Encryption that can be enabled. And secure access to the data plane objects (such as blobs) can be granted using a shared access signature (SAS). Let's talk a little more about each of these.

For more detail and guidance about any of these security features, please check out the Azure Storage Security Guide at https://azure.microsoft.com/documentation/articles/storage-security-guide/.

Securing your storage account

The first thing to think about is securing your storage account.

Storage account keys

Each storage account has two authentication keys—a primary and a secondary—either of which can be used for any operation. There are two keys to allow occasional rollover of the keys to enhance security. It is critical that these keys be kept secure because their possession, along with the account name, allows unlimited access to any data in the storage account.

Say you're using key 1 for your storage account in multiple applications. You can regenerate key 2 and then change all the applications to use key 2, test them, and deploy them to production. Then, you can regenerate key 1, which removes access from anybody who is still using it. A good example of when you might want to do this is if your team uses a storage explorer that retains the storage account keys, and someone leaves the team or the company—you don't want them to have access to your data after they leaves. This can happen without a lot of notice, so you should have a procedure in place to know all the apps that need to change, and then practice rotating keys on a regular basis so that it's simple and not a big problem when it is necessary to rotate the keys in a hurry.

Using RBAC, Azure AD, and Azure Key Vault to control access to Resource Manager storage accounts

RBAC and Azure AD With Resource Manager RBAC, you can assign roles to users, groups, or applications. The roles are tied to a specific set of actions that are allowed or disallowed. Using RBAC to grant access to a storage account only handles the management operations for that storage account. You can't use RBAC to grant access to objects in the data plane like a specific container or file share. You can, however, use RBAC to grant access to the storage account keys, which can then be used to read the data objects.

For example, you might grant someone the Owner role to the storage account. This means they can access the keys and thus the data objects, and they can create storage accounts and do pretty much anything.

You might grant someone else the Reader role. This allows them to read information about the storage account. They can read resource groups and resources, but they can't access the storage account keys and therefore can't access the data objects.

If someone is going to create VMs, you must grant them the Virtual Machine Contributor role, which grants them access to retrieve the storage account keys but not to create storage accounts. They need the keys to create the VHD files that are used for the VM disks.

Azure Key Vault Azure Key Vault helps safeguard cryptographic keys and secrets used by Azure applications and services. You could store your storage account keys in an Azure Key Vault. What does this do for you? While you can't control access to the data objects directly using Active Directory, you can control access to an Azure Key Vault using Active Directory. This means you can put your storage account keys in Azure Key Vault and then grant access to them for a specific user, group, or application.

Let's say you have an application running as a Web App that uploads files to a storage account. You want to be really sure nobody else can access those files. You add the application to Azure Active Directory and grant it access to the Azure Key Vault with that storage account's keys in it. After that, only that application can access those keys. This is much more secure than putting the keys in the *web.config* file where a hacker could get to them.

Securing access to your data

There are two ways to secure access to your data objects. We just talked about the first one—by controlling access to the storage account keys.

The second way to secure access is by using shared access signatures and stored access policies. A shared access signature (SAS) is a string containing a security token that can be attached to the URI for an asset that allows you to delegate access to specific storage objects and to specify constraints such as permissions and the date/time range of access.

You can grant access to blobs, containers, queue messages, files, and tables. With tables, you can grant access to specific partition keys. For example, if you were using geographical state for your partition key, you could give someone access to just the data for California.

You can fine-tune this by using a separation of concerns. You can give a web application permission to write messages to a queue, but not to read them or delete them. Then, you can give the worker role or Azure WebJob the permission to read the messages, process the messages, and delete the messages. Each component has the least amount of security required to do its job.

Here's an example of an SAS, with each parameter explained:

```
http://mystorage.blob.core.windows.net/mycontainer/myblob.txt (URL to the blob)

?sv=2015-04-05 (storage service version)

&st=2015-12-10T22%3A18%3A26Z (start time, in UTC time and URL encoded)

&se=2015-12-10T22%3A23%3A26Z (end time, in UTC time and URL encoded)

&sr=b (resource is a blob)

&sp=r (read access)

&sip=168.1.5.60-168.1.5.70 (requests can only come from this range of IP addresses)

&spr=https (only allow HTTPS requests)
```

```
&sig=Z%2FRHIX5XcgOMq2rqI3OlWTjEg2tYkboXr1P9ZUXDtkk%3D (signature used for the authentication of the
SAS)
```

Note that the SAS query parameters must be URL encoded, such as %3A for colon (:) and %20 for a space. This SAS gives read access to a blob from 12/10/2015 10:18 PM to 12/10/2015 10:23 PM.

When the storage service receives this request, it will take the query parameters and create the *&sig* value on its own and compare it to the one provided here. If they agree, it will verify the rest of the request. If our URL pointed to a file on a file share instead of a blob, the request would fail because blob is specified. If the request were to update the blob, it would fail because only read access has been granted.

There are both account-level SAS and service-level SAS. With account-level SAS, you can do things like list containers, create containers, delete file shares, and so on. With service-level SAS, you can only access the data objects. For example, you can upload a blob into a container.

You can also create stored access policies on container-like objects such as blob containers and file shares. This will let you set the default values for the query parameters, and then you can create the SAS by specifying the policy and the query parameter that is different for each request. For example, you might set up a policy that gives read access to a specific container. Then, when someone requests access to that container, you create an SAS from the policy and use it.

There are two advantages to using stored access policies. First, this hides the parameters that are defined in the policy. So if you set your policy to give access to 30 minutes, it won't show that in the URL—it just shows the policy name. This is more secure than letting all of your parameters be seen.

The second reason to use stored access policies is that they can be revoked. You can either change the expiration date to be prior to the current date/time or remove the policy altogether. You might do this if you accidentally provided access to an object you didn't mean to. With an ad hoc SAS URL, you have to remove the asset or change the storage account keys to revoke access.

Shared access signatures and stored access policies are the two most secure ways to provide access to your data objects.

Securing your data in transit

Another consideration when storing your data in Azure Storage is securing the data when it is being transferred between the storage service and your applications.

First, you should always use the HTTPS protocol, which ensures secure communication over the public Internet. Note that if you are using SAS, there is a query parameter that can be used that specifies that only the HTTPS protocol can be used with that URL.

For Azure File shares, SMB 3.0 running on Windows encrypts the data going across the public Internet. When Apple and Linux add security support to SMB 3.0, you will be able to mount file shares on those machines and have encrypted data in transit.

Last, you can use the client-side encryption feature of the .NET and Java storage client libraries to encrypt your data before sending it across the wire. When you retrieve the data, you can then unencrypt it. This is built in to the storage client libraries for .NET and Java. This also counts as encryption at rest because the data is encrypted when stored.

Encryption at rest

Let's look at the various options available to encrypt the stored data.

Storage Service Encryption (SSE)

This is a new feature currently in preview. This lets you ask the storage service to encrypt blob data when writing it to Azure Storage. This feature has been requested by many companies to fulfill security and compliance requirements. It enables you to secure your data without having to add any code to any of your applications. Note that it only works for blob storage; tables, queues, and files will be unaffected.

This feature is per-storage account, and it can be enabled and disabled using the Azure portal, PowerShell, the CLI, the Azure Storage Resource Provider REST API, or the .NET storage client library. The keys are generated and managed by Microsoft at this time, but in the future you will get the ability to manage your own encryption keys.

This can be used with both Standard and Premium storage, but only with the new Resource Manager accounts. During the preview, you have to create a new storage account to try out this feature.

One thing to note: after being enabled, the service encrypts data written to the storage account. Any data already written to the account is not encrypted. If you later disable the encryption, any future data will not be encrypted, but it does retain encryption on the data written while encryption was enabled.

If you create a VM using an image from the Azure Marketplace, Azure performs a shallow copy of the image to your storage account in Azure Storage, and it is not encrypted even if you have SSE enabled. After it creates the VM and starts updating the image, SSE will start encrypting the data. For this reason, Microsoft recommends that you use Azure Disk Encryption on VMs created from images in the Azure Marketplace if you want them fully encrypted.

Azure Disk Encryption

This is another new feature that is currently in preview. This feature allows you to specify that the OS and data disks used by an IaaS VM should be encrypted. For Windows, the drives are encrypted with industry-standard BitLocker encryption technology. For Linux, encryption is performed using DM-Crypt.

> **Note** For Linux VMs already running in Azure or new Linux VMs created from images in the Azure Marketplace, encryption of the OS disk is not currently supported. Encryption of the OS volume for Linux VMs is supported only for VMs that were encrypted on-premises and uploaded to Azure. This restriction only applies to the OS disk; encryption of data volumes for a Linux VM is supported.

Azure Disk Encryption is integrated with Azure Key Vault to allow you to control and manage the disk encryption keys.

Unlike SSE, when you enable this, it encrypts the whole disk, including data that was previously written. You can bring your own encrypted images into Azure and upload them and store the keys in Azure Key Vault, and the image will continue to be encrypted. You can also upload an image that is not encrypted or create a VM from the Azure Gallery and ask that its disks be encrypted.

This is the method recommended by Microsoft to encrypt your IaaS VMs at rest. Note that if you turn on both SSE and Azure Disk Encryption, it will work fine. Your data will simply be double-encrypted.

Client-side encryption

We looked at client-side encryption when discussing encryption in transit. The data is encrypted by the application and sent across the wire to be stored in the storage account. When retrieved, the data is decrypted by the application. Because the data is stored encrypted, this is encryption at rest.

For this encryption, you can encrypt the data in blobs, tables, and queues, rather than just blobs like SSE. Also, you can bring your own keys or use keys generated by Microsoft. If you store your encryption keys in Azure Key Vault, you can use Azure Active Directory to specifically grant access to the keys. This allows you to control who can read the vault and retrieve the keys being used for client-side encryption.

This is the most secure method of encrypting your data, but it does require that you add code to perform the encryption and decryption. If you only have blobs that need to be encrypted, you may choose to use a combination of HTTPS and SSE to meet the requirement that your data be encrypted at rest.

Using Storage Analytics to audit access

You may want to see how people are accessing your storage account. Do all the requests use an SAS? How many people are accessing the storage account using the actual storage account keys?

To check this, you can enable the logging in the Azure Storage Analytics and check the results after a while. Enabling the logging tells the Azure Storage service to log all requests to the storage account. (Note that at this time, only blobs, tables, and queues are supported.)

The logs are stored in a container called $logs in blob storage. They are stored by date and time, collected by hour. If there is no activity, no logs are generated.

Here are the fields that are stored in the logs.

<version-number>;<request-start-time>;**<operation-type>;<request-status>**;<http-status-code>;<end-to-end-latency-in-ms>;<server-latency-in-ms>;**<authentication-type>**;<requester-account-name>;<owner-account-name>;<service-type>;<request-url>;<requested-object-key>;<request-id-header>;<operation-count>;<requester-ip-address>;<request-version-header>;<request-header-size>;<request-packet-size>;<response-header-size>;<response-packet-size>;<request-content-length>;<request-md5>;<server-md5>;<etag-identifier>;<last-modified-time>;<conditions-used>;<user-agent-header>;<referrer-header>;<client-request-id>

The fields in bold are the ones in which we are interested. So if you look at a log file, these are the three cases we can look for:

1. The blob is public, and it is accessed using a URL without an SAS. In this case, the request-status will be *AnonymousSuccess* and the authentication type will be *anonymous*.

   1.0;2015-11-17T02:01:29.0488963Z;GetBlob;**AnonymousSuccess**;200;124;37;**anonymous**;;mystorage…

2. The blob is private and was used with an SAS. In this case, the request-status is *SASSuccess* and the authentication type is *sas*.

   1.0;2015-11-16T18:30:05.6556115Z;GetBlob;**SASSuccess**;200;416;64;**sas**;;mystorage…

3. The blob is private, and the storage key was used to access it. In this case, the request-status is *Success* and the authentication type is *authenticated*.

   1.0;2015-11-16T18:32:24.3174537Z;GetBlob;**Success**;206;59;22;**authenticated**;mystorage…

To view and analyze these log files, you can use the Microsoft Message Analyzer (free from Microsoft). You can download the Message Analyzer here: https://www.microsoft.com/download/details.aspx?id=44226. The operating guide is here: https://technet.microsoft.com/library/jj649776.aspx.

The Message Analyzer lets you search and filter the data. An example of when you might want to do this is if you have your keys stored in Azure Key Vault and only one application has access to the Azure Key Vault. In that case, you might search for instances where *GetBlob* was called and make sure there aren't any calls that were authenticated in any other way.

Important For Azure Analytics, the metrics tables start with $metrics, and the logs container in blob storage is called $logs. You cannot even see the tables and container using PowerShell, the Visual Studio Cloud Explorer, or the Azure portal.

You can see the tables and container and even open and view the entities and blobs using the Microsoft Azure Storage Explorer (http://storageexplorer.com). The Cerebrata Azure Management Studio and Cloud Portam allow you to access and view these objects (http://www.cerebrata.com) as well.

You can also write your own code using one of the storage client libraries to retrieve the data from table storage and blob storage. Other storage explorers listed in the article at the beginning of this chapter may also enable you to view this data.

Using Cross-Origin Resource Sharing (CORS)

When a web browser running in one domain makes an HTTP request for a resource in another domain, it's called a cross-origin HTTP request. If the request is made in a script language such as JavaScript, the browser will not allow the request.

For example, if a web application running on contoso.com makes a request for a jpeg on fabrikam.blob.core.windows.net, it will be blocked.

What if you actually *want* to share the images in your storage account with Contoso? Azure Storage allows you to enable CORS—Cross-Origin Resource Sharing. For this example, you would enable CORS on the fabrikam storage account and allow access from contoso.com. You can do this by using the Rest API or the storage client library.

Creating and managing storage

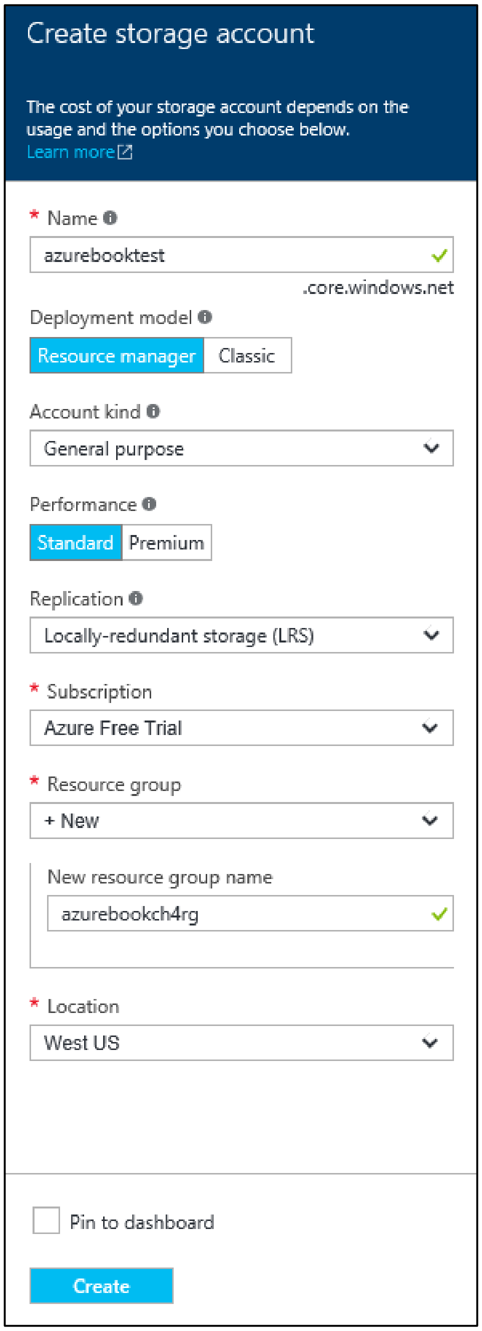
In this section, we are going to go through several exercises to show the different ways you can access your data objects. First, we'll use the Azure portal and the Visual Studio Cloud Explorer, then we'll do some of the same operations using PowerShell. Here's what we'll do:

- Create a storage account using the Azure portal.
- Create a blob container and upload blobs using the Visual Studio Cloud Explorer.
- Create a file share and upload files using the Azure portal.
- Create a table and add records to it using Visual Studio Cloud Explorer.
- Create a storage account using Azure PowerShell.
- Create a blob container and upload blobs using PowerShell.
- Create a file share and upload files using PowerShell.

To do the Azure PowerShell demos, you need to install Azure PowerShell. If you haven't used Azure PowerShell before, please check out Chapter 8, "Management tools," or this article that shows how to install and configure Azure PowerShell: https://azure.microsoft.com/documentation/articles/powershell-install-configure/.

Create a storage account using the Azure portal

To create a storage account, log into the Azure portal. Click New > Data + Storage > Storage Account. You see a screen similar to Figure 4-1.



Create storage account

The cost of your storage account depends on the usage and the options you choose below.
Learn more

* Name
azurebooktest ✓
.core.windows.net

Deployment model
[✓ Resource manager] Classic

Account kind
General purpose

Performance
[✓ Standard] Premium

Replication
Locally-redundant storage (LRS)

* Subscription
Azure Free Trial

* Resource group
+ New

New resource group name
azurebookch4rg ✓

* Location
West US

□ Pin to dashboard

Figure 4-1 Add a new storage account.

1. First, fill in a name for the storage account. The name must be globally unique because it is used as part of the URL. This will be used in the endpoints for blobs, files, tables, and queues. In Figure 4-1, the storage account name is azurebooktest. This means the blobs (for example) will be addressable as http://azurebooktest.blob.core.windows.net.

2. The next field displayed is the Deployment Model. You want to create a Resource Manager storage account, so select Resource Manager.

3. Account Kind can be General Purpose or Blob Storage. Select General Purpose so you can use the same account for blobs, files, and tables.

4. For Replication, the default is GRS—Globally Redundant Storage. Change this to LRS (Locally Redundant Storage), which has the lowest cost. For test data, you don't need it to be replicated in a completely different region.

5. If you manage multiple subscriptions, select the one you want to be used for this storage account.

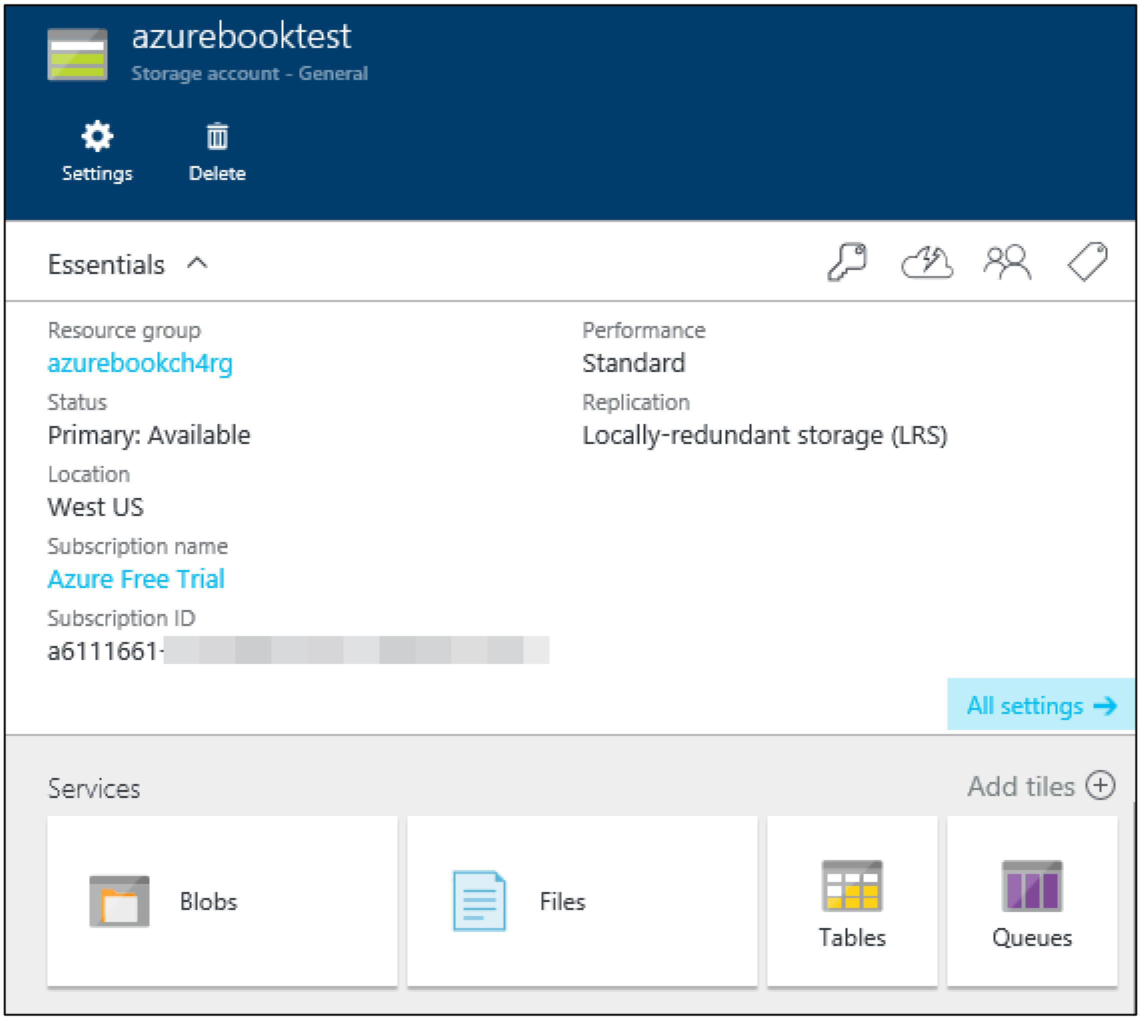
6. For Resource Group, let's create a new one just for this chapter. Specify the name of the resource group. In Figure 4-1, the resource group is called azurebookch4rg.

7. For Location, select the Azure region closest to you for the best performance.

8. Select the Pin To Dashboard check box and click Create. Azure will provision the storage account and add it to the Dashboard.

   Now that you've created a Resource Manager storage account in its own resource group, let's take a look at it.

9. If your storage account wasn't automatically displayed after being created, click your new storage account from the Dashboard. A blade will be displayed with information about your storage account (Figure 4-2).



azurebooktest

Storage account - General

Essentials

Resource group: azurebookch4rg

Status: Primary: Available

Location: West US

Subscription name: Azure Free Trial

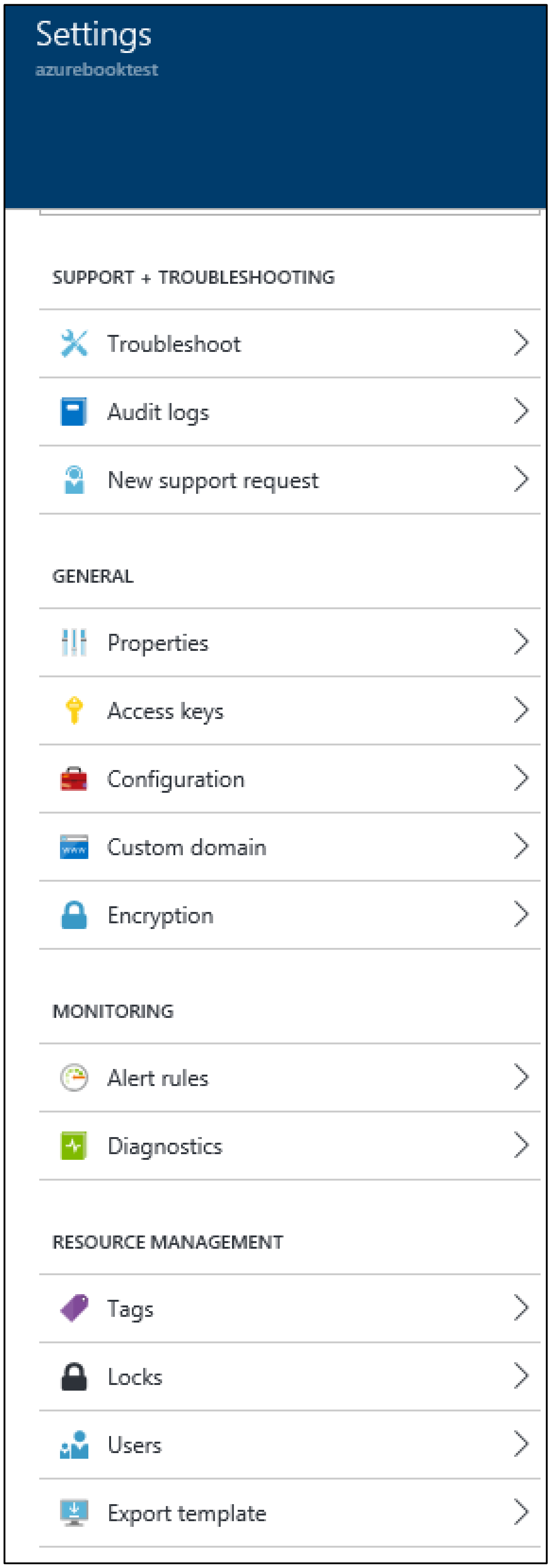
Subscription ID: a6111661-[redacted]

Performance: Standard

Replication: Locally-redundant storage (LRS)

Figure 4-2 View your new storage account.

10. Click All Settings to bring up the Settings blade (Figure 4-3).



Settings

azurebooktest

SUPPORT + TROUBLESHOOTING

- Troubleshoot
- Audit logs
- New support request

GENERAL

- Properties
- Access keys
- Configuration
- Custom domain
- Encryption

MONITORING

- Alert rules
- Diagnostics

RESOURCE MANAGEMENT

- Tags
- Locks
- Users
- Export template

Figure 4-3 Settings blade for the new storage account.

Here are some of the options in the Settings blade:

- **Access Keys** This shows you your storage account name and the two access keys. From the Access Keys blade, you can copy any of the values to the Windows clipboard. You can also regenerate the storage account access keys here.
- **Configuration** This allows you to change the replication. Yours is LRS if that's what you selected when creating the storage account. You can change it here to GRS or RA-GRS.
- **Custom Domain** This is where you can configure a custom domain for your storage account. For example, rather than calling it robinscompany.blob.core.windows.net, you can assign a domain to it and refer to it as storage.robinscompany.com.
- **Encryption** This is where you can sign up for the Storage Service Encryption preview. At some point, this will be where you enable and disable SSE for the storage account.
- **Diagnostics** This is where you can turn on the Storage Analytics and the logging.
- **Users** This is where you can grant management-plane access for this specific storage account.

Create a container and upload blobs using Visual Studio Cloud Explorer

Now you want to create a container and upload some files to it using Visual Studio Cloud Explorer.

1. Run Visual Studio. If you don't have the Azure Tools installed, you can use the Web Platform Installer to install them.
2. Click View > Cloud Explorer. You see a screen like the one in Figure 4-4.

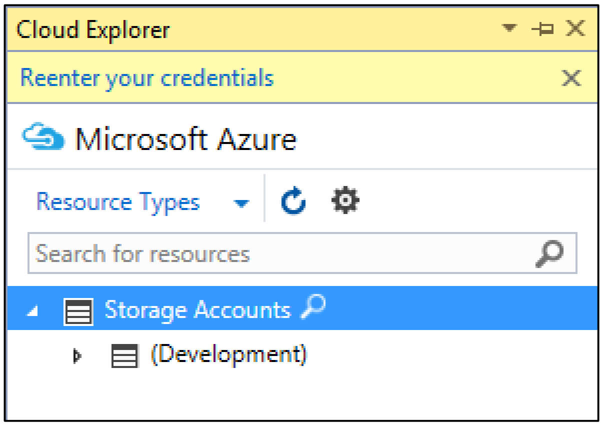
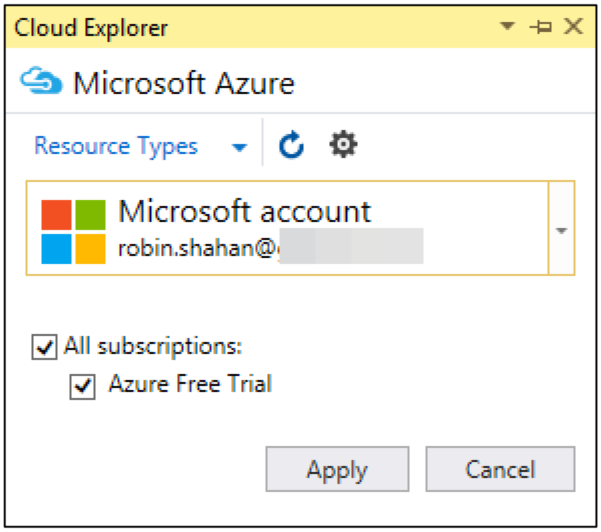


Figure 4-4 Cloud Explorer.

3. Click the Settings icon to get to the login screen (Figure 4-5).

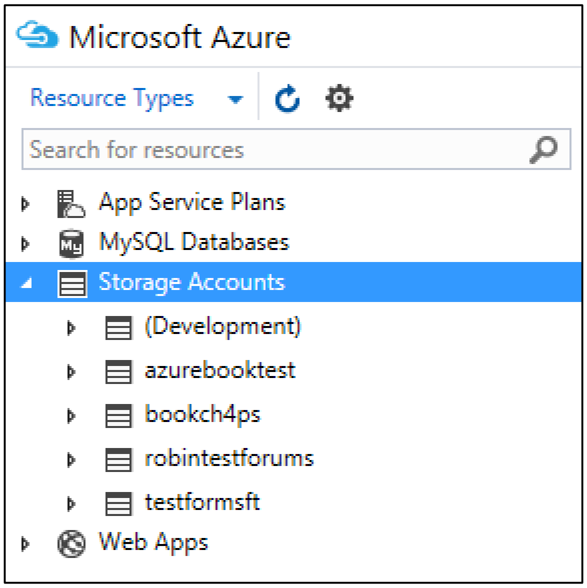


Microsoft account
robin.shahan@[redacted]

☑ All subscriptions:
☑ Azure Free Trial

Figure 4-5 Select the Azure account with which to log into the Cloud Explorer.

4. If you don't have any Azure accounts displayed in the list, click the drop-down list and select Add An Account. If you do have accounts displayed, select the one you want to use and log into it. Click Apply. After logging in, you see something like Figure 4-6.



- App Service Plans
- MySQL Databases
- Storage Accounts
  - (Development)
  - azurebooktest
  - bookch4ps
  - robintestforums
  - testformsft
- Web Apps

Figure 4-6 Visual Studio Cloud Explorer, showing resources.

5. Open the storage account you created with the portal. In the example, that's azurebooktest. The storage account has Blob Containers, Queues, and Tables. Right-click Blob Containers and select Create Blob Container, as displayed in Figure 4-7.

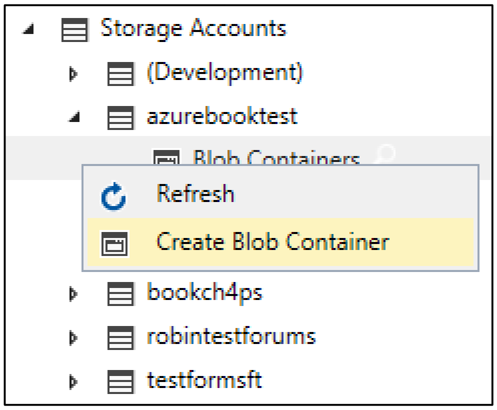
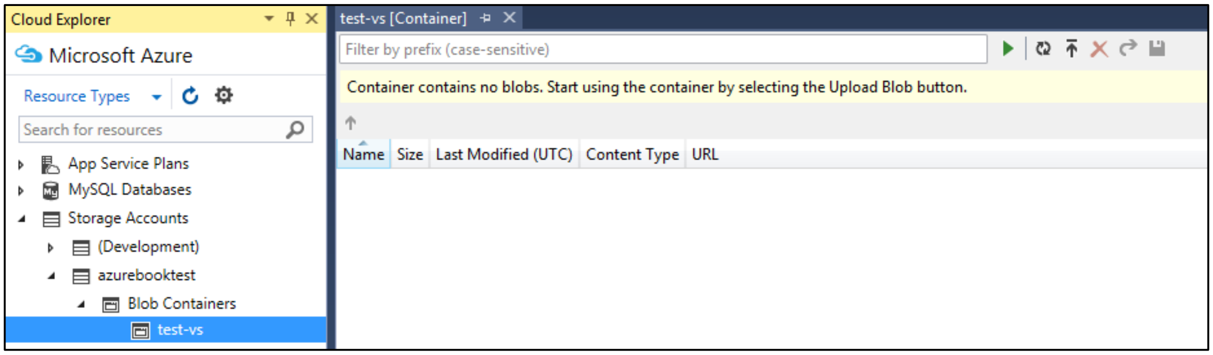


Figure 4-7 Create blob container.

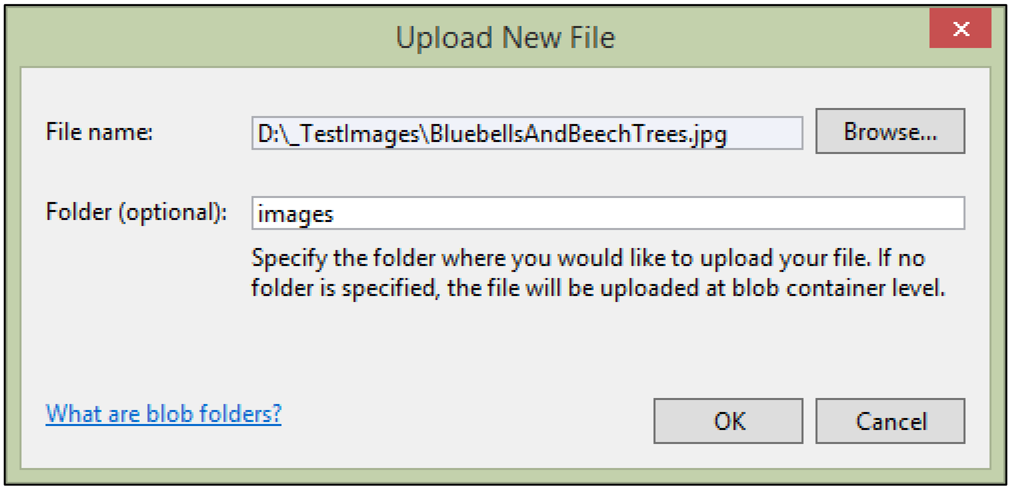
6. It shows a text box; type in the container name. The example uses test-vs. Press Enter; now it shows your new container under Blob Containers. Double-click the container name to open a screen where you can upload blobs (Figure 4-8).



Container contains no blobs. Start using the container by selecting the Upload Blob button.

Figure 4-8 Ready to upload blobs into the container.

7. To upload blobs into the container, click the icon on the top row next to the filter that shows an up arrow with a line over it (this is the same icon used in Figure 4-14). The Upload New File dialog opens (Figure 4-9). Browse to find a file. You can set a folder name here. Note that this is the pseudo-foldering discussed earlier—it includes the folder name in the blob name with a forward slash. If you leave the folder blank, it will put the file in the root of the container.



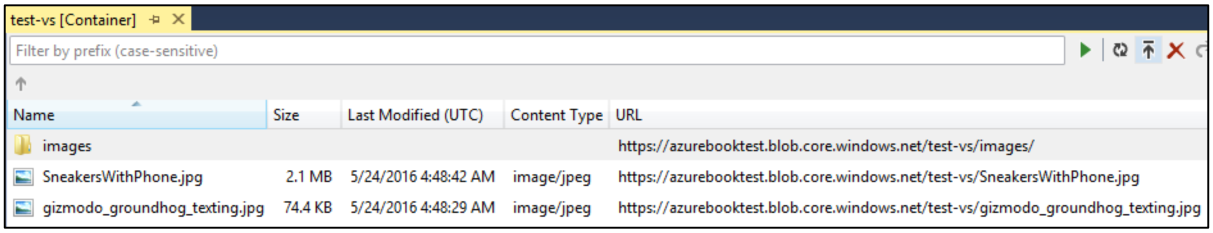
File name: D:_TestImages\BluebellsAndBeechTrees.jpg

Folder (optional): images

Specify the folder where you would like to upload your file. If no folder is specified, the file will be uploaded at blob container level.

Figure 4-9 Dialog for uploading blobs into the container.

8. Upload some files into the root and some files into a folder. You should see something similar to Figure 4-10. This figure shows a folder called images and two blobs in the root. Note that it shows the URL to the blobs. If you open the images folder, it will show the blobs there, and all of the URLs will have /images/ in them.



Name	Size	Last Modified (UTC)	Content Type	URL
images				https://azurebooktest.blob.core.windows.net/test-vs/images/
SneakersWithPhone.jpg	2.1 MB	5/24/2016 4:48:42 AM	image/jpeg	https://azurebooktest.blob.core.windows.net/test-vs/SneakersWithPhone.jpg
gizmodo_groundhog_texting.jpg	74.4 KB	5/24/2016 4:48:29 AM	image/jpeg	https://azurebooktest.blob.core.windows.net/test-vs/gizmodo_groundhog_texting.jpg

Figure 4-10 Screen showing blobs uploaded into the container.

9. You can delete blobs from the container by using the red X icon, and you can download blobs and view them in the picture viewer by double-clicking the entry in the table or by clicking the forward arrow icon.

One thing this tool does not allow you to do is set the Access Type of the container. By default, the Cloud Explorer sets it to Private. The Access Type defines who can access the blobs and the container. If this is Private, the container and the blobs in the container can only be accessed by someone who has the account credentials (account name and key) or a URL that includes an SAS. If you set this to Blob, then anyone with a URL can view the associated blob but cannot view the container properties and metadata or the list of blobs in the container. If you set this to Container, then everyone has read access to the container and the blobs therein.

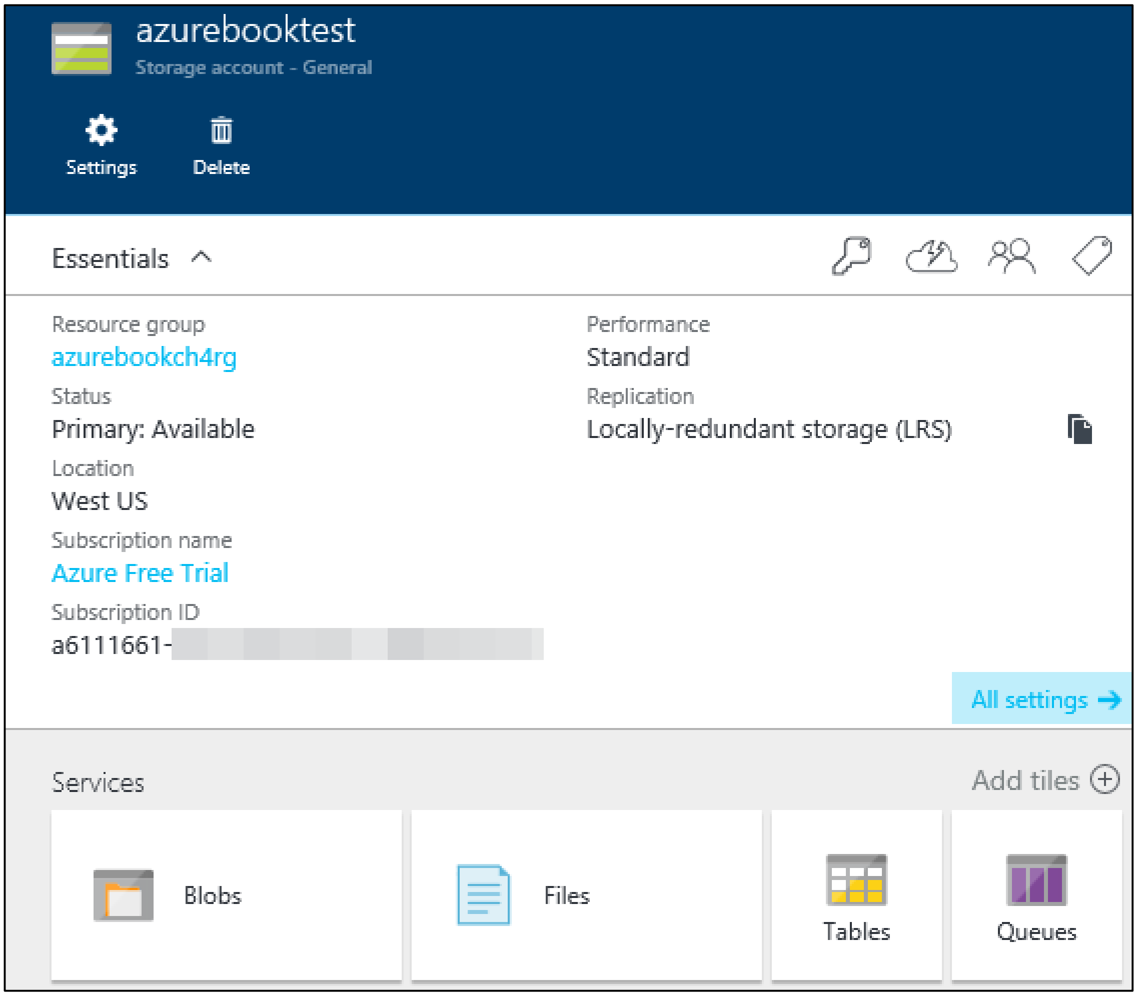
You can change this in the Azure portal and through some storage explorers. In the Azure portal, go to the storage account, click Blobs, and then select the container. A blade will open on the right showing the blobs in the container. Click Access Policy to set it to Blob or Public.

The Cloud Explorer is a pretty simple implementation of accessing blob storage. It does not allow you to upload or download folders full of images. For more sophisticated applications, check out the list of storage explorers provided earlier in this section.

Create a file share and upload files using the Azure portal

In this section, you will create an Azure File share and then upload some files to it. For this demo, you'll use the Azure portal. You can't use the Cloud Explorer in Visual Studio because it doesn't support Azure Files.

1. Log into the Azure portal. Click All Resources and then select the storage account you created using the portal. In the examples, this was azurebooktest. You should see something like Figure 4-11.


azurebooktest

Storage account - General

Resource group: azurebookch4rg

Status: Primary: Available

Location: West US

Subscription name: Azure Free Trial

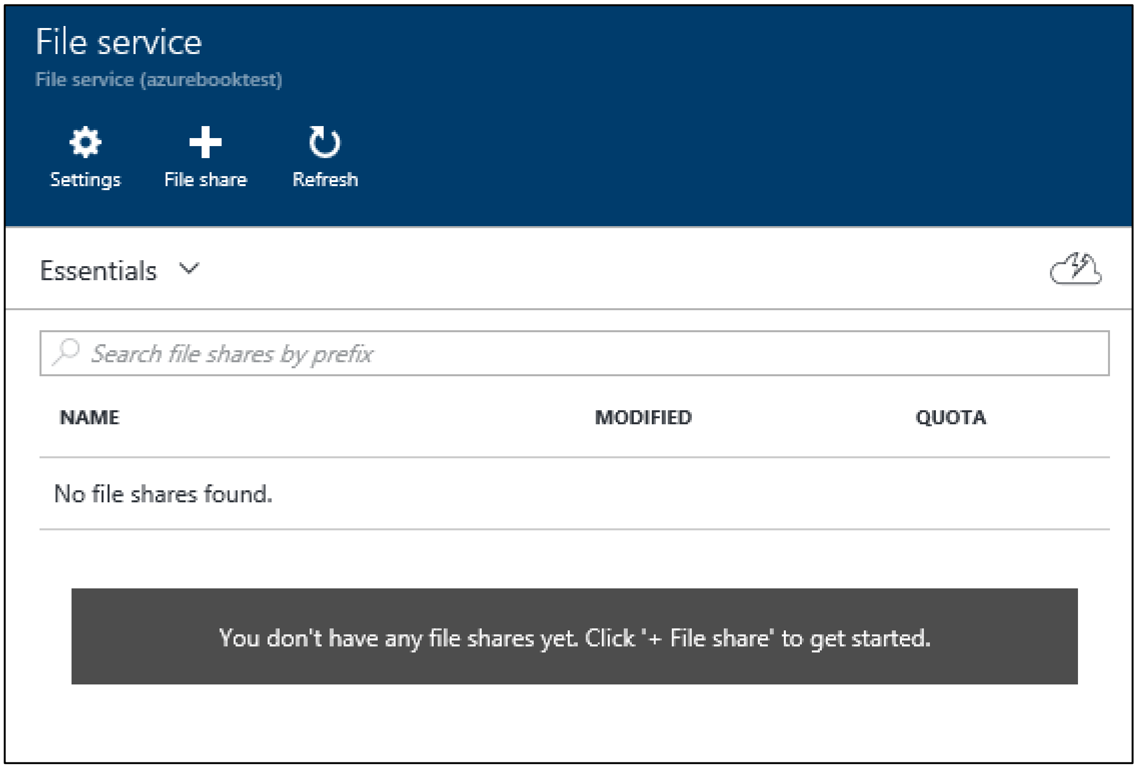
Subscription ID: a6111661-[redacted]

Performance: Standard

Replication: Locally-redundant storage (LRS)

Figure 4-11 View storage account.

2. Click Files to open the File Service blade shown in Figure 4-12.


File service

File service (azurebooktest)

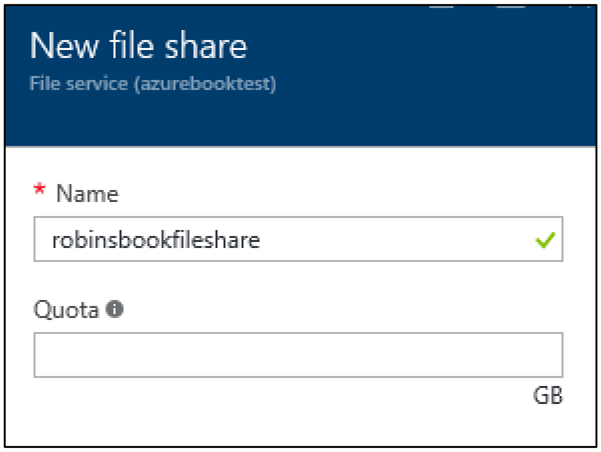
Search file shares by prefix

NAME	MODIFIED	QUOTA
No file shares found.		

You don't have any file shares yet. Click '+ File share' to get started.

Figure 4-12 File Service blade.

3. You don't have any file shares yet. Create one by clicking File Share. This will show the New File Share blade (Figure 4-13).


New file share

File service (azurebooktest)

* Name

robinsbookfileshare

Quota

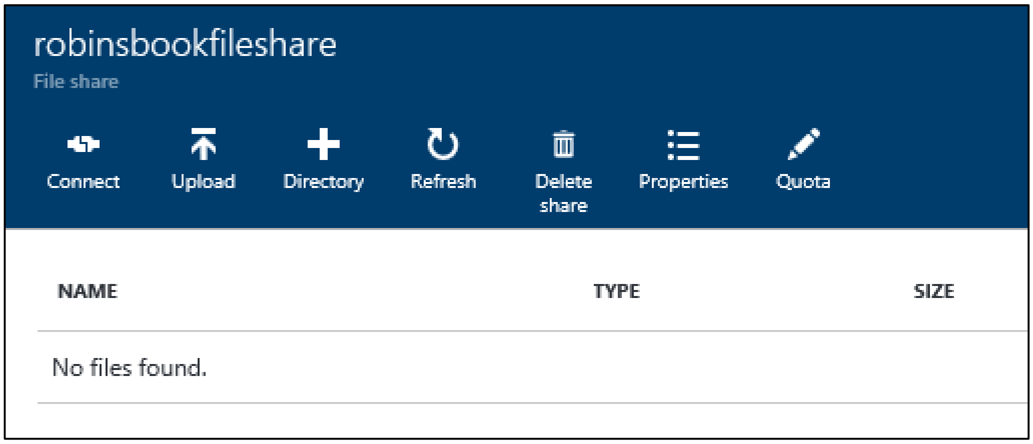
GB

Figure 4-13 Create a new file share.

4. Provide a name for the file share. If you want the maximum size of the file share to be less than the allowed 5,120 GB, specify the desired value in the Quota field. To maximize the size of the file share, leave the Quota blank.

   Click Create at the bottom of the blade, and Azure will create the file share for you and display it in the File Service blade.

5. Click the new file share to bring up the file share's blade. You see something like Figure 4-14.



NAME	TYPE	SIZE
No files found.		

Figure 4-14 Create a new file share.

Let's look at what the icons do.

- **Connect** This gives you the NET USE statement that you can use in a command window to map the network share to a local drive letter.
- **Upload** This allows you to upload files.
- **Directory** This lets you create a directory in the folder currently displayed. For you, that's the root folder.
- **Refresh** This refreshes the displayed information.
- **Delete share** This will delete the file share and all the files on it.
- **Properties** This shows the Properties blade for the file share. This shows the name, URL, quota, usage, and so on.
- **Quota** This lets you modify the quota specified.

Now upload some files. Click the Upload icon to show the Upload Files blade (Figure 4-15).

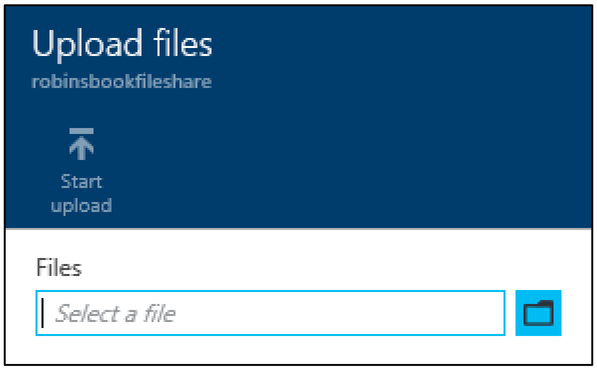
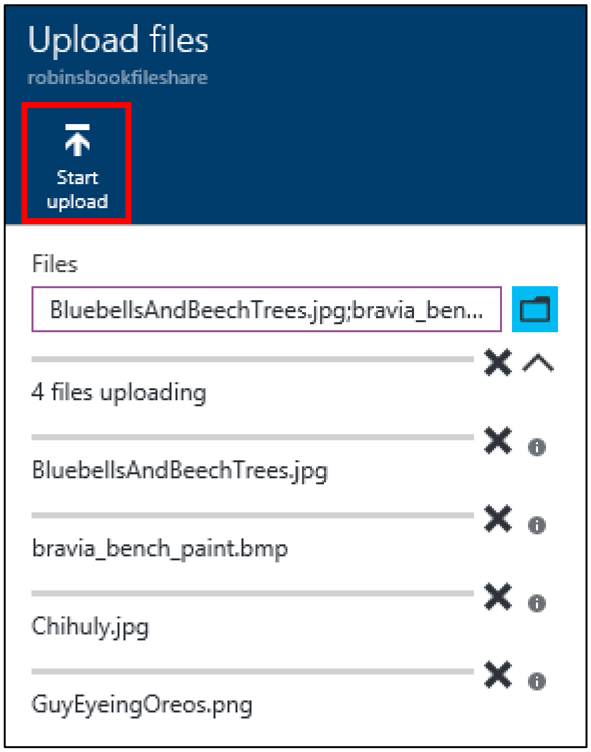


Figure 4-15 The Upload Files blade.

6. Click the file folder icon. In the Choose File To Upload dialog that displays, browse to any folder and select some files to upload. You can upload up to four files at a time. If you select more than four, it will ignore the extras. After selecting them and returning to the Upload Files blade, it shows the files in a list. Click the Start Upload button displayed in Figure 4-16 to upload the files.



Files

BluebellsAndBeechTrees.jpg;bravia_ben...

4 files uploading

BluebellsAndBeechTrees.jpg

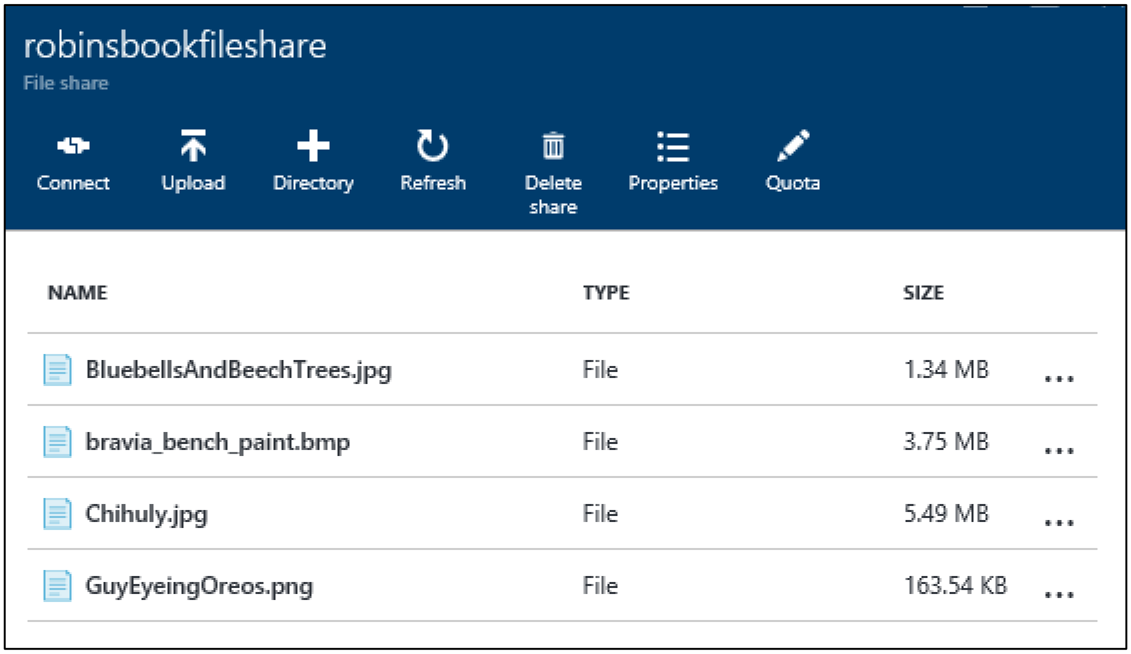
bravia_bench_paint.bmp

Chihuly.jpg

GuyEyeingOreos.png

Figure 4-16 Uploading files.

The portal will show the progress while uploading the files and then show the files in the File Share blade, as illustrated in Figure 4-17.



NAME	TYPE	SIZE
BluebellsAndBeechTrees.jpg	File	1.34 MB
bravia_bench_paint.bmp	File	3.75 MB
Chihuly.jpg	File	5.49 MB
GuyEyeingOreos.png	File	163.54 KB

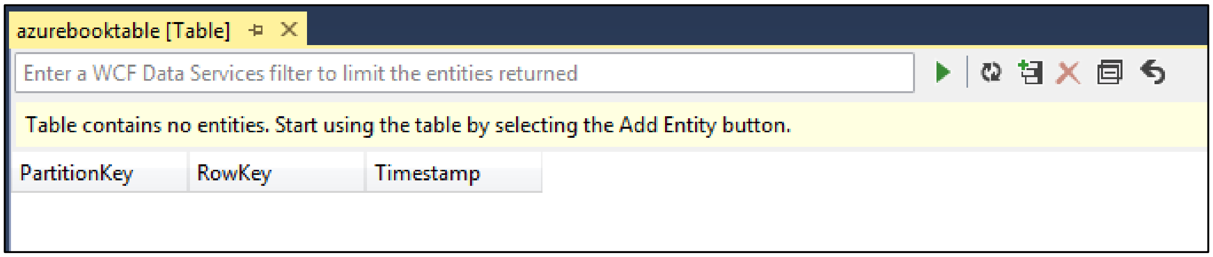
Figure 4-17 Uploaded files.

Create a table and add records using the Visual Studio Cloud Explorer

Now you can create a table in your storage account and add some entities to it. You can use one of the storage explorer tools mentioned earlier in this book, but let's see how easy it is to use the Visual Studio Cloud Explorer to do this task.

If you've done the steps in the last section that showed how to use the Cloud Explorer to add blobs to blob storage, this will be just as easy. If you don't still have the Cloud Explorer open, open it again and log in to your Azure account again.

In Cloud Explorer, right-click Tables and select Create Table. You will be prompted for the name of the table, which must be unique within your storage account. After pressing Enter to create the new table, double-click the table name to see something similar to Figure 4-18.



Enter a WCF Data Services filter to limit the entities returned

Table contains no entities. Start using the table by selecting the Add Entity button.

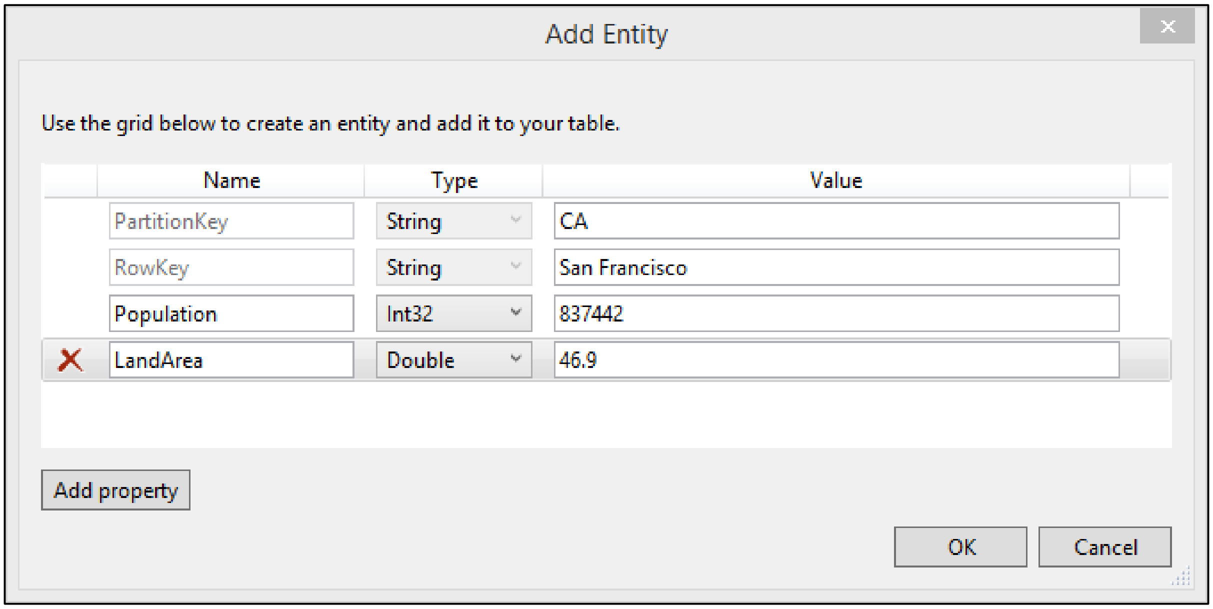
PartitionKey	RowKey	Timestamp

Figure 4-18 Editing a new table.

You don't have any entities, so add one by clicking the icon with the + in it.

As discussed in the section "Table storage" earlier in this chapter, you have to think about what you want to use for PartitionKey and RowKey to get the best performance.

For this example, use geographic state abbreviation for the PartitionKey and city name for the RowKey. For properties, add Population as Int32 and LandArea as a Double. Fill in values for each of the fields. Figure 4-19 shows what the entity looks like before adding it to the table.



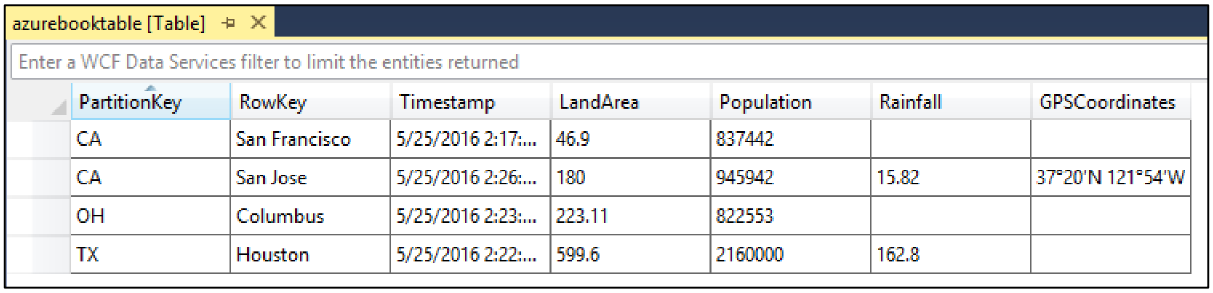
Use the grid below to create an entity and add it to your table.

Name	Type	Value
PartitionKey	String	CA
RowKey	String	San Francisco
Population	Int32	837442
LandArea	Double	46.9

Figure 4-19 Add an entity to the table.

Click OK to save the entity. Add another entity, and this time, add another property besides Population and LandArea, such as GPSCoordinates. Add a couple more entities, including whatever properties you want. If you want to edit an entity after saving it, you can right-click the entity and select Edit. You also can delete entities using this view.

After entering a few entities, you should have something similar to Figure 4-20.


azurebooktable [Table]
Enter a WCF Data Services filter to limit the entities returned

PartitionKey	RowKey	Timestamp	LandArea	Population	Rainfall	GPSCoordinates
CA	San Francisco	5/25/2016 2:17:...	46.9	837442		
CA	San Jose	5/25/2016 2:26:...	180	945942	15.82	37°20'N 121°54'W
OH	Columbus	5/25/2016 2:23:...	223.11	822553		
TX	Houston	5/25/2016 2:22:...	599.6	2160000	162.8	

Figure 4-20 View the table after adding entities.

You can see the PartitionKey and RowKey combination is unique for all of the entities. The rest of each row in the table is the list of key/value pairs. Not all entities have the same properties. The entity for San Francisco only has LandArea and Population; the entity for San Jose is the only one with GPSCoordinates. This is a strength of Azure Tables—the key/value pairs can vary for each entity.

You can create tables by using a designer such as this one in Visual Studio, but for adding, changing, and deleting entities in an application, you will probably want to write your own code using the storage client library. For examples, please check out this link: http://azure.microsoft.com/documentation/articles/storage-dotnet-how-to-use-tables/.

Create a storage account using PowerShell

Let's see how to do many of the same operations using Azure PowerShell cmdlets.

1. First, you need to run Azure PowerShell ISE.

2. Log into your Azure account using the PowerShell cmdlet *Login-AzureRmAccount*. You will be prompted for your Azure credentials; go ahead and log in.

   ```
   > Login-AzureRmAccount
   ```

   Note: There is also a cmdlet called *Add-AzureAccount*. This is for using classic resources. All of the cmdlets for Resource Manager accounts have "Rm" after the word "Azure" in the cmdlet.

   After logging into the account, it should show the subscription in the command window.

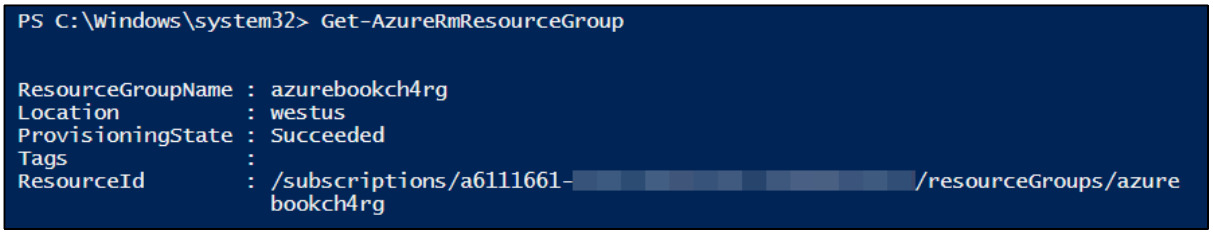
3. Now you need a resource group in which to put your storage account. Use the same one you created in the portal when you created the storage account there. If you put all of the resources created in this chapter in the same resource group, then at the end you can delete them in one fell swoop by deleting the resource group.

   If you want to create a new resource group, you can do that with the *New-AzureRmResourceGroup* cmdlet like this:

   ```
   > New-AzureRmResourceGroup "nameofgroup" -Location “location”
   ```

   An example of Location is West US.

   You can retrieve a list of resource groups by using the *Get-AzureRmResourceGroup* cmdlet. When you run this, you see the resource group you set up when creating the storage account in the portal (Figure 4-21).



```
PS C:\Windows\system32> Get-AzureRmResourceGroup


ResourceGroupName : azurebookch4rg
Location          : westus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/a6111661-[redacted]/resourceGroups/azure
                    bookch4rg
```

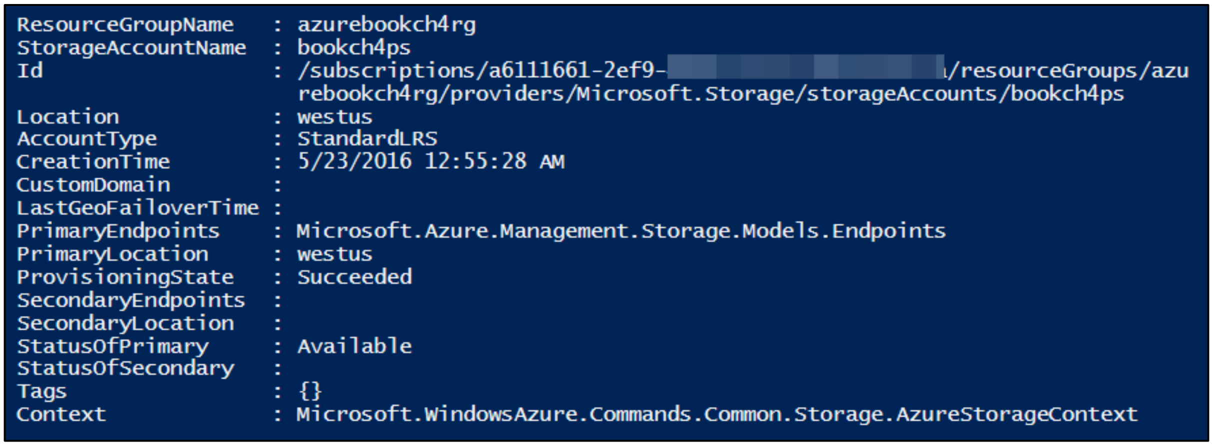
Figure 4-21 Show available resource groups.

4. Now let's create the storage account. You want to create a Resource Manager storage account and specify the resource group. You also specify the storage account name, the location, and the type, which is for the redundancy type. You want to use locally redundant storage for the same reasons mentioned when creating the storage account using the Azure portal. Select your own storage account name. Here's what the command looks like:

   ```
   > New-AzureRmStorageAccount –ResourceGroup "bookch4rg" –StorageAccountName "bookch4ps" –Location
   "West US" –Type "Standard_LRS"
   ```

   For a full list of locations, you can run the PowerShell cmdlet *Get-AzureRmLocation*.

   Fill in your own values, and when you're ready, press Enter to execute the command. It will take a couple of minutes. When it's done, it will show you your new storage account. It should look like Figure 4-22.



```
ResourceGroupName   : azurebookch4rg
StorageAccountName  : bookch4ps
Id                  : /subscriptions/a6111661-2ef9-[redacted]/resourceGroups/azu
                      rebookch4rg/providers/Microsoft.Storage/storageAccounts/bookch4ps
Location            : westus
AccountType         : StandardLRS
CreationTime        : 5/23/2016 12:55:28 AM
CustomDomain        :
LastGeoFailoverTime :
PrimaryEndpoints    : Microsoft.Azure.Management.Storage.Models.Endpoints
PrimaryLocation     : westus
ProvisioningState   : Succeeded
SecondaryEndpoints  :
SecondaryLocation   :
StatusOfPrimary     : Available
StatusOfSecondary   :
Tags                : {}
Context             : Microsoft.WindowsAzure.Commands.Common.Storage.AzureStorageContext
```

Figure 4-22 The PowerShell output from creating the storage account.

If you log into the Azure portal, you can see your new resource group and the new storage account in the resource group.

Create a container and upload blobs using PowerShell

Now you'll create a container and upload some blobs. In the example, the test files are in D:_TestImages. That path is used when uploading those files to Blob storage.

> **Note** These cmdlets are Azure Storage data-plane cmdlets, not Azure Service Management (ASM) or Azure Resource Manager cmdlets, which are management-plane cmdlets. The cmdlet to create a storage account is a management-plane cmdlet. These data-plane cmdlets can be used with both ASM and Resource Manager storage accounts.

If you’re not running the PowerShell ISE and are logged into your Azure account, do that now. You’re going to create a script that you can save and use later. In addition to the path to your local pictures, you will need the name and access key of your storage account.

1. Set up variable names for the storage account name and key—*$StorageAccountName* and *$StorageAccountKey*. Fill in your storage account name and key here.

   ```
   $StorageAccountName = "yourStorageAccountName"

   $StorageAccountKey = "yourStorageAccountKey"
   ```

2. Next, you’ll define the storage account context using the storage account name and key. You will use this context for authentication with subsequent commands against the storage account. This is easier (and safer) than specifying the storage account name and key all the time.

   ```
   $ctx = New-AzureStorageContext -StorageAccountName $StorageAccountName `

       -StorageAccountKey $StorageAccountKey
   ```

   Note that there is a continuation character (the backward tick mark) at the end of the first line.

3. Next, you’ll add a variable for the name of your container, then you’ll create the container. The example uses *test-ps*.

   ```
   $ContainerName = "test-ps"

   #create a new container with public access to the blobs

   New-AzureStorageContainer -Name $ContainerName -Context $ctx -Permission Blob
   ```

   This creates a container in your storage account (as defined by the context) with a permission of Blob, which means the blobs can be accessed on the Internet with a URL.

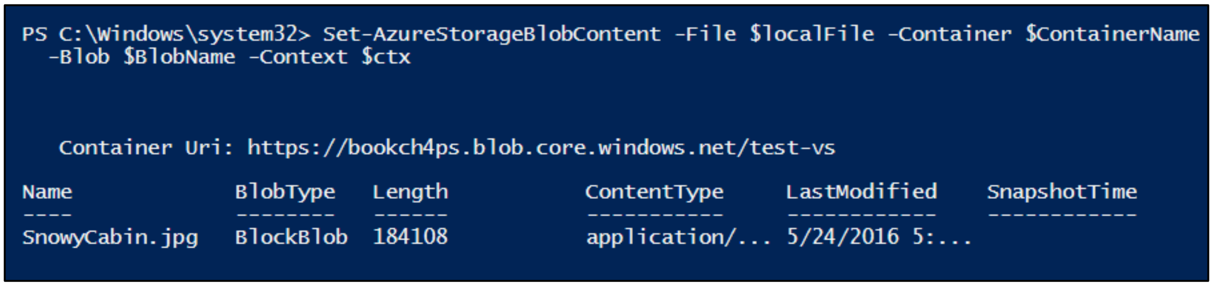
4. Now you need to set a variable pointing at the local directory with the images. You can upload any files, just remember the larger they are, the longer it will take to upload! Using a variable here makes it easier to change it later in case you use this in multiple places.

   ```
   $localFileDirectory = "D:\_TestImages\"
   ```

5. Now you can upload a blob. First, you’ll set a variable name for the blob name to be the same as the file name. Then, append it to the *$localFileDirectory* variable. The file will be uploaded from the local disk to the specified container.

   ```
   $BlobName = "SnowyCabin.jpg"

   $localFile = $localFileDirectory + $BlobName

   Set-AzureStorageBlobContent -File $localFile -Container $ContainerName `

     -Blob $BlobName -Context $ctx
   ```

   To run the script, press F5. To run parts of the script, highlight the bits you want to run and press F8 (or click the Run Selection icon). If you have to run this repeatedly, you only want to create the container once, so once that’s successful, only select commands starting after that. When you run this and upload the file, you get back verification in the command window (Figure 4-23).



```
PS C:\Windows\system32> Set-AzureStorageBlobContent -File $localFile -Container $ContainerName
  -Blob $BlobName -Context $ctx


   Container Uri: https://bookch4ps.blob.core.windows.net/test-vs

Name                BlobType   Length           ContentType      LastModified     SnapshotTime
----                --------   ------           -----------      ------------     ------------
SnowyCabin.jpg      BlockBlob  184108           application/...  5/24/2016 5:...
```

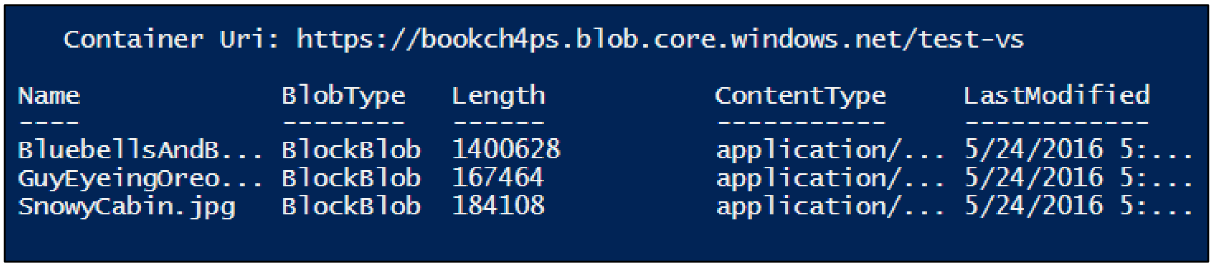
Figure 4-23 Upload file to blob storage.

6. To upload more files, copy and paste the three lines of PowerShell, changing the *$BlobName* variable for each set you paste.
7. After uploading some files, you can list them by using the *Get-AzureStorageBlob* PowerShell cmdlet.

   ```
   # get list of blobs and see the new one has been added to the container
   ```

   ```
   Get-AzureStorageBlob -Container $ContainerName -Context $ctx
   ```

In our example, the list is displayed in Figure 4-24.



```
   Container Uri: https://bookch4ps.blob.core.windows.net/test-vs

Name                BlobType   Length           ContentType      LastModified
----                --------   ------           -----------      ------------
BluebellsAndB...    BlockBlob  1400628          application/...  5/24/2016 5:...
GuyEyeingOreo...    BlockBlob  167464           application/...  5/24/2016 5:...
SnowyCabin.jpg      BlockBlob  184108           application/...  5/24/2016 5:...
```

Figure 4-24 List of files uploaded to blob storage.

You can also see the container and blobs if you log into the Azure portal and go to the storage account.

There are also PowerShell commands for downloading blobs, deleting blobs, copying blobs, etc.

Create a file share and upload files using PowerShell

Now you're going to create a file share in the storage account and upload some files to it using PowerShell. This is very similar to the PowerShell for uploading blobs.

In our example, the storage account is called bookch4ps; the test files are in D:_TestImages. That path is needed when uploading those files to File storage.

If needed, run the PowerShell ISE and log into your Azure account. You're going to create a script that you can save and use later. In addition to the path to your local pictures, you will need the name and access key of your storage account.

1. Set up variable names for the storage account name and key: *$StorageAccountName* and *$StorageAccountKey*. Fill in your storage account name and key.

   ```
   $StorageAccountName = "yourStorageAccountName"
   ```

   ```
   $StorageAccountKey = "yourStorageAccountKey"
   ```

2. Next, you'll define the storage account context using the storage account name and key. You will use this context for authentication with subsequent commands against the storage account. This is easier (and safer) than specifying the storage account name and key all the time.

   ```
   $ctx = New-AzureStorageContext -StorageAccountName $StorageAccountName `
       -StorageAccountKey $StorageAccountKey
   ```

   Note that there is a continuation character at the end of the first line—the backward tick mark.

3. Now you'll set the variable for the name of the file share to whatever you like; the example will use *psfileshare*. Then, you'll create the new file share, assigning it to the variable *$s*.

   ```
   $shareName = "psfileshare"
   $s = New-AzureStorageShare $shareName -Context $ctx
   ```

4. Now set a variable for the local location of the files to be uploaded.

   ```
   $localFolderName = "D:\_TestImages\"
   ```

5. Now you can do the actual upload of the file. Set a variable for the file name, create the local path (directory + file name), and then use the PowerShell cmdlet *Set-AzureStorageFileContent* to upload the file.

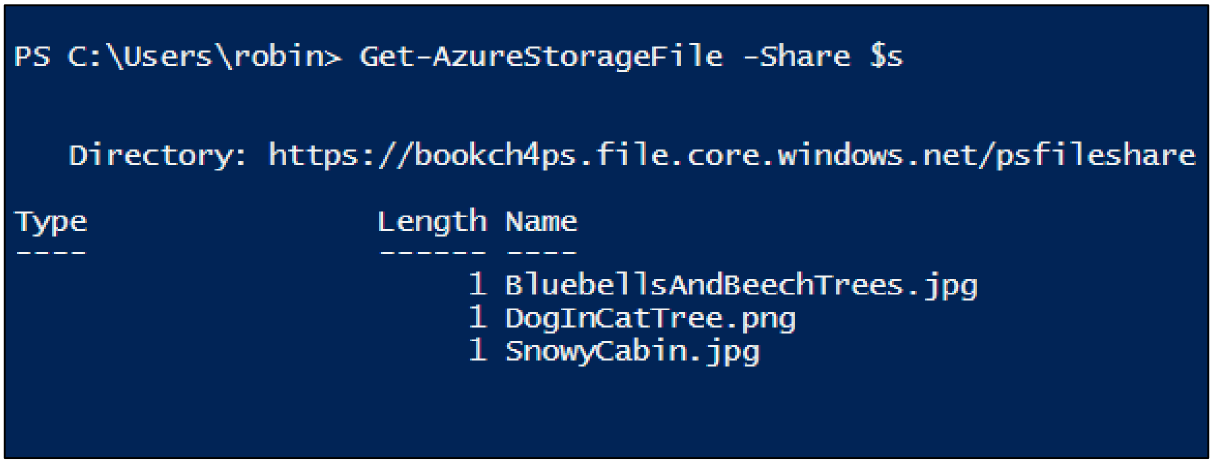
   ```
   $fileName = "DogInCatTree.png"
   $localFile = $localFolderName + $fileName
   Set-AzureStorageFileContent -Share $s -Source $localFile -Path images
   ```

6. Copy this a couple of times and run it with different file names to upload multiple files. Now run the script and watch as it echoes the successful commands back to you.

7. You can call *Get-AzureStorageFile* to retrieve the list of files in the root of the file share.

   ```
   Get-AzureStorageFile -Share $s
   ```

8. Figure 4-25 shows the output from the example.



```
PS C:\Users\robin> Get-AzureStorageFile -Share $s


   Directory: https://bookch4ps.file.core.windows.net/psfileshare

Type                 Length Name
----                 ------- ----
                          1 BluebellsAndBeechTrees.jpg
                          1 DogInCatTree.png
                          1 SnowyCabin.jpg
```

Figure 4-25 Files uploaded to the file share.

There are also PowerShell commands for downloading files, deleting files, copying files, etc.

AzCopy: A very useful tool

Before finishing the chapter on Azure Storage, you need to know about AzCopy. This is a free tool provided by the Azure Storage team to move data around. The core use case is asynchronous server-side copies. When you copy blobs or files from one storage account to another, they are not downloaded from the first storage account to your local machine and then uploaded to the second storage account. The blobs and files are copied directly within Azure.

Here are some of the things you can do with AzCopy:

- Upload blobs from the local folder on a machine to Azure Blob storage.
- Upload files from the local folder on a machine to Azure File storage.
- Copy blobs from one container to another in the same storage account.
- Copy blobs from one storage account to another, either in the same region or in a different region.
- Copy files from one file share to another in the same storage account.
- Copy files from one storage account to another, either in the same region or in a different region.
- Copy blobs from one storage account to an Azure File share in the same storage account or in a different storage account.
- Copy files from an Azure File share to a blob container in the same storage account or in a different storage account.
- Export a table to an output file in JSON or CSV format. You can export this to blob storage.
- Import the previously exported table data from a JSON file into a new table. (Note: It won't import from a CSV file.)

As you can see, there are a lot of possibilities when using AzCopy. It also has a bunch of options. For example, you can tell it to only copy data where the source files are newer than the target files. You can also have it copy data only where the source files are older than the target files. And you can combine these options to ask it to copy only files that don't exist in the destination at all.

AzCopy is frequently used to make backups of Azure Blob storage. Maybe you have files in Blob storage that are updated by your customer frequently, and you want a backup in case there's a problem. You can do something like this:

- Do a full backup on Saturday from the source container to a target container and put the date in the name of the target container.
- For each subsequent day, do an incremental copy—copy only the files that are newer in the source than in the destination.

If your customer uploads a file by mistake, if they contact you before end of day, you can retrieve the previous version from the backup copy.

Here are some other use cases:

- You want to move your data from a classic storage account to a Resource Manager storage account. You can do this by using AzCopy, and then you can change your applications to point to the data in the new location.

- You want to move your data from general-purpose storage to cool storage. You would copy your blobs from the general-purpose storage account to the new Blob storage account, then delete the blobs from the original location.

For more information and a ton of examples, check out https://azure.microsoft.com/documentation/articles/storage-use-azcopy/.

The Azure Data Movement Library

Many people wanted to be able to call AzCopy with their own specialized case. Because of this, the Azure Storage team open sourced the Azure Storage Data Movement Library, giving you programmatic access to AzCopy. For more information, check out the repository and samples on GitHub at https://github.com/Azure/azure-storage-net-data-movement.

CHAPTER 5

Azure Virtual Networks

In this chapter, we take a look at Azure Virtual Networks. We discuss the various components and see how to create a virtual network in Azure. We also see how to set up and use a point-to-site network.

What is a virtual network (VNet)?

Let's take a look at what a virtual network is and what they're used for. We'll define the terms used (such as subnets), and see how to create a VNet in Azure.

Overview

Virtual networks (VNets) are used in Azure to provide private connectivity for Azure Virtual Machines (Azure VMs) and some Azure services. VMs and services that are part of the same virtual network can access one another. By default, services outside the virtual network cannot connect to services within the virtual network. You can, however, configure the network to allow access to the external service.

Services that talk to each other within a virtual network do not travel through the Azure Load Balancer, which gives you better performance. It's not significant, but sometimes every little bit counts.

Here's an example of how you might want to use that virtual network for connectivity to a private resource. Let's say you have a front-end web application running in a VM, and that application uses a back-end database running in a different VM. You can put the back-end database in the same virtual network as the web application; the web application will access the database over the virtual network. This allows you to use the back-end database from the web application without the database being accessible on the public Internet.

A Virtual Network Gateway is a fully managed service in Azure that is used for cross-premises connectivity. You can add a Virtual Network Gateway to a virtual network and use it to connect your on-premises network to Azure, effectively making the virtual network in Azure an extension of your on-premises network. This provides the ability to deploy hybrid cloud applications that securely connect to your on-premises datacenter.

More complex features available include multisite VPNs, in-region VNet-to-VNet, and cross-region VNet-to-VNet. Most cross-premises connections involve using a VPN device to create a secure connection to your virtual network in Azure. VNet-to-VNet connectivity uses the Azure Virtual Network Gateway to connect two or more virtual networks with IPsec/IKE S2S VPN tunnels. Being able to have cross-premises connectivity gives you flexibility when connecting one or more on-premises sites with your virtual networks. For example, it gives you the ability to have cross-region geo-redundancy, such as SQL Always On across different Azure regions.

Definitions

When creating a virtual network, there are a few things you need to know, such as the address space, subnets, and DNS servers that you want to use.

Virtual network address spaces

When you set up a virtual network, you specify the topology of the virtual network, including the available address spaces and subnets. If the virtual network is to be connected to other virtual networks, you must select address ranges that are not overlapping. This is the range of addresses that the VMs and services in your network can use. These will be private and cannot be accessed from the public Internet. This used to be true only for the unrouteable IP addresses such as 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16, but now Azure will treat any address range as part of the private VNet IP address space that is only reachable within the VNet, within interconnected VNets, and from your on-premises location.

CIDR specifies an address range using a combination of an IP address and its associated network mask. CIDR notation uses this format: xxx.xxx.xxx.xxx/n, where *n* is the number of leftmost "1" bits in the mask. For example, 192.168.12.0/23 applies the network mask 255.255.254.0 to the 192.168 network, starting at 192.168.12.0. This notation therefore represents the address range 192.168.12.0–192.168.13.255. Fortunately, the Azure portal displays this information on the screen after you type the CIDR into one of the address space fields so you don't have to do bit-wise math to figure it out!

10.0.0.0/8 gives you a usable address range of 10.0.0.0–10.255.255.255. You can certainly use this, but if 10.0.0.0 is being used elsewhere in the network, either on-premises or within Azure, you want to be sure you have no overlap. One way to do this is to specify an address space that is smaller but still has the capacity to hold everything you want to put in it. For example, if you are just going to put a handful of VMs in your virtual network, you could use 10.0.0.0/27, which gives you a usable address range of 10.0.0.0–10.0.0.31.

If you work within an organization in which someone else is responsible for the internal networks, you should confer with that person before selecting your address space to make sure there is no overlap and to let them know what space you want to use so they don't try to use the same range of IP addresses.

Subnets

After specifying your virtual network address space(s), you can create one or more subnets for your virtual network. You do this to break up your network into more manageable sections. For example, you might assign 10.1.0.0 to VMs, 10.2.0.0 to back-end services, and 10.3.0.0 to SQL Server VMs. Note that Azure reserves the first four addresses and the last address in each subnet for its own use.

By default, there is no security boundary between subnets, so services in each of these subnets can talk to one another. However, you can now set up Network Security Groups (NSGs), which allow you to control the traffic flow to and from subnets and to and from VMs. We'll talk more about these later in this chapter.

DNS servers

If you want to refer to your VMs by host name or fully qualified domain name (FQDN) directly, rather than using an IP address and port number, you need a DNS service to provide name resolution. It is helpful to figure this out before deploying VMs or role instances. You can change it later, but if you do you will have to reboot all of the VMs in the virtual network because the DNS server information is injected into the settings at startup.

There are two options: you can use the Azure-provided name resolution or you can specify a DNS server that is not maintained by Azure, such as one that is used by your on-premises infrastructure or one that you set up and maintain in an Azure VM.

If you need name resolution between VMs located in different virtual networks, resolution of on-premises computer names from VMs in Azure, or resolution of Azure hostnames from on-premises computers, you need to use your own DNS server.

For more information, please take a look at this article on Name Resolution (DNS) for VMs and Role Instances: https://azure.microsoft.com/documentation/articles/virtual-networks-name-resolution-for-vms-and-role-instances/.

Creating a virtual network

To put VMs into a virtual network, you create the virtual network and then as you create each VM, you assign it to the virtual network and subnet. VMs acquire their network settings during deployment or startup.

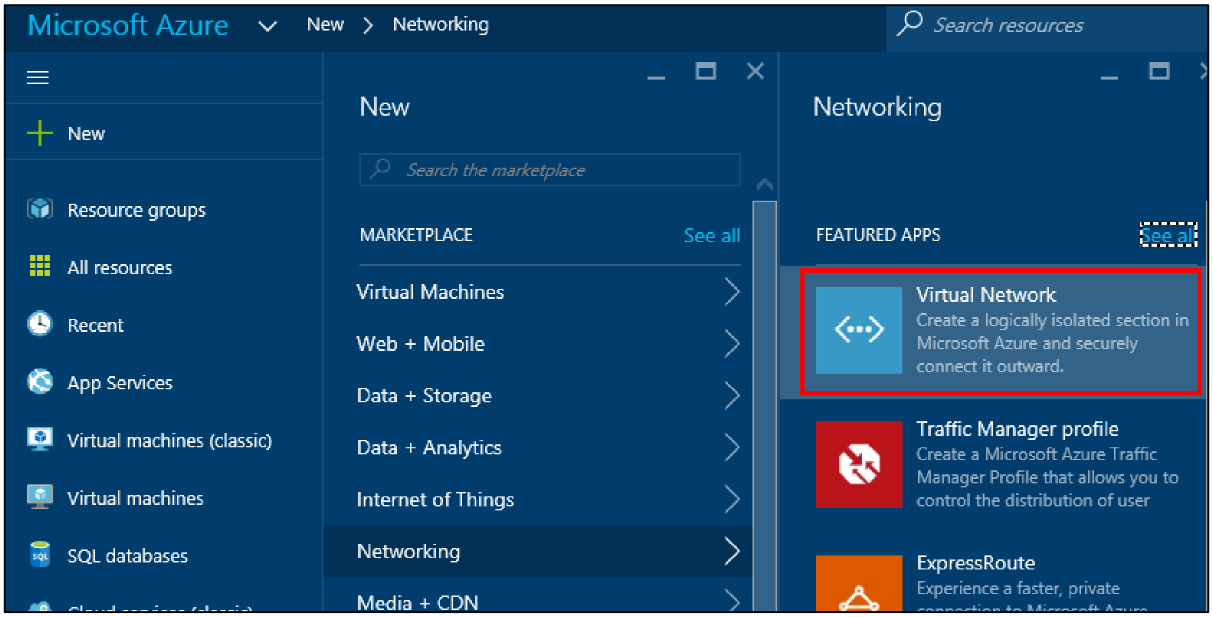
VMs are assigned an IP address when they are deployed. If you deploy multiple VMs into a virtual network or subnet, they are assigned IP addresses as they boot up. A dynamic IP address (DIP) is the internal IP address associated with a VM. You can allocate a static DIP to a VM. If you do this, you should consider using a specific subnet for static DIPs to avoid accidentally reusing a static DIP for another VM.

If you create a VM and later want to migrate it into a virtual network, it is not a simple configuration change. You have to redeploy the VM into the virtual network. The easiest way to do this is to delete the VM, but not any disks attached to it, and then re-create the VM using the original disks in the virtual network.

Creating a virtual network using the Azure portal

Now I'll show you how to create a virtual network using the Azure portal. You'll create a VNet with two subnets—one with the address range 10.0.0.0/24 and one with 10.0.1.0/24.

1. Log into the Azure portal. Click New > Networking > Virtual Network, as shown in Figure 5-1.



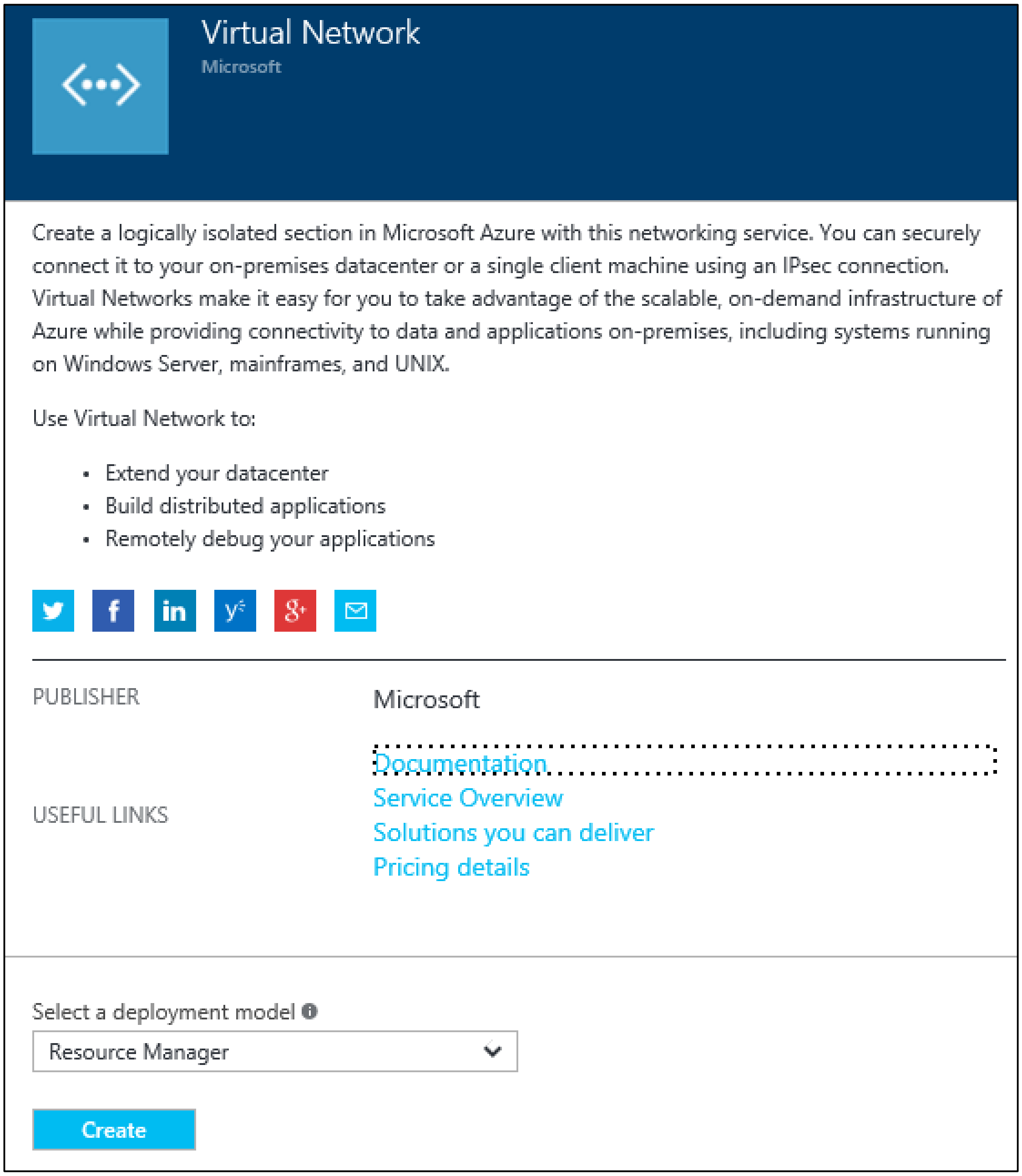
Virtual Network: Create a logically isolated section in Microsoft Azure and securely connect it outward.

Traffic Manager profile: Create a Microsoft Azure Traffic Manager Profile that allows you to control the distribution of user

ExpressRoute: Experience a faster, private

Figure 5-1 Use the Azure portal menu to create a new virtual network.

2. The Virtual Network blade in Figure 5-2 appears.



Virtual Network

Microsoft

Create a logically isolated section in Microsoft Azure with this networking service. You can securely connect it to your on-premises datacenter or a single client machine using an IPsec connection. Virtual Networks make it easy for you to take advantage of the scalable, on-demand infrastructure of Azure while providing connectivity to data and applications on-premises, including systems running on Windows Server, mainframes, and UNIX.

Use Virtual Network to:

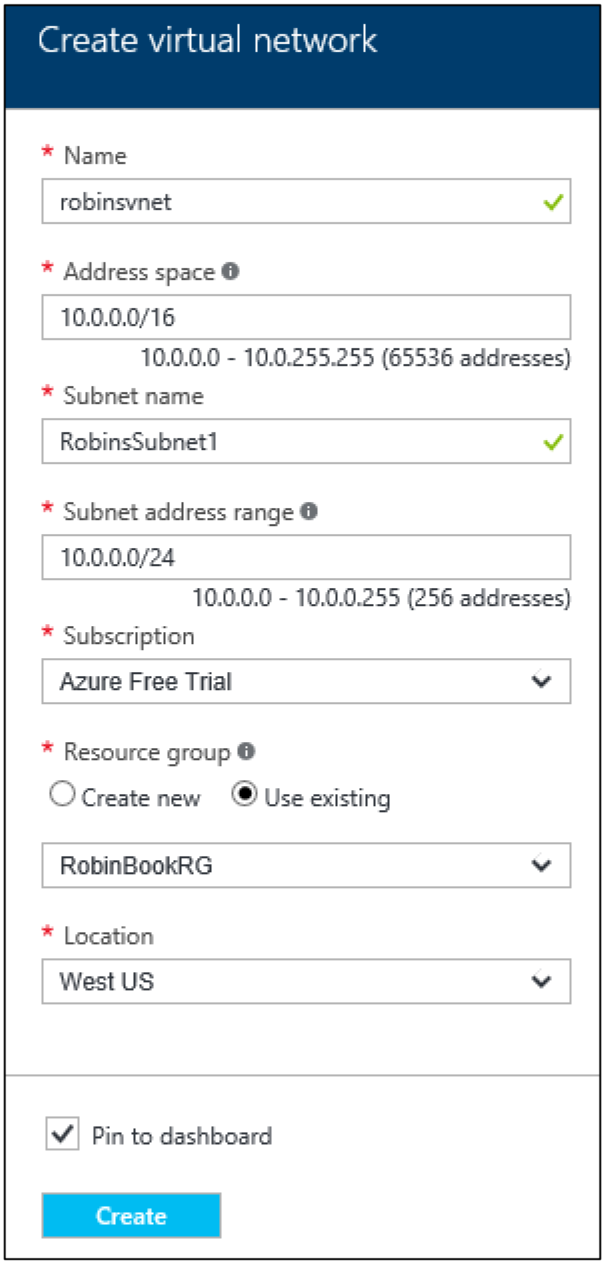
- Extend your datacenter
- Build distributed applications
- Remotely debug your applications

PUBLISHER	Microsoft
USEFUL LINKS	Documentation Service Overview Solutions you can deliver Pricing details

Select a deployment model: Resource Manager

Figure 5-2 The Virtual Network blade.

3. Set the Deployment Model to Resource Manager and click Create. The Create Virtual Network blade in Figure 5-3 appears.



Create virtual network

* Name

robinsvnet ✓

* Address space

10.0.0.0/16

10.0.0.0 - 10.0.255.255 (65536 addresses)

* Subnet name

RobinsSubnet1 ✓

* Subnet address range

10.0.0.0/24

10.0.0.0 - 10.0.0.255 (256 addresses)

* Subscription

Azure Free Trial

* Resource group

○ Create new ● Use existing

RobinBookRG

* Location

West US

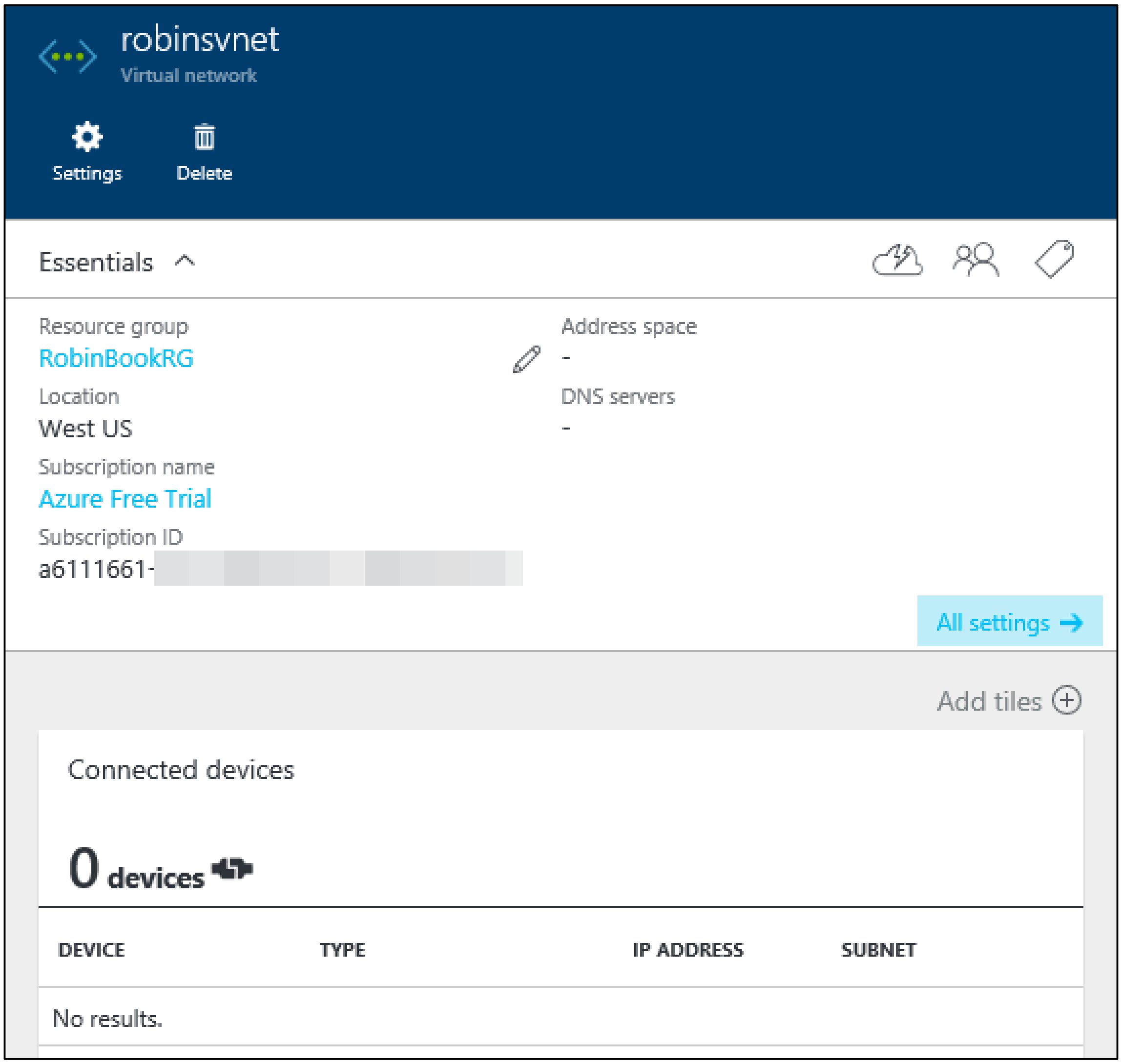
☑ Pin to dashboard

Figure 5-3 Create Virtual Network blade.

4. Specify a name for the VNet.
5. For Address Space, specify the whole range for the VNet in CIDR notation. In our example, set this to 10.0.0.0/16.
6. Next, specify the name of the subnet in the VNet and then specify the address range. This must be within the range of the whole VNet; set this to 10.0.0.0/24.
7. If you have multiple subscriptions administered by the account with which you're logged in, select the subscription in which you want to create the VNet.
8. Next, select the resource group—in this case, select the one that you created earlier. You can also ask to create a new resource group and specify a name for it.

9. Finally, specify the Location. Select the Azure region in which you want to create the VNet.

10. When you're finished, select the Pin To Dashboard check box and click Create to create the virtual network as specified. The network is created and the blade for the virtual network appears, as shown in Figure 5-4.



robinsvnet
Virtual network

Resource group
RobinBookRG

Location
West US

Subscription name
Azure Free Trial

Subscription ID
a6111661-[redacted]

Address space
-

DNS servers
-

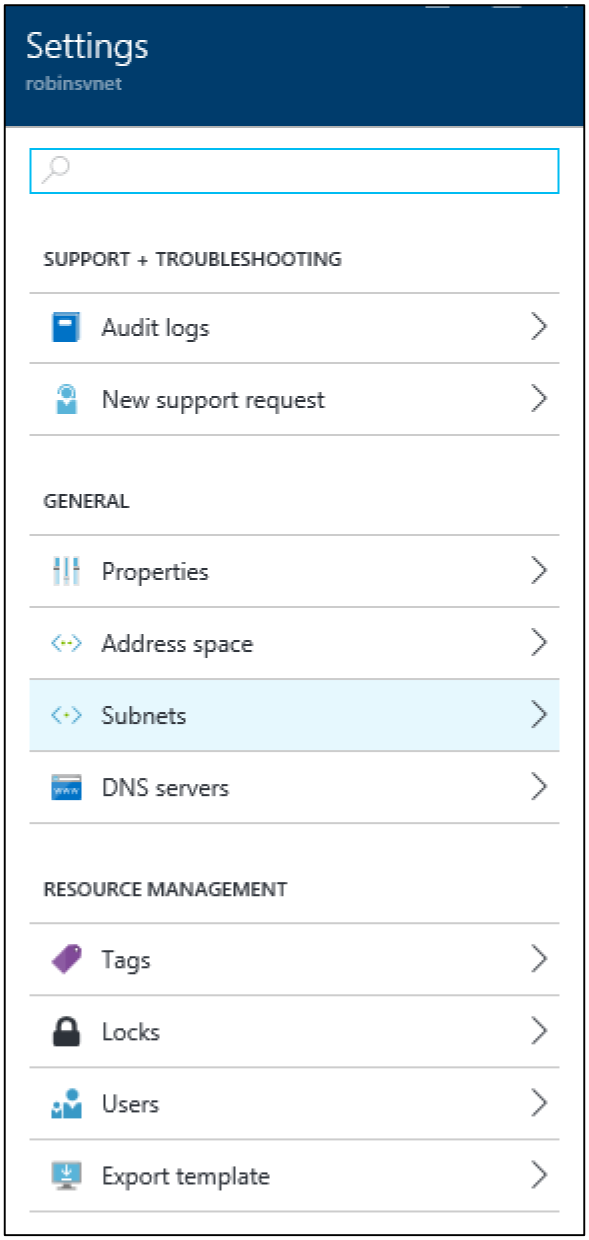
Connected devices

0 devices

DEVICE	TYPE	IP ADDRESS	SUBNET
No results.			

Figure 5-4 The Virtual Network blade.

11. Now add a second subnet. Go to the Settings blade for the virtual network (Figure 5-5).



Settings

robinsvnet

SUPPORT + TROUBLESHOOTING

- Audit logs
- New support request

GENERAL

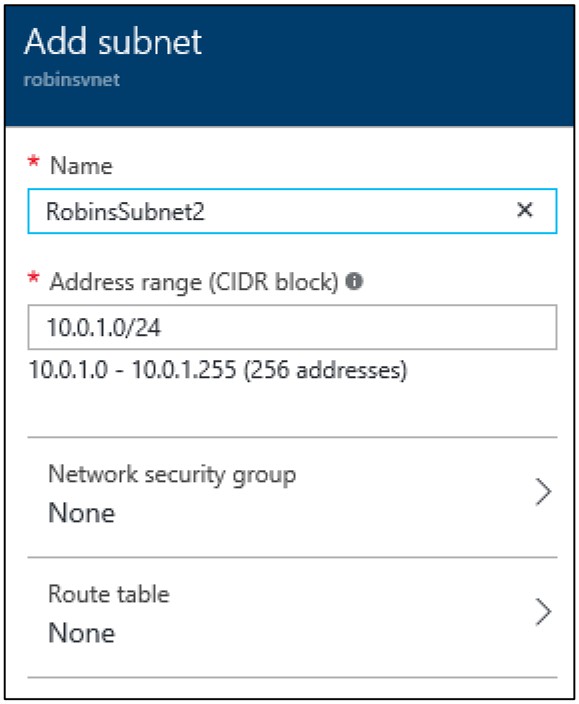
- Properties
- Address space
- Subnets
- DNS servers

RESOURCE MANAGEMENT

- Tags
- Locks
- Users
- Export template

Figure 5-5 The Settings blade for the new virtual network.

12. Click Subnets in the Settings blade. It shows the subnet that was initially created. Create another one with address range 10.0.1.0/24. In the Add Subnet blade that appears (Figure 5-6), fill in the subnet name and address range and click OK.



Add subnet

robinsvnet

* Name

RobinsSubnet2

* Address range (CIDR block)

10.0.1.0/24

10.0.1.0 - 10.0.1.255 (256 addresses)

Network security group

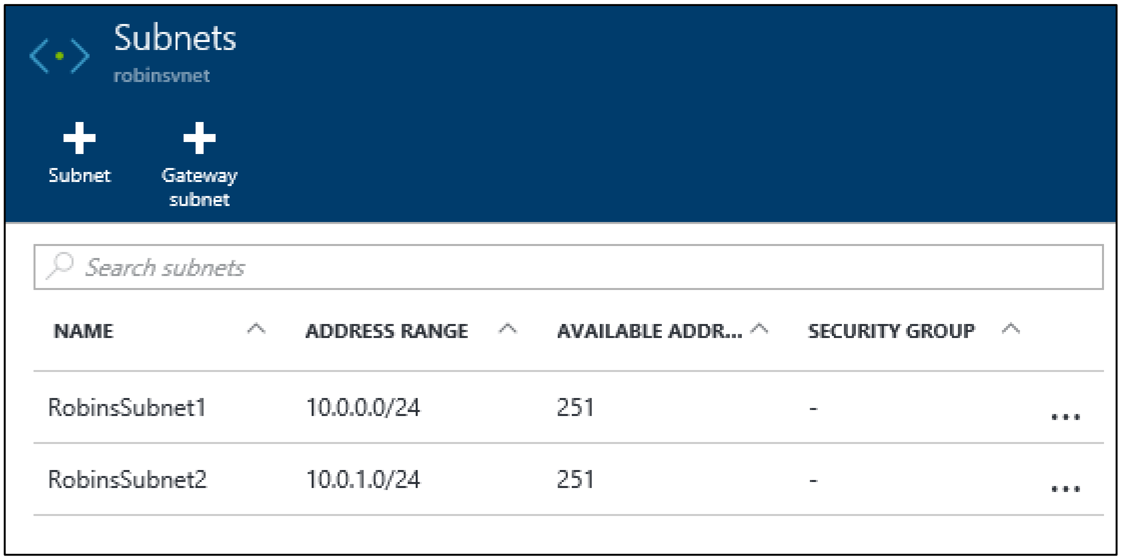
None

Route table

None

Figure 5-6 Add a second subnet to the virtual network.

13. Click OK to add the subnet. When it's finished, you see both of your subnets in the list (Figure 5-7).



Subnets

robinsvnet

NAME	ADDRESS RANGE	AVAILABLE ADDR...	SECURITY GROUP
RobinsSubnet1	10.0.0.0/24	251	-
RobinsSubnet2	10.0.1.0/24	251	-

Figure 5-7 The list of subnets defined for the virtual network.

Now you have a virtual network with two subnets. In this case, you could deploy your VMs into the first subnet and your data services (such as SQL Server VMs) into the second subnet. They could see each other, but they would reside on separate subnets.

Creating a virtual network using a Resource Manager template

Another way to create a virtual network is to create a Resource Manager template for your virtual network and deploy it by using the Azure portal. To do this, you choose Template Deployment from the Azure Marketplace, copy and paste your template into the template editor in the portal, then assign it to the appropriate resource group. Let's see how this works.

You can start with one of the Azure Quickstart Templates, which are here: https://github.com/Azure/azure-quickstart-templates. Or you can create your own from scratch (this is more difficult because you have to get the format and syntax exactly right). You can also export a template from the Azure portal for a resource group and edit it.

We're going to start with a basic template that creates a virtual network with two subnets. Here are the specifications:

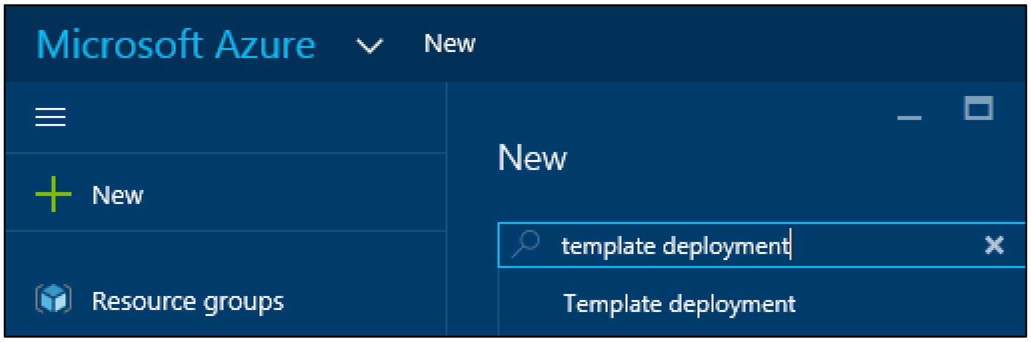
- Location: WestUS.
- VNet: name = AzureBookVNet, address prefix is 192.168.0.0/16.
- Subnet #1: name = SubnetOne, address prefix is 192.168.20.0/24.
- Subnet #2: name = SubnetTwo, address prefix is 192.168.30.0/24.

The template includes all of this information. It includes parameters for the input specifications with default values that you can override after loading the template in the portal. If you look through the following JSON template, you will see there is a restriction on allowed values for the location, and you will see the VNet information specified.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "West US",
      "allowedValues": [
          "West US",
          "East US"
       ],
      "metadata": {
        "description": "Deployment location"
      }
    },
    "vnetName": {
      "type": "string",
      "defaultValue": "AzureBookVNet",
      "metadata": {
        "description": "VNet name"
      }
    },
    "vnetAddressPrefix": {
      "type": "string",
      "defaultValue": "192.168.0.0/16",
      "metadata": {
        "description": "Address prefix"
      }
    },
    "subnet1Prefix": {
      "type": "string",
      "defaultValue": "192.168.20.0/24",
      "metadata": {
        "description": "Subnet 1 Prefix"
      }
    },
    "subnet1Name": {
      "type": "string",
      "defaultValue": "SubnetOne",
      "metadata": {
        "description": "Subnet 1 Name"
      }
    },
    "subnet2Prefix": {
      "type": "string",
      "defaultValue": "192.168.30.0/24",
      "metadata": {
        "description": "Subnet 2 Prefix"
      }
    },
    "subnet2Name": {
      "type": "string",
      "defaultValue": "SubnetTwo",
      "metadata": {
        "description": "Subnet 2 Name"
      }
    }
  },
  "variables": { },
  "resources": [
    {
      "apiVersion": "2015-06-15",
      "type": "Microsoft.Network/virtualNetworks",
      "name": "[parameters('vnetName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vnetAddressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[parameters('subnet1Name')]",
            "properties": {
              "addressPrefix": "[parameters('subnet1Prefix')]"
            }
          },
          {
            "name": "[parameters('subnet2Name')]",
            "properties": {
              "addressPrefix": "[parameters('subnet2Prefix')]"
            }
          }         ]
      }
    }
  ]
}
```

Now deploy this template through the Azure portal. When you do this, you will create a new resource group for your network.

1. Log into the Azure portal. Click New, then type **template deployment** into the search box, as seen in Figure 5-8. When it displays in the search list, select it.

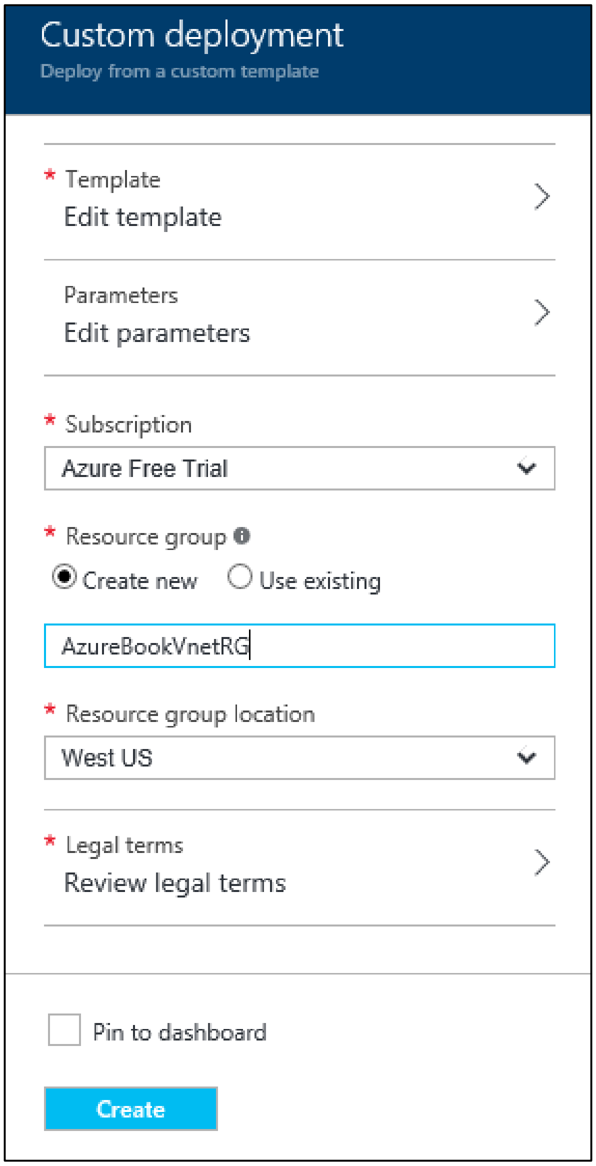


template deployment

Template deployment

Figure 5-8 Search for template deployment.

2. In the search results, click Template Deployment. This shows the Template Deployment blade that gives a brief explanation of what it is. Click Create. This shows the Custom Deployment blade shown in Figure 5-9.


Custom deployment

Deploy from a custom template

* Template
Edit template

Parameters
Edit parameters

* Subscription
Azure Free Trial

* Resource group
● Create new ○ Use existing
AzureBookVnetRG

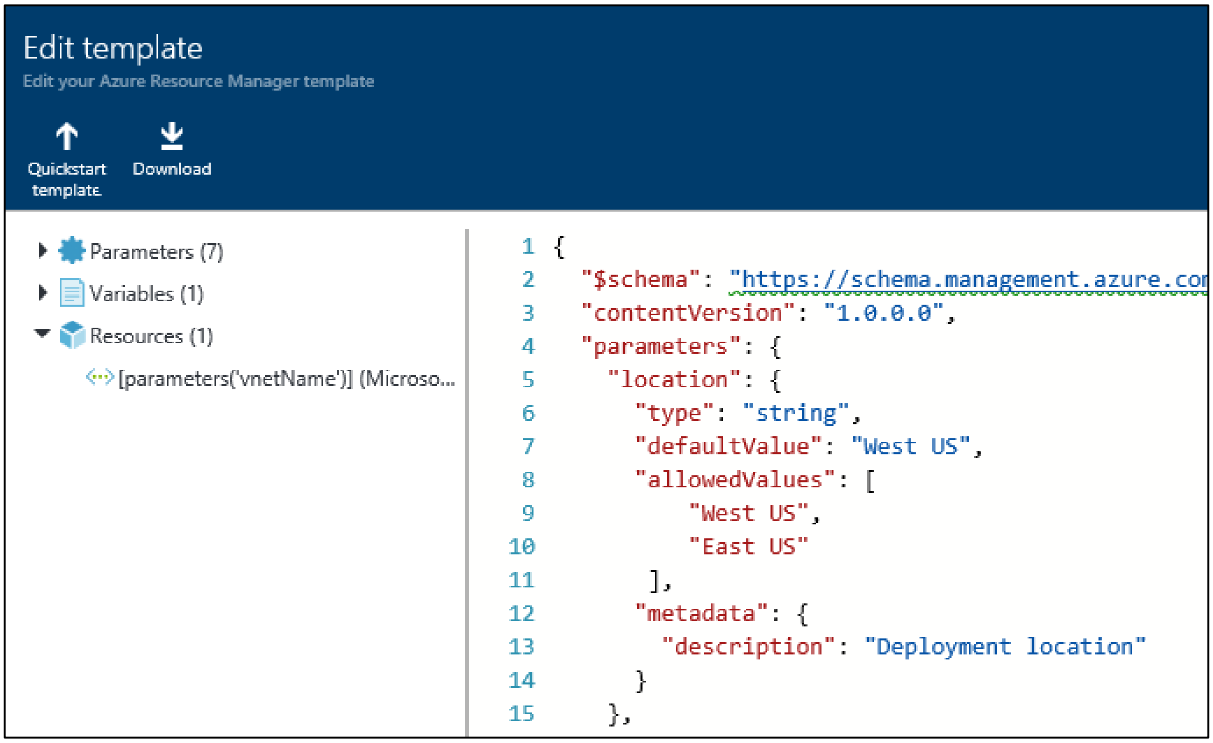
* Resource group location
West US

* Legal terms
Review legal terms

□ Pin to dashboard

Figure 5-9 The Custom Deployment blade.

3. Click Edit Template. This shows the template editor. Copy the template JSON that you want to use from a text editor and paste it into the template editor, replacing everything that's there. You see something like Figure 5-10.



```
{
  "$schema": "https://schema.management.azure.co
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "West US",
      "allowedValues": [
          "West US",
          "East US"
       ],
      "metadata": {
        "description": "Deployment location"
      }
    },
```

Figure 5-10 The Edit Template blade.

4. Click Save. If there are no parsing errors in your template, it returns to the Custom Deployment blade. If you have parameters, the Azure portal marks that option with an error symbol. You must click Edit Parameters and verify the values of the parameters. You can change them from the defaults here if desired.

5. On the Custom Deployment blade, verify the subscription selected is correct and then select Create New under Resource Group and provide a name. As you can see back in Figure 5-9, our example uses AzureBookVnetRG.

6. Set the location of the Resource Group. Although not required, it's best to specify the same Azure region that will be used for the resources when they are created.

7. Next, click Review Legal Terms. After reading the legal terms as carefully as you read all legal terms, click Purchase to indicate that you understand you are creating resources in Azure and will have to pay for them.

8. Click Create. The Azure portal will now verify the template, create your new resource group, and then create your virtual network in that resource group.

9. When it's finished, select it from the Dashboard if it's there, or select All Resources and then select your new VNet. It will open the Virtual Network blade. If you click Subnets in the Settings blade, you see your two subnets.

That's grand, but what if you want to add another subnet or change something about this deployment? You can edit the template and deploy it again. By default, the Azure Resource Manager treats deployments as incremental updates to the resource group. Here are the rules:

- If you remove a resource from the template that is not in the resource group, that resource will be unchanged. It won't be removed simply because it's gone from the template. (If you want to remove a resource, you have to specifically remove it using the Azure portal, PowerShell, the Azure CLI, etc.)
- If you add a resource to the template that is not in the resource group, it will create that resource for you when you redeploy the template.
- Resources that are unchanged but are still in the template will be ignored.
- Resources that are changed and still in the template will be updated. For example, if we change the address prefixes of our virtual network and redeploy the template, it will change them in the deployed VNet.

This means you can repeatedly edit a template and add or change a section, and when you redeploy it, it applies the changes and ignores the current (unchanged) specs for the other resources.

If you use PowerShell, you can also do a complete deployment, which *does* delete resources that are in the resource group but not in the template when you redeploy. You can't do complete deployments through the Azure portal. It's safer to delete the resource group yourself (and verify that that's what you want to do) than to request a complete deployment. What if you accidentally remove a section from your template? It will remove that resource, and you can't recover it. If you're not convinced that it's dangerous, think about the consequences if you accidentally remove a storage account in which you've already stored data. Ouch.

Now you can try out this theory of adding something and seeing it be deployed with the existing resources. Let's add a third subnet to the template and redeploy it.

1. Edit the template. Assuming you used the template given before, copy the sections for subnet2 and paste them after subnet2 and change the "subnet2" to "subnet3." Also change the address prefix and the name of the subnet. You should have something like this right after subnet2Name in the template I posted above:

   ```
   "subnet3Prefix": {
     "type": "string",
     "defaultValue": "192.168.40.0/24",
     "metadata": {
       "description": "Subnet 3 Prefix"
     }
   },
   "subnet3Name": {
     "type": "string",
     "defaultValue": "SubnetThree",
     "metadata": {
       "description": "Subnet 3 Name"
     }
   }
   ```

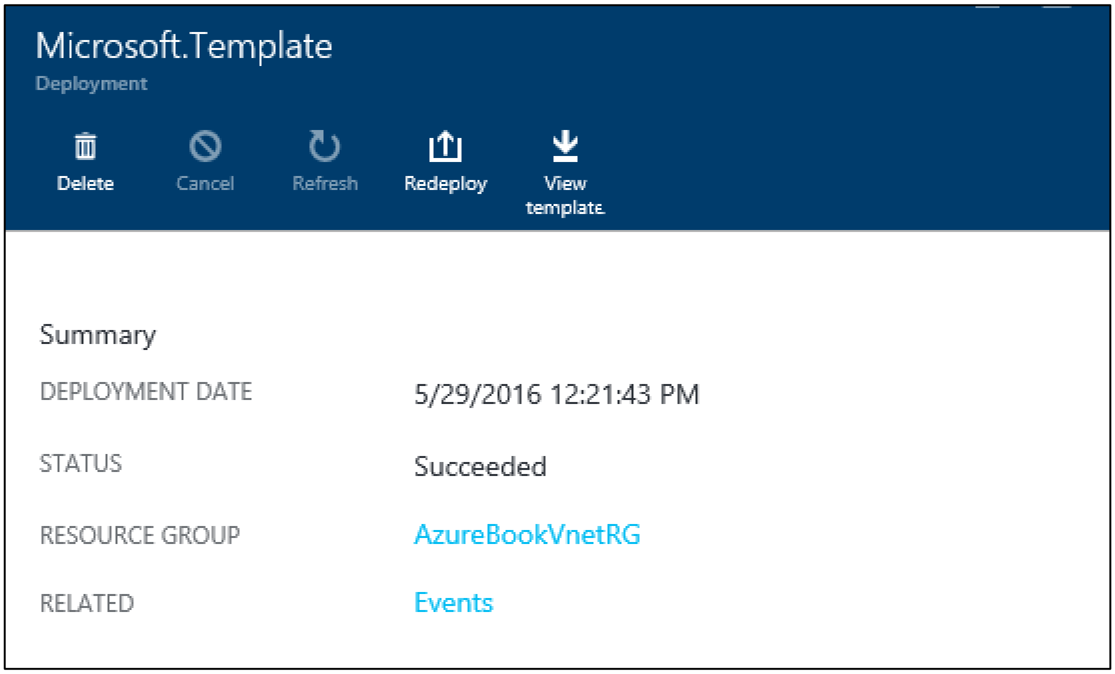
   And in the bottom half, under resources, add a resource for subnet3.

   ```
   {
     "name": "[parameters('subnet3Name')]",
     "properties": {
   ```

```
        "addressPrefix": "[parameters('subnet3Prefix')]"
      }
    }
```

(Don't forget the commas after the entry just above these as you add them.)

2. Now in the portal, go to Resource Groups and select the resource group with your new VNet in it.
3. In the Settings blade, select Deployments. This shows the history of deployments to the resource group. Click the most recent one. The Microsoft.Template blade appears (Figure 5-11).



Summary

DEPLOYMENT DATE	5/29/2016 12:21:43 PM
STATUS	Succeeded
RESOURCE GROUP	AzureBookVnetRG
RELATED	Events

Figure 5-11 The Microsoft.Template blade, where you can redeploy the template for the resource group.

4. Click Redeploy. This puts you at the Custom Deployment blade again. Edit the template and add the two new sections for the third subnet. Click Save.
5. On the Custom Deployment blade, you will have to click Edit The Parameters again to change or verify them. For Resource Group, select Use Existing and specify the current resource group. Review the legal terms again and click Create.
6. It will look at the template, ignore the unchanged sections, and add the new subnet.
7. After it's finished deploying, it returns to the Resource Group blade. Select the VNet in the list of resources in the resource group, then click Subnets, and you see your third subnet has been added.

You can also use PowerShell, the Azure CLI, or even the REST APIs to deploy a template to your Azure subscription. To be honest, using PowerShell is much easier than using the Azure portal to deploy resources using a template.

Network Security Groups

When you create a VM, by default it's going to ask you to create a Network Security Group (NSG). You don't *have* to create one. You can create your VMs in Azure and network them together without an

NSG. However, if a VM has a public IP address, it is hosted on the public Internet, making it subject to attack. This means there is nothing protecting your VMs except the internal Windows firewall.

Microsoft created NSGs to provide a flexible method for defining the access rules allowing traffic into and out of a VM in a VNet—or even an entire subnet in the VNet. When a Windows Server with a public IP address is created in the portal, an NSG is created that blocks all inbound Internet traffic except RDP on port 3389. Similarly, for a Linux VM with a public IP address, the default NSG created blocks all inbound traffic from the Internet except SSH on port 22. You have to specifically open any other ports you want open, including HTTP and HTTPS. If you do nothing, you are protected by default. The same set of rules can be applied to a single VM or multiple VMs. You can also apply an NSG to a subnet, which applies it to all of the VMs in that subnet.

For example, let's say you have four VMs running front-end applications. These connect to eight back-end servers that consist of web services and database servers. You could create one NSG that says "allow access to/from the public Internet, and access to the back end" and apply that to all the front-end VMs. Then, you can create another NSG that says "allow access from these four front-end servers, and allow access to the internal Azure services, but don't allow access to the public Internet" and apply it to the back-end servers. The back-end servers will not be accessible from the public Internet. Note that NSGs are actually applied to a NIC attached to a VM (rather than the VM itself). If a VM has multiple NICs, the NSG needs to be applied separately to each NIC.

If we later add two front-end servers to our resources, we can simply assign them to the same NSG as the other front-end servers and add the new servers to the allowed servers for the back end. This allows you to implement changes with no updates to the running VMs themselves.

How are the rules applied? You can apply an NSG to a VM or to a subnet in a VNet. When you apply an NSG to a subnet, it applies it to all of the VMs in that subnet. For incoming traffic, the rules are applied before the traffic enters the VM. For outgoing traffic, they are applied after the traffic leaves the VM. This means the rules cannot be changed by a user process on the VM or even by the operating system because the rules are external to the VM.

For more details about Network Security Groups, please check out https://azure.microsoft.com/documentation/articles/virtual-networks-nsg/. For an example and instructions for setting up an NSG in the Azure portal, check out https://azure.microsoft.com/documentation/articles/virtual-networks-create-nsg-arm-pportal/.

Cross-premises connection options

There are many cases in which you might want to connect to your infrastructure in Azure from your on-premises network, a customer's site, your home network, or even a coffee shop, and you want to do this without compromising security. There are three options available in Azure to help you set up these cross-premises connections: site-to-site VPN, point-to-site VPN, and private VPN (Azure ExpressRoute).

A VPN Gateway is an Azure managed service that is deployed into a VNet and provides the endpoint for VPN connectivity for point-to-site VPNs, site-to-site VPNs, and ExpressRoute. This gateway is the connection point into Azure from either the on-premises network (site-to-site) or the client machine (point-to-site). You'll see how to do this when we set up a point-to-site network later in this chapter.

Site-to-site connectivity

A site-to-site VPN lets you connect securely from your on-premises network to your virtual network in Azure. You have to have a public-facing IPv4 IP address and a compatible VPN device or Routing and Remote Access Service (RRAS) running on Windows Server 2012. For a list of valid devices and the configuration thereof, please refer to https://azure.microsoft.com/documentation/articles/vpn-gateway-about-vpn-devices/.

Once you have the connection up and running, resources on your local network such as computers and VMs can communicate with the resources in the virtual network on Azure. For example, if you host a company application on Azure, your employees can access and run that application securely using your site-to-site network.

You actually can use site-to-site connectivity to connect entire on-premises networks to virtual networks in Azure. A good example is a company that has multiple branch offices. You can establish a connection between each branch office's network and Azure.

Point-to-site connectivity

Point-to-site VPN enables you to connect from your local machine over a Secure Socket Tunneling Protocol (SSTP) tunnel to your virtual network in Azure. This uses certificate authentication between the client machine and the virtual network in Azure. This means you have to create some certificates and install them in the right places; we'll cover this in detail later in this chapter when we create a point-to-site network.

It is recommended that you create a separate client certificate for each client that is going to access the point-to-site network and keep track of the certificate's thumbnail and on which machine it was installed. If you do this, and you later need to turn off access for one person, you can do that by invalidating the client certificate using the Azure subscription ID, the virtual network name, and the certificate thumbprint.

If you use the same client certificate on multiple machines, the only way to revoke access is to remove the root certificate in Azure, which revokes access for every client certificate that chains back to that root certificate.

You can connect up to 128 clients to the virtual network in Azure. (The maximum bandwidth is 80 Mbps per gateway.) The connection has to be configured on each client machine that you want to use. Once configured, the user can start the VPN by starting the connection manually, although you can configure the VPN to start automatically if needed.

We are going to cover how to set up a point-to-site network in depth later in this chapter.

Comparing site-to-site and point-to-site connectivity

There are several differences between these two forms of secure connections:

- You don't need a VPN device or RRAS to use a point-to-site network.
- With point-to-site, configuration must be done on each client machine. With site-to-site, no changes are required to the client machines.
- Point-to-site is a good choice when:
  - You only have a few clients that need to have access.
  - You don't have access to a VPN device that you can use for a site-to-site connection.
  - You want to connect securely when off site (such as at a customer site or a coffee shop).
- You can have both point-to-site and site-to-site networks running simultaneously. If you can create a site-to-site network, you might use site-to-site for people on premises but allow point-to-site for people who need to connect from a remote location.

Private site-to-site connectivity (ExpressRoute)

Last, but not least, is private site-to-site connectivity, which in Azure means ExpressRoute. This is called private because the network traffic occurs over your network provider and does not go across the public Internet as it does with both site-to-site and point-to-site connectivity. This capability ensures that applications with privacy requirements can be developed and run on Azure. Also, in most cases, using ExpressRoute gives you increased reliability and speed and lower latency.

With this option, the network connects on your end to one of two scenarios: hardware colocated at an Exchange provider (such as Equinix or Level 3) or an additional site on your MPLS VPN-based WAN through a network server provider. The connection from the Exchange provider or the network service provider connects directly to Azure. A single ExpressRoute circuit can connect to multiple virtual networks in the same Azure geography.

Inbound bandwidth to Azure is always free. ExpressRoute now provides either metered or unmetered billing. If you have workloads in which huge amounts of data are leaving the Azure datacenter, using ExpressRoute with unmetered billing can significantly lower the cost of that data transfer. Depending on the provider you select, the bandwidth can range from 50 Mbps to 10 Gpbs.

This is the best solution for delivering enterprise-grade solutions. It is a good fit for applications or workloads that are mission critical to your company. The consistent network performance provided by ExpressRoute also makes it a good solution if you have SLAs in place with groups internal or external to your organization.

Point-to-site network

In this section, you will see how to set up a point-to-site network and test it, connecting to it from the local machine and then checking the IP address. You can't do this in the Azure portal; we are going to use PowerShell instead. You will need to have PowerShell and the Azure Tools installed.

To set up point-to-site (P2S) connectivity, you first have to set up a virtual network with at least one subnet for the VPN Gateway that must be named GatewaySubnet. You will put this network in a resource group in the West US region. Then, you'll create the certificates required and hook up the client to the network.

Overview of setup process

Here are the steps you're going to follow to configure a point-to-site network to access your network in Azure from your local machine. We're going to use PowerShell. If you use the PowerShell ISE, you can create a script as you go along. You can run the script in sections and make sure each section has no errors before continuing; at the end, you will have a finished and working script that you can reuse.

1. Setup: specs, PowerShell variables, resource group, and so on.
2. Create the virtual network.
3. Request a dynamically assigned public IP address for the gateway.
4. Create the required certificates and install where needed.
5. Create a network gateway.
6. Configure the client, then establish and verify the VPN connection.

Configuring point-to-site VPN

Setup

1. Open the PowerShell ISE as an administrator. Log into your Azure account.

```
> Login-AzureRmAccount
```

2. As you go through these, add them as variables in your PowerShell script. You can reuse the script later and just change the values of the variables as needed.

```
$VNetName = "AzureBookP2SVNet"  #name of the virtual network; this will have 2 address spaces.

$VNetAddressPrefix = "10.254.0.0/16" #address prefix for the VNet

$FrontEndSubnetName = "FrontEndSubnet"  #name of the first subnet

$FrontEndSubnetAddressPrefix = "10.254.10.0/24"  #address prefix of the first subnet

$BackEndSubnetName = "BackEndSubnet"  #name of the second subnet

$BackEndSubnetAddressPrefix = "10.254.20.0/24"  # address prefix of the second subnet

$GatewaySubnetName = "GatewaySubnet" #must use this name in order to work

$GatewaySubnetAddressPrefix = "10.254.200.0/26"  #address prefix for the gateway subnet

$VPNClientAddressPool = "172.16.201.0/24"  # VPN clients that connect to our VNet using the P2S
connection will receive an IP address from this range.

$ResourceGroup = "AzureBookRGP2S"  #resource group we will create and use

$Location = "West US"  # location of the VNet

#This needs to be a DNS server that can resolve the names

#   for the resources you are connecting to. For this demo,

#   we will use a public IP address, but you will want to

#   put your own values here if you use this in a production environment.

$DNS = "8.8.8.8"  # IP address of the DNS server that you want to use for name resolution

$GatewayName = "BookVnetGway"  # name of the gateway

$GatewayIPName = "BookVnetGwayIP"  # name gateway IP

$GatewayIPConfigName = "gatewayipconfig"  # name of the public IP

$P2SrootCertName = "BookP2SRootCert.cer"  #name of root certificate uploaded to Azure
```

3. Create a new resource group. You will put your network in this resource group.

```
New-AzureRmResourceGroup -Name $ResourceGroup -Location $Location
```

4. Create the subnet configurations. Note that some of these lines have continuation characters (backward tick).

```
$FrontEndSubnetConfig = New-AzureRmVirtualNetworkSubnetConfig `

    -Name $FrontEndSubnetName -Address $FrontEndSubnetAddressPrefix

$BackEndSubnetConfig = New-AzureRmVirtualNetworkSubnetConfig `

    -Name $BackEndSubnetName -Address $BackEndSubnetAddressPrefix

$GatewaySubnetConfig = New-AzureRmVirtualNetworkSubnetConfig `

    -Name $GatewaySubnetName -Address $GatewaySubnetAddressPrefix
```

Create the virtual network

5. Create the virtual network.

```
New-AzureRmVirtualNetwork -Name $VNetName -ResourceGroupName $ResourceGroup `
```

```
    -Location $Location -AddressPrefix $VNetAddressPrefix `
    -Subnet $FrontEndSubnetConfig, $BackEndSubnetConfig, $GatewaySubnetConfig `
    -DnsServer $DNS
```

6. Save the values for the VNet you just created in variables to be used later.

   ```
   $vnet = Get-AzureRmVirtualNetwork -Name $VNetName -ResourceGroupName $ResourceGroup
   $subnetConfig = Get-AzureRmVirtualNetworkSubnetConfig -Name $GatewaySubnetName `
         -VirtualNetwork $vnet
   ```

Request a dynamically assigned public IP address for the gateway

7. Request a dynamically assigned public IP address. This is required for the gateway to work properly. You will connect this to the gateway IP configuration in a future step.

   ```
   $pip = New-AzureRmPublicIpAddress -Name $GatewayIpName -ResourceGroupName $ResourceGroup `
         -Location $Location -AllocationMethod Dynamic
   $ipConfig = New-AzureRmVirtualNetworkGatewayIpConfig -Name $GatewayIpConfigName `
         -Subnet $subnetConfig -PublicIpAddress $pip
   ```

Create the required certificates and install where needed

As discussed before, you need to create a self-signed root certificate and use this to create a client certificate for authentication when connecting to the VM on the network from the client. This is because rather than use password authentication, which is fairly weak, point-to-site connectivity uses certificate authentication. Someone without the correct client certificate installed will not be able to connect to the virtual network, even if they somehow obtains the IP address of the network.

To create certificates, you need the *makecert.exe* file. If you have any version of Visual Studio installed, you should have this. Some people can find this under C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Bin. If your machine is 32-bit, it might be under C:\Program Files\Microsoft SDKs\Windows\v7.1A\Bin. If you are running Windows 8.1, you might see it under C:\Program Files (x86)\Windows Kits\8.1\Bin\x64 (or x86). The fastest way to find it is to open a command prompt window, go to the root of the C drive, and search for it by issuing this command:

```
dir/s makecert.exe
```

If you can't find it, install Visual Studio Community or the Windows 10 SDK, both of which include it in the installation.

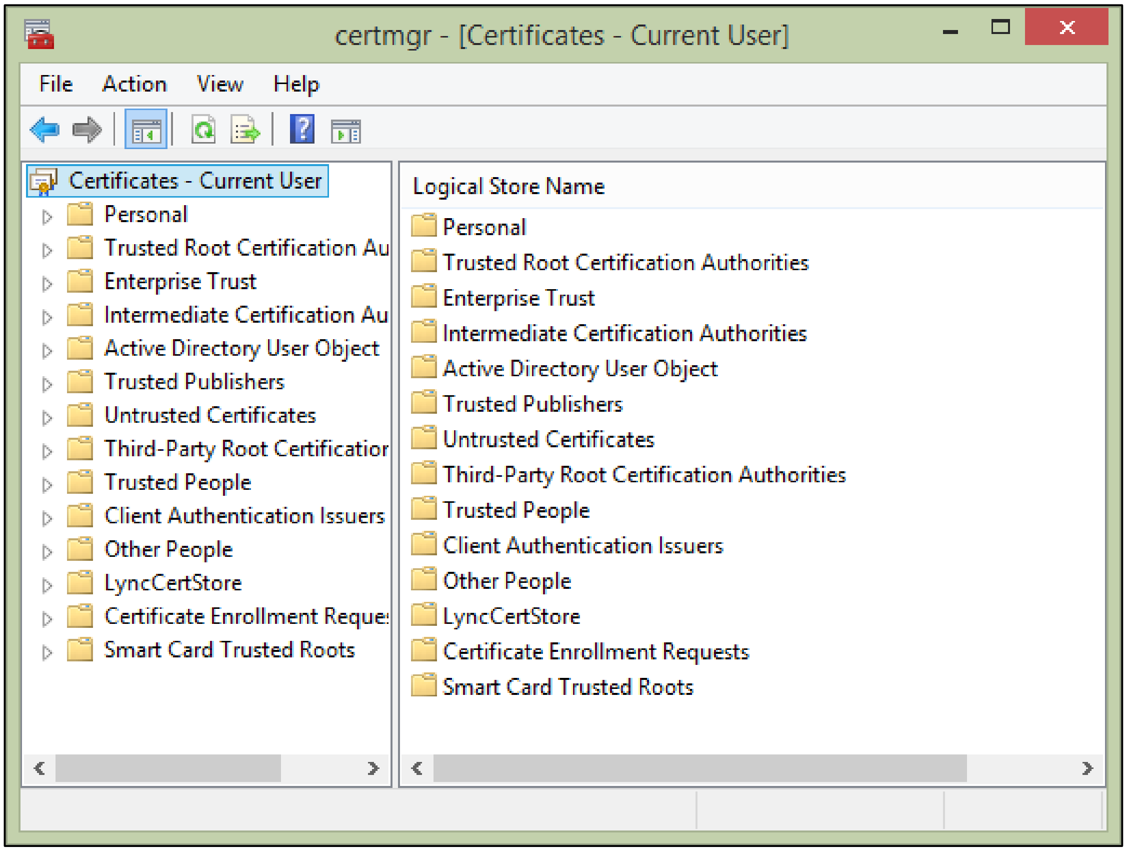
Copy *makecert.exe* to a place to which you can easily navigate in the command window, such as C:\makecert\. Next, open a command window and navigate to that directory (cd C:\makecert). Now you're ready to create your certificates.

8. Create a self-signed root certificate.

   This command will create a root certificate with the name ContosoP2SRoot stored in the current directory as ContosoP2SRoot.cer:

   ```
   makecert -sky exchange -r -n "CN=ContosoP2SRoot" -pe -a sha1 -len 2048 -ss My
   .\ContosoP2SRoot.cer
   ```

9. Now you need to export the root certificate so you can upload it to Azure. To do this, first open the Certificate Manager. To open the Certificate Manager, open the run box (WindowsKey+R), type **certmgr.msc**, and press Enter. You will see the Certificate Manager (Figure 5-12).


certmgr - [Certificates - Current User]
File
Action
View
Help
Certificates - Current User
Personal
Trusted Root Certification Au
Enterprise Trust
Intermediate Certification Au
Active Directory User Object
Trusted Publishers
Untrusted Certificates
Third-Party Root Certificatior
Trusted People
Client Authentication Issuers
Other People
LyncCertStore
Certificate Enrollment Reque:
Smart Card Trusted Roots

Logical Store Name
Personal
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Active Directory User Object
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
Client Authentication Issuers
Other People
LyncCertStore
Certificate Enrollment Requests
Smart Card Trusted Roots

Figure 5-12 The Certificate Manager.

10. Open Personal, then Certificates. You will see a list of certificates on the right side. Find the one both issued *to* ContosoP2SRoot and issued *by* ContosoP2SRoot.

11. Right-click the ContosoP2SRoot certificate and select All Tasks > Export. This opens the Export dialog box. Click Next.

12. Select No, Do Not Export The Private Key, then click Next.

13. Select Base-64 Encoded X.509 (.CER), then click Next.

14. Specify the location where you want the file created and the name of the file. This demo will use *ContosoP2SRoot_NoPk.pfx*. Click Save to accept the name and then click Next on the Certificate Export Wizard page. On the next page, click Finish. It will now export the root certificate to the location specified. Leave the Certificate Manager open; you're going to need it to manage the client certificate in a minute.

15. Now you have to upload the root certificate file to Azure. Using Notepad, edit the .cer file you just created. First, remove the lines that say BEGIN CERTIFICATE and END CERTIFICATE. Now, the challenge is that the file has to be one continuous line, so if your file has line breaks in it, remove the ones on each line so your file is one continuous line of alphanumeric characters. Copy that entire line and put it in your PowerShell script, assigning it to a variable. In the example below, the value is truncated for readability.

```
$MyP2SRootCertPubKeyBase64 = "MIIDUzCCAj+gAwIBAgIQRggGm…ThisIsVeryLong"
```

Now upload the certificate.

```
$p2srootcert = New-AzureRmVpnClientRootCertificate –Name $P2SRootCertName `

    -PublicCertData $MyP2SRootCertPubKeyBase64
```

The next step is to create certificates for the VPN client.

16. Create a self-signed client certificate using the root certificate you just created. The following command will create a client certificate with the name ContosoP2SClient:

    ```
    makecert.exe -n "CN=ContosoP2SClient" -pe -sky exchange -m 96 -ss My -in "ContosoP2SRoot" -is
    my -a sha1
    ```

    This creates and installs the client certificate on the local machine.

    To use that certificate on other client machines, you must export the certificate, copy it to the other machine, and import it.

17. To export the client certificate to a file, go back to the Certificate Manager, find the certificate, and export it. Open Personal, then Certificates. You will see a list of certificates on the right side. Find the one issued *by* ContosoP2SRoot and issued *to* ContosoP2SClient.

18. Right-click the ContosoP2SClient certificate and select All Tasks > Export. This opens the Export dialog box. Click Next.

19. On the next screen, select Yes, Export The Private Key and click Next.

20. Accept the default selection of Personal Information Exchange—PKCS #12 (.PFX), with the Include All Certificates In The Certification Path If Possible check box selected, and click Next.

21. Click the Password check box, enter a password, and enter the confirmation password. Click Next.

22. Fill in a path and file name. This demo will use *ContosoP2SClient.pfx*. Click Save to accept the name and then click Next on the Certificate Export Wizard page. On the next page, click Finish. It will now export the client certificate to the location specified. Close the Certificate Manager and command window.

    All client certificates that chain back to that root certificate will be valid when connecting a client machine to the point-to-site network. Microsoft recommends creating a separate client certificate for each client that is going to connect. If you do this and keep track of them, if you later need to revoke someone's access, you can invalidate that single client certificate.

    The client machine must have the certificate in its certificate store so the handshake with the virtual network will authenticate. If you're going to test it on the same computer you used to create the certificate, then it's already installed and you can skip the next step. Otherwise, you need to follow the instructions in the next step to install the certificate. Note: It won't hurt if you install it and it's already installed.

23. To install the PFX file on a different computer, copy the file (*ContosoP2SClient.pfx* in this demo) to the other computer, then navigate to the file.

    Double-click the file to open it. This will launch the Certificate Import Wizard. Accept all defaults when stepping through the Certificate Import Wizard and enter the password when prompted. This is the password you set when you exported the certificate earlier.

    When prompted to install the certificate, click Yes. You should see a dialog box indicating the import was successful.

    Now we're ready to create the VPN gateway and test out the client.

Create the network gateway

24. Create a network gateway. Note that the type of gateway must be Vpn and the VPN type must be RouteBased. When you run this command, it will take a while, maybe even as long as an hour.

```
New-AzureRmVirtualNetworkGateway -Name $GatewayName -ResourceGroupName $ResourceGroup `
    -Location $Location -IpConfigurations $IpConfig -GatewayType Vpn -VpnType RouteBased `
    -EnableBgp $false -GatewaySku Standard -VpnClientAddressPool $VPNClientAddressPool `
    -VpnClientRootCertificates $p2srootcert
```

Configure the client, then establish and verify the VPN connection

25. Now you're going to download and install the client VPN package.

    Run the following command, which will return a URL link. Copy the link and paste it into a browser to download the package to your computer.

```
Get-AzureRmVpnClientPackage -ResourceGroupName $ResourceGroup -VirtualNetworkGatewayName `
      $GatewayName -ProcessorArchitecture Amd64
```

    The URL will look something like this:

```
https://mdsbrketwprodsn1prod.blob.core.windows.net/cmakexe/REALLY-LONG-ALPHA-NUMERIC-STRING
/amd64/ANOTHER-LONG-ALPHA-NUMERIC-STRING.exe
?sv=2014-02-14&sr=b&sig=bO2SRPG%2Fcs%2BHA7%2FzVnu
q7AwTEo8NCVhCcR59q7RCV9o%3D
&st=2016-05-31T01%3A28%3A43Z
&se=2016-05-31T02%3A28%3A43Z&sp=r&fileExtension=.exe
```

    Because the file came from a location outside your computer, it might be blocked to help protect your computer, especially if you are running Windows 8 or higher. To check this and unblock it if needed, download the file and navigate to it using Windows Explorer. Right-click it and select Properties. If it's blocked, click Unblock and then click OK. Double-click the .exe file and install the VPN client.

    If you choose to run it directly rather than download the file first, when you get the Smart Screen warning message, you can click More Info and then Run Anyway.

26. To establish and verify the VPN connection, navigate to VPN connections on the client machine and find the VPN network you just created. Select the virtual network and then click Connect.

    You are now connected to the virtual network.

27. Open a Windows command window and type **IPCONFIG/ALL**. You will see the connection with the virtual network name, and you can see that the IP address is within the address pool you specified, 172.16.201.0/24.

At this point, you could deploy a VM into the same virtual network, then remove the RDP from the network security group's rules, connect the point-to-site VPN, and RDP into the VM using its internal IP address. This will work even though the RDP port is not opened externally because you have the point-to-site network running.

Databases

A persistent data store is at the heart of many applications. As you migrate existing applications to the Azure cloud or create new applications, you will likely find yourself needing to interact with a database. The Azure platform provides several options from which to choose. You can choose from relational database offerings such as Azure SQL Database, SQL Server running in Azure Virtual Machines, or non-Microsoft databases such as Oracle or MySQL. If a non-relational (or NoSQL) database fits your application needs better, services such as DocumentDB and Azure Table Storage might be a good fit. Furthermore, with Azure Virtual Machines you can install a wide range of database platforms (see Chapter 3, “Azure Virtual Machines,” for more information).

When the time comes to determine a data storage approach for your application, the Azure platform offers a variety of database choices, enabling you to balance reduced friction and management with fully customizable virtual machines.

Azure SQL Database

Azure SQL Database provides a relational database as a service, targeted at online transaction processing (OLTP; that is, data entry and retrieval transactions) workloads. This falls firmly in the platform as a service (PaaS) category of cloud computing. Using SQL Database enables you to give up the physical management responsibilities of a database server but retain the vast majority of logical management and administrative responsibilities. SQL Database provides many attractive features, such as elastic scale, predictable performance, business continuity, near-zero maintenance, and the use of familiar development languages and tools.

It's important to understand that with SQL Database you do not get a physical server that you can manage. Because SQL Database is a database as a service, the underlying physical implementation details are outside your control. There are still logical services when working with SQL Database, but these are not the same as the Microsoft SQL Servers you may be used to working with on-premises. More will be included later in this chapter about SQL Database servers.

SQL Database is available in two different, yet similar, models: elastic database pools and single databases. Elastic database pools enable you to manage multiple databases in a pool, scaling performance up and down as demand changes while maintaining a predictable budget. One of the key features of the elastic pool model is the ability to share performance across many databases in the pool. Alternatively, if you have only a handful of databases, the single database model might be more appropriate. Both models allow you to adjust performance as necessary with no downtime and provide a 99.99 percent service level agreement (SLA).

Both the elastic database model and single database model are available in three service tiers: Basic, Standard, and Premium. Within these tiers, performance is expressed in database throughput units (DTUs). A DTU is a synthetic measure that allows a quick comparison of the relative performance of the various database tiers. Within each tier, there are also performance levels (for Standard, they are S0, S1, S2, and S3). These performance levels provide a way to increase or decrease the DTUs available within the tier. The maximum database size will vary across tiers, ranging from 2 GB to 1 TB. Table 6-1 lists some of the pertinent details for each of the database tiers.

Table 6-1 SQL Database tiers and performance levels

Service tier	Performance level	Maximum database size	DTUs	Target usage scenario
Basic		2 GB	5	Small databases with minimal concurrent operations. Typically development or test usage scenarios.
Standard	S0	250 GB	10	Cloud applications with multiple concurrent transactions.
	S1		20	
	S2		50	
	S3		100	
Premium	P1	500 GB	125	Mission-critical, enterprise-grade applications with high transaction rates and advanced business continuity features.
	P2		250	
	P4		500	
	P6		1,000	
	P11	1 TB	1,750	

Note Web and Business editions for SQL Database were retired in September 2015. For more information, please reference https://azure.microsoft.com/documentation/articles/sql-database-web-business-sunset-faq/.

Determining which service tier to use often depends on monitoring your application performance and then adjusting SQL Database tiers. You can start with a Basic tier, and based on performance indicators, scale up to a Standard or Premium tier if needed. Integrated tools such as Database Advisor (formerly known as Index Advisor), Query Performance Insight, and Query Store can help you gain a better understanding of how to best tune either the application or the database. Adjusting service tiers and performance levels is an online operation, so you can continue to use the database while the operation completes. When doing so, some database connections might be dropped. Be sure to include retry logic in your application to be resilient to such transient errors.

If you are migrating an existing on-premises Microsoft SQL Server database, you can use a third-party tool, the SQL Database DTU Calculator (available at http://dtucalculator.azurewebsites.net/) to estimate the performance level and service tier needed for a SQL Database instance. The tool works for both single and elastic databases, providing recommendations and DTU requirements.

See Also For more information on SQL Database tiers and performance levels, including DTUs and monitoring database performance, please visit https://azure.microsoft.com/documentation/articles/sql-database-service-tiers/.

SQL Database V12

In December 2014, Microsoft released to public preview a new version of SQL Database, dubbed V12. The V12 version of SQL Database reached general availability status in July 2015. One of the primary goals for the V12 version was to improve compatibility with Microsoft SQL Server. For example, previously unavailable features such as Change Tracking, Transparent Data Encryption (TDE), and Full-Text Search are available with SQL Database V12.

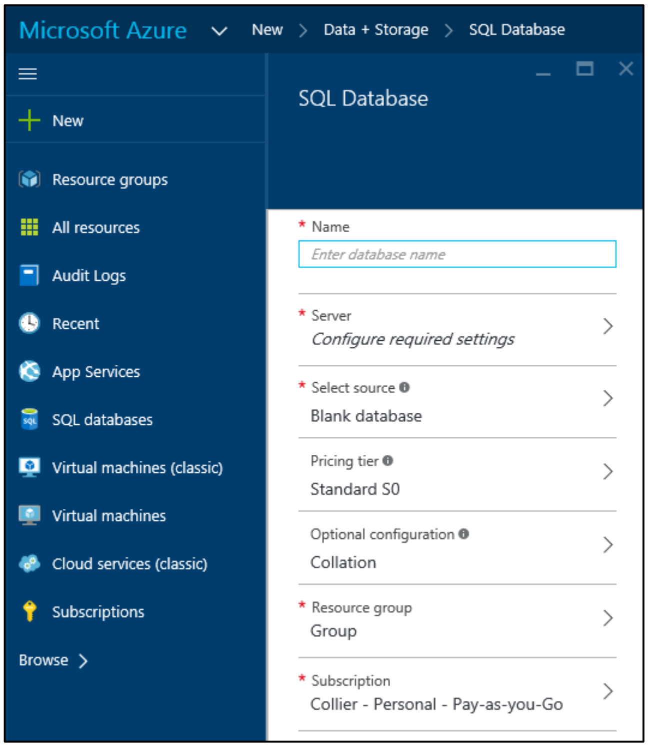
You should plan to use V12 for your SQL Database needs because new features will be added to V12 and not to V11.

For more information on SQL Database V12, including how to determine your version and upgrade information, please see https://azure.microsoft.com/documentation/articles/sql-database-v12-whats-new/.

This chapter will assume the usage of SQL Database V12.

It is important to understand the relationship between a SQL Database server and a database. When you create a SQL Database server, you are creating a logical server that hosts a Tabular Data Stream (TDS) endpoint. TDS is the same communication protocol that's used with SQL Server. The logical server endpoint is identified by a URI, for example, contoso.database.windows.net. Each logical server can contain zero or more SQL database instances.

Creating a new SQL Database is a quick operation. To create a new SQL Database using the Azure portal, click the New button in the navigation pane and then select the Data + Storage category. From there, select the SQL Database option. This will open a new blade that provides an overview of SQL Database, along with links to additional helpful information. Clicking the Create button will open a new blade, as seen in Figure 6-1, allowing you to configure key information for SQL Database.



SQL Database

* Name
Enter database name

* Server
Configure required settings

* Select source
Blank database

Pricing tier
Standard S0

Optional configuration
Collation

* Resource group
Group

* Subscription
Collier - Personal - Pay-as-you-Go

Figure 6-1 Create a new SQL Database instance.

On the SQL Database blade, you can enter several key pieces of information, including the following:

- **Name** Provide the name for the new database.
- **Server** Select an existing SQL Database server or create a new server. When creating a new server, you will be able to provide the server name (for example, contoso.database.windows.net), the administrative login and password, and the Azure region.
- **Source** Select the source for the new database: a blank or empty database, a restore from a previous database backup, or a sample AdventureWorksLT database.
- **Pricing Tier** Select one of the available service tiers (Basic, Standard, or Premium) and associated performance levels.
- **Collation** Set the collation used for rules related to sorting and comparing data.
- **Resource Group** Select an existing resource group or create a new one where the SQL database will reside. Resource groups are helpful for grouping related Azure resources.
- **Subscription** Select the desired Azure subscription.

When finished, click Create. It might take a few minutes for Azure to provision the new SQL database. If you’re creating a new database on an existing server, the new database likely will be ready within a few seconds.

As indicated in Table 6-1, the maximum size for a SQL Database instance is 1 TB at the P11 level. If your data needs exceed the capacity of a single database, you will need to use an alternative strategy to persist the necessary data. One such strategy is to spread the data across multiple databases, a process referred to as database sharding. The ability to create new database shards quickly allows for elastic scale. Application owners can decide how and when to create new database shards to scale out quickly, thereby enabling an application to scale out across multiple databases. For more information on elastically growing and shrinking databases, please read the guidance on SQL Database Elastic

Database features at https://azure.microsoft.com/documentation/articles/sql-database-elastic-scale-introduction/. As of this writing, many of the elastic database features of SQL Database are offered in a preview capacity.

Administration

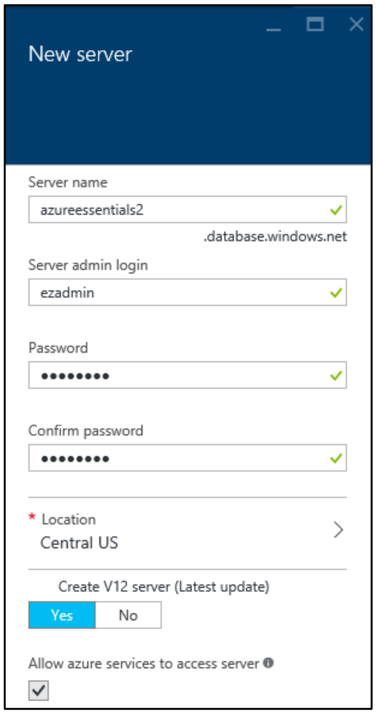
One of the attractive features of SQL Database is the near-zero maintenance it provides. Microsoft handles all patching, server configuration, load balancing, and database platform upgrades automatically. Additionally, SQL Database handles management of system tables and filegroups automatically. You are required to perform common administrative tasks such as managing logical aspects of the database, including logins, tuning indexes, and query optimization.

When using SQL Database, you can use the same tools, programming languages, and frameworks you are accustomed to using with SQL Server. SQL Database and SQL Server are similar in many ways, although not every SQL Server feature is available in SQL Database (see the section "Comparing SQL Database with SQL Server in Azure Virtual Machines" later in this chapter). However, the two do share one very important feature: both use TDS as the client protocol. This allows tools such as SQL Server Management Studio (SSMS) to connect to SQL Database.

Firewall settings

Before you can connect to SQL Database from any tool, including SQL Server Management Studio, you will need to adjust a firewall setting. SQL Database is preconfigured with firewall settings that will explicitly deny access from any IP address, even those originating from within Azure.

When creating a new SQL Database server in the Azure portal, the default is to allow any Azure service (such as your Azure Web App) in any Azure subscription to access the server, as you can see in Figure 6-2.



New server

Server name
azureessentials2 ✓
.database.windows.net

Server admin login
ezadmin ✓

Password
•••••••• ✓

Confirm password
•••••••• ✓

* Location
Central US

Create V12 server (Latest update)
[✓ Yes] No

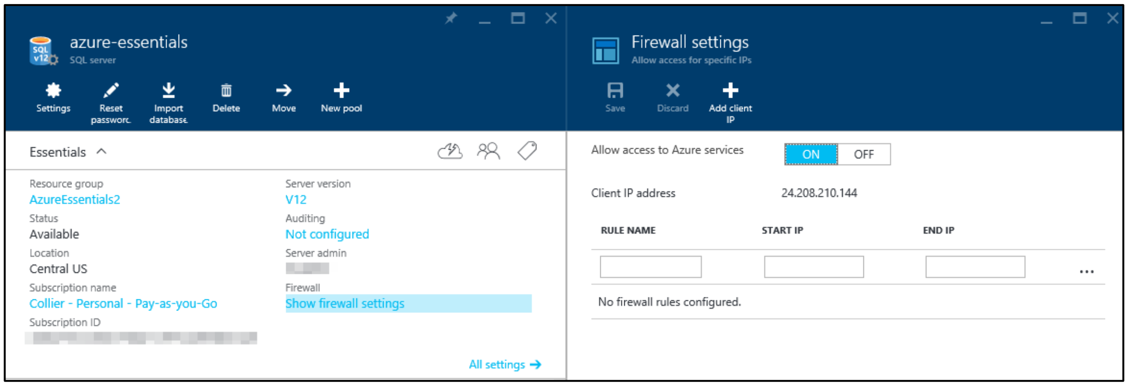
Allow azure services to access server
☑

Figure 6-2 Create a new server.

It's generally not recommended to allow server access (via firewall rules) to all Azure services. Instead, it's recommended to enable access to only specific IP addresses that require access.

> **Note** The act of creating a SQL Database instance via the Azure portal will automatically enable access to the database from other Azure services. This is not the case when creating a SQL Database instance via other means, such as by using PowerShell or an Azure Resource Manager template.

To access the SQL Database server from outside Azure—for example, from SQL Server Management Studio—you will need to modify the server firewall to allow access from the desired IP address (or range). From the SQL Server blade, in the Essentials section, click the Show Firewall Settings link, as seen in Figure 6-3.



Resource group: AzureEssentials2
Status: Available
Location: Central US
Subscription name: Collier - Personal - Pay-as-you-Go
Subscription ID: [redacted]
Server version: V12
Auditing: Not configured
Server admin: [redacted]
Firewall: Show firewall settings

Allow access to Azure services: [✓ ON] OFF

Client IP address: 24.208.210.144

RULE NAME	START IP	END IP

No firewall rules configured.

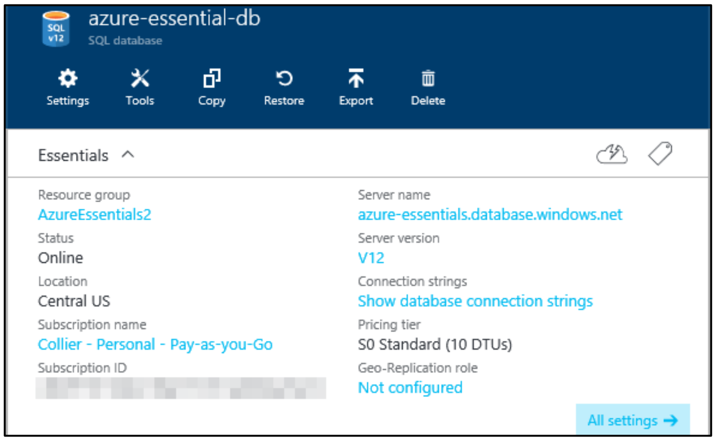
Figure 6-3 SQL Database server firewall settings.

Doing so opens a new blade that allows you to manage various aspects of the SQL Database server firewall settings. On the Firewall Settings blade, you can enable or disable access to the server for Azure services or provide rules to allow access to the server from a specific IP address (including your specific client IP address). If you are hosting other services in Azure (such as Azure Web Apps or Cloud Services) that need access to the SQL Database instance, you will need to set Allow Access To Azure Services to On.

> **Note** It is also possible to set database-level firewall rules in addition to the server-level firewall rules available in the Azure portal. Database-level firewall rules can be set programmatically via T-SQL statements. For more details, see https://azure.microsoft.com/documentation/articles/sql-database-firewall-configure/#creating-database-level-firewall-rules.

Connect using SQL Server Management Studio

Once you have added your IP address to the list of allowed IP addresses, open SQL Server Management Studio. You will need to know the full name of the SQL Database server. You can find the full server name in the Essentials section of the SQL Database blade, along with other key information, as shown in Figure 6-4.



azure-essential-db

SQL database

Resource group: AzureEssentials2

Status: Online

Location: Central US

Subscription name: Collier - Personal - Pay-as-you-Go

Subscription ID: [redacted]

Server name: azure-essentials.database.windows.net

Server version: V12

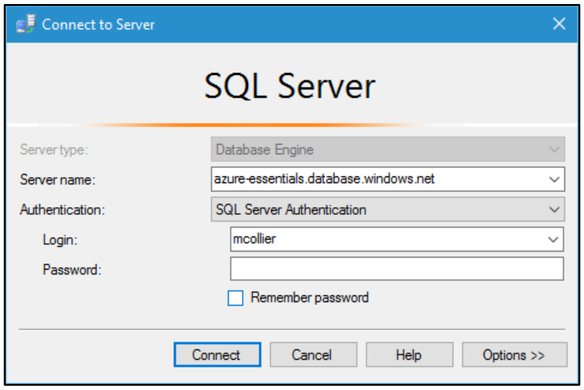
Connection strings: Show database connection strings

Pricing tier: S0 Standard (10 DTUs)

Geo-Replication role: Not configured

Figure 6-4 Database Essentials section.

When the Connect To Server dialog box in SQL Server Management Studio opens, as seen in Figure 6-5, enter the full server name, select SQL Server Authentication, and provide the administrative login and password you set when creating the database.



Connect to Server

SQL Server

Server type: Database Engine

Server name: azure-essentials.database.windows.net

Authentication: SQL Server Authentication

Login: mcollier

Password:

□ Remember password

Figure 6-5 Connect to SQL Database from SQL Server Management Studio.

Note It is also possible to connect to SQL Database by using Azure Active Directory authentication, thus providing a centralized location for management of database users and other Microsoft services. For more information on connecting to SQL Database via Azure Active Directory authentication, please see https://azure.microsoft.com/documentation/articles/sql-database-aad-authentication/.

Billing

Because SQL Database is sold as a service, there is not a separate SQL Server license like you might find with on-premises SQL Server or with SQL Server in an Azure VM. Instead, you are charged per hour based on the highest SQL Database service tier and performance level used during the hour (recall you can change tiers at any time). For example, if you start at 3 PM with an S1 and at 3:20 PM change to an S3, you are charged the S3 rate for the entire hour. The pricing change becomes effective when the change in tier or performance level is completed. For a detailed breakdown of SQL Database pricing, please visit http://azure.microsoft.com/pricing/details/sql-database/.

See Also Please reference the SQL Database FAQ at https://azure.microsoft.com/documentation/articles/sql-database-faq/ for additional helpful information.

In addition to understanding how SQL Database is priced, it's important to understand how many databases you can get. After all, the number of databases you have will directly affect the price you pay. By default, you can have up to six logical SQL Database servers per Azure subscription. Each server can host a maximum of 5,000 databases and 45,000 DTUs. The default limits are soft limits, and they can often be raised by submitting a support ticket with Azure Support.

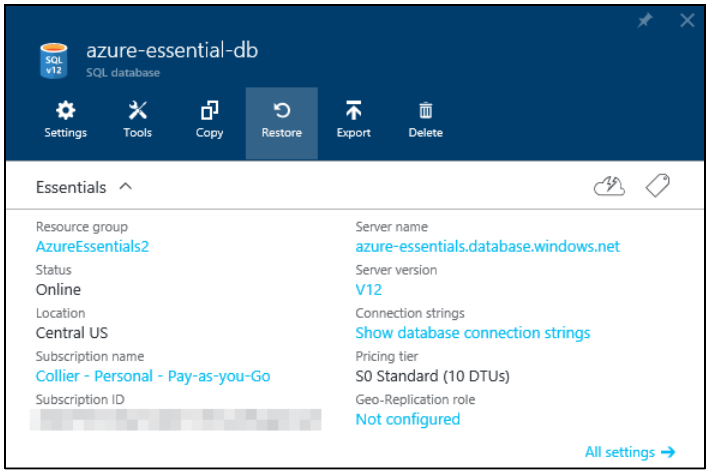
> **Note** When planning capacity for a SQL Database logical server, it is important to note the specific SKU (S0, P1, and so on) being used. Each SKU has associated DTUs. The DTU capacity is the real limit on the maximum number of databases supported.
>
> As performance is highly workload dependent, it is important to test your solution thoroughly—don't assume 5,000 databases on a SQL Database server will all perform equally well.

Business continuity

SQL Database provides several options to address business continuity requirements. One way in which SQL Database provides protection is through infrastructure redundancy. At any time in an Azure datacenter, there could be a hardware failure (such as hard drive, network, or entire servers). SQL Database provides high availability in the case of such hardware failures by keeping copies of the data on physically separate nodes. Three database nodes, or replicas, are always running: one primary replica and two secondary replicas. For write operations, data is written to the primary and one of the secondary replicas before the write transaction is considered complete. In the event of a failure of the primary replica, SQL Database detects the failure and fails over to a secondary replica. If needed, a new replica is then created.

Furthermore, business continuity with respect to databases often includes two categories: database recovery and disaster recovery. Database recovery refers to the ability to mitigate risk and recover from database corruption or an unintentional modification or deletion of data. To assist with database recovery, SQL Database provides a feature called Point-in-Time Restore. Point-in-Time Restore allows you to restore a database to any previous point. The timeframe from which you can restore varies based on the selected SQL Database tier: 7 days for Basic, 14 days for Standard, and 35 days for Premium.

To restore a database to a previous point, first select the desired database in the Azure portal and then click Restore, as shown in Figure 6-6.



Resource group
AzureEssentials2

Status
Online

Location
Central US

Subscription name
Collier - Personal - Pay-as-you-Go

Subscription ID
[redacted]

Server name
azure-essentials.database.windows.net

Server version
V12

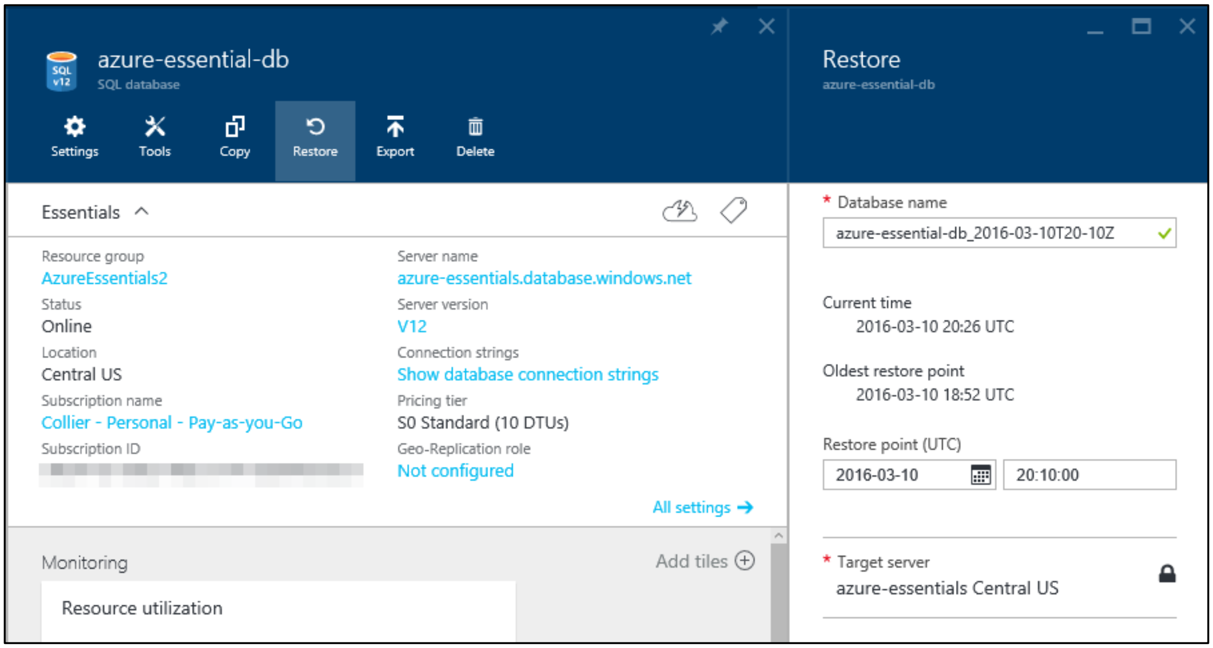
Connection strings
Show database connection strings

Pricing tier
S0 Standard (10 DTUs)

Geo-Replication role
Not configured

Figure 6-6 Option to restore a database.

Clicking the Restore button opens a new blade, as shown in Figure 6-7, that allows you to enter the name for the restored database (or keep the autogenerated default name) and the restore point (date and time, at one-minute intervals).



azure-essential-db
SQL database

Resource group
AzureEssentials2

Status
Online

Location
Central US

Subscription name
Collier - Personal - Pay-as-you-Go

Subscription ID
[redacted]

Server name
azure-essentials.database.windows.net

Server version
V12

Connection strings
Show database connection strings

Pricing tier
S0 Standard (10 DTUs)

Geo-Replication role
Not configured

Monitoring

Resource utilization

Restore
azure-essential-db

* Database name
azure-essential-db_2016-03-10T20-10Z ✓

Current time
2016-03-10 20:26 UTC

Oldest restore point
2016-03-10 18:52 UTC

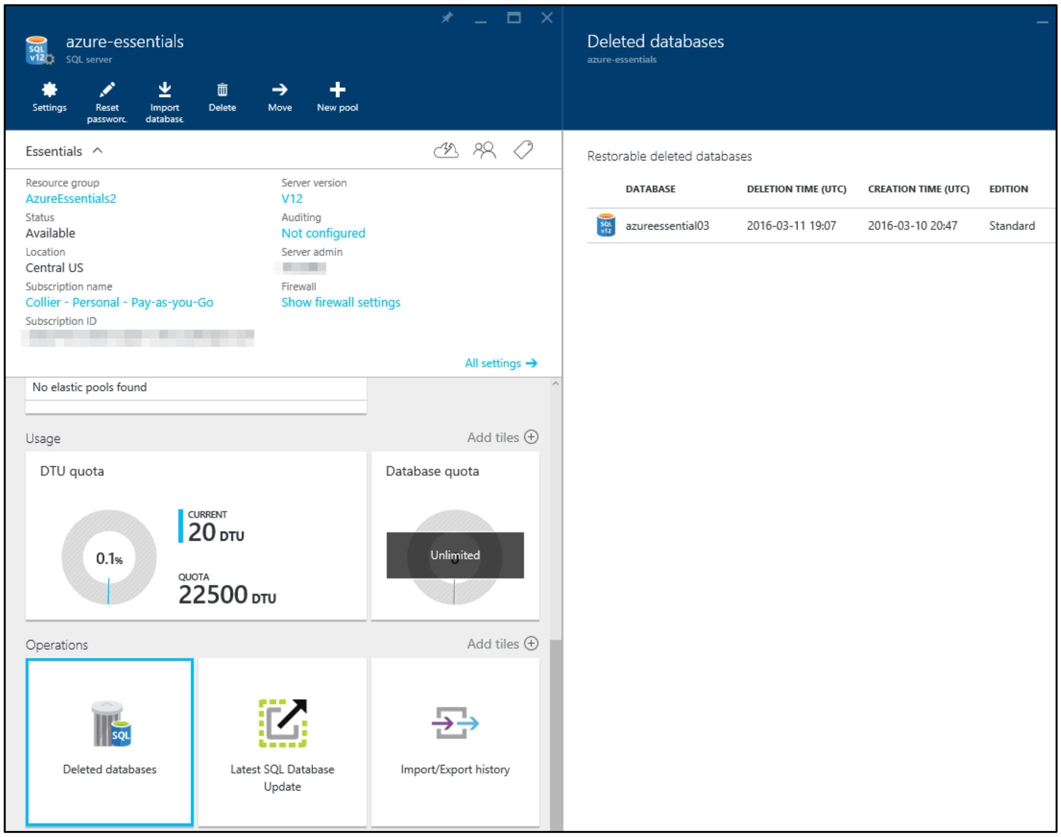
Restore point (UTC)
2016-03-10 | 20:10:00

* Target server
azure-essentials Central US

Figure 6-7 Database restore settings.

The restore operation might a long time to complete. The exact time to restore can be difficult to predict because it depends on several factors, including the size of the database, the restore point in time selected, and the activity log that needs to be replayed to get to the restore point. For some large databases, this process could take several hours.

If you have deleted a database, you can restore the entire database. To do so, first select the SQL Database server that contained the database and then select Deleted Databases in the Operations group of the SQL Server blade. This opens a new Deleted Databases blade, as shown in Figure 6-8.



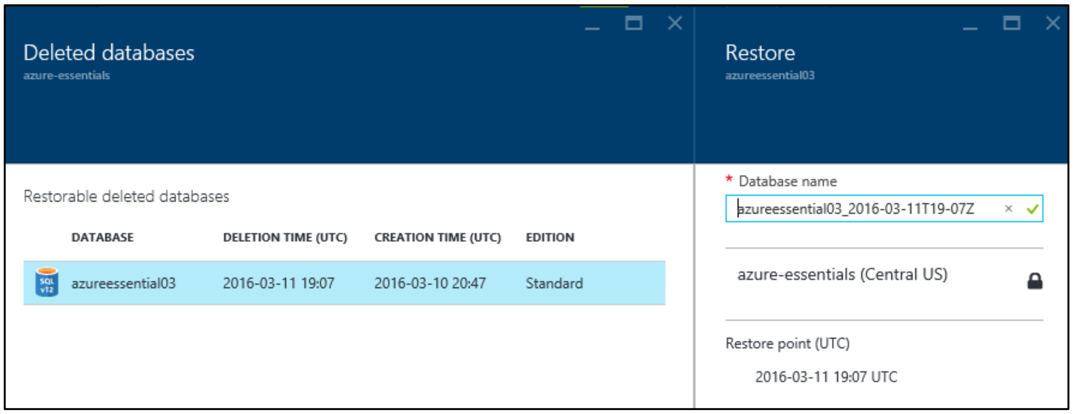
Resource group: AzureEssentials2
Status: Available
Location: Central US
Subscription name: Collier - Personal - Pay-as-you-Go
Subscription ID: [redacted]
Server version: V12
Auditing: Not configured
Server admin: [redacted]
Firewall: Show firewall settings

Restorable deleted databases

DATABASE	DELETION TIME (UTC)	CREATION TIME (UTC)	EDITION
azureessential03	2016-03-11 19:07	2016-03-10 20:47	Standard

Figure 6-8 Restoring a deleted database.

If there are multiple deleted databases, select the database to restore. On the resulting Restore blade, provide a name for the database to be restored, as shown in Figure 6-9. The database can only be restored to the point at which it was deleted. After you click Create, the restore request will be submitted. Just like a point-in-time restore, the process to restore a deleted database could take a long time to complete.



Restorable deleted databases

DATABASE	DELETION TIME (UTC)	CREATION TIME (UTC)	EDITION
azureessential03	2016-03-11 19:07	2016-03-10 20:47	Standard

* Database name: azureessential03_2016-03-11T19-07Z

azure-essentials (Central US)

Restore point (UTC): 2016-03-11 19:07 UTC

Figure 6-9 Settings for restoring a deleted database.

Point-in-Time Restore is helpful when you need to recover a database to a known good point, which is often a result of user error. However, this is only one aspect of business continuity, the other being disaster recovery. Disaster recovery refers to the ability to restore operations to a working state in the event a disaster renders the primary region unrecoverable or unavailable for an extended period of time. SQL Database provides additional features that can be helpful in preparing a disaster recovery plan: Geo-Restore, Standard Geo-Replication, and Active Geo-Replication.

Geo-Restore

The Geo-Restore feature in SQL Database allows you to restore a SQL database from a backup to any SQL Database server in any Azure region. The time to restore will vary based on size of the database, performance level, and number of concurrent restore requests in the target Azure region. Both Point-in-Time Restore (as discussed earlier in this chapter) and Geo-Restore are possible because SQL Database automatically creates backups of every database. Full backups are performed once a week, differential backups once a day, and transaction log backups every five minutes. The backup data is persisted in Azure Blob storage (RA-GRS) in a geo-redundant paired region (for example, East US and West US, North Europe and West Europe), as seen in Figure 6-10.

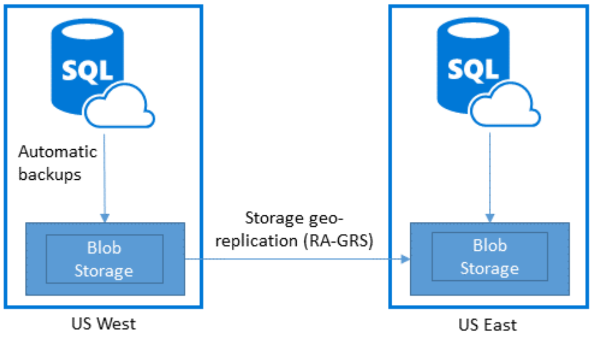
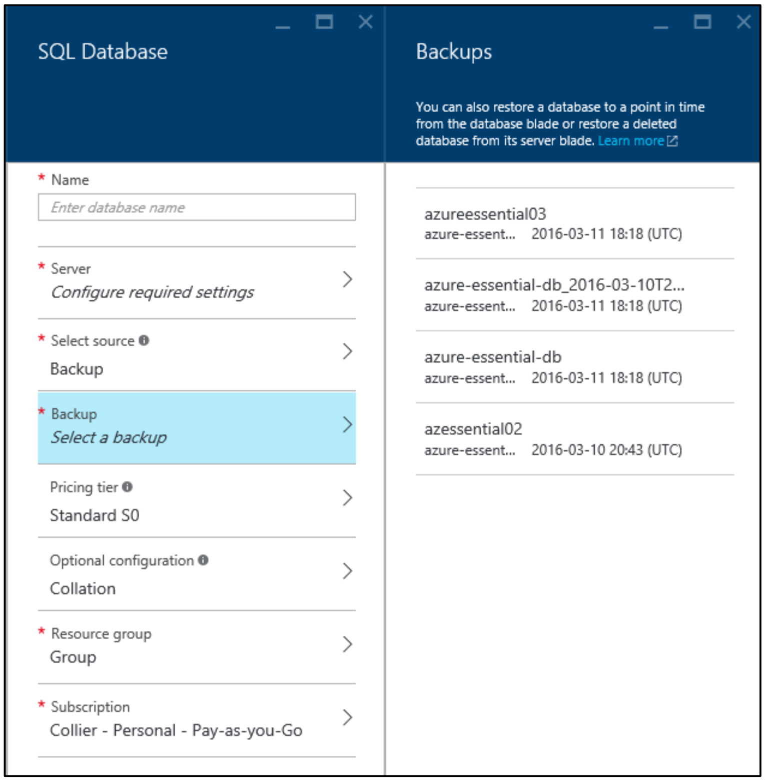


Figure 6-10 SQL Database configured to automatically back up to geo-replicated storage.

To restore from a backup, start by following the same steps you would if creating a new SQL Database. Instead of choosing the source database to be blank or a sample, select the Backup option. Selecting Backup as the source will enable you to then select one of the available backups, as seen in Figure 6-11.



SQL Database

* Name
Enter database name

* Server
Configure required settings

* Select source
Backup

* Backup
Select a backup

Pricing tier
Standard S0

Optional configuration
Collation

* Resource group
Group

* Subscription
Collier - Personal - Pay-as-you-Go

Backups

You can also restore a database to a point in time from the database blade or restore a deleted database from its server blade. Learn more

azureessential03
azure-essent... 2016-03-11 18:18 (UTC)

azure-essential-db_2016-03-10T2...
azure-essent... 2016-03-11 18:18 (UTC)

azure-essential-db
azure-essent... 2016-03-11 18:18 (UTC)

azessential02
azure-essent... 2016-03-10 20:43 (UTC)

Figure 6-11 Select the desired backup as the source for a new SQL Database.

Complete the remaining sections by entering the desired new database name, target server (either select an existing one or create a new one), resource group, and other information.

Standard Geo-Replication

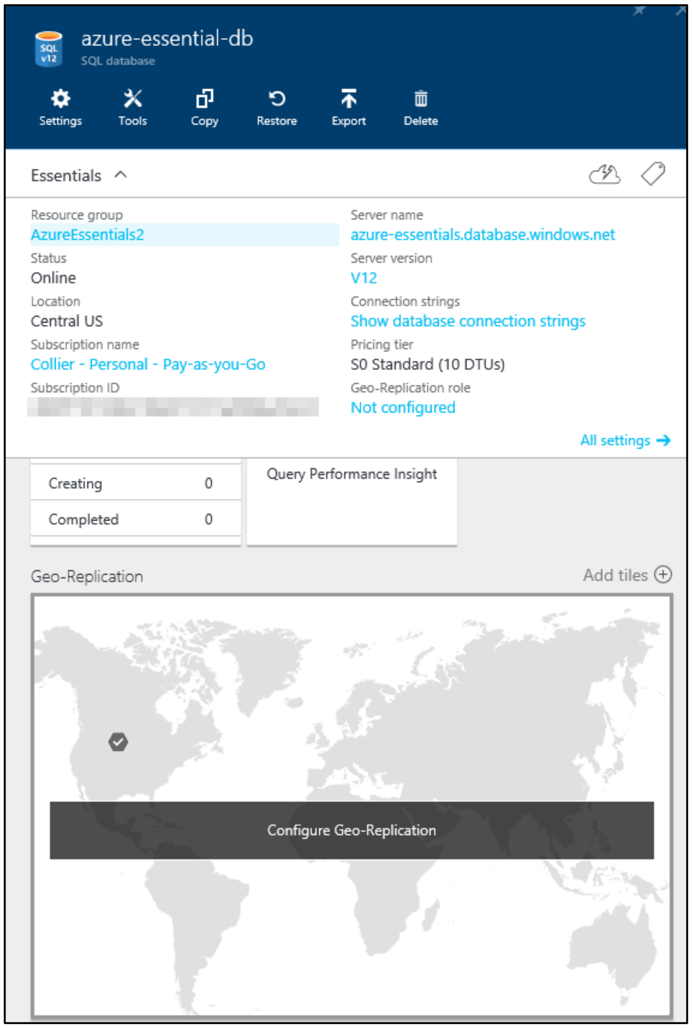
Available for Standard and Premium databases, Standard Geo-Replication enables you to create a single offline secondary database in the paired region of the primary database. The secondary database is unavailable for client connections until the region hosting the primary database fails. The secondary database is charged at 75 percent of the primary.

> **Note** In April 2016, Microsoft announced the retirement of Standard Geo-Replication effective in March 2017.

Active Geo-Replication

Active Geo-Replication, which is available for all database tiers, enables you to create up to four readable secondary databases across multiple Azure regions. It is up to you to determine when to fail over one of the secondary databases (unlike Standard Geo-Replication). Each readable secondary is charged at the same rate as the primary.

To enable Standard or Active Geo-Replication, use the Azure portal and select the desired database. From the Geo Replication group or the link available under Geo-Replication Role in the Essentials section, shown in Figure 6-12, you will be able to see a map displaying any existing secondary database or an option to configure geo-replication if none has been configured.



azure-essential-db
SQL database

Essentials

Resource group
AzureEssentials2

Status
Online

Location
Central US

Subscription name
Collier - Personal - Pay-as-you-Go

Subscription ID
[redacted]

Server name
azure-essentials.database.windows.net

Server version
V12

Connection strings
Show database connection strings

Pricing tier
S0 Standard (10 DTUs)

Geo-Replication role
Not configured

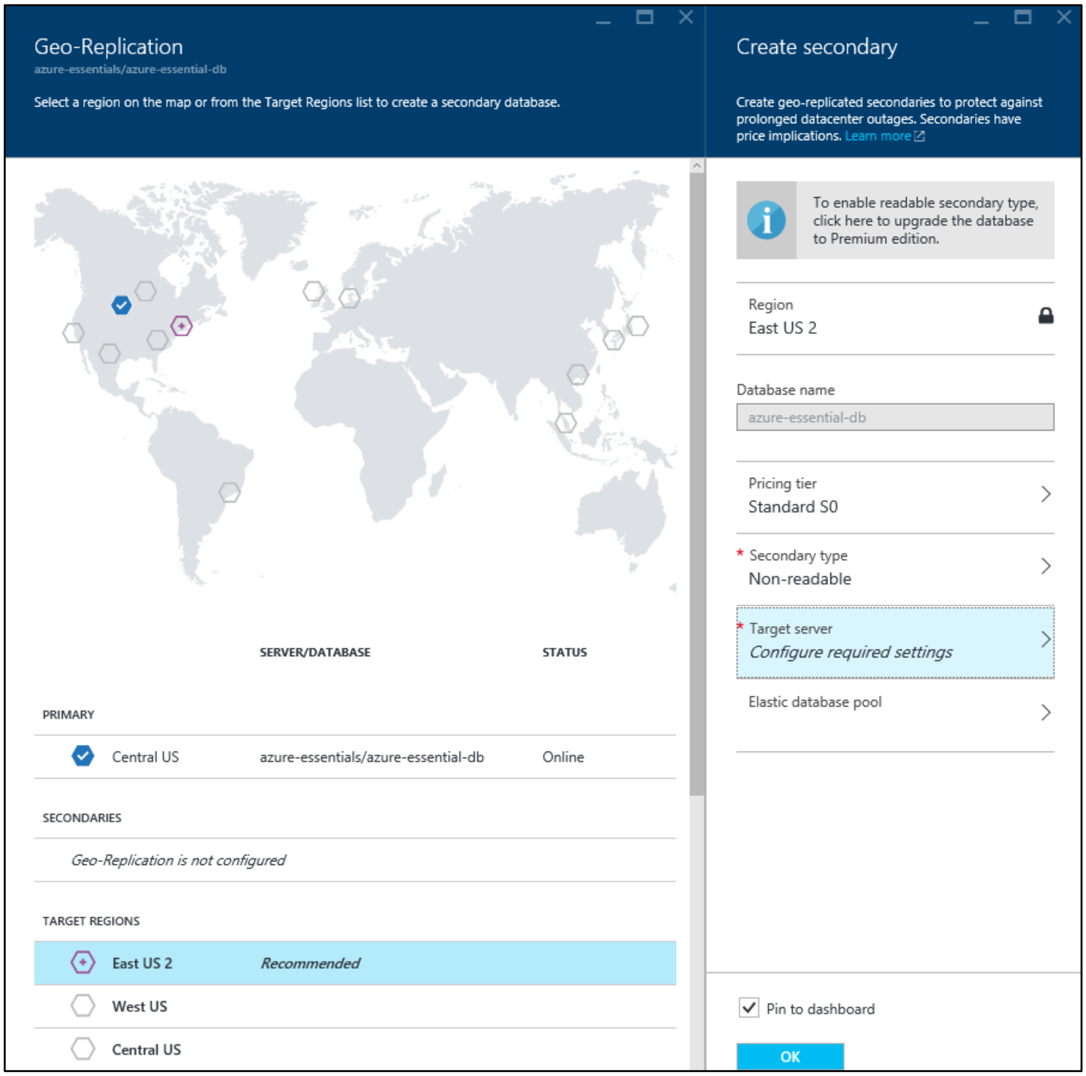
All settings →

Creating	0
Completed	0

Figure 6-12 Configure Geo-Replication.

Select the Geo-Replication part (displaying the map) to open a new Geo-Replication blade. On this blade, you can view all of the potential secondary locations and then select the desired location. Note that for a Standard database, only the paired region will be available. For a Premium database, you can select from any of the available locations.

Click the location in the map, as seen in Figure 6-13, or select from the list of Target Regions displayed on the blade. This will open a new Create Secondary blade to enable you to configure properties related to the secondary database.



Geo-Replication

azure-essentials/azure-essential-db

Select a region on the map or from the Target Regions list to create a secondary database.

		SERVER/DATABASE	STATUS
PRIMARY			
✓	Central US	azure-essentials/azure-essential-db	Online
SECONDARIES			
	Geo-Replication is not configured		
TARGET REGIONS			
⬡	East US 2	*Recommended*	
⬡	West US		
⬡	Central US		

Create secondary

Create geo-replicated secondaries to protect against prolonged datacenter outages. Secondaries have price implications. Learn more

To enable readable secondary type, click here to upgrade the database to Premium edition.

Region
East US 2

Database name
azure-essential-db

Pricing tier
Standard S0

* Secondary type
Non-readable

* Target server
Configure required settings

Elastic database pool

☑ Pin to dashboard

OK

Figure 6-13 Geo-replicated secondary configuration settings.

Alternative configuration options

In addition to using the Azure portal to configure geo-replication, it is possible to configure geo-replication using Transact-SQL (T-SQL), PowerShell, or a REST API. To find out more about these options, go to:

T-SQL https://azure.microsoft.com/documentation/articles/sql-database-geo-replication-transact-sql/

PowerShell https://azure.microsoft.com/documentation/articles/sql-database-geo-replication-powershell/

REST API https://msdn.microsoft.com/library/azure/mt163571.aspx

For Standard Geo-Replication, the Secondary Type will default to Non-Readable only. For Active Geo-Replication, the Secondary Type will default to Readable and you will be able to select from multiple Azure regions for the region.

Design considerations

You'll need to familiarize yourself with several important concepts when determining which SQL Database business continuity feature to use. The business continuity features for SQL Database across the service tiers, including RTO and RPO, are described in the following list and are depicted in Table 6-2.

- **Estimated Recovery Time (ERT)** Indicates the estimated time needed before the database is fully available after a restore or failover request
- **Recovery Time Objective (RTO)** Indicates the maximum downtime before the application is functional after a disaster
- **Recovery Point Objective (RPO)** Indicates the maximum amount of recent data loss (in terms of time) before the application is functional after a disaster

Table 6-2 Business continuity options for SQL Database tiers

Business continuity feature	Basic tier	Standard tier	Premium tier
Point-in-Time Restore	Last 7 days	Last 35 days	Last 35 days
Geo-Restore	ERT < 12 hours RPO < 1 hour	ERT < 12 hours RPO < 1 hour	ERT < 12 hours RPO < 1 hour
Active Geo-Replication	N/A	N/A	ERT < 30 seconds RPO < 5 seconds

> **Note** It is important to understand that the ERT and RPO values provided above are not part of the official service level agreement for SQL Database. They are engineering goals only and provided for guidance and planning.

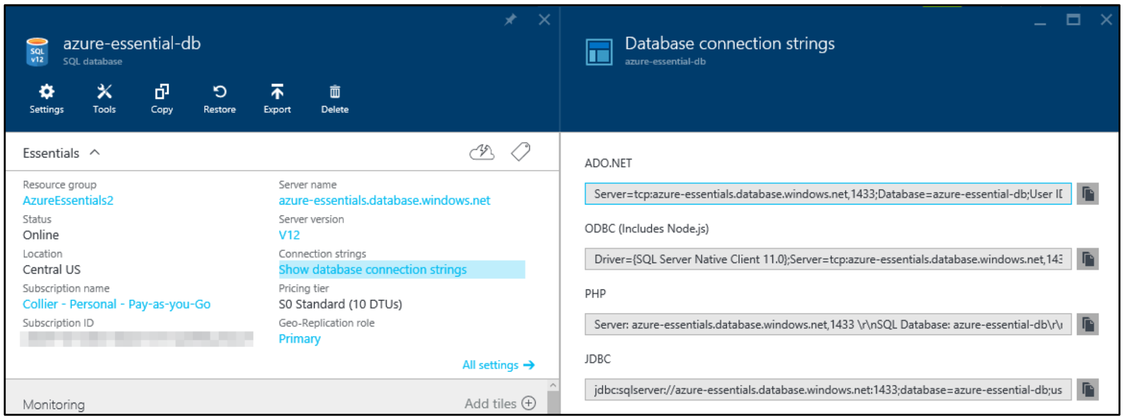
Please see the guidance available at https://azure.microsoft.com/documentation/articles/sql-database-business-continuity-design/ for a detailed breakdown of business continuity design considerations with respect to SQL Database.

Service level agreement

Microsoft provides a 99.99 percent database connectivity service level agreement for Basic, Standard, and Premium tiers. The service level agreement only applies to being able to connect to the database, not to any performance targets with respect to the various tiers.

Applications connecting to SQL Database

When writing applications that need to connect to SQL Database, you can use popular programming languages such as .NET, PHP, Java, and many more. Entity Framework, starting with .NET Framework 3.5 Service Pack 1, is also supported. One of the first things you'll need is the connection string. You can obtain the connection string from the Azure portal by clicking the Show Database Connection Strings link in the Essentials group for the desired SQL Database instance. Doing so opens a new Database Connection Strings blade, as seen in Figure 6-14, displaying the connection string in multiple formats, including ADO.NET, PHP, JDBC, and ODBC (which works for Node.js applications).



Resource group
AzureEssentials2
Status
Online
Location
Central US
Subscription name
Collier - Personal - Pay-as-you-Go
Subscription ID
[redacted]
Server name
azure-essentials.database.windows.net
Server version
V12
Connection strings
Show database connection strings
Pricing tier
S0 Standard (10 DTUs)
Geo-Replication role
Primary

ADO.NET
Server=tcp:azure-essentials.database.windows.net,1433;Database=azure-essential-db;User ID

ODBC (Includes Node.js)
Driver={SQL Server Native Client 11.0};Server=tcp:azure-essentials.database.windows.net,143

PHP
Server: azure-essentials.database.windows.net,1433 \r\nSQL Database: azure-essential-db\r\

JDBC
jdbc:sqlserver://azure-essentials.database.windows.net:1433;database=azure-essential-db;us

Figure 6-14 Database Connection Strings blade.

The connection string for SQL Database is similar to what you would use for SQL Server. For example, for ADO.NET, the connection string format is as follows:

ADO.NET Database connection string

```
Server=tcp:{your_db_server_name_here}.database.windows.net,1433;Database={your_db_name_here};User
ID={your_username_here}@{your_db_server_name_here};Password={your_password_here};Encrypt=True;Tru
stServerCertificate=False;Connection Timeout=30;
```

Note that the connection string sets the *TrustServerCertificate* property to *False* and the *Encrypt* property to *True*. This is to provide additional protection while accessing SQL Database over the Internet. Doing so helps thwart potential man-in the-middle attacks. SQL Database will force the connection to be encrypted regardless of the setting.

When writing code against SQL Database, it is important to defend your code against transient errors. Transient errors are errors that are intermittent and likely will be resolved if the command is retried. These errors are more common with SQL Database than with databases accessed via a local area network (LAN). This is due to the inherently unreliable network that is the Internet and the fact that as a managed service, SQL Database might periodically undergo maintenance activities that could cause connections to drop temporarily. Applications should plan for and defend against transient errors by incorporating retry logic when creating connections or executing commands against SQL Database. Be sure to choose a retry strategy that provides sufficient time for the platform to attempt to recover from whatever caused the initial failure, does not saturate the platform with a rapid succession of retry attempts, and has a maximum number of retry attempts and/or retry duration. For example, an exponential back-off approach with a preset maximum number of attempts is often suitable—be sure to validate for your application scenario.

For .NET applications using Entity Framework, Entity Framework 6 contains connection resiliency/retry logic that will detect transient errors from SQL Database and retry the command. For other .NET applications, Enterprise Library 5 and 6 (see http://msdn.microsoft.com/library/ff648951.aspx) from Microsoft Patterns & Practices contain an application block called the Transient Fault Handling Application Block. This library can also be used to detect transient errors and retry commands.

SQL Database connection and retry guidance

Microsoft no longer maintains Enterprise Library 6, and its source code has been made available to the public. Microsoft does provide example retry logic at https://azure.microsoft.com/documentation/articles/sql-database-develop-csharp-retry-windows/. Either approach is valid.

Please reference the SQL Database best practices and design guidelines available at https://azure.microsoft.com/documentation/articles/sql-database-connect-central-recommendations/ for more detailed information and recommendations on connecting to SQL Database.

Finally, Microsoft's Patterns & Practices team has published detailed retry guidance for a variety of Azure services, including SQL Database. Please reference the guidance at https://azure.microsoft.com/documentation/articles/best-practices-retry-service-specific/ for more information, including code samples.

There is much more to designing a resilient cloud-based application than implementing basic retry logic. It may also be necessary to reconsider how an application processes data. For instance, it is often helpful to implement an eventually consistent solution, potentially using queues and background tasks to complete requests. It may also be worth considering breaking read and write operations across separate databases, potentially using the Active Geo-Replication feature in SQL Database to replicate operations on the write database to the read database.

SQL Server in Azure Virtual Machines

Although SQL Database provides database as a service with enterprise-grade features and virtually no administration, there are still situations in which running your own SQL Server deployment may be necessary. A common reason is the requirement to use features that are not available in SQL Database.

As discussed in Chapter 3, Azure Virtual Machines provides the ability to host and manage your own virtual machines (VMs). What you use the VM for is largely your responsibility, and this includes using it to install, configure, and manage your own full SQL Server VM or cluster of SQL Server VMs.

Billing

When running your own SQL Server deployment on Azure Virtual Machines, you must understand three important cost factors. First is the cost of the Windows VM itself. Recall that Azure VMs are charged on a per-minute usage model. Second is the SQL Server license cost. When using a SQL Server image from the Azure Marketplace, you will pay an additional per-minute SQL Server license cost, which varies according to the version of SQL Server (Web, Standard, or Enterprise) and the target size of the VM. Finally, you'll also pay for the Azure Storage cost. Azure Storage (specifically page blobs) is used as the persistence mechanism for Azure Virtual Machines disks. To summarize, the cost for SQL Server in Azure Virtual Machines can be represented as Total cost = Windows Server cost + SQL Server license cost + Azure Storage cost.

If you have your own SQL Server license, you can use that instead of paying the per-minute charge associated with using a SQL Server license obtained from an Azure Virtual Machines image. In this case, you pay only for the Windows Server license and any related Azure Storage costs. The ability to use your own SQL Server license is a feature of License Mobility through Microsoft's Software Assurance on Azure program. For more information, see http://azure.microsoft.com/pricing/license-mobility/.

Virtual machine configuration

When configuring SQL Server in Azure Virtual Machines, take into consideration the following:

- VM considerations
  - Use a DS2 VM or higher for SQL Server Standard or Web edition.
  - Use a DS3 VM or higher for SQL Server Enterprise edition.
- Storage considerations
  - Use Azure Premium Storage.
  - If using multiple data disks to store database data and log files, disable geo-replication because consistent write order across disks is not guaranteed.
- Disk considerations
  - Use at least two P30 disks: one for log files and one for data files.
  - Don't store database data and log files on the D drive. The D drive is a physical temporary disk and is not persisted to Azure Blob storage. However, if you're using a D-series or G-series VM, you might consider storing the tempdb database on the D drive. D-series and G-series VMs use an SSD drive for the D drive, and thus tempdb performance could be improved.
  - For workloads that exceed the IOPS limit for a single data disk, you may attach multiple data disks (up to the maximum allowed by the VM size) and use disk striping as a way to increase IOPS.

See Also For a comprehensive review of performance best practices for SQL Server in Azure Virtual Machines, please read the MSDN guidance at https://azure.microsoft.com/documentation/articles/virtual-machines-sql-server-performance-best-practices/.

Business continuity

Most of the high availability and disaster recovery (HADR) solutions you might run for on-premises SQL Server deployments are also available when running SQL Server in Azure Virtual Machines. But why do you need to be concerned about HADR for SQL Server in Azure Virtual Machines? As discussed in Chapter 3, Azure provides high-availability features for the VMs, but not necessarily for SQL Server running on the VM. It is possible for the VM to be online but the SQL Server instance to be offline, unhealthy, or both. Additionally, it is possible for the VM to unavailable due to hardware failure or software upgrades. Therefore, a practiced HADR strategy should be considered.

SQL Server in Azure Virtual Machines supports many of the same HADR technologies that are available for on-premises SQL Server deployments: AlwaysOn, database mirroring, log shipping, and backup to and restore from Azure Blob storage (available in SQL Server 2012 and SQL Server 2014). Depending on the technology used, it might be possible to establish a hybrid topology to allow the HADR technology to span between an Azure region and an on-premises datacenter. Some options, such as SQL Server AlwaysOn, allow for a topology that can even span multiple Azure regions.

See Also For more detailed information on how to configure various HADR solutions with SQL Server in Azure Virtual Machines, please refer to the guidance at https://azure.microsoft.com/documentation/articles/virtual-machines-sql-server-high-availability-and-disaster-recovery-solutions/.

Comparing SQL Database with SQL Server in Azure Virtual Machines

The decision to use SQL Database or SQL Server in Azure Virtual Machines can be difficult. On the one hand, SQL Database is ideal for reducing the administrative cost related to provisioning and managing relational databases because many tasks such as upgrades, patching, backups, and business continuity scenarios are handled automatically. On the other hand, SQL Server in Azure Virtual Machines provides the option to migrate or extend existing on-premises SQL Server workloads to Azure. Even though there are additional administrative costs with running SQL Server in Azure Virtual Machines, the ability to maintain fine-grained control over those tasks could be worthwhile for some users and scenarios.

Although SQL Database and SQL Server are similar in many areas, some key differences exist, most notably SQL Server features that are not currently supported in SQL Database, such as the following:

- Windows authentication.
- FILESTREAM data.
- Database mirroring.
- Extended stored procedures.
- SQL Server Agent/Jobs.
- SQL Server Reporting Services (SSRS) and SQL Server Integration Services (SSIS) are not supported. Alternatively, run a SQL Server on-premises or in an Azure VM and connect to a SQL database.
- T-SQL features
  - USE statement is not supported. To change databases, a new connection must be established.
  - Common language runtime (CLR).
  - Cross-database queries using three or four part names.

Note The limitations listed above are only a few of those of which you should be aware when working with SQL Database. For a complete list, please see the guidance on MSDN at https://azure.microsoft.com/documentation/articles/sql-database-general-limitations/. You can also find the related T-SQL statement references at https://azure.microsoft.com/documentation/articles/sql-database-transact-sql-information/.

Additionally, elastic database query provides limited support for querying across databases in SQL Database. Please see https://azure.microsoft.com/blog/querying-remote-databases-in-azure-sql-db/ for additional information.

There are many factors to consider when choosing between SQL Database and SQL Server in Azure Virtual Machines: database size, existing application versus new application, level of administrative control (including hardware infrastructure), business continuity strategy, and hybrid scenarios, just to name a few. SQL Database is often the right solution for cloud-designed applications that are not using unsupported features and for which near-zero administration is a key priority. SQL Server in Azure Virtual Machines is often the right choice for new or existing applications that require a high level of control and customization (that is, full compatibility with SQL Server) and for which there is a desire to no longer maintain on-premises hardware.

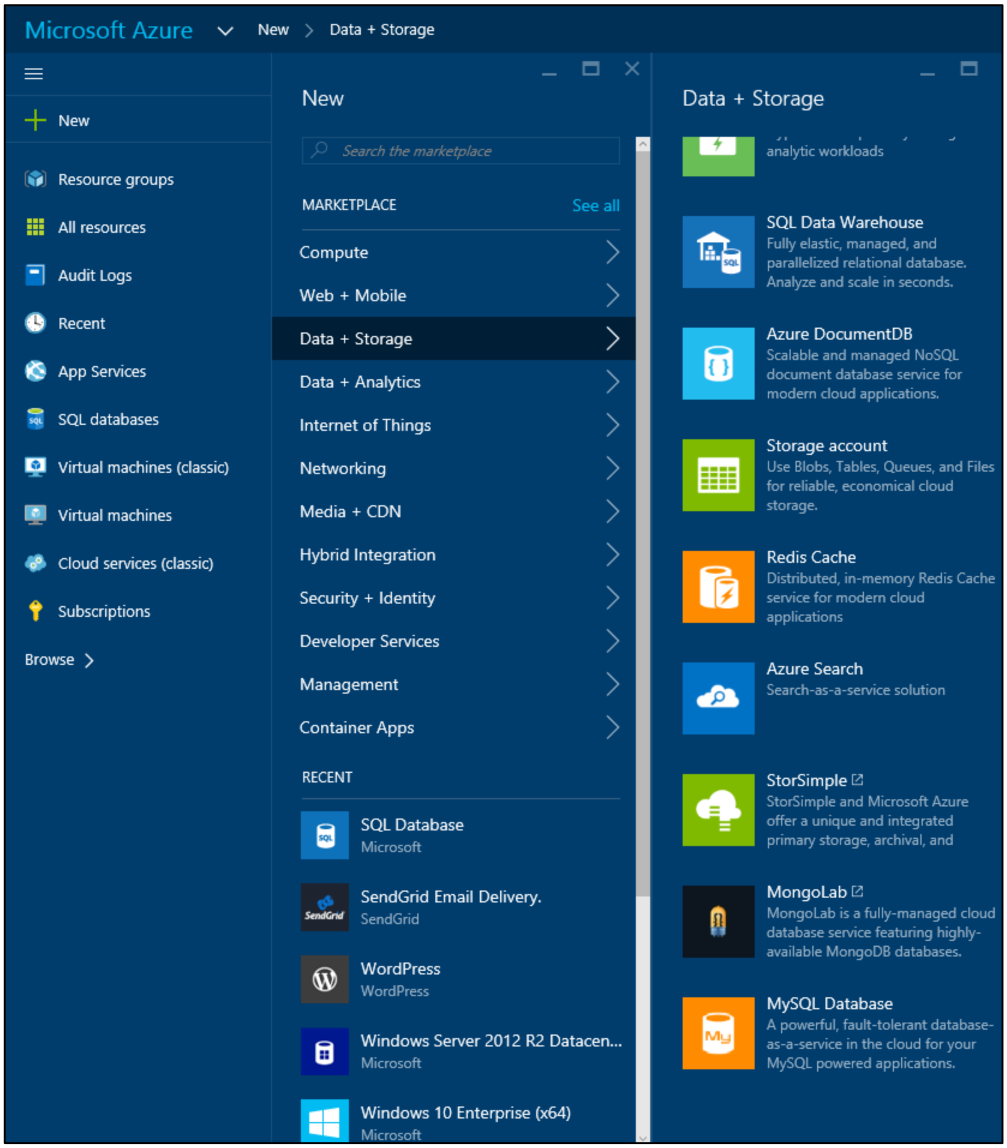
Database alternatives

There are many database options available in the market today. The Microsoft Azure platform makes it easy to run a wide range of popular databases—you don't have to run SQL Database or Microsoft SQL Server. As discussed in Chapter 3, you can run the software of your choosing on an Azure VM, including the database platform you desire. You also have the option to run MySQL as a service via Microsoft's partnership with SuccessBrick's ClearDb offering. If a relational database management system (RDBMS) is not what you're after, using a NoSQL service such as DocumentDB or Azure Table storage is also an option.

MySQL

Another popular relational database is MySQL. Microsoft has collaborated with SuccessBricks to bring SuccessBricks' ClearDb database as a service for MySQL to the Azure platform.

To get started, open the Azure portal and click the green New button in the upper-left corner. From the Data + Storage category, find the MySQL Database feature in the list of available services, as shown in Figure 6-15.



analytic workloads

SQL Data Warehouse
Fully elastic, managed, and parallelized relational database. Analyze and scale in seconds.

Azure DocumentDB
Scalable and managed NoSQL document database service for modern cloud applications.

Storage account
Use Blobs, Tables, Queues, and Files for reliable, economical cloud storage.

Redis Cache
Distributed, in-memory Redis Cache service for modern cloud applications

Azure Search
Search-as-a-service solution

StorSimple
StorSimple and Microsoft Azure offer a unique and integrated primary storage, archival, and

MongoLab
MongoLab is a fully-managed cloud database service featuring highly-available MongoDB databases.

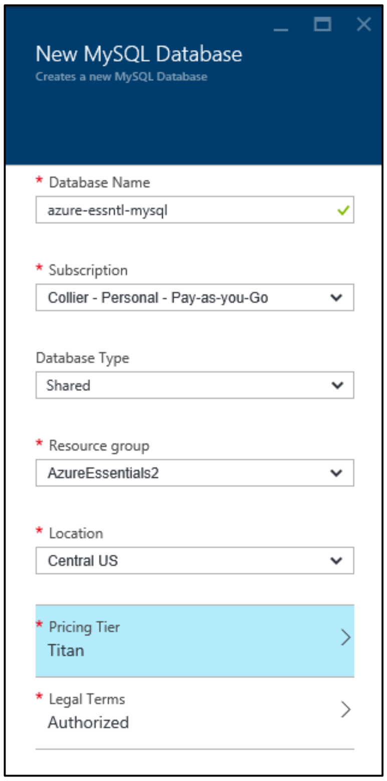
MySQL Database
A powerful, fault-tolerant database-as-a-service in the cloud for your MySQL powered applications.

Figure 6-15 Creating a new MySQL database.

When the New MySQL Database blade opens (Figure 6-16), you'll have the opportunity to provide the necessary details about your new MySQL database, including the following:

- **Database Name** The name for the new database.
- **Subscription** The desired Azure subscription.
- **Database Type** Select Shared for lightweight database workloads and Dedicated for a Premium tier database hosted in single-tenant clusters.
- **Resource Group** Select an existing group or create a new logical group where the new MySQL Database will reside. Resource groups are helpful for grouping related Azure resources.
- **Location** The desired Azure region.
- **Pricing Tier** Select one of the available pricing tiers. These tiers are not related in any way to the SQL Database tiers or performance levels.

- **Legal Terms** Agree to the legal terms, which detail that the service is provided by SuccessBricks and not Microsoft, to continue.



New MySQL Database
Creates a new MySQL Database

* Database Name
azure-essntl-mysql ✓

* Subscription
Collier - Personal - Pay-as-you-Go

Database Type
Shared

* Resource group
AzureEssentials2

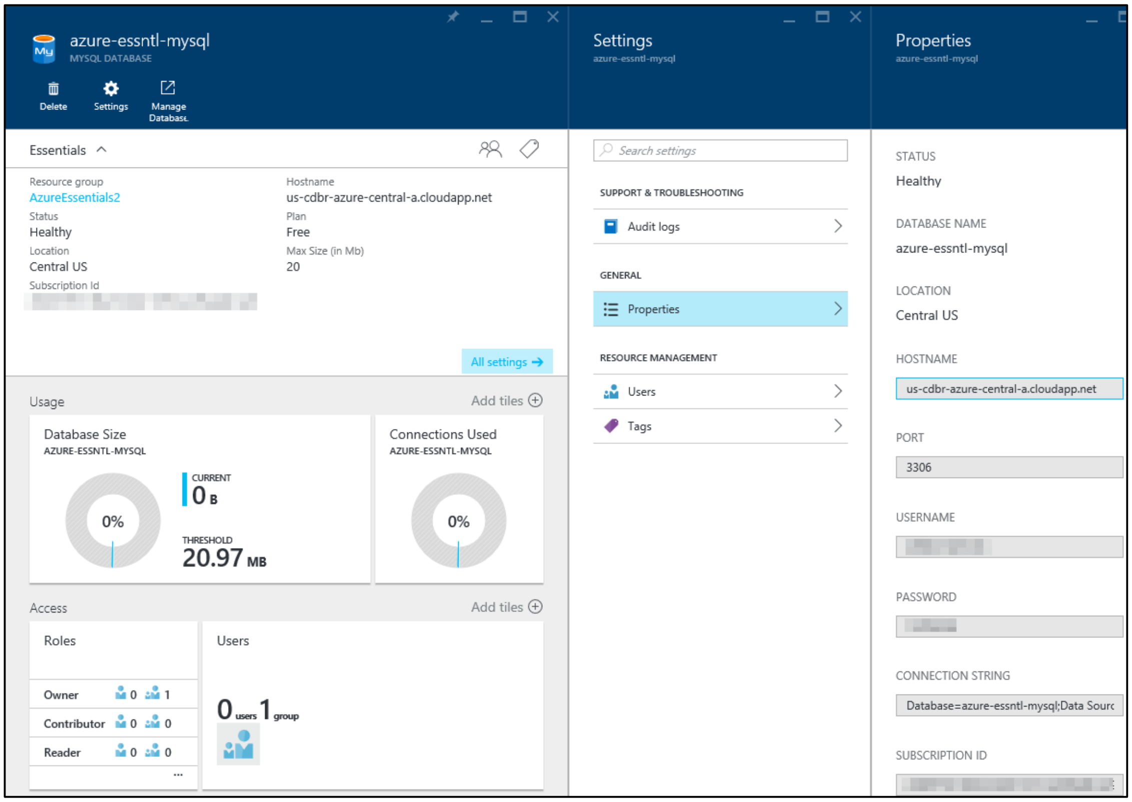
* Location
Central US

* Pricing Tier
Titan

* Legal Terms
Authorized

Figure 6-16 New MySQL Database blade settings.

When finished, click the blue Create button at the bottom of the blade to submit the request to create the new MySQL database. Once the database is created, the database blade will automatically open. Clicking the Properties option on the Settings blade will open a new blade, as seen in Figure 6-17, allowing you to view details about the new MySQL database, such as the full hostname, username, password, connection string, and more.



Resource group
AzureEssentials2
Status
Healthy
Location
Central US
Subscription Id
[redacted]

Hostname
us-cdbr-azure-central-a.cloudapp.net
Plan
Free
Max Size (in Mb)
20

Roles		
Owner	0	1
Contributor	0	0
Reader	0	0

STATUS
Healthy

DATABASE NAME
azure-essntl-mysql

LOCATION
Central US

HOSTNAME
us-cdbr-azure-central-a.cloudapp.net

PORT
3306

USERNAME
[redacted]

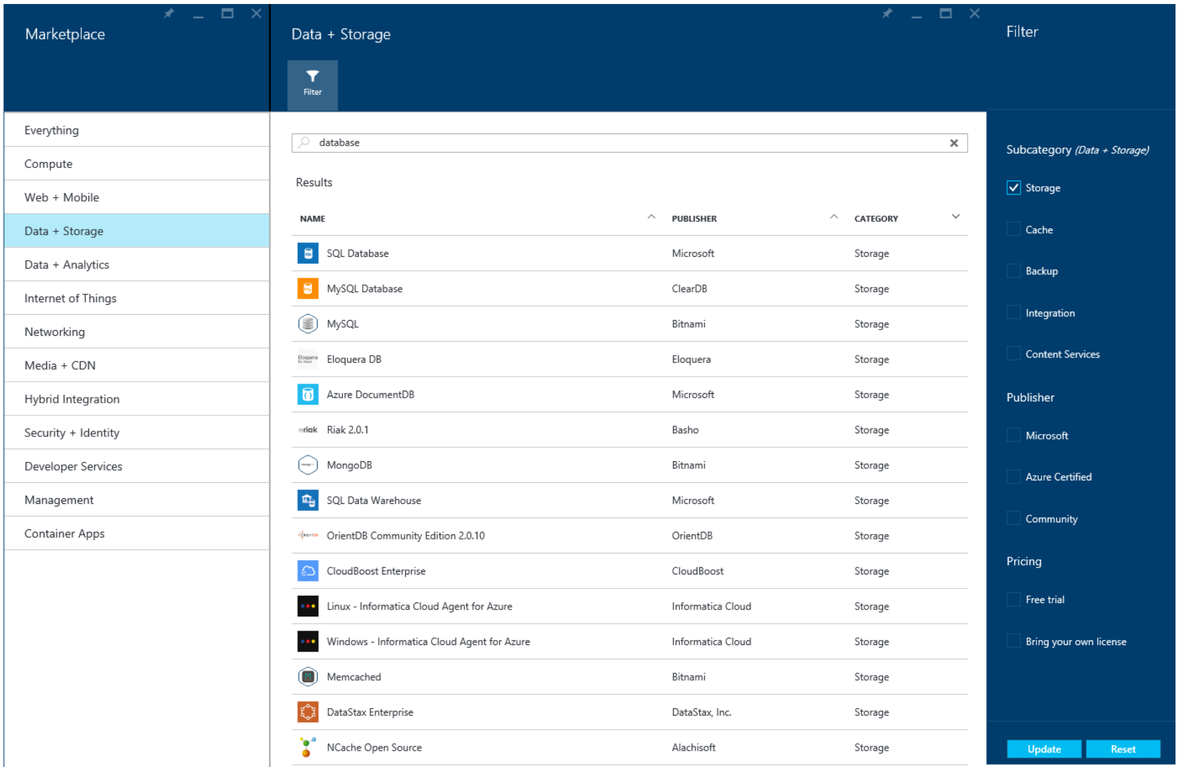
PASSWORD
[redacted]

CONNECTION STRING
Database=azure-essntl-mysql;Data Sourc

SUBSCRIPTION ID
[redacted]

Figure 6-17 MySQL database Properties blade.

Furthermore, the Azure Marketplace provides numerous additional database options, such as MongoDB, Bitnami's MySQL, and DataStax Enterprise, as seen in Figure 6-18. All are available to be hosted using Azure Virtual Machines.



database

Results

NAME	PUBLISHER	CATEGORY
SQL Database	Microsoft	Storage
MySQL Database	ClearDB	Storage
MySQL	Bitnami	Storage
Eloquera DB	Eloquera	Storage
Azure DocumentDB	Microsoft	Storage
Riak 2.0.1	Basho	Storage
MongoDB	Bitnami	Storage
SQL Data Warehouse	Microsoft	Storage
OrientDB Community Edition 2.0.10	OrientDB	Storage
CloudBoost Enterprise	CloudBoost	Storage
Linux - Informatica Cloud Agent for Azure	Informatica Cloud	Storage
Windows - Informatica Cloud Agent for Azure	Informatica Cloud	Storage
Memcached	Bitnami	Storage
DataStax Enterprise	DataStax, Inc.	Storage
NCache Open Source	Alachisoft	Storage

Subcategory *(Data + Storage)*

☑ Storage

□ Cache

□ Backup

□ Integration

□ Content Services

Publisher

□ Microsoft

□ Azure Certified

□ Community

Pricing

□ Free trial

□ Bring your own license

Figure 6-18 Database options available in the Azure Marketplace.

NoSQL options

You don’t always need a relational database management system (RDBMS)–based data storage solution such as SQL Database or SQL Server in Azure Virtual Machines. The proper choice, based on specific application needs, might be a NoSQL solution. Azure provides two NoSQL options: DocumentDB and Azure Table storage.

DocumentDB

DocumentDB is a fully managed, highly scalable, NoSQL document database service available on the Azure platform. DocumentDB is designed to natively support JSON documents, and it automatically indexes all JSON documents added to the database. You use familiar SQL syntax to query the documents.

> **See Also** For more information on DocumentDB, please see http://azure.microsoft.com/services/documentdb/.

Table storage

Azure Table storage is a cost-effective, highly scalable, key/value NoSQL store available on the Azure platform. Table storage is capable of storing up to 500 TB per storage account, with a single subscription supporting 100 storage accounts. Table storage is a semi-structured NoSQL data store that uses two keys, a Partition Key and a Row Key, as the primary composite index for the table.

> **See Also** For more information on Azure Table storage, please see https://azure.microsoft.com/documentation/articles/storage-introduction/#table-storage.

CHAPTER 7

Azure Active Directory

Our identity defines who we are and what we do. Identity is at the heart of many applications and services. Identity tells the story of who uses the application and what actions the user can perform. Without identity, applications often lose the closeness, or personal relationship, so many users find appealing.

The identity story in the Microsoft Azure platform centers on Azure Active Directory (Azure AD). Azure AD provides a cloud-friendly, secure, scalable, modern identity solution that can serve cloud-hosted and on-premises solutions alike.

Overview of Azure Active Directory

Before discussing what Azure AD is, it can be helpful to understand what it is not. First, Azure AD is not a 100 percent replacement for Windows Server Active Directory. For example, you cannot allocate objects such as printers to Azure AD. If you need the full capabilities of Windows Server Active Directory, consider installing and configuring Windows Server Active Directory on Azure Virtual Machines or potentially Azure AD Domain Services.

What is Azure Active Directory?

Azure AD is a robust, secure, multitenant directory service that provides identity and access management in the cloud. In fact, Azure AD is the directory store for many of Microsoft's premium cloud services, such as Microsoft Office 365, Microsoft Dynamics CRM Online, Windows Intune, and, of course, Microsoft Azure. Much like Windows Server Active Directory provides identity and access management for on-premises solutions, Azure AD does so as a service available in Azure. However, instead of you assuming the responsibility of provisioning and configuring the multiple servers necessary for on-premises Active Directory, Microsoft is responsible for managing the entirety of the Azure AD infrastructure (high availability, scalability, disaster recovery, and so on). As a consumer of the Azure AD service (directory as a service), you decide what users and which of their related information should reside in the directory, who can use the information, and what applications have access to the information.

Azure AD should not be considered a full replacement for Windows Server Active Directory. Instead, Azure AD is a complementary service. If you already have Active Directory on-premises, the users and groups can be synchronized to your Azure AD directory by using Azure AD Connect.

> **Note** Azure AD Connect synchronization services is the successor to DirSync, Azure AD Sync, and Forefront Identity Manager with Azure AD Connector.

Azure AD can be associated with an on-premises Active Directory to support single sign-on (SSO). This can be either true SSO using Active Directory Federation Services (AD FS) to federate the on-premises identity to Azure AD or shared sign-on, in which Azure AD Connect is used to sync a password hash between Active Directory and Azure AD. Shared sign-on is simpler to configure at the cost of a small delay in the synchronization of password changes (synchronization is usually completed in a matter of minutes).

By enabling SSO with Azure AD, organizations are able to provide an easy way for employees (or other users) to access a wide range of software as a service (SaaS) applications such as Office365, Salesforce.com, Dropbox, and more. This topic will be discussed in more detail later in this chapter.

Azure AD is a multitenant directory service. Each tenant is a dedicated instance of Azure AD that you own when you sign up for a Microsoft cloud service (Azure, Office 365, and so on). Each tenant directory is isolated from the others in the service and designed to ensure user data is not accessible from other tenants, meaning others cannot access data in your directory unless an administrator grants explicit access.

It is important to note that Azure AD is not just for cloud or Azure-hosted solutions. Azure AD can be used by both cloud (hosted in Azure or elsewhere) and on-premises solutions. Instead of using technologies like Kerberos or Lightweight Directory Access Protocol (LDAP) to access Active Directory (as you would on-premises), Azure AD is accessible via a modern REST API. This allows a wide range of applications—on-premises, cloud, mobile, and so on—to access the rich information available in the Azure AD directory. For developers, this opens up a vast opportunity that previously, with on-premises solutions, either wasn't possible or was difficult to achieve. By leveraging Azure AD and its Graph REST API, developers are able to easily establish SSO for cloud applications and to query and write (create, update, delete) against the directory data.

Azure AD serves as a key component for identity management in the Microsoft cloud. Azure AD include a wide range of capabilities, such as Multi-Factor Authentication, device registration, Role-Based Access Control (RBAC), application usage monitoring, security monitoring and alerting, self-service password management, and much more. All of these features are designed to help organizations provide security for cloud-based applications, including meeting required compliance targets, in an efficient and cost-effective manner. The list below provides a brief description of several important Azure AD features that are beyond the scope of this book.

- **Azure AD B2C (business to consumer)** Azure AD B2C is a solution for enabling consumer-facing web and mobile applications to leverage existing social accounts (Facebook, Microsoft, Google, Amazon, LinkedIn) or custom local accounts. This is essentially the evolution of Azure AD Access Control Service (ACS). For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-b2c-overview/.

- **Azure AD B2B (business to business)** Azure AD B2B is a solution that allows you to enable access to your organization's applications from external business partner identities. Instead of creating (guest) accounts in your organization's directory for business partners, Azure AD B2B allows your business partners to use their own authentication credentials. This enables you to focus on your application and not identity management of external users. For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-b2b-collaboration-overview/.

- **Azure AD Application Proxy** Application Proxy enables users to leverage SSO to securely access on-premises web applications such as SharePoint sites and Outlook Web Access—without the need for building or maintaining a VPN or complicated network infrastructure. Application Proxy is available for the Basic and Premium editions of Azure AD. For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-application-proxy-get-started/.

- **Azure AD Directory Join** Directory Join enables Windows 10 devices to connect with Azure AD, thus allowing users to sign in to Windows using Azure AD accounts. Doing so will enable SSO to Azure AD resources, access to the enterprise Windows Store, device access restrictions using group policy, and more. Directory Join is suitable for devices that cannot domain join. For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-azureadjoin-windows10-devices-overview/.

- **Azure AD Domain Services** Domain Services provide fully managed domain services such as domain join, group policy, LDAP, Kerberos/NTLM, and so on that are compatible with Windows Server Active Directory. This features enables you to use these services without the need to build and manage Azure virtual machines (VMs) running Windows Server Active Directory or maintain a site-to-site VPN connection between Azure and your on-premises directory infrastructure. For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-ds-overview/.

- **Azure AD Device Registration** Azure AD Device Registration is an Azure AD feature that enables mobile devices (such as iOS, Android, and Windows devices) to be registered in Azure AD. The registered device, and thus attributes of the device, can be used to enable conditional access to on-premises or Office 365 applications. For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-conditional-access-device-registration-overview/.

- **Azure AD Cloud App Discovery** Cloud App Discovery enables IT departments to discover cloud applications used in their organization, thus allowing the applications to be brought under IT control to help mitigate risk of potential data leakage or other security threats. Cloud App Discovery finds the applications being used (including various usage metrics), identifies users, and enables offline data analysis. Cloud App Discovery is a feature of Azure AD Premium. For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-cloudappdiscovery-whatis/.

- **Azure AD Connect Health** Azure AD Connect Health enables you to monitor and gain insights into the overall health of the integration between your on-premises Windows Server Active Directory/Active Directory Federation Service and Azure AD (or Office 365). For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-aadconnect-health/.

- **Azure AD Identity Protection** Azure AD Identity Protection is a security service that enables you to gain insights into potential security vulnerabilities affecting users in your organization (more specifically, their identities). For example, Identity Protection can use knowledge of leaked credentials or sign-ins from geographic locations to which it would be impossible to travel (that is, time between sign-ins is less than the time needed to travel to the different locations). Identity

Protection leverages Azure Machine Learning and heuristics to detect potential risks. For more information, please see https://azure.microsoft.com/documentation/articles/active-directory-identityprotection/.

More information on Azure Active Directory

Azure AD is a large topic—encompassing much more than can be included in a single chapter in this book. For more detailed information on Azure AD, please start with the article available at https://azure.microsoft.com/documentation/articles/active-directory-whatis/.

For a deep dive into building web applications that leverage Azure AD, you are encouraged to read the book *Modern Authentication with Azure Active Directory for Web Applications* by Vittorio Bertocci. More information on the book can be found at https://blogs.msdn.microsoft.com/microsoft_press/2016/01/04/new-book-modern-authentication-with-azure-active-directory-for-web-applications/.

Active Directory editions

As of this writing, there are three tiers for Azure AD:

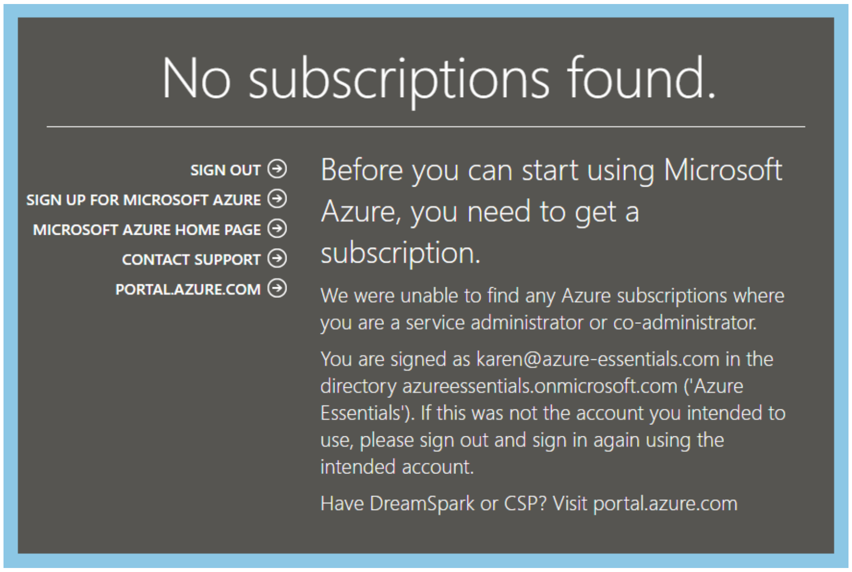
- **Free** Provides the ability to manage users, synchronize with on-premises Active Directory, establish SSO across Azure and Office 365, and access SaaS applications in the Azure AD application gallery.
- **Basic** Provides all the features of the Free tier, plus self-service password resets, group-based application access, customizable branding, Azure AD Application Proxy, and a 99.9 percent availability service level agreement (SLA).
- **Premium** Provides all the features of the Free and Basic tiers, plus self-service group management, advanced security reports and alerts, Multi-Factor Authentication, and licenses for Microsoft Identity Manager.

See Also For more details on the Azure AD tiers, please refer to https://azure.microsoft.com/documentation/articles/active-directory-editions/.

Creating a directory

It is easy to create your own Azure AD directory. In fact, as mentioned earlier, if you are using a Microsoft cloud service such as Office 365, you already have an Azure AD directory. You can associate a new Azure subscription with an existing directory used for authenticating with other Microsoft cloud services. To do so, sign into the Azure classic portal (http://manage.windowsazure.com) using your existing work or school account (formerly known as an organizational account). If there are no existing Azure subscriptions associated with your account, the portal will return a message indicating this, as seen in Figure 7-1. You will need to sign up for an Azure subscription, which you can do at http://aka.ms/AccessAAD. Once you have an Azure subscription, you will be able to use your work or school account to access the Azure subscription.

Note As of this writing, Azure AD is not yet available in the Azure portal at https://portal.azure.com. As a result, all screenshots and directions in this chapter will refer to the Azure classic portal.



No subscriptions found.

Before you can start using Microsoft Azure, you need to get a subscription.

We were unable to find any Azure subscriptions where you are a service administrator or co-administrator.

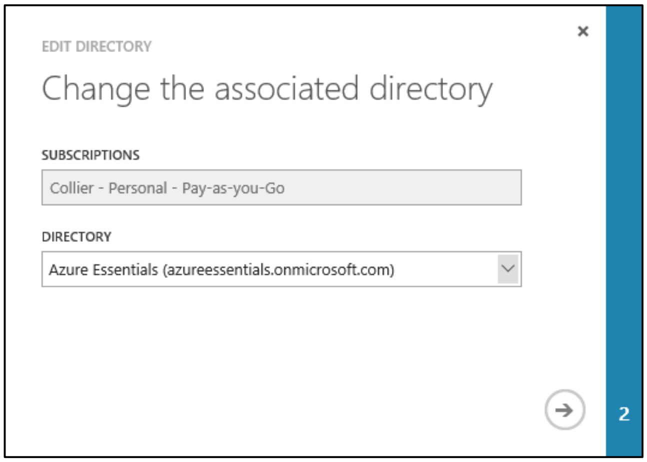
You are signed as karen@azure-essentials.com in the directory azureessentials.onmicrosoft.com ('Azure Essentials'). If this was not the account you intended to use, please sign out and sign in again using the intended account.

Have DreamSpark or CSP? Visit portal.azure.com

Figure 7-1 No subscriptions found.

See Also For more information on managing the directory for your Office 365 subscription in Azure, please see https://azure.microsoft.com/documentation/articles/active-directory-manage-o365-subscription/.

If you don't yet have a subscription to a Microsoft cloud service such as Azure or Office 365, the act of signing up for the service will automatically create an Azure AD directory. You cannot use Azure without a directory. If you signed up for Azure prior to July 7, 2013, you likely were automatically assigned a default directory. You can associate your subscription with a different Azure AD directory by using the Azure classic portal. Proceed to the Settings extension in the left navigation area, select your subscription, and then click Edit Directory on the bottom command bar. The resulting dialog box, shown in Figure 7-2, will allow you to select a different Azure AD directory to be associated with the selected Azure subscription.



EDIT DIRECTORY

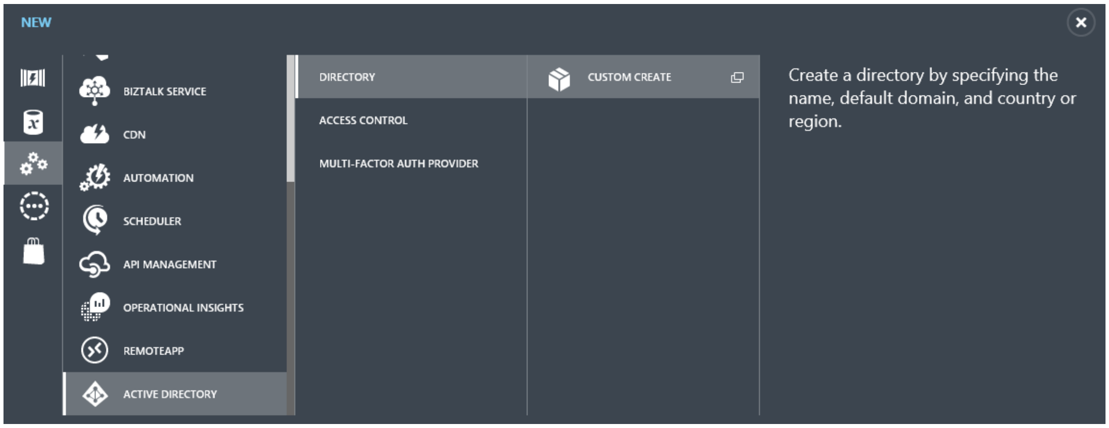
Change the associated directory

SUBSCRIPTIONS: Collier - Personal - Pay-as-you-Go

DIRECTORY: Azure Essentials (azureessentials.onmicrosoft.com)

Figure 7-2 Change the Azure AD directory associated with an Azure subscription.

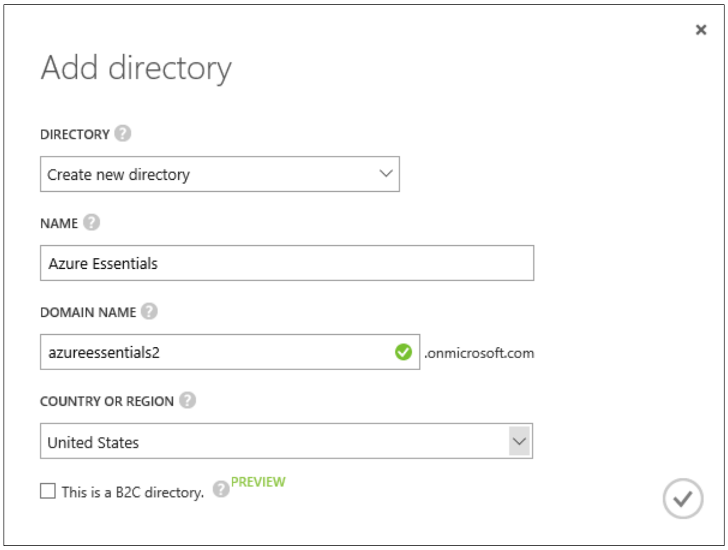
You can add a new Azure AD directory from the Azure classic portal. To do so, select the New button on the command bar and then navigate to App Services, Active Directory, Directory, and finally Custom Create, as seen in Figure 7-3.



Create a directory by specifying the name, default domain, and country or region.

Figure 7-3 Create a new Azure AD directory.

In the resulting Add Directory dialog box, provide a friendly name for the directory, provide a unique domain name, and select your country or region, as shown in Figure 7-4.



Add directory

DIRECTORY

Create new directory

NAME

Azure Essentials

DOMAIN NAME

azureessentials2 .onmicrosoft.com

COUNTRY OR REGION

United States

□ This is a B2C directory. PREVIEW

Figure 7-4 Add a new Azure AD directory.

What’s the relationship between an Azure subscription and Azure AD?

One topic that can cause a lot of confusion is the relationship between an Azure subscription and Azure AD. In the simplest of terms, an Azure subscription is a billing (and in some ways a security and management) entity for Azure services such as Virtual Machines, App Services, and so on. An Azure subscription is free—you pay only for services used.

Azure Active Directory (Azure AD) is a multitenant service that hosts a multitude of directories. An Azure subscription is automatically associated with a directory. In other words, a subscription belongs to a directory.

Please see the Active Directory team blog post at https://blogs.technet.microsoft.com/ad/2016/02/26/azure-ad-mailbag-azure-subscriptions-and-azure-ad-2/ for a really good explanation of how Azure subscriptions and Azure AD are related.

Custom domains

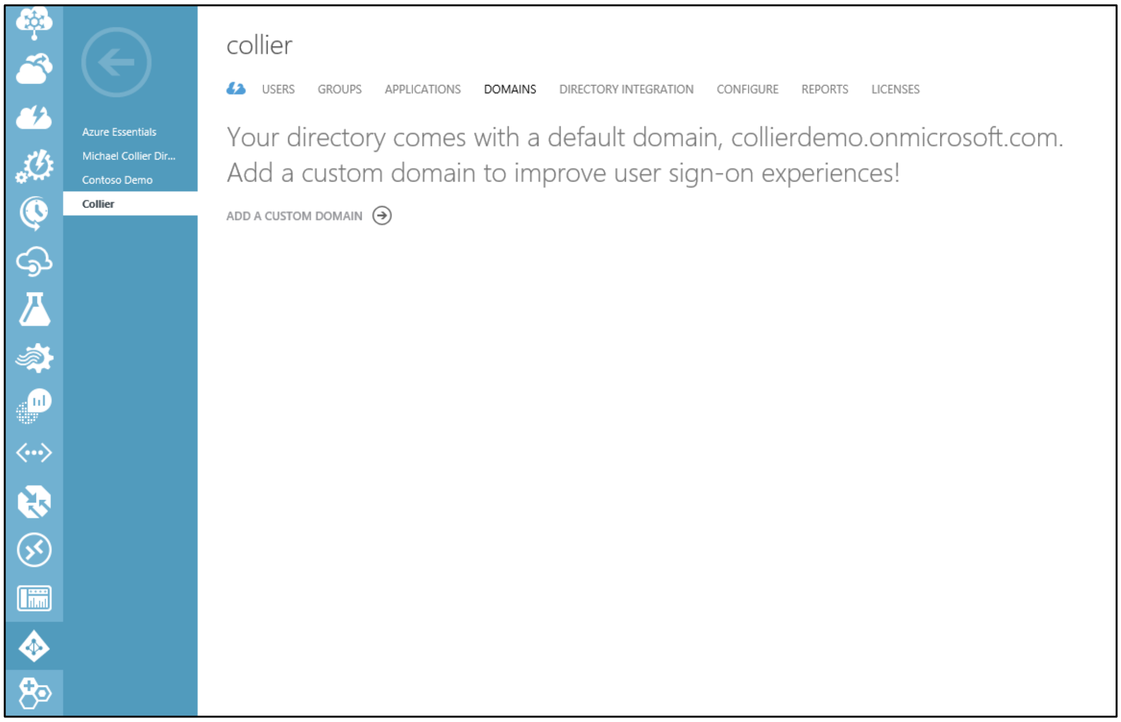
Notice the domain name associated with the directory: [*directory_name*].*onmicrosoft.com*. Every Azure AD directory gets a unique name associated with the *.onmicrosoft.com domain. As a result, every user in the directory would have a name such as *mike@azureessentials.onmicrosoft.com*.

You are not forced to always use the *.onmicrosoft.com domain name. Instead, you can assign a custom domain, one that you own. Once a custom domain is established, users in the directory would reference the custom domain name instead (for example, *mike@azure-essentials.com*).

The process to associate a custom domain with your Azure AD directory is relatively easy. The Azure AD section in the Azure classic portal will walk you through all the steps in an easy-to-follow wizard. There are three basic steps:

1. Get basic information about your domain (or obtain a new domain if needed).
2. Create a DNS record to prove ownership of the domain.
3. Verify the domain.

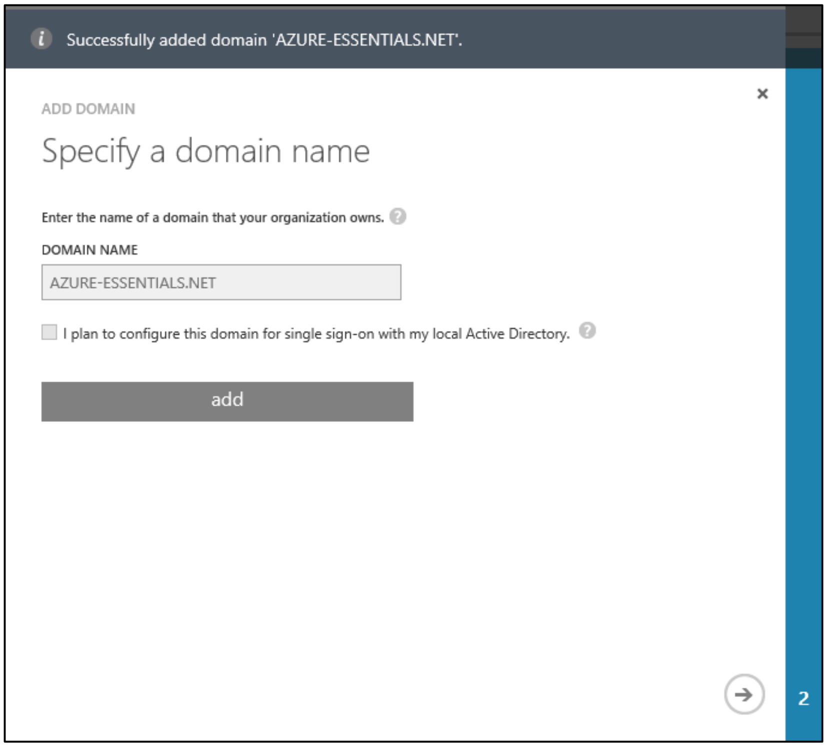
First, select the desired directory in the Azure AD section of the Azure classic portal and then click Domains in the top navigation section. As seen in Figure 7-5, if you don't already have a custom domain, the portal will prompt you to create a custom domain and provide an Add A Custom Domain link to get started.



Your directory comes with a default domain, collierdemo.onmicrosoft.com.
Add a custom domain to improve user sign-on experiences!

Figure 7-5 Add a custom domain.

After you click the Add A Custom Domain link, a dialog box will open, as shown in Figure 7-6, prompting you to add the desired domain name to the list of potential domain names associated with your Azure AD directory.



Successfully added domain 'AZURE-ESSENTIALS.NET'.

ADD DOMAIN

Specify a domain name

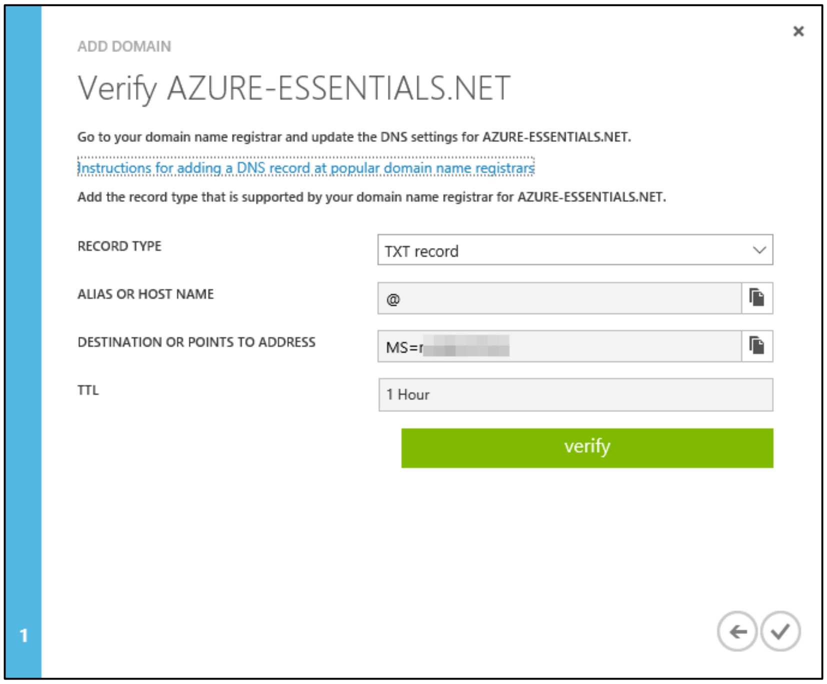
Enter the name of a domain that your organization owns.

DOMAIN NAME: AZURE-ESSENTIALS.NET

□ I plan to configure this domain for single sign-on with my local Active Directory.

Figure 7-6 Specify a domain name.

The next step will be to prove that you own the domain name. Step 2 of the wizard, shown in Figure 7-7, prompts you to add a DNS setting (TXT or MX record) at your domain name registrar. For the purposes of this example, GoDaddy is used; however, the wizard provides a link to an article that provides detailed steps for several popular registrars.



ADD DOMAIN

Verify AZURE-ESSENTIALS.NET

Go to your domain name registrar and update the DNS settings for AZURE-ESSENTIALS.NET.

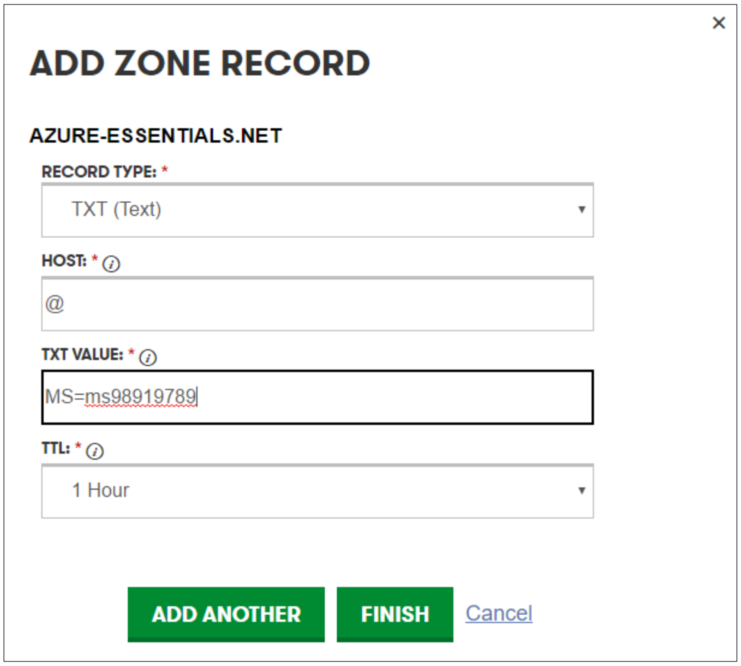
Instructions for adding a DNS record at popular domain name registrars

Add the record type that is supported by your domain name registrar for AZURE-ESSENTIALS.NET.

RECORD TYPE	TXT record
ALIAS OR HOST NAME	@
DESTINATION OR POINTS TO ADDRESS	MS=r[redacted]
TTL	1 Hour

Figure 7-7 Verify domain settings.

For GoDaddy, proceed to the DNS Manager section where you can work with DNS records. There you will need to add a new TXT record (if that was the record type selected in the Azure classic portal). As shown in Figure 7-8, provide the Host, TXT Value, and TTL settings that were also specified in the Azure classic portal wizard.



ADD ZONE RECORD

AZURE-ESSENTIALS.NET

RECORD TYPE: * TXT (Text)

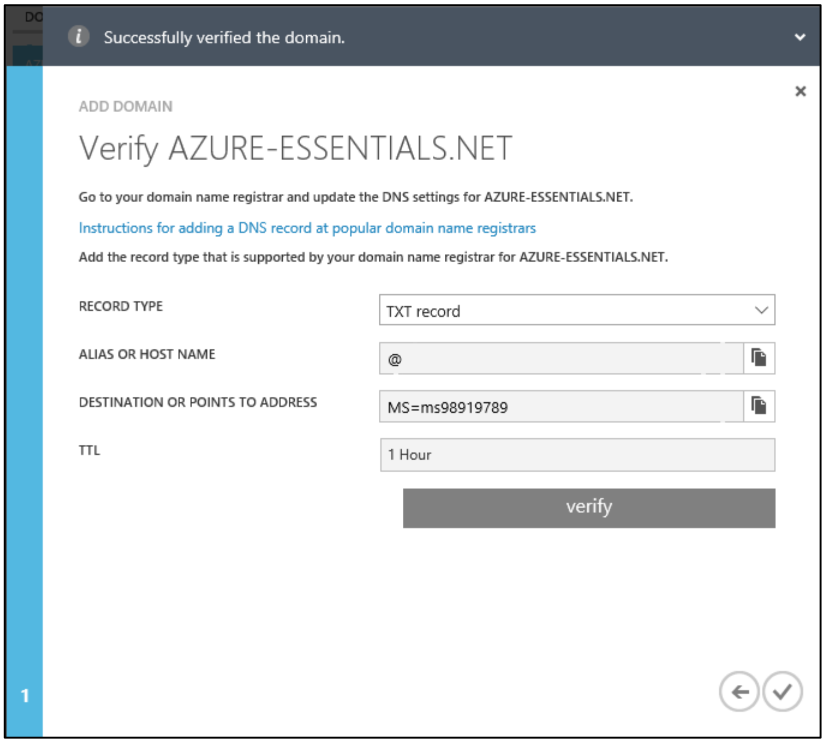
HOST: * @

TXT VALUE: * MS=ms98919789

TTL: * 1 Hour

Figure 7-8 Add Zone Record dialog box.

Click Finish in the GoDaddy editor and then be sure to save the zone file. After adding the TXT record and saving the zone file, you can attempt to verify the domain from the Azure classic portal wizard by clicking Verify. If the verification is successful, you will receive a notification, as seen in Figure 7-9.



Successfully verified the domain.

ADD DOMAIN

Verify AZURE-ESSENTIALS.NET

Go to your domain name registrar and update the DNS settings for AZURE-ESSENTIALS.NET.

Instructions for adding a DNS record at popular domain name registrars

Add the record type that is supported by your domain name registrar for AZURE-ESSENTIALS.NET.

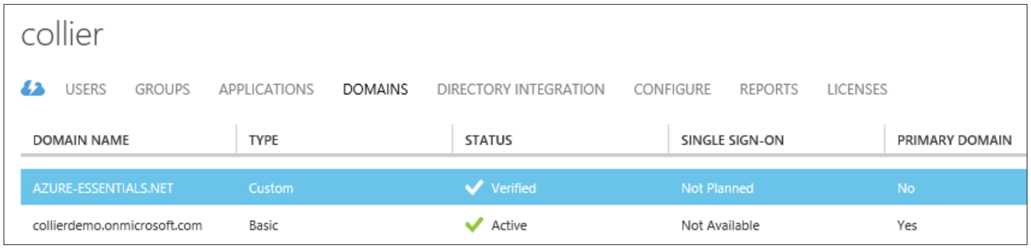
RECORD TYPE	TXT record
ALIAS OR HOST NAME	@
DESTINATION OR POINTS TO ADDRESS	MS=ms98919789
TTL	1 Hour

Figure 7-9 Successfully verifying the domain.

It could take 15 minutes for the DNS changes to take effect. In some cases, it might take up to 72 hours for the DNS records to propagate. If you are unable to verify the domain after 72 hours, you should return to the domain registrar site to verify the DNS record information is correct.

> **Note** This TXT DNS record is used only to verify that you own the domain. You can safely delete it later if you like.

After exiting the wizard, you should notice both the custom domain and the default (or basic) domain listed in the Domains section, as shown in Figure 7-10. You can now change the primary domain for the directory to be the custom domain. If you want to add more custom domains, repeat the steps by first clicking the Add button in the command bar. Adding more custom, verified domains will allow you to change the primary domain associated with the default *.onmicrosoft.com domain.



collier

DOMAIN NAME	TYPE	STATUS	SINGLE SIGN-ON	PRIMARY DOMAIN
AZURE-ESSENTIALS.NET	Custom	✓ Verified	Not Planned	No
collierdemo.onmicrosoft.com	Basic	✓ Active	Not Available	Yes

Figure 7-10 Verified custom domain.

Delete a directory

It is possible to create (or be associated with, including being a member of) a maximum of 20 directories. Creating multiple Azure AD directories can be helpful in development and testing scenarios in which you might not have access to the production subscription. When finished with a directory, you can easily delete that directory. Because deleting a directory is a potentially significant action, a few conditions (enforced by Azure) must be met to delete a directory:

- No users in the directory other than the global administrator
- No applications in the directory
- No subscriptions (for example, Azure, Office 365, and so on) associated with the directory
- No Multi-Factor Authentication providers linked to the directory

To delete the directory, just select the directory and click Delete on the bottom command bar.

Users and groups

After creating the Azure AD directory, a common next step is to add users to the directory. Once users are in the directory, those users will be able to take advantage of Azure AD features such as SSO, access application gallery, and Multi-Factor Authentication (more details of which are provided later in this chapter in the "Multi-Factor Authentication" section).

Add users

Users in Azure AD can be one of four types:

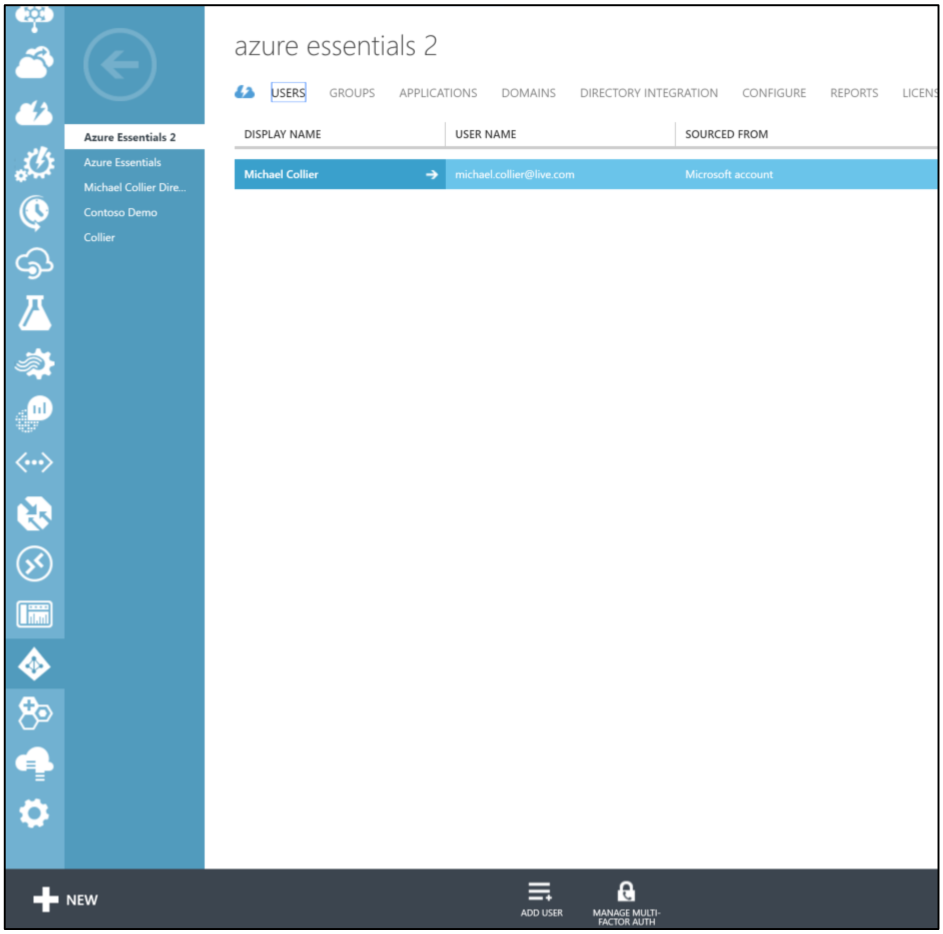
- **A user in your organization** The user is created and managed in your directory.
- **A user with a Microsoft account (for example, hotmail.com or outlook.com)** This is often done as a way to collaborate on Azure resources by granting the user coadministrative rights to the Azure subscription.
- **A user in another Azure AD directory** The user is sourced from another Azure AD directory.
- **A user in a partner company** The user is from a separate organization and is invited and authorized to use your directory. This is a feature of Azure AD B2B collaboration.

See Also For more information on Azure AD B2B capabilities, please see https://azure.microsoft.com/documentation/articles/active-directory-b2b-what-is-azure-ad-b2b/.

To add new users to the directory, there a few different approaches you can take:

- Users can be in the directory as a result of the directory originating from a Microsoft cloud service such as Office 365.
- Users can be synchronized from an on-premises Windows Server Active Directory instance by using Azure AD Connect sync.
- Users can be added programmatically via PowerShell or the Azure AD Graph API.
- Users can be added manually via the Azure classic portal.
- Users can be added by uploading a CSV file for partner organizations.

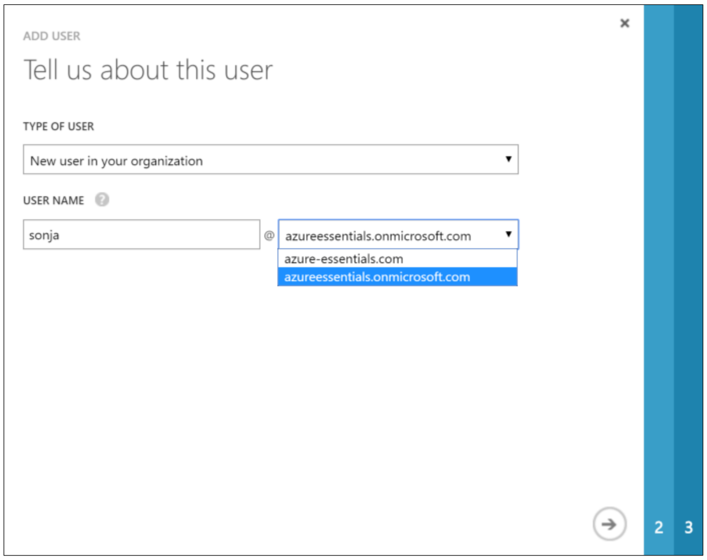
When you first create a new Azure AD directory and you are using an Azure subscription associated with a Microsoft account, there will already be one user account in the directory: yours. You can then add users to the directory by going to the Users section in the desired directory and clicking the Add User button on the bottom command bar, as shown in Figure 7-11.



DISPLAY NAME	USER NAME	SOURCED FROM
Michael Collier	michael.collier@live.com	Microsoft account

Figure 7-11 Listing of users in the Azure AD directory.

Clicking the Add User button opens a new dialog box that allows you to provide details for the new user. As seen in Figure 7-12, if the user is in your organization, you can select the domain name suffix for that user, either the *.onmicrosoft.com or a custom domain (if configured).



TYPE OF USER

New user in your organization

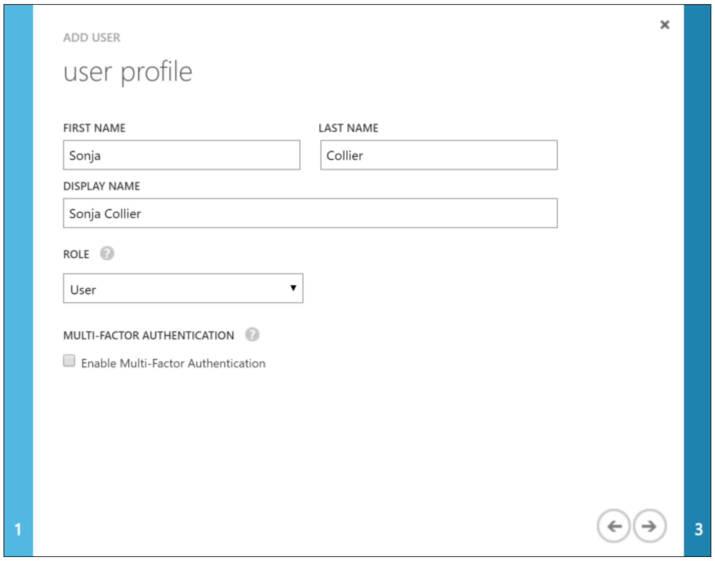
USER NAME

sonja @ azureessentials.onmicrosoft.com

azure-essentials.com
azureessentials.onmicrosoft.com

Figure 7-12 Add a user in your organization.

Figure 7-13 shows the next step, in which you will be able to provide more information about the user, such as the user's first and last name, display name, and potentially an administrative role (within Azure AD only—not related to RBAC features in the Azure portal) for the user. You can also enable Multi-Factor Authentication for the user.



ADD USER

user profile

FIRST NAME: Sonja

LAST NAME: Collier

DISPLAY NAME: Sonja Collier

ROLE: User

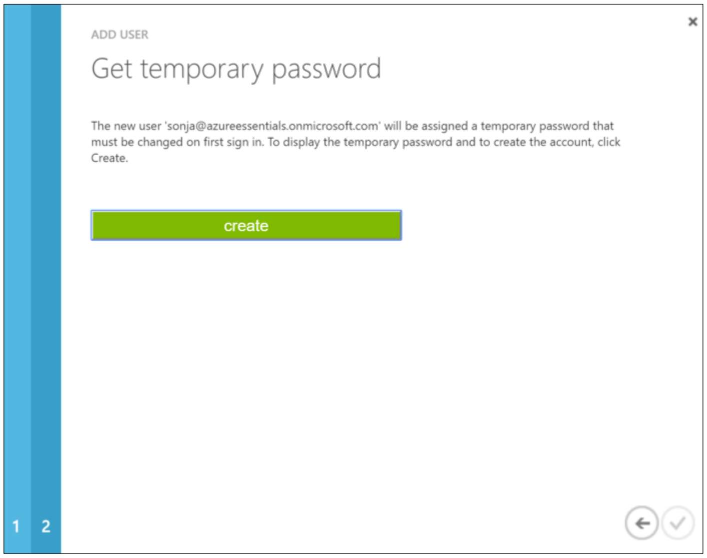
MULTI-FACTOR AUTHENTICATION

□ Enable Multi-Factor Authentication

Figure 7-13 New user profile.

See Also For more details on the administrative roles, please reference https://azure.microsoft.com/documentation/articles/active-directory-assign-admin-roles/.

The final step in creating a new user in your organization is to get a temporary password for the user. Clicking the Create button, as shown in Figure 7-14, creates and assigns a temporary password for the newly created user.



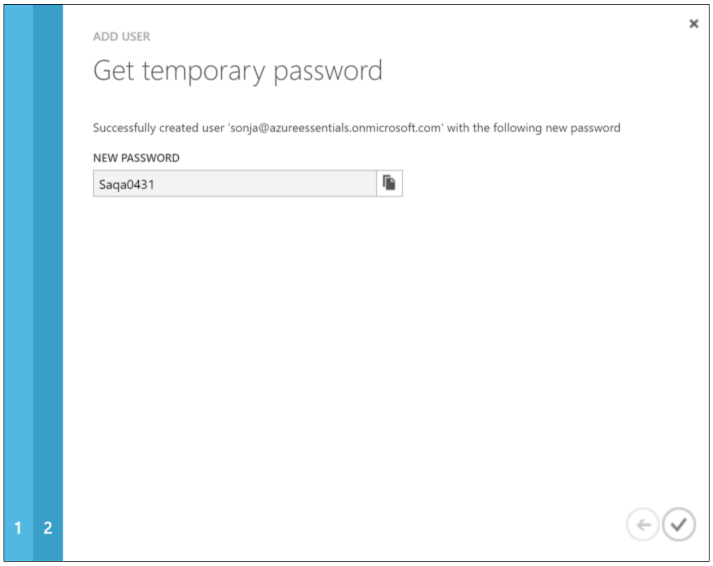
ADD USER

Get temporary password

The new user 'sonja@azureessentials.onmicrosoft.com' will be assigned a temporary password that must be changed on first sign in. To display the temporary password and to create the account, click Create.

Figure 7-14 Get temporary password.

After creating the temporary password, the password is displayed, as shown in Figure 7-15, and you have the opportunity to copy it to someplace safe.


ADD USER

Get temporary password

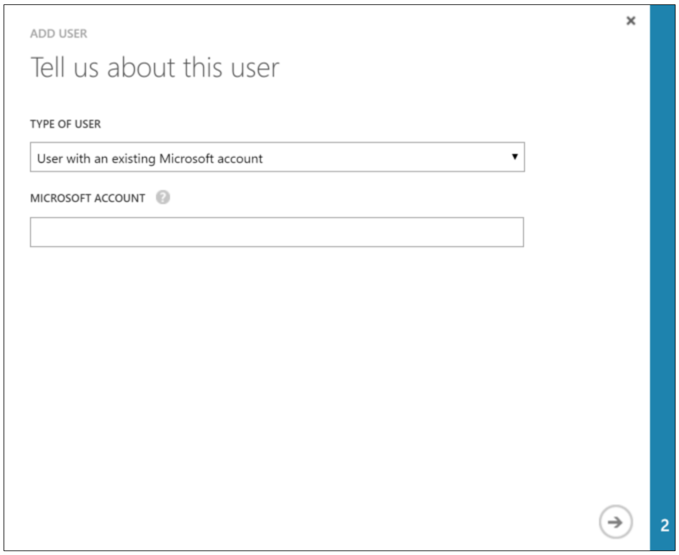
Successfully created user 'sonja@azureessentials.onmicrosoft.com' with the following new password

NEW PASSWORD

Saqa0431

Figure 7-15 Display the temporary password.

The process for adding a new user with an existing Microsoft account is similar. Instead of assigning the user to a domain name associated with your organization, you will use the user's Microsoft account (email address), as shown in Figure 7-16.


ADD USER

Tell us about this user

TYPE OF USER

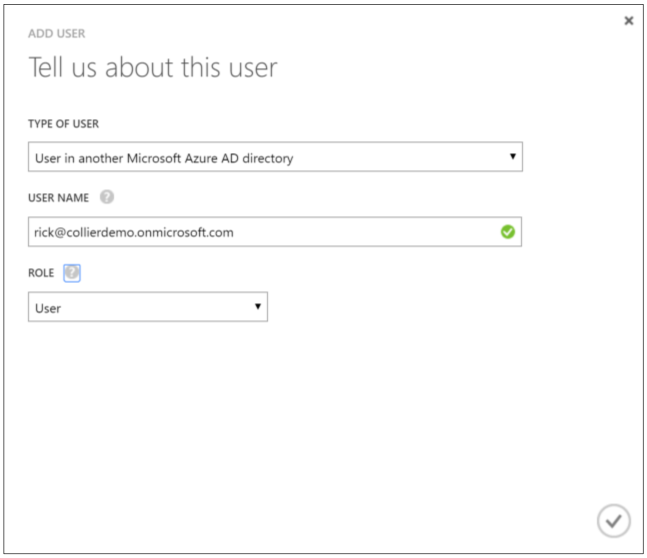
User with an existing Microsoft account

MICROSOFT ACCOUNT

Figure 7-16 Add a user with an existing Microsoft account.

After providing the user's email address, you will be able to provide the user's first name, last name, display name, and role, just like when adding a user in your organization.

Finally, when adding a new (external) user who is a user in another Azure AD directory, you will need to provide the name for the user. This is done in the form of the user's email account (or in Active Directory terms, the user's User Principal Name, or UPN), as seen in Figure 7-17. Keep in mind that to select a user in another Azure AD directory, you will also need to be an administrator in the other directory.


ADD USER

Tell us about this user

TYPE OF USER

User in another Microsoft Azure AD directory

USER NAME

rick@collierdemo.onmicrosoft.com

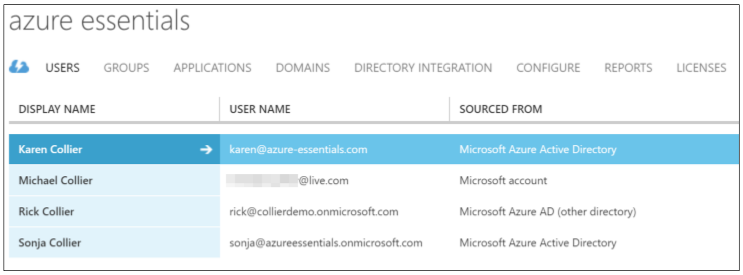
ROLE

User

Figure 7-17 Add an external user.

Adding external users to your directory copies the display names and user names from the source or home directory to your directory. The users would still authenticate against their home directory, but any changes to the users' properties (email address, display name, job title, and so on) do not propagate to your directory.

To determine where users in the directory originated, look at the Sourced From column in the Users section, as seen in Figure 7-18. If users were synchronized from a Windows Server Active Directory using Azure AD Connect sync, the Sourced From column would indicate Local Active Directory. Similarly, users from Office 365 would have the value *Office 365* in the Sourced From column, and users created in Azure AD would have the value *Microsoft Azure Active Directory* in the Sourced From column.


azure essentials

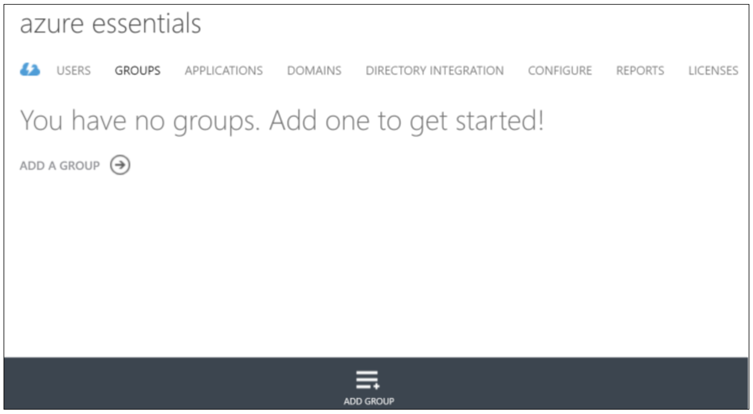
DISPLAY NAME	USER NAME	SOURCED FROM
Karen Collier	karen@azure-essentials.com	Microsoft Azure Active Directory
Michael Collier	[redacted]@live.com	Microsoft account
Rick Collier	rick@collierdemo.onmicrosoft.com	Microsoft Azure AD (other directory)
Sonja Collier	sonja@azureessentials.onmicrosoft.com	Microsoft Azure Active Directory

Figure 7-18 User list and source location.

Add groups

It is a common practice in Active Directory to organize users into groups. Groups make it easier to assign rights or grant access to resources. Instead of granting access for each individual user, access is granted to the group of which the user is a member. The user inherits the access rights of the group.

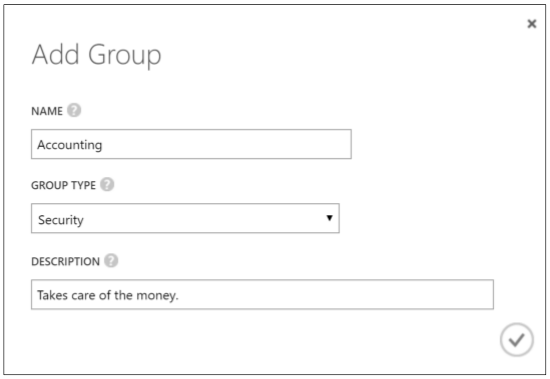
You can also create and manage groups in an Azure AD directory. Groups in Azure AD can be helpful when granting access to SaaS applications available in the Azure AD application gallery. The Groups section of the selected Azure AD directory allows you to create and manage groups. If no groups exist, create a group by clicking the Add Group button on the bottom command bar or the Add A Group link, as shown in Figure 7-19.



You have no groups. Add one to get started!

Figure 7-19 Add a group.

In the Add Group dialog box, you can provide a name, type (a security group or an Office 365 group), and description for the new group, as seen in Figure 7-20.



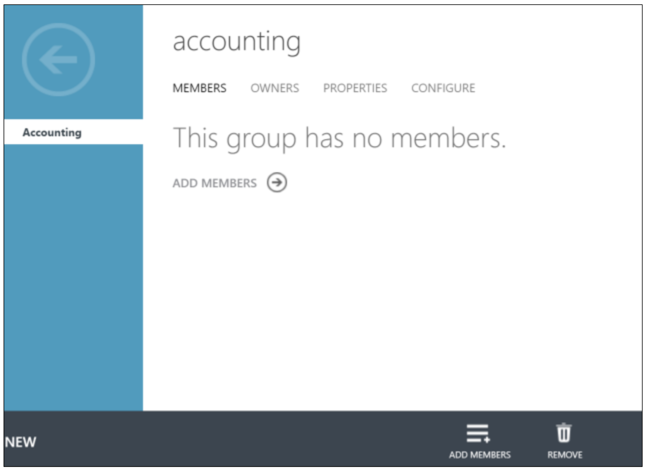
NAME: Accounting

GROUP TYPE: Security

DESCRIPTION: Takes care of the money.

Figure 7-20 Add group details.

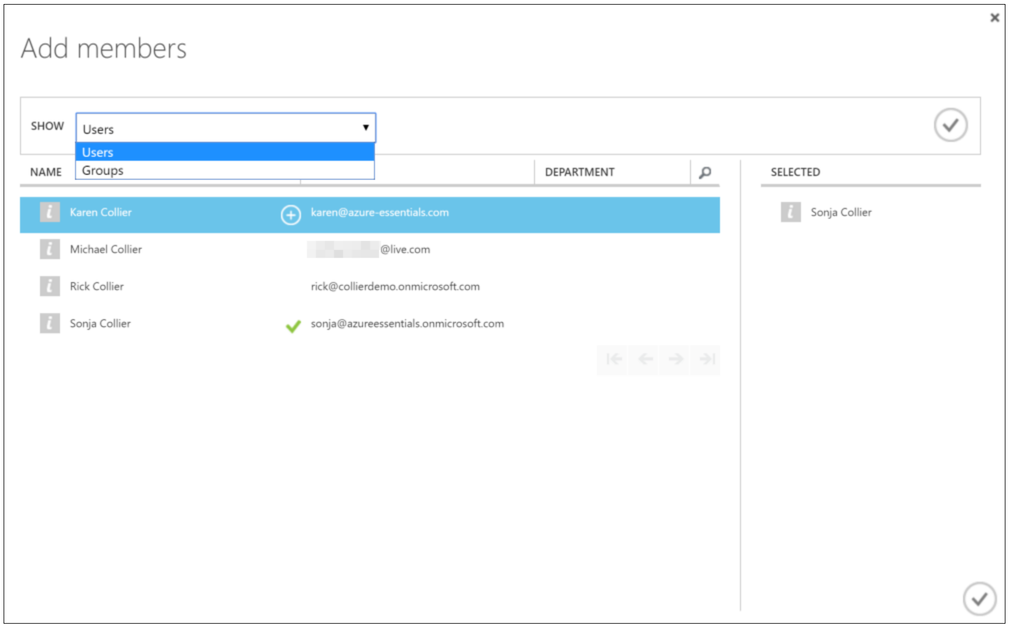
To add users to the group, select the group name to open a new screen that allows you to manage the group and then click the Add Members link or the Add Members button on the bottom command bar, as shown in Figure 7-21.



This group has no members.

Figure 7-21 Group with no members.

As seen in Figure 7-22, the Add Members dialog box enables you to select users or other groups in the current directory to add as members to the group selected.



Add members

SHOW Users

NAME		DEPARTMENT
Karen Collier	karen@azure-essentials.com	
Michael Collier	[redacted]@live.com	
Rick Collier	rick@collierdemo.onmicrosoft.com	
Sonja Collier	sonja@azureessentials.onmicrosoft.com	

SELECTED

Sonja Collier

Figure 7-22 Add members to a group.

Azure Multi-Factor Authentication

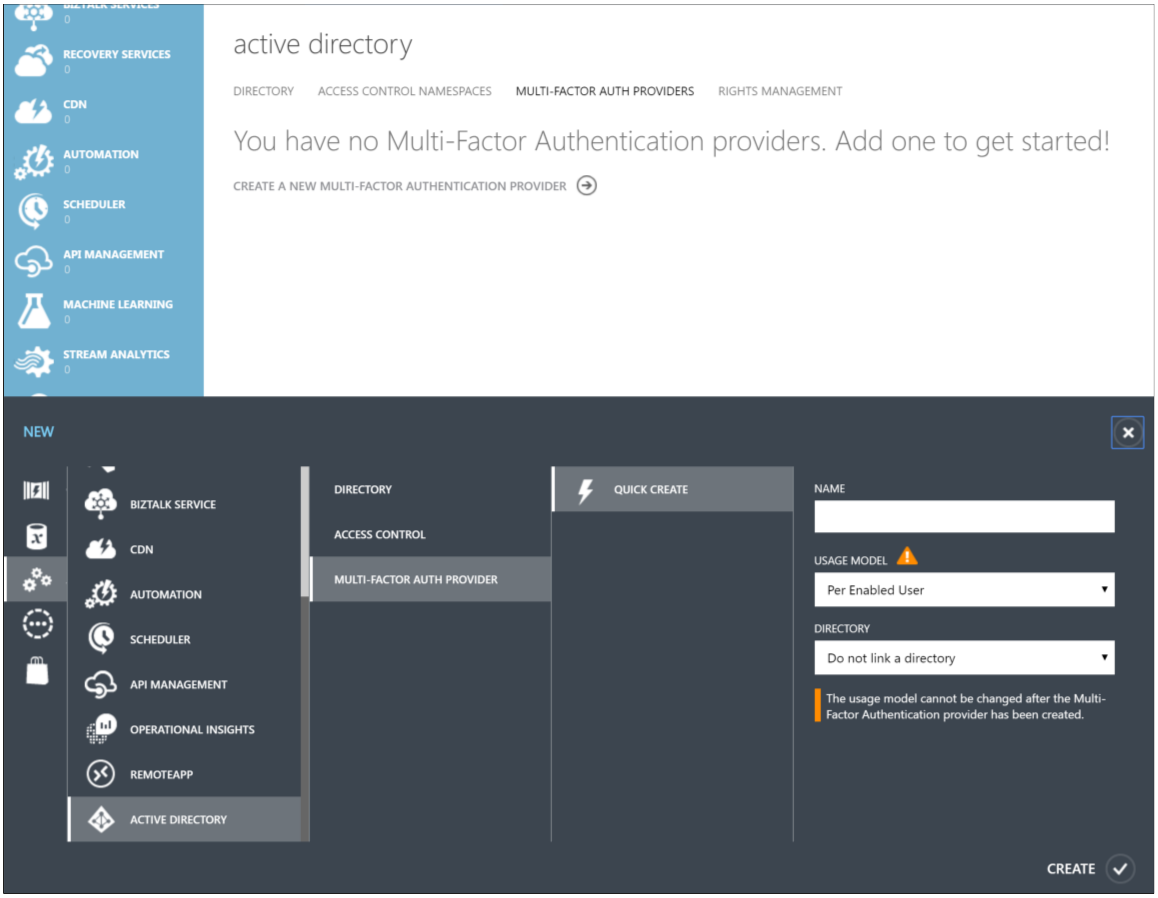
Azure Multi-Factor Authentication (MFA) provides additional security for on-premises or Azure-hosted solutions. For the purposes of this chapter, only Azure-hosted solutions are discussed. Azure MFA works by injecting a second authentication challenge that the user must complete successfully. Azure MFA complements a password-based authentication challenge (something you know) with a challenge based on something you have: a phone call, text message, or mobile app notification. Having multiple layers of protection makes it harder for attackers to access and compromise the account.

MFA comes in a few different varieties. Azure AD Free and Azure AD Basic support MFA for Azure administrators, and Azure AD Premium adds MFA support for users. MFA for Azure administrators provides security for their administrative account, thus adding security related to creating and managing resources such as Azure Virtual Machines, Azure App Services, and so on. MFA for users provides additional security for users signing into applications configured to support Azure AD.

Azure Multi-Factor Authentication

To begin using the full capabilities of Azure MFA, you will first need to add a Multi-Factor Authentication provider (MFA provider) in Azure AD. To add an MFA Provider, go to the Active Directory section in the Azure classic portal and then click the Multi-Factor Auth Providers area. If no providers exist, you are presented with an option to create one, as seen in Figure 7-23.

> **Note** A Multi-Factor Authentication provider is needed if you do not have an MFA license through Azure MFA (licenses purchased separately), Azure AD Premium, or Enterprise Mobility Suite (EMS). Azure MFA is included in Azure AD Premium and EMS. If you have licenses available, you do not need to create a Multi-Factor Authentication provider.



You have no Multi-Factor Authentication providers. Add one to get started!

CREATE A NEW MULTI-FACTOR AUTHENTICATION PROVIDER

NAME

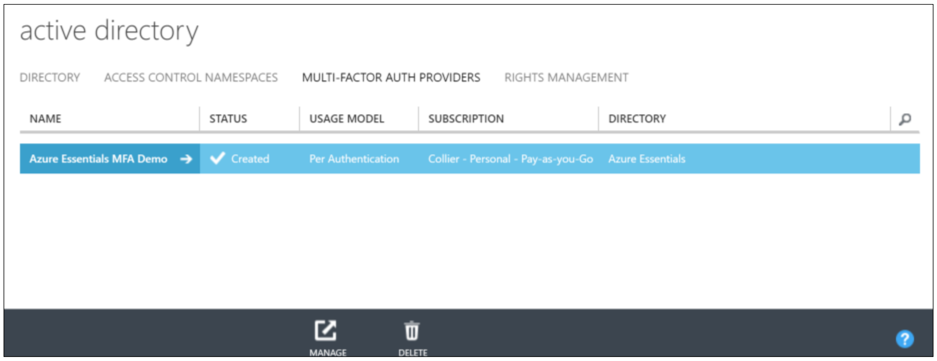
USAGE MODEL: Per Enabled User

DIRECTORY: Do not link a directory

The usage model cannot be changed after the Multi-Factor Authentication provider has been created.

Figure 7-23 Create a new Azure Multi-Factor Authentication provider.

After creating a new Multi-Factor Authentication provider, you can customize many advanced features of the service by clicking the Manage button, shown in Figure 7-24, to launch a new browser window that loads the Azure Multi-Factor Authentication management portal.



NAME	STATUS	USAGE MODEL	SUBSCRIPTION	DIRECTORY
Azure Essentials MFA Demo	✓ Created	Per Authentication	Collier - Personal - Pay-as-you-Go	Azure Essentials

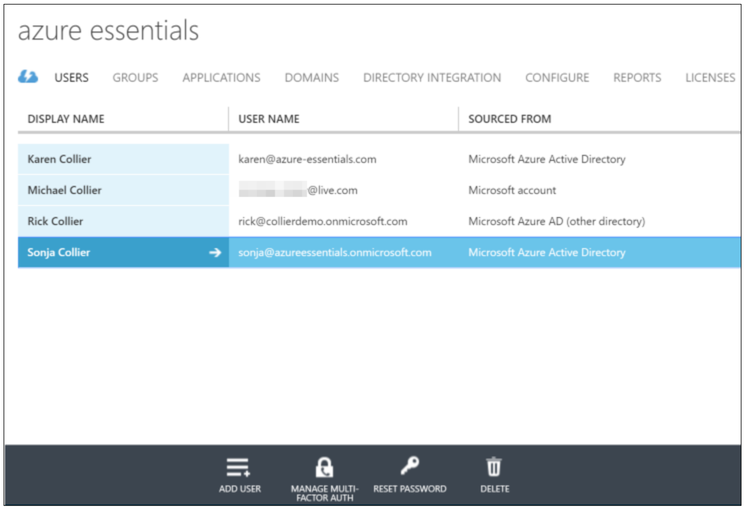
Figure 7-24 Azure MFA providers.

Note Azure MFA is a result of Microsoft's acquisition of PhoneFactor in October 2012. The Azure MFA management portal continues to reside on a phonefactor.net domain.

See Also For more information on Azure MFA, please refer to http://azure.microsoft.com/documentation/services/multi-factor-authentication/.

Multi-Factor Authentication for Azure administrators

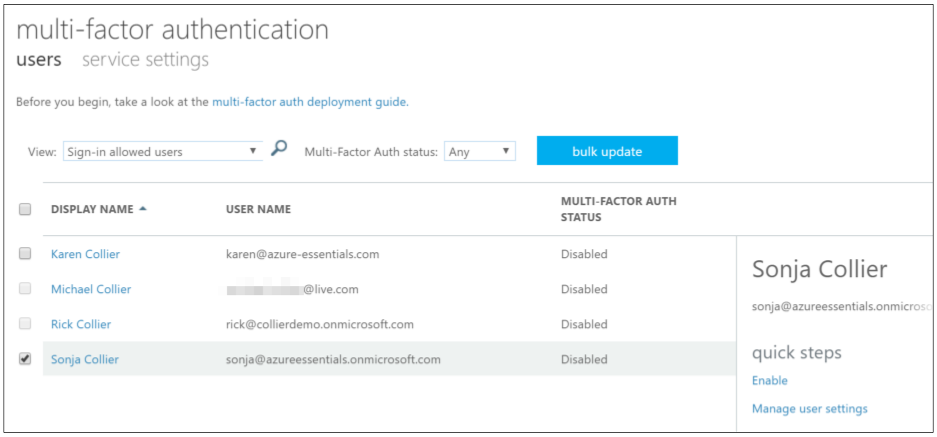
Recall from earlier in this chapter that when creating a new user, you had the option to enable MFA. Selecting the user's name and then the Manage Multi-Factor Auth button on the command bar, as shown in Figure 7-25, launches a new browser window and loads a site allowing you to manage MFA settings.



DISPLAY NAME	USER NAME	SOURCED FROM
Karen Collier	karen@azure-essentials.com	Microsoft Azure Active Directory
Michael Collier	[redacted]@live.com	Microsoft account
Rick Collier	rick@collierdemo.onmicrosoft.com	Microsoft Azure AD (other directory)
Sonja Collier →	sonja@azureessentials.onmicrosoft.com	Microsoft Azure Active Directory

Figure 7-25 Manage Multi-Factor Authentication.

Alternatively, if the user is not enabled for MFA, you can enable it from this site by selecting the desired user(s) and clicking Enable, as shown in Figure 7-26.



Before you begin, take a look at the multi-factor auth deployment guide.

View: Sign-in allowed users ▾ Multi-Factor Auth status: Any ▾

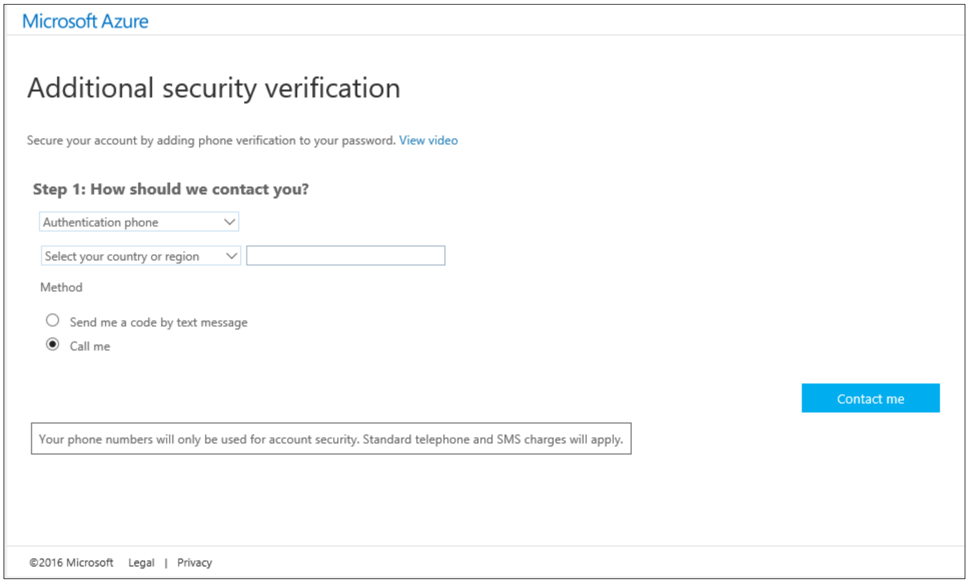
	DISPLAY NAME ▲	USER NAME	MULTI-FACTOR AUTH STATUS
☐	Karen Collier	karen@azure-essentials.com	Disabled
☐	Michael Collier	[redacted]@live.com	Disabled
☐	Rick Collier	rick@collierdemo.onmicrosoft.com	Disabled
☑	Sonja Collier	sonja@azureessentials.onmicrosoft.com	Disabled

Sonja Collier
sonja@azureessentials.onmicros

quick steps
Enable
Manage user settings

Figure 7-26 Enable Multi-Factor Authentication for selected users.

If MFA is enabled for a user, and the user attempts to access the Azure portal (the user must also be a coadministrator on the Azure subscription), the user will be presented with an additional security verification step. As seen in Figure 7-27, the first time the user attempts to log in, the user will be asked to specify the desired contact method—phone, text message, or mobile application—and complete the necessary steps.



Microsoft Azure

Additional security verification

Secure your account by adding phone verification to your password. View video

Step 1: How should we contact you?

Authentication phone

Select your country or region

Method

○ Send me a code by text message

● Call me

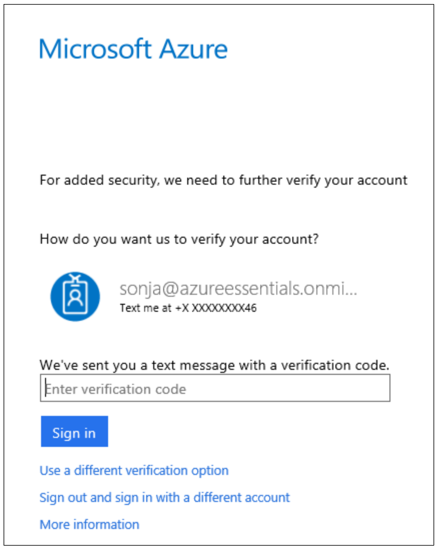
Contact me

Your phone numbers will only be used for account security. Standard telephone and SMS charges will apply.

©2016 Microsoft Legal | Privacy

Figure 7-27 Additional security verification details.

On subsequent login attempts, the user will need to provide the requested additional security verification answer. As shown in Figure 7-28, a text message verification code is requested.



Microsoft Azure

For added security, we need to further verify your account

How do you want us to verify your account?

sonja@azureessentials.onmi...

Text me at +X XXXXXXXX46

We've sent you a text message with a verification code.

Enter verification code

Sign in

Use a different verification option

Sign out and sign in with a different account

More information

Figure 7-28 Text message verification code.

Application gallery

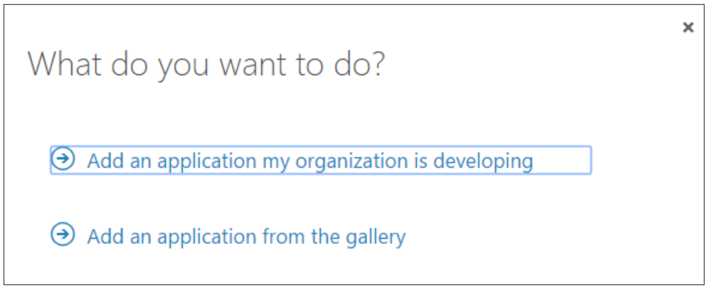
The Azure AD application gallery provides access to more than 2,500 popular SaaS applications such as Box, DocuSign, Salesforce, ServiceNow, Google Apps, and many more. Instead of IT administrators configuring access to each application separately and potentially managing various disparate logins, Azure AD simplifies the process by enabling SSO for the applications.

Azure AD supports three options for SSO:

- **Azure AD Single Sign-On** Uses the user's account information directly from Azure AD. If the user is already signed into Azure AD (or Office 365), there is no need for the user to reauthenticate when accessing the third-party SaaS application. A limited number of applications in the Azure AD application gallery support federation-based SSO.
- **Password Single Sign-On** Uses the user's account information from the third-party SaaS application. With this approach, the user's account information and password is collected, securely stored in Azure AD, and provided to the SaaS application via a web browser extension. The majority of applications in the Azure AD application gallery support this form of SSO. The browser extension is supported on Internet Explorer (IE8 through IE11 on Windows 7 or later), Chrome (Windows 7 or later; MacOS X or later), and Firefox (26.0 or later on Windows XP SP2 or later and Mac OS X 10.6 or later).
- **Existing Single Sign-On** Uses an existing Active Directory Federation Services (AD FS) or other third-party provider for SSO.

Adding gallery applications

To add gallery applications to your Azure AD directory, first select the desired directory and then the Applications section. If you don't have any applications in the gallery, click Add An Application to open a new dialog box, as shown in Figure 7-29, presenting you with two options: add an application you are developing or add an application from the gallery.



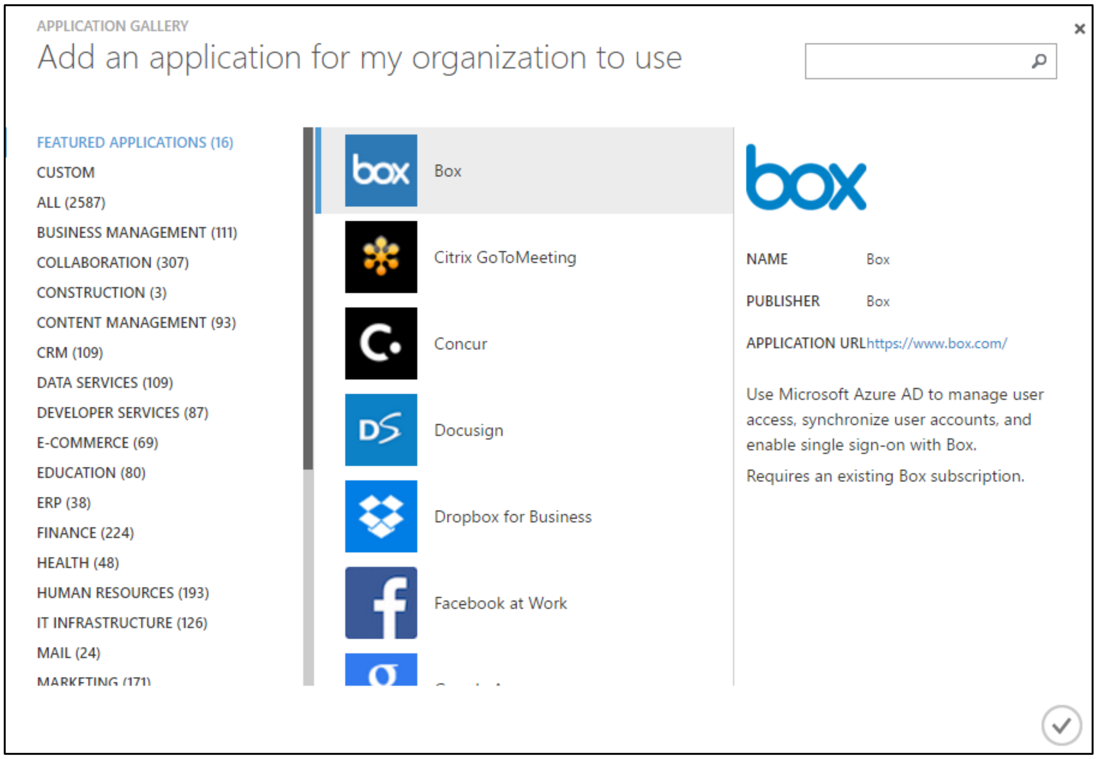
What do you want to do?

×
Add an application my organization is developing
Add an application from the gallery

Figure 7-29 Add an application to Azure AD.

If you or another member of your organization develop an application in which you would like to take advantage of Azure AD features (for example, SSO, ability to query the Azure AD Graph API, and so on), then select the first option.

Selecting the second option will allow you to register third-party SaaS applications with your Azure AD directory. As can be seen in Figure 7-30, there are many applications available. Use the filter box to search by name for a specific application.



NAME Box

PUBLISHER Box

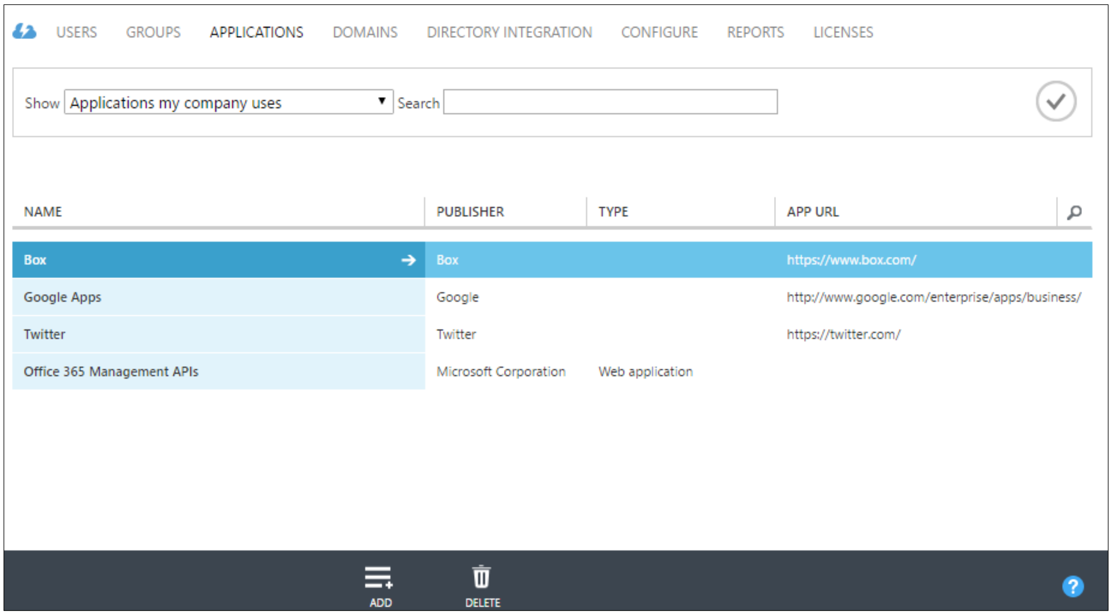
APPLICATION URL https://www.box.com/

Use Microsoft Azure AD to manage user access, synchronize user accounts, and enable single sign-on with Box.

Requires an existing Box subscription.

Figure 7-30 Select a gallery application.

After you select the desired application(s), each application will appear on the list in the Applications section. As shown in Figure 7-31, you can add an application by clicking Add, and you can remove an application by selecting the application and clicking Delete.



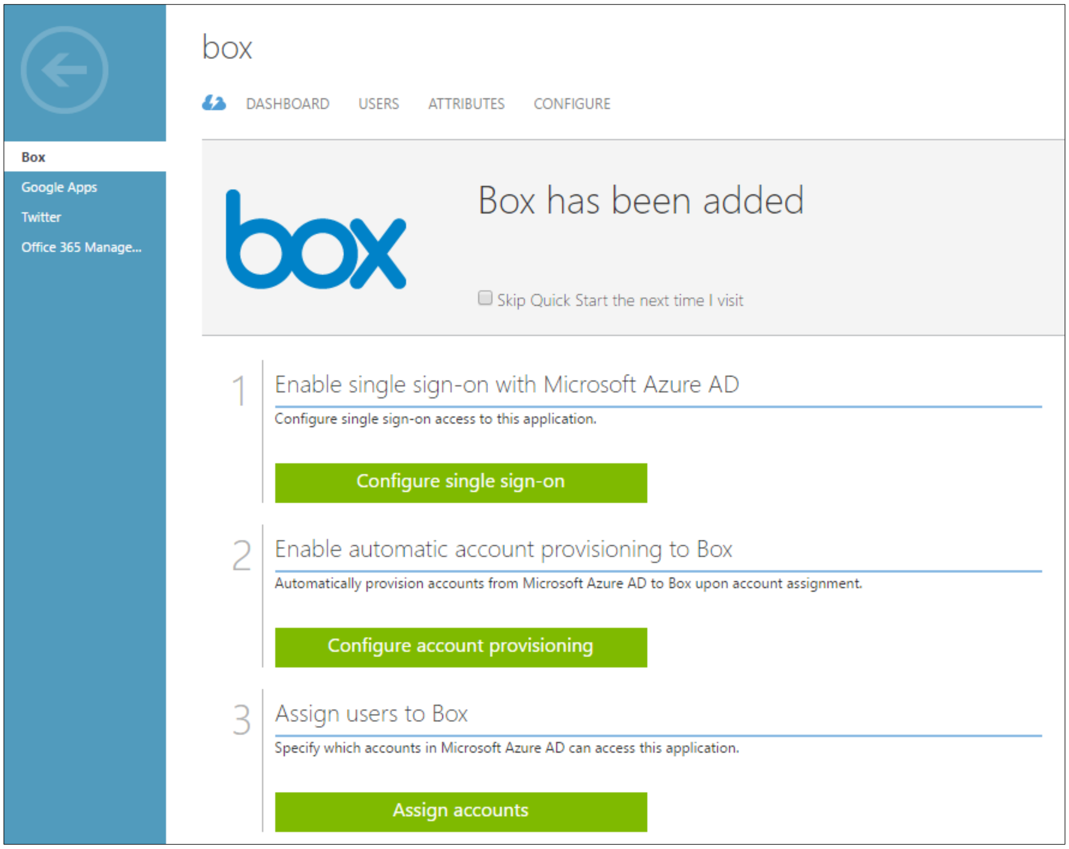
Show Applications my company uses Search

NAME	PUBLISHER	TYPE	APP URL
Box	Box		https://www.box.com/
Google Apps	Google		http://www.google.com/enterprise/apps/business/
Twitter	Twitter		https://twitter.com/
Office 365 Management APIs	Microsoft Corporation	Web application	

Figure 7-31 List, add, and delete applications.

Assigning users to applications

As mentioned previously, there are different ways applications can leverage Azure AD for SSO. For example, Box provides all three options (Azure AD Single Sign-On, Password Single Sign-On, and Existing Single Sign-On), and Twitter provides only Password Single Sign-On and Existing Single Sign-On. Follow the on-screen guidance, as shown in Figure 7-32, to configure SSO for the selected application.



Box has been added

☐ Skip Quick Start the next time I visit

1 Enable single sign-on with Microsoft Azure AD

Configure single sign-on access to this application.

2 Enable automatic account provisioning to Box

Automatically provision accounts from Microsoft Azure AD to Box upon account assignment.

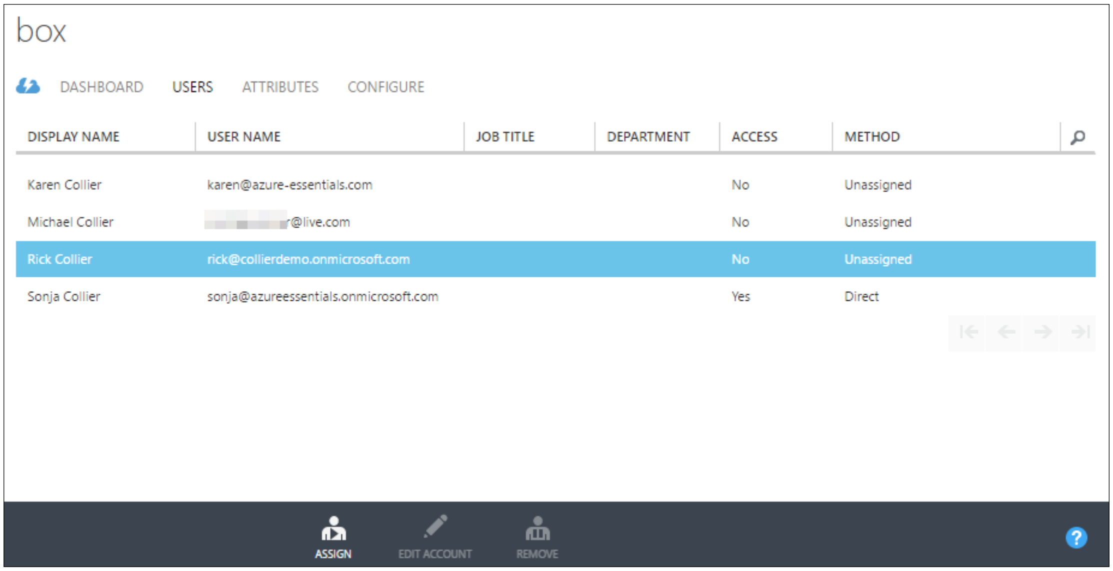
3 Assign users to Box

Specify which accounts in Microsoft Azure AD can access this application.

Figure 7-32 Configuration for gallery application.

Some applications, such as Box, have the ability to provision (create) users automatically into the application once the user is assigned access in Azure AD. If a user is deleted from Azure AD or the user's access is revoked, this change is automatically propagated to the SaaS application. This can greatly simplify IT administration responsibilities.

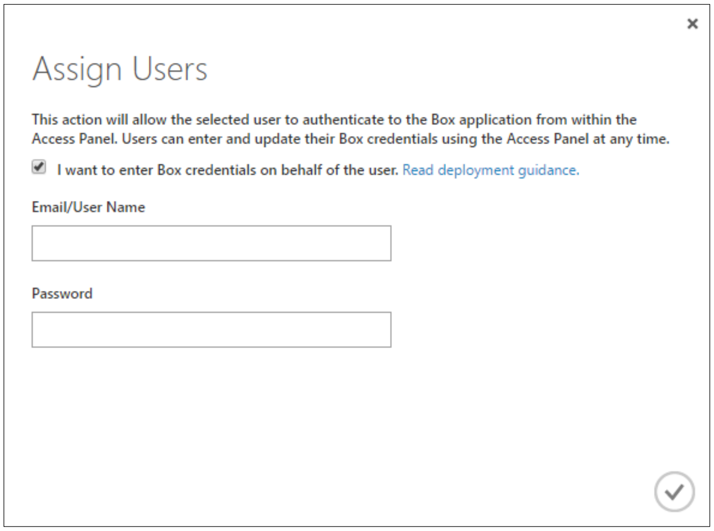
To grant Azure AD directory users access to the selected application, first click Assign Accounts to access a new screen allowing you to assign or revoke access. As shown in Figure 7-33, select the desired user and then click either Assign or Remove on the bottom command bar.



DISPLAY NAME	USER NAME	JOB TITLE	DEPARTMENT	ACCESS	METHOD
Karen Collier	karen@azure-essentials.com			No	Unassigned
Michael Collier	[redacted]r@live.com			No	Unassigned
Rick Collier	rick@collierdemo.onmicrosoft.com			No	Unassigned
Sonja Collier	sonja@azureessentials.onmicrosoft.com			Yes	Direct

Figure 7-33 Assign users.

For applications that use Password Single Sign-On, after assigning users you will be presented with an option to allow users to enter and update their own credentials for the application, or you can enter them on their behalf, as shown in Figure 7-34.



Assign Users

This action will allow the selected user to authenticate to the Box application from within the Access Panel. Users can enter and update their Box credentials using the Access Panel at any time.

☑ I want to enter Box credentials on behalf of the user. Read deployment guidance.

Email/User Name

Password

Figure 7-34 Assign user credentials.

See Also For a step-by-step tutorial on integrating SaaS applications with Azure AD, please refer to https://azure.microsoft.com/documentation/articles/active-directory-saas-tutorial-list/.

MyApps

Once users are assigned access to the SaaS applications available via the Azure AD application gallery, they can access those applications from the MyApps for Azure Active Directory site available at http://myapps.microsoft.com. Users will need to first authenticate with their Azure AD credentials (either *.onmicrosoft.com or associated custom domain name) to gain access to the site. If the user is already signed into a Microsoft cloud service site (Azure, Office 365, and so on), the user will automatically be signed into the MyApps site. As can be seen in Figure 7-35, there are two main sections of the site: Applications and Profile.

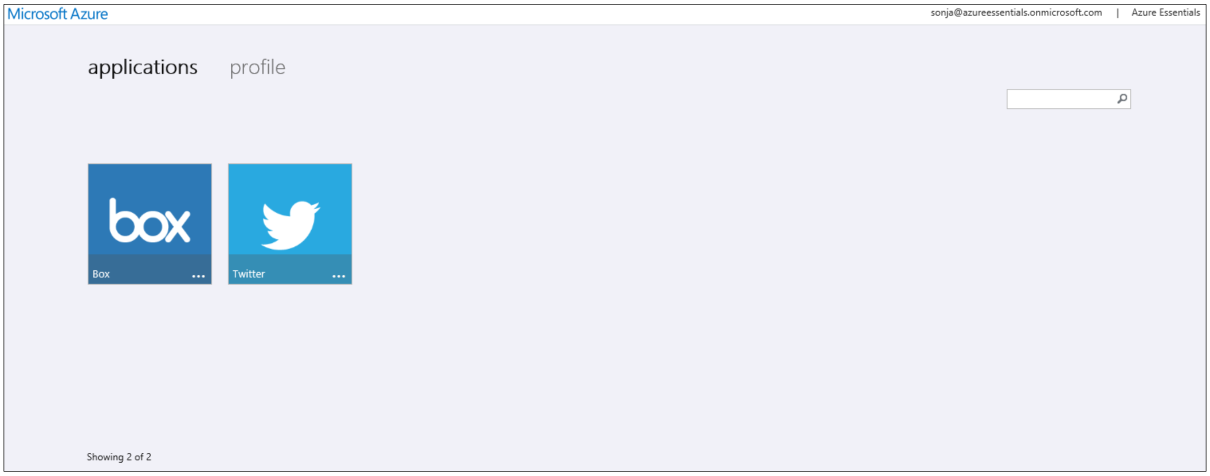
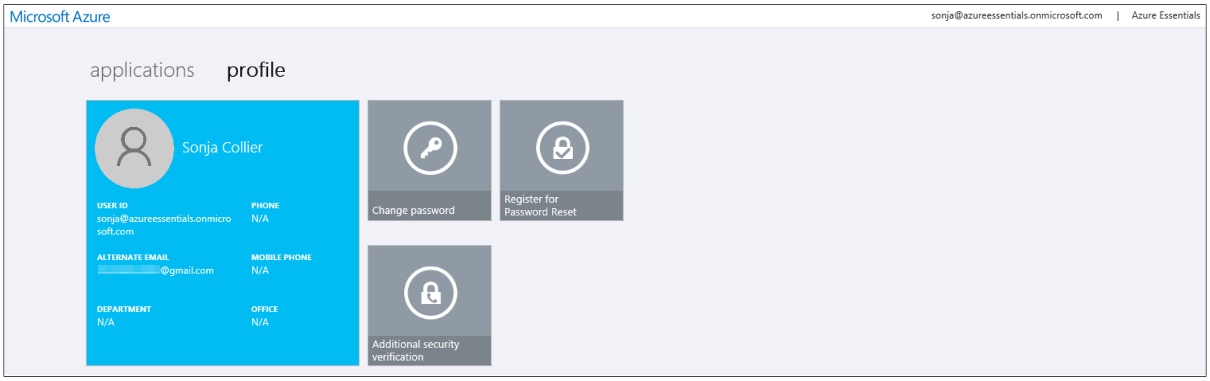


Figure 7-35 Applications in MyApps site.

The Applications section provides a tile for each application to which the user has been granted access. If password-based SSO was selected as the authentication method when the application was added to the organization's Azure AD directory, the user might be prompted to install a browser component the first time the application is accessed.

To access the desired application, users need to click the application's tile. If password-based SSO was selected, users will be prompted to enter their credentials for the SaaS application, unless the administrator who added the application did that on the users' behalf. The users are then redirected to the application and signed in using the appropriate credentials.

The Profile section, shown in Figure 7-36, provides some basic details about the current user and an easy way for users to change their passwords.


Microsoft Azure
sonja@azureessentials.onmicrosoft.com | Azure Essentials
applications
profile
Change password
Register for Password Reset
Additional security verification

Sonja Collier

USER ID
sonja@azureessentials.onmicrosoft.com

PHONE
N/A

ALTERNATE EMAIL
[redacted]@gmail.com

MOBILE PHONE
N/A

DEPARTMENT
N/A

OFFICE
N/A

Figure 7-36 User profile in MyApps site.

Management tools

In earlier chapters in this book, you have explored several prominent features of the Microsoft Azure platform. You have learned how to create Azure web applications, cloud services, virtual machines, storage accounts, Azure SQL database instances, and much more. The majority of examples have demonstrated using the Azure portal. Although the portal is a great way to work with Azure resources, other tools will also prove useful during development and management of those resources.

Management tools overview

There are many excellent tools available to aid in the development and management of Azure solutions—in fact, too many to cover in this chapter. Microsoft Visual Studio, PowerShell, and the Azure cross-platform command-line (also commonly referred to as the Azure CLI) tools are covered in this chapter. To assist in your awareness, Table 8-1 lists a few Azure features and related tools.

Table 8-1 Various tools for Azure management

Azure Feature	Tool Options
Azure SQL Database	SQL Server Management Studio
Azure Virtual Machines	• Microsoft System Center • Microsoft Operations Management Suite • Puppet • Chef • PowerShell DSC
Azure Storage	• Microsoft Azure Storage Explorer • See the Azure Storage team blog post at http://blogs.msdn.com/b/windowsazurestorage/archive/2014/03/11/windows-azure-storage-explorers-2014.aspx for a list of popular tools. • AzCopy (included in the Azure SDK)
Azure Service Bus	Service Bus Explorer
Azure Management APIs	Microsoft Azure Management Library REST APIs

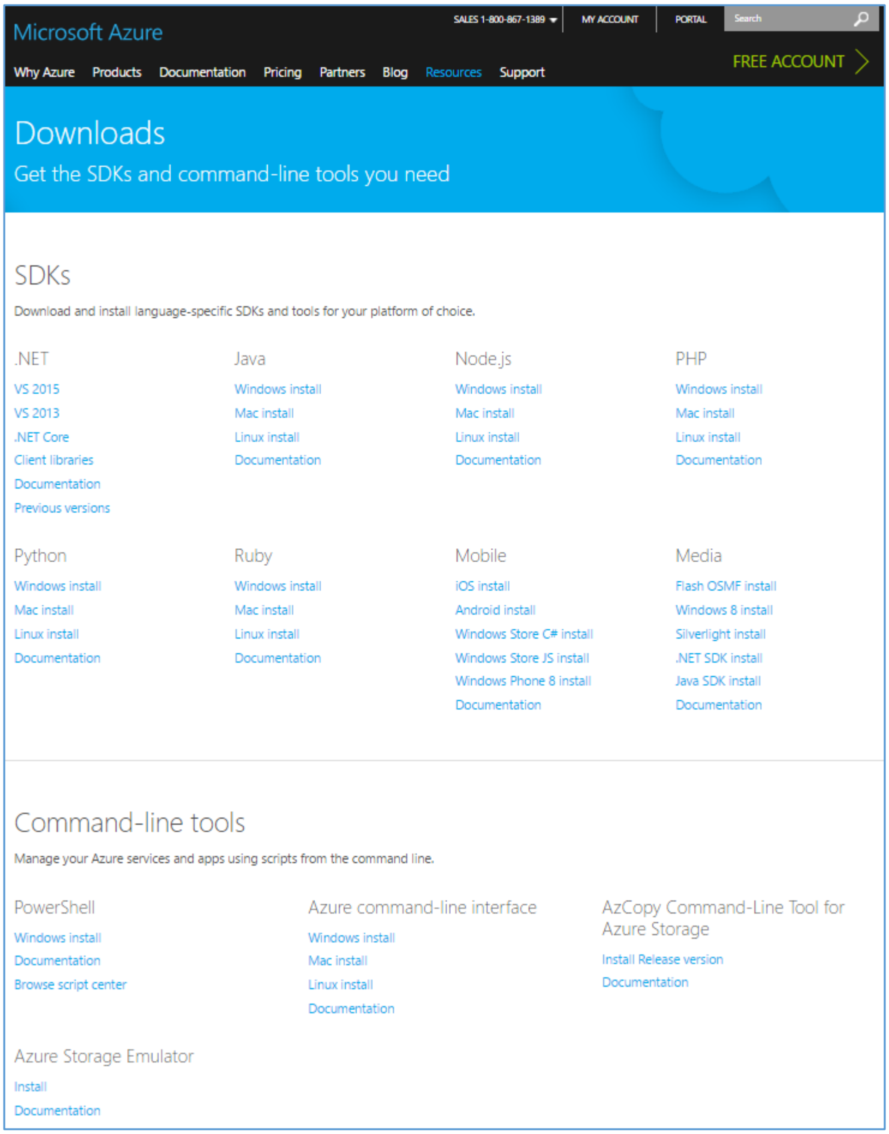
For developers, Visual Studio provides a rich, integrated experience to develop, deploy, and maintain applications in Azure. For IT professionals, the Azure PowerShell cmdlets and Azure CLI provide a robust and powerful scripting environment to manage resources deployed in Azure. Some advanced features are only available via PowerShell. The Azure CLI tools provide a simple yet powerful way to manage Azure resources regardless of your operating system because the Azure CLI works equally well across Windows, Linux, and Mac systems.

Visual Studio 2015 and the Azure SDK

Visual Studio is likely to be an Azure developer's primary interface for the development and management of Azure resources. This is especially the case for those developers creating solutions on the Microsoft technology stack (Windows, .NET, and so on). Developers using Linux or Mac systems will focus primarily on the Azure CLI. Any developer or IT professional can use either of the Azure management portals.

Install the Azure SDK

As an Azure developer, one of the first things you will want to do (after installing Visual Studio) is install the Microsoft Azure SDK for .NET. You can obtain the Azure SDK from the Azure Downloads page at http://azure.microsoft.com/downloads/, as shown in Figure 8-1. Install the SDK appropriate for your version of Visual Studio.



Downloads

Get the SDKs and command-line tools you need

SDKs

Download and install language-specific SDKs and tools for your platform of choice.

.NET	Java	Node.js	PHP
VS 2015	Windows install	Windows install	Windows install
VS 2013	Mac install	Mac install	Mac install
.NET Core	Linux install	Linux install	Linux install
Client libraries	Documentation	Documentation	Documentation
Documentation			
Previous versions			

Python	Ruby	Mobile	Media
Windows install	Windows install	iOS install	Flash OSMF install
Mac install	Mac install	Android install	Windows 8 install
Linux install	Linux install	Windows Store C# install	Silverlight install
Documentation	Documentation	Windows Store JS install	.NET SDK install
		Windows Phone 8 install	Java SDK install
		Documentation	Documentation

Command-line tools

Manage your Azure services and apps using scripts from the command line.

PowerShell	Azure command-line interface	AzCopy Command-Line Tool for Azure Storage
Windows install	Windows install	Install Release version
Documentation	Mac install	Documentation
Browse script center	Linux install	
	Documentation	

Azure Storage Emulator

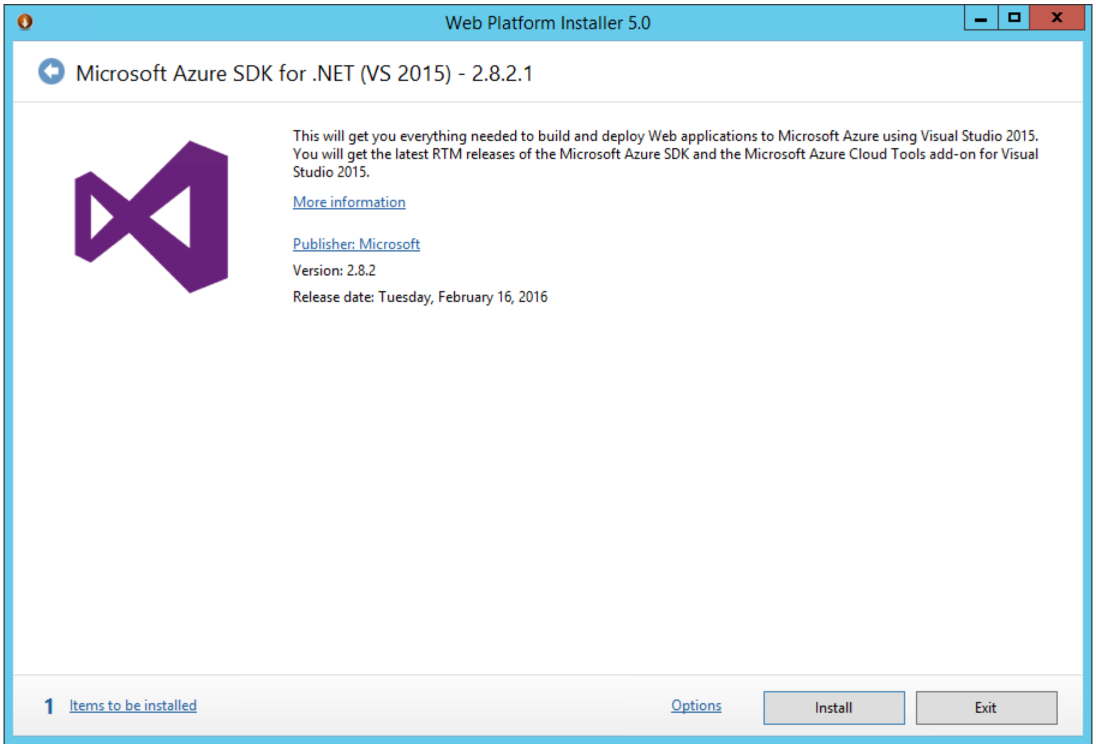
Install

Documentation

Figure 8-1 Azure Downloads page.

You likely will want to bookmark this page. You will come back to it often because it is the source for various language SDKs (Java, PHP, Ruby, and so on) and the PowerShell and Azure command-line interface tools.

Installing the Microsoft Azure SDK for .NET is done by using the Microsoft Web Platform Installer (Web PI), as shown in Figure 8-2. The SDK includes any necessary client libraries required to work with Azure services such as Storage, Service Bus, Cloud Services, and so on. The SDK installation will also configure Visual Studio for full development and debugging support in Azure Web Apps and Cloud Services. Additionally, the SDK includes the Storage and Compute emulators—great for developing projects when you cannot be connected to the Internet (and thus Azure). Web PI will also take care of installing any necessary dependencies. The entire process will take several minutes.



Microsoft Azure SDK for .NET (VS 2015) - 2.8.2.1

This will get you everything needed to build and deploy Web applications to Microsoft Azure using Visual Studio 2015. You will get the latest RTM releases of the Microsoft Azure SDK and the Microsoft Azure Cloud Tools add-on for Visual Studio 2015.

More information

Publisher: Microsoft

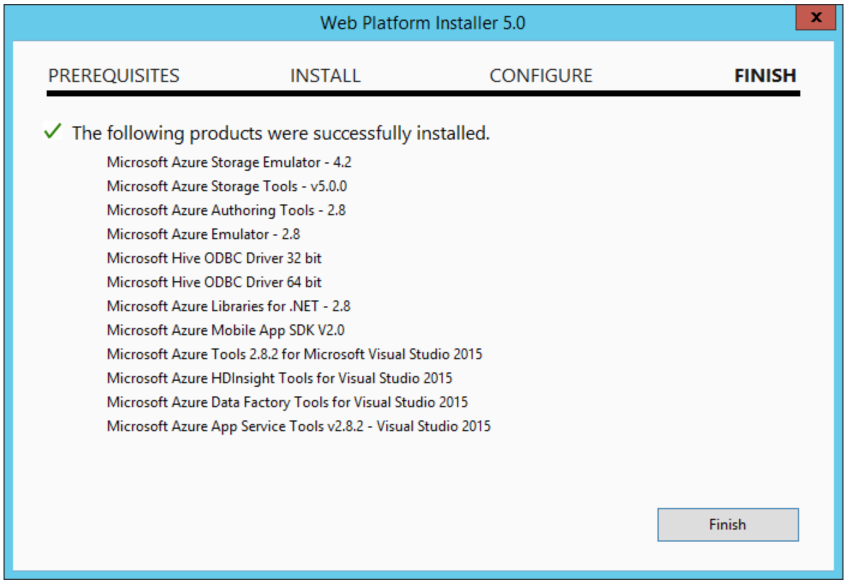
Version: 2.8.2

Release date: Tuesday, February 16, 2016

1 Items to be installed

Figure 8-2 Microsoft Azure SDK for .NET (Visual Studio 2015).

When finished, you should see a dialog box like that shown in Figure 8-3, showing the various products installed as part of Microsoft Azure SDK for .NET.



✓ The following products were successfully installed.

- Microsoft Azure Storage Emulator - 4.2
- Microsoft Azure Storage Tools - v5.0.0
- Microsoft Azure Authoring Tools - 2.8
- Microsoft Azure Emulator - 2.8
- Microsoft Hive ODBC Driver 32 bit
- Microsoft Hive ODBC Driver 64 bit
- Microsoft Azure Libraries for .NET - 2.8
- Microsoft Azure Mobile App SDK V2.0
- Microsoft Azure Tools 2.8.2 for Microsoft Visual Studio 2015
- Microsoft Azure HDInsight Tools for Visual Studio 2015
- Microsoft Azure Data Factory Tools for Visual Studio 2015
- Microsoft Azure App Service Tools v2.8.2 - Visual Studio 2015

Figure 8-3 Completed Azure SDK installation.

See Also For release notes on the various Azure SDK for .NET versions, please refer to http://msdn.microsoft.com/library/azure/dn627519.aspx.

Manage resources with Cloud Explorer

Cloud Explorer is a new tool in Visual Studio that enables you to manage various Azure resources supported under the Azure Resource Manager (ARM) model. It is installed as part of the Azure SDK installation and is available as a dockable pane in Visual Studio (similar to the Sever Explorer or Toolbox panes).

Note Visual Studio contains two tools for managing Azure resources: Cloud Explorer and Server Explorer. Cloud Explorer, introduced in Azure SDK 2.7, is the preferred tool for managing Azure resources based on Resource Manager. Server Explorer works with Azure classic resources, but it does not work with all Resource Manger–based resources. For the purposes of this chapter, the focus will be on Cloud Explorer.

The first thing you need to do is connect to your Azure subscription(s), which you can do by clicking the gear icon near the top of Cloud Explorer and then selecting the Add An Account link, as seen in Figure 8-4.

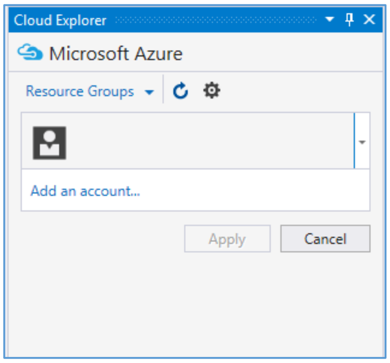
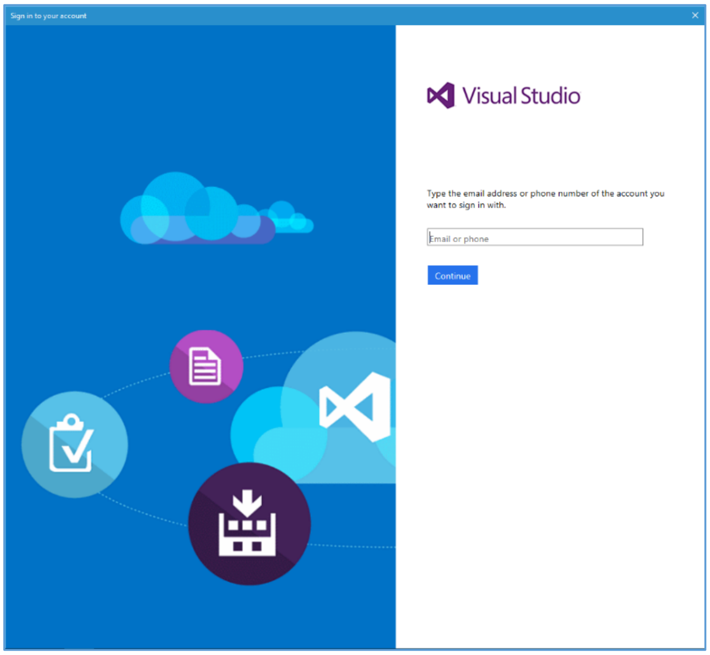


Figure 8-4 Add a new account to Cloud Explorer.

This opens a dialog box, as seen in Figure 8-5, prompting for the email address of the account used to sign into Azure. This should be the same email address used when signing into the Azure portal.

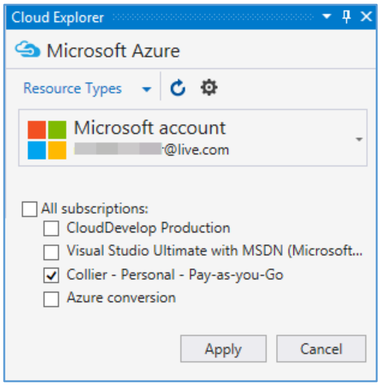


Type the email address or phone number of the account you want to sign in with.

Figure 8-5 Sign into your Azure subscription.

After authenticating, Visual Studio will automatically download all the necessary configuration details to connect to any Azure subscriptions to which the provided account has access.

After authenticating, you will be able to view all Azure subscriptions to which you have access. From here, you can filter the Azure subscriptions, as seen in Figure 8-6. If you want to filter out certain subscriptions so that you do not see them (or the related Azure resources) while working with Azure resources in Visual Studio, just clear the appropriate check boxes.



[redacted]@live.com

□ All subscriptions:
□ CloudDevelop Production
□ Visual Studio Ultimate with MSDN (Microsoft...
☑ Collier - Personal - Pay-as-you-Go
□ Azure conversion

Figure 8-6 Manage Microsoft Azure subscriptions.

To manage an Azure resource, just expand the node for that resource. Expanding the node lists all the resources under the selected type, as seen in Figure 8-7.

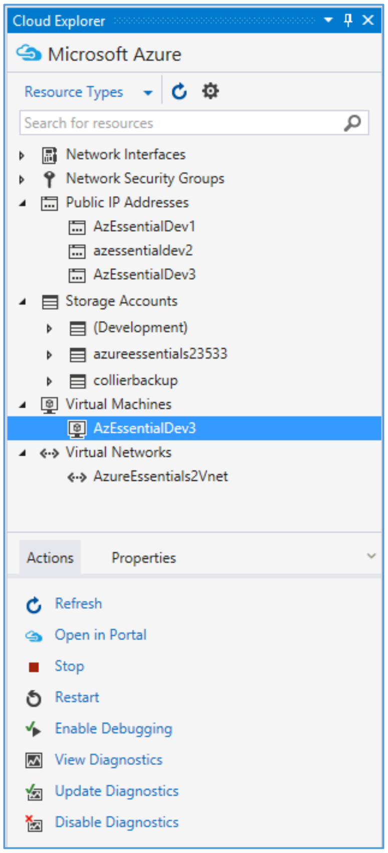
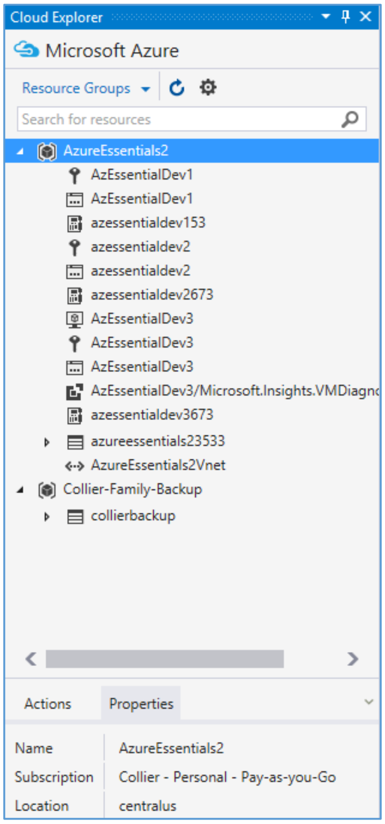


Figure 8-7 Expanding Azure resources.

To work with a specific resource, select it from the list. An Actions tab and a Properties tab will list additional information about the selected resource. The available actions will vary by resource. For example, if you select a virtual machine (VM), you will be able to start or stop the VM or work with its diagnostics. For other resources, such as a storage account, the only available option is to launch into the Azure portal.

Cloud Explorer also allows you to filter resources by Resource Groups or Resource Type (the default). By selecting the drop-down arrow next to Resource Type (at the top of the Cloud Explorer window), you can change between views. As can be seen in Figure 8-8, the Resource Groups view will collate resources according to their respective resource groups.



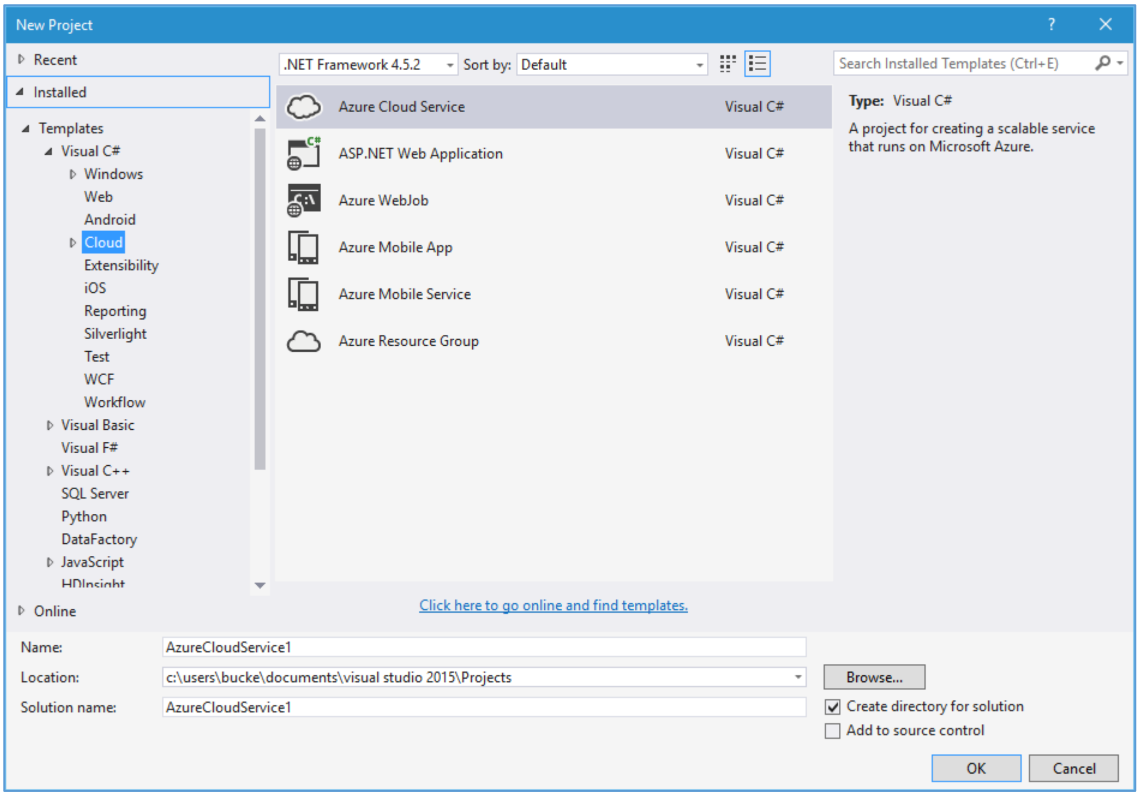
- AzureEssentials2
  - AzEssentialDev1
  - AzEssentialDev1
  - azessentialdev153
  - azessentialdev2
  - azessentialdev2
  - azessentialdev2673
  - AzEssentialDev3
  - AzEssentialDev3
  - AzEssentialDev3
  - AzEssentialDev3/Microsoft.Insights.VMDiagno
  - azessentialdev3673
  - azureessentials23533
  - AzureEssentials2Vnet
- Collier-Family-Backup
  - collierbackup

Name	AzureEssentials2
Subscription	Collier - Personal - Pay-as-you-Go
Location	centralus

Figure 8-8 View Azure resources by resource group.

Create an Azure resource

When the time comes to create a new Azure resource and write code to go with it, Visual Studio and the Azure SDK can be helpful tools. When creating a new Visual Studio project, navigate to the Cloud template section in the New Project Wizard, as shown in Figure 8-9. From there, you can select the appropriate option, such as create a new Azure Cloud Service, an Azure Mobile App, a new Azure Resource Group, and so on.



.NET Framework 4.5.2 Sort by: Default

Azure Cloud Service	Visual C#
ASP.NET Web Application	Visual C#
Azure WebJob	Visual C#
Azure Mobile App	Visual C#
Azure Mobile Service	Visual C#
Azure Resource Group	Visual C#

Type: Visual C#

A project for creating a scalable service that runs on Microsoft Azure.

Name: AzureCloudService1

Location: c:\users\bucke\documents\visual studio 2015\Projects

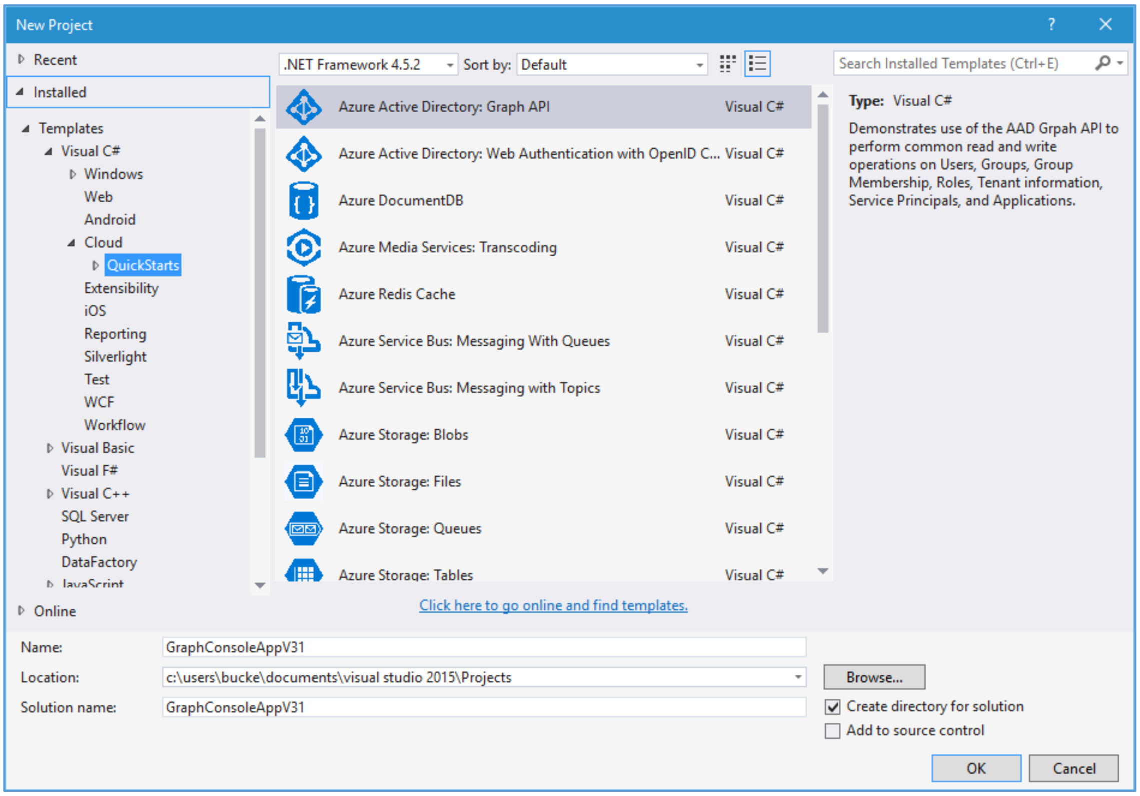
Solution name: AzureCloudService1

☑ Create directory for solution

□ Add to source control

Figure 8-9 Create a new Cloud project in Visual Studio.

The QuickStarts section, shown in Figure 8-10, provides a list of Microsoft-provided sample projects for specific Azure technologies. This can be a great way to learn basic features of a new or unfamiliar technology.



Type: Visual C#

Demonstrates use of the AAD Grpah API to perform common read and write operations on Users, Groups, Group Membership, Roles, Tenant information, Service Principals, and Applications.

Name: GraphConsoleAppV31

Location: c:\users\bucke\documents\visual studio 2015\Projects

Solution name: GraphConsoleAppV31

☑ Create directory for solution

☐ Add to source control

Figure 8-10 QuickStarts with Azure technologies.

Windows PowerShell

Visual Studio is an incredibly powerful tool for creating and managing solutions hosted in Azure. As great as it is, Visual Studio might not be the right tool for all scenarios. Instead, a scripting tool such as PowerShell could be the right choice. Scenarios in which PowerShell might be preferred include the following:

- **Tool for IT professionals** Many IT professionals do not use Visual Studio for management of on-premises assets. It makes sense that they would not want to use Visual Studio for Azure-hosted assets, either. Instead, PowerShell is often one of the preferred tools, especially for managing Windows environments.
- **Automating provisioning and deployment of Azure resources** PowerShell provides a rich scripting environment for automating the provisioning and deployment of Azure resources. By using PowerShell scripts, you can automate the provisioning and management of Azure Virtual Machines, Azure Storage accounts, Azure Web Apps, Azure Virtual Networks, Azure Cloud Services, and much more.

  Using PowerShell scripts to automate common tasks in Azure is also a great way to reduce potential errors. The scripts can be tested thoroughly, secured in a source control system, and repeatedly used with confidence that the same results will always be achieved. Azure Resource Manager (ARM) templates are often invoked via a PowerShell script.

  If you have a task that you are going to perform repeatedly, it is beneficial to automate the task. It might take a greater time commitment initially to develop the script, but doing so will save a substantial amount of time every time you reuse it.

- **Accessing advanced or new Azure features not included in the Azure tools for Visual Studio or the Azure portal** The Azure PowerShell cmdlets provide features that may not currently be available in Visual Studio. The update and release frequency for Azure PowerShell cmdlets is faster than that of the Azure SDK for Visual Studio. Because of this, you will often see new features appear in the PowerShell cmdlets (and REST API) before they appear in Visual Studio.

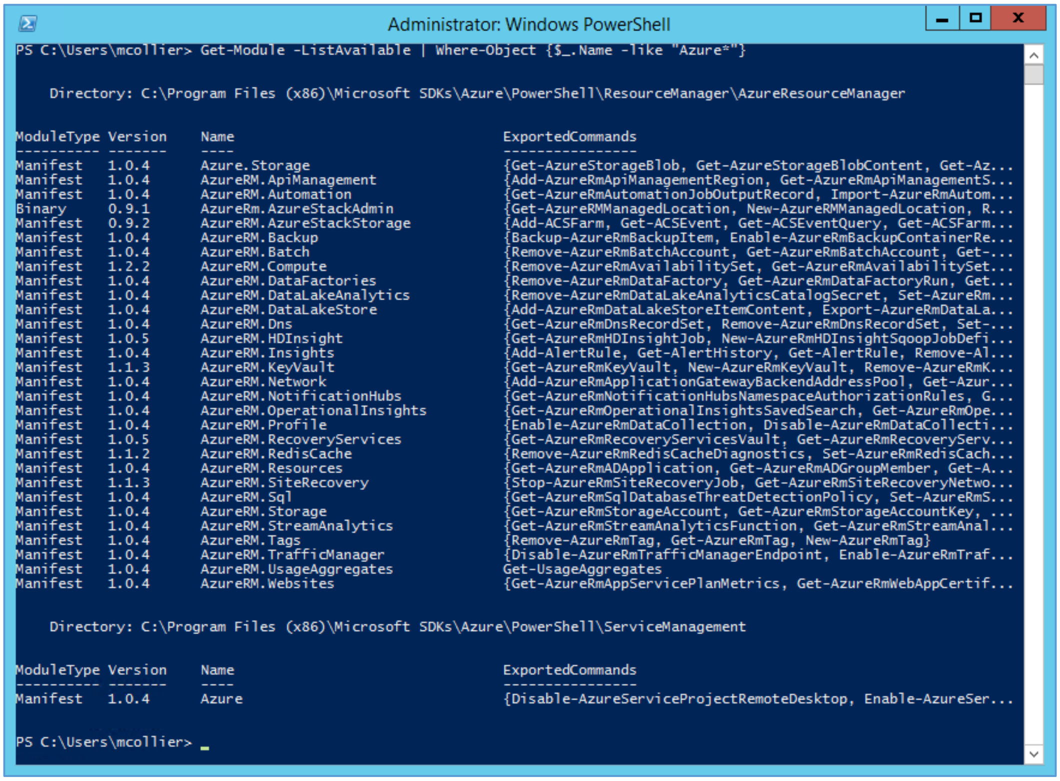
  Additionally, new features are often released that are surfaced in the Azure PowerShell cmdlets but are not surfaced in the Azure portal (at least initially). This allows the Microsoft Azure product team to release the feature—and in some cases, fine-tune it—before releasing the feature to the Azure portal and any related UI elements.

Azure PowerShell cmdlet installation

The Azure PowerShell cmdlets can be obtained one of two ways: either via the Web Platform Installer (Web PI) or the PowerShell Gallery.

Starting with Azure PowerShell 1.0, both Azure management interfaces (Azure Service Management and Azure Resource Manager) are supported without the need to explicitly switch between Azure PowerShell modes. Previously, it was necessary to execute a special command, *Switch-AzureMode*, to switch between Azure Service Management and Azure Resource Manager modes.

The coexistence of Azure Service Management and Azure Resource Manager (via PowerShell) is enabled via distinct PowerShell modules that support specific management operations. Azure Service Management features are enabled via the Azure module, while Azure Resource Manager features are enabled via one or more AzureRM modules, as can be seen in Figure 8-11.



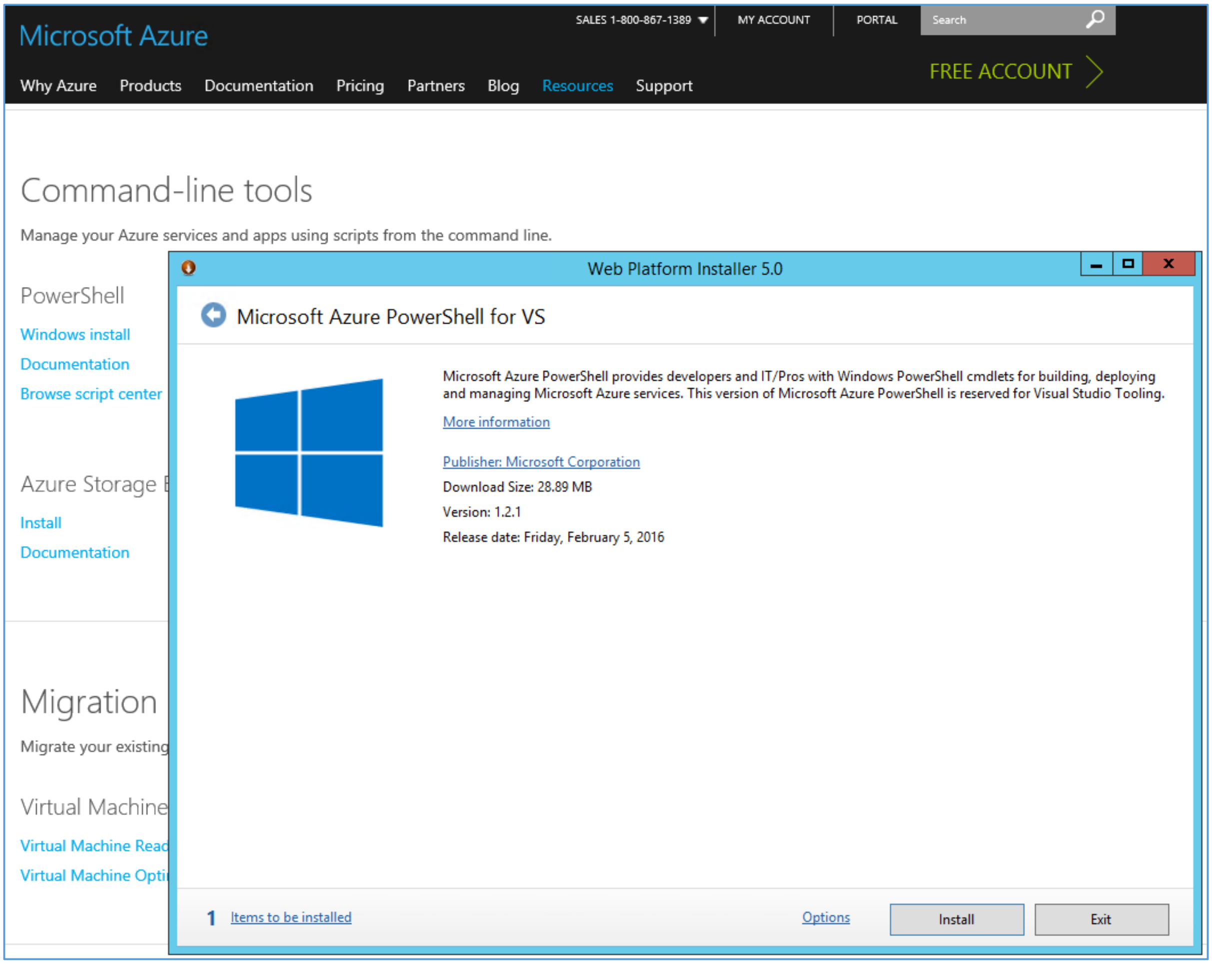
```
PS C:\Users\mcollier> Get-Module -ListAvailable | Where-Object {$_.Name -like "Azure*"}


    Directory: C:\Program Files (x86)\Microsoft SDKs\Azure\PowerShell\ResourceManager\AzureResourceManager


ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   1.0.4      Azure.Storage                       {Get-AzureStorageBlob, Get-AzureStorageBlobContent, Get-Az...
Manifest   1.0.4      AzureRM.ApiManagement               {Add-AzureRmApiManagementRegion, Get-AzureRmApiManagementS...
Manifest   1.0.4      AzureRM.Automation                  {Get-AzureRmAutomationJobOutputRecord, Import-AzureRmAutom...
Binary     0.9.1      AzureRm.AzureStackAdmin             {Get-AzureRMManagedLocation, New-AzureRMManagedLocation, R...
Manifest   0.9.2      AzureRM.AzureStackStorage           {Add-ACSFarm, Get-ACSEvent, Get-ACSEventQuery, Get-ACSFarm...
Manifest   1.0.4      AzureRM.Backup                      {Backup-AzureRmBackupItem, Enable-AzureRmBackupContainerRe...
Manifest   1.0.4      AzureRM.Batch                       {Remove-AzureRmBatchAccount, Get-AzureRmBatchAccount, Get-...
Manifest   1.2.2      AzureRM.Compute                     {Remove-AzureRmAvailabilitySet, Get-AzureRmAvailabilitySet...
Manifest   1.0.4      AzureRM.DataFactories               {Remove-AzureRmDataFactory, Get-AzureRmDataFactoryRun, Get...
Manifest   1.0.4      AzureRM.DataLakeAnalytics           {Remove-AzureRmDataLakeAnalyticsCatalogSecret, Set-AzureRm...
Manifest   1.0.4      AzureRM.DataLakeStore               {Add-AzureRmDataLakeStoreItemContent, Export-AzureRmDataLa...
Manifest   1.0.4      AzureRM.Dns                         {Get-AzureRmDnsRecordSet, Remove-AzureRmDnsRecordSet, Set-...
Manifest   1.0.5      AzureRM.HDInsight                   {Get-AzureRmHDInsightJob, New-AzureRmHDInsightSqoopJobDefi...
Manifest   1.0.4      AzureRM.Insights                    {Add-AlertRule, Get-AlertHistory, Get-AlertRule, Remove-Al...
Manifest   1.1.3      AzureRM.KeyVault                    {Get-AzureRmKeyVault, New-AzureRmKeyVault, Remove-AzureRmK...
Manifest   1.0.4      AzureRM.Network                     {Add-AzureRmApplicationGatewayBackendAddressPool, Get-Azur...
Manifest   1.0.4      AzureRM.NotificationHubs            {Get-AzureRmNotificationHubsNamespaceAuthorizationRules, G...
Manifest   1.0.4      AzureRM.OperationalInsights         {Get-AzureRmOperationalInsightsSavedSearch, Get-AzureRmOpe...
Manifest   1.0.4      AzureRM.Profile                     {Enable-AzureRmDataCollection, Disable-AzureRmDataCollecti...
Manifest   1.0.5      AzureRM.RecoveryServices            {Get-AzureRmRecoveryServicesVault, Get-AzureRmRecoveryServ...
Manifest   1.1.2      AzureRM.RedisCache                  {Remove-AzureRmRedisCacheDiagnostics, Set-AzureRmRedisCach...
Manifest   1.0.4      AzureRM.Resources                   {Get-AzureRmADApplication, Get-AzureRmADGroupMember, Get-A...
Manifest   1.1.3      AzureRM.SiteRecovery                {Stop-AzureRmSiteRecoveryJob, Get-AzureRmSiteRecoveryNetwo...
Manifest   1.0.4      AzureRM.Sql                         {Get-AzureRmSqlDatabaseThreatDetectionPolicy, Set-AzureRmS...
Manifest   1.0.4      AzureRM.Storage                     {Get-AzureRmStorageAccount, Get-AzureRmStorageAccountKey, ...
Manifest   1.0.4      AzureRM.StreamAnalytics             {Get-AzureRmStreamAnalyticsFunction, Get-AzureRmStreamAnal...
Manifest   1.0.4      AzureRM.Tags                        {Remove-AzureRmTag, Get-AzureRmTag, New-AzureRmTag}
Manifest   1.0.4      AzureRM.TrafficManager              {Disable-AzureRmTrafficManagerEndpoint, Enable-AzureRmTraf...
Manifest   1.0.4      AzureRM.UsageAggregates             Get-UsageAggregates
Manifest   1.0.4      AzureRM.Websites                    {Get-AzureRmAppServicePlanMetrics, Get-AzureRmWebAppCertif...


    Directory: C:\Program Files (x86)\Microsoft SDKs\Azure\PowerShell\ServiceManagement


ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   1.0.4      Azure                               {Disable-AzureServiceProjectRemoteDesktop, Enable-AzureSer...


PS C:\Users\mcollier> _
```

Figure 8-11 Listing of the available Azure modules in PowerShell.

You may notice in Figure 8-11 that the version number of the modules may be different among the Azure Resource Manager modules. This is intentional. The various Azure feature product teams release the modules independently. Releasing individually allows the teams to update at a much faster pace—enabling them to release new features or fix defects without waiting for a full release of the Azure PowerShell cmdlet package/installer. The proceeding sections will discuss some of the pros and cons associated with the two installation mediums, which are related to the versioning and release cadence of the various modules.

Installing Azure PowerShell from Web PI

Earlier in this chapter, you saw the Azure Downloads page at http://azure.microsoft.com/downloads/, from which you were able to download the Azure SDK for .NET (Visual Studio). You can initiate the Web PI installation from this same location, as seen in Figure 8-12.



Manage your Azure services and apps using scripts from the command line.

Microsoft Azure PowerShell provides developers and IT/Pros with Windows PowerShell cmdlets for building, deploying and managing Microsoft Azure services. This version of Microsoft Azure PowerShell is reserved for Visual Studio Tooling.

Download Size: 28.89 MB
Version: 1.2.1
Release date: Friday, February 5, 2016

Figure 8-12 Azure PowerShell installation via Web PI.

One of the advantages of installing the Azure PowerShell cmdlets via the Web PI process is you are guaranteed that the cmdlets (as part of the various modules) will work well together. It should be noted that if you previously installed the Azure modules via the PowerShell Gallery, the Web PI–based installer will automatically remove them. This is done to ensure the modules are at a known version related to the installer.

Installing Azure PowerShell from the PowerShell Gallery

Starting with Azure PowerShell 1.0, it is possible to install various modules from the PowerShell Gallery. One of the reasons to do so would be to get the latest, or a specific, version of a module. As mentioned previously, the modules installed as part of the Web PI installation process are of a set, known version. By installing via the PowerShell Gallery, it is possible to obtain a specific version—perhaps a module version that is not yet available in the Web PI–based installer.

> **Note** To use the PowerShell Gallery, you will need the latest version of the PowerShellGet module, which is available in Windows 10, in Windows Management Framework (WMF) 5.0, or via an MSI-based installer for PowerShell 3 and PowerShell 4. For more information, please reference https://www.powershellgallery.com/.

To install the Azure modules, you will need to execute a series of PowerShell commands, as seen below.

Installing the Azure PowerShell modules

```
# Install the Azure Resource Manager modules from the PowerShell Gallery

Install-Module AzureRM

Install-AzureRM



# Install the Azure Service Management module from the PowerShell Gallery

Install-Module Azure



# Import AzureRM modules for the given version manifest in the AzureRM module

Import-AzureRM



# Import Azure Service Management module

Import-Module Azure
```

> **Tip** Be sure to read the detailed documentation available in the Azure PowerShell project repository available on GitHub at https://github.com/Azure/azure-powershell. The page contains a wealth of information on getting started with the Azure cmdlets, various features, and much more.

Connecting to Azure

To connect to Azure, you must have an Azure subscription. To sign up for a free trial (at http://azure.microsoft.com), you need a Microsoft account or a work or school (formerly organizational) account.

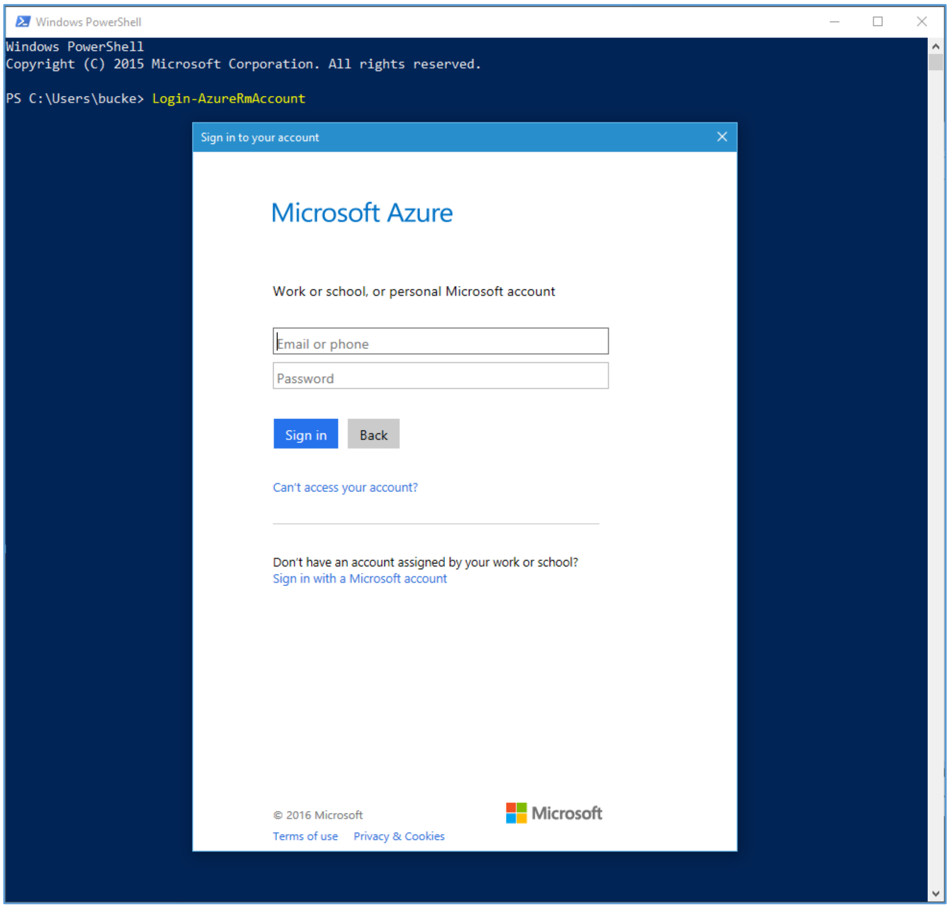
After installing the Azure PowerShell cmdlets, you will connect PowerShell to your Microsoft Azure subscription(s) by using either a Microsoft account (such as hotmail.com, outlook.com, and so on), or a work or school account (formerly organizational account; for example, contoso.com). Azure Resource Manager does not support client certificate authentication.

Tip Although Azure Resource Manager does not support client certificate authentication, it does support certificate authentication for a service principle (for example, for use with an automated process or service). Please reference the article at https://azure.microsoft.com/documentation/articles/resource-group-authenticate-service-principal/ for detailed instructions.

You can use your Microsoft account or a work or school account to access your Microsoft Azure subscription(s). For this method of connecting, Azure Active Directory (Azure AD) is used. This is the preferred method in many cases because it is easier to manage access to a subscription (especially for a shared subscription that many people use). Additionally, using the Azure AD approach is required to work with the Azure Resource Manager API.

To connect to your Microsoft Azure subscription, follow these steps:

1. Open a Windows PowerShell console or the Windows PowerShell ISE.
2. Enter the *Login-AzureRmAccount* command.
3. A dialog box prompts you for your email address and password. Your credentials are authenticated via Azure AD, and your subscription details are saved in your roaming user profile. An access token is also retrieved that allows PowerShell to access your Azure resources. This token is not permanent and will expire, at which point you will need to reauthenticate by using *Login-AzureRmAccount*.



```
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\bucke> Login-AzureRmAccount
```

Windows PowerShell
Sign in to your account
Microsoft Azure
Work or school, or personal Microsoft account
Email or phone
Password
Sign in
Back
Can't access your account?
Don't have an account assigned by your work or school?
Sign in with a Microsoft account
© 2016 Microsoft
Microsoft
Terms of use
Privacy & Cookies

To view all available subscriptions for the specified account, use the *Get-AzureRmSubscription* cmdlet. To change to a different subscription, use the *Select-AzureRmSubscription* cmdlet.

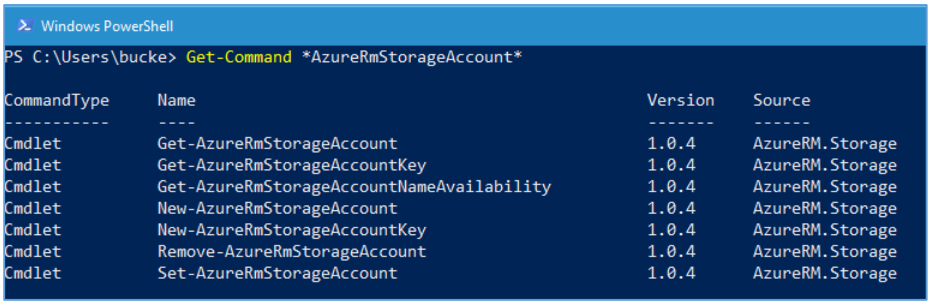
If you are using PowerShell for an automation script and you are not using a service principal (see the previous note in this chapter), you will want to avoid the pop-up window. In this case, use the *Credential* parameter to provide your credentials, as seen in the following example. The *Credential* parameter works for work or school accounts only.

```
$userName = "<your work or school account user name>"
$securePassword = ConvertTo-SecureString -String "<your work or school account password>" -
AsPlainText -Force

$cred = New-Object System.Management.Automation.PSCredential($userName, $securePassword)

Login-AzureRmAccount -Credential $cred
```

> **Note** For additional details on configuring the Azure PowerShell cmdlets, please refer to the Microsoft documentation at https://azure.microsoft.com/documentation/articles/powershell-install-configure/.

Usage

There are many cmdlets available for working with the various Azure resources. To get a list of all the cmdlets, enter the command *Get-Help azure*, although it would be helpful to filter by a specific module (such as Azure, AzureRM.Compute, AzureRM.Network, and so on) or cmdlet name because there is a large number of Azure cmdlets. You can get help for specific cmdlets by executing the command *Get-Help* followed by the name of the desired cmdlet, such as *Get-Help New-AzureRmResourceGroup -Full* (passing *-Full* is not necessary, but it is often helpful). It is also often helpful to get a list of all the possible cmdlets for a specific resource type, and the PowerShell command *Get-Command* can be useful, as seen in Figure 8-13.



```
PS C:\Users\bucke> Get-Command *AzureRmStorageAccount*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Get-AzureRmStorageAccount                          1.0.4      AzureRM.Storage
Cmdlet          Get-AzureRmStorageAccountKey                       1.0.4      AzureRM.Storage
Cmdlet          Get-AzureRmStorageAccountNameAvailability          1.0.4      AzureRM.Storage
Cmdlet          New-AzureRmStorageAccount                          1.0.4      AzureRM.Storage
Cmdlet          New-AzureRmStorageAccountKey                       1.0.4      AzureRM.Storage
Cmdlet          Remove-AzureRmStorageAccount                       1.0.4      AzureRM.Storage
Cmdlet          Set-AzureRmStorageAccount                          1.0.4      AzureRM.Storage
```

Figure 8-13 Executing *Get-Command* against a specific resource type.

Creating a new Azure Web App is a common task that can be simplified by using the Azure PowerShell cmdlets, as seen in the following example.

Creating a new Azure App Service Plan and Azure Web App

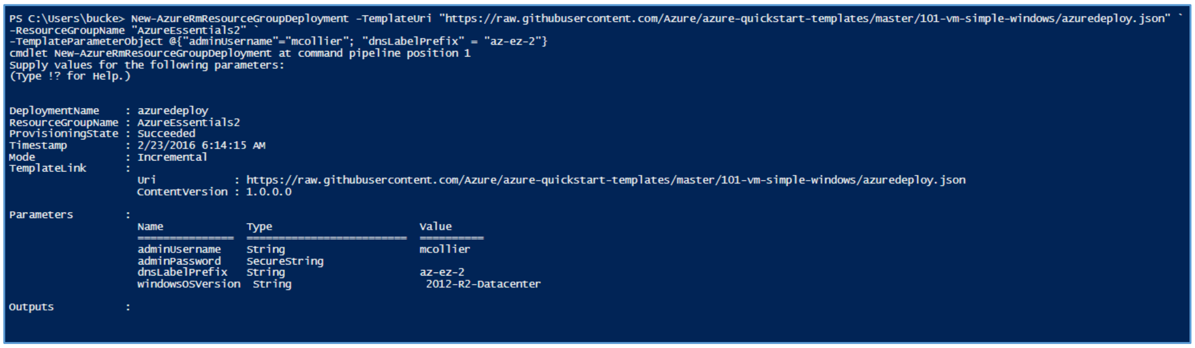
```
$resourceGroupName = "AzureEssentials2"

$location = "eastus"

$appServicePlanName = "azure-essential-plan"

New-AzureRmAppServicePlan -Location $location -ResourceGroupName $resourceGroupName -Name
$appServicePlanName -Tier Free -WorkerSize Small

New-AzureRmWebApp -Location $location -ResourceGroupName $resourceGroupName -Name "azure-
essential-web" -AppServicePlan $appServicePlanName
```

For a more complex deployment or one requiring detailed configuration, an Azure Resource Manager (ARM) template can be used. You can find a wealth of templates at https://github.com/Azure/azure-quickstart-templates (and searchable at https://azure.microsoft.com/documentation/templates/). These templates also serve as a great example for starting to create your own templates. The following example demonstrates one approach for deploying an Azure VM using one of the templates available on GitHub.

Creating a new VM using an ARM template

```
New-AzureRmResourceGroupDeployment -TemplateUri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json" `

-ResourceGroupName "AzureEssentials2" `

-TemplateParameterObject @{"adminUsername"="mcollier"; "dnsLabelPrefix" = "az-ez-2"}
```

When finished, you should see output similar to that in Figure 8-14.



```
PS C:\Users\bucke> New-AzureRmResourceGroupDeployment -TemplateUri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json" `
-ResourceGroupName "AzureEssentials2" `
-TemplateParameterObject @{"adminUsername"="mcollier"; "dnsLabelPrefix" = "az-ez-2"}
cmdlet New-AzureRmResourceGroupDeployment at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)


DeploymentName    : azuredeploy
ResourceGroupName : AzureEssentials2
ProvisioningState : Succeeded
Timestamp         : 2/23/2016 6:14:15 AM
Mode              : Incremental
TemplateLink      :
                    Uri            : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json
                    ContentVersion : 1.0.0.0

Parameters        :
                    Name             Type                       Value
                    ===============  =========================  ==========
                    adminUsername    String                     mcollier
                    adminPassword    SecureString
                    dnsLabelPrefix   String                     az-ez-2
                    windowsOSVersion  String                     2012-R2-Datacenter

Outputs           :
```

Figure 8-14   Creating a new Azure VM by using an ARM template.

Tip Set the PowerShell preference variable *$DebugPreference* to *Continue* to see the debug statements when executing an Azure PowerShell cmdlet, including the HTTP Request and HTTP Response. This can aid in debugging efforts.

Tip For more information about and examples of using Azure PowerShell with Azure Resource Manager, please reference https://azure.microsoft.com/documentation/articles/powershell-azure-resource-manager/. The Azure Script Center at https://azure.microsoft.com/documentation/scripts/ can also be helpful in learning more about how to work with Azure PowerShell for a variety of Azure services.

Cross-platform command-line interface

For Windows users, the PowerShell cmdlets discussed in the previous section are your best option for automating tasks and working from the command line, especially if you are scripting the provisioning of several Azure resources. However, for mixed environments, the Azure cross-platform command-line interface provides a consistent experience for Linux, Mac OS, and Windows users.

Just like the Azure PowerShell project repository, the repository for the Azure CLI is available on GitHub, at https://github.com/Azure/azure-xplat-cli. Please refer to this GitHub page for additional installation instructions.

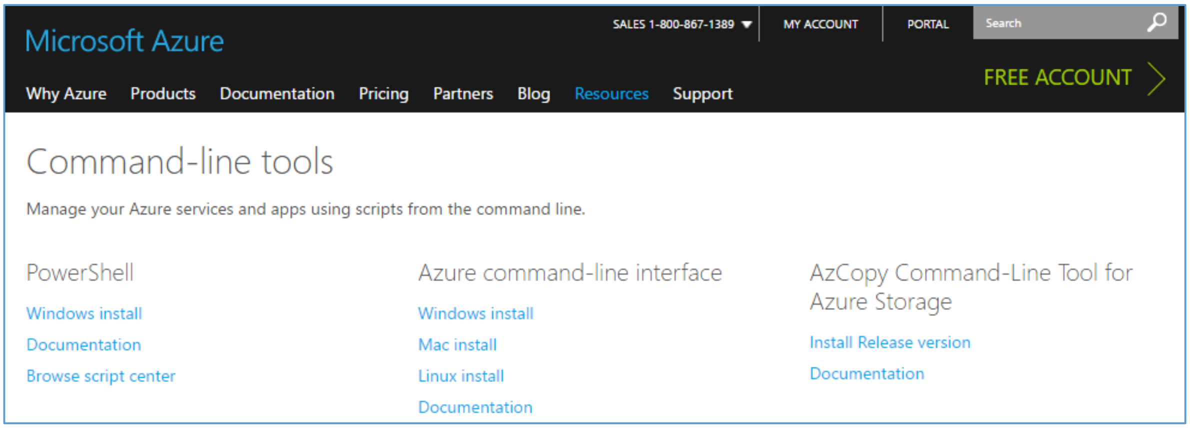
Installation

The Azure CLI is a Node.js application implemented by using the Azure SDK for Node.js. Therefore, you will need to ensure Node.js is installed on your system.

> **See Also** Additional installation instructions can be found at https://azure.microsoft.com/documentation/articles/xplat-cli-install/.

Installing on Windows

If you do not have Node.js installed on your system, you can use Windows install from the Microsoft Azure Downloads page (http://azure.microsoft.com/downloads/), as seen in Figure 8-15. Web PI will handle installing Node.js and the Azure CLI.



Command-line tools

Manage your Azure services and apps using scripts from the command line.

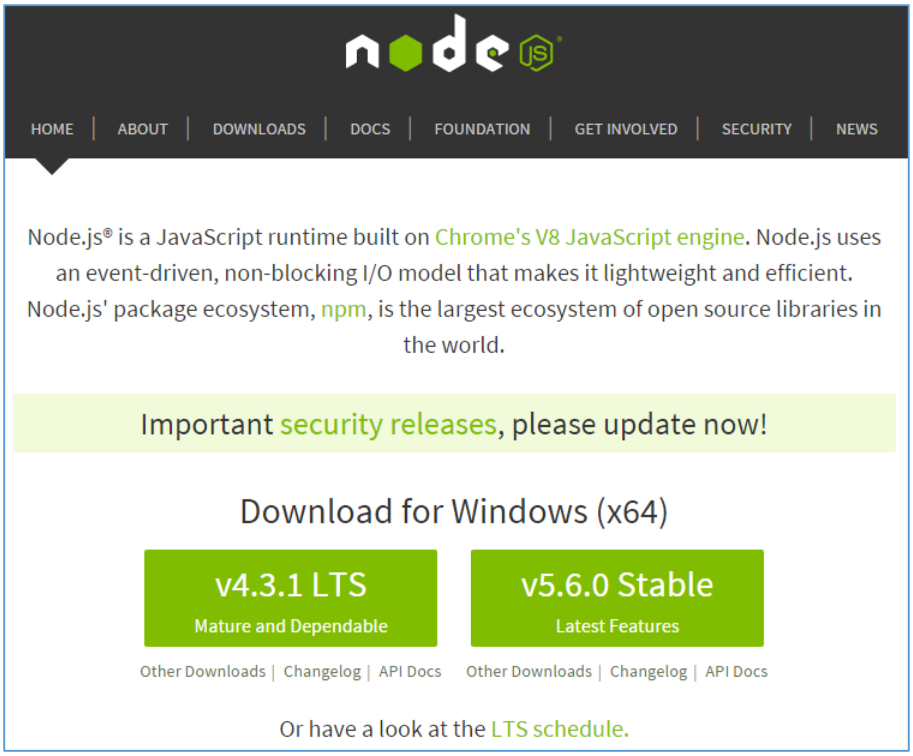
PowerShell
- Windows install
- Documentation
- Browse script center

Azure command-line interface
- Windows install
- Mac install
- Linux install
- Documentation

AzCopy Command-Line Tool for Azure Storage
- Install Release version
- Documentation

Figure 8-15 Install Azure command-line interface using Windows install.

Alternatively, you can install Node.js from the http://nodejs.org site. Just click Install, as shown in Figure 8-16, to begin the process of installing the latest version.



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Important security releases, please update now!

Download for Windows (x64)

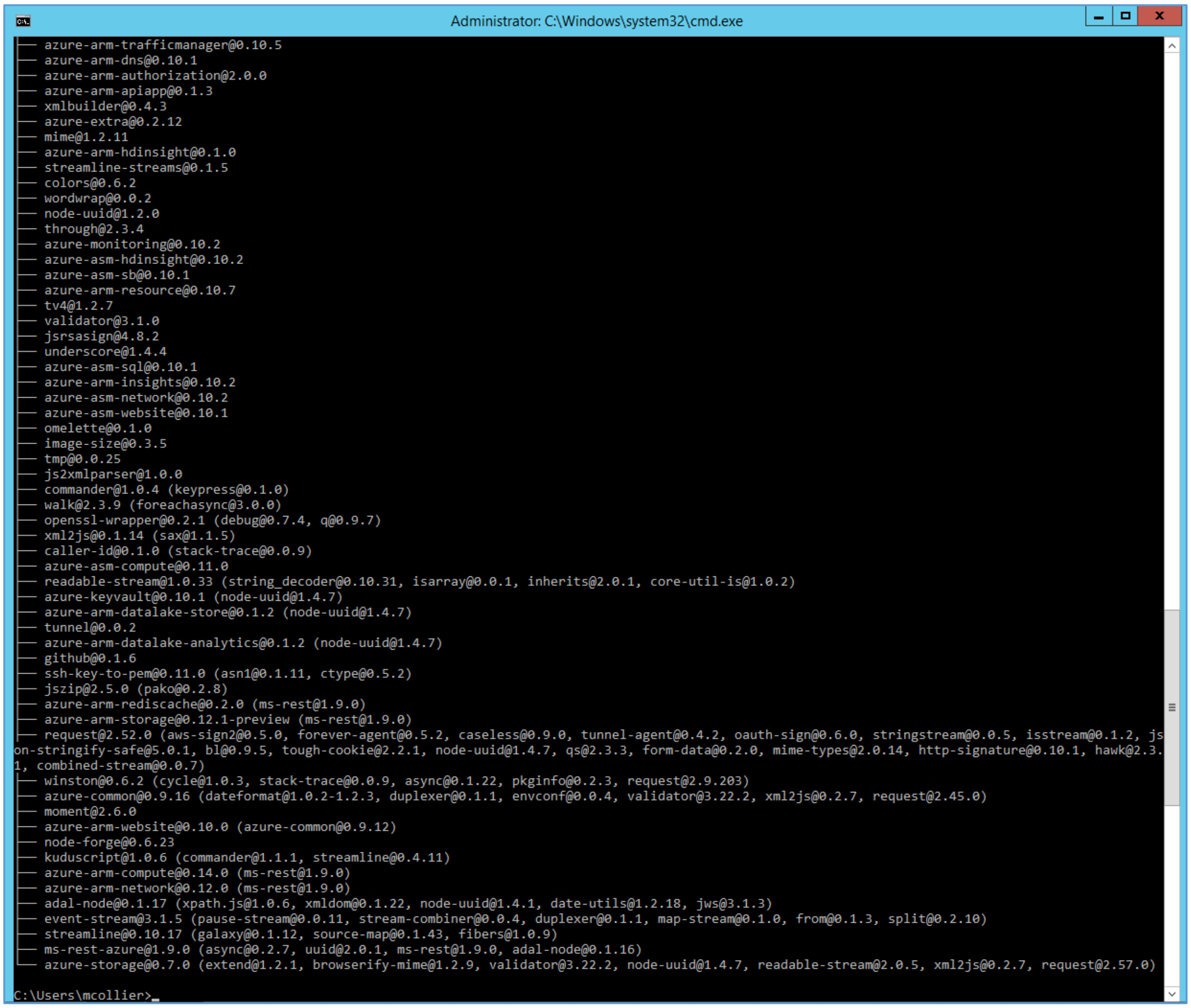
Or have a look at the LTS schedule.

Figure 8-16 Install Node.js from the http://nodejs.org site.

Once Node.js is installed on your system, you can use npm to install the Azure CLI.

```
npm install azure-cli -g
```

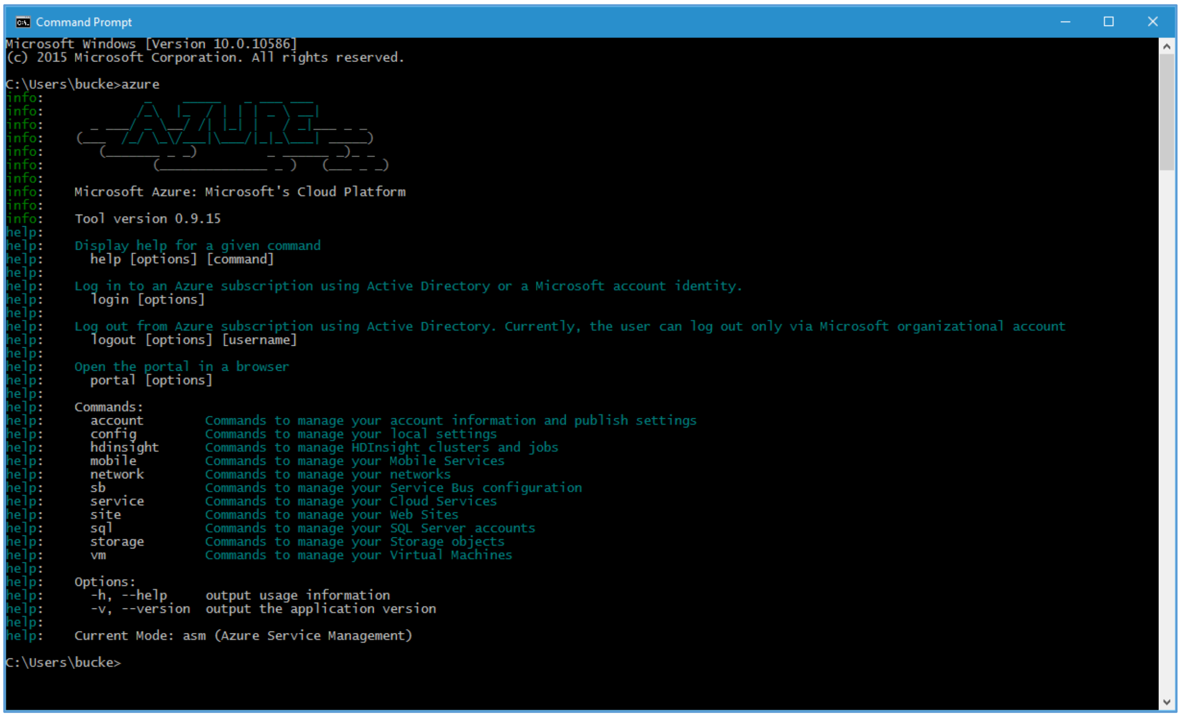
All the necessary dependencies will be installed by npm. When finished, you should see something similar to the results in Figure 8-17.



```
├── azure-arm-trafficmanager@0.10.5
├── azure-arm-dns@0.10.1
├── azure-arm-authorization@2.0.0
├── azure-arm-apiapp@0.1.3
├── xmlbuilder@0.4.3
├── azure-extra@0.2.12
├── mime@1.2.11
├── azure-arm-hdinsight@0.1.0
├── streamline-streams@0.1.5
├── colors@0.6.2
├── wordwrap@0.0.2
├── node-uuid@1.2.0
├── through@2.3.4
├── azure-monitoring@0.10.2
├── azure-asm-hdinsight@0.10.2
├── azure-asm-sb@0.10.1
├── azure-arm-resource@0.10.7
├── tv4@1.2.7
├── validator@3.1.0
├── jsrsasign@4.8.2
├── underscore@1.4.4
├── azure-asm-sql@0.10.1
├── azure-arm-insights@0.10.2
├── azure-asm-network@0.10.2
├── azure-asm-website@0.10.1
├── omelette@0.1.0
├── image-size@0.3.5
├── tmp@0.0.25
├── js2xmlparser@1.0.0
├── commander@1.0.4 (keypress@0.1.0)
├── walk@2.3.9 (foreachasync@3.0.0)
├── openssl-wrapper@0.2.1 (debug@0.7.4, q@0.9.7)
├── xml2js@0.1.14 (sax@1.1.5)
├── caller-id@0.1.0 (stack-trace@0.0.9)
├── azure-asm-compute@0.11.0
├── readable-stream@1.0.33 (string_decoder@0.10.31, isarray@0.0.1, inherits@2.0.1, core-util-is@1.0.2)
├── azure-keyvault@0.10.1 (node-uuid@1.4.7)
├── azure-arm-datalake-store@0.1.2 (node-uuid@1.4.7)
├── tunnel@0.0.2
├── azure-arm-datalake-analytics@0.1.2 (node-uuid@1.4.7)
├── github@0.1.6
├── ssh-key-to-pem@0.11.0 (asn1@0.1.11, ctype@0.5.2)
├── jszip@2.5.0 (pako@0.2.8)
├── azure-arm-rediscache@0.2.0 (ms-rest@1.9.0)
├── azure-arm-storage@0.12.1-preview (ms-rest@1.9.0)
├── request@2.52.0 (aws-sign2@0.5.0, forever-agent@0.5.2, caseless@0.9.0, tunnel-agent@0.4.2, oauth-sign@0.6.0, stringstream@0.0.5, isstream@0.1.2, js
on-stringify-safe@5.0.1, bl@0.9.5, tough-cookie@2.2.1, node-uuid@1.4.7, qs@2.3.3, form-data@0.2.0, mime-types@2.0.14, http-signature@0.10.1, hawk@2.3.
1, combined-stream@0.0.7)
├── winston@0.6.2 (cycle@1.0.3, stack-trace@0.0.9, async@0.1.22, pkginfo@0.2.3, request@2.9.203)
├── azure-common@0.9.16 (dateformat@1.0.2-1.2.3, duplexer@0.1.1, envconf@0.0.4, validator@3.22.2, xml2js@0.2.7, request@2.45.0)
├── moment@2.6.0
├── azure-arm-website@0.10.0 (azure-common@0.9.12)
├── node-forge@0.6.23
├── kuduscript@1.0.6 (commander@1.1.1, streamline@0.4.11)
├── azure-arm-compute@0.14.0 (ms-rest@1.9.0)
├── azure-arm-network@0.12.0 (ms-rest@1.9.0)
├── adal-node@0.1.17 (xpath.js@1.0.6, xmldom@0.1.22, node-uuid@1.4.1, date-utils@1.2.18, jws@3.1.3)
├── event-stream@3.1.5 (pause-stream@0.0.11, stream-combiner@0.0.4, duplexer@0.1.1, map-stream@0.1.0, from@0.1.3, split@0.2.10)
├── streamline@0.10.17 (galaxy@0.1.12, source-map@0.1.43, fibers@1.0.9)
├── ms-rest-azure@1.9.0 (async@0.2.7, uuid@2.0.1, ms-rest@1.9.0, adal-node@0.1.16)
└── azure-storage@0.7.0 (extend@1.2.1, browserify-mime@1.2.9, validator@3.22.2, node-uuid@1.4.7, readable-stream@2.0.5, xml2js@0.2.7, request@2.57.0)

C:\Users\mcollier>_
```

Figure 8-17 Installing the Azure CLI.

When the installation is finished, you can launch the Azure CLI by entering **azure** at the command prompt. At this point, the Azure CLI will launch (although it might take a few seconds) and you will see AZURE in ASCII art, along with the version number and some basic help information, as shown in Figure 8-18.



```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\bucke>azure
info:             _    _____   _ ___ ___
info:            /_\  |_  / | | | _ \ __|
info:      _ ___/ _ \__/ /| |_| |   / _|___ _ _
info:    (___  /_/ \_\/___|\___/|_|_\___| _____)
info:       (_______ _ _)         _ ______ _)_ _
info:              (______________ _ )   (___ _ _)
info:
info:    Microsoft Azure: Microsoft's Cloud Platform
info:
info:    Tool version 0.9.15
help:
help:    Display help for a given command
help:      help [options] [command]
help:
help:    Log in to an Azure subscription using Active Directory or a Microsoft account identity.
help:      login [options]
help:
help:    Log out from Azure subscription using Active Directory. Currently, the user can log out only via Microsoft organizational account
help:      logout [options] [username]
help:
help:    Open the portal in a browser
help:      portal [options]
help:
help:    Commands:
help:      account        Commands to manage your account information and publish settings
help:      config         Commands to manage your local settings
help:      hdinsight      Commands to manage HDInsight clusters and jobs
help:      mobile         Commands to manage your Mobile Services
help:      network        Commands to manage your networks
help:      sb             Commands to manage your Service Bus configuration
help:      service        Commands to manage your Cloud Services
help:      site           Commands to manage your Web Sites
help:      sql            Commands to manage your SQL Server accounts
help:      storage        Commands to manage your Storage objects
help:      vm             Commands to manage your Virtual Machines
help:
help:    Options:
help:      -h, --help     output usage information
help:      -v, --version  output the application version
help:
help:    Current Mode: asm (Azure Service Management)

C:\Users\bucke>
```

Figure 8-18   Starting the Azure CLI on Windows.

Installing on Linux

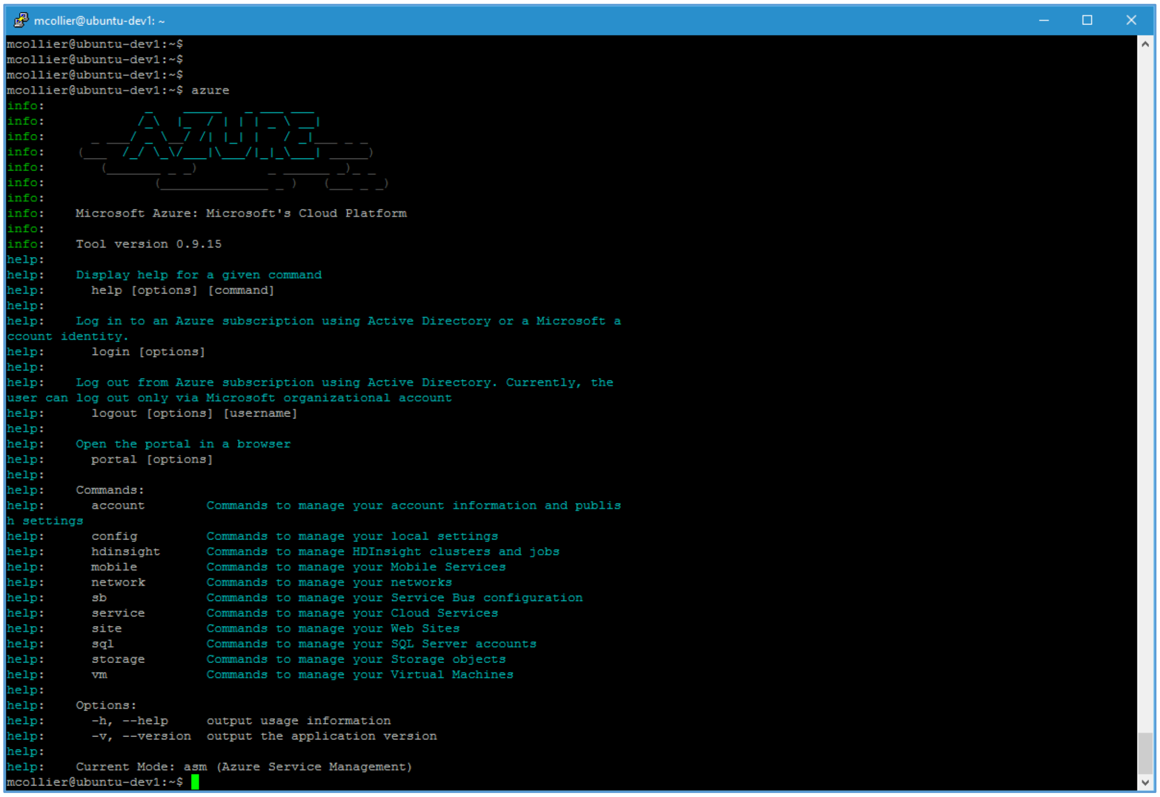
Installing the Azure CLI on a Linux system is functionally similar to installing it on a Windows system. You will need to ensure you have Node.js installed. You can use the appropriate package manager for your Linux distribution (see https://nodejs.org/en/download/package-manager/).

For example, on Ubuntu 15.10, execute the following commands:

Installing the Azure CLI on Ubuntu

```
sudo apt-get install nodejs-legacy

sudo apt-get install npm

sudo npm install -g azure-cli
```

When finished, execute the *azure* command, as shown in Figure 8-19, which will yield the same results as executing the command on a Windows system.



```
mcollier@ubuntu-dev1:~$
mcollier@ubuntu-dev1:~$
mcollier@ubuntu-dev1:~$
mcollier@ubuntu-dev1:~$ azure
info:             _    _____   _ ___ ___
info:            /_\  |_  / | | | _ \ __|
info:      _ ___/ _ \__/ /| |_| |   / _|___ _ _
info:    (___  /_/ \_\/___|\___/|_|_\___| _____)
info:       (_______ _ _)         _ ______ _)_ _
info:              (______________ _ )   (___ _ _)
info:
info:    Microsoft Azure: Microsoft's Cloud Platform
info:
info:    Tool version 0.9.15
help:
help:    Display help for a given command
help:      help [options] [command]
help:
help:    Log in to an Azure subscription using Active Directory or a Microsoft a
ccount identity.
help:      login [options]
help:
help:    Log out from Azure subscription using Active Directory. Currently, the
user can log out only via Microsoft organizational account
help:      logout [options] [username]
help:
help:    Open the portal in a browser
help:      portal [options]
help:
help:    Commands:
help:      account        Commands to manage your account information and publis
h settings
help:      config         Commands to manage your local settings
help:      hdinsight      Commands to manage HDInsight clusters and jobs
help:      mobile         Commands to manage your Mobile Services
help:      network        Commands to manage your networks
help:      sb             Commands to manage your Service Bus configuration
help:      service        Commands to manage your Cloud Services
help:      site           Commands to manage your Web Sites
help:      sql            Commands to manage your SQL Server accounts
help:      storage        Commands to manage your Storage objects
help:      vm             Commands to manage your Virtual Machines
help:
help:    Options:
help:      -h, --help     output usage information
help:      -v, --version  output the application version
help:
help:    Current Mode: asm (Azure Service Management)
mcollier@ubuntu-dev1:~$
```

Figure 8-19   Starting the Azure CLI on Linux (Ubuntu).

Connecting to Azure

Before you can begin to use the Azure CLI, you will need to connect to your Microsoft Azure subscription(s). The options for doing so with the Azure CLI are similar to doing so with the Azure PowerShell cmdlets. There are two options: you can use a management certificate, or you can use a work or school account or a Microsoft account identity.

Connect using a management certificate

If you are working with the Azure Service Management classic interface, you will need to connect using a management certificate. The easiest way to get a management certificate is to let the Azure classic portal create one. The management certificate comes in the form of a *.publishsettings* file.

1. If you do not already have a *.publishsettings* file, execute the following command to download one. This will open the Azure classic portal and prompt you to sign in.

    ```
    azure account download
    ```

2. Import the *.publishsettings* file by executing the following command (after changing the path to use the file on your system):

    ```
    azure account import [path to your .publishsettings file]
    ```

Tip The *.publishsettings* file contains sensitive information about your Microsoft Azure subscriptions, including the subscription ID and management certificate. Unauthorized users should not access the file. It is recommended that you delete the file after importing.

Connect using a work or school account

You can also use a work or school account (formerly organizational account) via Azure AD to connect to your Microsoft Azure subscription(s). Starting with Azure CLI versions 0.9.10 and above, you can use either a work or school account or a Microsoft account identity. Azure CLI versions prior to 0.9.10 work only with a work or school account.

The process for authenticating varies depending on whether you are using a work or school account or using a Microsoft account—one is an interactive approach, the other is more simplistic (non-interactive). You can connect using an interactive approach using either a work or school account (especially one that requires multi-factor authentication) or using a Microsoft account. The interactive approach is as follows:

1. Execute the following command to log in:

   ```
   azure login
   ```

2. You will be prompted to complete the authentication process by opening a web browser to the page at http://aka.ms/devicelogin and entering a specific code. You will also be prompted to authenticate using your username and password for the desired account identity (for example, your work or school account or your Microsoft account).
3. When the process is complete, you should see a result similar to the following:

Output of the interactive login approach

```
C:\Users\bucke>azure login

info:    Executing command login

|info:    To sign in, use a web browser to open the page https://aka.ms/devicelogin. Enter the
code xxxxxxx to authenticate.

/info:    Added subscription Visual Studio Ultimate with MSDN

info:    Added subscription Collier - Personal - Pay-as-you-Go

+

info:    login command OK
```

The non-interactive approach works only with a work or school account and is as follows:

1. Execute the following command to log in using a work or school account:

   ```
   azure login -u <username>
   ```

Output of the non-interactive login approach

```
C:\Users\bucke>azure login -u admin@mcollier.onmicrosoft.com
```

```
info:    Executing command login

Password: ********

/info:    Added subscription Visual Studio Ultimate with MSDN

info:    Setting subscription "Visual Studio Ultimate with MSDN" as default

+

info:    login command OK
```

2. To log out, execute the following command:

   ```
   azure logout -u <username>
   ```

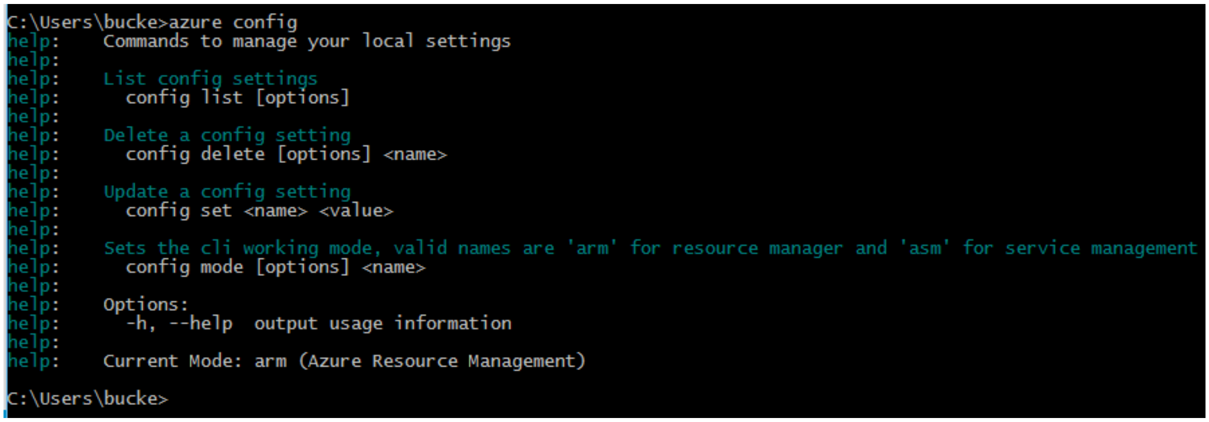
See Also For more information on connecting to Azure by using the Azure CLI, please reference https://azure.microsoft.com/documentation/articles/xplat-cli-connect/.

Usage

The Azure CLI supports working with both Azure Service Management (classic) resources, and Azure Resource Manager resources. You will need to switch between Azure Service Management (asm) and Azure Resource Manager (arm) modes to work with the necessary resources. To switch, execute the following command:

```
azure config mode <name>
```

You can tell which mode you're currently in by executing the *azure* command or any other topic-level command, such as *azure config*, as seen in Figure 8-20.

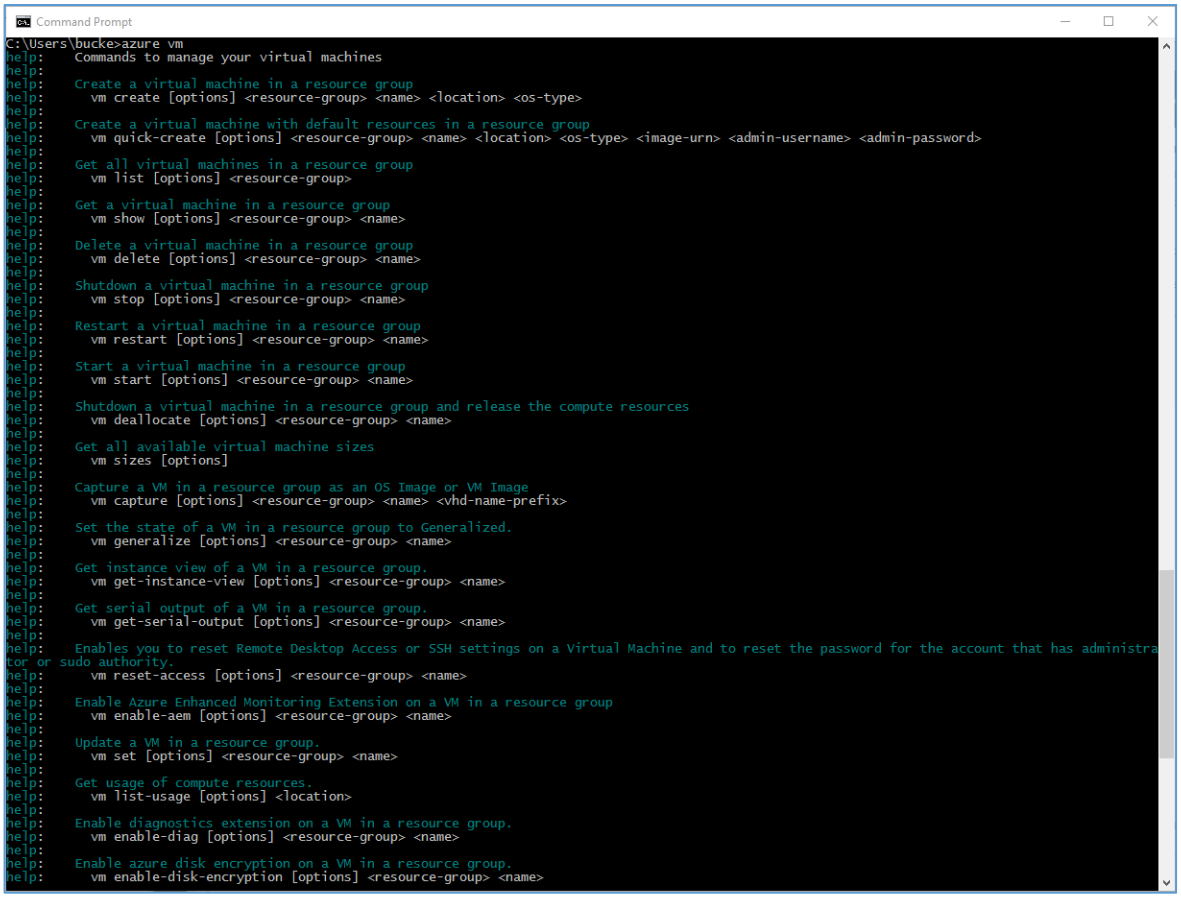


```
C:\Users\bucke>azure config
help:    Commands to manage your local settings
help:
help:    List config settings
help:      config list [options]
help:
help:    Delete a config setting
help:      config delete [options] <name>
help:
help:    Update a config setting
help:      config set <name> <value>
help:
help:    Sets the cli working mode, valid names are 'arm' for resource manager and 'asm' for service management
help:      config mode [options] <name>
help:
help:    Options:
help:      -h, --help  output usage information
help:
help:    Current Mode: arm (Azure Resource Management)

C:\Users\bucke>
```

Figure 8-20 The Azure CLI configuration commands.

Using the Azure CLI follows an intuitive *azure [topic] [verb] [options]* syntax; for example, *azure account list or azure vm start web-fe-1*.

Entering just the *azure* command will show you the top-level list of topics available (account, vm, service, and so on). To learn more about a specific topic, just enter **azure** followed by the topic. For example, *azure vm* will show the commands available for working with Azure Virtual Machines, as shown in Figure 8-21.



```
C:\Users\bucke>azure vm
help:    Commands to manage your virtual machines
help:
help:    Create a virtual machine in a resource group
help:      vm create [options] <resource-group> <name> <location> <os-type>
help:
help:    Create a virtual machine with default resources in a resource group
help:      vm quick-create [options] <resource-group> <name> <location> <os-type> <image-urn> <admin-username> <admin-password>
help:
help:    Get all virtual machines in a resource group
help:      vm list [options] <resource-group>
help:
help:    Get a virtual machine in a resource group
help:      vm show [options] <resource-group> <name>
help:
help:    Delete a virtual machine in a resource group
help:      vm delete [options] <resource-group> <name>
help:
help:    Shutdown a virtual machine in a resource group
help:      vm stop [options] <resource-group> <name>
help:
help:    Restart a virtual machine in a resource group
help:      vm restart [options] <resource-group> <name>
help:
help:    Start a virtual machine in a resource group
help:      vm start [options] <resource-group> <name>
help:
help:    Shutdown a virtual machine in a resource group and release the compute resources
help:      vm deallocate [options] <resource-group> <name>
help:
help:    Get all available virtual machine sizes
help:      vm sizes [options]
help:
help:    Capture a VM in a resource group as an OS Image or VM Image
help:      vm capture [options] <resource-group> <name> <vhd-name-prefix>
help:
help:    Set the state of a VM in a resource group to Generalized.
help:      vm generalize [options] <resource-group> <name>
help:
help:    Get instance view of a VM in a resource group.
help:      vm get-instance-view [options] <resource-group> <name>
help:
help:    Get serial output of a VM in a resource group.
help:      vm get-serial-output [options] <resource-group> <name>
help:
help:    Enables you to reset Remote Desktop Access or SSH settings on a Virtual Machine and to reset the password for the account that has administra
tor or sudo authority.
help:      vm reset-access [options] <resource-group> <name>
help:
help:    Enable Azure Enhanced Monitoring Extension on a VM in a resource group
help:      vm enable-aem [options] <resource-group> <name>
help:
help:    Update a VM in a resource group.
help:      vm set [options] <resource-group> <name>
help:
help:    Get usage of compute resources.
help:      vm list-usage [options] <location>
help:
help:    Enable diagnostics extension on a VM in a resource group.
help:      vm enable-diag [options] <resource-group> <name>
help:
help:    Enable azure disk encryption on a VM in a resource group.
help:      vm enable-disk-encryption [options] <resource-group> <name>
```

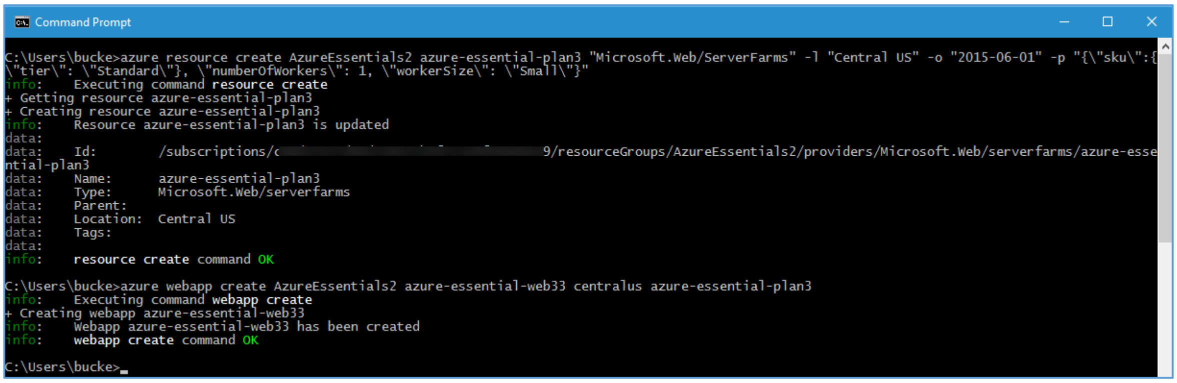
Figure 8-21 Options for working with Azure Virtual Machines in Azure Resource Manager mode.

You can perform many tasks with the Azure CLI. For example, to create a new Azure Web App and App Service Plan, you would execute the following commands:

Creating a new App Service Plan and Azure Web App

```
azure resource create AzureEssentials2 azure-essential-plan2 "Microsoft.Web/ServerFarms" -l
"centralus" -o "2015-06-01" -p "{\"sku\":{\"tier\": \"Standard\"}, \"numberOfWorkers\": 1,
\"workerSize\": \"Small\"}"

azure webapp create AzureEssentials2 azure-essential-web22 centralus azure-essential-plan2
```

When complete, you should see a result similar to that shown in Figure 8-22 that indicates the web app and plan were created successfully.



```
C:\Users\bucke>azure resource create AzureEssentials2 azure-essential-plan3 "Microsoft.Web/ServerFarms" -l "Central US" -o "2015-06-01" -p "{\"sku\":{
\"tier\": \"Standard\"}, \"numberOfWorkers\": 1, \"workerSize\": \"Small\"}"
info:    Executing command resource create
+ Getting resource azure-essential-plan3
+ Creating resource azure-essential-plan3
info:    Resource azure-essential-plan3 is updated
data:
data:    Id:        /subscriptions/c[redacted]9/resourceGroups/AzureEssentials2/providers/Microsoft.Web/serverfarms/azure-esse
ntial-plan3
data:    Name:      azure-essential-plan3
data:    Type:      Microsoft.Web/serverfarms
data:    Parent:
data:    Location:  Central US
data:    Tags:
data:
info:    resource create command OK

C:\Users\bucke>azure webapp create AzureEssentials2 azure-essential-web33 centralus azure-essential-plan3
info:    Executing command webapp create
+ Creating webapp azure-essential-web33
info:    Webapp azure-essential-web33 has been created
info:    webapp create command OK

C:\Users\bucke>_
```

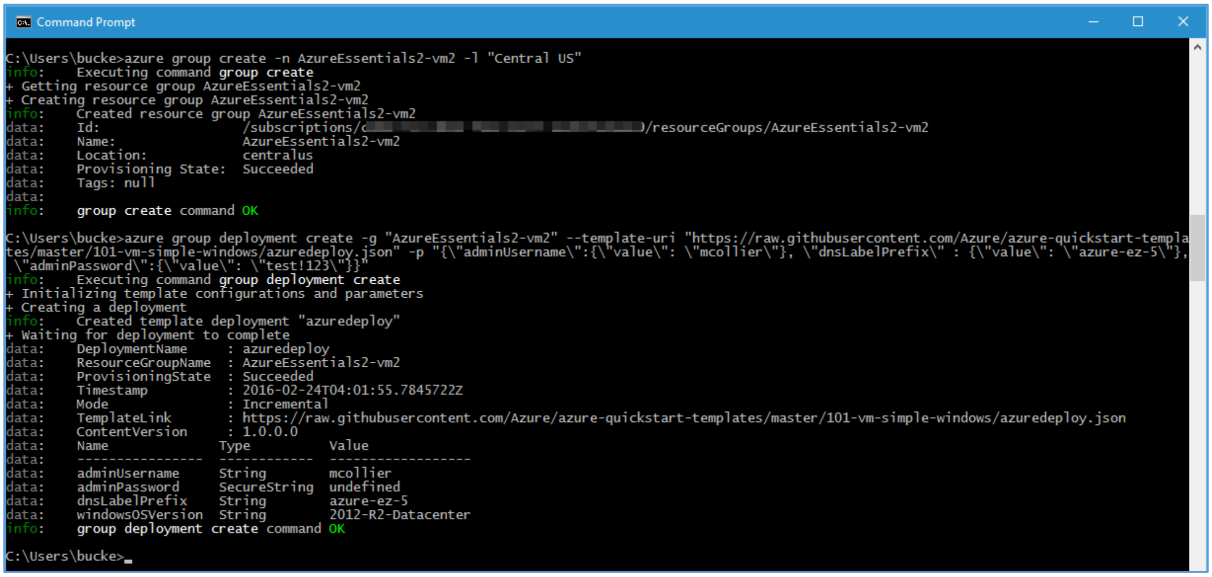
Figure 8-22 The result of creating a new Azure Web App and App Service Plan from the Azure CLI.

It is also possible to deploy an ARM template by using the Azure CLI, similarly to how it is done by using the Azure PowerShell cmdlets (as seen earlier in this chapter).

Creating a new VM by using an ARM template

```
azure config mode arm

azure group create -n AzureEssentials2-vm2 -l “centralus”

azure group deployment create -g "AzureEssentials2-vm2" --template-uri
"https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-
windows/azuredeploy.json" -p "{\"adminUsername\":{\"value\": \"mcollier\"}, \"dnsLabelPrefix\" :
{\"value\": \"azure-ez-5\"}, \"adminPassword\":{\"value\": \"test!123\"}}"
```

When complete, you should see a result similar to that shown in Figure 8-23 that indicates the Azure VM was deployed successfully to the newly created resource group.



```
C:\Users\bucke>azure group create -n AzureEssentials2-vm2 -l "Central US"
info:    Executing command group create
+ Getting resource group AzureEssentials2-vm2
+ Creating resource group AzureEssentials2-vm2
info:    Created resource group AzureEssentials2-vm2
data:    Id:                  /subscriptions/c[redacted])/resourceGroups/AzureEssentials2-vm2
data:    Name:                AzureEssentials2-vm2
data:    Location:            centralus
data:    Provisioning State:  Succeeded
data:    Tags: null
data:
info:    group create command OK

C:\Users\bucke>azure group deployment create -g "AzureEssentials2-vm2" --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templa
tes/master/101-vm-simple-windows/azuredeploy.json" -p "{\"adminUsername\":{\"value\": \"mcollier\"}, \"dnsLabelPrefix\" : {\"value\": \"azure-ez-5\"},
 \"adminPassword\":{\"value\": \"test!123\"}}"
info:    Executing command group deployment create
+ Initializing template configurations and parameters
+ Creating a deployment
info:    Created template deployment "azuredeploy"
+ Waiting for deployment to complete
data:    DeploymentName     : azuredeploy
data:    ResourceGroupName  : AzureEssentials2-vm2
data:    ProvisioningState  : Succeeded
data:    Timestamp          : 2016-02-24T04:01:55.7845722Z
data:    Mode               : Incremental
data:    TemplateLink       : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json
data:    ContentVersion     : 1.0.0.0
data:    Name               Type          Value
data:    ----------------   ------------  ------------------
data:    adminUsername      String        mcollier
data:    adminPassword      SecureString  undefined
data:    dnsLabelPrefix     String        azure-ez-5
data:    windowsOSVersion   String        2012-R2-Datacenter
info:    group deployment create command OK

C:\Users\bucke>_
```

Figure 8-23 The result of creating the new resource group and Azure VM from the Azure CLI.

See Also For additional information on using the Azure CLI with Azure Resource Manager, please refer to https://azure.microsoft.com/documentation/articles/xplat-cli-azure-resource-manager/. For additional information on using the Azure CLI with Azure Service Management, please refer to https://azure.microsoft.com/documentation/articles/virtual-machines-command-line-tools/.

Additional Azure services

This book focuses on the fundamentals of Azure. The services in Azure that are considered fundamental can be debated. A service such as Azure SQL Database may be considered fundamental to some, but not others. The fundamental services laid out in this book are those that the authors feel, after many years of real-world projects, to be fundamental to a wide majority of Azure users.

Some other Azure services we think you should know about

One of the joys of working with the Azure platform is the breadth of services it offers. There are many more services offered today than just a few years ago . . . or even one year ago when the first edition of this book was published! This section will briefly review several of the Azure services not covered in earlier chapters. It should be noted that this chapter does not provide an exhaustive listing of Azure services. The goal here is to provide exposure to other services available in the Azure platform. Some may be fundamental to you—today or in the future.

Azure Service Fabric

Azure Service Fabric represents the next generation hosting environment for cloud-scale solutions. Service Fabric makes it much easier to deploy and manage highly scalable, available, and reliable services. Applications designed as microservices are well suited to run on Service Fabric, primarily due to Service Fabric's runtime and lifecycle management capabilities—providing comprehensive capabilities for failover, leader election, state management, live upgrades with rollback, and automatic scale-up and scale-down.

Service Fabric is a proven platform for running critical, cloud-scale solutions. In fact, Microsoft uses Service Fabric to power some of Microsoft's most critical services, such as Skype for Business, Cortana, Azure SQL Database, Azure Event Hubs, and several more.

Service Fabric is not a platform that runs exclusively on Azure. You can deploy Service Fabric on-premises, in other cloud platforms, or, of course, in Azure. Microsoft has also committed to providing Service Fabric for both Windows and Linux operating systems. All this makes Service Fabric a platform you can run pretty much anywhere!

Cloud Services

Cloud Services is a platform as a service (PaaS) compute feature in which applications are deployed into instances (managed virtual machines) of server types referred to as web roles and worker roles. The deployment of the instances is fully managed by Microsoft, making it easy to scale applications in and out.

After writing your application, you create a deployment package for your application and upload it to Azure; Azure will create the requested number of VMs and install your software on them. Azure manages the VMs, bringing up new instances if one crashes, and handling the updates without downtime. Azure also manages the load balancing and autoscaling for the VMs. You can easily change the number of VMs by modifying the instance count in the classic Azure portal. If you increase the instance count, Azure takes the original deployment package and deploys additional instances. If you decrease the instance count, Azure shuts down and removes instances.

Cloud Services web and worker roles are classic resources, and can only be used with the classic version of resources such as virtual networks and storage accounts. The Resource Manager deployment model is the recommended deployment model moving forward in Azure. For Cloud Services, the recommendation is to use other parts of Azure to run these kinds of workloads. New web workloads are better handled by the App Service feature. The worker roles that loop infinitely and do work such as reading messages from a queue can be migrated to WebJobs or the new Service Fabric product.

Azure Container Service

One of the hottest technology trends in the last two years is containerization, with the open source project, Docker, being one of the leading platforms for managing containers. Docker containers provide an efficient, lightweight approach to application deployment by allowing different components of the application to be deployed independently into different containers. Multiple containers can reside on a single machine, and containers can be moved between various machines. The extreme portability of the container makes it very easy for applications to be deployed in multiple environments—either on-premises or in the cloud—often with no changes to the application.

There is more to deploying and managing container applications than simply using Docker. Supporting services such as monitoring, clustering, orchestration, scheduling, and a way to visualize the environment are also needed. There is a rich, yet still maturing, ecosystem to help with these needs. This is where the Azure Container Service (ACS) provides value.

ACS supports Docker container images and simplifies the process of creating, configuring, and managing the VMs that provide the infrastructure for a Docker cluster. ACS includes industry-leading tooling from Apache Mesos-based DC/OS and Docker Swarm to provide an optimized configuration for resource orchestration. For workload scheduling, ACS includes the popular Marathon and Chronos frameworks. By providing an optimized configuration of open source tools and technologies, ACS makes it much quicker and easier for you to migrate container applications to and from Azure. You use an Azure Resource Manager template to deploy the cluster, selecting the size, number of host VMs, and orchestration tool of choice. Once the ACS cluster is deployed, you can use your existing management tools to connect and manage your container applications.

DocumentDB

DocumentDB is a fully managed document database as a service designed to handle large amounts of data with no defined, rigid schema. It is highly available and performant, and it can be scaled up and down on demand. DocumentDB easily supports applications that need key value, document, or tabular data models.

A DocumentDB database is comprised of JSON documents. By default, all of these documents are indexed automatically, so you don't need to define secondary indexes for advanced searching. Because the data is schema-free, as your applications or their data requirements change, you don't need to make modifications to a schema to ensure your data models continue to work.

DocumentDB enables complex ad hoc queries using a SQL dialect. It is also integrated with JavaScript, allowing you to execute application logic within the database engine in a database transaction. This capability enables you to perform multidocument transaction processing using stored procedures, triggers, and user-defined functions.

Another interesting feature of DocumentDB is that it has protocol support for MongoDB. This means if you have applications written for MongoDB, you can use DocumentDB as the data store by installing some drivers and (in some cases) simply changing the connection string to point to the DocumentDB.

DocumentDB may look like other NoSQL database options out there, but the ability to perform transactional processing and complex queries make it particularly useful.

Azure Redis Cache

Redis is an open source, in-memory data structure store often used as a cache, database, or message broker. Azure Redis Cache is based on the popular open source Redis. The difference is that Azure manages Redis for you, saving you the trouble of spinning up a VM and installing and managing Redis yourself while still giving you a secure and dedicated Redis cache that can be accessed from any application within Azure. You can provision a Redis cache using the Azure portal.

Azure Redis Cache is available in three tiers:

- **Basic** This tier provides one node in multiple sizes. This tier is ideal for dev/test scenarios, and it has no service level agreement (SLA).
- **Standard** This tier provides resources for a replicated cache on two nodes in a primary/secondary configuration managed by Microsoft. This tier has a high availability SLA.
- **Premium** This tier includes everything in the Standard tier plus better performance, enhanced security, the ability to handle bigger workloads, and disaster recovery. Additional features include the following:
    - You can use Redis persistence to persist data stored in the Redis cache. You can also take snapshots and back up the data (which can be reloaded later in case of failure).
    - You can use Redis cluster to shard data across multiple Redis nodes, creating workloads of bigger memory sizes for better performance.
    - You can deploy your Redis cache in a VNet, providing enhanced security and isolation for your Redis cache, as well as subnets, access control policies, and so on.

Azure HDInsight

Azure HDInsight is a fully managed Apache Hadoop service, using the Hortonworks Data Platform (HDP) Hadoop distribution. HDInsight also includes other popular platforms that are commonly deployed alongside Hadoop, such as Apache HBase, Apache Storm, Apache Spark, and R Server for Hadoop.

HDInsight is used in big data scenarios. In this case, big data refers to a large volume of collected—and likely continually growing—data that is stored in a variety of unstructured or structured formats. This can include data from web logs, social networks, Internet of Things (IoT), or machine sensors—either historical or real-time. For such large amounts of data to be useful, you have to be able to ask the right question. To ask the right question, the data needs to be readily accessible, cleansed (removing elements that may not be applicable to the context), analyzed, and presented. This is where HDInsight comes into the picture.

The Hadoop technology stack has become the de facto standard in big data analysis. The Hadoop ecosystem includes many tools—HBase, Storm, Pig, Hive, Oozie, and Ambari, just to name a few. You can certainly build your own custom Hadoop solution using Azure VMs. Or you can leverage the Azure platform, via HDInsight, to provision and manage one for you. You can even deploy HDInsight clusters on Windows or Linux. Provisioning Hadoop clusters with HDInsight can be a considerable time saver (versus manually doing the same).

Azure Search

Azure Search is a search as a service solution. You populate the service with your data, and then you add search capabilities to your web or mobile applications that call the service to search that data. Microsoft manages the search infrastructure for you and offers a 99.9 percent SLA. You can scale to handle more document storage, higher query loads, or both.

You can search your data using the simple query syntax that includes logical operators, phrase search operators, suffix operators, precedence operators, and so on. You can also use the Lucene query syntax to enable fuzzy search, proximity search, and regular expressions. Data integrations allow Azure Search to automatically crawl Azure SQL Database, DocumentDB, or Azure Blob storage to create an index for your search.

At this time, 56 languages are supported. Azure Search can analyze the text your customer types in the search text box to intelligently handle language-specific terms such as verb tenses, gender, and more. You can even enable autocomplete for the search text boxes. Additionally, Azure Search includes geo-spatial support so you can process, filter, and display geographic locations. This means you can show search results ordered by proximity, such as the closest Starbucks.

Azure Service Bus

Azure Service Bus is a managed service for building reliable and connected applications (either on-premises or in the cloud) leveraging messaging patterns. Service Bus is often used as a key component in eventually consistent solution architectures—providing asynchronous messaging integrated with additional Azure resources such as SQL Database, Storage, Web Apps for App Service, or applications hosted on Azure Virtual Machines.

Service Bus features four different communication patterns:

- **Queues** Provide a basic FIFO (first in, first out) messaging pattern. Messages in a queue are stored until they are retrieved and deleted. Service Bus queues are conceptually similar to Azure Storage queues, yet they offer a few more advanced middleware capabilities (dead lettering, auto-forwarding, sessions, duplicate detection, etc.).
- **Topics** Provide a publish-and-subscribe messaging pattern. A message can be written to a topic, and multiple subscriptions can be attached to that topic, with different subscriptions receiving different messages depending on a filter.

- **Relays** Provide a bidirectional (two-way) communication pattern. Instead of storing messages like queues and topics, a relay simply proxies calls from the client/producer to the server/receiver. Service Bus Relays is also one of the older services in Azure. It was publicly announced in May 2006 as Live Labs Relay (since incorporated into the Service Bus family).
- **Event Hubs** Provide a highly scalable event and telemetry ingestion service for supporting scenarios requiring low latency and high reliability. The section below discusses Event Hubs in more detail.

Azure Event Hubs

Azure Event Hubs is a highly scalable managed service capable of ingesting millions of events per second, enabling you to capture, process, and analyze massive amounts of data originating from connected devices (often IoT scenarios) and applications. You can think of Event Hubs as a gateway, or entry point, for an event processing pipeline. Data is collected into an Event Hub, then transformed and stored. You have control over what data transformations and storage are needed.

The programmatic interface for Event Hubs is AMQP (Advanced Message Queuing Protocol) or HTTP(S), making it very easy for a wide range of clients to publish event data to Event Hubs. To support the need for massive scale, Event Hubs uses a partitioning pattern to scale the load internally. Receiving messages from an Event Hub is handled via consumer groups. Consumer groups are responsible for knowing from which partition to read and maintaining a view (state, position in the stream, etc.) of the Event Hub.

You will often see Azure Event Hubs used to ingest data in a big data or IoT scenario. A characteristic of both scenarios is the generation and processing of large volumes of (often relatively small in size) data. To process and analyze the data, another Azure service, Azure Stream Analytics, is often paired with Event Hubs.

It is important not to confuse Event Hubs with Azure Service Bus queues or topics. While the two are similar in that they are both messaging systems, Event Hubs is designed specifically for handling message events at high scale. It does not implement some of the messaging capabilities of Service Bus queues and topics, such as dead lettering, filters (property based routing), and various message retrieval, delivery, and scale semantics. Service Bus is better suited for per-message needs, while Event Hubs is better suited for event streaming needs.

Azure Notification Hubs

While Event Hubs allow you to take in millions of events per second, Azure Notification Hubs send data in the other direction—they enable you to send push notifications to mobile devices from any backend, whether in the cloud or on-premises. With a single API call, you can target individual users or entire audience segments of millions of users across all of their devices.

Push notifications are challenging. In general, the app developer still has to do much of the work to implement even common push notification scenarios, like sending notifications to a specific group of customers. To make them work, you have to build infrastructure that is complicated and, in most cases, unrelated to the business logic for the app.

Notification Hubs remove that complexity, eliminating the need for you to manage the challenges of push notifications. Notification Hubs are cross-platform—they can be used to support Windows, iOS, Android, and Windows Phone apps; they reduce the amount of push-specific code you have to put in your backend. They are fully scalable, allowing you to send notifications to millions of devices with a single API call.

All of the functionality of a push infrastructure is implemented in Notification Hubs for you. The devices only have to register their PNS handles, and the backend can send messages to customers without worrying about the platform the customers are using.

Azure Media Services

Azure Media Services enables you to provide audio or video content that can be consumed on-demand or via live streaming. For example, NBC used Azure Media Services to stream the 2014 Olympics (http://blogs.microsoft.com/blog/2014/02/06/going-for-gold-windows-azure-media-services-provide-live-and-on-demand-streaming-of-2014-olympic-winter-games-on-nbc/#sm.00001fhr9yr2zfciwlu2fkqhgu8kp).

To use Media Services, you can call the .NET or REST APIs, which allow you to securely upload, store, encode, and package your content. You can build workflows that handle the process from start to finish and even include third-party components as needed. For example, you may use a third-party encoder and do the rest (upload, package, deliver) using Media Services.

Media Services is easy to scale. You can set the number of Streaming Reserved Units and Encoding Reserved Units for your account. Also, although the storage account data limit is 500 TB, if you need more storage, you can add more storage accounts to your Media Services account to increase the amount of available storage to the total of the combined storage accounts. And last but not least, you can use the Azure CDN with Media Services for the fastest content delivery possible.

Azure Backup

Azure Backup is a backup as a service offering that provides protection for physical or virtual machines no matter where they reside—on-premises or in the cloud. Azure Backup encompasses several components (Azure Backup agent, System Center Data Protection Manager [DPM], Azure Backup Server, and Azure Backup [VM extension]) that work together to protect a wide range of servers and workloads.

Azure Backup uses a Recovery Services vault for storing the backup data. A vault is backed by Azure Storage (block) blobs, making it a very efficient and economical long-term storage medium. With the vault in place, you can select the machines to back up and define a backup policy (when snapshots are taken and for how long they're stored).

Azure Backup can be used for a wide range of data backup scenarios, such as the following:

- Files and folders on Windows OS machines (physical or virtual)
- Application-aware snapshots (VSS—Volume Shadow Copy Service)
- Popular Microsoft server workloads such as Microsoft SQL Server, Microsoft SharePoint, and Microsoft Exchange (via System Center DPM or Azure Backup Server)
- Linux support (if hosted on Hyper-V)
- Native support for Azure Virtual Machines, both Windows and Linux
- Windows 10 client machines

Even though Azure Backup and Azure Site Recovery share the same Azure portal experience, they are different services and have different value propositions. Azure Backup is for the backup and restore of data on-premises and in the cloud—it keeps your data safe and recoverable. Azure Site Recovery is about replication of virtual or physical machines—it keeps your workloads available in an outage.

Azure Site Recovery

Azure Site Recovery (ASR) provides a disaster recovery as a service solution for Hyper-V, VMware, and physical servers, using either Azure or your secondary datacenter as the recovery site. ASR can be a key part of your organization's business continuity and disaster recovery (BCDR) strategy by orchestrating the replication, failover, and recovery of workloads and applications if the primary location fails.

While there are many attractive technical features to ASR, there are at least two significant business advantages:

- ASR enables the use of Azure as a destination for recovery, thus eliminating the cost and complexity of maintaining a secondary physical datacenter.
- ASR makes it incredibly simple to test failovers for recovery drills without impacting production environments. This makes it easy to test your planned or unplanned failovers. After all, you don't really have a good disaster recovery plan if you've never tried to fail over.

The recovery plans you create with ASR can be as simple or as complex as your scenario requires. They can include custom PowerShell scripts, Azure Automation runbooks, or manual intervention steps. You can leverage the recovery plans to replicate workloads to Azure, easily enabling new opportunities for migration, temporary bursts during surge periods, or development and testing of new applications.

Azure Key Vault

Azure Key Vault is used to safeguard cryptographic keys and secrets in hardware security modules (HSMs) and allows Azure applications and services to use them. For example, you might use Key Vault to store storage account keys, data encryption keys, authentication keys, .PFX files, or passwords.

You can use Azure Active Directory (Azure AD) to control access to a Key Vault, which means you can control access to your keys and secrets using Azure AD. We talked about one example in Chapter 4, "Azure Storage," where you can store your storage account keys that are used by a service principal (an identity representing an application) into an Azure Key Vault and give access only to that service principal, thus protecting your storage account keys.

You can generate keys using Key Vault, but you can also store keys you have generated outside Azure. For security purposes, Microsoft cannot see or extract your keys. There is also logging capability that allows you to monitor the use of your keys in Key Vault.

More Azure services

The list of Azure services in the preceding pages is a sampling of the many services available in the Azure platform. Azure moves at a rapid pace, and new services and features are offered frequently. The rapid pace of innovation is one of the many fun aspects of working with a dynamic platform like Azure.

You are encouraged to review the main Azure site at http://azure.microsoft.com to learn more about the many services available.

Also, there is a web application that shows the many services of Azure and allows you to drill down to learn more. See http://aka.ms/azposterapp.

CHAPTER 10

Business cases

There are many business cases for using Microsoft Azure: from spinning up temporary development and test environments to extending your on-premises infrastructure into the cloud or developing new applications that take advantage of the features available in Azure. In this chapter, we discuss a few common scenarios to give you some ideas for how you can use Azure.

Development and test scenarios

One of the common workloads in Azure is development and test (dev/test). In most cases, you can replicate all or part of your production infrastructure in Azure, whether it be on-premises or already running in Azure, and use the replica for development, staging, or testing.

If you have an on-premises datacenter and you want to set up a dev/test environment, you have to procure hardware, install the operating system and the rest of the software, set up networking, configure the firewall, and so on. This can take a substantial amount of time. Once the testing is over, you have to either leave the hardware idle or repurpose it until you need it for other testing.

With Azure, you can provision what you need (virtual machines [VMs], web apps, databases, storage, and so on) and proceed with the testing within minutes. When you are finished testing, you can tear down all of the services and stop paying for them. In fact, using Azure you can automate the deployment and teardown of your dev/test environment by using PowerShell, the command-line interface (CLI), and/or Azure Resource Manager (ARM) templates.

Best of all, as your infrastructure grows, you can easily scale your dev/test environment to fit current needs. With an on-premises dev/test infrastructure, you have to go through the procurement and configuration process again.

If everything you have is on-premises, you can still use Azure for dev/test. You can set up a virtual network and extend your on-premises network into Azure. For example, you might want to test your application against a new version of SQL Server; you can have a web application running in your local datacenter that accesses SQL Server hosted in Azure.

If you have an MSDN subscription, you get a monthly credit to use for your dev/test infrastructure in Azure. In addition, several of the services are discounted. For example, Windows VMs are billed at the equivalent Linux rate (effectively removing the Windows license cost). This can significantly lower the overall cost of setting up and using a dev/test infrastructure.

Here are some other business cases that you can cover by using Azure to quickly replicate the parts of your infrastructure.

- **The flexibility of trying something small really fast** Let's say you only want to test one thing. For example, you want to change the way something is displayed in your website, but you're not sure if it will work or how it will work. You can make the modifications to the website and then deploy it as a new website with the configuration pointing to the production backend. Then, you can check the workflow and visual layout and decide if it is worth setting up a complete dev/test environment to proceed.
- **Load testing** You can create an entire copy of your production environment and then do load testing on the copy. This can include different VMs, websites, storage, virtual networks, and so on. This gives you the isolation to do load testing without affecting any of the production services, and it can help you pinpoint potential bottlenecks in your workflow so you can handle them before they affect the customer.

  You can use load testing to figure out the scope of the resources you need to handle different loads, such as the size or number of VMs or App Service instances. You can then change these services to autoscale when needed. For example, you might discover that as the percentage of CPU utilization exceeds 60 percent, your website becomes unacceptably slow, so you decide to implement autoscaling to increase the instance count when the CPU utilization hits the target value.

  Load testing leads to a better overall experience for your customer.
- **Software upgrades** If you are using software from an external company, you will want to test your software with the company's software for compatibility before upgrading your production services. For example, if you use SQL Server 2012, and SQL Server 2014 is released, you will want to test your application against the new version before upgrading. You might need to modify your software to work with the new version of SQL Server and go through the whole cycle of testing, staging, and implementation.

  In an on-premises environment, you probably can get a prerelease copy or a free short-term trial of the new version. However, you need to have infrastructure on which to run it, so as with the previous examples, you might need to procure hardware and so on. With Azure, the software might be available in a preconfigured VM, as is the case with Windows, SQL Server, Oracle, and Linux, among others. In that case, you can just provision a new VM with the new version in a dev/test scenario and run your software against it.

  If there is no preconfigured VM in the Azure Marketplace, you can provision a Windows or Linux VM, install the new version of software on it, and use that for your dev/test scenario.
- **A/B testing** Let's say you want to perform A/B testing on your website without repeatedly redeploying the different versions of the website. Azure Web Apps allows multiple deployment slots. You can publish version A to one slot and version B to another and then swap them in and out of production as needed to perform the testing and metrics collection.

  Another option is to use the DNS query and routing policies available with Azure Traffic Manager. Traffic Manager gives you the ability to balance incoming traffic among multiple services using the following methods: performance, weighted, and priority. With the weighted method, you can distribute load to various endpoints based upon an assigned weight for each endpoint. This means you can divert a small percentage of traffic to a separate deployment to perform A/B testing.

For more information about using Traffic Manager load balancing, check out https://azure.microsoft.com/documentation/articles/traffic-manager-overview/.

Hybrid scenarios

The number of companies running solutions in the cloud is increasing at an incredible rate. Their success encourages other organizations to take the same step. Some organizations will not be able to move all of their workloads into the cloud, either because of regulatory issues or because some workloads cannot run in a virtualized environment. In these cases, hybrid computing, in which a company runs part of its infrastructure in the cloud and part on-premises, will be an important strategy.

The Microsoft Azure platform provides a great hybrid computing story. There are multiple ways to connect an on-premises datacenter to one or more Azure regions. As discussed in Chapter 5, "Azure Virtual Networks," Azure provides both site-to-site and point-to-site virtual network connectivity. Either option provides a secure VPN connection between on-premises assets and resources hosted in Azure. An additional hybrid connectivity option is Azure ExpressRoute, which enables a private connection between Azure and your on-premises infrastructure or colocation facility, all without going over the public Internet.

Network connectivity

Regardless of the chosen option—site-to-site, point-to-site, or ExpressRoute—hybrid connectivity is a key scenario for the Azure platform. Creating a hybrid connection opens a wide range of possibilities to extend an on-premises infrastructure to the cloud. Two common scenarios for network-enabled hybrid connectivity are the following:

- **Hosting a website in Azure but keeping the database on premises** In an organization's journey to the cloud, migrating the on-premises data to Azure can be one of the more difficult tasks. The difficulty usually comes in one of two forms: a technical issue or a compliance requirement. On the technical front, as an example, the application in question is designed to use a database that is not supported in Azure. On the compliance front, perhaps there is a regulatory requirement that cannot be met with Azure SQL Database or by running a database (SQL Server, MongoDB, and so on) on Azure Virtual Machines. In these cases, an organization might choose to host the website in Azure using Azure Web Apps or Azure Virtual Machines, with the database remaining on premises. Connectivity between the website and the database could then be established using one of the aforementioned technologies: a site-to-site connection, a point-to-site connection, or ExpressRoute.

- **Accessing an on-premises service** Sometimes, a website has a dependency on a particular service that cannot be moved to the cloud. Perhaps the website depends on an API that performs a crucial business calculation, and that API cannot be moved due to security because other on-premises services also depend on the service or because it is legacy technology that is not supported in Azure. In such a scenario, a hybrid connection is established between Azure and the on-premises infrastructure to allow the Azure-hosted website to freely communicate with the necessary API that continues to reside on-premises.

  Besides using a network connection in this scenario, an Azure Service Bus Relay could be used to access an on-premises service. For information on how to use the Azure Service Bus Relay service, please refer to http://azure.microsoft.com/documentation/articles/service-bus-dotnet-how-to-use-relay/.

Internet connectivity

There are many scenarios in which all that is needed is an Internet connection rather than a special hybrid connectivity solution. After all, the ability to connect to Internet-accessible services is one of the attractive features of cloud computing. A few common scenarios include these:

- **Storage of archival data** Large amounts of data, especially archival data that is rarely accessed, can be very expensive to store on-premises. The cost in terms of infrastructure, people, software licenses, and physical space can quickly put a tremendous financial burden on an organization. As discussed in Chapter 4, "Azure Storage," Azure provides virtually limitless storage capacity at an incredibly low price. An organization might wish to use the scalable storage provided by Azure Blob storage as a data archival store. When the data is needed, the on-premises service(s) download the data from Azure Blob storage and perform the necessary processing. A basic Internet connection will often suffice, but an ExpressRoute connection could also be used for improved speed and security.

  Another option for storage of archival data is Microsoft Azure StorSimple. StorSimple includes a hardware appliance that is installed on-premises. The appliance keeps frequently accessed data local (on the device). As data ages (is accessed less frequently), it is automatically moved to Azure Blob storage. For more information on StorSimple, please refer to http://azure.microsoft.com/documentation/services/storsimple/.

- **Azure Active Directory** As discussed in Chapter 7, "Azure Active Directory," organizations can choose to synchronize their Azure AD users and groups with user and group information from their on-premises Active Directory. In doing so, they can use Azure Active Directory Connect to synchronize the user data and a password hash, making Azure AD the authority for user authentication. Alternatively, an organization might wish to synchronize the user data but require users to authenticate via an Active Directory Federation Services (AD FS) endpoint residing on-premises, effectively redirecting the user to an on-premises AD FS site for authentication before redirecting to the desired location.

- **Burst to the cloud** Sometimes, an organization's on-premises infrastructure is not able to handle the required load. Maybe there is a holiday season rush or a government-mandated period to sign up for an important service. Instead of building the on-premises infrastructure to handle the temporary surge in demand, an organization might choose to leverage the elastic nature of the cloud to burst to the cloud when needed and scale back to only on-premises services when the load returns to normal. In this scenario, an organization could use Azure Web Apps or Azure Virtual Machines to host the service and could implement autoscale rules to ensure capacity keeps up with user demand.

Application and infrastructure modernization and migration

There comes a time in every application's life when it is time to upgrade. It could be a user interface redesign or a hardware refresh. The Azure platform cannot help create an appealing, modern user interface, but it can modernize the supporting infrastructure.

Many organizations will go through a periodic hardware refresh cycle; typically, this happens about every three years. When it is time for a hardware refresh, organizations today have a new question to ask: Should we buy new on-premises hardware, or should we leverage our infrastructure and services to the cloud?

Besides a required hardware refresh, an organization might choose to migrate to the cloud because it has reached physical capacity limits in its existing on-premises datacenter or because it is going to in the very near future. Perhaps the current datacenter does not have enough physical space for more servers or cannot supply the necessary power or cooling. Maybe there is a desire to eliminate or reduce the management of hardware infrastructure going forward. Moving to the cloud might enable the organization to get out of the datacenter business completely, or at least partially (see the section "Hybrid scenarios" earlier in this chapter). In this case, Microsoft is responsible for the hardware and related infrastructure components of the datacenter, and the organization can focus on providing great business solutions.

Some organizations will choose to migrate to the cloud to get capacity in new geographies they can't currently support because they have no presence in that area or because it would be cost-prohibitive. There are Azure datacenters in over 22 regions around the world from Melbourne to Amsterdam and from Sao Paulo to Singapore. Additionally, Microsoft has an arrangement with 21Vianet, making Azure available in two regions in China. Microsoft has also announced the deployment of Azure to another eight regions. Instead of building and maintaining a global datacenter presence, an organization can elect to take advantage of Microsoft's existing investments and deploy to multiple regions with ease.

See Also For the current list of Azure regions, please refer to http://azure.microsoft.com/regions/.

Should the choice be to modernize or migrate to the cloud, there is certainly a wealth of Azure resources available. In choosing to adopt these resources, an organization could have many questions to answer, including these:

- Do we leverage platform as a service (PaaS), infrastructure as a service (IaaS), or both?
- Instead of maintaining a custom solution, should we leverage platform-provided services such as Azure Search or Azure Media Services?
- Should we move everything, or just some components? What hybrid model works best for our requirements?
- Which Azure region(s) should we use?
- How does using Azure affect our business and operations model?
- What is our service level agreement (SLA)? What is our disaster recovery story?

Azure Mobile Apps

In today's world, mobile devices—from tablets to phones to watches to fitness bands—are everywhere you look. Having a mobile application can be a big plus for a company, whether it's used externally, internally, or both.

Azure Mobile Apps, included as part of Azure App Service, is a backend as a service that provides multiple features to make it easier and quicker to create a mobile application. Mobile Apps is both flexible and scalable, so when your application becomes widely used, you can scale appropriately to handle your customers' needs.

Another advantage of Azure Mobile Apps is that you only have to write one version of your backend. The backend can be used by devices running iOS, Android, and Windows, allowing you to reach every user on every platform without extra work.

The following are some of the features provided by Azure Mobile Apps. You can certainly program a service to implement these features from the ground up, but using Azure Mobile Apps saves you the time and money it would take to do that.

- **Data storage** You can choose for your data storage to be powered by SQL Database, which has an interface simple enough to use without being a DBA. You can also integrate with SQL Server, Azure Table Storage, MongoDB, DocumentDB, or via an API to software as a service (SaaS) providers such as Salesforce.com and Office 365.

  You can write your application to work offline and synchronize the data when the application can go online again. This is helpful when the customer loses Internet connectivity—the customer can continue to work, knowing the work will be stored on the backend when connectivity is regained.
- **User authentication and data authorization are greatly simplified** You can easily implement single sign-on (SSO) with Azure AD, a Microsoft account, Facebook, Twitter, and Google.
- **Push notifications** You can send information for customer and enterprise applications to any customer's mobile device by using Microsoft Azure Notification Hubs. This can come from any backend, whether it runs in Azure or is on-premises. Notification Hubs automatically handles the server-side code to push messages to the push notification services for iOS, Android, and Windows devices.

  Notification Hubs has a tagging feature that can be used to target audiences based on activity, interest, location, or preference. In addition, the templates feature of Notification Hubs enables you to send localized push notifications in the customer's own language.
- **Because Mobile Apps runs in Azure, you can easily scale in and out to meet customer demand** You can even set up autoscaling that will automatically scale out as demand increases, handling millions of devices.
- **You can use Microsoft Azure WebJobs to perform backend processing on the server at a scheduled time** For example, you might want to create a scheduled job that requests an update from your on-premises database and stores the new information in a table, waiting to be retrieved by your mobile application.
- **You can create a hybrid connection** This connection can be used to connect the mobile application to on-premises systems, Office 365, and SharePoint.

Machine learning

One of the most talked about emerging technologies in the last few years is machine learning. Machine learning isn't new by any stretch; however, the ability for a wide range of people to easily access the technology is relatively new—largely enabled by the ubiquitous nature of public cloud computing.

By using machine learning, we are able to leverage computers to analyze existing data to predict future behavior or outcomes. Maybe you want to predict when machines will break down instead of sending technicians to check on machines that may be working fine. Perhaps you want to analyze historical shopping data to predict what customers are likely to purchase in the future. These are just two of the many potential scenarios that are possible with machine learning.

It's becoming increasingly common to see machine learning mentioned along with Internet of Things (IoT). IoT solutions typically generate a lot of data from various sensors, such as temperature, vibrations, speed, and so on. The data itself is often largely useless—the true value comes in being able to analyze the data and determine what to do with the data to improve the overall solution. This is where machine learning comes into the picture. Combining the data from IoT solutions with machine learning can lead to interesting and useful insights about the data.

At first this might seem daunting, but it isn't—especially when using a service such as Azure Machine Learning. With Azure Machine Learning, there is no hardware to purchase or virtual machines to manage. In fact, you don’t even need an Azure subscription to get started with the Free tier of Azure Machine Learning! You can learn more about the pricing options for Azure Machine Learning at https://azure.microsoft.com/pricing/details/machine-learning/.

See Also Azure Machine Learning is also a component of the Cortana Intelligence Suite. Learn more at https://www.microsoft.com/server-cloud/cortana-intelligence-suite/.

The basic workflow in Azure Machine Learning is relatively simple:

1. Build a model from existing data. The data can come from numerous data stores in Azure, such as Azure Storage tables or blobs, Azure HDInsight (Hadoop), Azure SQL Database, or Azure Data Lake.

2. Publish the model as a web service.

3. Optionally, consume that web service from any number of tools such as mobile applications, websites, or business intelligence tools.

With the data in place, you can create your predictive model in Azure Machine Learning Studio, a browser-based tool with drag-and-drop capabilities that make it easy to get started. If you’re familiar with machine learning and understand how to use R (a programming language commonly used for data analysis) or Python, you can get started right away with Azure Machine Learning. If you’re not familiar with machine learning, you can get started by using solution templates in the Cortana Intelligence Gallery or by leveraging existing solutions in the Azure Marketplace (Data Market).

Once the model is created and properly trained (a process for validating that the model works as expected), you can publish the model as a web service. This will allow you—or others, based on your usage needs—to send data to your service and receive the predictions!

Azure Machine Learning is the perfect complement to the voluminous amount of data generated by many of today’s IoT solutions. It’s never been easier to gather, store, analyze, and make decisions based on data.

About the authors

Robin E. Shahan has over 25 years of experience developing complex, business-critical applications for Fortune 100 companies. As President of Nightbird Consulting, she provided training and helped companies architect and develop scalable, efficient solutions on the Azure platform. She is a six-time Microsoft MVP, and now works for Microsoft as a Sr. Content Developer for Azure Storage.

Robin regularly speaks about Microsoft Azure at various .NET User Groups and Code Camps and runs the San Francisco Azure meetup. She can be found on Twitter as @RobinDotNet, and you can read her articles about Microsoft Azure (and other subjects) at http://robindotnet.wordpress.com. You can reach her via e-mail at robin.shahan@microsoft.com.

Michael S. Collier is currently a Senior Software Development Engineer in the DX TED Commercial ISV team at Microsoft, and previously served as a Cloud Solution Architect. Prior to joining Microsoft in January 2015, Michael was a five-time Azure MVP and served as a Principal Cloud Architect with Aditi Technologies. He has over 15 years of experience with various consulting and technology firms where he was instrumental in leading and developing solutions for a wide range of clients. He has a vast amount of experience in helping companies determine the best strategy for adopting cloud computing, and providing the insight and hands-on experience to ensure they are successful. Michael is also a respected technology community leader, and is often found sharing his Microsoft Azure insights and experiences at regional and national conferences. Michael is also the co-founder of CloudDevelop Conference in Columbus, OH. You can follow Michael's experiences with Azure on his blog at http://www.michaelscollier.com and on Twitter at @MichaelCollier (http://www.twitter.com/MichaelCollier).

Michael lives in Marysville, Ohio with his wife and two sons. He is a 2003 graduate of The Ohio State University and is a passionate Buckeyes fan. Michael is also an avid golfer, although golf doesn't always like him.

Free ebooks

From technical overviews to drilldowns on special topics, get *free* ebooks from Microsoft Press at:

www.microsoftvirtualacademy.com/ebooks

Download your free ebooks in PDF, EPUB, and/or Mobi for Kindle formats.

Look for other great resources at Microsoft Virtual Academy, where you can learn new skills and help advance your career with free Microsoft training delivered by experts.

Microsoft Press