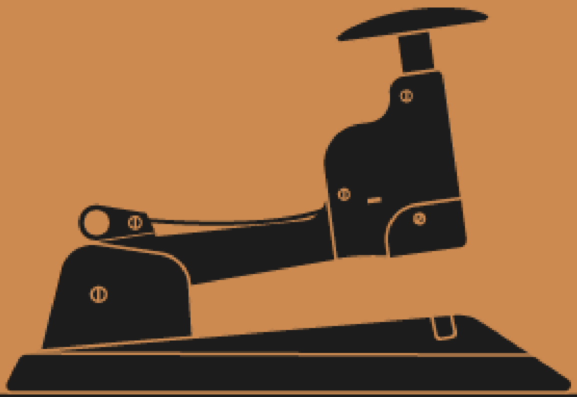


Windows Azure Websites Succinctly

by Kyle Burns

Syncfusion | Technology Resource Portal

Windows Azure Websites Succinctly

By

Kyle Burns

Foreword by Daniel Jebaraj

Important licensing information. Please read.

Technical Reviewer: Ricardo Peres
Copy Editor: Suzanne Kattau
Acquisitions Coordinator: Hillary Bowling, marketing coordinator, Syncfusion, Inc.
Proofreader: Darren West, content producer, Syncfusion, Inc.

Table of Contents

The Story behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the Internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

Free forever

Syncfusion will be working to produce books on several topics. The books will always be free. Any updates we publish will also be free.

Free? What is the catch?

There is no catch here. Syncfusion has a vested interest in this effort.

As a component vendor, our unique claim has always been that we offer deeper and broader frameworks than anyone else on the market. Developer education greatly helps us market and sell against competing vendors who promise to “enable AJAX support with one click” or “turn the moon to cheese!”

Let us know what you think

If you have any topics of interest, thoughts or feedback, please feel free to send them to us at succinctly-series@syncfusion.com.

We sincerely hope you enjoy reading this book and that it helps you better understand the topic of study. Thank you for reading.

Please follow us on Twitter and “Like” us on Facebook to help us spread the word about the *Succinctly* series!

About the Author

Kyle Burns is a Technical Architect with Perficient. He first discovered a love for writing computer applications when his father brought home their first Apple II computer and he would spend hours transcribing programs from BASIC magazine to see what the programs would do. After serving as a tuba and euphonium player in the United States Marine Band, Kyle realized that people were writing software for pay so he started his professional career.

During his career, Kyle has worked in companies ranging from a six-person startup to a Fortune 100 company. He is constantly looking for new ways to explore solving problems with people, process, and technology. Kyle has remained heavily focused on using the Microsoft stack to deliver solutions and has maintained certifications on Microsoft tools ranging from Visual Studio 6 to Visual Studio 2013. He lives in Indianapolis.

Introduction

Cloud Computing

If you have been developing software, managing infrastructure or even just watching television in the past several years, you have been inundated with references to cloud computing. You also may have been left scratching your head trying to figure out what it all means. It seems that every vendor has a slightly different definition for what cloud computing is and, appropriately, each vendor's definition is tailored to the products and services that they offer. In simplest terms, cloud computing is just a compelling name to describe the concept of distributed computing over a network that has been a staple of application development for decades (much like when Google popularized the term "Ajax" in the early 2000s and people perceived a new paradigm).

Types of Cloud Computing

Most vendors do agree on a set of categories into which most cloud computing applications can be placed. The categories are based on the characteristics of the application, most commonly driven by the management tasks distributed among the host platform and customer. These categories (described in more detail later) include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). For the purposes of discussion, we'll start with the assumption that a typical enterprise application has the following set of concerns at the infrastructure level (shown in Figure 1):

- Network: This includes the physical equipment such as switches and routers as well as the logical configuration of subnets. Many application developers often take the network part of the solution for granted, but there is no amount of code that can overcome the devastating effect of a poorly configured network on the stability and performance of an application.
- Storage: Most software applications and the businesses they support are highly dependent on the storage of data and in keeping that data safe from loss and unauthorized disclosure. Management tasks at the storage layer include configuring storage devices for optimum throughput, monitoring for device failure, tracking utilization, and ensuring that a backup plan exists and is followed.
- Servers: While my earliest memory of managing a network included wall-to-wall tower computers, each with a special function, the modern data center is characterized by fewer and larger physical machines, each hosting a number of virtualized servers. This model is much easier to manage than the model without virtualization, but it still brings a number of management tasks such as monitoring the servers for hardware failure, ensuring that adequate power is available for the equipment, and maintaining environmental conditions for optimal performance of the hardware.
- Virtualization: As noted in the discussion of physical servers, virtualization is the norm in the modern data center. Products such as Microsoft's Hyper-V, VMware's vSphere, and Citrix's ZenServer all let you perform management tasks such as managing the

distribution of physical resources among the virtual hosts, monitoring to optimize utilization of hardware, and provisioning and decommissioning of virtual hosts.

- Host Operating System: Virtualization does not remove the management tasks of host operating systems. These still need to be appropriately configured for performance, availability, and security. Management tasks often include ongoing application of operating system and security software (such as antivirus and intrusion detection) patches and any operating system-specific monitoring tasks that cannot be performed by the virtualization tool.
- Middleware: Middleware includes software that is used as a foundation on which to build applications without having to reinvent functionality commonly found within specific applications—allowing developers to focus on the specific value intended to be provided by their application. Common examples of middleware used by applications built on the Microsoft stack include (but certainly is not limited to) Microsoft Internet Information Services (IIS) (formerly Internet Information Server) to handle tasks such as managing request/response for HTTP requests, Microsoft AppFabric for providing a distributed cache, and Microsoft SQL Server to handle storage and retrieval of relational data. Each middleware comes with its own set of configuration and management tasks that varies by the complexity of the particular middleware, ranging from the “set it and forget it” nature of Microsoft AppFabric to the constant monitoring and tuning provided by database administrators for some Microsoft SQL Server implementations.
- Application Run Time: The application run time includes the container in which applications execute. For web applications built on the Microsoft stack, this container is most commonly Microsoft ASP.NET running either ASP.NET Web Forms or ASP.NET MVC. Certain run time configuration parameters (such as processing and memory resources to allocate to the container) are managed at the host level while other parameters are available for customization by the individual application.
- Application Data: The application data is usually the tangible product of having executed the application. Whether it’s recording bills of sale for a multinational product supplier or high scores in the latest bubble app craze, the management of what data to store and how to keep it consistent with the applicable rules is usually performed by the application code itself. Therefore, very little management activity that is not performed either at the storage or middleware layers is necessary.
- Application Code: The application code is the executable code that executes within the run time to perform actions directly related to achieving the goal of the application. This is typically the layer of the application stack most visible to the business stakeholders of the application. It is the layer on which the code most specific to solving problems related to the business domain can be found. I am often heard telling co-workers and clients that developers should work on solving problems that only they can solve. By that I mean developers should focus on the code in the application that is most focused on solving the specific business problem, instead of trying to reinvent frameworks for solving problems that are already addressed in common frameworks or by the run time.

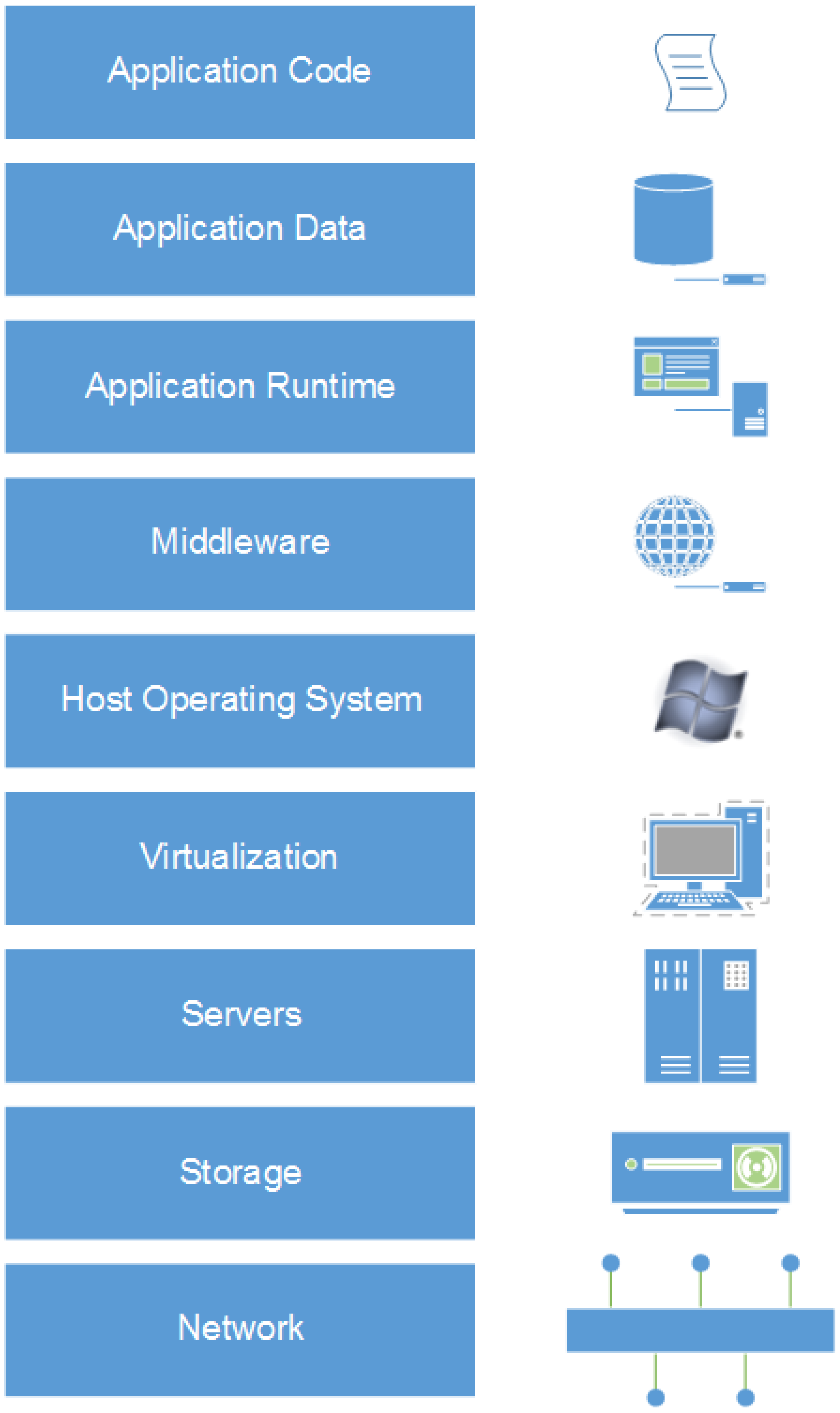


Figure 1: Full application stack

On-Premises Solution

On-premises solution is a term used to describe the traditional data center where all of the infrastructure and equipment is managed by the owner of the application. It is not a cloud computing model but is included in this discussion to establish a baseline for when cloud computing is not being used. It is also included because some scenarios will include resources that are not hosted in a cloud environment. In this model, all aspects displayed in Figure 1 are under the control and the responsibility of the application owner. When managed well, this often results in having to keep several departments' worth of specialized staff to operate the pieces of the application as well as significant ongoing capital expenditure.

Infrastructure as a Service

The Infrastructure as a Service (IaaS) model takes everything below the host operating system on the application stack as the responsibility of the host or host platform, leaving the host operating system and everything above it on the stack as the responsibility of the application owner. This arrangement (shown in Figure 2) allows for fine-grained control of the operating characteristics of the application and its host operating system. It is often seen as the first step that a company takes towards achieving a virtual data center because virtualized servers running the application can often be transferred to the hosted environment with no code changes to the application itself (and with only the configuration changes necessary to reflect that the application has a new home).

Organizations that move from an on-premises model to IaaS typically find cost savings in the form of reduced utility costs from no longer needing to power and cool servers, reduced capital equipment costs from no longer needing to purchase big servers, and reduced personnel expenses from no longer needing staff to perform the physical portions of managing server hardware in the data center.

While significant savings can be achieved with IaaS, organizations maintain the ability and responsibility to control the configuration and management of pieces of the stack that are not directly related to the application code or its data—namely, the host operating system, middleware, and application run time. This can be beneficial in some cases because of the additional control it gives to the owner of the application. But it also requires attention and specialized skills, which makes the IaaS option most appropriate when there is something special about the needs of an application that makes it unsuitable to give up control over these pieces of the stack.

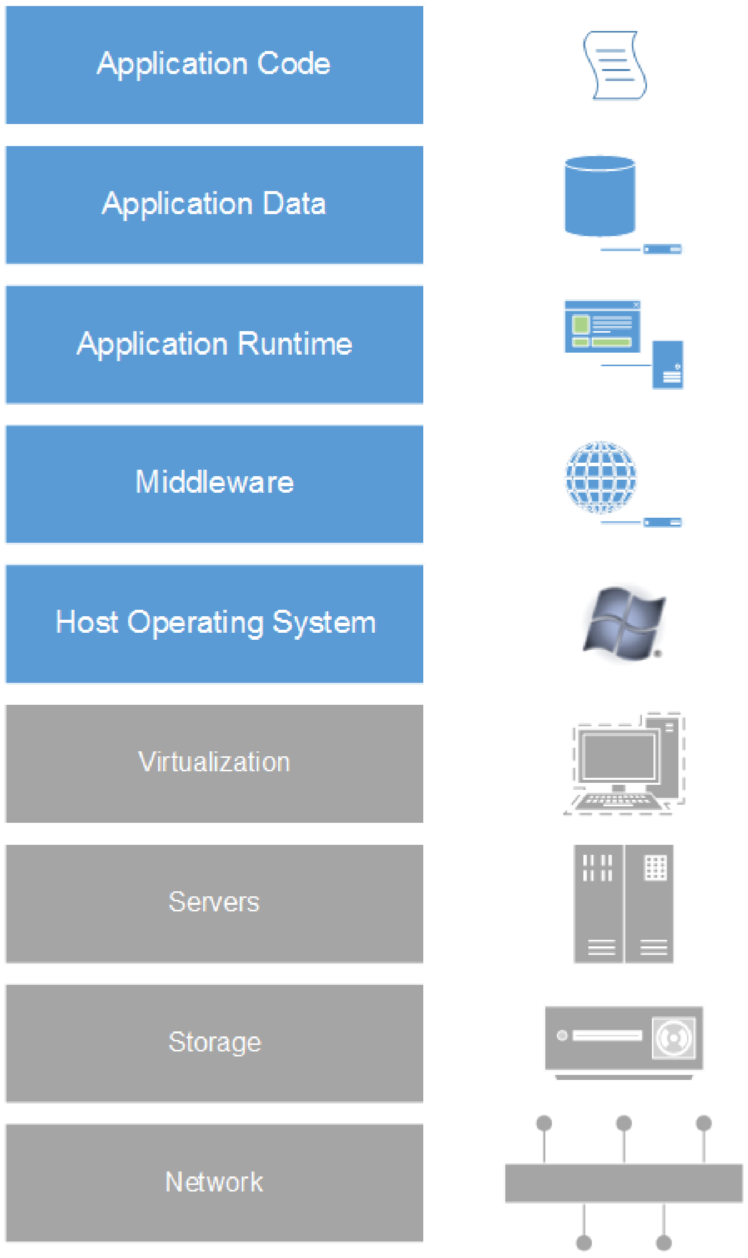


Figure 2: IaaS stack

Platform as a Service

The Platform as a Service (PaaS) model further reduces the pieces of the application stack that must be managed by the application owner to only the application data and application code itself as shown in Figure 3. In this model, the application owner writes code that is designed to run within an application run time that is hosted by the service provider—sometimes limited to a specific set of application programming interface (API) functionality. This model requires far less specific knowledge and skill in the operation and management of network and infrastructure, but leaves some application owners with the uneasy feeling of having critical business assets outside of their reach or control. The mitigation for this uneasiness is often a combination of growing accustomed to the model and finding a partner that can be trusted to keep these assets protected—whether that protection comes from their ability to keep things safe or from a contractual commitment to provide adequate relief if a compromise occurs.

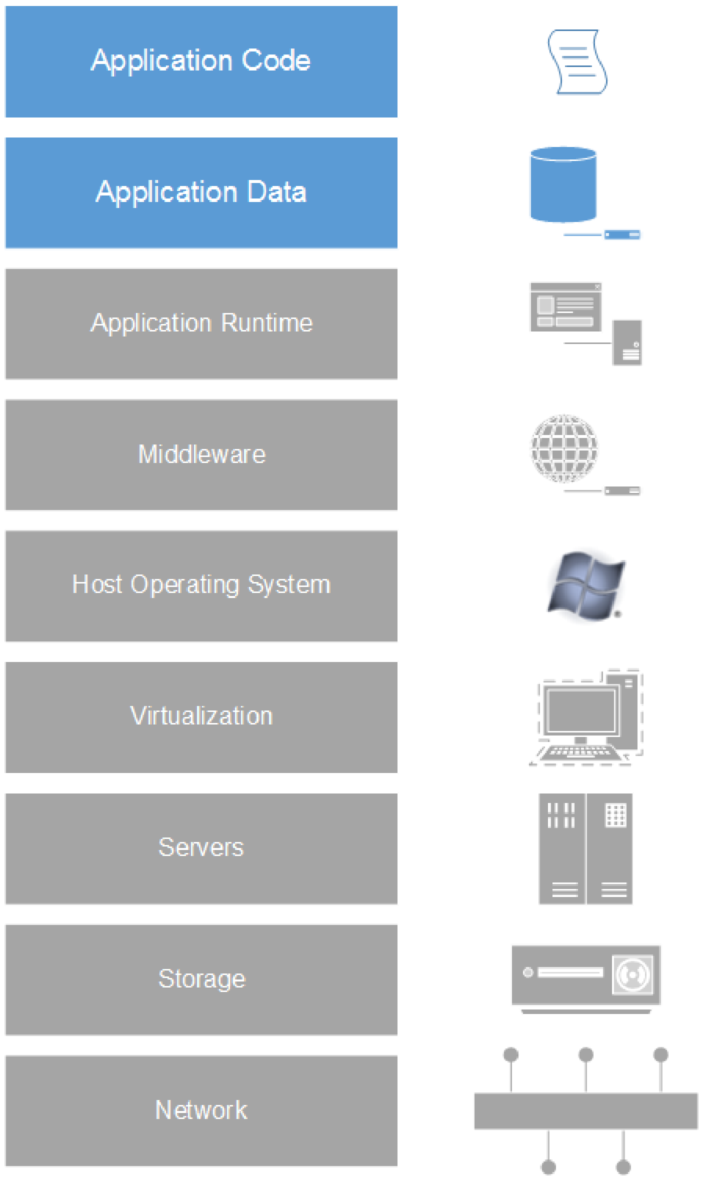


Figure 3: PaaS stack

Software as a Service

The Software as a Service (SaaS) cloud computing model takes all of the configuration and management responsibilities away from the application owner (as illustrated in Figure 4) and allows the application to simply be used as is (although sometimes with limited customization through configuration) to meet specific goals. This model is often seen used with general-purpose software such as email, word processing, and collaboration tools as well as with certain specific-use offerings that have use across industries. Some examples of SaaS offerings include Microsoft's Outlook.com and Office365 as well as out-of-the-box Salesforce.com (Salesforce also has a PaaS offering that allows you to build your own custom applications).

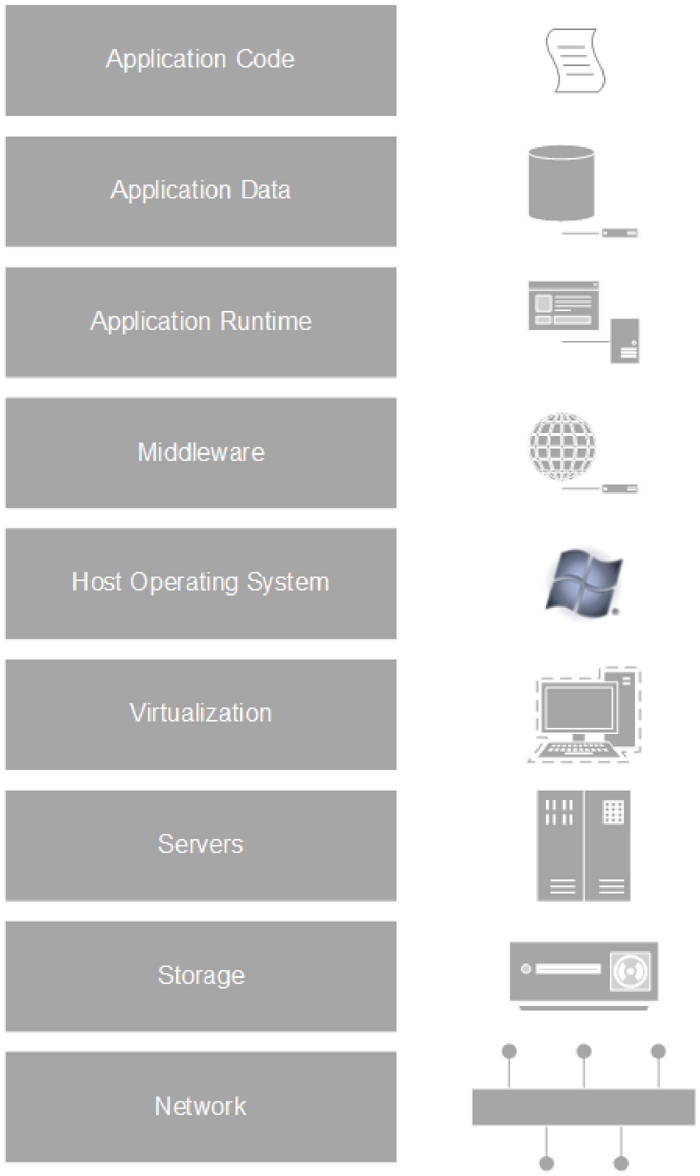


Figure 4: SaaS stack

Hybrid

While we have looked at on-premises, IaaS, PaaS, and SaaS as distinct models with clearly defined responsibilities (marking the difference between which model an organization is participating in), the reality is most organizations have either settled on an approach that is some combination of these models or are in a transitory state—moving to a desired future state that is one model but still retaining some portion of their previous model. These organizations will either temporarily or permanently use a hybrid model that is some combination of two or more of these models. One example of a hybrid model would be a customer application built on the IaaS or PaaS model, connecting to an on-premises mainframe application that is critical to the organization's business goals but cannot be migrated to a hosted environment.

What This Book is About

In this book we will be looking at the suite of products and services designed to enable cloud computing that Microsoft collectively markets under the Windows Azure brand. Our main focus will be on a specific product of that suite called Windows Azure Websites which provides the facilities to quickly and simply host data-driven websites that are suitable for many people's needs—without additional services and complexity. We will also, at times, look beyond the out-of-the-box features of Windows Azure Websites at the other capabilities of Windows Azure products to meet more advanced needs that aren't covered by the Windows Azure Web Site product.

Note: Windows Azure implements a pay-as-you-go model, so care will be taken in this book to call out when features that incur additional costs are being described. With any cloud-based product, you should make sure that you fully understand the costs of the features that you are using.

Chapter 1 Windows Azure

What is Windows Azure?

Windows Azure is a collection of products and services hosted in Microsoft's data centers to enable developers and organizations to quickly assemble the right combination of pieces for their application's needs. The tools represented in the Windows Azure suite almost exclusively focus on either hybrid, IaaS, or PaaS—with the only potential exception being the Visual Studio Online product. A key characteristic of all of the Windows Azure products is the idea that customers only pay for what they use, whether that is central processing unit (CPU) time on a virtual machine (VM) or disk space for persistent storage. This presents an extremely low barrier to entry for customers who initially have very low computing requirements, as well as a powerful tool to keep budget under control for customers with periods of heavy volume separated by periods where the volume is much lower.

Windows Azure Services

Microsoft separates the Windows Azure products into four major categories based on the type of service that they provide. In this section, we will briefly discuss the services in each category, going into more detail for those services more likely to be encountered when building Windows Azure Websites. If you are already familiar with the various services offered in Azure or just prefer to discover them as you go along, you can safely skip reading this section.

Compute

Compute services are products that focus on utilization of the CPU capability in the Microsoft data center hardware. These are all priced based on a combination of CPU cores and available random access memory (RAM) and share the same capability for health monitoring and auto-scaling to meet customer demand in a flexible way that still controls cost.

Virtual Machines

Virtual Machines (VMs) are the only product in the Compute category that is considered an IaaS offering. With VMs, you can run dedicated virtual servers in Microsoft's data center, starting from a number of predefined templates or even upload your own virtual hard disk (VHD) containing a configuration that you have created. These machines can run by themselves or participate in an ecosystem where they interact collaboratively with PaaS solutions.

Websites

The Windows Azure Websites product is focused on providing a targeted set of services to allow for the creation and maintenance of websites using many of the features required to make a website useful. This includes support for various server-side technologies including ASP.NET, classic ASP, PHP, Node.js, and Python, and includes either a Windows Azure SQL Database or MySQL to provide a backing store for your applications. As with the VMs product, application owners can either bring their own custom application or choose from a growing collection of site templates based on popular frameworks in the Application Gallery.

Mobile Services

Mobile Services are a set of services aimed at providing the most commonly required server-side functionality to power mobile applications. These services allow developers the opportunity to quickly add features such as:

- Cloud storage to ensure that your app users can share with each other and that users' application states can follow them from device to device.
- Authentication integrated with many popular identity providers such as Google, Microsoft, Facebook, Twitter or an Active Directory hosted within Azure.
- Push notifications to allow your server-side code to reach out via broadcast to millions of devices when important events occur within your application.

It's important to the reader of this book to note that, even though these services are marketed as services geared to mobile devices, many of them can be extremely useful in developing highly interactive Web 2.0 applications.

Cloud Services

Cloud Services are run time containers for executable code that are often used to create the back end service layer for applications that may be running on desktops, mobile devices or even other servers, and provide an alternative to managing a VM to applications built in a way that allows them to be deployed directly into Azure services. These services have commonly been used to create application components known either as Worker Roles or Web Roles. Worker Roles are application components that run as an independent process executing a task with no user interface (UI), while Web Roles share mostly the same run time characteristics as a Worker Role but add being publicly addressable on the Internet in order to serve either as a UI or public service API. Cloud Services are an integral part of many enterprise applications but will not be required for many small to medium-sized websites because Azure provides capabilities to meet their needs within the Azure Websites product.

Data Services

One of the driving forces behind the success of Azure is that Microsoft has accounted for the fact that there is no one answer to “What is the best tool for ___?” Instead, many choices are available for accomplishing most tasks to allow architects to review the need in full context and identify the best (or sometimes just the least bad) way to solve the challenges that are presented when developing modern applications in the cloud. This is extremely apparent when looking at the choices made available when selecting how the application will store and retrieve data. Architects can examine such questions as:

- How quickly does the data need to be retrieved?
- For how long does it need to be stored?
- How much data will there be?
- How consistent will the structure of the data be?
- How searchable does the data need to be?

For those questions and more, a variety of storage tools is available to provide a tailored solution for the needs of your application—either through bringing a single tool to bear or potentially building a solution that combines the strength of multiple tools.

Azure Storage

Azure Storage is actually a subset of the Azure data services that encompasses a few different types of storage mechanisms itself, each aimed at solving a different storage problem:

- Blob storage is used to store unstructured data and can be considered analogous to the file system on your desktop PC. Because not all files have the same needs when it comes to being optimized for the cloud, blob storage is further divided into two types:
    - Block blobs are used to store files that can be accessed sequentially and are not subject to frequent updates. Some common examples of where Block blobs are leveraged include video files and images.
    - Page blobs are used to store files that need random access or are updated frequently.
- Azure Table Storage provides a highly scalable NoSQL database option. Data stored in Azure Tables can be accessed via software development kits (SDKs) for many popular languages or via a REST API where the use of an SDK is not required.
- Azure Queues are used to store data which is intended to be consumed by another component within the solution and does not need to be persisted after it is consumed by the receiving component. Queuing’s importance to a cloud application will be discussed in greater detail later in this book.

Azure SQL Database

Azure SQL Database can be reasonably thought of as Microsoft SQL Server’s extension to the cloud. With some features that could not scale well to a cloud solution removed, most on-premises applications built with Microsoft SQL Server will transfer to Azure SQL Database without any modification.

HDInsight

HDInsight is a service based on the Apache Hadoop platform. HDInsight is used to store and access large quantities of structured and unstructured data (commonly known as Big Data). A common use case for this tool is for data to be warehoused for analysis with business intelligence tools such as PowerPivot.

Azure Cache

Azure Cache is used as a transient store to keep data that may be expensive to retrieve and is used frequently within the application close to where it is needed without having to again incur the cost of subsequent retrievals when it is needed again. Azure's solution can either keep the cache within the application's worker role or utilize it as a shared cache available to other components of your solution.

Azure Backup

Azure Backup is an infrastructure tool used to provide off-site backup of your on-premises server data, providing additional features such as versioning and automated execution of data retention policies.

Azure Site Recovery

Azure Site Recovery (formerly Azure Hyper-V Recovery Manager) is an infrastructure tool used to provide off-site replication of private clouds running in on-premises data centers to a separate location and to coordinate recovery from the secondary location in case of outage at the primary.

App Services

App Services are a suite of tools that can be leveraged by application developers to add features to their solution without spending a tremendous amount of time and effort solving common problems that others have already solved. These services allow for developers to write the code that only they can write and focus on things that directly contribute to the problem that they are trying to solve with their application. A given website might need any number of these services. As with Data Services, many of the App Services have some degree of overlap with other App Services, requiring architects to thoroughly consider the strengths and weaknesses of the tools along with cost when determining the right tool for a given job.

Azure Media Services

Handling of media content is a problem area that many application developers could specialize in and make the focus of their entire career because it's a complex and dynamic problem area. In addition to the complexity of dealing with the various encoding methods and ensuring that the content is delivered in a manner compatible with the consuming agent, most applications of media content also require protection of the content from unauthorized use, monetization of authorized use, and a way to measure how the content is used. Azure Media Services provides tools and APIs to handle all of these tasks either as a stand-alone service or integrated with your solution. Examples of applications using Azure Media Services may include a broadcaster streaming a live sporting event to Internet subscribers or a content provider renting Digital Rights Management (DRM)-protected content to a customer that is viewable for a limited amount of time.

Azure Service Bus

Azure Service Bus is primarily known as a more robust queuing mechanism than Azure Storage Queues. The Service Bus is used to provide messaging between applications whether they are within the same Azure solution, part of another Azure solution, or a hybrid solution with on-premises components interacting with cloud components. Azure Service Bus allows for a loose coupling between these components that allow for massive scalability and also for flexibility if the implementation of one of the components within the system ever needs to be entirely reworked. Azure Service Bus will be discussed in a dedicated queuing chapter.

Notification Hubs

Notification Hubs provide a publish/subscribe framework for components within a solution to broadcast messages to all clients or target those messages only to specific clients. Publish/subscribe is an important pattern for scalable applications and could easily be the topic of its own book. Therefore, developers looking to move their skills to the cloud who are not familiar with this pattern should take the time to learn about it.

Azure Scheduler

Azure Scheduler is a simple tool that enables jobs to be set up for execution and run outside the execution path of your application. Scheduled jobs can either invoke an HTTP web service or can post a message to a storage queue. Typical use cases include performing data management tasks (such as preaggregating data or clearing temp storage) and batch communications with external services.

Azure BizTalk Services

As Azure SQL Database is the cloud evolution of the Microsoft SQL Server tool, Azure BizTalk Services brings the Microsoft BizTalk Server integration platform to Azure. This tool is primarily used to handle complex integration either between enterprise application platforms such as SAP and PeopleSoft or between businesses such as in electronic data interchange (EDI) transaction processing.

Visual Studio Online

Visual Studio Online is a bit of an outlier when it comes to the tools categorized by Microsoft as App Services because, rather than providing run time services to add functionality to the application, it is used for the actual creation of the application. Visual Studio Online provides tools to host source code repositories either in a centralized or distributed manner (via Git or Team Foundation Version Control), plan and track projects, manage the testing cycle, and evaluate the initial and ongoing performance of the application.

Azure Active Directory

Azure Active Directory provides authentication and authorization services, and can either stand alone or be integrated with an on-premises Active Directory through either synchronization or federation.

Multi-factor Authentication

Multi-factor authentication is used to address the problem often caused by compromised passwords by providing steps beyond just supplying a password for the user to prove their identity. These steps typically involve a system-generated temporary passcode which is sent to a device or telephone known to belong to the user whose credentials are being used through a mobile app, text message or phone call.

Network Services

Azure Network Services are aimed at providing support for IT professionals to achieve network configurations to enable and optimize applications running within their Azure account or in combination with on-premises solutions. These services are not directly applicable to building Azure Websites but are listed here and very briefly described to provide a complete picture of the tools available in the ecosystem to solution developers.

Azure ExpressRoute

Azure ExpressRoute is an advanced tool made available through network service providers that provides a direct connection to the Azure data centers rather than connecting through the public Internet. This is used when reliability, speed or security requirements justify the cost of maintaining the direct connection.

Azure Virtual Network

Azure Virtual Network allows resources hosted in the Azure data center to exist in the same logical network as on-premises resources. This is useful when building hybrid applications where components of the application must be on the same network.

Azure Traffic Manager

Azure Traffic Manager is a load-balancing solution used to control the distribution of requests that reach Azure services. This can be used to add fault tolerance and/or increase the performance of applications. For example, Azure Traffic Manager could use a balancing strategy known as round robin to cycle through a set of destinations in order to keep an even load across servers (assuming each request would produce the same amount of load as another) when used to increase performance. Azure Traffic Manager could use health monitoring to detect that an instance of your service has gone down and redirect traffic to another instance to achieve fault tolerance.

Getting Started on Azure

Before you can start developing websites on Azure, some steps need to be taken, both in setting up a development environment and in getting an Azure account. In this section, we'll cover the basics necessary to get a developer up and running on the Azure platform.

Prepare Yourself for Cloud Development

This might seem like an obvious step but the first thing as a developer that you should do when preparing to venture into developing solutions on Azure is to familiarize yourself with the tools, techniques, and patterns applicable to making a successful application in the cloud. Reading this book is one step. Another very valuable resource is the Patterns & Practices section of Microsoft's MSDN website (http://msdn.microsoft.com) which includes a collection of cloud development topics designed specifically to discuss problem and solution patterns particularly applicable to a cloud environment. Additionally, Microsoft and others sponsor many free, in-person and virtual training opportunities such as code camps and Microsoft Virtual Academy (http://www.microsoftvirtualacademy.com). Also, many popular online paid training opportunities exist.

Note: The URLs referenced in this section are current as of the time of this writing but may not remain consistent over time. If any URL is not accessible to the reader, I recommend searching for the current location using the search engine of your choice. Additionally, actions to be taken during an on-page workflow (such as sign up wizards) will be described as generically as possible because Microsoft may change steps at any point in time.

Setting Up Your Machine for Azure Development

The steps for setting up your local development environment can vary greatly depending on the tools that you will be using to create applications and the individual Azure services that will be leveraged in your solution. In this section, we'll cover the most basic setup for building websites using Visual Studio 2013.

Install Visual Studio 2013

If you don't already have it, the first step in getting your .NET-based development environment set up is to obtain and install Visual Studio 2013. Visual Studio is available in many editions, ranging from the free Visual Studio Express 2013 product to Visual Studio Ultimate 2013 which comes at a premium cost. Many sources exist comparing the features available in the different editions of the tool, so this book will not go into those details. Visual Studio 2013 can be downloaded at http://www.visualstudio.com/downloads. You will be given the option of downloading either the free Express edition or a trial version of one of the paid editions. All of the descriptions and screen shots from this book were created using Visual Studio 2013 Premium, but Visual Studio Express 2013 for Web should be sufficient for creating and working with Azure Websites.

Install the Azure SDK

With Visual Studio installed, you're ready to install the Azure tools for .NET and Visual Studio. More than one option exists for how to obtain and install the SDK but, for the purposes of this book, we'll discuss the direct download option. Azure SDK and tool downloads are available at http://azure.microsoft.com/en-us/downloads. For Visual Studio 2013, you will want to select the "VS 2013 Install" link and then run the downloaded executable and follow the prompts.

Install Command Line Tools

In addition to the SDK and Visual Studio tools, you will most likely find value in installing the Azure command-line tools which are also available at http://azure.microsoft.com/en-us/downloads. Under the "Command-line tools" heading, you will find a Windows PowerShell download and Azure command-line interface installer as well as documentation for each. The Azure command-line interface tool can be downloaded for either Windows, Mac or Linux.

Signing Up for Azure

To establish an account on Azure, you need to first have a Microsoft account. If you don't already have a Microsoft account, you'll need to create one. Assuming that you already have a Microsoft account, go to http://windows.azure.com. If you are not already logged into your Microsoft account, log in as prompted. Once logged in, select the option to sign up for Azure (at the time of this writing, this was a call to action on the page labeled "Sign up now"). This will present several options from which to select including a free trial, a pay-as-you-go option, and some prepaid options intended for businesses that anticipate higher utilizations and want to benefit from receiving volume discounts for prepaid services. For the readers of this book, I would recommend going with the free trial option when available. Once the desired option is selected, follow the prompts to provide required information and complete the registration process.

Note: Pay close attention to the details of the information that you are providing and to ensure that you understand the costs associated with services that you utilize. If you provide payment details such as a credit card number and do not select spending limits, you could find yourself surprised

by a bill received for services that you activated to explore and didn't subsequently decommission.

After completing the registration process, you can access the Windows Azure Management portal to manage your accounts, subscriptions, and services at http://manage.windowsazure.com.

Summary

In this chapter, you learned about many of the services offered on the Azure platform and focused more on those services which may be particularly valuable in the creation of Azure Websites. You also learned how to prepare yourself and your development environment to successfully build applications to run on the Azure platform. This knowledge, along with the account that you now have set up (if you didn't actually set up an account, you will need to do so in order to fully benefit from this book), will be used and built upon in the following chapters as you create and program websites on the Azure platform.

Chapter 2 Provisioning Resources

Once you have established a local development environment and an Azure account, you're ready to start provisioning services on the Azure platform. In this section, you will learn how to provision some of the services most pertinent to website development. The primary method of provisioning that will be described in all cases will be through the use of the Windows Azure Management Portal. However, PowerShell integration will also be briefly discussed to demonstrate how PowerShell can be used within the Azure ecosystem to automate many repetitive management tasks executed by DevOps.

For any resource to be provisioned, the process starts by selecting the "New" command in the Management Portal at http://manage.windowsazure.com as shown in Figure 5:

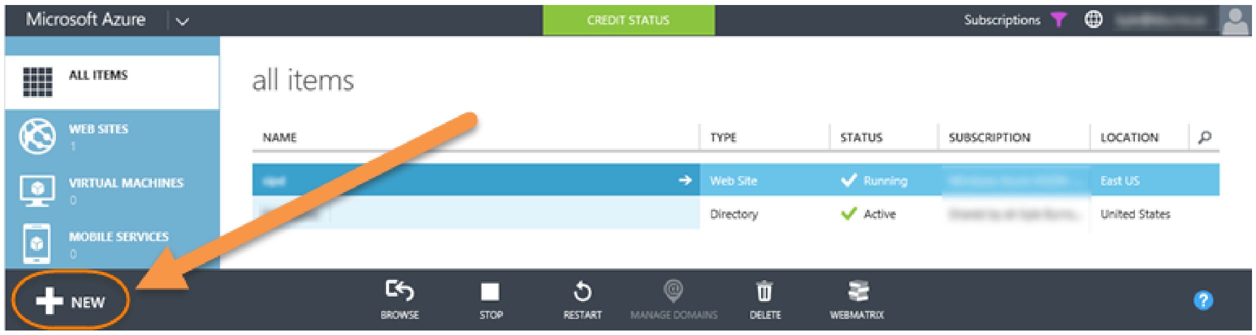


Figure 5: Create new resource

Azure Websites

To provision a new Azure Web Site using the Azure Management Portal, the first step is to determine the URL at which you want the website exposed. By default, Azure Websites will be exposed to the public with the URL format of http://**your-site**.azurewebsites.net, where **your-site** is specified by you and must be unique among Azure-hosted websites. Options exist to specify custom DNS names instead of the default naming but those options will not be discussed in this book.

While defaults will be provided for you in most cases, it is also good to put some thought into where you want your service to be hosted. Microsoft has several data centers hosting Azure services across the globe. They are currently in the following locations:

- North-Central U.S.
- South-Central U.S.
- West U.S.

- East U.S.
- Brazil
- Beijing, China
- Shanghai, China
- East Asia
- Southeast Asia
- North Europe
- West Europe
- East Japan
- West Japan

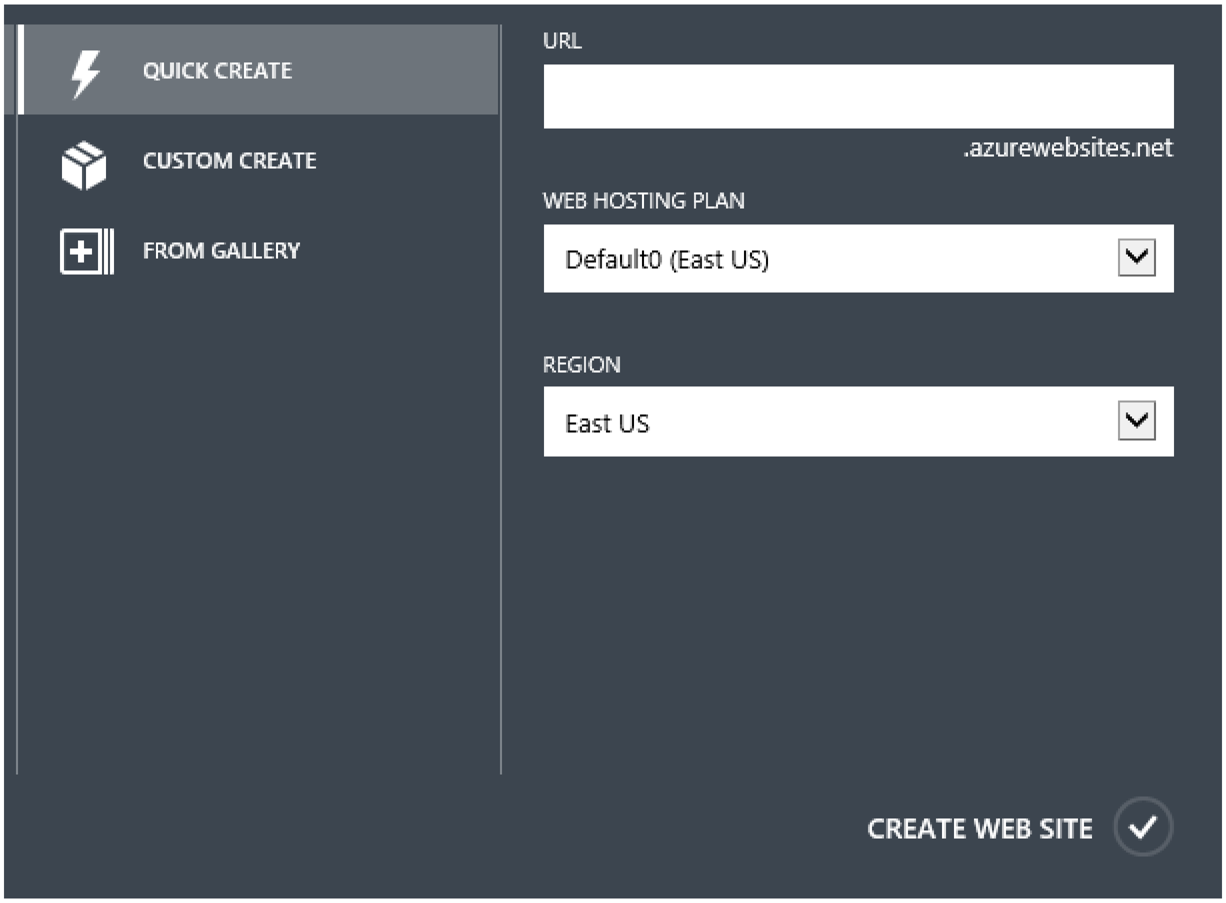
As you look through the list of regions, you may notice that, for the most part, they come in pairs which are opposite each other on the same continent. This arrangement is used by Microsoft to provide redundancy where the alternate location for the hosted services and data is still on the same continent as the primary source yet far enough away to make it unlikely that both locations would be affected by major catastrophic events such as floods and earthquakes.

When choosing a region for your Azure Web Site, the most important consideration is latency. When your Azure Web Site's intended visitors are geographically close together, hosting your Azure Web Site as close to them as possible will ensure that fewer network hops are required for the round trip to and from your Azure Web Site and provide reduced response time. It's also important to take into consideration the location of any other hosted services that your Azure Web Site will consume because locating the website in the same region as the services that it consumes will reduce latency between them and also, at times, will prevent charges incurred as data enters and leaves the Azure data center.

Once you know the desired URL and region for your website, you can begin the process of creating the website by selecting the "NEW" command within the Azure Management Portal. This command will present a hierarchical menu and you should navigate to "Compute…Web Site" in the menu. You will be presented with three options from which to choose: "Quick Create," "Custom Create," and "From Gallery".

Quick Create

When "Quick Create" is selected, additional options are presented to specify the desired URL and to optionally override the defaults which have been presented for your web hosting plan that you want attached to the site and the region in which you want it hosted. These options are shown in Figure 6. After specifying the URL and selecting "Create Web Site," your website will be provisioned within the Azure data center and prepared for you to deploy content:



URL

.azurewebsites.net

WEB HOSTING PLAN

Default0 (East US)

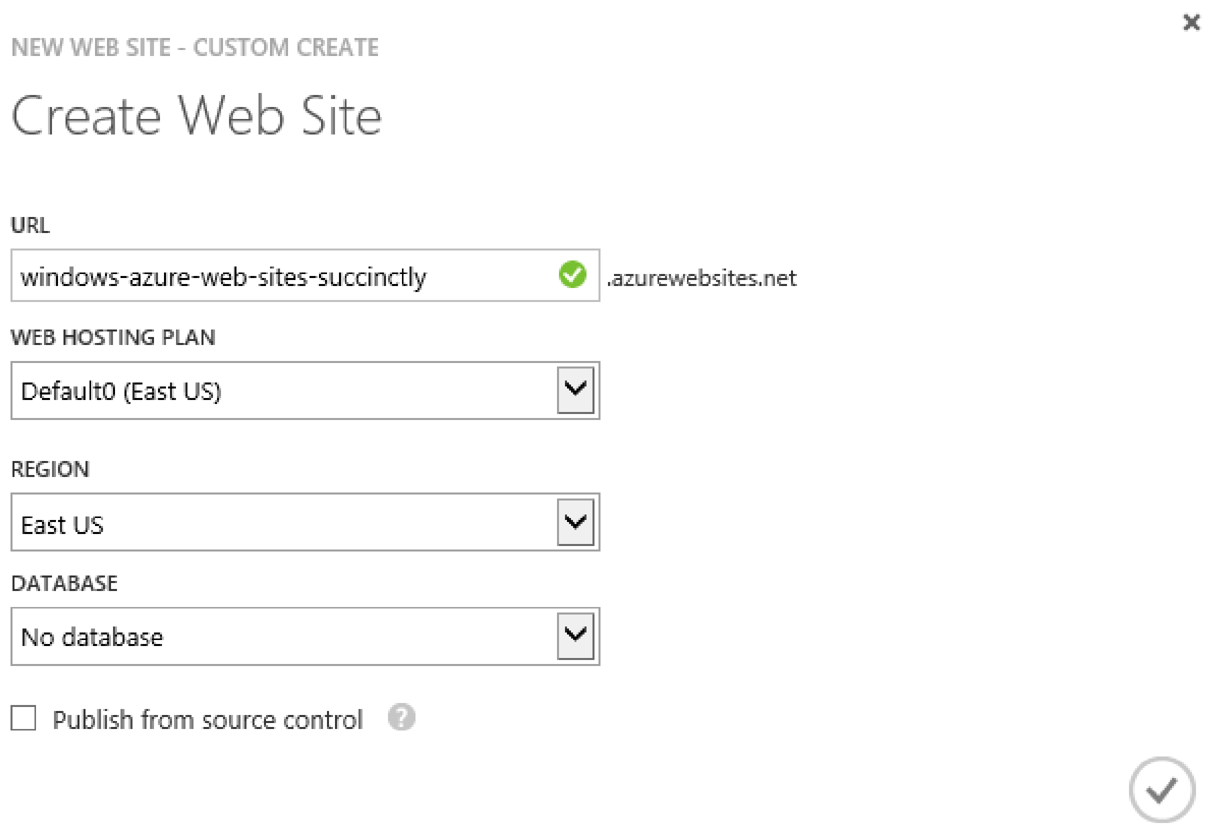
REGION

East US

Figure 6: Quick Create options

Custom Create

When “Custom Create” is selected, a dialog (shown in Figure 7) is presented allowing you, in addition to the options available for “Quick Create,” to specify a database which is available to the site (this can be a free, 20MB SQL database for light needs or a paid SQL Server or MySQL database instance) and set up the site for deployment from source control providers such as Visual Studio Online, GitHub, and others. Depending upon the options selected, additional steps may be added to the setup wizard such as selection and configuration of the source control provider and database details. After completing the wizard, your website and, optionally, database, will be provisioned. If a source control provider was selected, the website content will be deployed from the source control provider and will be ready to be viewed by website visitors.


NEW WEB SITE - CUSTOM CREATE

Create Web Site

URL: windows-azure-web-sites-succinctly .azurewebsites.net

WEB HOSTING PLAN: Default0 (East US)

REGION: East US

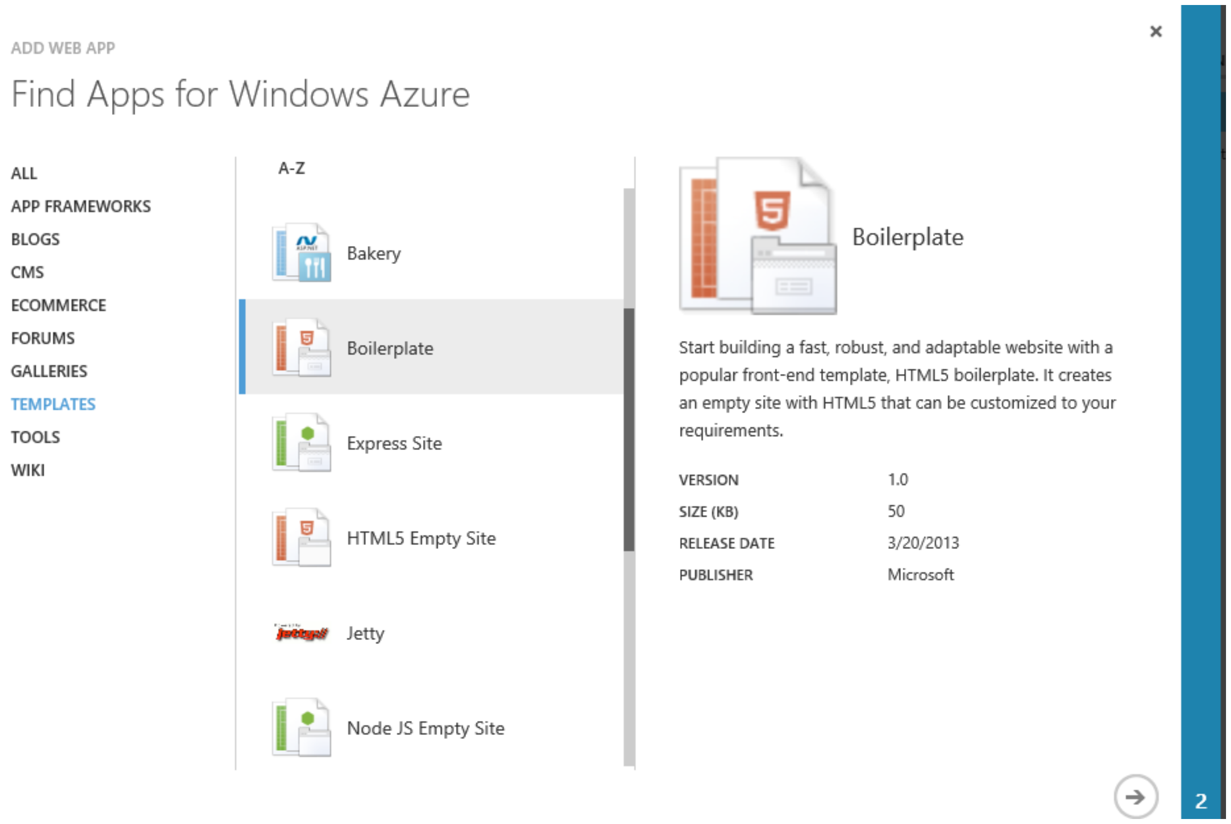
DATABASE: No database

☐ Publish from source control

Figure 7: Custom Create options

From Gallery

When “From Gallery” is selected, the Web App Gallery (shown in Figure 8) is displayed as the first page in a wizard to help configure your new website. Within the Gallery there is an ever growing list of preconfigured options that range from starter templates to jump-start, laying down the framework for your website to complete applications including popular CMS, blogging, and e-commerce frameworks. Each application available in the gallery may require different information to complete the setup and configuration, so the wizard will dynamically walk you through collecting the information that is applicable to your website. Upon completion of the wizard, your website will be fully provisioned and ready, either for you to finish building upon the starter template or for visitors to start consuming your content.


ADD WEB APP

Find Apps for Windows Azure

Boilerplate

Start building a fast, robust, and adaptable website with a popular front-end template, HTML5 boilerplate. It creates an empty site with HTML5 that can be customized to your requirements.

VERSION	1.0
SIZE (KB)	50
RELEASE DATE	3/20/2013
PUBLISHER	Microsoft

Figure 8: Web App Gallery

Storage

A common need for Azure Websites is additional storage beyond the free 20MB SQL Database that is provided. In cases where this additional storage is needed, Azure Storage is often used because of its flexibility, scalability, and low cost in comparison with other data services. As with Azure Websites, the first step in provisioning a new storage account is to decide the URL at which the storage account will be accessible. By default, Azure Storage accounts will be exposed to the public with the URL format of http://**your-site**.core.windows.net, where **your-site** is specified by you and must be unique among Azure-hosted storage accounts. The difference in subdomains leaves open the convention, if desired, of using the same value for the **your-site** portion of the URL in both your Azure Web Site and Azure Storage account.

Aside from the URL, the other two pieces of information that must be supplied when creating a Azure Storage account are the Location/Affinity Group and Replication options desired. By default, the Location/Affinity group options will only include Location (the data center in which your data will reside), but you have the ability in the Management Portal to create logical containers called Affinity Groups. Affinity Groups are used to instruct Azure that certain Compute and Storage resources should be grouped close to each other within the data center in order to reduce the network hops between them, reducing transfer cost and latency. Replication options determine how the data is replicated to balance performance, cost, and durability. These include the following:

- Geo-redundant: This option is selected by default and specifies that the data is replicated to a geographically separated data center in addition to having multiple copies (three to be precise) stored within the local data center. The copying of the data between data centers provides protection against losing data if a man-made or natural disaster were to destroy the local data center or data housed there.
- Read access geo-redundant: This option is similar to the geo-redundant option but adds the ability to provide direct read-only access to the copies of the data housed on the separate store. This can provide fault tolerance and allow for load balancing across the resources.
- Locally redundant: This option provides the option to forego the replication of the data to a secondary data center. While this may seem undesirable (there's a reason why it is not the default), in certain circumstances applications may not require the additional redundancy. This option offers a discounted pricing structure over geo-replicated storage. Additionally, data transfer using a locally redundant account is faster than that of a geo-replicated account.

When you know the configuration options that you will specify for your Storage account, you can provision the account by selecting the "NEW" command in the Azure Management Portal and navigating the menu hierarchy to "Data Services…Storage…Quick Create." Specify the desired options and select "Create Storage Account" and Azure will provision your new storage account.

Other Services

SQL Database, Service Bus Queues or any of the other services offered on the Azure platform can be provisioned for use by your application by selecting the "NEW" command in the Azure Management Portal, navigating the hierarchy to find the service, and then following intuitive wizards intended to gather configuration options for the provisioned service.

Summary

In this chapter, you learned how to make services available within your Azure account. Depending on the options selected, many of the services that you may have created will not yet be useful or be in a state where you would want anyone to come visit your website. In the coming chapter, you will learn how to develop and deploy content and code to be hosted within these services.

Chapter 3 Scaling Up, Scaling Out

An important characteristic of a cloud application is the ability to respond to increased demand by scaling the application either by scaling up or scaling out. Scaling up refers to increasing the capacity of the server (such as adding CPU cores or memory) to meet increased demand while scaling out refers to adding new instances to the application to handle the increased demand. While the scaling out option is preferred in a cloud environment because it can be achieved without reconfiguring or interrupting existing instances of the application, both methods are available within Azure.

Note: Scaling options are seriously limited within Azure for free services such as the free Azure Websites hosting plan mode. If you change your hosting plan mode to experiment with scaling features, be sure that you return it to free mode to avoid incurring unexpected charges. Many of the scaling options can be explored in the Azure Management Portal by making changes and then selecting the Discard command instead of Save.

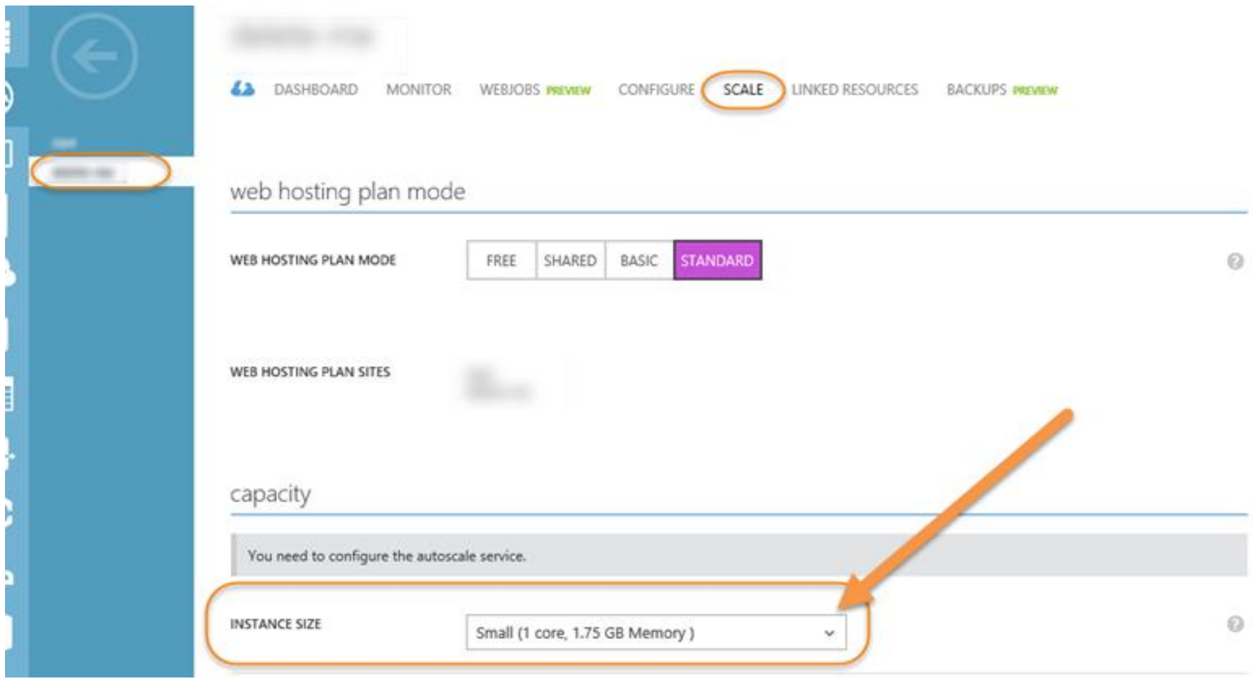
Sizing Instances

Azure offers three options for the sizing of VMs hosted with the Basic or Standard web hosting mode (other options are available for VMs which are not exposed through the Websites feature). These options include:

- Small instances featuring a single processor core and 1.75GB memory
- Medium instances featuring two processor cores and 3.5GB memory
- Large instances featuring four cores and 7GB memory

As noted earlier, preference should be given to using the smallest appropriate instance size for your application and scaling out when demand warrants it. One potential use case for scaling up rather than out would be if a third-party component is being used that is licensed per instance and adding more instances would create a significant increase in the licensing costs.

To change the instance size for an Azure Web Site hosted using a supported tier, select the site in the Azure Management Portal by clicking on the website name, which will bring up details for the selected website. Use the top navigation to navigate to the “Scale” tab and find the “Instance Size” drop-down menu (shown in Figure 9) and use it to specify the desired instance size. After you’re satisfied with your changes, select Save at the bottom of the window to persist your changes and Azure will initiate resizing of the instance.



web hosting plan mode

WEB HOSTING PLAN MODE: FREE | SHARED | BASIC | [✓ STANDARD]

WEB HOSTING PLAN SITES

capacity

You need to configure the autoscale service.

INSTANCE SIZE: Small (1 core, 1.75 GB Memory)

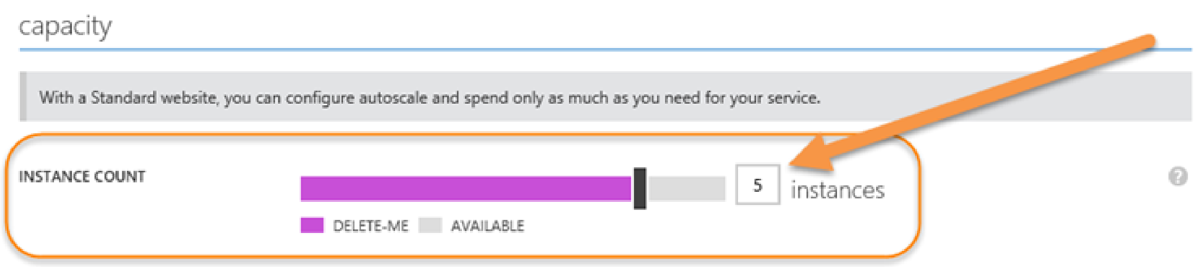
Figure 9: Scale up

Adding/Removing Instances

Azure Websites using the Shared, Basic or Standard pricing tier have the ability to scale out by adding and removing instances of the same size to the resource pool serving the site. The number of available instances for scaling per site varies depending on the specific hosting plan tier:

- Shared: Sites can scale to a maximum of six shared instances (no dedicated Compute resources)
- Basic: Sites can scale to a maximum of three dedicated VMs
- Standard: Sites can scale to a maximum of 10 dedicated VMs

To scale out an Azure Web Site with a supported hosting plan tier, navigate to the Scaling options for the site as described in the previous section and find the Instance Count slider depicted in Figure 10. Either drag the slider to the desired value or type the selected value into the text box and then select Save to persist the configuration. After changes have been saved, the Azure fabric controller will work in the background to provision or decommission instances to reflect the newly selected instance count.


capacity

With a Standard website, you can configure autoscale and spend only as much as you need for your service.

INSTANCE COUNT 5 instances

🟪 DELETE-ME ⬜ AVAILABLE

Figure 10: Scale out

Auto-scaling

The scaling mechanisms that have been discussed so far have focused around the person who is managing an Azure account, logging into the Azure Management Portal application, and manually either changing instance sizes or adding/removing instances through the UI. But Azure also makes available a powerful feature to help balance cost and performance by automatically adjusting the number of instances that are running based on a few available controlling factors. This feature is called "Auto Scale" and, for Azure Websites, the controlling factors can be either based on time or CPU utilization percentage.

Schedule-based scaling

When controlling scaling based on time, the Azure Web Site operator can take advantage of knowing when periods of particularly high or low utilization will occur, and then make sure that resources are added or removed from the solution to respond to these periods.

One example of where schedule-based scaling would be used is the usage pattern of an enrollment website for a health insurance company. Most insurance companies have set periods of time on the calendar that mark open enrollment, the period in which consumers of the insurance product can obtain new insurance policies or make changes to existing coverage. Outside of these periods, consumers must meet very stringent criteria to be eligible for such changes. Let's assume, for illustrative purposes, that an insurance company with two million subscribers has an open-enrollment period lasting two weeks. During those two weeks, it can be expected that the company's enrollment website will be actively used by nearly all two million subscribers as they examine options and select the coming year's benefits. Outside of that two-week period, the website will be lightly used by a few subscribers at a time as they are newly hired or have other events that make them eligible for changes outside the enrollment period. With an Azure Web Site at the Standard hosting tier, an instance count of 10 can be specified for during the enrollment period and 2 for all other times.

Default Schedules

By default, Azure's auto-scale feature allows you to enable schedules that establish unique periods for day, night, weekday and weekend day. The day/night divide is configurable to provide control over starting and ending time for the day period as well as the time zone that should be used as the point of reference. This feature may be used to ensure that server resources are added during the working day and then deallocated during off hours.

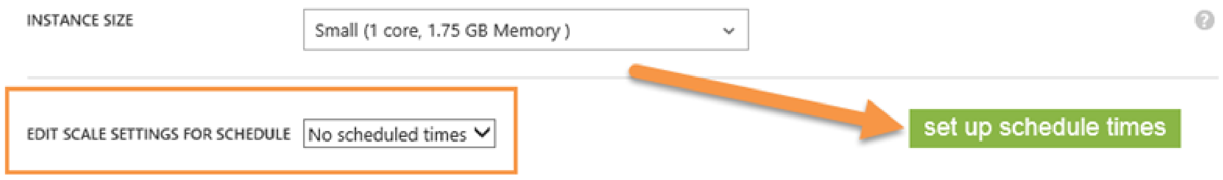
A great example of how the default schedules could save an Azure subscriber money on their bill is a scenario that was described to me when I was first introduced to Azure. The scenario involved a customer who processed trading data from the stock market, which required a lot of processing power during the trading day but literally dropped to needing no processing as soon as the markets closed every day. The auto-scale feature didn't exist at the time so the subscriber wrote PowerShell scripts which were scheduled on-premises to provision and deallocate Azure Compute instances on a schedule that allowed their Azure costs to be close to zero during the time periods where the processing power was not needed and handled the massive load during the day.

Custom Schedules

In addition to the default schedules, Azure subscribers can configure additional periods by specifying a start date/time and end date/time and providing a name for the custom schedule. This feature would be used to specify the time period during which heavy volume is expected in the open enrollment example earlier.

Configuring Schedule-based Scaling

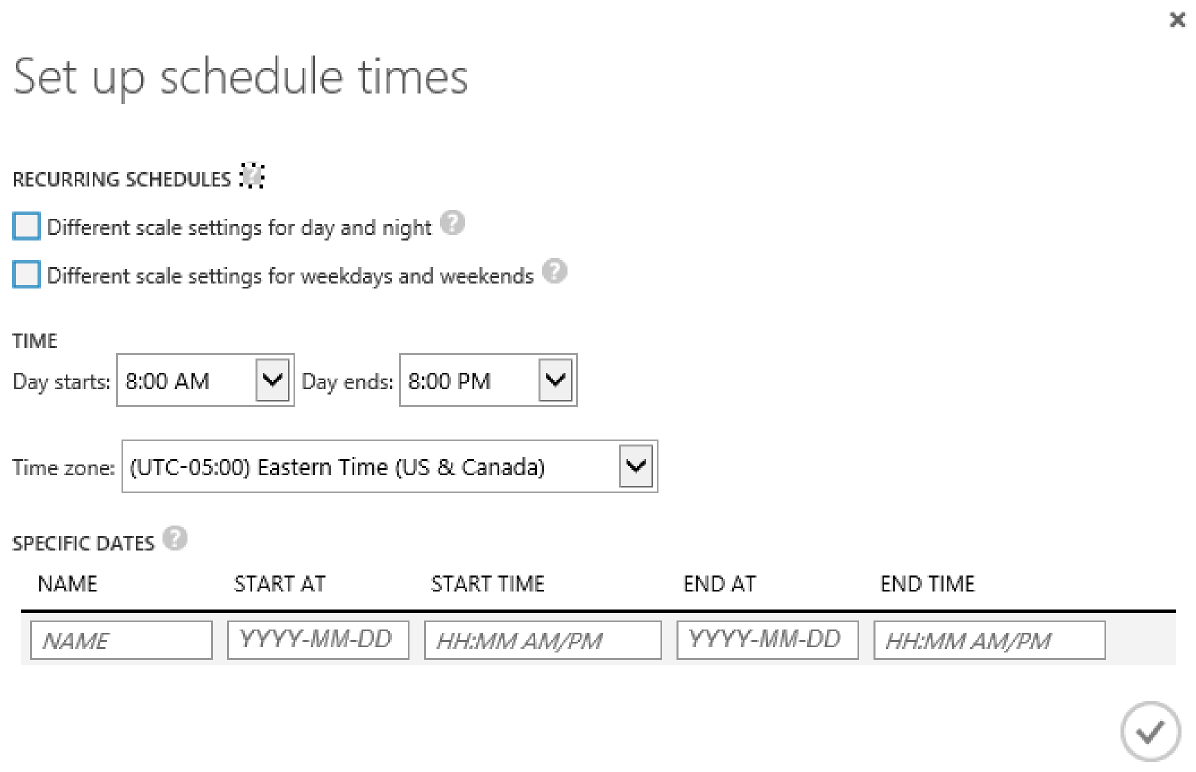
To scale an Azure Web Site that's hosted using the Standard web hosting plan tier based on schedule, select the site in the Azure Management Portal by clicking on the site name, which will bring up details for the selected site. Use the top navigation to navigate to the Scale tab and find the scheduling options, which consist of a drop-down menu that allows you to select from existing schedules and a command to create new schedule periods or edit existing ones. These are shown in Figure 11:


INSTANCE SIZE: Small (1 core, 1.75 GB Memory)

EDIT SCALE SETTINGS FOR SCHEDULE: No scheduled times

Figure 11: Scale by schedule options

Assuming that schedule periods have not previously been established, you will first invoke the "Set up schedule times" command, displaying the dialog shown in Figure 12:



Set up schedule times

RECURRING SCHEDULES

☐ Different scale settings for day and night

☐ Different scale settings for weekdays and weekends

TIME

Day starts: 8:00 AM Day ends: 8:00 PM

Time zone: (UTC-05:00) Eastern Time (US & Canada)

SPECIFIC DATES

NAME	START AT	START TIME	END AT	END TIME
NAME	YYYY-MM-DD	HH:MM AM/PM	YYYY-MM-DD	HH:MM AM/PM

Figure 12: Set up schedule times dialog

In the dialog, you have the options of enabling the default schedules for day/night and weekday/weekend as well as creating additional named periods that require specification of name, start date, start time, end date, and end time. After you have worked with the schedule periods in the dialog to create periods that are appropriate for the anticipated load profile of your website, accept the changes. You will be returned to the scale configuration page with the new scheduling options now displayed in the schedule drop-down menu as shown in Figure 13:

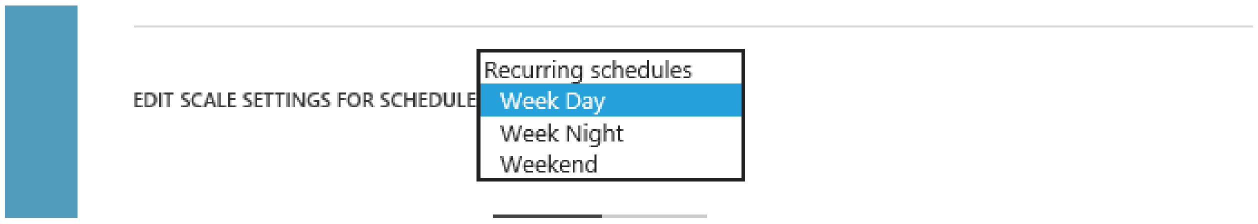


Figure 13: Select schedule

For each schedule presented in the drop-down menu, you can now select the schedule and configure the instance count desired to be active when that schedule is current. The Azure fabric scans for scaling about every five minutes, so scaling may not occur precisely at the configured time.

Scaling by Metric

Many times it is more difficult to predict scaling needs simply by saying, “I need five instances during the day and three at night.” Instead, the definition of when you need more or fewer resources should be driven by some measure that’s tied to how busy your website is. To meet this need, Azure offers the “scale by metric” option in the auto-scale feature. This measure varies by the individual service being measured. For example, some Azure Websites are measured strictly on CPU utilization percentage while other services can be measured for auto-scale based on the number of messages waiting to be processed on a queue.

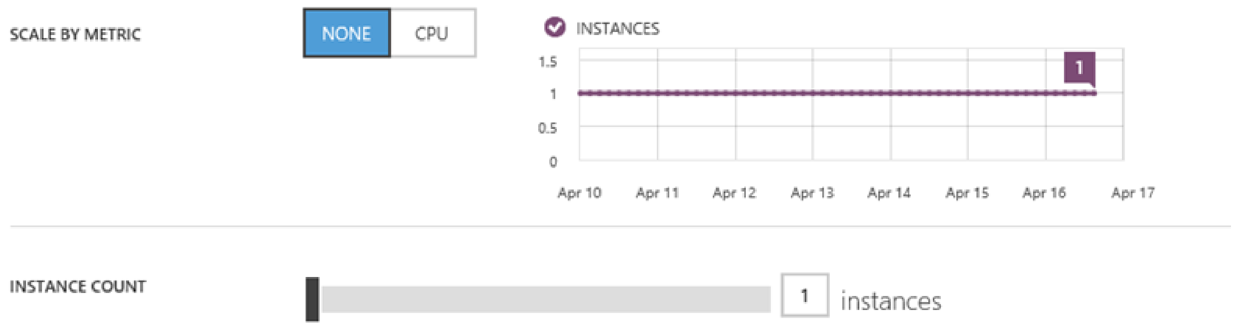
Regardless of the chosen measure, the Azure controller follows an algorithm similar to what is described in the following pseudo-code for each pass of the scheduling loop:

```
if current measure >= max threshold
    if current instance count < max instance count
        provision additional instance
    end if
else
    if current instance count > min instance count
        if current measure < min threshold
            de-allocate one instance
        end if
    end if
end if
```

The ability to scale by metric is a powerful tool that allows you to finely tune a balance between saving on your Azure bill and ensuring that your application responds well to user requests.

Configuring Metric-based Scaling

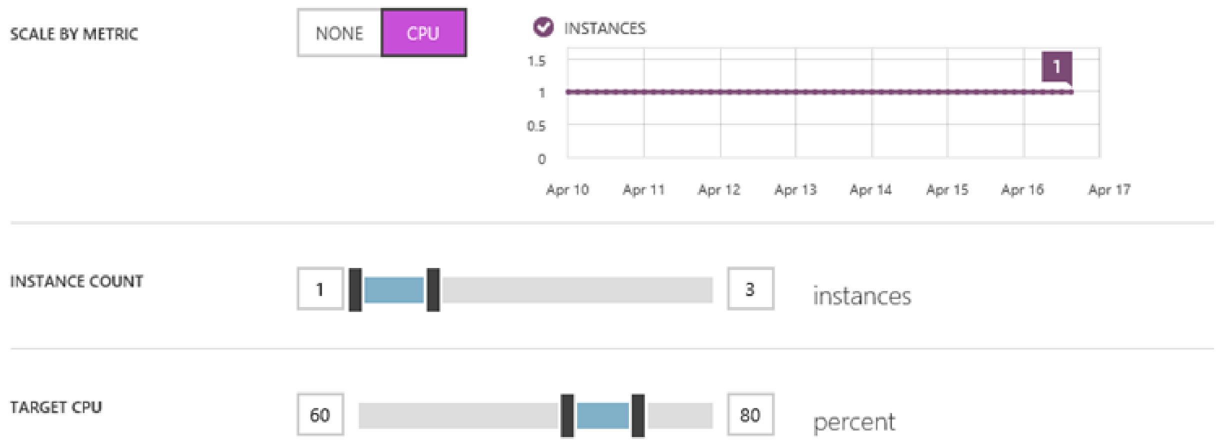
To scale an Azure Web Site hosted using the Standard web hosting plan tier based on schedule, select the website in the Azure Management Portal by clicking on the website name, which will bring up details for the selected website. Use the top navigation bar to navigate to the Scale tab and find the “scale by metric” section which is shown in Figure 14:


SCALE BY METRIC [✓ NONE] CPU

INSTANCE COUNT 1 instances

Figure 14: Scale by metric options

By default, the “None” option is selected within the options but selecting “CPU” (which is currently the only other option for Azure Websites) changes the “instance count” slider to accept a range rather than a scalar value, and a “target CPU” slider is presented which allows you to select the range of CPU utilization percentage considered to be appropriate for your application. The default range of 60 to 80 percent is generally a good value to start with, unless you are aware of specific needs of your application that drive selection of other values. The CPU scaling options are shown in Figure 15:


SCALE BY METRIC: NONE [✓ CPU]

INSTANCE COUNT: 1 – 3 instances

TARGET CPU: 60 – 80 percent

Figure 15: Scale by CPU options

Selecting Appropriate Values for Scale by CPU

The selection of appropriate scaling values may seem rather obvious and the initial answer from many people would be to include the following characteristics:

- Minimum instance count of 1 to minimize cost if load allows
- Maximum instance count of either what your plan allows for or some lower limit to meet what you’ve chosen as a maximum cost
- Minimum target CPU close to 100 percent to allow instances to drop off as soon as possible
- Maximum target CPU at 100 percent to ensure additional instances are only paid for when maximizing the utilization of the resources already being paid for

This approach seems pretty solid as a list of bullet points and will make absolute sense to somebody in charge of paying the bills. But the translation from bullet points to the real world reveals some additional variables that should be factored into the decision about what is right for you:

- Minimum instance count: Many website operators will want to ensure at least one additional instance of their website beyond what is expected to be needed to ensure rapid response to an instance dropping unexpectedly without a noticeable hit to website visitors. The adage “Three is two, two is one, one is none” applies here so, for a website with needs that justify a Standard hosting plan mode and configuration of scaling options at all, I would never recommend less than two instances.

- Maximum instance count: There’s nothing I would change about this one. The maximum instance count should either be as many instances allowed to you, or you should work out the maximum amount that you are willing to pay for Azure Web Site compute time, limiting the number of instances to a value that does not potentially have you exceeding this value.

- Minimum target CPU: This value ideally should be low enough to clearly indicate that the high-volume peak has passed on and that dropping an instance won’t immediately throw you back into a CPU utilization that will have it again provisioned on the next pass.

- Maximum target CPU: This value should be set with the understanding that sampling is done at intervals and, if allowed to be too close to 100 percent, you could spike immediately following a sampling interval, leaving your users with a floundering website while waiting for the scheduler to notice that another instance needs to be provisioned and spun up.

Combined Scaling by Schedule and Metric

The factors that should drive selection of appropriate configuration options for scaling by metric are not only more complicated than one would initially think, but can often be impacted by known periods of website activity that differ from each other in anticipated volume. During a period of low volume of traffic, it might be a reasonable, cost-saving measure to have the minimum instance count of the website be a lower value than at other times while, during a period that is anticipated to be very busy, it may be worth the additional cost to have a higher instance count in order to avoid times where the website becomes less responsive while additional instances are provisioned. Azure meets this need by allowing a separate set of Scale by Metric options to be configured for each of the time periods that you have created.

Summary

In this chapter, you learned about scaling, which is one of the most important defining features of a cloud platform. You learned the difference between scaling up and scaling out, and saw Azure features for dynamically adding and removing instances that make scaling out the scaling option of choice on the platform.

Chapter 4 Deploying Websites

It almost goes without saying that, except in cases where a gallery template fully meets your needs right out of the box, your shiny new Azure Web Site is relatively useless to you unless you are able to deploy either new content or changes to the content created from the selected gallery template. As with many tasks in Azure, there is more than one right way to accomplish deployments. In this chapter, we will examine several of these options.

Deploy via FTP

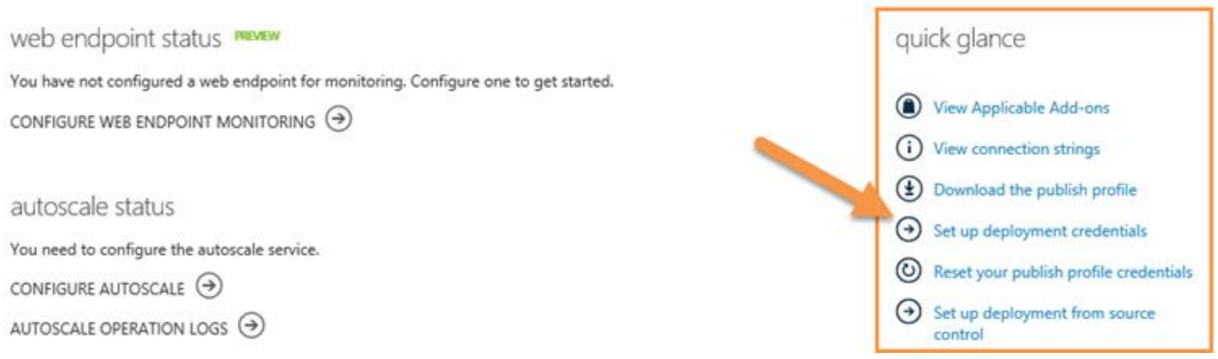
File Transfer Protocol (FTP) is a method for moving files between network-connected hosts that has been in common use for decades. As a deployment mechanism, it has many limitations that make it less desirable for large or complex deployments. But its simplicity and almost universal availability across platforms of FTP client and server software make it almost guaranteed that it will be available to you to move files.

Most operating systems in use today provide a command line FTP client through which you can enter commands to perform, including such actions as listing directory contents, uploading files, and downloading files. Many also provide a graphical client that allows you to interact with files and directories on the server in much the same manager as the native file manager application for the operating system. For the most part, however, these clients tend to be limited in features that are present in dedicated FTP clients such as automatically performing date checks to avoid download or uploading of content that has not changed since the last synchronization. Today the go-to FTP client for many people is the open source FileZilla software which can be found at http://filezilla-project.org and is distributed under the GNU General Public License.

Configuring Azure for FTP Deployment

Before you can use an FTP client to deploy content to an Azure Web Site, it must be configured for FTP access. This primarily consists of setting up the credentials which can be used to connect with the FTP client because FTP cannot use the Microsoft Account credentials that you use for most Azure management tasks. Because the credentials will be used for other deployment options that cannot use the Microsoft Account credentials, they are referred to within the Azure Management Portal as “Deployment Credentials” rather than specifically being referred to as FTP credentials.

To configure deployment credentials for an Azure Web Site, select the site in the Azure Management Portal by clicking on the site name, which will bring up details for the selected site. Use the top navigation bar to navigate to the Dashboard tab and find “Set up deployment credentials” in the quick glance section which is shown in Figure 16:


web endpoint status PREVIEW

You have not configured a web endpoint for monitoring. Configure one to get started.

CONFIGURE WEB ENDPOINT MONITORING

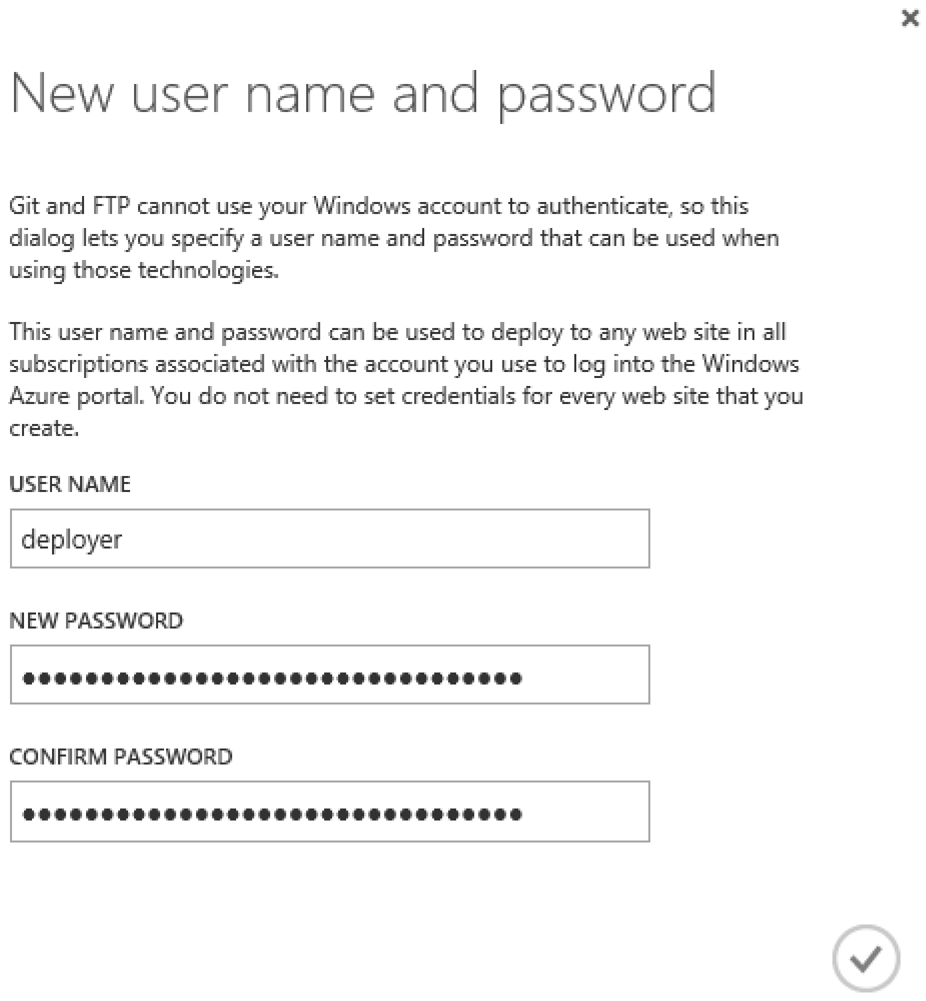
autoscale status

You need to configure the autoscale service.

CONFIGURE AUTOSCALE
AUTOSCALE OPERATION LOGS
quick glance
View Applicable Add-ons
View connection strings
Download the publish profile
Set up deployment credentials
Reset your publish profile credentials
Set up deployment from source control

Figure 16: Dashboard deployment credentials command

The Management Portal will respond by displaying the dialog shown in Figure 17 which prompts you for the username and password to be used for deployment via either Git or FTP. Enter a username and password and save changes:


New user name and password

Git and FTP cannot use your Windows account to authenticate, so this dialog lets you specify a user name and password that can be used when using those technologies.

This user name and password can be used to deploy to any web site in all subscriptions associated with the account you use to log into the Windows Azure portal. You do not need to set credentials for every web site that you create.

USER NAME: deployer

NEW PASSWORD: ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

CONFIRM PASSWORD: ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

Figure 17: Set deployment credentials

Your Azure Web Site is now ready to accept connections from the FTP client of your choice to deploy assets.

Obtaining Connection Information for FTP Client

Once your Azure environment is configured with deployment credentials, you need to know where to point your FTP client to interact with the Azure FTP server. This information, along with the credentials themselves, can be obtained by downloading a file called "Publish Profile" from the Azure Management Portal. To obtain deployment credentials for an Azure Web Site, select the site in the Azure Management Portal by clicking on the site name which will bring up details for the selected site. Unless you have selected the option to skip the Quick Start screen, the Quick Start screen will be displayed. You can find the "Download publish profile" command in the "Publish your app" section shown in Figure 18:

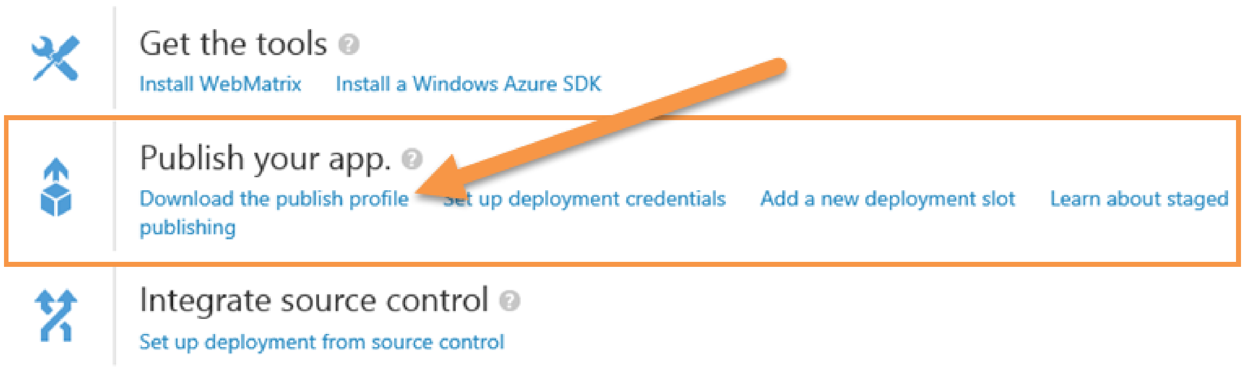

Get the tools
Install WebMatrix
Install a Windows Azure SDK
Publish your app.
Download the publish profile
Set up deployment credentials
Add a new deployment slot
Learn about staged publishing
Integrate source control
Set up deployment from source control

Figure 18: Quick Start download publish profile

If you have already selected the option to skip the Quick Start screen, the "Download the publish profile" command can be found on the Dashboard tab under quick glance as shown in Figure 19:

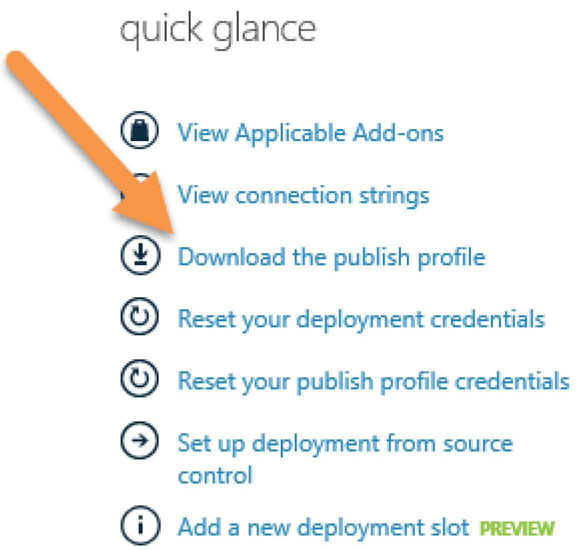

quick glance
View Applicable Add-ons
View connection strings
Download the publish profile
Reset your deployment credentials
Reset your publish profile credentials
Set up deployment from source control
Add a new deployment slot PREVIEW

Figure 19: Dashboard download publish profile

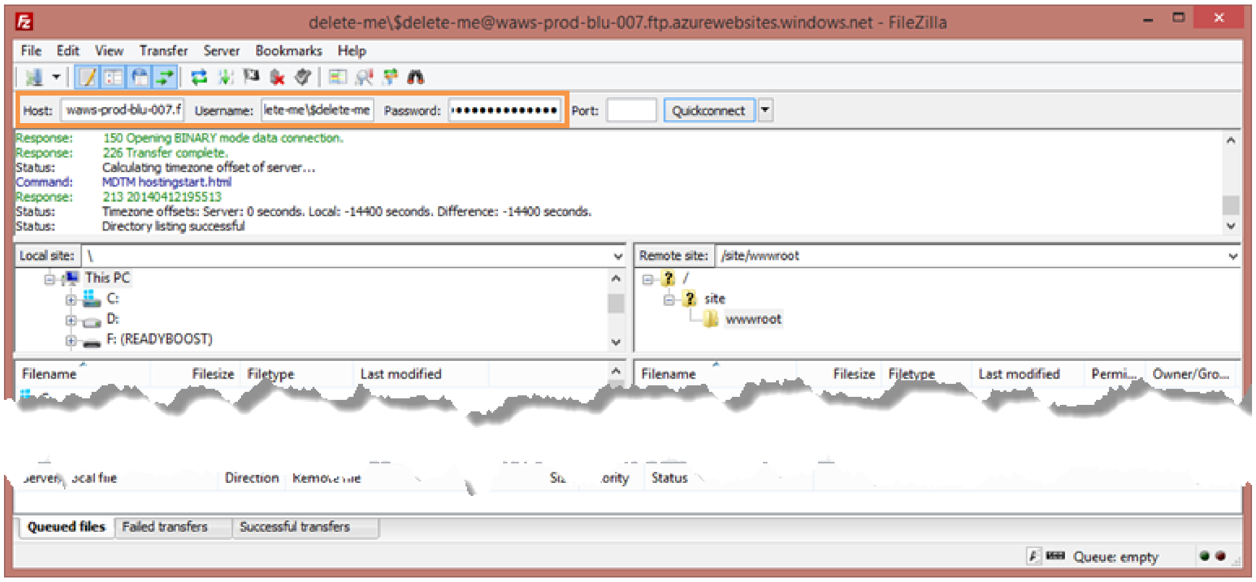
Regardless of where you find the command, the Azure Management Portal will respond by serving up a file named ***`your-site`*****`.azurewebsites.net.PublishSettings`** where ***`your-site`*** is the name of your website. This file is an XML document that includes sections for various deployment methods that are each contained in a node called **`publishProfile`**. The XML format for a **`publishProfile`** node is as follows:

```
<publishProfile profileName="delete-me - FTP"
      publishMethod="FTP"
      publishUrl="ftp://waws-prod-blu-
007.ftp.azurewebsites.windows.net/site/wwwroot"
      ftpPassiveMode="True"
      userName="delete-me\$delete-me"
      userPWD="[omitted]"
      destinationAppUrl="http://delete-me.azurewebsites.net"
      SQLServerDBConnectionString=""
      mySQLDBConnectionString=""
      hostingProviderForumLink=""
      controlPanelLink="http://windows.azure.com">
    <databases/>
</publishProfile>
```

The appropriate **`publishProfile`** node for finding the FTP connection information is the one where the **`publishMethod`** attribute has a value of “FTP”. The **`publishUrl`** node contains the URL of the root directory for publishing website content and the credentials are contained in the **`userName`** and **`userPWD`** (password) attributes. You’ll notice that these differ from the actual credentials entered when setting up the deployment credentials. This is because Azure uses the information that you use as a seed to generate the actual credentials.

Note: It’s important to remember that the credentials contained in this file will grant whomever obtains them the ability to control the content on your website, so the PublishSettings file should be kept safe. FTP is not a secure protocol so I would also recommend frequently changing the deployment credentials and generating a new PublishSettings file for whatever application you’re using.

Once you have obtained the appropriate URL and credentials, you can connect to the website using your preferred FTP client and website files. Figure 20 demonstrates connecting with the FileZilla FTP client:


delete-me\$delete-me@waws-prod-blu-007.ftp.azurewebsites.windows.net - FileZilla
File Edit View Transfer Server Bookmarks Help
Quickconnect
Local site: \
This PC
C:
D:
F: (READYBOOST)
Remote site: /site/wwwroot
/
site
wwwroot
Filename Filesize Filetype Last modified
Filename Filesize Filetype Last modified Permi... Owner/Gro...
Direction
Status
Queued files
Failed transfers
Successful transfers
Queue: empty

Host: waws-prod-blu-007.f Username: lete-me\$delete-me Password: •••••••••••••• Port:

```
Response:   150 Opening BINARY mode data connection.
Response:   226 Transfer complete.
Status:     Calculating timezone offset of server...
Command:    MDTM hostingstart.html
Response:   213 20140412195513
Status:     Timezone offsets: Server: 0 seconds. Local: -14400 seconds. Difference: -14400 seconds.
Status:     Directory listing successful
```

Figure 20: FileZilla FTP connection

Note: In addition to dedicated FTP clients, some web integrated development environments (IDEs) such as Adobe Dreamweaver feature integrated FTP management tools that allow you to manage and deploy assets from within the development environment.

Resetting the Publish Profile and Deployment Credentials

Username/Password credentials are only as secure as the ability to keep them secret and that can often be hard to do. PublishSettings files can unwittingly be made available on a network drive, trusted employees can suddenly become disgruntled former employees, and vulnerabilities such as OpenSSL Heart Bleed are discovered with alarming regularity. Sometimes you might even find that you’ve forgotten to obscure your credentials when capturing screen shots for a blog entry or a book. These factors all support the assertion that even the most secure credentials must be changed on a regular basis if they are protecting anything of importance.

When deployment credentials have been previously established, two additional commands are made available within the Azure Management Portal. These commands are:

- Reset your deployment credentials: This command establishes new "seed" values to be used for the username/password.
- Reset your publish profile credentials: This command generates new values for the actual credentials used in the publish profile. After using the command, the information in existing PublishSettings files will no longer be able to be used to connect and deploy to your website. So, if multiple people are deploying to the same website, it is important to communicate when the PublishSettings files that each individual has is no longer current.

To access the commands used to change the credentials used to deploy content, navigate to the Quick Start or Dashboard screen as described earlier. The "Reset your deployment credentials" command appears in the Quick Start screen in the "Publish your app" section and both commands appear on the Dashboard screen in the quick glance section.

Deploy from Visual Studio

While FTP deployment is an option for cross-platform compatibility and opens up the broadest range of tools that you can use to manage your website (every major operating system in use today includes an FTP client and a simple text editor), there's a very good chance that the majority of people creating websites on the Azure platform will at least have access to Visual Studio and will want a more user-friendly approach to development and deployment than UI and command-line FTP (not that there's anything wrong with that). For developers using Visual Studio 2013, deployment of websites hosted in Azure is well integrated into the developer workflow. In this section, we will cover some of the available options for deployment using Visual Studio.

Copy Web Site

One option that you can leverage to deploy websites from Visual Studio is to use the "Copy Web Site" command to initiate an FTP-based copy of the contents. This command can be found in Visual Studio either as a sub item of the "Website" menu as shown in Figure 21 or by right-clicking on the project for the website within the Solution Explorer as shown in Figure 22:

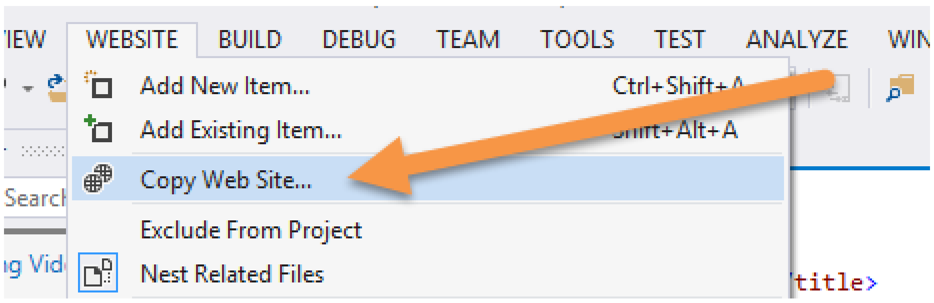


Figure 21: Copy Web Site from website menu

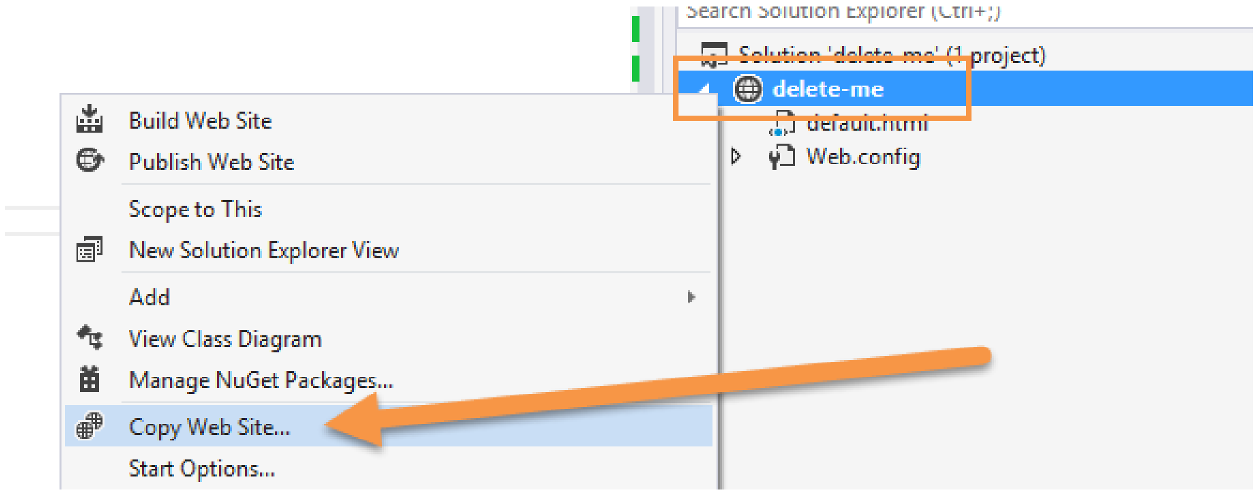
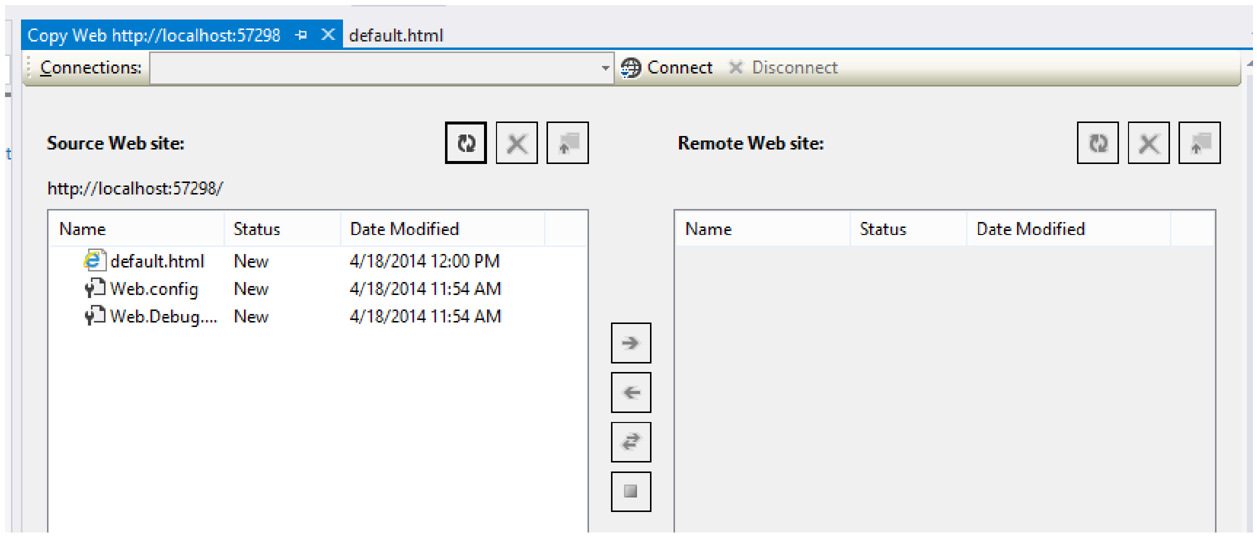


Figure 22: Copy Web Site context menu

Regardless of how "Copy Web Site" is invoked, Visual Studio displays a window that includes the files within the web project in a "Source Web Site" section and a "Remote Web Site" section that is initially empty. This interface, which is shown in Figure 23, looks very similar to many popular graphical FTP clients because it essentially has the same job:

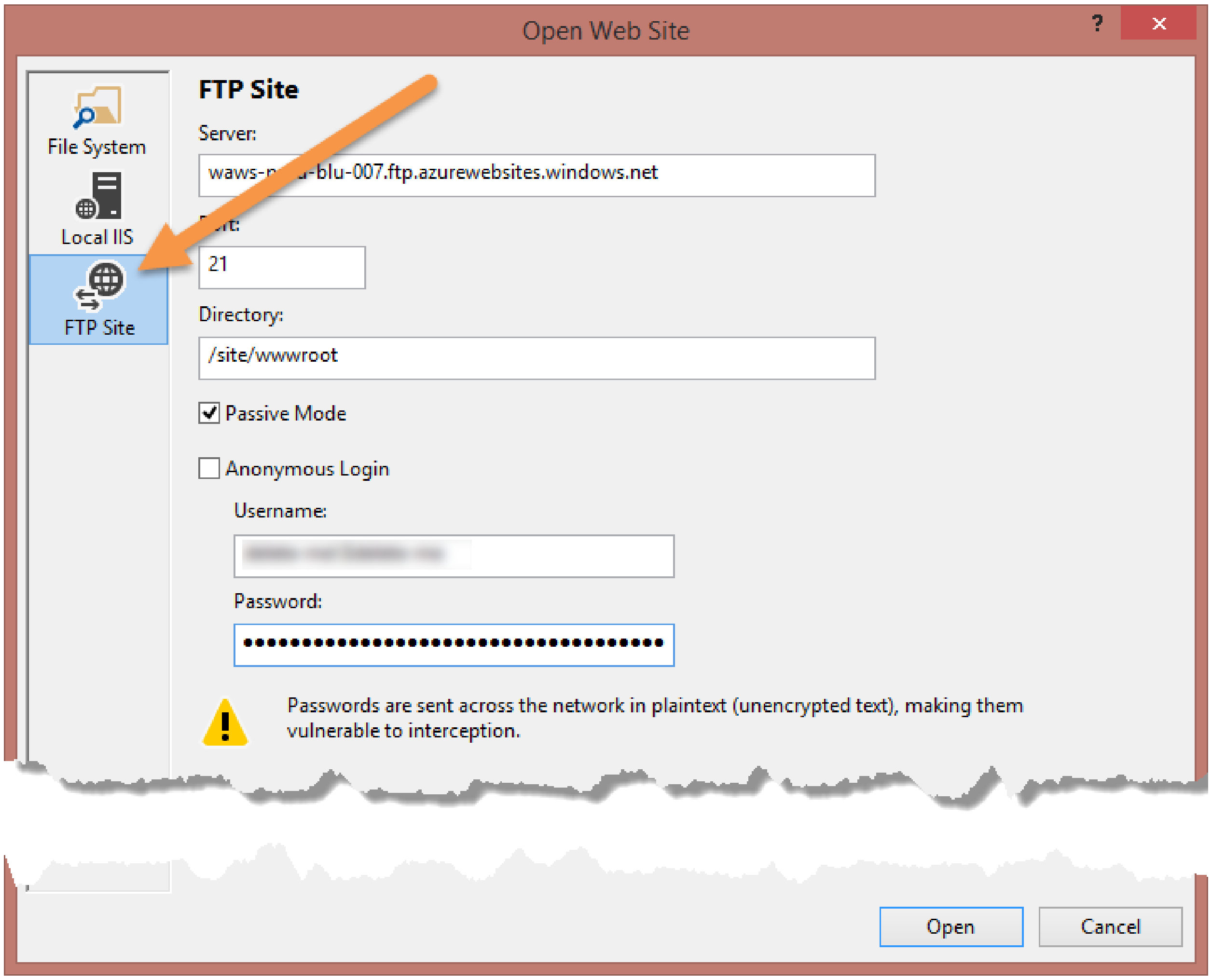


Name	Status	Date Modified
default.html	New	4/18/2014 12:00 PM
Web.config	New	4/18/2014 11:54 AM
Web.Debug....	New	4/18/2014 11:54 AM

Name	Status	Date Modified

Figure 23: Copy Web Site UI

From the Copy Web Site UI, the first thing that you need to do is establish a connection to your Azure Web Site using the information obtained from your PublishSettings file. This is initiated by invoking the Connect command which is next to the Connections drop-down menu at the top of the window. This displays a dialog that allows you to connect to either a folder on your local file system, a local IIS instance or an FTP server. Select FTP and fill in the connection information fields using values from your PublishSettings file to appear similar to what is seen in Figure 24. For the Server and Directory fields, you will have to extract the host name and path values from the **`publishUrl`** attribute:



FTP Site

Server: waws-p[illegible]u-blu-007.ftp.azurewebsites.windows.net

Port: 21

Directory: /site/wwwroot

☑ Passive Mode

☐ Anonymous Login

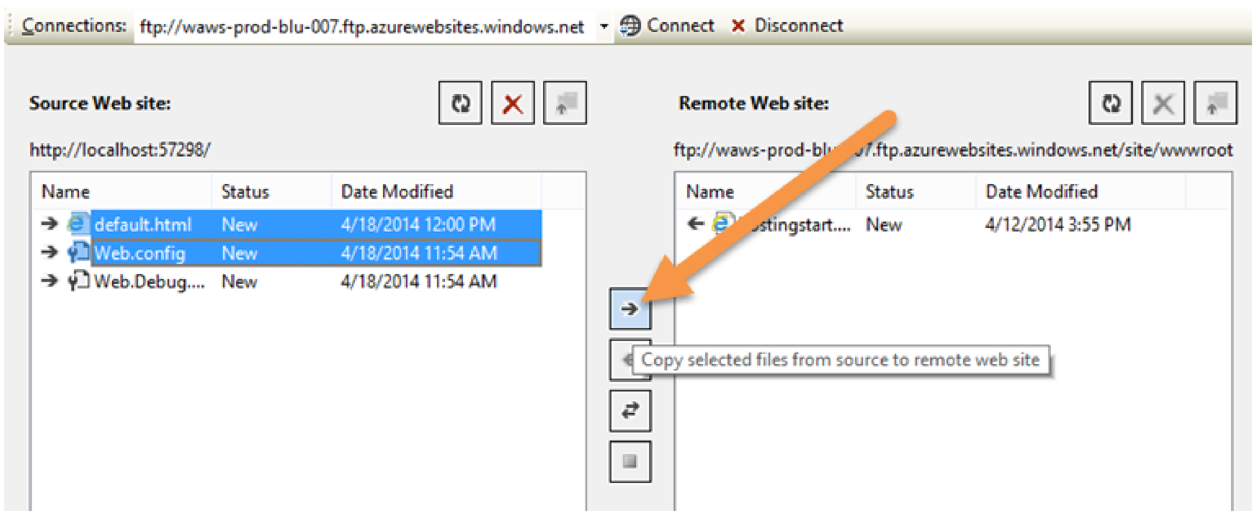
Username: [redacted]

Password: ••••••••••••••••••••••••••••••••••••••

Passwords are sent across the network in plaintext (unencrypted text), making them vulnerable to interception.

Figure 24: Copy Web FTP connection

Once connection information and credentials have been supplied, accept the entries by selecting the Open command. Visual Studio will then connect via FTP to Azure and the web root will be displayed in the “Remote Web Site” section. From this point, you manually select files to synchronize and use the commands located between the Source and Remote Web Site sections to perform the synchronization. Figure 25 demonstrates copying files from the Source Web to the Remote Web:


Connections: ftp://waws-prod-blu-007.ftp.azurewebsites.windows.net Connect Disconnect

Source Web site: http://localhost:57298/

Name	Status	Date Modified
default.html	New	4/18/2014 12:00 PM
Web.config	New	4/18/2014 11:54 AM
Web.Debug....	New	4/18/2014 11:54 AM

Remote Web site: ftp://waws-prod-bl…7.ftp.azurewebsites.windows.net/site/wwwroot

Name	Status	Date Modified
…stingstart....	New	4/12/2014 3:55 PM

Figure 25: Copy files to remote

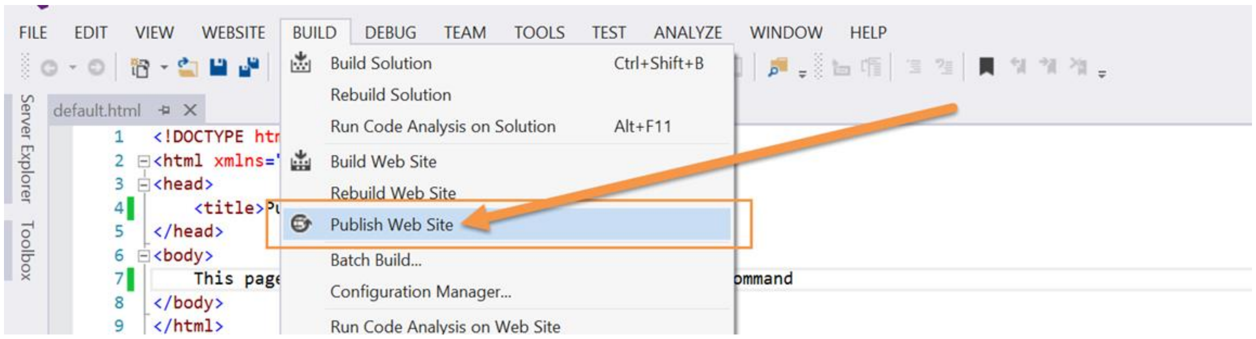
One thing that is very important to remember when using the Copy Web Site tool is that it is essentially just an FTP client integrated into Visual Studio. Therefore, many advanced tasks such as integrated build and configuration transforms cannot be achieved with this tool. This makes the tool only suitable for the simplest of websites and it is not even available for Web Application projects.

Publish Web Site

Note: The tooling for Publish Web Site is not integrated into the free Express versions of Visual Studio so, if you have chosen to or must use Express, this section will not be applicable to you.

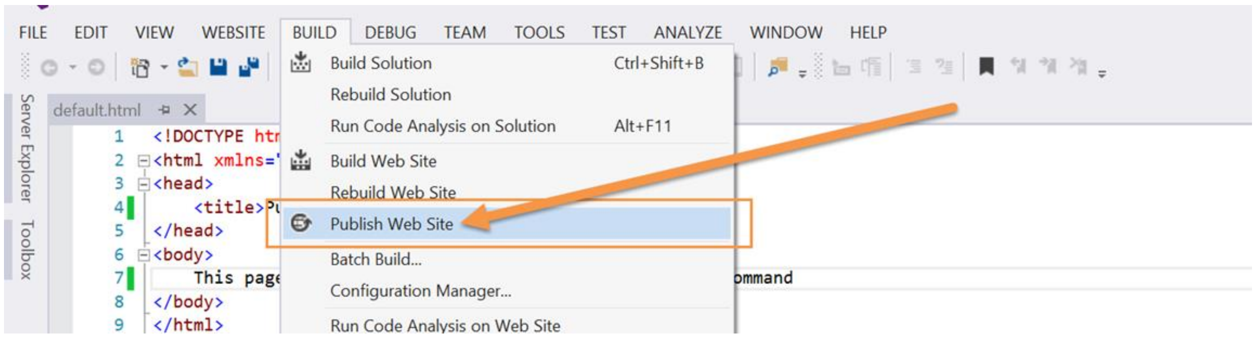
In the previous section, I discussed that a fairly significant drawback to the “Copy Web Site” deployment method in Visual Studio is that it is just a straight copy of website files and doesn’t take into consideration things such as databases and build transformations. The “Publish Web Site” command is a more powerful alternative to “Copy Web Site” that does handle these type of tasks.

The Publish Web Site command and the Copy Web Site command are both available from the main Visual Studio menu and as a context menu item in the Solution Explorer. In the main Visual Studio menu system, the command is found in the Build menu (as shown in Figure 26) and the context menu item (shown in Figure 27) is presented when right-clicking on the web project in the Solution Explorer:



```
<!DOCTYPE htr
<html xmlns='
<head>
    <title>Pu
</head>
<body>
    This page                                              ommand
</body>
</html>
```

Figure 26: Publish Web Site from the Build menu



```
<!DOCTYPE htr
<html xmlns='
<head>
    <title>Pu
</head>
<body>
    This page                                              ommand
</body>
</html>
```

Figure 27: Publish Web Site from the context menu

Regardless of the manner with which the menu was invoked, Visual Studio responds by displaying the “Publish Web” dialog. When the dialog is displayed, the first thing you have to do is provide information to Visual Studio about the intended destination of your website. In the case of many third-party hosting providers, this may include manually importing a PublishSettings file or manually entering the type of information found in a PublishSettings file. But, with the Azure integration built into the tooling, much of the manual effort is eliminated by selecting “Windows Azure Websites” in the “Select a publish target” section displayed in Figure 28. At that point, you will be asked to log into Azure using your Microsoft Account, and the Visual Studio will connect to the Azure Management Portal and download your PublishSettings file to automatically import your settings:

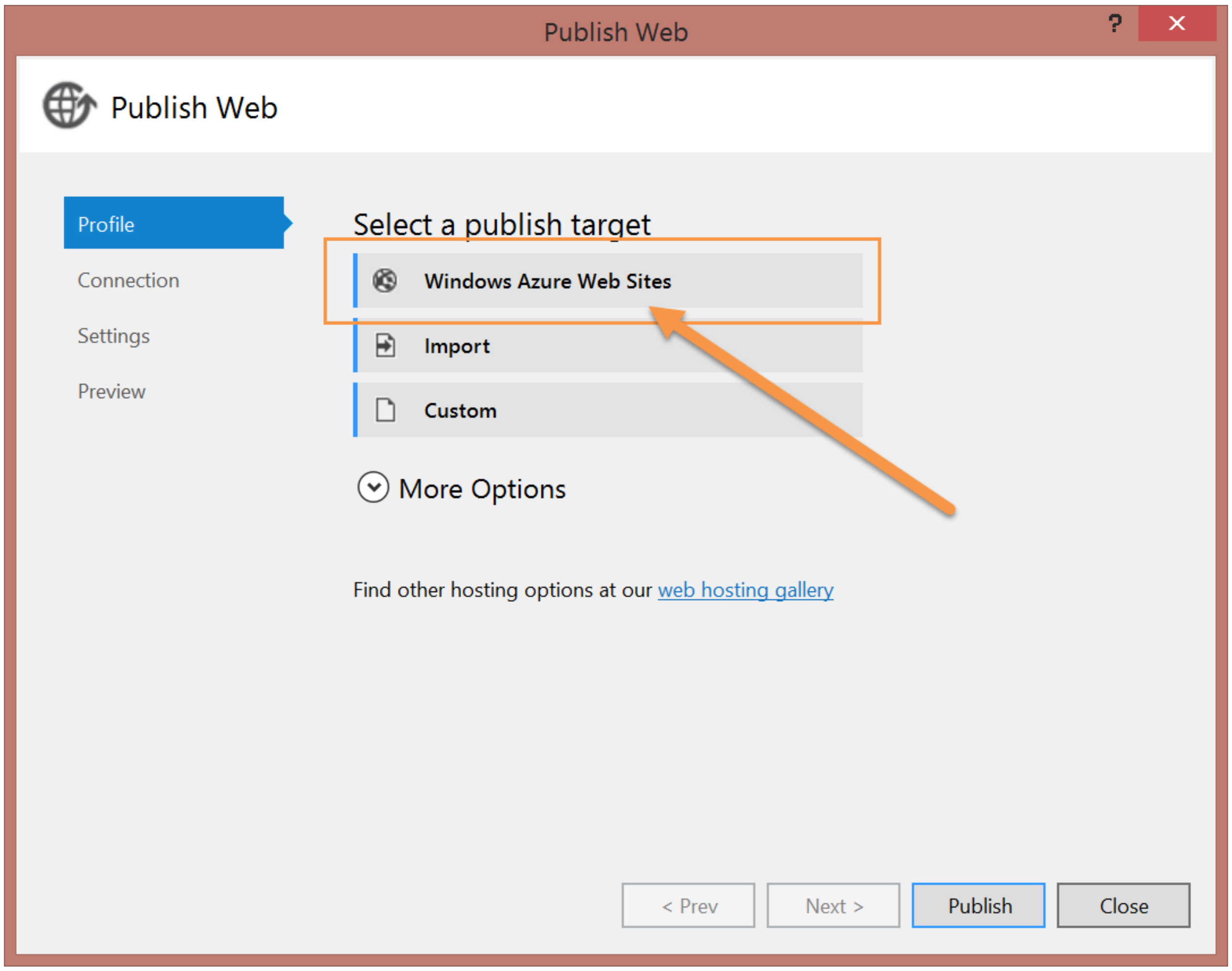


Figure 28: Publish Web Site dialog

Once your PublishSettings file is imported, Visual Studio will ask you to select which of your provisioned websites on Azure should be used as the target for the current project, as illustrated in Figure 29. Select the appropriate site and then accept your selection:

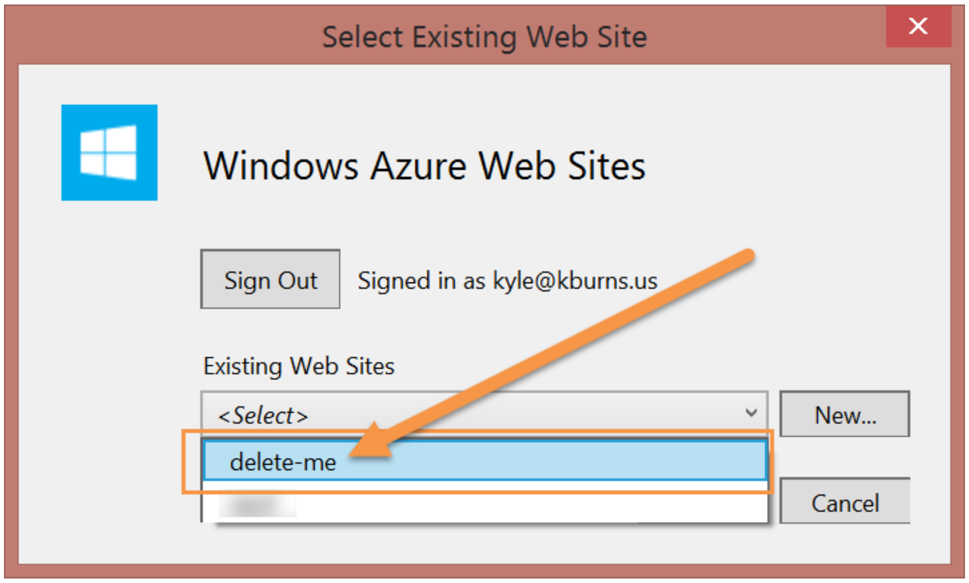
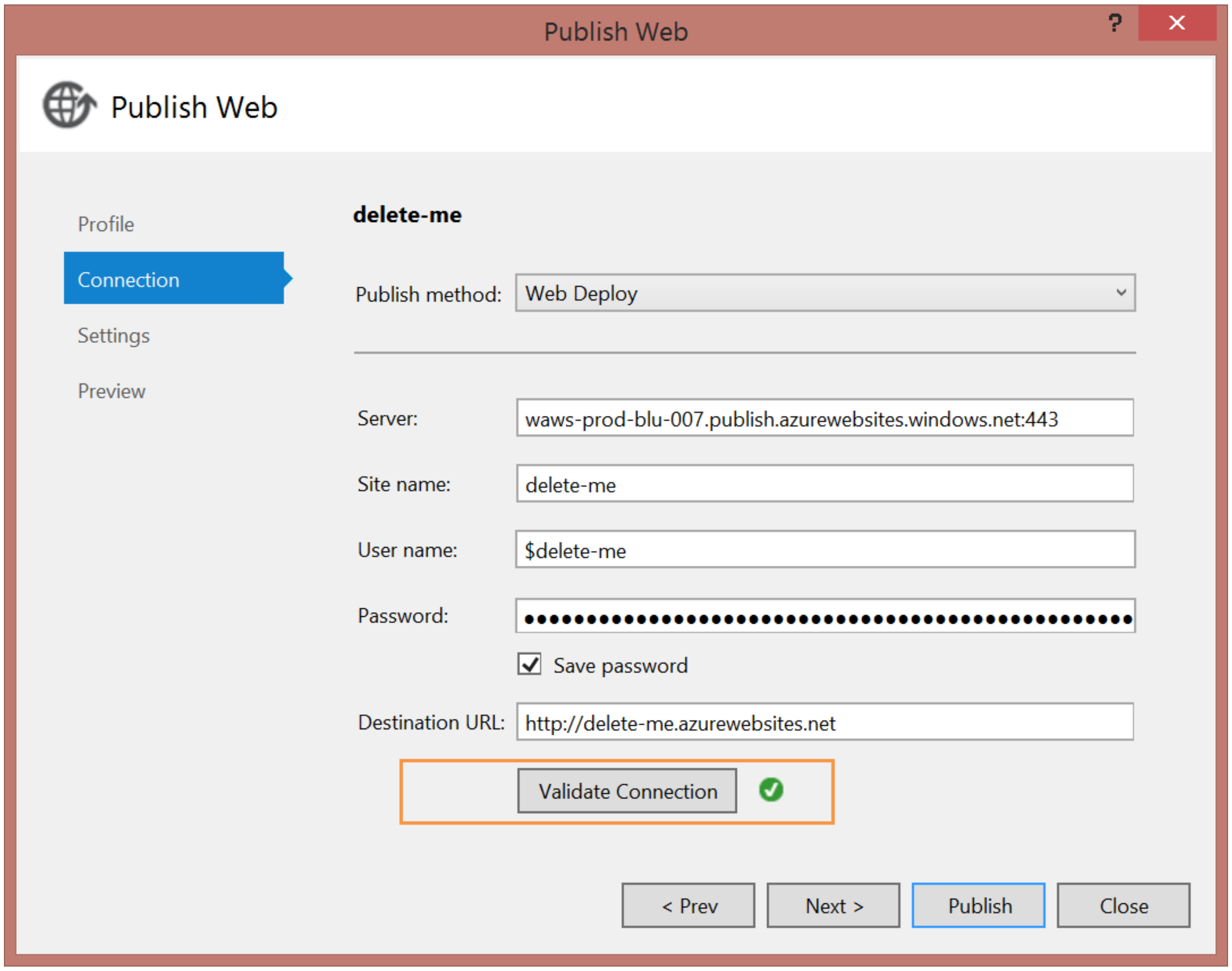


Figure 29: Publish Web Site select site

At this point, the Publish Web dialog is fully populated using the information from the website's PublishSettings and the deployment can already be initiated by selecting the Publish command. However, you may also need to customize settings on one of the tabs prior to beginning the process.

Connection Tab

The Connection tab, which is shown in Figure 30, is used to specify the publish method to be used for the deployment (in this example, we will stick with Web Deploy) as well as the connection and credential information applicable to the selected type. The Azure integration allowed this to be completely filled out without intervention, but you may want to (as an exploratory step) select the Validate Connection command to see the result, which can be seen in Figure 30:


Publish Web

Publish Web

Profile
Connection
Settings
Preview

delete-me

Publish method: Web Deploy

Server: waws-prod-blu-007.publish.azurewebsites.windows.net:443

Site name: delete-me

User name: $delete-me

Password: ●●

☑ Save password

Destination URL: http://delete-me.azurewebsites.net

Figure 30: Publish Web Site connection settings

Settings Tab

The Settings tab, shown in Figure 31, is where you will most commonly need to customize your deployment options prior to publishing your website to Azure. On this tab, you're given the option to select the configuration which will be used both for your build mode and the generation of appropriate configuration transform. Additionally, you can select from several options related to publishing files including:

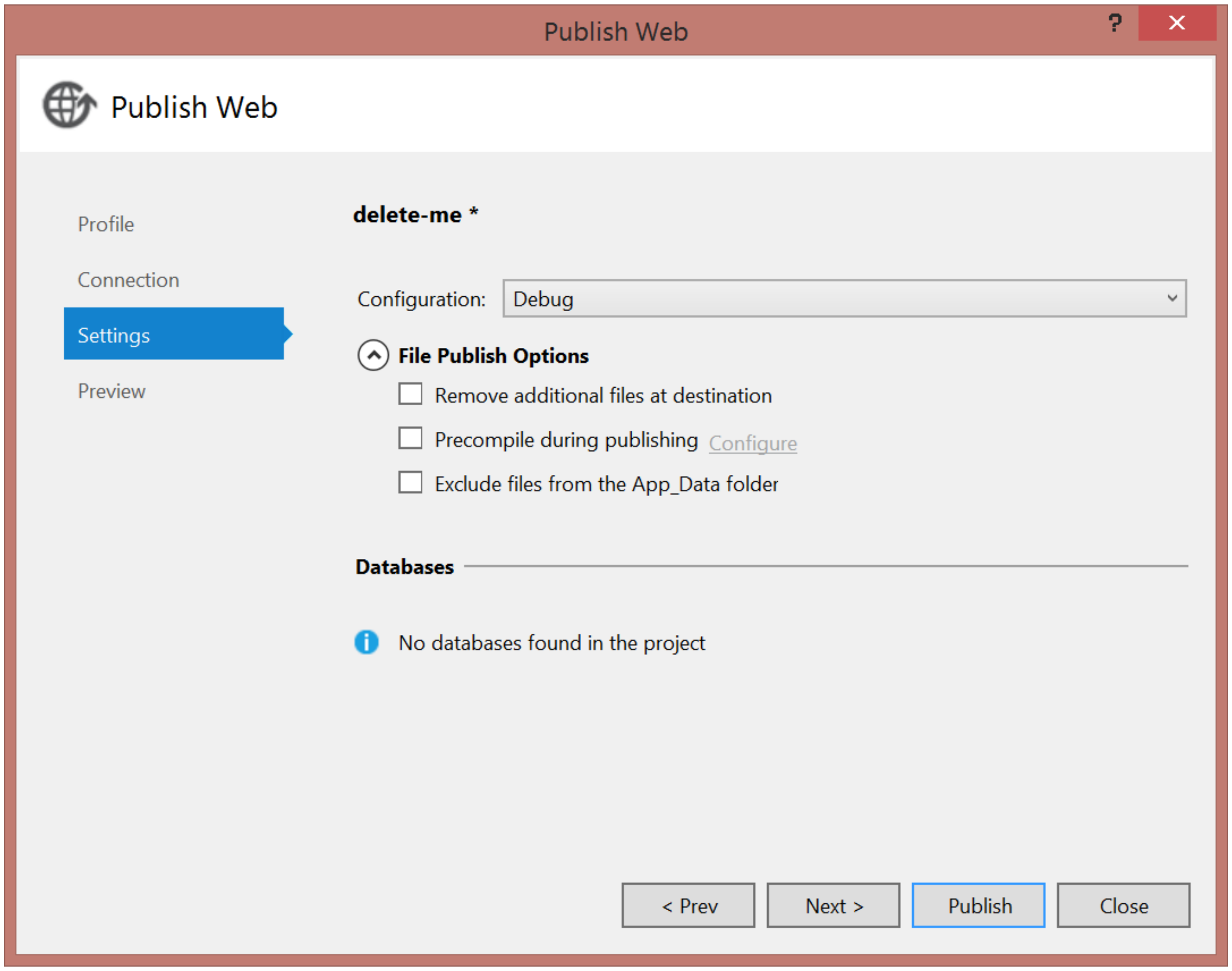
- Remove additional files at destination: This option is used to establish your Visual Studio project as the one true source of which files should exist, and it specifies that no files that are not represented in your project should remain deployed within the Azure Web Site directory.
- Precompile during publishing: ASP.NET, by default, performs on-demand compilation of web content based on markup being in one file and the code related to that webpage or control being in a separate file known as a "code-behind". This provides a great deal of flexibility in being able to modify code in place after it has been deployed. But, in certain

circumstances, you will prefer either the performance impact of having the code compiled before visitors attempt to access your website or the comfort of knowing that the raw source file is not published directly to the web host file system. In these cases, you can execute a build prior to deploying the website and have generated assemblies in the **bin** folder instead of code-behind files. This will still allow you to modify markup without a build and deploy cycle but any application code change will require a new deployment.

Note: When considering the precompile option, it's important to think about all of the application information surrounding what it does and why you are using it. If you're choosing the tool simply to protect the application code from disclosure to website visitors, you can instead be comfortable knowing that the code-behind files will not be directly served up by IIS. If you're attempting to protect the application code from somebody who has direct access to your site's deployment location (such as FTP access and deployment credentials), additional steps such as obfuscation will be required to keep the individual from being able to simply reverse engineer the compiled assemblies using highly available .NET decompilation tools.

- Exclude files from the **App_Data** folder: Once a website is live, it is likely that subsequent deployments that overwrite the application's data (which is, by default, stored in the **App_Data** folder) would have a negative and potentially destructive impact. You most certainly would not want to wipe out all of your website's user registrations or blog postings every time you update the look and feel of a page or add new features. This options allows you to specify that your publish action should consider the **App_Data** directory to be off-limits to avoid a potentially costly mistake.

In addition to the release configuration and file publishing options, this tab also gives you the ability to specify options related to your database deployment, including the connection string which is to be used for your database hosted in Azure and whether or not your application should execute database migration code against this database when the application starts using Entity Framework's Code-First Migrations tool.



delete-me *

Configuration: Debug

File Publish Options

- [ ] Remove additional files at destination
- [ ] Precompile during publishing Configure
- [ ] Exclude files from the App_Data folder

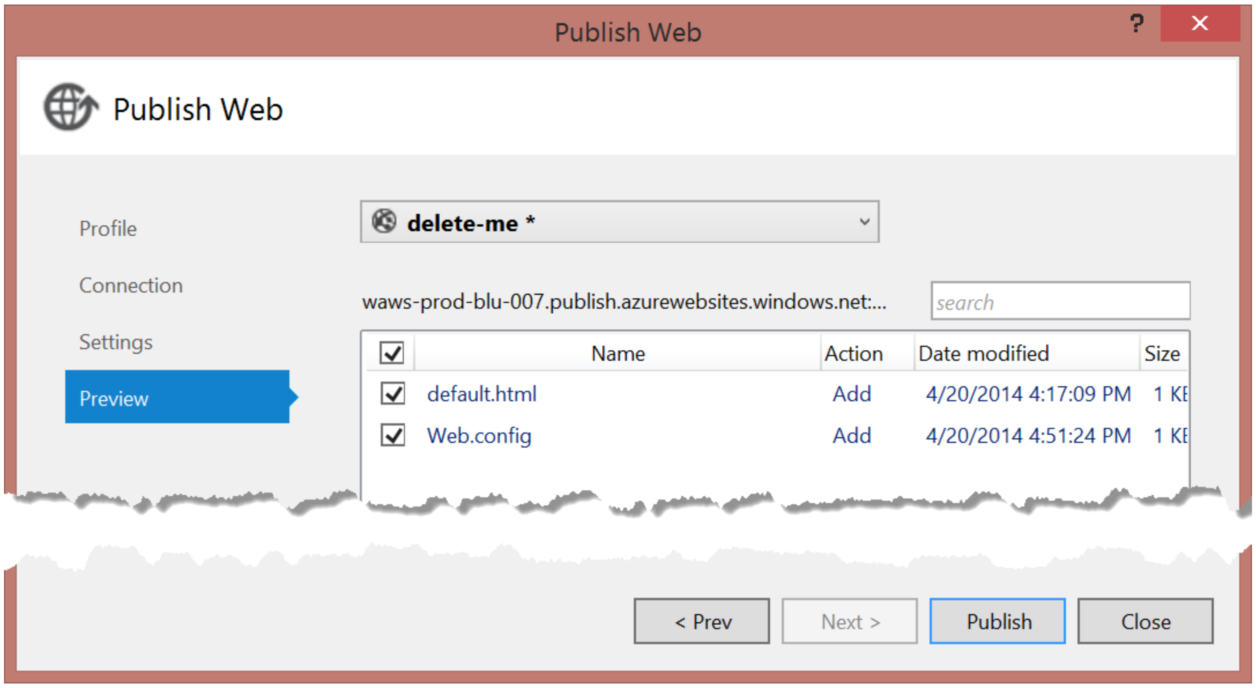
Databases

No databases found in the project

Figure 31: Publish Web Site settings

Preview Tab

The name Preview might suggest that you will have the opportunity to view what your deployed website will look like to the end user of the website prior to actually committing the changes, but the Preview tab actually gives you a preview of the actions that the deployment tool has determined are necessary to achieve a successful deployment. The output of generating preview actions is illustrated in Figure 32 and includes the list of assets upon which some action will be taken, the action to be taken (in our case, everything is to be added), the date that the asset was last modified, and the size of the asset. It might be valuable as you are growing used to the tool and learning what to expect from its behavior to use the preview output as a sanity check of sorts to make sure that you know what will happen when you initiate the deployment action by selecting the Publish command:



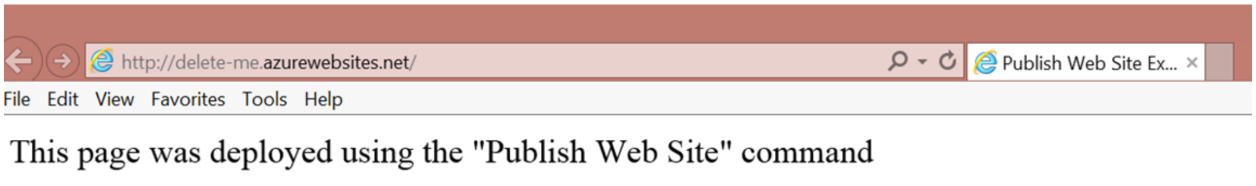
waws-prod-blu-007.publish.azurewebsites.windows.net:...

☑	Name	Action	Date modified	Size
☑	default.html	Add	4/20/2014 4:17:09 PM	1 KB
☑	Web.config	Add	4/20/2014 4:51:24 PM	1 KB

Figure 32: Publish Web Site preview

Publish Command

Once you've either jumped straight to the Publish command or first reviewed and configured the additional tabs, the deployment tool connects to Azure and executes all of the necessary changes. As a final step in the deployment, Visual Studio will launch your web browser to the site root as shown in Figure 33:



This page was deployed using the "Publish Web Site" command

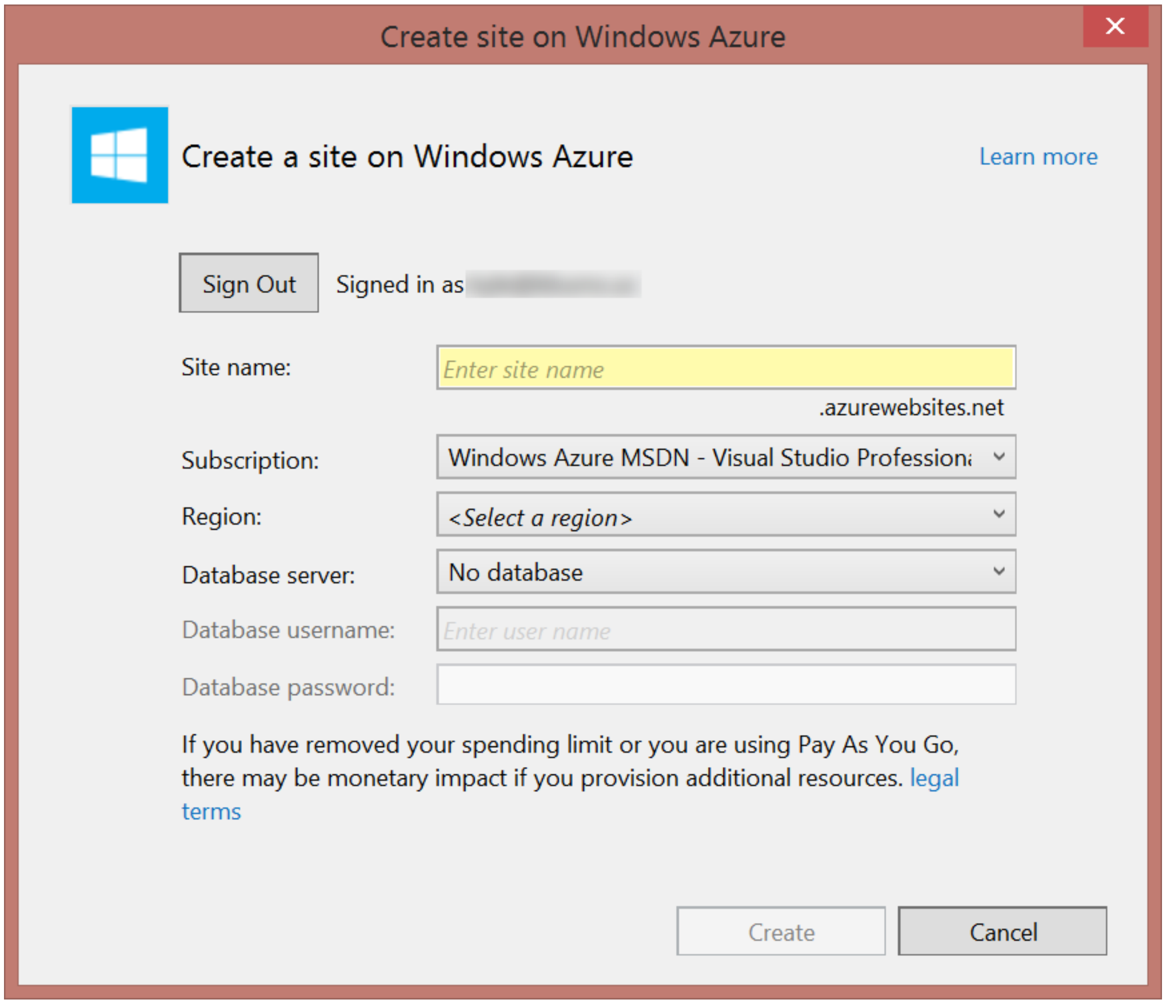
Figure 33: Publish Web Site completed

Provisioning New Azure Websites

In the previous sections on deployment from within Visual Studio, the focus and assumption has been that the developer using Visual Studio is working with a new or existing web project within Visual Studio and desires to deploy the contents of that project to an Azure Web Site that has previously been provisioned. Visual Studio also offers the ability to create a new Azure Web Site either when creating a new Visual Studio web project or when configuring the target Azure Web Site for deployment in the Publish Web Site tool.

Publish Web Site

In order to create a new Azure Web Site during execution of the “Publish Web Site” tool, the tool is invoked either through the Build menu or context menu from the Solution Explorer as described in the previous section and “Windows Azure Websites” is selected as the publishing target. After logging into Azure, instead of selecting an existing Azure Web Site as described earlier, the “New” command is invoked which will display the dialog shown in Figure 34:



Create site on Windows Azure

Create a site on Windows Azure

Learn more

Sign Out Signed in as [redacted]

Site name: Enter site name .azurewebsites.net

Subscription: Windows Azure MSDN - Visual Studio Professiona

Region: <Select a region>

Database server: No database

Database username: Enter user name

Database password:

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. legal terms

Create Cancel

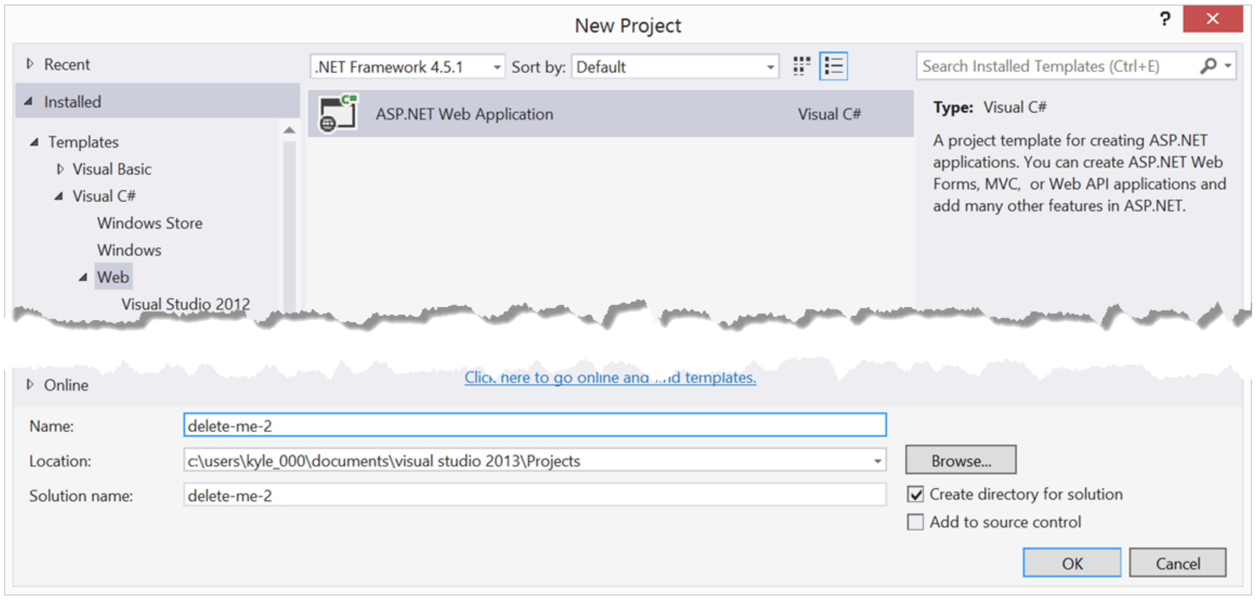
Figure 34: Create Site dialog

In the Create Site dialog, you will need to specify a name for the new site, select the subscription to which the site should be tied for billing purposes, and select the region in which the site should be hosted. Additionally, if you choose to do so, you can select a database which the site should use to store data and create a new database, if necessary. After selecting the desired options, select the Create command and continue using the Publish Web Site tool as described earlier to deploy content to the newly created Azure Web Site.

New Web Project

In addition to creating a new Azure Web Site during the publishing process of a previously created web project, you can also use Visual Studio tooling to create a new Azure Web Site when you first create the web project. One advantage that this brings is it provides the opportunity to lock in the website's base URL prior to completing the coding of the website. This ensures that you don't come up with the perfect URL only to find that somebody else scooped it up while you were busy trying to develop a website as good as the URL.

To create a new Azure Web Site as part of the creation of a new web project, start by invoking the "New Project" dialog either by selecting "New Project" on the Visual Studio start page or by selecting "File…New…Project" from Visual Studio's main menu. Visual Studio will display the New Project dialog which is shown in Figure 35:



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.

Name: delete-me-2

Location: c:\users\kyle_000\documents\visual studio 2013\Projects

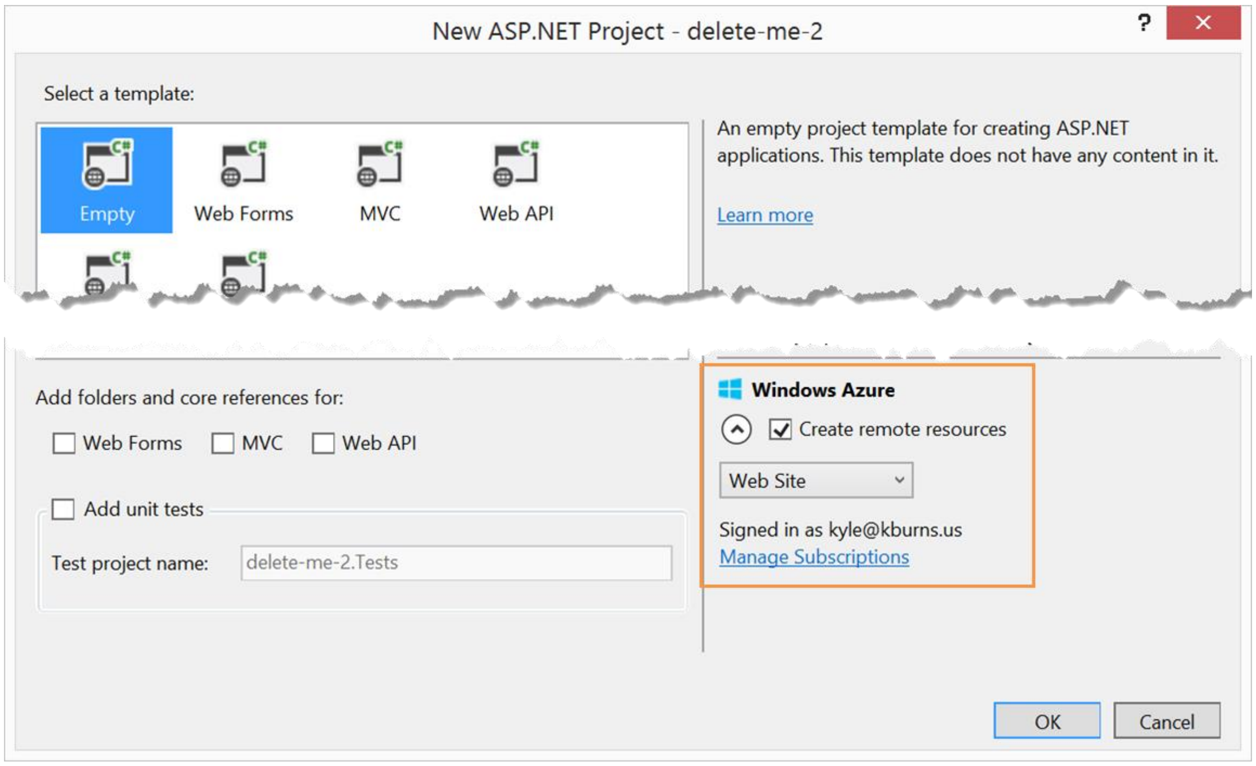
Solution name: delete-me-2

[✓] Create directory for solution

[ ] Add to source control

Figure 35: New Project dialog

After selecting the ASP.NET Web Application project type, Visual Studio displays the "New ASP.NET Project" dialog (shown in Figure 36) to collect details such as the specific type of web application to create, selection from among commonly selected references for convenience, and (most important to us now) the option to specify that new resources in Azure should be created for your project. The dialog gives you the option to select between creating the Azure resources using the Azure Web Site service or in a VM. Since our focus is on Azure Websites, select "Web Site":



New ASP.NET Project - delete-me-2

Select a template:

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Add folders and core references for:

- [ ] Web Forms
- [ ] MVC
- [ ] Web API

- [ ] Add unit tests

Test project name: delete-me-2.Tests

Windows Azure

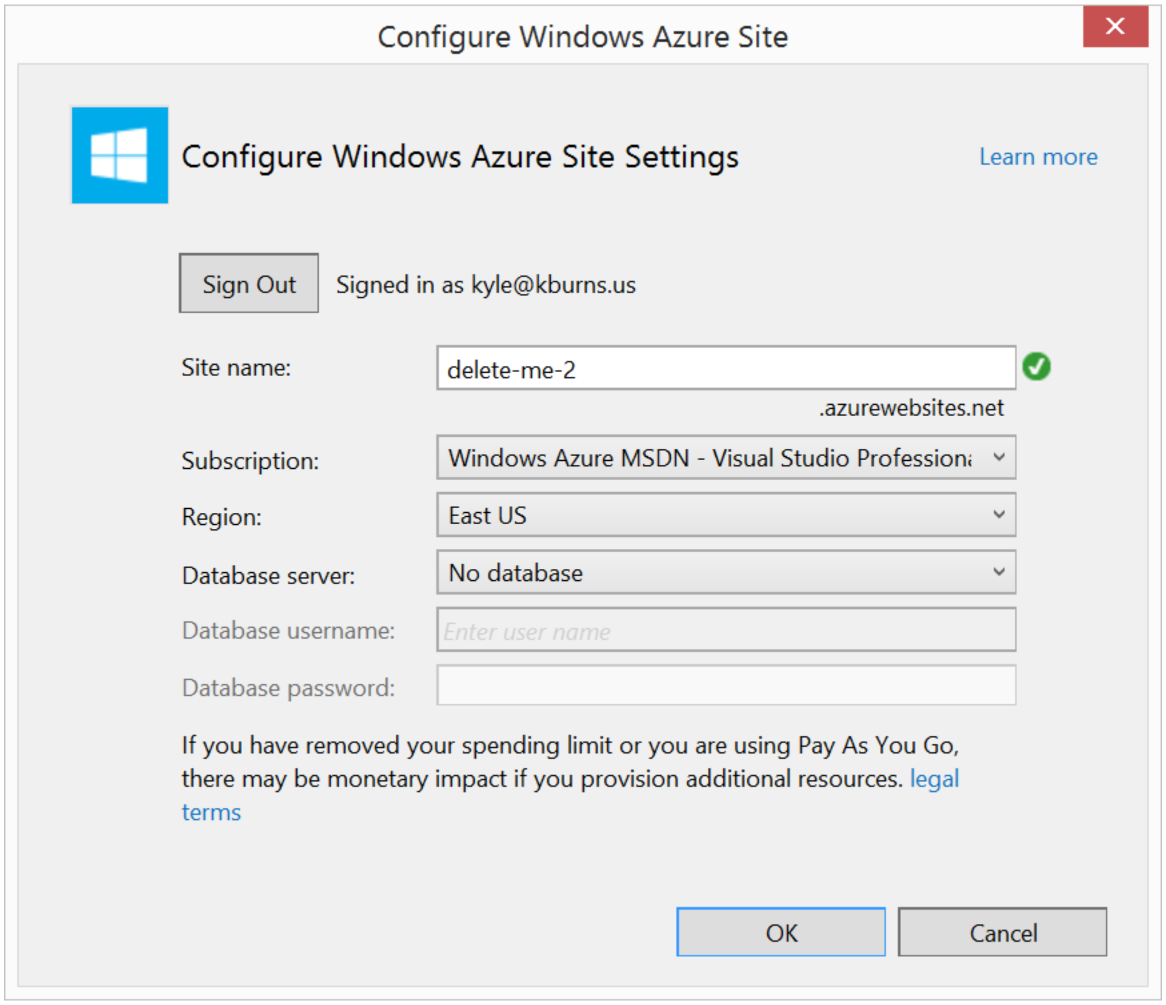
- [✓] Create remote resources

Web Site

Signed in as kyle@kburns.us

Figure 36: New ASP.NET Project dialog

After specifying the website type and designating that the Windows Azure remote resource should be created, select OK. Visual Studio will then display the “Configure Windows Azure” dialog as shown in Figure 37. At a minimum, in this dialog you will specify a website name (a default will be specified based on the project name) and ensure that Subscription and Region are set appropriately. If your application will also require the creation of a database, a database server, username, and password will be specified in this dialog as well. After choosing the desired options, select OK and Visual Studio will finish creating the project and will interact with the Azure Management API to provision your Azure Web Site and, if applicable, database:


Configure Windows Azure Site
Configure Windows Azure Site Settings
Learn more
Sign Out
OK
Cancel

Signed in as kyle@kburns.us

Site name: delete-me-2 .azurewebsites.net

Subscription: Windows Azure MSDN - Visual Studio Professiona

Region: East US

Database server: No database

Database username: Enter user name

Database password:

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. legal terms

Figure 37: Configure Windows Azure dialog

Once Visual Studio finishes setting up your project and Azure Web Site, work with the project just as you would any other web project. When you have a product that is suitable to share with the world, use the Publish Web Site command to deploy your assets.

Deploy from WebMatrix

In addition to Visual Studio, which is geared more to professional developers (even though the Express versions are marketed to the student and hobbyist), Microsoft has made WebMatrix available as a tool to allow for easy creation, editing, and deployment of websites.

The easiest way to work with your Azure Web Site that has previously been provisioned using the Azure Management Portal or some other tool is to sign into WebMatrix using the same Microsoft Account attached to your Azure account and from the startup page select “Open…Windows Azure”. This action displays a window showing the Azure Websites that have been configured in your Azure portal as shown in Figure 38. The dialog provides several categories to select from in the left-hand navigation. Selecting Windows Azure will display a list of your provisioned Azure Websites and allow you to select the website with which you want to work:

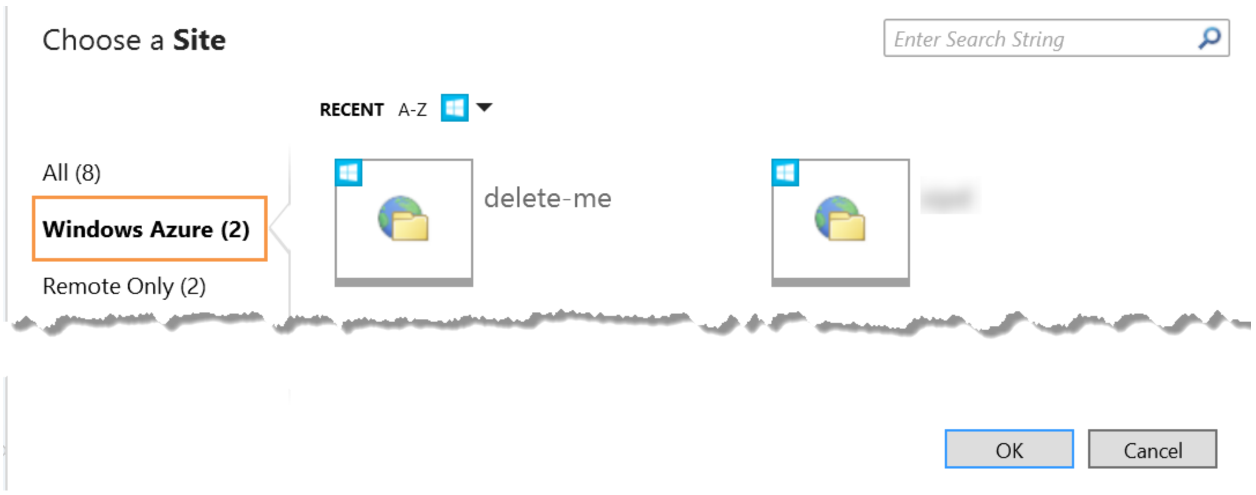


Figure 38: WebMatrix Open Site dialog

After selecting OK from the Open dialog, the WebMatrix will display the website contents within the IDE and allow you to edit the files, create new files, and delete files. The connection to the website is a live remote connection so saving files in WebMatrix will automatically deploy the updates to your Azure Web Site.

Deploy from Source Control

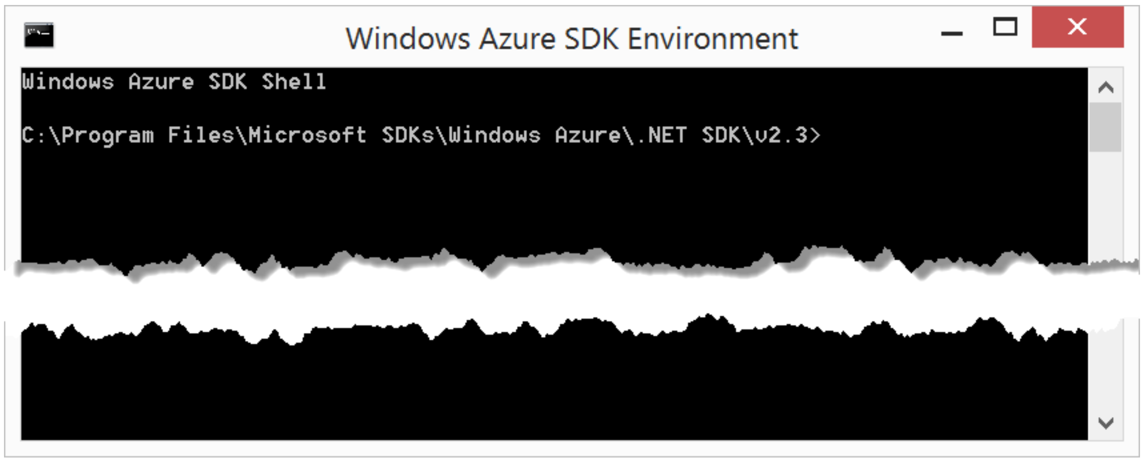
Azure Websites has integration options to connect many popular source control repositories to your deployment location. Source control integration is included in the list for the sake of completeness but these options will be discussed in detail in the next chapter and will not be expanded upon here.

Automate Deployment

The deployment methods that have been discussed until this point have all revolved around clicking commands and selecting options in a graphical interface. But many administrators and developers find that management tasks such as provisioning resources and deploying code are completed more efficiently through a command line and, especially if the command line task can be scripted, are also less prone to error that way. In addition to a management REST API which could be used to produce your own command-line tools, Microsoft makes available a cross-platform command-line interface which has an installation package for Windows, Mac, and Linux as well as a set of scripting tools for use within Windows PowerShell. Any of these tools can be used to easily automate management tasks.

Azure Command-line Interface

The Azure command-line interface provides a common command syntax for automating Azure management tasks from Windows, Mac, or Linux clients. In Windows, the tool is accessed by opening "Windows Azure Command Prompt" which will open a DOS-style command window similar to the one pictured in Figure 39:



```
Windows Azure SDK Shell

C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\v2.3>
```

Figure 39: Windows Azure command prompt

This window is a standard Windows command window, but the path and other environment variables are set to ensure that the Azure command-line tools can be invoked. To access Azure commands, the **`azure`** command is typed followed by applicable options. If you're at all like me, the first thing typed after opening the command window will be:

```
>azure help
```

This displays documentation on the basic commands and services that can be controlled with the command-line tool—each with the ability to separately invoke help documentation for more detail. Since this book is focused primarily on Azure Websites, view the commands specific to this service by typing:

```
>azure help site
```

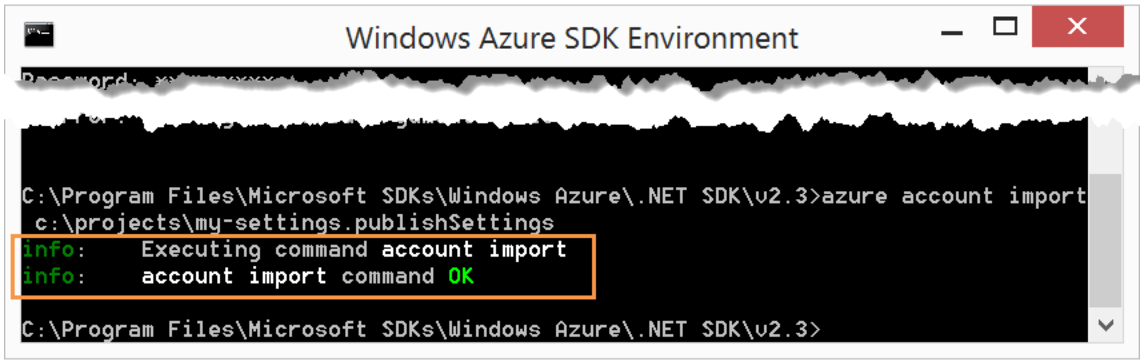
Now that you have viewed the list of available commands, you may already be forming scripts in your mind. In order to get started using the tool, you need to get the environment ready to work with your Azure subscription. Intuitively, that would mean typing **`azure login`** at the command prompt and providing credentials but that would only work if you're using an organizational account and a standard Microsoft account will not successfully log in. Instead, you will need to obtain your **`.publishSettings`** file and import it into the tool. This is initiated with the command:

```
>azure account download
```

After typing the command, a web browser window will open to the Azure Management Portal and you will be prompted to download your **`.publishSettings`** file. Because you will use the path to the **`.publishSettings`** file in the next step from the command line, I highly recommend saving the file to a location that will be easy to remember and type. Assuming that the path **`C:\projects\my-settings.publishSettings`** was used during the download, next import the file into your command-line interface by using the following command:

```
>azure account import c:\projects\my-settings.publishSettings
```

The command-line tool should import the file at this point and successful completion will display output similar to what is seen in Figure 40:

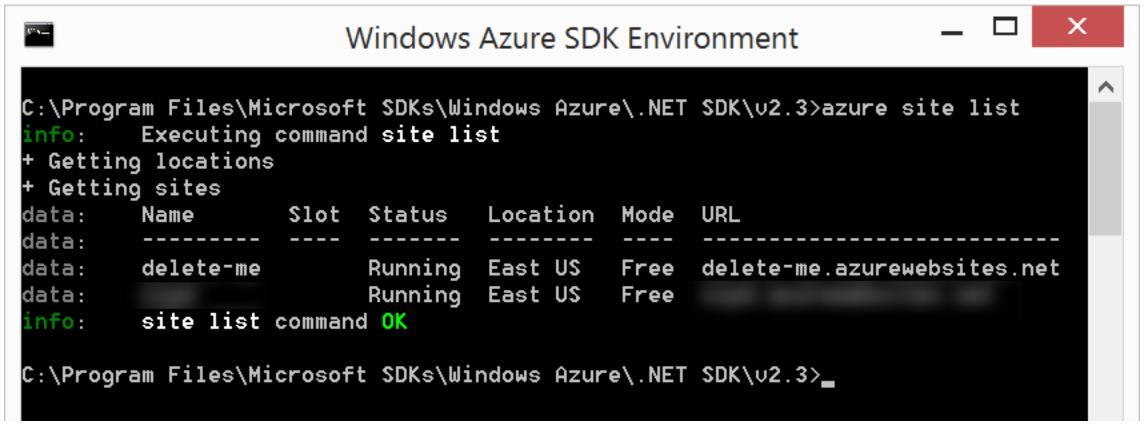


```
C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\v2.3>azure account import
 c:\projects\my-settings.publishSettings
info:    Executing command account import
info:    account import command OK

C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\v2.3>
```

Figure 40: Azure command-line interface import account

To see that the command-line tool is appropriately interacting with your Azure account at this point, obtain a list of Azure Websites that you have created by typing the following command and observing output similar to what is seen in Figure 41:

```
>azure site list
```



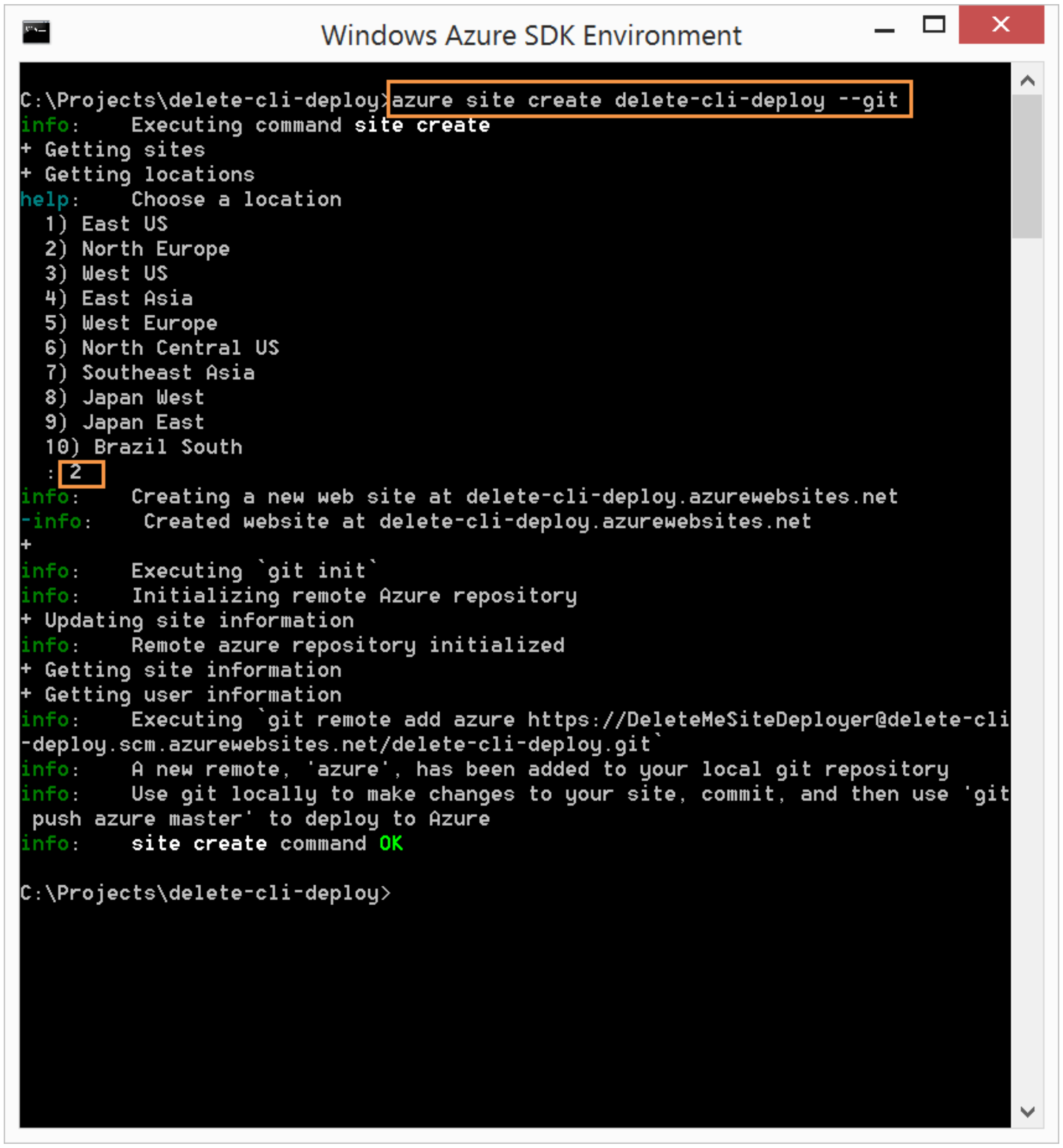
```
C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\v2.3>azure site list
info:    Executing command site list
+ Getting locations
+ Getting sites
data:    Name       Slot  Status   Location  Mode  URL
data:    ---------  ----  -------  --------  ----  --------------------------
data:    delete-me        Running  East US   Free  delete-me.azurewebsites.net
data:    [redacted]       Running  East US   Free  [redacted]
info:    site list command OK

C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\v2.3>_
```

Figure 41: Azure CLI list sites

Now that you are interacting with Azure from the command, it's time to provision a new site and deploy. We'll walk through the process here but it's important to point out that the example will make use of a Git repository, which will be explained in further detail in the next chapter.

Note: The following example references a site called delete-cli-deploy. If you're following along on your own computer, you will encounter errors if you use the same site name used by the example because the site name must be unique within Azure. Be sure to pick your own unique site name when trying the example.

In the command-line interface, change your working directory to the directory containing the website files. The example website is stored on a directory on my computer with the path **`C:\projects\delete-cli-deploy`** to match what I will use as the Web Site name in Azure. Now issue the command **`azure site create delete-cli-deploy --git`** to initiate a new website creation. The tool will prompt for an appropriate region, which you should supply (I chose North Europe in the example). After providing several status updates, the tool will notify you that the website creation is complete. The command and output should be similar to what is shown in Figure 42:



```
C:\Projects\delete-cli-deploy>azure site create delete-cli-deploy --git
info:    Executing command site create
+ Getting sites
+ Getting locations
help:    Choose a location
  1) East US
  2) North Europe
  3) West US
  4) East Asia
  5) West Europe
  6) North Central US
  7) Southeast Asia
  8) Japan West
  9) Japan East
  10) Brazil South
  : 2
info:    Creating a new web site at delete-cli-deploy.azurewebsites.net
-info:    Created website at delete-cli-deploy.azurewebsites.net
+
info:    Executing `git init`
info:    Initializing remote Azure repository
+ Updating site information
info:    Remote azure repository initialized
+ Getting site information
+ Getting user information
info:    Executing `git remote add azure https://DeleteMeSiteDeployer@delete-cli
-deploy.scm.azurewebsites.net/delete-cli-deploy.git`
info:    A new remote, 'azure', has been added to your local git repository
info:    Use git locally to make changes to your site, commit, and then use 'git
 push azure master' to deploy to Azure
info:    site create command OK

C:\Projects\delete-cli-deploy>
```

Figure 42: Azure CLI create site

Once you have created the website, you need to add the contents of your local directory to the Git repository. This can be accomplished with the following commands:

```
git add .

git commit -m "this is a checkin comment"
```

Now that you have committed the changes to your Git repository, it's time to deploy them to Azure. Use the following command to deploy the content:

```
git push azure master
```

After prompting for deployment credentials, Git will deploy the content to your Azure Web Site. The result of having deployed can be seen by executing the final Azure command-line interface command that we will cover in this section:

```
azure site browse delete-cli-deploy
```

The Azure command-line interface has a wealth of commands to explore and learn to leverage; we have only just scratched the surface in this section. I encourage anyone who may find themselves in a DevOps role or is just interested in automation to explore this tool either intuitively through the integrated help system or with the help of Microsoft's online documentation.

Windows PowerShell

As an alternative to the Azure command-line interface, Windows PowerShell provides an environment tailored specifically to scripting management tasks, and integrates with the Microsoft .NET Framework to provide the ability to accomplish anything in your scripts that you are capable of writing .NET code to do. It also integrates with the same MS Deploy engine that is used within the tools used to deploy via the Visual Studio environment, so things such as deploying directly from Visual Studio without involving an intermediary such as Git are much more easily accomplished. PowerShell itself is a very powerful tool with the complexity that you would expect along with the power. "PowerShell Succinctly" could very easily get its own title in this e-book series so I will not make an attempt to provide instruction on PowerShell here. Instead, I will narrowly and clearly discuss the most basic commands necessary to connect to Azure through PowerShell and deploy a website created in Visual Studio.

The Azure PowerShell commands are most easily made available by launching the "Windows Azure PowerShell" item that was added to the Windows Start Screen when the PowerShell tool was installed. This will import the appropriate Azure `cmdlets` into the PowerShell environment and, on first load, you will be prompted to allow import of the modules. As with the command-line tools, the inquisitive developer (and I hope that includes everyone reading this book) may want to have their first act after launching the tool to explore the commands offered by typing **`help azure`** and exploring a bit.

Once you've completed any initial exploring from the help screens, the first thing you need to do in order to get productive is import your **`.publishSettings`** file from the Azure Management Portal. This is initiated with the command:

```
>Get-AzurePublishSettingsFile
```

After typing the command, a web browser window will open to the Azure Management Portal and you will be prompted to download your **`.publishSettings`** file. Because you will use the path to the **`.publishSettings`** file in the next step from the command line, I highly recommend saving the file to a location that will be easy to remember and type. Assuming that the path **`C:\projects\delete-me\publish.publishSettings`** was used during the download, next import the file into your command-line interface by using the following command (wrapped for readability but enter on a single line):

```
>Import-AzurePublishSettingsFile -PublishSettingsFile
“c:\projects\delete-me\publish.publishSettings”
```

The command-line tool will import the file at this point and, upon successful completion, prompt for the next command. If you have multiple subscriptions, you will be shown a message detailing the subscription that will be set as default and current, along with instructions for viewing additional subscriptions.

As with the command-line tool, a quick way to verify that you can successfully interact with your Azure account is to issue a command to retrieve a listing of Azure Web Site instances. At the PowerShell command prompt, type the following command:

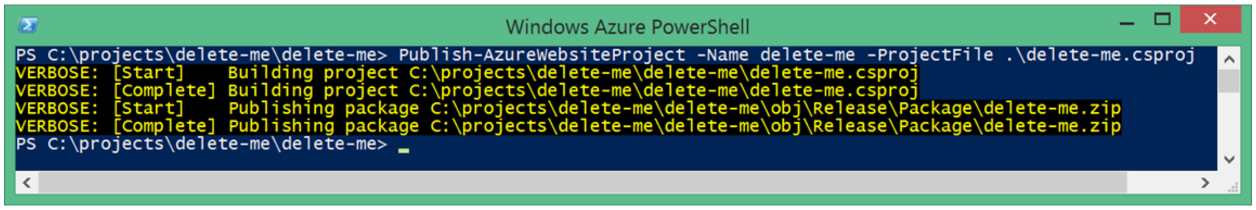
```
Get-AzureWebsite
```

After a few seconds, the system will respond by listing each website that you have added in Azure including the website name, running state, and the DNS address for the website. If this output is shown, you are set to use the **`.publishSettings`** file you imported to interact with your Azure account.

Once you have set up and verified your connection to Azure, deploying your project is a simple task accomplished with a single command. In the Azure PowerShell console, change the working directory to the directory containing the **`.csproj`** file for your web project and enter the following command:

```
Publish-AzureWebsiteProject -Name delete-me -ProjectFile .\delete-me.csproj
```

The PowerShell `cmdlet` will invoke tools to build the project, create a package, and deploy the package to your Azure Web Site. Successful execution should produce results similar to what is seen in Figure 43:



```
PS C:\projects\delete-me\delete-me> Publish-AzureWebsiteProject -Name delete-me -ProjectFile .\delete-me.csproj
VERBOSE: [Start]    Building project C:\projects\delete-me\delete-me\delete-me.csproj
VERBOSE: [Complete] Building project C:\projects\delete-me\delete-me\delete-me.csproj
VERBOSE: [Start]    Publishing package C:\projects\delete-me\delete-me\obj\Release\Package\delete-me.zip
VERBOSE: [Complete] Publishing package C:\projects\delete-me\delete-me\obj\Release\Package\delete-me.zip
PS C:\projects\delete-me\delete-me> _
```

Figure 43: Azure PowerShell Publish project

Once the PowerShell command has completed execution, you can navigate a web browser to your website and view the updated contents.

As mentioned earlier, the functionality made available in the Azure PowerShell `cmdlets` is extensive and lets you wield a lot of control in your scripts. We've just barely scratched the surface here. I would encourage readers to visit Microsoft's Azure Script Center at http://azure.microsoft.com/en-us/documentation/scripts/ where you will find a wealth of scripts either to use as is, or to study and tweak to solve exactly the problems that you need solved.

Summary

In this chapter, you have learned about several deployment options available to make your assets available to visitors of your Azure Web Site. For individual developers, working on the Microsoft platform deployment from Visual Studio will most likely be the first and most commonly used deployment method. But, as the application becomes more complex or additional developers are working in a team, automated deployment options will often enter into consideration. In the next chapter, you will focus a little more specifically on source control and on how two major source control systems integrate with Windows Azure.

Chapter 5 Source Control Integration

Source control is an extremely important part of your development environment, whether you are working as part of a team of 300 developers on an enterprise-class product or as a single developer building a website for a friend. Source control (or revision control as it is also referred to as) provides a safety net to your development that provides a central point of backup as well as a way to easily revert back to previous versions of your code after a change introduces unanticipated side effects that keep your application from working correctly. In this chapter, we will look at two popular source-control providers: Team Foundation Server and Git. Each represent two different approaches to version control, and you will see how they can be easily integrated into your Azure Web Site development.

Team Foundation Server

Microsoft's Team Foundation Server product comes in two flavors: the Enterprise tool that is typically installed on an organization's on-premises server and the Azure offering that is now known as Visual Studio Online. While version control is one aspect of the tool, it is intended to be a complete application lifecycle management (ALM) solution with features to manage not only the source code but requirements, testing, and schedule as well. In this section, we'll be looking at Visual Studio Online tool's integration with Azure Websites.

Because Visual Studio Online is a complete ALM solution and not just a version-control system, Microsoft has actually tooled it to allow for selection between a centralized approach to version control (which is provided in the Team Foundation Version Control product) and a more distributed approach (using Git). Git will get its own coverage later in this chapter so, in this section, we will focus solely on Team Foundation Version Control. I will, however, point out decision points where your choice of version control strategy might cause you to take different actions in Visual Studio Online if you are using version control other than Team Foundation Version Control.

As of the time of this writing, Microsoft is making it easy for small teams to get started using Visual Studio Online by providing free accounts to service teams of up to five members in a service offering called Visual Studio Online Basic. This free offering allows adding team members for an additional monthly cost, and includes the following features:

- Unlimited number of projects
- Hosted code repositories using Team Foundation Version Control or Git. These repositories are private and available only to members of your team
- Project planning tools such as Sprint and Kanban boards and other tools commonly used both in agile and waterfall development
- Integration with Visual Studio, Xcode, and Eclipse
- Continuous integration using cloud build services

In addition to Visual Studio Online Basic, Microsoft also has the paid Visual Studio Online Professional product that adds the Visual Studio Professional IDE and Visual Studio Online Advanced product, which still uses Visual Studio Express but adds more advanced collaboration features, project management tools, and opportunities for stakeholders to make their voices heard. As a final option, Visual Studio Premium with MSDN also includes a Visual Studio Online subscription that adds test case management and the option to host team projects on-premises in addition to the cloud.

Assuming that you are going to start out exploring the Visual Studio Online Basic offering, you can start getting set up by navigating a web browser to http://www.visualstudio.com. On this page, you should find a promotional spot advertising Visual Studio Online with a call to action labeled “Get started for free” (if it has changed since the time of writing, you may have to search for something similar on the page). Select the call to action and you should be taken (after possibly stopping to log into your Microsoft Account) to a page where you are asked for additional information such as your name and a name to assign to your Visual Studio Online account. Complete and submit the form.

Once you’ve submitted your registration form, Visual Studio Online will create your account and transfer you to your account dashboard, which will include a form prompting you to create your first project. The form expects the following information:

- Project Name: This is the name that you assign to the project and it will show up on project links and reporting. This field is required.
- Description: This is an optional field but, as a good practice and in preparation for when you are managing several projects out of this portal, you should include a reasonable description of the purpose of the project.
- Version Control: This allows you to choose whether Team Foundation Version Control or Git will be used as version control for the project. Depending on your selection, an appropriate repository will be created to service your project.
- Process Template: A process template is used to define the tools and processes that will be used to manage the work of building the software product that is the goal of your project. Depending on the way the team will work, the tools and how they are leveraged can vary greatly. Because of this, Microsoft has created three templates to capture the way that a large number of modern development teams work today:
    - Microsoft Visual Studio Scrum
    - Microsoft Solutions Framework (MSF) for Agile Software Development
    - Microsoft Solutions Framework (MSF) for Capability Maturity Model Integration (CMMI) Process Improvement

For the purpose of this walk-through, select Team Foundation Version Control for your version control and Microsoft Visual Studio Scrum for your process template and select “Create Project”. Visual Studio Online should respond by initiating creation of your Team Project and, after a few moments, navigates to your project portal which will look similar to Figure 44.

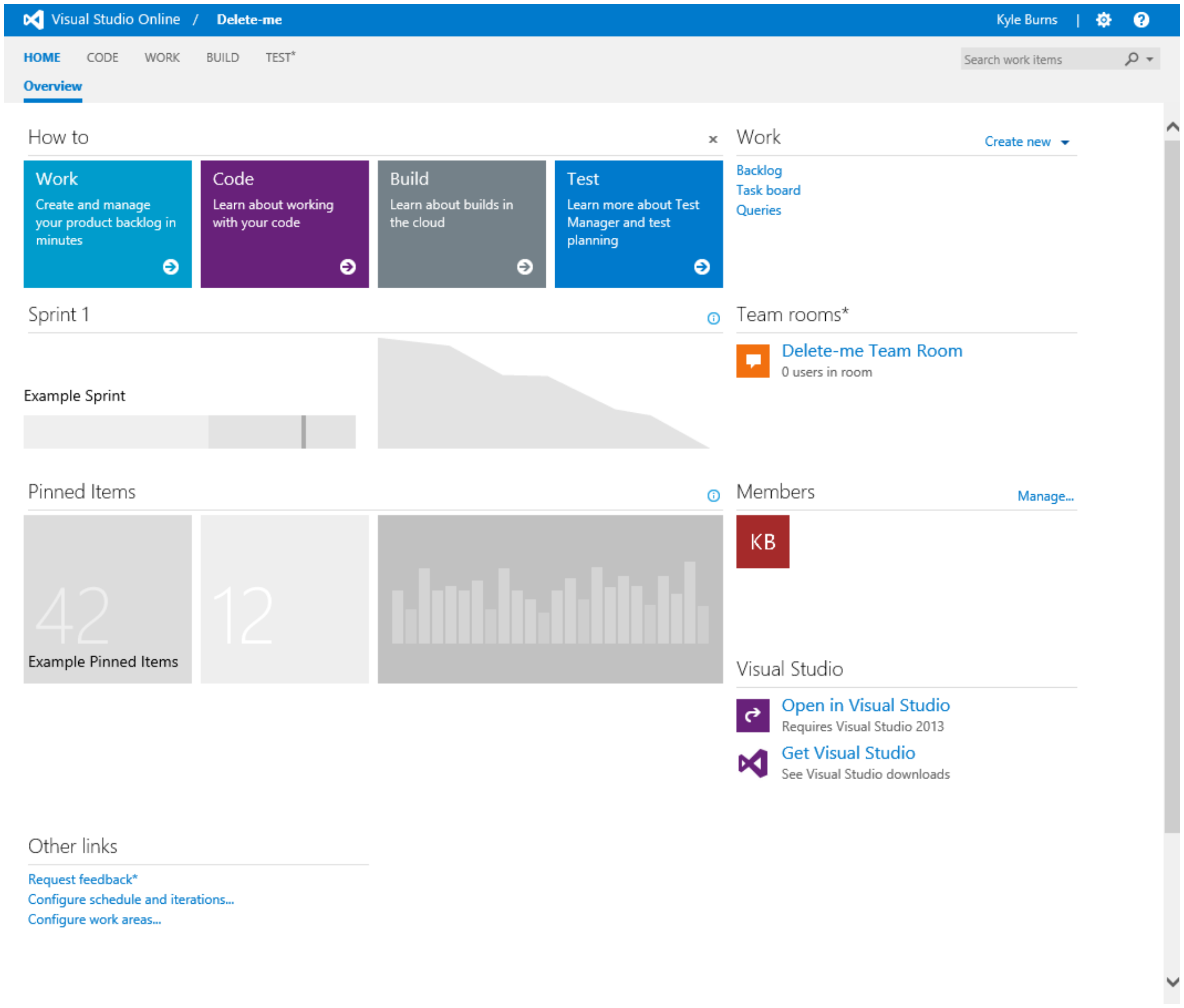


Figure 44: Visual Studio Online Project Portal

There are many options from which to choose in the Project Portal such as inviting additional users to the team and starting to add work items to your backlog. But, for now, just select "Open in Visual Studio" which will cause Visual Studio to open and attempt to connect to your team project. You'll notice that you cannot begin working immediately because you have not yet configured a local workspace. Visual Studio's "Team Explorer" should appear as pictured in Figure 45:

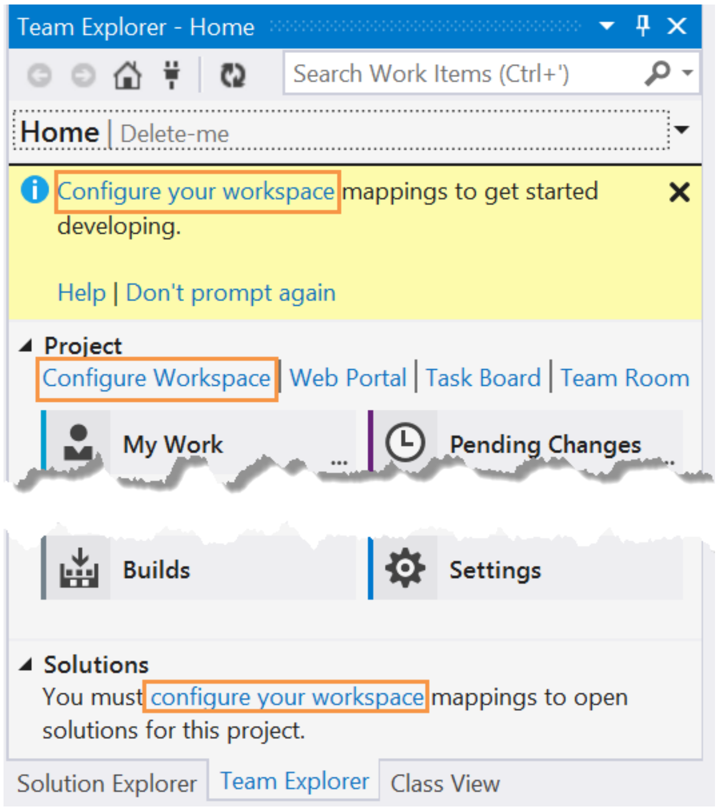

Team Explorer - Home
Search Work Items (Ctrl+')
Home | Delete-me
Configure your workspace mappings to get started developing.
Help | Don't prompt again
Project
Configure Workspace | Web Portal | Task Board | Team Room
My Work
Pending Changes
Builds
Settings
Solutions
You must configure your workspace mappings to open solutions for this project.
Solution Explorer
Team Explorer
Class View

Figure 45: Team Explorer Configure Workspace prompt

The Team Explorer window provides several opportunities to select “Configure Workspace” or “configure your workspace”. Select one of these commands to continue and the Team Explorer window will provide the opportunity to map the project root to local folders as shown in Figure 46. Accept the defaults and select the “Map & Get” command:

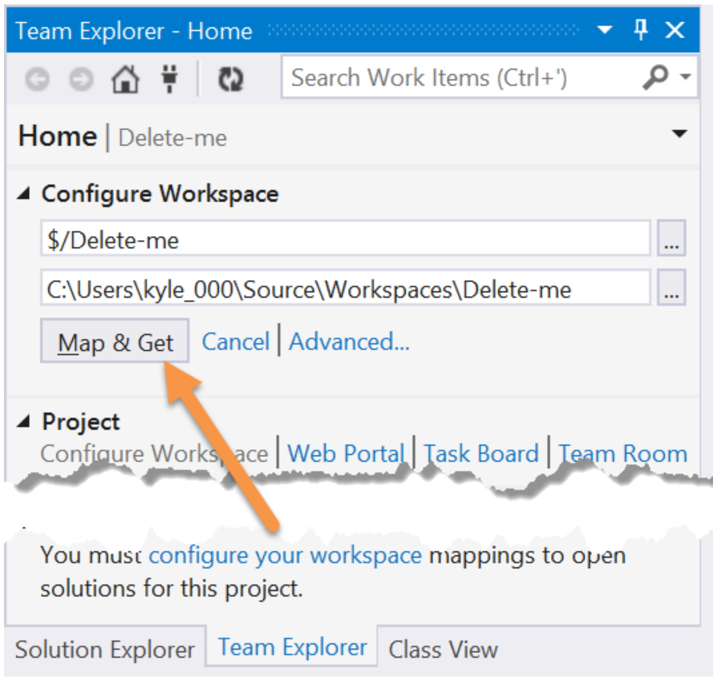


Figure 46: Team Explorer Configure Workspace options

After connecting to the Team Project and mapping your workspace, Team Explorer will indicate that no solutions currently exist for the Team Project (because you just created it) and give you the option to add a new one or open an existing one as shown in Figure 47. For this walk-through, you will create a new Visual Studio Solution so select the “New” command:

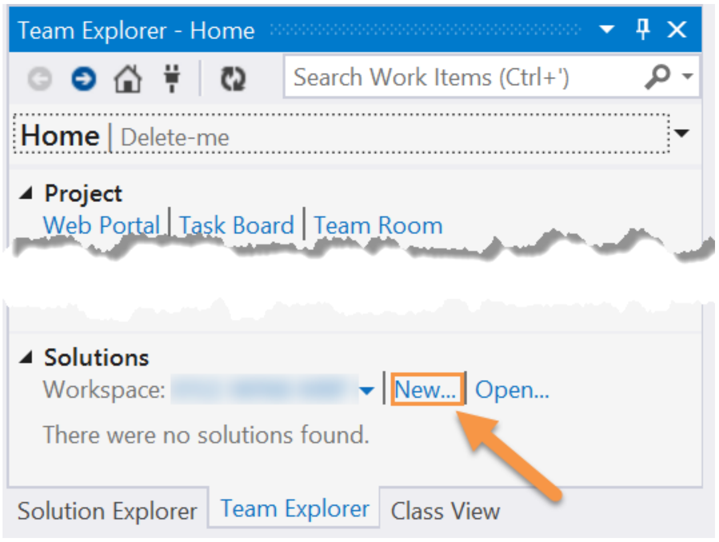
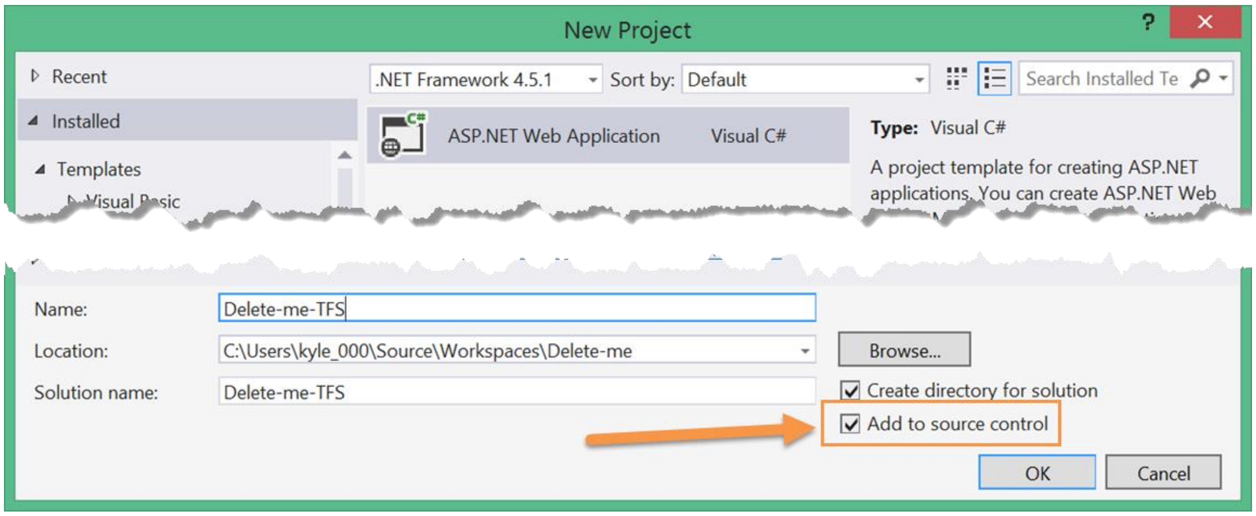


Figure 47: Team Explorer New Solution

After selecting “New…” Visual Studio will display the “New Project” dialog. You should select the ASP.NET Web Application project type and observe that the “Add to source control” option is selected as shown in Figure 48:



New Project

Recent

Installed

Templates

Visual Basic

.NET Framework 4.5.1 | Sort by: Default | Search Installed Te

ASP.NET Web Application | Visual C#

Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web

Name: Delete-me-TFS

Location: C:\Users\kyle_000\Source\Workspaces\Delete-me

Solution name: Delete-me-TFS

Browse...

☑ Create directory for solution

☑ Add to source control

OK | Cancel

Figure 48: New Project dialog

Select OK in the New Project dialog and, when prompted for additional options applicable to ASP.NET Web Applications, select the Empty project template and leave "Create remote resources" unselected because we're going to use the Azure Management Portal for that part in this walk-through. After selecting OK, Visual Studio will create your web project.

In order to provide something to visually verify the source control integration later, add an HTML page called "default.html" to your project in the Solution Explorer. Place whatever content you would like in the webpage. Once you're happy with the contents, save the HTML file and also ensure that your project and solution files have been saved by invoking the "Save All" command (Ctrl-Shift-S).

At this point, you have a local solution file with content in a workspace that is connected to source control but you have not yet committed any files to the remote repository. Right-click the solution file in Solution Explorer and select the "Check In" command. This will cause the Team Explorer window to display check-in options as shown in Figure 49. Complete the options and select "Check In". Visual Studio may confirm that you really want to check in your changes and will then interact with the Visual Studio Online Team Foundation Server Version Control service to commit your changes. After a few moments, the Team Explorer windows should display a message indicating that your change set has been successfully checked in.

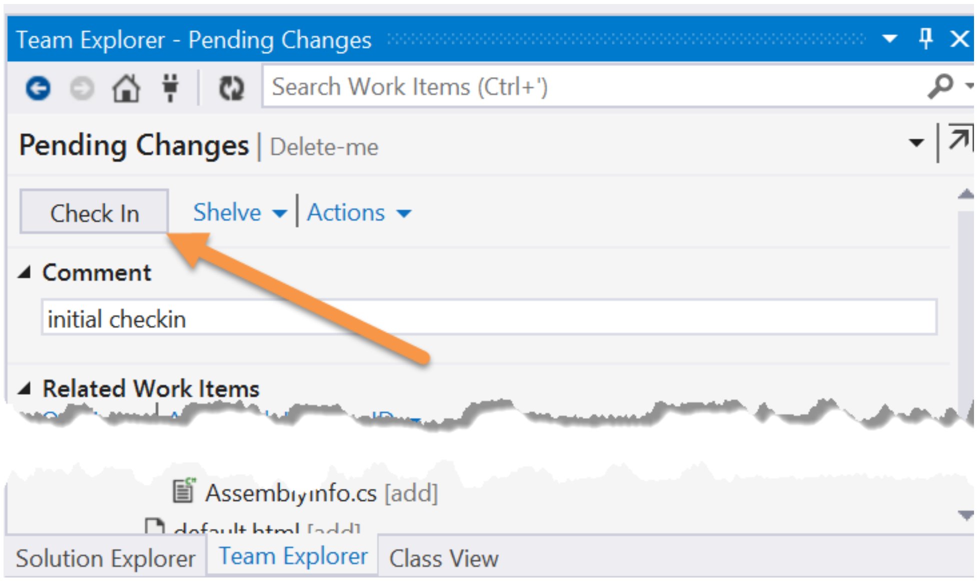


Figure 49: Check In Pending Changes

Now that you have checked in your changes and they are sitting in TFS Version Control, open the Azure Management Portal. Using what you learned in the "Provisioning Resources" chapter, create a new Azure Web Site using the "Quick Create" option. Once Azure has finished provisioning the website, you should be taken to the Quick Start screen for the new website. This is where the option to set up deployment from source control will be presented in the "Integrate source control" section as shown in Figure 50:

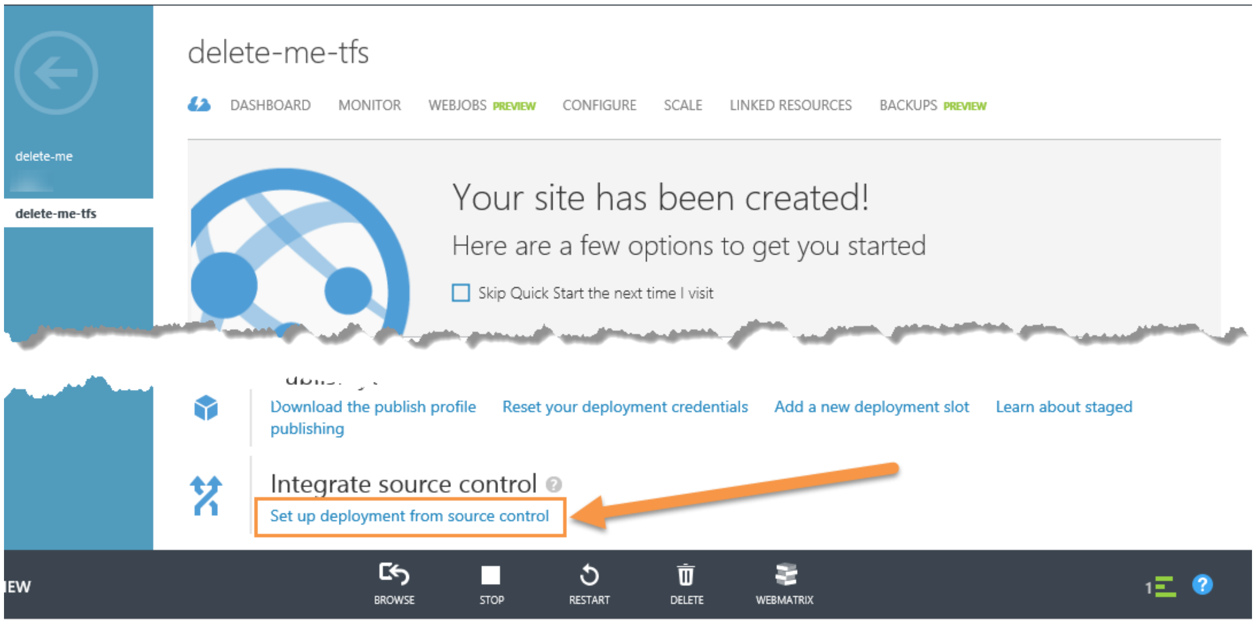

delete-me-tfs
DASHBOARD
MONITOR
WEBJOBS PREVIEW
CONFIGURE
SCALE
LINKED RESOURCES
BACKUPS PREVIEW
delete-me
delete-me-tfs
Your site has been created!
Here are a few options to get you started
Skip Quick Start the next time I visit
Download the publish profile
Reset your deployment credentials
Add a new deployment slot
Learn about staged publishing
Integrate source control
Set up deployment from source control
BROWSE
STOP
RESTART
DELETE
WEBMATRIX

Figure 50: Quick Start source control integration

Invoke the “Set up deployment from source control” command and the Azure Portal responds by first asking where the source for your project is located. The list contains a number of preconfigured providers as well as a configurable option to allow advanced configuration to point to a source control provider that is not on the list but uses a supported protocol. For this walk-through, select “Visual Studio Online” and advance to the next step in the wizard. At this point, you will be prompted for the URL of your Visual Studio Online account and asked to authorize the Azure Portal as a user of your Visual Studio Online repositories. Grant the requested permissions. The wizard will then ask you to select the appropriate repository from which your code for this website should be deployed. Select the project that you created at the beginning of the walk-through and invoke the “Complete” command.

At this point, your Team Project in Visual Studio Online should be successfully linked to your Azure Web Site and a message should be displayed in the Azure Management Portal indicating that the next check-in operation to source control will initiate a build-and-deploy cycle to your website. At this point, there are two options that can be used to cause the code to be deployed from TFS Version Control:

- Return to the Visual Studio Online project and go to the Build tab. Find the only entry under “Build definitions”, right-click the entry, and select “queue build…”. This will bring up the Queue Build dialog and you can simply select OK, accepting the default arguments. Use the refresh command while viewing the queue until the build and deployment complete.

- Within Visual Studio, edit “**`default.html`**”. Save your changes and check them into source control. Visual Studio Online will show a new build in the build queue for the project. Use the refresh command while viewing the queue until the build and deployment complete.

Regardless of which method you chose (or maybe you chose both), once the deployment completes, you can view your Azure Web Site and see the content that you created in Visual Studio and checked into Team Foundation Server Version Control. You can also view deployment activity in the Azure Management Portal’s Deployments tab.

Git

Tip: This section gives a basic overview of Git and its integration with Azure Websites. For more complete coverage of Git, check out Git Succinctly, a book written by Ryan Hodson. It is freely available as part of Syncfusion’s Succinctly series.

Git fundamentally differs from Team Foundation Server Version Control in that TFS is grounded around the concept of developers connecting to a single centralized “server” repository while Git is based on developers working in a more distributed manner, each having their own local repository and their changes pushed to and pulled from a shared repository as they are considered mature enough to introduce into the shared codebase.

For this walk-through, create a new folder on your computer and a file called **`default.html`** created with a text or HTML editor of your choice. This will be the content that is eventually deployed to your Windows Azure Web Site.

Now that you have content to deploy, go to the Azure Management Portal and, using what you learned in the “Provisioning Resources” chapter, create a new Web Site using the “Quick Create” option. Once Azure has finished provisioning the website, you should be taken to the Quick Start screen for the new website where the option to set up deployment from source control is presented in the “Integrate source control” section.

Invoke the “Set up deployment from source control” command and the Azure Portal responds by asking where the source for your project is located. Select “Local Git repository” from the list. At this point, the Azure Management Portal will create a new Git repository hosted in Azure and connected to your new Azure Web Site. Pushing to this repository will trigger a deployment from the repository. Note the Git URL that is provided on this screen (conveniently, Microsoft has included a “copy to clipboard” command for this element) because you will need it in future steps.

Back on your local machine, open a Git client of your choice and create a new repository with the folder and file created earlier. If you are using a command-line client, navigate to the desired directory and use the following commands to create the repository and perform the initial commit:

```
git init
git add .
git commit -m "initial commit"
```

Once the initial commit is complete, you can set up your Azure Web Site as a remote repository and push from your local repository to Azure. On a single line, enter the following command to add a connection to the remote repository (remember to use your own value for the Git URL in bold):

```
git remote add azure https://DeleteMeSiteDeployer@delete-me-
git.scm.azurewebsites.net:443/delete-me-git.git
```

Now, issue the following command to push from your local repository to the remote:

```
git push azure master
```

Git will prompt you for the password that you have set up for your deployment credentials and then proceed with pushing from your local master repository to the remote Azure repository. At this point, your content is deployed to your Azure Web Site and you can view this both by viewing the website in a browser and seeing the updated content, and by viewing the Deployments tab for the website in the Azure Management Portal.

Summary

In this chapter, you have learned about Team Foundation Server Version Control (and with it, Visual Studio Online) and Git, and seen how these can be used to achieve an Azure Web Site that is integrated with source control. This base can be built upon to achieve a solution in which changes can be quickly integrated into your live site simply by checking in code. For smaller sites with less stringent audit and control requirements, this can be an ideal deployment solution where you test changes on your local machine and then simply commit the changes to source control. In instances where a check-in should not be allowed to automatically change a live site, you may instead choose to use continuous integration with a target of a staging site that is then migrated to your live site once it is deemed production-ready.

Chapter 6 Accessing Data

Up to this point, most of the work that has been done producing any code for examples has been limited to static HTML because we were focused on concerns such as configuring source control and publishing content as opposed to writing code. In dynamic real-world applications, however, static HTML has limited value unless you also have some sort of server-side code interacting with data behind it. In this chapter, we will take a look at some of the options within Azure Websites for accessing and manipulating data. Some of these options are available out of the box with the Azure Websites service while others may require additional Azure services.

Windows Azure Table Storage

For years, relational database management systems using SQL to manage and retrieve data have been considered to be the go-to way for applications to work with data. This is based on advantages brought with them such as minimizing storage and improving consistency of data by normalizing structures and providing easy navigation of well-defined data structures through Structured Query Language (SQL). SQL databases do, however, run into many challenges when it comes to scaling massively. This has caused an explosion in the popularity of a category of database alternatives that are collectively referred to as NoSQL databases.

Azure Table storage is the default NoSQL database service within Azure and has shipped since the earliest Azure releases. It features the scalability and low storage costs associated with many NoSQL databases, and is easy to set up and configure within Azure.

For developers familiar with traditional SQL databases, some of the characteristics of Azure Table Storage that take a while to get used to are:

- While you will see references to columns in a table, the idea of a fixed schema is not present. Azure would allow you to have records stored in a single table where one record contains the data attributes appropriate to a bottle of pharmaceutical product and another record contains data attributes appropriate to a tricycle. This does not, however, mean that it is often a recommended to store vastly different entities within the same table.
- The concept of having multiple indexes is not present. Azure table storage segments data based on a single partition key, and then identifies rows within a partition by a single row key. Beyond these two keys, retrieving a subset of data based on the value of a data attribute will be an exercise in programmatically filtering a larger set of data. This is sometimes shielded from the developer somewhat by APIs that do this work for you.

Azure Table Storage does not come out of the box with your Azure Web Site service so, before you can utilize Azure Table Storage, you must first create a Storage account. This was described briefly in the earlier chapter “Provisioning Resources”. When you have provisioned a Storage account, you are set to start using Azure Table Storage in your application.

Note: Remember that Azure services are billed based on utilization when setting up any new service.

The first step in getting ready to use Storage from your web application is to configure a reference to the Azure storage library “WindowsAzure.Storage”. The easiest way to do this in Visual Studio is to right-click the web project and select “Manage Nuget Packages…”. The package manager dialog will display and you can search the online repository for “**WindowsAzure.Storage**”. Find the **Microsoft Windows Azure Storage Client Library** in the search results and install the package. This will install all of the libraries required to interact with the storage account.

After your project references are appropriately configured, the next step to set up your application to use Azure Table Storage is to create an entry for the Storage account in your connection strings. In this example, we’ll add an application setting to reflect the Azure Storage account that we’ll be using. Add the following to the **`appSettings`** in **`web.config`** (make sure to use your own account):

```
<add key="CloudStorageConnectionString"
value="DefaultEndPointsProtocol=https;AccountName=[youraccount];AccountKey=[y
ourkey]" />
```

Now that the connection string is available, you’re set to reference the storage in code. You’ll want to first make sure that the following are represented in the using statements at the top of your code module:

- **`Microsoft.WindowsAzure.Storage`**
- **`Microsoft.WindowsAzure.Storage.Table`**
- **`System.Configuration`**

The top-level object that gives you access to all of the storage functionality is the **`CloudStorageAccount`** object. In an Azure Web Site project, an instance of this object of this type is obtained through calling the class’s static **`Parse`** method, passing the connection string that you stored in the **`web.config`** such as in the following:

```
var account = CloudStorageAccount
  .Parse(ConfigurationManager.AppSettings[“CloudStorageConnectionString”]);
```

Now that you have access to your account, you can use the account object to produce an appropriately configured **`CloudTableClient`** which will be used to interact with the Azure Table Storage service. To access an instance of **`CloudTableClient`**, call the **`CreateCloudTableClient`** method of the **`CloudStorageAccount`** instance as follows:

```
var tableClient = account.CreateCloudTableClient();
```

With the **CloudTableClient** object, you can access and create tables within the storage account. In the following code, a reference to a table named “tricycle” is set up and the existence of the table is guaranteed prior to using the table by using the **CreateIfNotExists** method:

```
var table = tableClient.GetTableReference(“tricycle”);
table.CreateIfNotExists();
```

In order to work with the data in a table from .NET code, the easiest thing to do is to create .NET classes to represent the records in the table. These classes should inherit from the **TableEntity** class present in the **Microsoft.WindowsAzure.Storage.Table** namespace and, in addition to including the fields necessary to describe the entity, should establish the partition and row keys in the constructor. Continuing with our tricycle example, we’ll use the following to describe the entity that is to be stored in the “tricycle” table:

```
namespace TableStorageSample
{
    public class Tricycle : TableEntity
    {
        public Tricycle(string manufacturer, string serialNumber)
        {
            PartitionKey = manufacturer;
            RowKey = serialNumber;
        }

        public string Manufacturer
        {
            get { return PartitionKey; }
            set { PartitionKey = value; }
        }

        public string SerialNumber
        {
            get { return RowKey; }
            set { RowKey = value; }
        }

        public string Color { get; set; }
        public DateTime ManufacturedDate { get; set; }
    }
}
```

This class will be used to perform all of the Create, Read, Update and Delete (CRUD) operations with the table. These are accomplished using the **TableOperation** class which exposes static methods to do the following operations:

- **Delete**: Deletes the entity.
- **Insert**: Inserts a new entity into the table.

- **InsertOrMerge**: If the entity with a given key does not yet exist in the table, it will be inserted. If it does exist, the data attributes of the stored item will be merged with those of the entity passed to the method. In cases where the entity already exists, attributes of the entity will be updated to reflect the new values being passed in, but any attributes that exist on the stored record and are not provided in the new entity will remain stored as before.
- **InsertOrReplace**: If the entity with a given key does not yet exist in the table, it will be inserted. If it does exist, the stored version will be replaced with the data from the entity passed to the method. Unlike with the merge command, when the entity is already stored and the list of attribute differs between the stored and new entity, attributes which are not present in the new entity will not be retained.
- **Merge**: This operation merges the data in the entity stored in the table with a given key with that contained in the entity passed to the method. The object must already exist in the database or the operation will cause a **StorageException** to be thrown when executed.
- **Replace**: This operation replaces the entity stored in the table with a given key with the entity passed to the method. The object must already exist in the database or the operation will cause a **StorageException** to be thrown when executed.
- **Retrieve**: This operation is used to retrieve the entity with a given partition and row key. If the key specified does not correlate to an entity stored in the table, the fetch operation will contain a null result so, when retrieving directly by key, it is advisable to always check to ensure that the operation result contains a non-null value before attempting to process the record.

The **TableOperation** does not actually change any data itself but is passed as a sort of "command object" to the **CloudTable** object. The following code builds upon the earlier example to use the constructed table reference to retrieve an instance of a **Tricycle** entity where the partition key (**Manufacturer**) is "Schwinn" and the row key (**SerialNumber**) is "ABC123":

```
Tricycle tricycle = null;
var fetchOperation = TableOperation.Retrieve<Tricycle>("Schwinn", "ABC123");
var result = table.Execute(fetchOperation);
if (result.Result != null) tricycle = (Tricycle)result.Result;
```

If you wanted to update the stored data for the entity, you would retrieve the object (or potentially just create a new instance of the object with appropriate key values if you don't care what is currently stored), update field values, and then create an update operation such as **Replace** with the entity to execute with the **CloudTable** instance. Building off of the example where we just retrieved a **Tricycle** entity, the following code would be used to update the color and save the updated values to the database:

```
tricycle.Color = "Blue";
var updateOperation = TableOperation.Replace(tricycle);
table.Execute(updateOperation);
```

Azure Table Storage is relative easy to interact with, and the steepest part of the learning curve is that many developers today are just very accustomed to using relational databases. We've only scratched the surface enough to jumpstart your own exploration in this section so you will likely first want to explore filtering options and then batch operations.

Windows Azure Blob Storage

Azure Blob Storage utilizes the same storage account used by Azure Table Storage but is used to store unstructured data (files) rather than records in a database. These files can be optimized for transfer as read-only chunks called Block blobs. Block blobs can be quickly streamed sequentially to a client which works very well for many media types such as large video files, or they can be Page Blobs, optimized for random read/write access which is useful when the files are documents—or even a complete file system as is used when creating Virtual Hard Disk (VHD) files to mount as disk drives from Azure VM instances.

When building applications, Blob Storage should be considered whenever you have an application need on-premises such as writing files to the file system like in image-processing workflows. Because of the massive scalability achieved with Azure Storage accounts (at the time of this writing, all production storage accounts support scaling up to 200 terabytes and 20,000 requests per second) over what can be achieved by a hard drive on most on-premises servers, you may even find yourself considering Blob Storage for scenarios where you would like to have a file system solution but discounted the option out of concern that it would drag down your server with I/O requests when under heavy load.

In the following sections, interaction with Block and Page blobs will be demonstrated. In order to get started with the examples, you must first ensure that you have set up a Storage account in the Azure Management Portal and install the **`WindowsAzure.Storage`** Nuget package, both of which have been described in the previous section discussing Azure Table Storage. You will also need to make sure that any code files are referencing the following namespaces in the using statements:

- **`Microsoft.WindowsAzure.Storage`**
- **`Microsoft.WindowsAzure.Storage.Auth`**
- **`Microsoft.WindowsAzure.Storage.Blob`**
- **`System.Configuration`**

After your project references are appropriately configured, the next step to set up your application to use Azure Blob Storage is to create an entry for the Storage account in your connection strings. In this example, we'll add an application setting to reflect the Azure Storage account that we'll be using. Add the following to the **`appSettings`** in the **`web.config`** file (make sure to use your own account):

```
<add key="CloudStorageConnectionString"
value="DefaultEndPointsProtocol=https;AccountName=[youraccount];AccountKey=[y
ourkey]" />
```

The top-level object that gives you access to all of the storage functionality is the **CloudStorageAccount** object. In an Azure Web Site project, an instance of this object of this type is obtained through calling the class's static **Parse** method, passing the connection string that you stored in **web.config** such as in the following:

```
var account = CloudStorageAccount
  .Parse(ConfigurationManager.AppSettings["CloudStorageConnectionString"]);
```

Now that you have access to your account, you can use the account object to produce an appropriately configured **CloudBlobClient** instance, which is the primary object through which you will interact with storage. A **CloudBlobClient** instance is retrieved from the **CloudStorageAccount** object by calling its **CreateCloudBlobClient** method such as in the following code:

```
var client = account.CreateCloudBlobClient();
```

After obtaining the **CloudBlobClient** instance, you are ready to interact with blobs stored in your Azure Storage account. These blobs are organized into containers that are analogous to folders in the file system of your PC, with the exception that they are flat rather than forming a hierarchy. When the appearance of a hierarchical folder structure is desired, creative naming of these containers can help give that appearance. Whether working with Page or Block blobs, you start by obtaining a reference to the container in which the blob will be stored. This is done by calling the **GetContainerReference** of the **CloudBlobClient** object, passing the name of the container. For this walk-through, we call the **CreateIfNotExists** method of the retrieved **CloudBlobContainer** instance which will ensure that the container actually exists before we attempt to read from and write to it:

```
var container = client.GetContainerReference("container-name");
```

Reading and Writing Block Blobs

Access to blobs for reading or writing is made through the **CloudBlockBlob** class, which is obtained by calling the **GetBlockBlobReference** method of a **CloudBlobContainer** instance, passing it the name of the blob. The simplest way to store content in a Block blob is to use one of the **Upload** methods of the **CloudBlockBlob** class, which include:

- **UploadFromByteArray**
- **UploadFromByteArrayAsync**
- **UploadFromFile**
- **UploadFromFileAsync**
- **UploadFromStream**
- **UploadFromStreamAsync**
- **UploadText**

- **UploadTextAsync**

These methods are as intuitively named as they appear and allow you to specify various sources for the data which will be saved to the blob. In the following example, a new blob is created and populated with the contents of a file on the local file system:

```
var blob = container.GetBlockBlobReference("blob-name");
blob.UploadFromFile(@"c:\temp\myfile.dat");
```

There are also several options to read from a blob (including just providing a link directly to the blob). But the simplest way is to use one of the **Download** methods which include:

- **DownloadText**
- **DownloadTextAsync**
- **DownloadToByteArray**
- **DownloadToByteArrayAsync**
- **DownloadToFile**
- **DownloadToFileAsync**
- **DownloadToStream**
- **DownloadToStreamAsync**

As with the upload methods, these methods are very intuitively named and can be used to handle the downloaded content based on your needs. Additional **Download** methods exist but these are not for downloading the contents of a blob in a single call and, instead, are used to manage download of a blob in multiple calls (such as when a very large blob is to be downloaded and you want to create either an interruptible process or one that can communicate progress to the end user). In the following example, the contents of a blob containing text data are used to populate a **string** variable:

```
var blob = container.GetBlockBlobReference("text-blob-name");
string contents = blob.DownloadText();
```

Reading and Writing Page Blobs

While Block blobs are highly recommended for blob storage because they are much easier to deal with in an environment that must massively scale, there are times when random access is required and you can use Page blobs to meet this need. The **CloudPageBlob** class exposes the functionality and, similar to the Block blob, it is accessed by calling the **GetPageBlobReference** method of a **CloudBlobContainer** instance. The **CloudPageBlob** object has the methods described in the Block blob section for reading and writing data, but additional support exists for working with pages of data within the blob using methods such as **WritePages**.

Windows Azure SQL Database

In addition to the NoSQL Azure Table Storage service, Azure also makes available the ability to interact with a more familiar relational database through the SQL Database product. This service is very similar to the on-premises Microsoft SQL Server offering, with the list of features present in SQL Server that are not implemented in the Azure SQL Database product. This list seems to shrink with every release of Azure and I expect it will eventually whittle down to just those features that don't make as much sense in the Azure environment such as linked database servers. Some of the features currently available in the on-premises editions of Microsoft SQL Server that are not present in SQL Azure include:

- SQL Agent
- Master Data Services
- Change Data Capture
- FILESTREAM Data
- Replication
- Mirroring
- Limited XML support
- Backup/restore

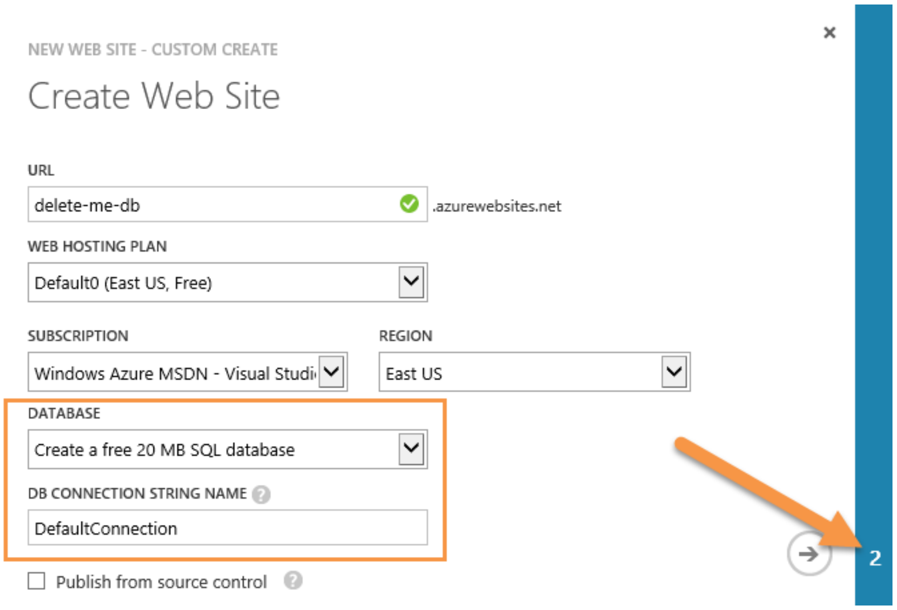
In addition to providing a familiar paradigm to developers who are used to working with relational data, Azure SQL Database uses many of the same libraries such as the ADO.NET data access API, LINQ to SQL, and Entity Framework. This means that moving many existing applications into the Azure environment from a data access perspective could be just a matter of changing connection strings.

Note: Entity Framework is Microsoft's currently recommended data access methodology of choice. The most recent versions include functionality optimized for Azure such as integrated support for retrying commands executed against unreliable connections.

While Azure SQL Database is a product offering of its own, it is also considered to be a part of the Azure Web Site offering and out of the box the Azure Websites product includes a free Azure SQL Database instance with 20MB of storage. This is not a lot of storage but, for many smaller websites that just need to manage a membership database and a little bit of information to drive what is displayed on webpages, it will be more than enough.

The easiest way to ensure that you can connect to an Azure SQL Database with your Azure Web Site is to select that database among the database options when creating the Azure Web Site. To this point, during all the walk-throughs and demonstrations that involved creating an Azure Web Site, either "quick create" was used or you were instructed to select "no database". But now it is time to explore those options.

To create an Azure SQL Database as part of a newly created Azure Web Site that is configured with connectivity enabled between the Web Site and the database, start by going to the Azure Management Portal and initiating the process of creating a new Azure Web Site with the Custom Create method. The Custom Create dialog is displayed, prompting for information such as the URL, billing subscription, and hosting region for the Web Site as well as a drop-down menu with database options. Selecting “Create a free 20 MB SQL database” in this drop-down menu adds a “DB connection string name” text box as well as a new page to the wizard as shown in Figure 51:


NEW WEB SITE - CUSTOM CREATE

Create Web Site

URL: delete-me-db .azurewebsites.net

WEB HOSTING PLAN: Default0 (East US, Free)

SUBSCRIPTION: Windows Azure MSDN - Visual Studi

REGION: East US

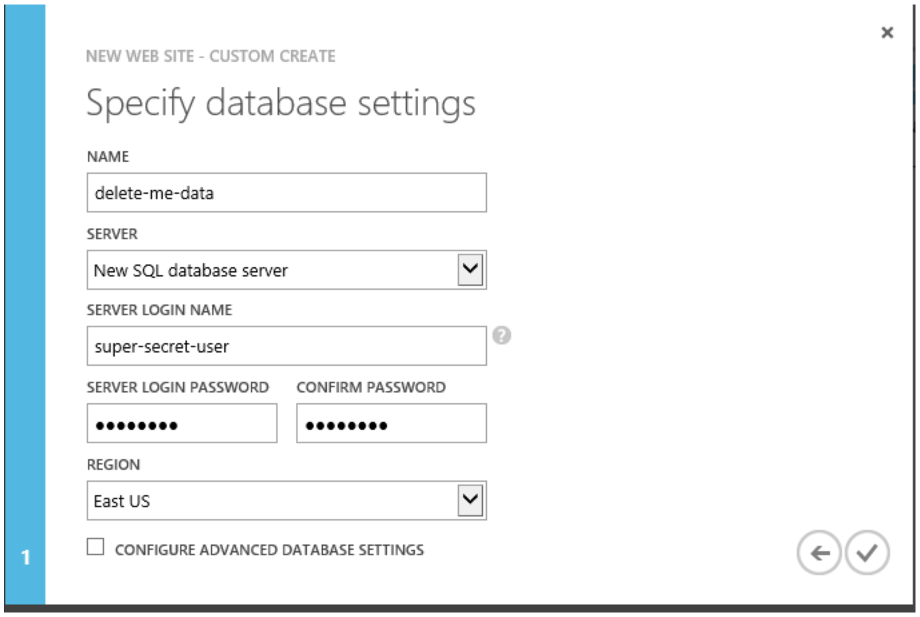
DATABASE: Create a free 20 MB SQL database

DB CONNECTION STRING NAME: DefaultConnection

☐ Publish from source control

Figure 51: Custom Create database selection

Advancing to the next step in the Wizard allows you to override the default name selected for your database with something that may be more meaningful to your application and prompts you to select a server to host your database. If you have not yet created a SQL Database in your Azure plan, the only option in the server drop-down menu will be “New SQL database server”. Selecting the “New SQL database server” option prompts you to specify the login credentials which will be used to connect to the database as well as the region in which the database server should be provisioned—which should be the same as the region in which the Azure Web Site is hosted to prevent having to jump across data centers when your application is retrieving and updating data. The completed form should look similar to what is shown in Figure 52:


NEW WEB SITE - CUSTOM CREATE

Specify database settings

NAME: delete-me-data

SERVER: New SQL database server

SERVER LOGIN NAME: super-secret-user

SERVER LOGIN PASSWORD: ●●●●●●●●●

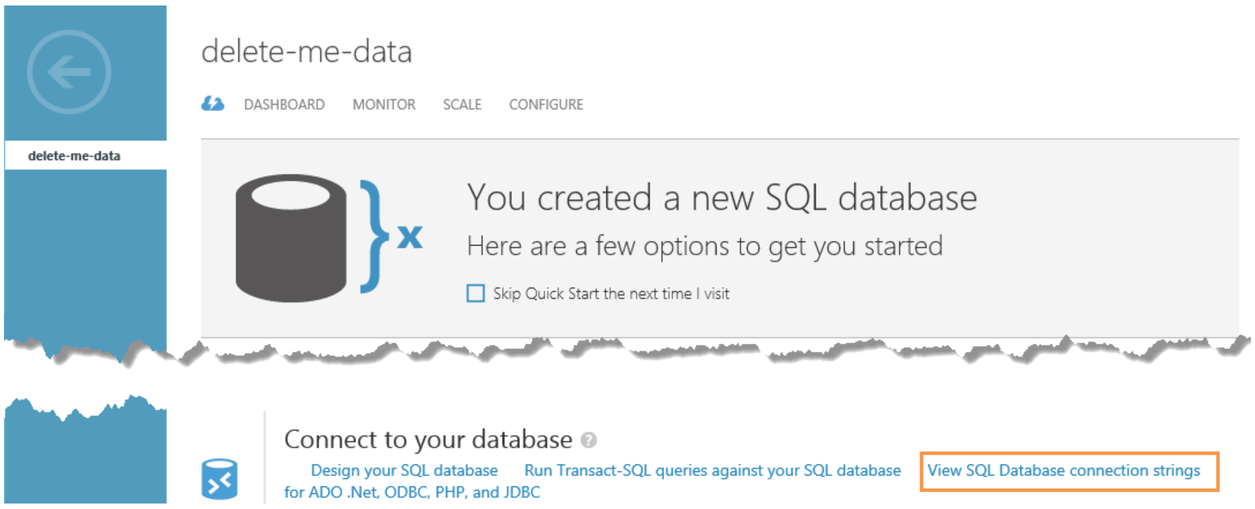
CONFIRM PASSWORD: ●●●●●●●●●

REGION: East US

☐ CONFIGURE ADVANCED DATABASE SETTINGS

Figure 52: Custom Create database settings

After you complete the wizard, Azure provisions the Web Site and SQL Database. Once complete, the SQL Database can be seen in the SQL Databases section of the Azure Management Portal as well as in the Linked Resources tab when viewing the Azure Web Site. Navigating to the SQL Database via either will display the SQL Database “Quick Start” screen that is shown in Figure 53, which includes the option to view database connection strings for various client connectivity methods:


delete-me-data

You created a new SQL database

Here are a few options to get you started

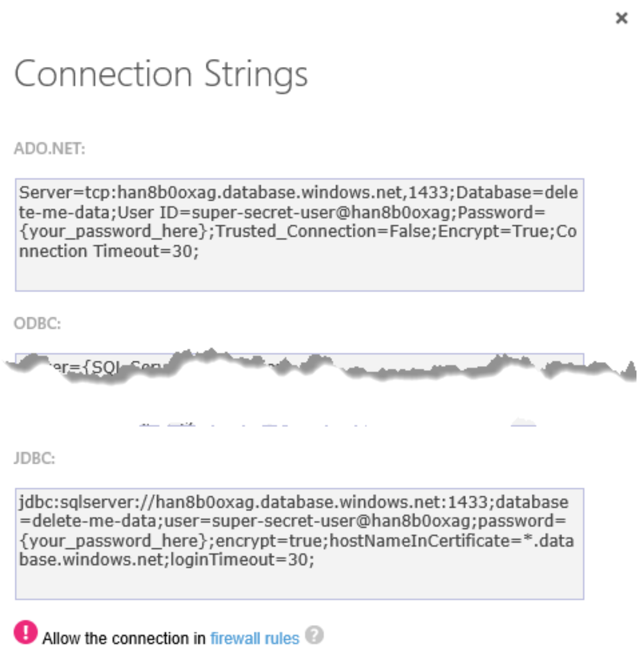
☐ Skip Quick Start the next time I visit

Connect to your database

Design your SQL database Run Transact-SQL queries against your SQL database View SQL Database connection strings for ADO .Net, ODBC, PHP, and JDBC

Figure 53: SQL Database Quick Start

Selecting the “View SQL Database connection strings” command will display the dialog shown in Figure 54 which provides the connection strings properly constructed for ADO.NET, Open Database Connectivity (ODBC), PHP, and JDBC. These connection strings are displayed within HTML **`textarea`** controls so that you can easily select the contents and then copy the value to your clipboard:



```
Server=tcp:han8b0oxag.database.windows.net,1433;Database=dele
te-me-data;User ID=super-secret-user@han8b0oxag;Password=
{your_password_here};Trusted_Connection=False;Encrypt=True;Co
nnection Timeout=30;
```

```
jdbc:sqlserver://han8b0oxag.database.windows.net:1433;database
=delete-me-data;user=super-secret-user@han8b0oxag;password=
{your_password_here};encrypt=true;hostNameInCertificate=*.data
base.windows.net;loginTimeout=30;
```

Figure 54: SQL database connection strings

Note: Heed the warning displayed at the bottom of the connection string dialog. By default, Azure SQL Databases do not allow connections from any client unless it has been specifically granted permission. This is handled automatically when setting up the Azure Web Site and selecting the SQL Database during the creation process. But, if you later try to connect to the database from some other source and do not configure the firewall rules, you could find yourself trying to figure out the connectivity problem for hours.

Now that you have set up your database, enabled connectivity between your Azure Web Site and the database (the setup wizard did this for you), and know the connection string, you're all set to use it within your application just as you would access an on-premises instance of Microsoft SQL Server.

Summary

In this chapter, we have covered some of the most common data access options used in Azure Websites to include the NoSQL Azure Table Storage option, the file-based approach with Azure Blob Storage, and the Azure SQL Database for a relational database option that is included with your Azure Web Site account. We have certainly not covered every possible database access option because there are a growing list of other database options hosted within Azure such as MySQL and Oracle, and you can also use advanced configuration options to connect to data hosted in your on-premises data center.

Chapter 7 Queuing

When I think about cloud computing, the first word that comes to mind is always either scalability or elasticity. The nature of PaaS platforms such as Windows Azure is to provide an environment that is conducive to always being just enough to meet your users' current demands while ready to, at a moment's notice, respond to a surge in demand. This type of capability in the platform does not, however, provide the instant guarantee that any system built on the platform will benefit from it, without building the application in a way that allows for it (I was once told the irreverent but accurate phrase, "Cloud doesn't fix stupid."). In this chapter, we will take a look at queuing, which can be used to help keep applications responsive under the most demanding circumstances.

Message queuing is a way for applications to record the need for work to be done within an application (or by a component outside the application) without needing to wait for the actual work to be completed prior to moving on. This not only allows for a quicker response to the user of an Azure Web Site than if the website does not respond until the task is completed, but also allows for components of the system to still function independently of each other when other parts of the application may be offline due to extreme load, catastrophic failure or some planned maintenance.

Besides removing the risk that pieces of a system will time out while waiting for other components to do work (a concept known as temporal decoupling), queuing in an environment also allows for application designs that allow for reacting to higher load periods by adding resources to work down queues or deciding instead to continue processing at a steady rate without adding additional resources and allowing the processing of queues to catch up during periods of less activity. This is known as "load leveling."

Windows Azure Queues is part of the Azure Storage service. With Azure Queue Storage, you can store messages that are up to 64KB in size for up to seven days and the queue can grow to a maximum size of 100TB. Windows Azure Queue Storage offers an API that allows applications to create, read, and peek at messages and leaves the task of monitoring the queue to applications consuming the service. In a typical usage scenario, Worker Role Cloud Service instances are programmed to poll the queue within a processing loop and, when messages exist, to process the next message on the queue with each pass.

Because Azure Storage Queues are designed to be fault tolerant, a key design feature is that a given message will never be assumed by the queue service to have been successfully processed by the worker that has picked it up. Instead, messages are assigned a timeout value where, if the worker processing the message has not yet explicitly instructed the service to delete the message, it will reappear in the queue to be available to the next client to retrieve a message. Other than the timeout value functionality, there is no transactional behavior associated with queuing. Therefore, a common and recommended pattern with queued solutions is to ensure that the application is written in such a way that the same message processed multiple times will have the same result. For example, a message-processing component that may store transient data to support processing multiple steps might have its first action in response to the message be to delete any residual data that exists as a result of having attempted to process the message previously or use the preexisting data to detect previous progress and resume where that left off.

Interacting with Azure Queue Storage is very similar to interacting with Azure Table and Blob Storage. A **`CloudStorageAccount`** object is initialized with a connection string, typically stored in the application's configuration file, and then is used to provide access to an object of type **`CloudQueueClient`** using the **`CreateCloudQueueClient`** method. The **`GetQueueReference`** method of the queue client is then used to provide access to a specific queue through an object of type **`CloudQueue`**, after which the following methods are most commonly used to provide access to all of the needed functionality for most basic queuing applications:

- **`AddMessage`**
- **`GetMessage`**
- **`DeleteMessage`**

The following code example demonstrates how an application which processes messages off of an Azure Storage Queue with 30 seconds between processing intervals might be constructed:

```
TimeSpan waitTime = TimeSpan.FromSeconds(30);
var account = CloudStorageAccount
  .Parse(ConfigurationManager.AppSettings["CloudStorageConnectionString"]);
var client = account.CreateCloudQueueClient();
var queue = client.GetQueueReference("my-queue-name");
queue.CreateIfNotExists();

while(isRunning)
{
    var message = queue.GetMessage();
    if(message != null)
    {
        // do something to process the message...
        queue.DeleteMessage(message);
    } // otherwise there was no message right now...
    Thread.Sleep((int)waitTime.TotalMilliseconds);

}
```

The web application can now place messages in the queue to get picked up by the worker process. The code to generate these messages in this example would appear as follows (notice the use of the Newtonsoft **`JsonSerializer`** to provide a compact string representation of the object to be included in the message):

```
var writer = new StringWriter();
var serializer = JsonSerializer.CreateDefault();
serializer.Serialize(writer, objectToSend);
string message = writer.ToString();

var account = CloudStorageAccount
  .Parse(ConfigurationManager.AppSettings["CloudStorageConnectionString"]);
var client = account.CreateCloudQueueClient();
var queue = client.GetQueueReference("my-queue-name");
queue.CreateIfNotExists();
queue.AddMessage(new CloudQueueMessage(message));
```

One important thing to remember when working with storage queues is that the pricing is based upon the number of times that you interact with the storage service. This means that there is a cost incurred each time that you peek into the queue, add a message, consume a message or delete a message. Because of this, a common pattern found to minimize the cost of checking an empty queue would be to build intelligence into the wait time between checking the queue that extends the time between loop iterations when the queue is found to be empty on a given iteration (up to the maximum acceptable amount of time for a message to sit) and decreases the wait time when messages are found to need processing—potentially even completely bypassing the wait time each time there is work to be done.

In addition to Azure Queue Storage, Azure Service Bus provides advanced queuing functionality that brings with it a more complicated API to represent the advanced capabilities such as larger message sizes, native WCF and WF support, and more durable message storage. Therefore, it will not be covered. I would recommend that anyone getting started with queuing to first use the Azure Queue Storage as a way to familiarize yourself with basic queuing concepts and, once comfortable with queuing, look to the features available in Azure Service Bus and problems that those features may solve.

Summary

In this chapter, you learned about the advantages of queuing in any application that must exhibit the scalability and resilience expected from cloud applications. Additionally, you learned how to interact with the Azure Queue Storage service, and you were exposed to Azure Service Bus as a more advanced alternative. I would strongly encourage any developer looking to build solid applications either in a cloud or enterprise environment to spend time learning more about queuing and making it an integral part of the way that you think about application design.