

Azure

IN ACTION

Chris Hay
Brian H. Prince

MEAP

 MANNING





MEAP Edition
Manning Early Access Program

Copyright 2009 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

Please post comments or corrections to the Author Online forum:
<http://www.manning-sandbox.com/forum.jspa?forumID=579>

Contents

Preface

Acknowledgments

About this book

About the authors

Part 1 Welcome to the cloud

1. Getting to know Windows Azure
2. Your first steps with a Web role

Part 2 Understanding the Azure service model

3. How Windows Azure works
4. It's time to run with the service
5. Configuring your service

Part 3: Running your site with Web roles

6. Scaling Web roles
7. Running native, full trust, and non .Net applications

Part 4 Working with blob storage

8. The basics of blobs
9. Uploading and downloading blobs
10. When the blob stands alone

Part 5 Working with structured data

11. Simple table storage
12. Working with the Table REST API
13. SQL Azure and relational data
14. Data storage patterns

Part 6 Doing work with messages

15. Processing with worker roles
16. Messaging with the queue
17. Connecting with .NET Services

1

Getting to know Windows Azure

This chapter covers:

- Overview of Windows Azure and the Windows Azure Platform
- Building your first Windows Azure Web Role
- Core Cloud Concepts

Imagine a world where your applications were no longer constrained by hardware and you could just consume whatever computing power you needed, when you needed it. More importantly imagine a world where you are only paying for the computing power that you use.

Now that your imagination is running wild, imagine you no longer need to care about the management of hardware infrastructure and you can focus just on the software that you develop.

If this is the sort of thing you daydream about then you should burn your server farm and watch the smoke form into a cloud in the perfect azure blue sky. Welcome to the Cloud and welcome to Windows Azure. I also suggest that if this is the sort of thing you daydream about you may wish to lie to your non IT friends ;)

Cloud Computing is like a big scary monster

With Cloud Computing in general there are lots of new concepts that we will slowly introduce you to throughout this book eventually giving you the complete picture but for this chapter we will keep things relatively light and simple and try not to scare you off. As you get more comfortable with this new paradigm we will introduce more of the complexities as the book progresses.

If Cloud Computing is a big scary monster, then think of this book as monster therapy. Let's say you were walking down the street and you saw a big scary monster, you would probably jump out of your skin. However, if you saw lots of cute little monsters that got progressively scarier you'd probably be okay when you saw that same big scary monster. Throughout this book we will introduce these new concepts as cute little monsters allowing you to eventually see that same monster but keep your skin intact. Obviously we would also like you to avoid being eaten too ☺

Over the course of this chapter we'll look at the following concepts

- What is the Windows Azure Platform?
- What is Windows Azure?
- Developing your first Web Role
- Saving money by running in the Cloud
- Overview of the other Windows Azure Platform services

To get the ball rolling, we'll start by looking at the big picture by looking at the entire platform.

1.1 What is the Windows Azure Platform?

As you might have gathered from the opening section, the Windows Azure Platform really describes Microsoft's complete Cloud offering. Every service that Microsoft considers as part of the Cloud will come under this banner.

The whole Cloud thing passed me by

There isn't really anything magical about the Cloud. It really just means a bunch of servers that can host and run your applications, or an offering of services that can be consumed (e.g. a Web Service ☺).

The main difference between a Cloud offering and a non-Cloud offering is that the infrastructure is abstracted away (i.e. you no longer care about the physical hardware that hosting your service)

Finally most Public Cloud solutions are offered as a metered service, i.e. you pay for the resources that you use (compute time, disk space, bandwidth etc) as and when you use them.

As of today (PDC09 release) the Windows Azure Platform really splits into 3 parts:

- The Operating System for the Cloud (Windows Azure)
- A Relational Database in the Cloud (SQL Azure)
- Enterprise Services in the Cloud (Windows Azure platform AppFabric)

As I said earlier, you can expect the above list to expand over time, in-fact I wouldn't be surprised if we saw Microsoft Flight Simulator in the Cloud ;)

As cool as AppFabric and SQL Azure are, for just now we are going to stay focused on the Windows Azure part of the Windows Azure Platform and ignore all the other platformy stuff until the end of the chapter.

The names confuse me

So this is where the naming gets a little confusing straight off. Unfortunately when most folks refer to Windows Azure it's not clear if they are referring to Windows Azure or the Windows Azure Platform.

It's not unlike the ESPN naming convention. The ESPN network has multiple channels (ESPN, ESPN2, ESPN News etc) yet we tend to refer to these channels collectively as ESPN rather than as the ESPN Network. To confuse matters further we also refer to the individual ESPN channel as ESPN also. Therefore if you ask someone what game is on ESPN tonight, it's not clear if are you meaning all the channels on ESPN (including ESPN News and ESPN2) or are you meaning just the channel named ESPN (not including ESPN2 etc).

To keep things consistent in this book whenever we talk about the platform as a whole we will refer to "The Windows Azure Platform" or "The Platform" but if we are talking about the core Windows Azure product then we will use the term "Windows Azure".

1.1.1 It must be an operating system it has Windows in the title

Windows Azure is what Microsoft calls their core *operating system* for the Cloud. Ok so now you know what Windows Azure is we can skip on right? I didn't think so! So let's strip away all the hype, what does it actually give us?

Well, Windows Azure simply provides us the ability to run our applications in a highly scalable manner on Microsoft servers in Microsoft's Data Centers in a manageable way. This basically means we can either host our web applications (e.g. Websites that sell Hawaiian shirts) or some backend processing services (e.g. an MP3 to WMA file converter) in Microsoft's Data Centers. If we need more computing power to run our application (i.e. more instances of our website or most instances of our backend service), we can just spin up more instances of our application that will be spread across many servers. By increasing the number of instances of our application, we will ultimately be able to process more data or handle more incoming traffic.

Hmmmm, How exactly is that an operating system? Well, to answer that we have to define what it really means to be a Cloud operating system.

The first thing we have to consider is that when Microsoft refer to Windows Azure as an operating system for the Cloud they don't literally mean an operating system as you may know it (Windows 7, Windows Vista, Leopard, Snow Leopard etc). What they really mean is that Windows Azure performs the same sort of job that a traditional Operating System may perform.

So what does an operating system do? Well it really has 4 tasks in life:

- Host and run our applications
- Abstract the complexities of hardware from our applications
- Provide an interface between users and applications
- Funds the development division of a very large blue company that produces Visual Studio

Figure 1.1 shows how these tasks are achieved in traditional operating system on a normal PC.

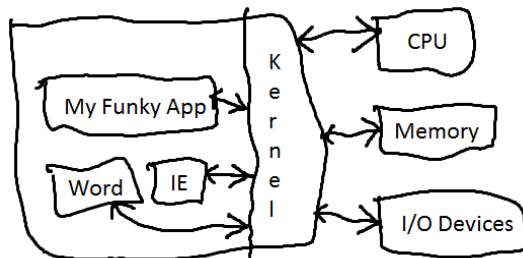


Figure 1.1 - A typical representation of an operating system how it interacts with applications and resources. Notice that applications do not directly interact with CPU, Memory and Devices.

Figure 1.1 shows a set of application running with an Operating System. As you can see from the diagram since the applications have no direct access to the hardware meaning that all interactions with the hardware must come through the Kernel.

The Kernel

The Kernel is the low level operating system component that pretty much performs all the tasks we will describe i.e. Process Management, Memory Management and Device Management.

Actually this analogy of Windows Azure being an Operating System looks like it could work out. Over the next few sections we will continue with this analogy and see how Windows Azure fares as an Operating System, by doing this we should get a pretty good overview of how Windows Azure works and what services it provides.

1.2 Hosting and running applications

So number 1 on the list is hosting and running applications. I kind of think this is the most important task of an operating system. Without applications we are really just moving a mouse around with no purpose ☺

Let's get started by looking at the types of applications that can be run in traditional operating systems as well as Windows Azure.

1.2.1 Types of Applications

In traditional operating system such as Windows 7, we could consider most things as an application e.g.

- Microsoft Word (yep it's an app)
- Internet Explorer or Firefox (still an app)
- Killer Mutant Donkey Blaster game (even that is an app)

Figure 1.1 shows some applications running in the context of normal PC operating system. Instead of hosting client applications (games, Word, Excel etc), the types of applications that you host in Windows Azure would be server applications such as:

- Web Applications (e.g. a Hawaiian Shirt Shop Website)
- Background Computational Applications (e.g. an MP3 Converter)

Figure 1.2 shows some server applications running in a traditional operating system

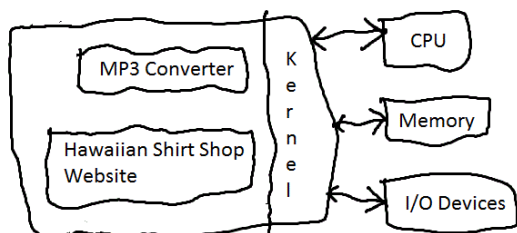


Figure 1.2 Windows Azure type applications running in a normal OS

As you can see from Figure 1.1 and Figure 1.2 there is no real difference between Excel and a Hawaiian Shirt Shop Website they are simply applications as far as a traditional operating system is concerned.

1.2.2 Running Applications across Thousands of Servers

In traditional operating systems, the operating system is responsible for allocating CPU time and memory space so that our application can run (as seen in Figure 1.1 and 1.2).

Not only is the operating system responsible for allocating these resources, it's responsible for managing these resources. For example if an application fails, then it's the Operating Systems job to clean up the applications resource usage and restart the application if necessary.

This level of abstraction is perfect for an Operating System that manages a single server but this just isn't scalable for a Cloud Operating System. With Windows Azure our application doesn't just run on a single server, it can potentially be running in parallel on thousands of servers. A Cloud Operating system can't be responsible for allocating CPU time and memory on thousands of physically separate servers. We need to abstract to a higher level.

In the case of Windows Azure that higher level of abstraction is Virtual Machines. Figure 1.3 shows how our applications could be distributed amongst the Virtual Machines in a Windows Azure Data Center.

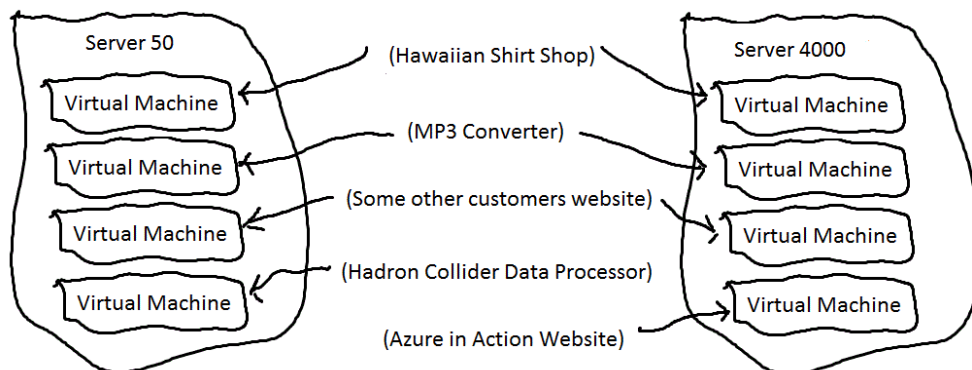


Figure 1.3 – Application split across Virtual Machines in the Windows Azure Data Center

As we can see from Figure 1.3 our Cloud Operating System is no longer responsible for assigning our applications resources by CPU and memory but is instead responsible for allocating resource by virtual machines. In this diagram our Hawaiian Shirt Shop Website is allocated 2 virtual machines hosted in 2 different physical servers, whereas the Azure in Action Website is only allocated a single Virtual Machine (shirt shops make more money so they get more roles).

VIRTUAL MACHINES

Finally Figure 1.4 shows what the virtual machine hosting a web application roughly looks like.



Figure 1.4 – A logical representation of the virtual machine that hosts our web application.

In figure 1.4 we can see the Virtual Machine hosts the Web Application within IIS7.0. As we could see from Figure 1.3 each physical server is split up into multiple virtual machines. Every instance of your service (web or worker role) is installed onto its own virtual machine which is a base installation of Windows 2008 server (with some extra Azure bits) as shown in Figure 1.4. The use of virtual machines is how Windows Azure achieves separation of services across physical servers; this means that another instance of a role hosted on the same physical hardware cannot interfere with your role instance.

Although your application runs on a virtual machine, the virtual machine is abstracted away from us and we only ever have a view of the role instance, never of the virtual machine. You should also notice that a single instance of your web application is assigned to a single virtual machine (also no other applications will be assigned to that virtual machine). This ensures that every instance of your web application is isolated from other applications running on the same physical server. The final thing to note that the Virtual Machine image also runs an agent process; we'll explain what this does a little later on in the book.

Web Role

A role is simply another name for your application. It actually refers to the base VM image that hosts your application. A web role is simply VM that hosts your application within IIS.

To be honest I'm now itching for a bit of code. Let's look at how we can build a simple ASP.NET Website that we can run in one of those Windows Azure Virtual Machines.

1.2.3 Building your first Windows Azure Web Application

Although we are going to build an ASP.NET Website in this example, the good news is that pretty much any website that can be currently hosted in IIS on Windows Server 2008 can be hosted in Windows Azure.

The following web applications are examples of the types of web applications supported out of the box:

- ASP.NET 3.5 Web Applications
- ASP.NET MVC 1.0 Web Applications

- Web Services (WCF & ASMX)
- Any FastCGI based Website
 - PHP
 - PythonEtc

Although Windows Azure supports the ability to host different types of website, for now we will just create a simple hello world web application using ASP.NET 3.5SP1. Later on in the book we will look at how you can create PHP websites, WCF Web Services and ASP.NET MVC Websites.

SETTING UP YOUR ENVIRONMENT

To get started developing an ASP.NET 3.5SP1 Website you will need to download the Windows Azure Software Development Kit (SDK). The SDK contains a whole bunch of things that will make your life easier when developing for Windows Azure including:

- Windows Azure Development Fabric (a simulation of the Live Fabric)
- Visual Studio Templates for creating Web Applications
- Windows Azure Storage Environment
- Deployment Tools
- A glimpse of a new bright new world (with a slight blueish tinge)

In the next chapter we will take a deeper look at some of the items in the SDK. For now we will just install it. If you are an experienced ASP.NET Developer you should be able to just install the SDK by clicking the next button a few times.

Before installing the SDK, check your version of Windows and Visual Studio.

SUPPORTED OPERATING SYSTEMS

Before you attempt to install the SDK, it's worth making sure that you have a suitable version of Windows. At present the only versions of Windows supported are:

- Windows 7 (which you should be running because it's lovely)
- Windows Vista
- Windows Server 2008 (and beyond)

Note: You cannot install the SDK on Windows XP or Windows 2003

Please note that Windows XP is not supported for Windows Azure. Before you jump up and down about Windows XP, there isn't some conspiracy against it (although there should be). The reason that XP isn't supported is that Windows Azure Web Roles are heavily built upon IIS 7. Windows XP (and Windows 2003) use earlier versions of IIS that won't work with Windows Azure. Later on in the book, we will look exactly what features of IIS7.0 is dependent upon and why it cannot work in earlier versions of IIS.

SUPPORTED VERSIONS OF VISUAL STUDIO

One final point before you go an install the Windows Azure SDK. To develop Windows Azure applications in Visual Studio, you will need either Visual Studio 2008 or Visual Studio 2010. If you are still running Visual 2005, you now have the excuse you needed to upgrade. If you can't afford Visual Studio or your company won't upgrade you (stupid company), then you can either use the Visual Studio 2010 Beta (until it time bombs), or you can use Visual Studio Web Developer 2008 Express Edition (which is of course, free).

Now that you have installed the Windows Azure SDK, let's fire Visual Studio up and create our first Azure web application.

FIRING UP VISUAL STUDIO

In order to launch your Windows Azure Application in the Development Fabric from Visual Studio you will need Administrator privileges. It's worth getting into the habit (for Azure development) of right clicking on your Visual Studio icon, and selecting (run as administrator).

NEWING UP OUR PROJECT

Our first step is to 'new up the project'. To do this, we need to open Visual Studio, and select File > New > Project. You should select the "Cloud Service" project type, which will then give you the option to select the "Cloud Service" template as shown in Figure 1.5.

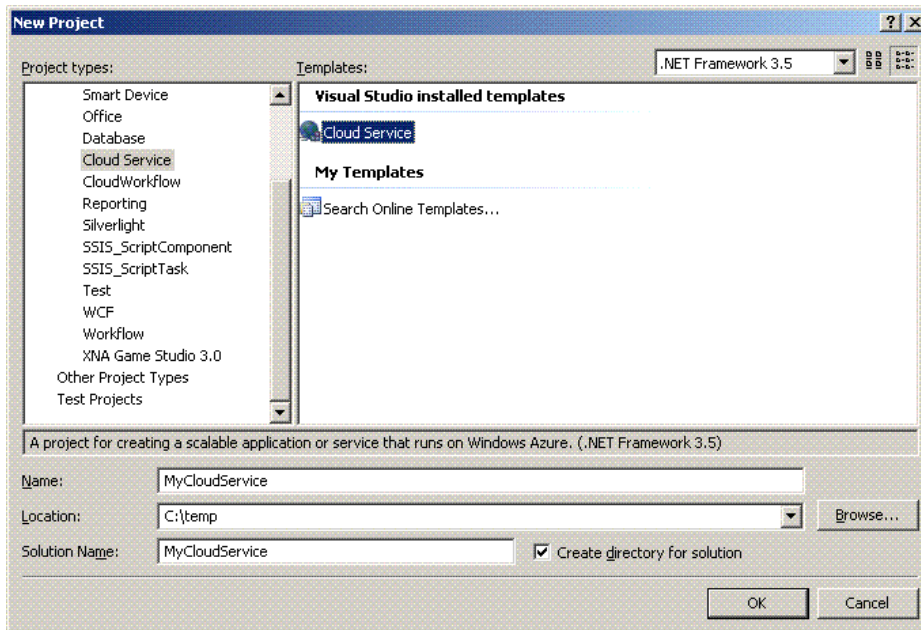


Figure 1.5 - The Cloud Service template in the New Project dialog of Visual Studio

Once you have selected the "Cloud Service" template, you should set the name for your project and solution then click OK. You will then be presented with the dialog as shown in Figure 1.6 that will allow you to select the type of Windows Azure project that you wish to create.

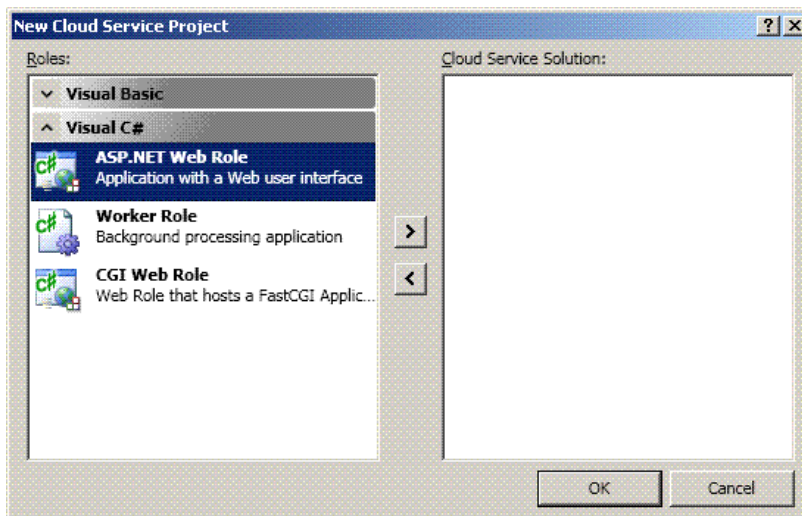


Figure 1.6 - New Cloud Service Project dialog.

As you can see from figure 1.6, you are able to create:

- ASP.NET Web Roles
- Worker Roles
- CGI Based Web Roles

It is also worth noting that you can create your projects in either Visual Basic or C#. In this book we will be using C# rather Visual Basic. This is no disrespect to Visual Basic but we have found over time that C# developers are typically not comfortable with Visual Basic but Visual Basic developers are comfortable with both languages (I guess you have to be though since most samples are in C#).

You should now select the "ASP.NET Web Role" project and click on the right arrow button so the project is displayed in the right hand listbox as shown in figure 1.7.

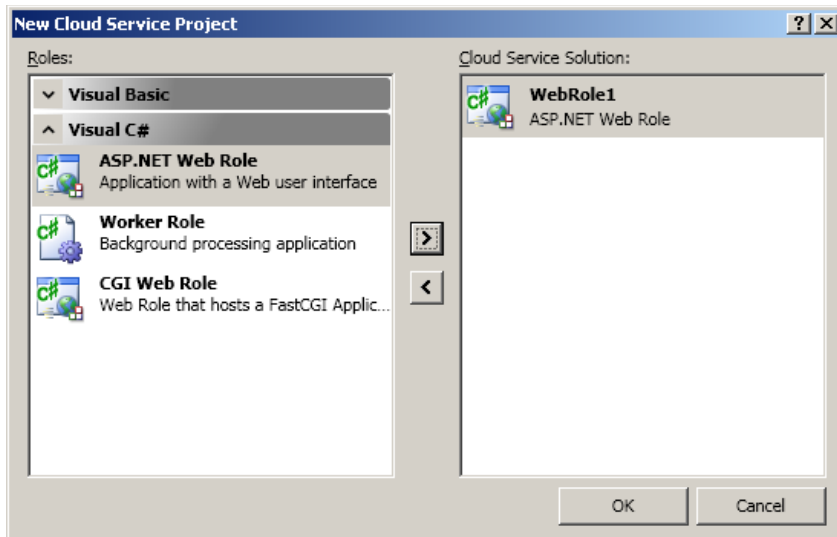


Figure 1.7 - Selected a web role project from the new cloud service project dialog

Now that you have selected your web project, you can click OK and wait for Visual Studio to generate your solution.

Once Visual Studio has come out of project creation meltdown, it will have create you a new solution with 2 new projects as shown in figure 1.8

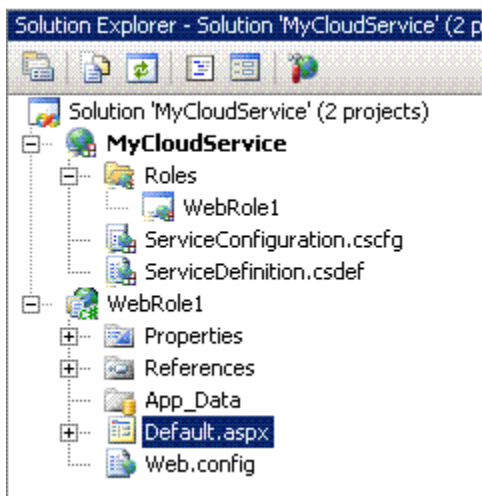


Figure 1.8 - Solution Explorer for our newly created web role project

The 2 projects as shown in Figure 1.8 are:

- Our Windows Azure Cloud Service Project
- A plain old ASP.NET ASP.NET 3.5 Web Application

The first project (MyCloudService) is our Cloud Service Project. This project contains a bunch of configuration that is specific to our Windows Azure Web Role. For now we won't look at the contents of this project and instead save that for later a chapter.

MODIFYING OUR WEB PAGE

As you may have already gathered the second project (WebRole1) is just a regular old ASP.NET Web Application. Therefore you can modify the Default.aspx file as you would normally. In this case you should modify the file to display Hello World as shown below:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="WebRole1._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Hello World</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      Hello World
    </div>
  </form>
</body>
</html>
```

RUNNING THE WEB PAGE

Now that we have created our web page we just want to fire it up in our development fabric. Before you run your new Web Role, you must ensure that the Cloud Service is your startup project rather than the ASP.NET project. By default Visual Studio will do this for you on creation of your new project.

Now for the exciting bit, you are now about to run your first web application in the Windows Azure Development Fabric. Drumroll..... Just hit F5 like any other Visual Studio application.

Once you have hit F5, Visual Studio will fire up the development fabric and launch your web page in your browser just like any other web page. Unlike regular ASP.NET Web Applications, the Development Fabric will host your web page rather than the Visual Studio Web Development Server (Cassini). Figure 1.9 shows our web page running in the development fabric.

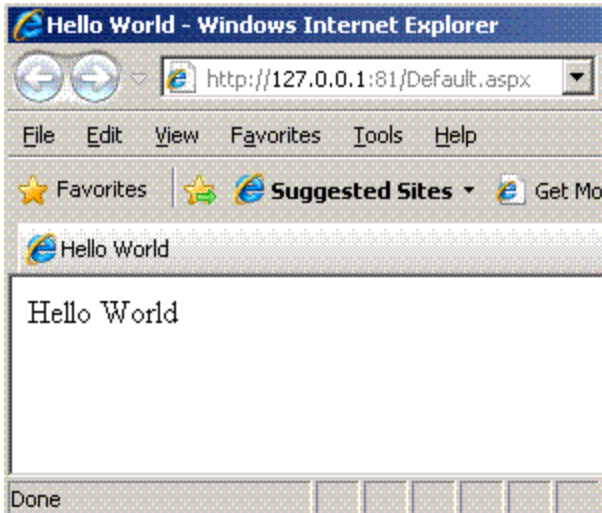


Figure 1.9 - ASP.NET 3.5 Hello World running in the development fabric

Congratulations, you have now developed your first cloud application. In the next chapter we will look in more detail at the development SDK, the development fabric and how to deploy your service to the live production servers.

1.2.4 How does it all fit together?

Later on in this chapter, we will take a look at how we can take our web role and scale it (Hello World web applications often need multiple instances due to the levels of traffic they receive) but before we do that we need to have an understanding of the overall Windows Azure logical infrastructure as shown in Figure 1.10

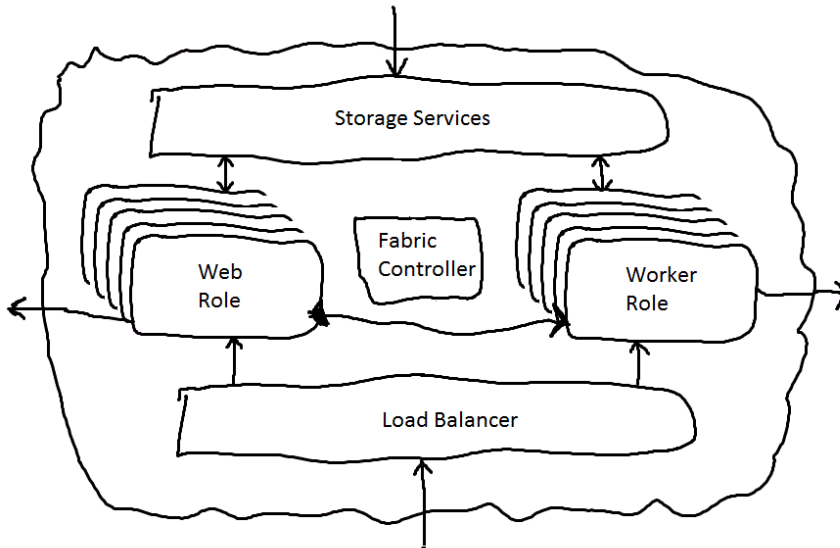


Figure 1.10 - Windows Azure

As you can see from Figure 1.10 the Web Role is just one part of the overall infrastructure. Over the next couple of sections we will look at how some of the other components fit into the overall service namely:

- Worker Roles
- Fabric Controller
- Storage Services

Before we do that it's worth pointing out the Load Balancer.

LOAD BALANCER

In figure 1.10 you will notice that neither our web roles (web applications) nor our worker roles (background services) have direct incoming traffic from the internet. In the case of both worker and web roles all incoming traffic must be forwarded on via the load balancers. The load balancers provide 4 very important functions:

- Reduces Surface Attack Area (improves security)
- Load Distribution (incoming requests can be forwarded to multiple instances)
- Fault Tolerance (traffic is routed to another instance during a fault)
- Maintenance (during an upgraded load balancer can route traffic to another instance)

Not only can a role receive incoming traffic, it should be noted that the web role can initiate communication with services hosted outside of the Windows Azure data centers, roles inside of the data center as well as storage services.

Now that you understand how Windows Azure can distribute requests across multiple instances of web roles using the load balancer, we can expand upon that knowledge and looking at some of the demands of scaling web applications slightly later on.

Before we look at scaling however, we will take a brief look at the other type of supported role, i.e. the Worker Role.

1.2.5 Worker Roles

TODO

So we have explored the first task of an operating system (hosting and running applications). I think we agree that Windows Azure ticks that box.

What we haven't explained is where the Kernel fits into this picture of the Operating System? We need something that will manage our applications and all our Virtual Machines running in the Windows Azure Data Center. In Windows Azure the Kernel is really the Fabric Controller.

1.3 The Fabric Controller

The Fabric

Windows Azure follows a model of Cloud Computing known as "The Fabric". The Fabric is really another way of describing the data center. Like "The Matrix", "The Fabric" is everywhere. Every single piece of hardware (server, router, switch, network cable etc) and every virtual machine is connected together to form the Fabric. Each resource in the Fabric is designed and monitored for fault tolerance. The Fabric forms an abstract representation of the physical data center allowing applications to be run within the Fabric without knowledge of the underlying infrastructure.

The Fabric is managed by a software overlord known as the Fabric Controller. The Fabric Controller is aware of every hardware and software asset within the Fabric. It is responsible for the installation of your web and worker roles onto the physical or virtual servers living within the Fabric (this is not unlike the Kernel assigning memory or CPU to an application in a traditional operating system). The fabric controller is responsible for maintaining its inventory by monitoring the health of all its assets. If any of the assets are unhealthy it is responsible for taking steps to resolve the fault. This may include:

- restarting your role

- restarting a server
- reprogramming a load balancer to remove the server from the active pool
- managing upgrades
- moving instances of your role in fault situations.

Figure 1.11 shows how the fabric controller monitors and interacts with the servers.

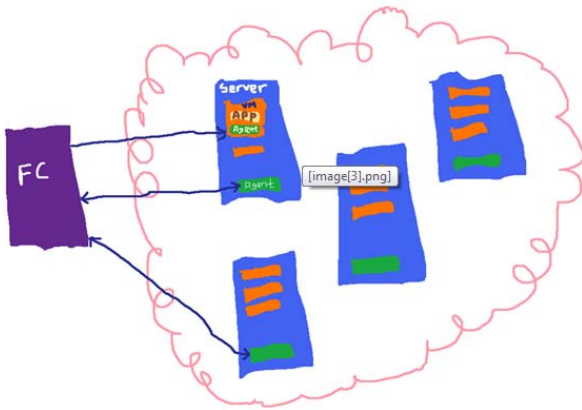


Figure 1.11 – The Fabric Controller: TODO new diagram and caption

As you can see from Figure 1.11, the Fabric Controller is performing the job of the kernel (except across multiple servers at a server level rather than CPU + Memory Level) in respect to the following items:

- Allocation of Resources
- Monitoring of Resources

One of the jobs that the Fabric Controller doesn't do that a Kernel will do, is the abstraction of the IO Devices. This job is performed by Storage Services.

1.4 Storing Data in the Cloud

Before we look at how Windows Azure abstracts storage in the cloud from the physical hardware. It's worth going back to our analogy of Windows Azure being an Operating System for the Cloud.

1.4.1 Abstracting the physicality's in a Regular OS?

Let's say you are developing a new timesheet application for Windows 7, and for this application you may wish to convert MP3 files to WMA. In order to be able to convert our MP3 file, we first need to read the file from a hard disk (and eventually write the result).

Even though there are thousands of different disk drives you do not need to concern yourself about the implementation of these drives as the operating system provides you an abstracted view of the disk drive. If you need to save your converted file to the disk, you can just write the file to the file system and the operating system will manage how this is written to the physical device. The same piece of code that you would use to save your timesheet will work regardless of the physical disk drive.

In the same way that Windows 7 abstracts the complexities of the physical hardware of a desktop PC from your application, Windows Azure abstracts the physical cloud infrastructure from your applications via configuration and managed API's. So for the MP3 example in Windows Azure rather than abstracting a single disk from our application, Windows Azure needs to abstract our application from the physical server (not just the disk). Also the data would need to be stored in abstracted storage array which is no longer tied to a physical server and can be accessed by multiple physical servers. Figure 1.11 shows this logical abstraction.

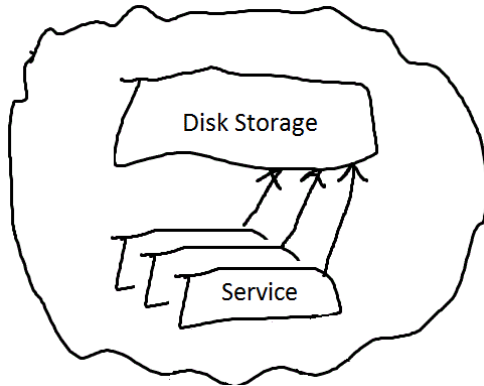


Figure 1.11 – Multiple instances of our service (that don't care what physical server they live on) talking to an abstracted logical file system, rather a physical drive

So we can see how storage is logically represented in Figure 1.1, but how does this translate into the world of Windows Azure?

As we have discovered our services will not always be continually running on the same physical machine. Our roles (web or worker) could be shutdown and moved to another machine at anytime in order to handle faults or upgrades. In the case of Web Roles the load balancer could be distributing requests to a pool of web servers meaning that an incoming request could be performed on any machine.

In order to run services in such an environment, all instances of our roles (web and worker) need access to a consistent durable scalable storage service. What this means is that all instances of our roles must have access to a shared storage mechanism.

Windows Azure provides its scalable storage service that can be accessed both inside and outside of the Microsoft data centers. On sign-up for Windows Azure you will be assigned your own storage account with a set of endpoint URI's that you can use to access the storage services for your account. The storage services are accessed via a set of REST API's secured by an authentication token. We will take a more detailed look at these APIs later on in the book.

Windows Azure Storage Services are hosted in The Fabric in the same way as our own roles are hosted. This means that it is a very scalable solution and we never need to worry about running out of capacity. We will now take a brief look at the services offered by Windows Azure Storage Services.

1.4.2 Binary File Store (Blobs)

Windows Azure provides the ability to storage binary files (blobs) within a storage area known as Blob Storage.

Within your account you create a set of containers (similar to folders) that you can store your binary files within. In Version 1 of the Blob Storage Service, containers can either be restricted to private access (i.e. you must use an authentication key to access the files held in this container), or to public access (i.e. anyone on the internet can access the file, without using an authentication key).

In figure 1.12 we return to our audio file conversion scenario. In this example we wish to convert a source recording of our podcast (Podcast01.mp3) to Windows Media Audio (Podcast01.wma). The source files will be held in Blob Storage in a private container called "Source Files", and the destination files will be held in Blob Storage in a public container called "Converted Files". Anyone in the world can access the converted files as it is held in a public container but only you can access the files in the private container as it secured by our authentication token. Both the private and public containers are held within the storage account known as "MyStorage" account.

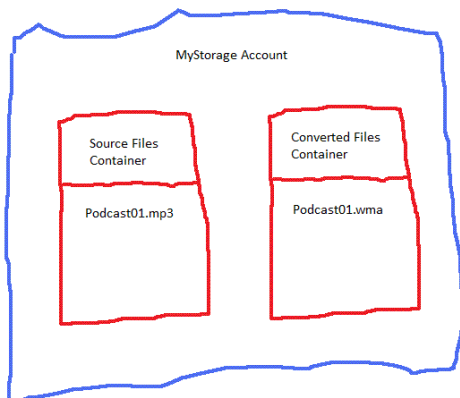


Figure 1.12 - Audio Files held in blob storage.

Blobs can be split up into more manageable chunks known as Blocks for more efficient uploading of files. In chapter 8 we will explore in much more detail the usage of Blob Storage in Windows Azure.

1.4.3 Messaging via Queues

Message Queues are the primary mechanism for communicating with Worker roles. Typically a Web Role or an external service will place a message in the queue for processing. Instances of the worker role would poll the queue for any new messages and then process the retrieved message. Once a message is read from the queue, it is no longer available to any other instances of the worker role.

In the case of our audio file conversion example, once the source podcast blob (Podcast01.mp3) is placed in the source container, a message could be placed in the queue (containing the location of the blob) ready for a worker role to perform the conversion. Once the worker role has converted the file from MP3 to WMA it can place the converted file (Podcast01.wma) in the Converted Files container.

In Chapter 16 we will look at message queues in much greater detail and expand upon these scenarios.

1.4.3 Storing data in tables

Windows Azure has the ability to store data in a highly scalable simple table storage service. This service provides the ability to store serialized entities in a big table; entities can then be partitioned across multiple servers.

This is a very simple storage mechanism that is particularly suitable for scenarios such as Session Management, User Authentication or Shopping Carts. This is not a relational database in the cloud, and if you need the power of a database (such as using server side joins) then SQL Azure is a more appropriate technology.

In Chapter 11 we will look at how to use Table Storage and in what scenarios it can be useful.

1.5 Saving Money by running in the Cloud

Ok, so far in this chapter, we've basically we've said isn't Azure shiny and cool. We've also said, wow it's so great I can take my existing app and put it in the Cloud. But what we haven't asked is why would I want to stick it in the Cloud?

Why might I want to host my applications with Microsoft rather than host them myself? What advantages do I get using this new platform? I guess some of the answers to these questions could be:

- Save lots of money
- I don't need to buy any infrastructure to run my application
- I don't need to manage the infrastructure to run my application

- My application runs on the same infrastructure that Microsoft uses to host their services, not some box under a desk
- I can scale my application out on-demand to use whatever resources it needs to meet the demands of my application
- I only pay for the resources that I use when I use it
- I am provided a framework that allows me to develop scalable software that runs within the Windows Azure Platform so my applications can run at Internet Scale
- I can focus on what I am good at, i.e. developing software
- I can watch Football and drink milkshakes without being disturbed because some idiot pulled out the server power cable so they could do the vacuuming
- Save lots of money

Just in case you think I'm repeating myself by saying **Save lots of money** twice, well it's the key point, **you save bucket loads**. So without giving away any commercials, I'm often involved in large scale systems and the infrastructure of these systems costs millions (and most of the time the servers sit idle), the cost of equivalent systems in Azure are about 10% of the cost. I'm not including the cost of running these system, you do the math (and this book seems pretty cheap now in comparison too) ☺.

With that in mind let's take a look at some of these scenarios in a little more detail and look at how the Windows Azure Platform can help us out.

1.5.1 Treating computing power as a utility service

In traditional on-premise or managed hosting solutions you either rent or own the infrastructure that your service is hosted on. This means that you are paying for future capacity that you are currently not using.

The Windows Azure Platform like other Cloud Platforms follows a model of Utility Computing. The premise of Utility Computing is the ability to offer computing power or storage in the same way as you would treat a utility service (such as gas or electricity). Your usage of the Windows Azure Platform is metered and you only pay for what you consume.

PAY AS YOU GROW

If you only have to pay for the resources that you use, it means that you can launch a scalable service without making a huge investment upfront in hardware. This is a very compelling model for start-ups. In the early days of a new venture a start-up company is surviving from investment funding and is generating very little income. The less money the company spends, the more chance it has of surviving long enough to generate sufficient income to sustain itself. If the service is as successful then the generated income will pay for the usage of the resources.

It is not unusual for technology start-ups to purchase very large and expensive hardware solutions for new ventures to cope with predicted future demand. If the service is successful then it will require the extra capacity but in the meantime the start-up is paying for resource that it is not utilizing. Utility computing offers the best of both worlds, allowing you the ability to utilize extra capacity as the service grows without making upfront investments in hardware and only paying for the resources that are actually used.

SCALE ON DEMAND

There are some situations where we have large un-sustained growth and we would like to handle the load but not pay for the unused capacity. This situation might appear in the following scenarios:

- viral marketing campaigns
- referrals by a popular website
- concert ticket sales
- etc

Let's say for example we run a Hawaiian Shirt Shop and we normally have a predictable pattern of usage. If for example Ashton Kutcher (who has 2,000,000 twitter followers) tweets that he buys his shirts from my website and he posts a link to my site to all his followers, it is likely that we can expect a surge in traffic.

If we look at the graph in Figure 1.13 it shows our website would normally receive around 1000 hits per day. After Ashton Kutcher tweeted about our website, it increased to 100,000 hits per day, the traffic dropped off after around a week and then the website had a new baseline of around 10,000 hits per day.

With Windows Azure we can dynamically scale up to handle the increased traffic load for that week (and get all the extra shirt sales), and then as the traffic decreases we can scale back down again, only paying for the resources we utilize.

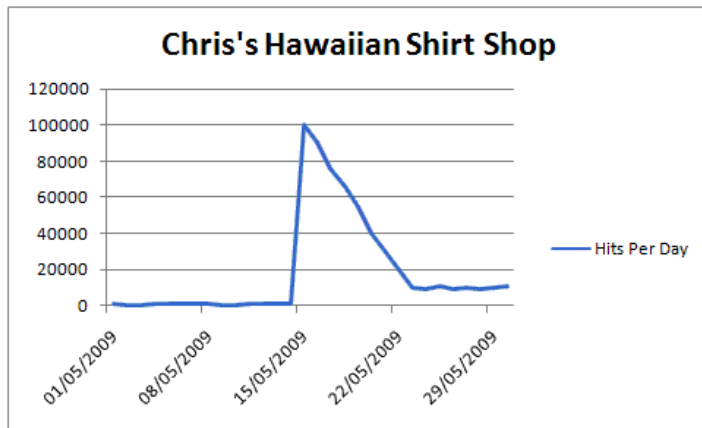


Figure 1.13 - Traffic before, during and after Ashton Kutcher plugged your site

TODO: Show how to scale

VARIED USAGE PATTERNS

It is not only possible to be able to scale up and down for predictable or unpredictable bursts of growth, it is possible to dynamically scale your service based on normal varied usage patterns.

If we return to the Hawaiian Shirt Shop example, after the Ashton Kutcher hype has died down a little, our website leveled off at around 10,000 hits per day. In Figure 1.14, we look at how this traffic is broken down over the course of a day. We can see that most of the time we have very little traffic on the site apart from at lunch time, and in the evening. It seems most people don't buy Hawaiian shirts when they are at work.

Since it only takes a few minutes to provision a new web server in Windows Azure, it is possible for us to dynamically scale our website as our usage patterns dictate. So in the case of our Hawaiian Shirt Shop, we may decide to run one instance of our Website during the day, but in the evening to run 3 instances to deal with the increased traffic.

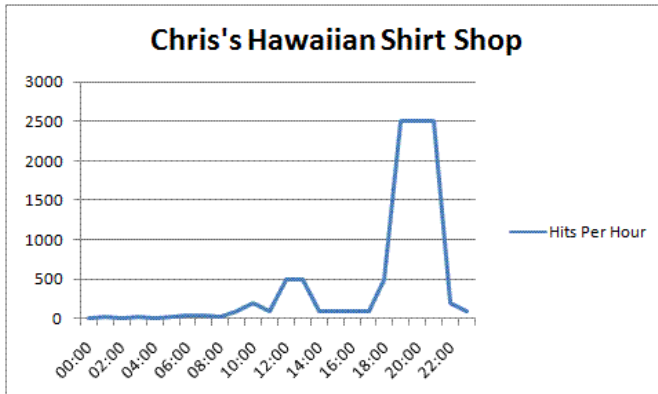


Figure 1.14 - Distribution of website traffic over a single day

This sort of scenario is a perfect example of when Cloud Computing / Windows Azure are a perfect fit for your business. If you need to scale beyond a single web server for certain periods of the day as traffic is too high then Windows Azure is a cost effective choice as it allows you to scale back down when traffic dies down. Cloud Computing solutions are really the only offerings that give you this elastic scalability. Other solutions typically mean that you have to over provision your hardware to cope with the peak periods but have that hardware underused at off-peak times.

Enough Capacity

This analogy of Utility Computing can be extended further in regards to available capacity. It is fair to say that most people do not have an idea of the available spare electricity capacity of the network supplying our home but most of us would be happy that if we plug in extra television to our home that the required electricity will be supplied. This analogy holds true in Windows Azure, we have no idea how much spare capacity the Microsoft data centers have but we do know there is enough. If we require an extra instance of our Website / Web Service / Backend Service to be hosted then this will be provisioned for us (within minutes). If the data that our service stores is much larger than we originally anticipated due to the level of growth, then more disk space will be allocated to us. We never have to worry about running out of disk space, or running out of computing power, we only have to worry about running out of money to pay for the services.

As you may have already gathered growth is very difficult to model effectively, so knowing there is always enough capacity to grow allows us to concentrate on providing our service rather than worrying about capacity planning, provisioning of new servers and all their associated tasks.

So far we've discussed the cost savings we can make by scaling our application up and down. Let's now look at how we can save money in the maintenance costs.

1.5.2 Simplified Data Center Management

In this section we will look at the effort involved in operationally maintaining your application is reduced within a Windows Azure environment.

BUILT-IN FAULT TOLERANCE

In Windows Azure, if the physical hardware that our service instance resides on fails, then our failed instance will be spun-up on another machine. The maintenance of hardware purely resides in Microsoft's domain and we do not have to worry about it.

Although fault tolerance is Microsoft's responsibility, it is worth considering how the distribution of the instances throughout the data center is performed with fault tolerance in mind.

Where we have more than two instances of our application spun up in Windows Azure, each instance of our web role will not live on the same physical server as another instance. This ensures that if the hardware for the instance dies then the other instance would be able to continue to perform incoming requests. Not only will it live on a different physical server, it will live on a different rack (in case the server rack fails). The server rack where the second instance resides will even be connected to a different network and power grid from the first rack. This level of fault tolerance ensures that if there is a failure on the physical server, the server rack, the network or in the electricity supply, our service will continue to run and be able to service requests.

On the installation of our application, Windows Azure decides what servers to place our instances on with the intention of providing the maximum levels of fault tolerance. Since all data center assets are tracked and mapped, the placement of applications on each physical server are algorithmically chosen to match the fault tolerance criteria. These considerations are pretty complex even with 2 instances of an application but Windows Azure will maximize fault tolerance even if there are hundreds of instances.

Todo: DIAGRAM

Big Scary Monsters Squishing Data Centers

Although fault tolerance is maintained within a physical data center, if a big scary monster came along and squished the data center then it's not quite clear what would happen (hopefully it will be by the time this prints). Your data is safe, it will be replicated to another data center, just not sure when the other data center comes up.

The good news the last time I was in San Antonio, I didn't notice any big scary monsters however I can't be sure about Chicago. Although it's not clear how cross data center fault tolerance is currently provided and how the switchover occurs. IF this is something you really need you could manually configure it (if monsters worry you that much). **[TODO: Diagram of a rack with 2 instances on different network etc]**

SERVER SOFTWARE MAINTENANCE BEGONE!

One of the key differences between Windows Azure hosted applications and regular on premise solutions or other cloud solutions, is that Windows Azure abstracts away everything about the infrastructure including the underlying operating system leaving us to focus on our application.

Whether you are running an entire data center, or hosting a website on a dedicated server at a hosting company, maintenance of the operating system is usually your responsibility. Maintenance tasks can include Anti-Virus, Windows Updates, applying service packs and locking down of security. If you are running your own dedicated machine on your own premises rather than it being hosted for you, then you would even be responsible for performing backups.

In Windows Azure, since you are purely focused on your application, the tasks associated with maintaining the server are the abstracted away from you and are purely the responsibility of Microsoft. This level of abstraction greatly simplifies and reduces the cost of running a service.

A final cost consideration is that if you have a service hosted in Windows Azure you do not have to worrying about the licensing costs for the underlying operating system. You gain all the benefits of running under the latest version of Windows without paying for the costs of maintaining that software. Although the underlying software is abstracted away from your service, the base underlying operating system of your service is Windows Server 2008. If you are running multiple servers the costs of licensing usually runs into thousands of dollars.

DESIGNING FOR DISTRIBUTION

Although we do not have to worry about hardware or software maintenance from an operational or cost perspective, we do have to worry about it from a software design perspective. Our services will not always be running on the same machine and they could be failed over to another machine at any time, this could be due to hardware failures, software maintenance cycles or load distribution. This means that we must design our software to be able to handle these failures. This may mean automatically retrying a failed operation on exception, or reloading any local caches on restart of a service. We will delve further into these issues later on in the book.

1.6 Windows Azure Platform (previously Azure Services Platform)

As we said earlier, the major difference between what is Windows Azure and what is Windows Azure platform is necessity. In this section, we'll give a brief overview of those services offered in the Windows Azure Platform (beyond Windows Azure itself), which are:

- Relational Database in the Cloud (SQL Azure)
- Enterprise Services in the Cloud (Windows Azure platform AppFabric)

In this book we do not cover every last aspect of every service that is offered across the Windows Azure Platform as each component could probably justify its own dedicated book (I'm really not pitching my next book proposal here ☺). We will however try and give you an understanding of what is offered, when it is useful, and how the more common scenarios can be utilized.

Let's get started by giving a brief overview of the service that you are most likely to use, SQL Azure.

1.6.1 Relational Databases in the Cloud (SQL Azure)

Although Windows Azure does offer support for storing data in tables, this is a very basic storage capability that is only suited for certain core scenarios.

If you need to create more advanced databases, migrate existing SQL databases to Azure or really just can't cope with learning another data storage technology then SQL Azure is really the best solution for you. Quite simply SQL Azure is a relational database (very similar to SQL Server Express Edition) that is hosted within the Windows Azure Platform.

A history Lesson

When SQL Azure was first announced and made available as a Community Technology Preview (CTP), it was architected very differently. The initial previews of what was known as SQL Server Data Services (SSDS), was a non-relational model that was very similar to Windows Azure Storage Services. The feedback given to the product teams made it very clear that customers wanted a relational database in the Cloud and SSDS was later retired (actually obliterated is probably a closer analogy). Prior to being renamed to SQL Azure, SQL Data Services was renamed to SQL Data Services but there was never a public CTP under this name.

WHAT IS SQL AZURE?

Version 1.0 of SQL Azure which was released at PDC09 provides the core capabilities of SQL Server in the cloud. The first release could be likened to running an instance of SQL Server Express edition on a shared host, with some changes to security so you can't mess with other databases on the same server.

Communication with SQL Azure is via the Tabular Data Stream protocol (TDS) which is the same protocol that is used for the on-premise editions of SQL Server. This means that you can connect SQL Management Studio directly to your database hosted in the cloud, as if it was hosted locally.

WINDOWS INTEGRATED SECURITY IS NOT SUPPORTED AT PRESENT

In the first release of SQL Azure, security is restricted to SQL Server User Accounts and Windows Integrated Security is not supported. I suspect that some sort of support beyond SQL Security will come at a later date

Since you can connect to SQL Azure with a regular connection string, this means any existing data access layers will continue to work as normal. Figure 1.15 shows communication between SQL Azure and applications that are hosted both inside Windows Azure and outside of Windows Azure.

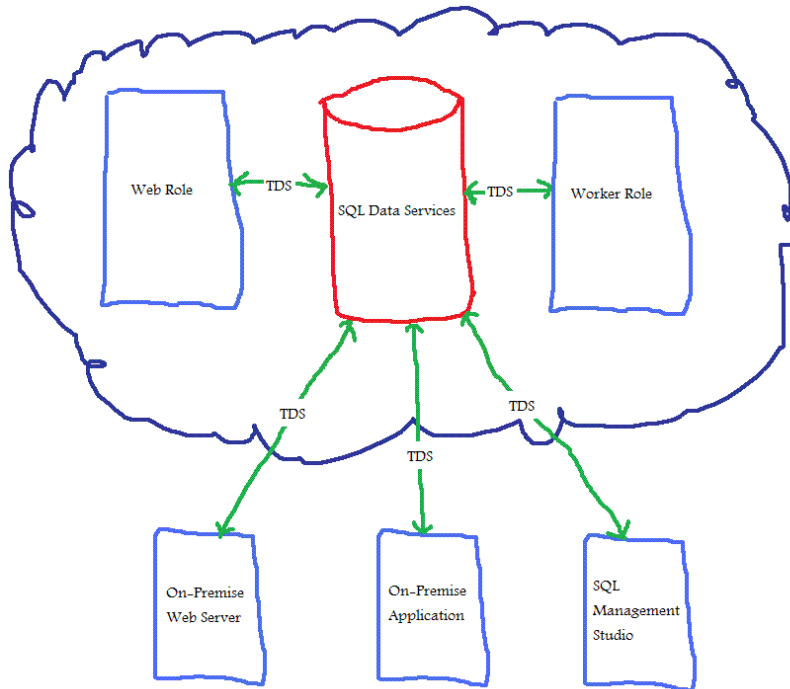


Figure 1.15 – On premise, Azure hosted applications and SQL Management Studio all communicating with SQL Azure via TDS

TODO: Need to a new version of this diagram

If your application works today against SQL Express edition and does not make use of some of the more advanced features of SQL Server (see the SQL Azure chapter) then your application should work in the Cloud with little or no modification.

ON PREMISE LATENCY

Although you can have on-premise applications talking to SQL Azure, latency may make this less attractive. The closer your application is to the database, the faster it will go. You can reduce the effects by making your application less chatty.

HOW SCALABLE IS SQL AZURE?

In version 1.0 of SQL Azure there is no built in support for data partitioning (i.e. the ability to split your data across multiple servers). The initial release is targeted for databases that are sized up to around 10Gb, larger database are not suitable for SQL Azure in this initial release but support will available in future releases. If you need to perform partitioning this would need to be implemented in the application layer.

WHAT WE WILL COVER LATER ON?

Since SQL Azure is a cut down version of SQL Server, we will not focus on standard SQL Server functionality but instead concentrate on the Azure specifics. We will look at the current architecture and how this is likely to change in the future, and dive deeper into what design patterns and scenarios are suitable today for building Cloud Applications.

1.6.2 Enterprise Services in the Cloud (Windows Azure platform AppFabric)

Windows Azure platform AppFabric (formerly .NET Services) is a set of services that are more orientated towards enterprise applications and is comprised of the following components:

- AppFabric Access Control

- AppFabric Service Bus

AppFabric is a very large set of technologies that really requires its own book to cover it in depth. In this book we will give an overview of the technologies, how to get started using it, and give a couple of key useful scenarios where you can use the technology.

Windows AppFabric Server

As well as the Windows Azure platform AppFabric product, there is also an on-premise product known as Windows Server AppFabric. This is a completely different product set which currently includes AppFabric Caching (formerly Velocity) & AppFabric Service Workflow and Management (formerly Dublin).

Although these services are not currently present in the Windows Azure Platform (PDC09), you can expect them to appear at some point. Who knows depending on when you are reading this book, they could be here right now. Alternatively, you could be reading this in the far future (relative to PDC09) and there is no need for this technology as we are all plugged into the Matrix.

ACCESS CONTROL

User Authentication security and management is a fairly standard requirement for any service used within an enterprise organization. Enterprises require the ability for their employees, customers and vendors to be able to access all services in the organization with a single login; typically the authentication process occurs via the Windows login. Once you are authenticated, you will typically be issued a token which will be automatically passed to other services in the Enterprise as they are accessed. The automatic passing and authentication of this token means you do not have to continually login, each time you access a service.

Services in the Enterprise therefore do not implement their own individual authentication / user management systems but hook directly into the organizations identity management service (such as Active Directory). This "Single Sign-On" process provides many benefits to the company (centralized and simplified user management and security), and is a much more integrated and less frustrating user experience.

Traditionally Identity Management and Access Control have been restricted to the enterprise space. However, with the advent of Web 2.0 social platforms such as Live Services and Facebook, this level of integration is now creeping into everyday websites. Web users are now more concerned about data privacy and are reluctant to cheaply give away personal information to third party websites, they don't want a long list of usernames and passwords for various sites and want a much richer social experience on the web. For example it is increasingly common for people to want to be able to tell their friends on Facebook about their latest purchase. Between Facebook, Live Services, OpenID (and its differing implementations), and all the various Enterprise Identity Management Systems, access control has now become a complex task.

AppFabric Access Control Service abstracts away the nuances of the various third party providers utilizing a simple rules based authentication service that can manage authentication across multiple providers for users with multiple credentials. Later on in this book we will look into the Access Control service in a little more detail, and use it in a couple of key scenarios.

SERVICE BUS

The service bus is a pretty cool piece of technology allows you to message with applications that are not necessarily running within the Azure data centers. Therefore if you have a custom proprietary service that you need to continue to host but want to use Azure for all other aspects of your service offering, then the Service Bus is a good way of integrating with those services.

The Service Bus is effectively an Enterprise Message Bus that is hosted in the Cloud. We will explore in more detail what this means and a couple of key scenarios where you could use this technology, in chapter 17.

1.7 Summary

Hopefully we have started to tame that big scary monster and started to make it look like a cute little teddy bear. In one chapter we have learned about Cloud Computing, Windows Azure and the Windows Azure Platform.

In this chapter we have seen how applications that run within Windows Azure can be easily scaled to support the future needs of your application whilst only paying for your current needs. We've also seen how developing Windows Azure builds upon your existing skills by developing your first Windows Azure Web Application.

In future chapters we will start to peel away the layers of this new platform and look at how we can use the platform effectively.