WINDOWS AZURE EXECUTION MODELS

Windows Azure provides three different execution models for running applications: Virtual Machines, Web Sites, and Cloud Services. Each one provides a different set of services, and so which one you choose depends on exactly what you're trying to do. This article looks at all three, describing each technology and giving examples of when you'd use it.

Contents

Virtual Machines

Windows Azure Virtual Machines lets developers, IT operations people, and others create and use virtual machines in the cloud. Providing what's known as *Infrastructure as a Service (IaaS)*, this technology can be used in variety of ways. Figure 1 shows its basic components.

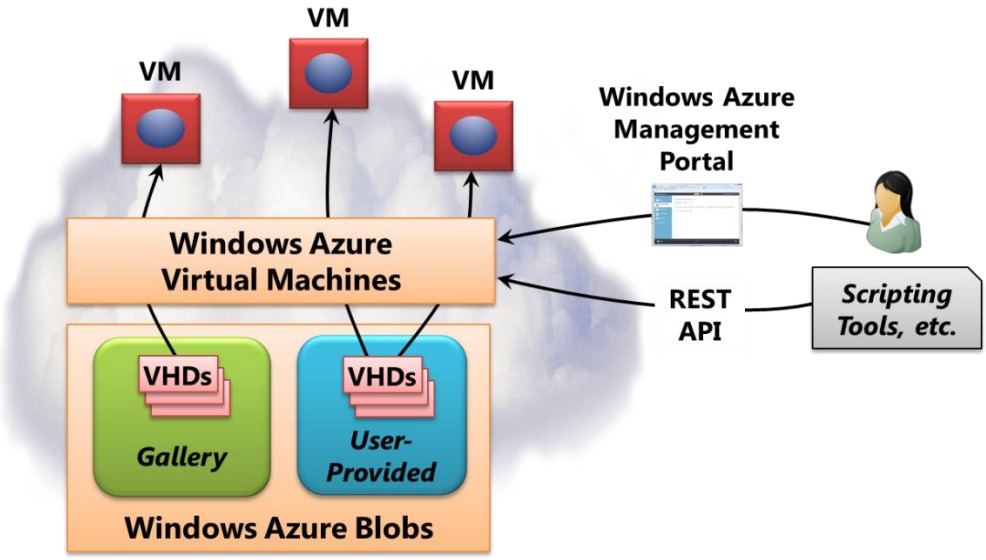

Figure 1: Windows Azure Virtual Machines provides Infrastructure as a Service.

As the figure shows, you can create virtual machines using either the Windows Azure Management Portal or the REST-based Windows Azure Service Management API. The Management Portal can be accessed from any popular browser, including Internet Explorer, Mozilla, and Chrome. For the REST API, Microsoft provides client scripting tools for Windows, Linux, and Macintosh systems. Other software is also free to use this interface.

However you access the platform, creating a new VM requires choosing a virtual hard disk (VHD) for the VM's image. These VHDs are stored in Windows Azure blobs, and you have two choices: upload your own or use VHDs provided by Microsoft and its partners in the Windows Azure Virtual Machines *gallery*. The VHDs in the gallery include Windows Server 2008 R2, Windows Server 2008 R2 with SQL Server, and Windows Server 2012. The gallery also holds Linux images, including Suse, Ubuntu, and CentOS, provided by Microsoft partners.

Along with a VHD, you specify the size of your new VM. The choices are:

- Extra Small, with a shared core and 768 megabytes of memory.
- Small, with 1 core and 1.75 gigabytes of memory.
- Medium, with 2 cores and 3.5 gigabytes of memory.
- Large, with 4 cores and 7 gigabytes of memory.
- Extra Large, with 8 cores and 14 gigabytes of memory.

Finally, you choose which Windows Azure datacenter your new VM should run in, whether in the US, Europe, or Asia.

Once a VM is running, you pay for each hour it runs, and you stop paying when you remove that VM. The amount you pay doesn't depend on how much your VM is used—it's based solely on wall-clock time. A VM that sits idle for an hour costs the same as one that's heavily loaded.

Each running VM has an associated *OS disk*, kept in a blob. When you create a VM using a gallery VHD, that VHD is copied to your VM's OS disk. Any changes you make to the operating system of your running VM are stored here. For example, if you install an application that modifies the Windows registry, that change will be stored in your VM's OS disk. The next time you create a VM from that OS disk, the VM continues running in the same state you left it in. For VHDs stored in the gallery, Microsoft applies updates when needed, such as operating system patches. For the VHDs in your own OS disks, however, you're responsible for applying these updates (although Windows Update is turned on by default).

Virtual Machines also monitors the hardware hosting each VM you create. If a physical server running a VM fails, the platform notices this and starts the same VM on another machine. And assuming you have the right licensing, you can copy a changed VHD out of your OS disk, then run it someplace else, such as in your own on-premises datacenter or at another cloud provider.

Along with its OS disk, your VM has one or more *data disks*. Even though each of these looks to your VM like a mounted disk drive, the contents of each one is in fact stored in a blob. Every write made to a data disk is stored persistently in the underlying blob. As with all blobs, Windows Azure replicates these both within a single datacenter and across datacenters to guard against failures.

Running VMs can be managed using the Management Portal, PowerShell and other scripting tools, or directly through the REST API. (In fact, whatever you can do through the Management Portal can be done programmatically through this API.) Microsoft partners such as RightScale and ScaleXtreme also provide management services that rely on the REST API.

Windows Azure Virtual Machines can be used in a variety of ways. The primary scenarios that Microsoft targets include these:

- *VMs for development and test*. Development groups commonly need VMs with specific configurations for creating applications. Windows Azure Virtual Machines provides a straightforward and economical way to create these VMs, use them, then remove them when they're no longer needed.

- *Running applications in the cloud*. For some applications, running on the public cloud makes economic sense. Think about an application with large spikes in demand, for example. It's always possible to buy enough machines for your own datacenter to run this application, but most of those machines are likely to sit unused much of the time. Running this application on Windows Azure lets you pay for extra VMs only when you need them, shutting them down when a demand spike has ended. Or suppose you're a start-up that needs on-demand computing resources quickly and with no commitment. Once again, Windows Azure can be the right choice.

- *Extending your own datacenter into the public cloud*. With Windows Azure Virtual Network, your organization can create a virtual network (VNET) that makes a group of Windows Azure VMs appear to be part of your own on-premises network. This allows running applications such as SharePoint and others on Windows Azure, an approach that might be easier to deploy and/or less expensive than running them in your own datacenter.
- *Disaster recovery*. Rather than paying continuously for a backup datacenter that's rarely used, IaaS-based disaster recovery lets you pay for the computing resources you need only when you really need them. For example, if your primary datacenter goes down, you can create VMs running on Windows Azure to run essential applications, then shut them down when they're no longer needed.

These aren't the only possibilities, but they're good examples of how you might use Windows Azure Virtual Machines.

Grouping VMs: Cloud Services

When you create a new VM with Windows Azure Virtual Machines, you can choose to run it standalone or make it part of a group of VMs running together in a *cloud service*. (Despite the similar names, don't confuse this concept with Cloud Services, the name of Windows Azure's PaaS technology; the two aren't the same.) Each standalone VM is assigned its own public IP address, while all of the VMs in the same cloud service are accessed through a single public IP address. Figure 2 shows how this looks.

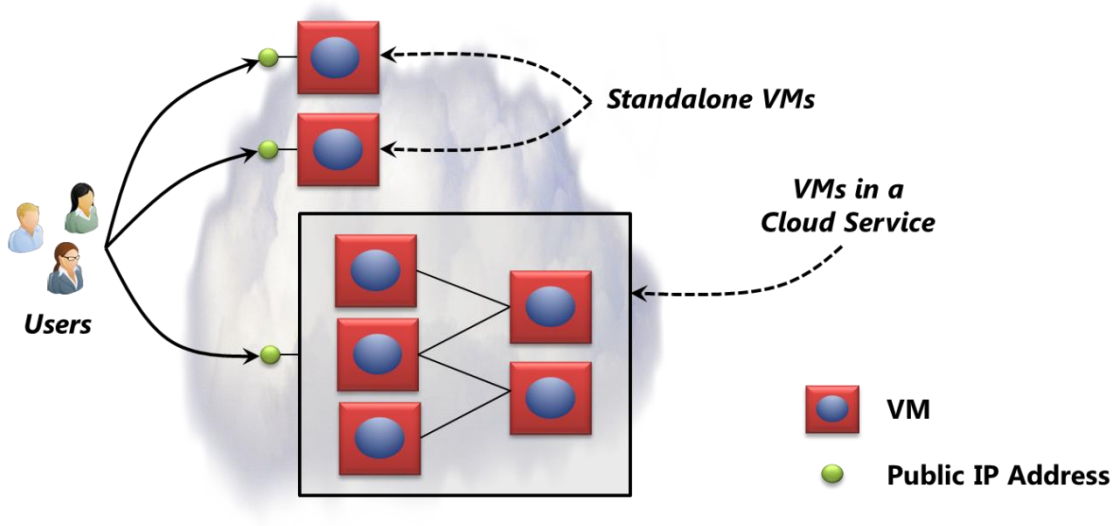

Figure 2: Each standalone VM has its own public IP address, while VMs grouped into a cloud service are exposed via a single public IP address.

For example, if you were creating a virtual machine to create and test a simple application, you'd probably use a standalone VM. If you're creating a multi-tier application, though, with a web front end, a database, and maybe even a middle tier, you'd most likely connect multiple VMs into a cloud service, as Figure 2 suggests.

Grouping VMs into a cloud service lets you use other options as well. Windows Azure provides load balancing for VMs in the same cloud service, spreading user requests across them. VMs connected in this way can also communicate directly with one another over the local network within a Windows Azure datacenter.

VMs in the same cloud service can also be grouped into one or more *availability sets*. To understand why this is important, think about a web application that runs multiple front-end VMs. If all of these VMs are deployed on the same physical machine or even in the same rack of machines, a single hardware failure can make them all inaccessible. If these VMs are grouped into an availability set, however, Windows Azure will deploy them across the datacenter so that none of them share a single point of failure.

Scenario: Running an Application with SQL Server

To get a better sense of how Windows Azure Virtual Machines works, it's useful to look at a couple of scenarios in a little more detail. Suppose, for example, that you want to create a reliable and scalable web application running on Windows Azure. One way to do this is to run the application's logic in one or more Windows Azure VMs, then use SQL Server for data management. Figure 3 shows how this looks.

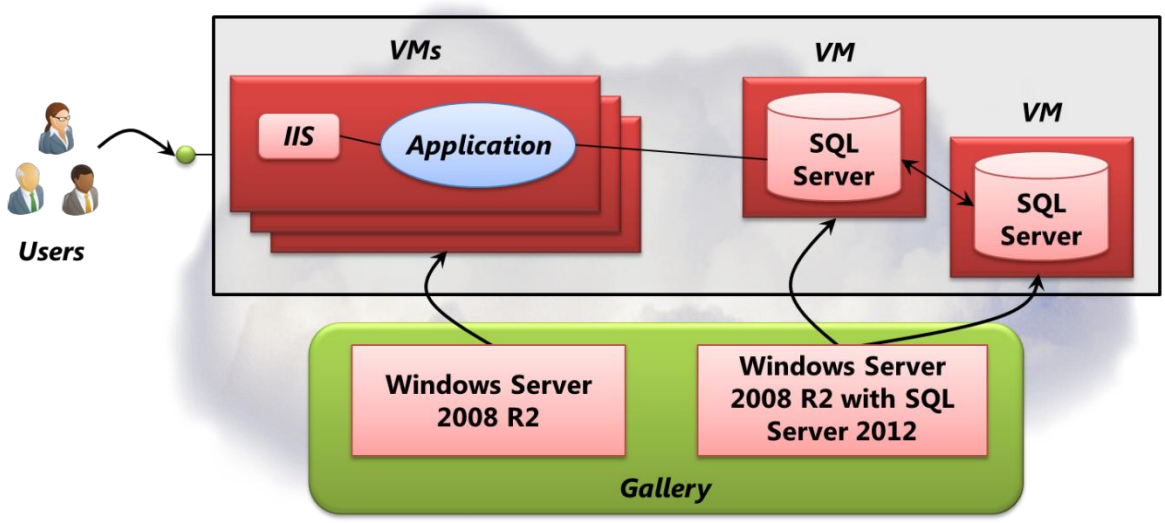

Figure 3: An application running in Windows Azure Virtual Machines can use SQL Server for storage.

In this example, both kinds of VMs are created from standard VHDs in the gallery. The application's logic runs on Windows Server 2008 R2, so the developer creates three VMs from this VHD, then installs his application in each one. Since all of these VMs are in the same cloud service, he can configure hardware load balancing to spread requests across them. The developer also creates two VMs from the gallery's VHD containing SQL Server 2012. Once they're running, he configures SQL Server in each instance to use database mirroring with automatic failover. This isn't required—the application could use just a single SQL Server instance—but taking this approach improves reliability.

Scenario: Running a SharePoint Farm

Suppose an organization wishes to create a SharePoint farm but doesn't want to run the farm in its own datacenter. Maybe the on-premises datacenter is short of resources, or perhaps the business unit creating the farm doesn't want to deal with its internal IT group. In cases like these, running SharePoint on Windows Azure Virtual Machines can make sense. Figure 4 shows how this looks.

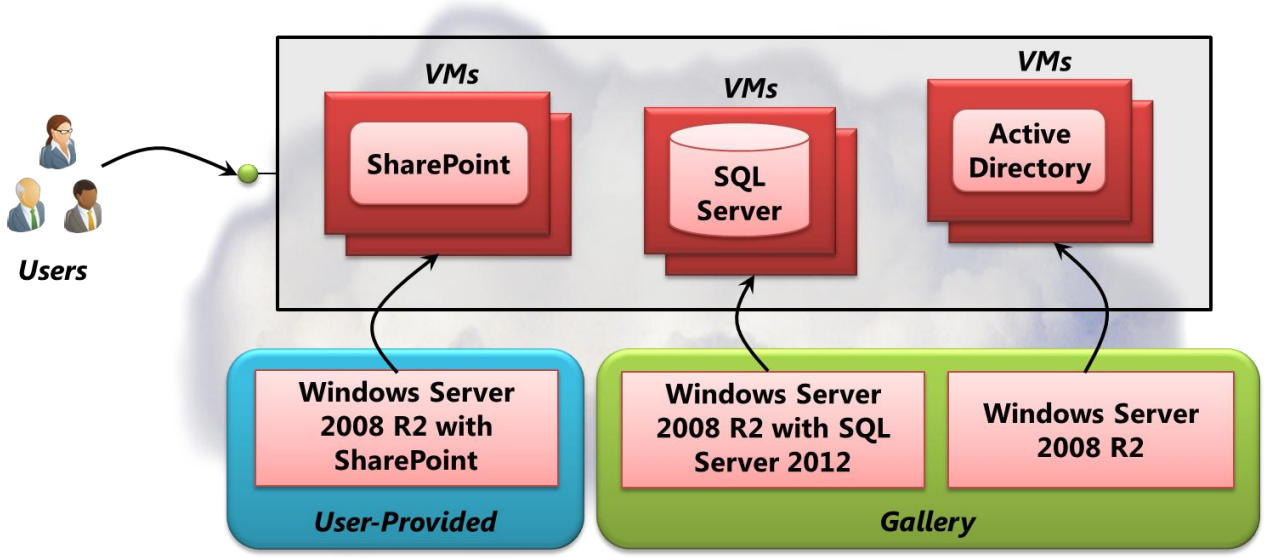

Figure 4: Windows Azure Virtual Machines allows running a SharePoint farm in the cloud.

A SharePoint farm has several components, each running in a Windows Azure VM created from a different VHD. What's needed is the following:

- Microsoft SharePoint. None of the images provided in the gallery include SharePoint, so the organization provides its own VHD for this.
- Microsoft SQL Server. SharePoint depends on SQL Server, so the organization creates VMs running SQL Server 2012 from a standard gallery VHD.
- Windows Server Active Directory. SharePoint also requires Active Directory, so the organization creates domain controllers in the cloud using a Windows Server image from the gallery. This isn't

strictly required—it's also possible to use on-premises domain controllers—but SharePoint interacts frequently with Active Directory, and so the approach shown here will have better performance.

Although it's not shown in the figure, this SharePoint farm is probably connected to an on-premises network using Windows Azure Virtual Network. This lets the VMs—and the applications they contain—appear to be local to the people using that network.

As these examples show, you can use Windows Azure Virtual Machines to create and run new applications in the cloud, to run existing applications, or in other ways. Whatever option you choose, the technology's goal is the same: providing a general-purpose foundation for public cloud computing.

Web Sites

People use web technologies in many different ways. A student might create a website for her basketball team using nothing but static HTML files. An IT consultant might use a popular open source application to set up a content management system for a small business. A web design agency might work with a team of developers to build a custom web application capable of handling thousands of users.

All of these things could be accomplished using Windows Azure Virtual Machines. But creating and managing raw VMs requires some skill and takes effort. If all you want is a web site or web application, there's an easier (and probably cheaper) solution: the approach commonly known as web hosting. As Figure 5 shows, Windows Azure provides this with Web Sites.

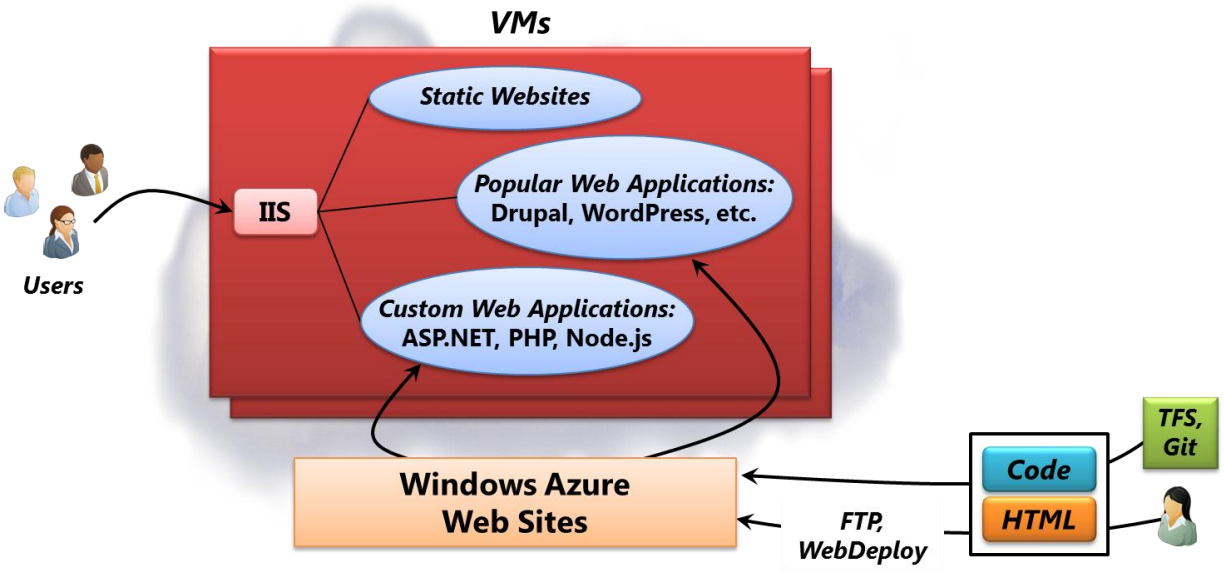

Figure 5: Windows Azure Web Sites supports static web sites, popular web applications, and custom web applications built with various technologies.

Windows Azure Web Sites runs Windows Server and Internet Information Services (IIS) in a virtual machine. As the figure shows, a single VM typically contains multiple web sites created by multiple users, although it's also possible for a site to run in its own reserved VM. Whether a VM is shared or not, you can think of Web Sites as supporting three main scenarios: building static web sites, deploying popular open source applications, and creating web applications. It's worth looking at each one separately.

Building a static web site requires nothing more than copying files containing HTML and other web content into the appropriate directories, then letting IIS serve those files to users. Windows Azure Web Sites looks much like a standard IIS environment, so doing this is straightforward.

Web Sites also has built-in support for a number of popular open source applications, including Drupal, WordPress, and Joomla. A user can choose the application she wants from a menu, then have it automatically installed and made available for her to use. There's no code to write—it's just configuration. And because many of these applications use MySQL, they rely on a MySQL service provided for Windows Azure by ClearDB, a Microsoft partner.

Developers can also create full-fledged web applications with Web Sites. The technology supports creating applications using ASP.NET, PHP, and Node.js, and once again, the environment looks like on-premises IIS. Applications can use sticky sessions, for example, and existing web applications can be moved to this cloud platform with no changes.

Applications built on Web Sites are free to use other aspects of Windows Azure, such as Service Bus, SQL Database, and Blob Storage. You can also run multiple copies of an application in different VMs,

with Web Sites automatically load balancing requests across them. And because each copy typically begins running in a VM that already exists, starting a new application instance happens very quickly; it's significantly faster than waiting for a new VM to be created.

As Figure 5 shows, you can load code and other web content into Web Sites in several ways. A simple web site might use FTP, for example, or Microsoft's WebDeploy technology. Web Sites also supports loading code from source control systems, including Git, Team Foundation Server, and the cloud-based Team Foundation Service.

Web hosting is a useful approach. Windows Azure Web Sites aims at providing this to a broad audience, whether they want to build static web sites, use popular open source web applications, or create custom web software.

Cloud Services

Windows Azure Virtual Machines provides IaaS, while Windows Azure Web Sites provides web hosting. The third compute option, Cloud Services, provides *Platform as a Service (PaaS)*. This technology is designed to support applications that are scalable, reliable, and cheap to operate. It's also meant to free developers from worrying about managing the platform they're using, letting them focus entirely on their applications. Figure 6 illustrates the idea.

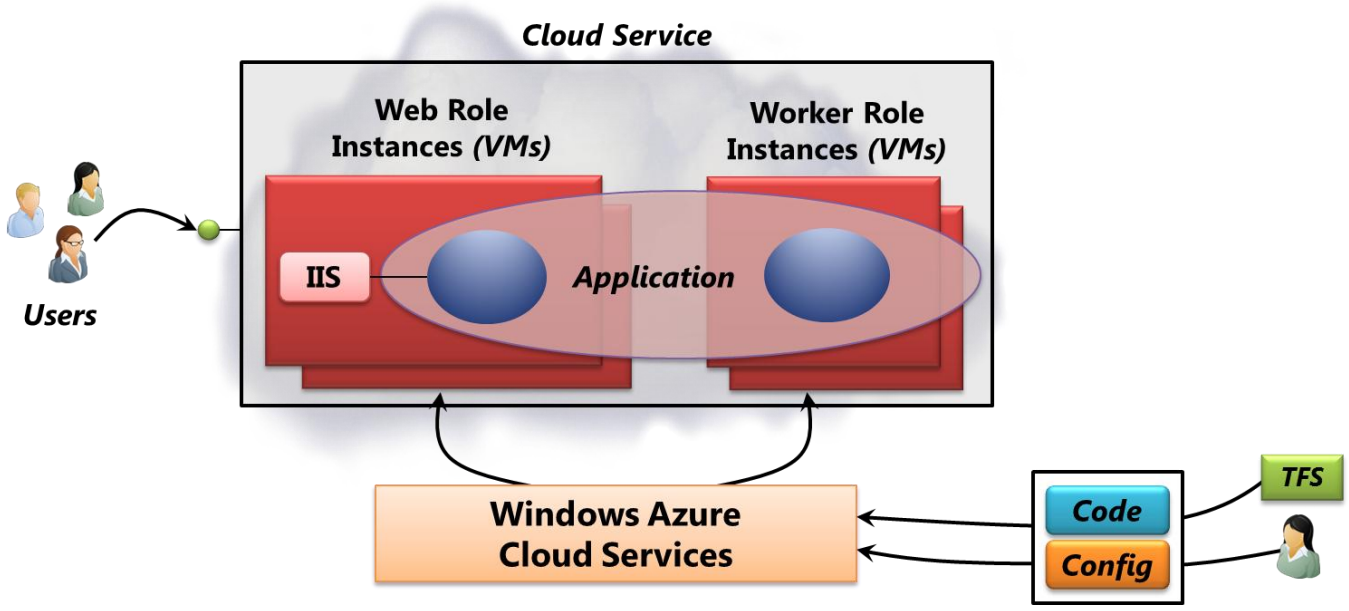

Figure 6: Windows Azure Cloud Services provides Platform as a Service.

Like the other Windows Azure compute options, Cloud Services relies on VMs. The technology provides two slightly different VM options: instances of *web roles* run a variant of Windows Server with IIS, while instances of *worker roles* run the same Windows Server variant without IIS. A Cloud Services application relies on some combination of these two options.

For example, a simple application might use just a web role, while a more complex application might use a web role to handle incoming requests from users, then pass the work those requests create to a worker role for processing. (This communication could use Service Bus or Windows Azure Queues.)

As the figure suggests, all of the VMs in a single application run in the same cloud service, as described earlier with Windows Azure Virtual Machines. Because of this, users access the application through a single public IP address, with requests automatically load balanced across the application's VMs. And as with cloud services created using Windows Azure Virtual Machines, the platform will deploy the VMs in a Cloud Services application in a way that avoids a single point of hardware failure.

Even though applications run in virtual machines, it's important to understand that Cloud Services provides PaaS, not IaaS. Here's one way to think about it: With IaaS, such as Windows Azure Virtual Machines, you first create and configure the environment your application will run in, then deploy your application into this environment. You're responsible for managing much of this world, doing things such as deploying new patched versions of the operating system in each VM. In PaaS, by contrast, it's as if the environment already exists. All you have to do is deploy your application. Management of the platform it runs on, including deploying new versions of the operating system, is handled for you.

With Cloud Services, you don't create virtual machines. Instead, you provide a configuration file that tells Windows Azure how many of each you'd like, such as three web role instances and two worker role instances, and the platform creates them for you. You still choose what size those VMs should be—the options are the same as with Windows Azure VMs—but you don't explicitly create them yourself. If your application needs to handle a greater load, you can ask for more VMs, and Windows Azure will create those instances. If the load decreases, you can shut those instances down and stop paying for them.

A Cloud Services application is typically made available to users via a two-step process. A developer first uploads the application to the platform's staging area. When the developer is ready to make the application live, she uses the Windows Azure Management Portal to request that it be put into production. This switch between staging and production can be done with no downtime, which lets a running application be upgraded to a new version without disturbing its users.

Cloud Services also provides monitoring. Like Windows Azure Virtual Machines, it will detect a failed physical server and restart the VMs that were running on that server on a new machine. But Cloud Services also detects failed VMs and applications, not just hardware failures. Unlike Virtual Machines, it has an agent inside each web and worker role, and so it's able to start new VMs and application instances when failures occur.

The PaaS nature of Cl oud Services has other implications, too. One of the most important is that applications built on this technology should be written to run correctly when any web or worker role

instance fails. To achieve this, a Cloud Services application shouldn't maintain state in the file system of its own VMs. Unlike VMs created with Windows Azure Virtual Machines, writes made to Cloud Services VMs aren't persistent; there's nothing like a Virtual Machines data disk. Instead, a Cloud Services application should explicitly write all state to SQL Database, blobs, tables, or some other external storage. Building applications this way makes them easier to scale and more resistant to failure, both important goals of Cloud Services.

What Should I Use? Making a Choice

All three Windows Azure execution models let you build scalable, reliable applications in the cloud. Given this essential similarity, which one should you use? The answer depends on what you're trying to do.

Virtual Machines provides the most general solution. If you want the most control possible, or if you need generic VMs, such as for development and test, this is the best option. Virtual Machines is also the best choice for running off-the-shelf on-premises applications in the cloud, as illustrated by the SharePoint example described earlier. And because the VMs you create with this technology can look just like your on-premises VMs, it's also likely to be the best choice for disaster recovery. The trade-off, of course, is that with great power comes great responsibility—IaaS requires you to take on some administrative work.

Web Sites is the right option when you want to create a simple website. This is especially true if your site will be based on an existing application such as Joomla, WordPress, or Drupal. Web Sites is also a good choice for creating a low-administration web application, even one that must be quite scalable, or moving an existing IIS web app to the public cloud. It provides fast deployment as well—a new instance of your application can start running almost immediately, while deploying a new VM with either Virtual Machines or Cloud Services can take several minutes.

Cloud Services, which was the initial execution model provided by Windows Azure, is an explicitly PaaS approach. While the line between PaaS and web hosting is blurry, Cloud Services differs in some important ways from Web Sites, including the following:

- Unlike Web Sites, Cloud Services gives you administrative access to your application's VMs. This lets you install arbitrary software that your application needs, something that's not possible with Web Sites.
- Because Cloud Services offers both web roles and worker roles, it's a better choice than Web Sites for multi-tier applications that need separate VMs for their business logic.
- Cloud Services provides separate staging and production environments, making application updates somewhat smoother than Web Sites.

- Unlike Web Sites, you can use networking technologies such as Windows Azure Virtual Network and Windows Azure Connect to hook on-premises computers to Cloud Services applications.
- Cloud Services lets you use Remote Desktop to connect directly to an application's VMs, something that's not possible with Web Sites.

Because it's PaaS, Cloud Services also offers some advantages over Windows Azure Virtual Machines. More management tasks are done for you, for instance, such as deploying operating system updates, and so your operations costs are likely to be lower than with the IaaS approach of Windows Azure Virtual Machines.

All three Windows Azure execution models have pros and cons. Making the best choice requires understanding these, knowing what you're trying to accomplish, then choosing the one that's the best fit.

Sometimes, no single execution model is right. In situations like this, it's perfectly legal to combine options. For example, suppose you're building an application where you'd like the management benefits of Cloud Services web roles, but you also need to use standard SQL Server for compatibility or performance reasons. In this case, the best option is to combine execution models, as Figure 7 shows.

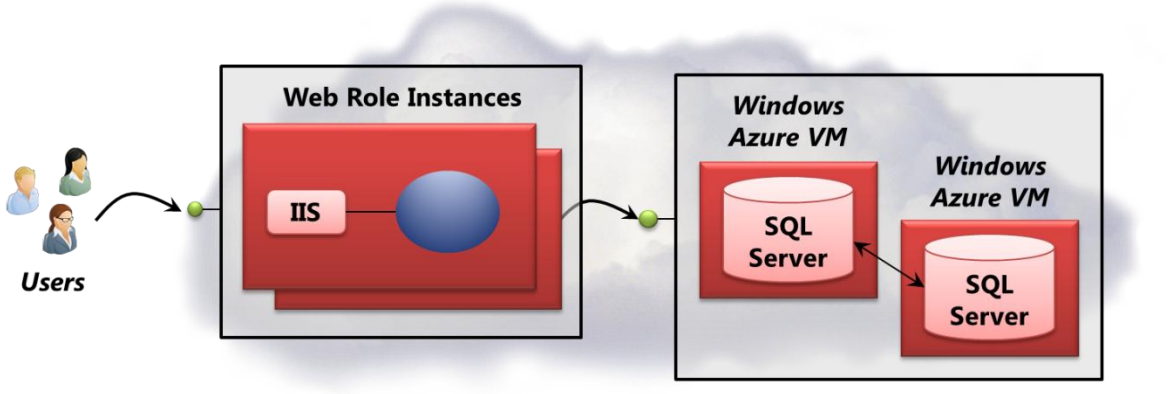

Figure 7: A single application can use multiple execution models.

As the figure illustrates, the Cloud Services VMs run in a separate cloud service from the Virtual Machines VMs. Still, the two can communicate quite efficiently, so building an app this way is sometimes the best option.

Windows Azure provides different execution models because cloud platforms need to support many different scenarios. Anybody who wants to use this platform effectively—and if you've read this far, that probably includes you—needs to understand each one.

About the Author

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technologies.