WINDOWS AZURE NETWORKING

The easiest way to connect to Windows Azure applications and data is through an ordinary Internet connection. But this simple solution isn't always the best approach. Windows Azure also provides three more technologies for connecting users to Windows Azure datacenters: Virtual Network, Connect, and Traffic Manager. This article takes a look at each of these.

Contents

Windows Azure Virtual Network

Windows Azure lets you create virtual machines (VMs) that run in Microsoft datacenters. Suppose your organization wants to use those VMs to run enterprise applications or other software that will be used by your firm's employees. Maybe you want to create a SharePoint farm in the cloud, for example, or run an inventory management application. To make life as easy as possible for your users, you'd like these applications to be accessible just as if they were running in your own datacenter.

There's a standard solution to this kind of problem: create a virtual private network (VPN). Organizations of all sizes do this today to link, say, branch office computers to the main company datacenter. This same approach can work with Windows Azure VMs, as Figure 1 shows.

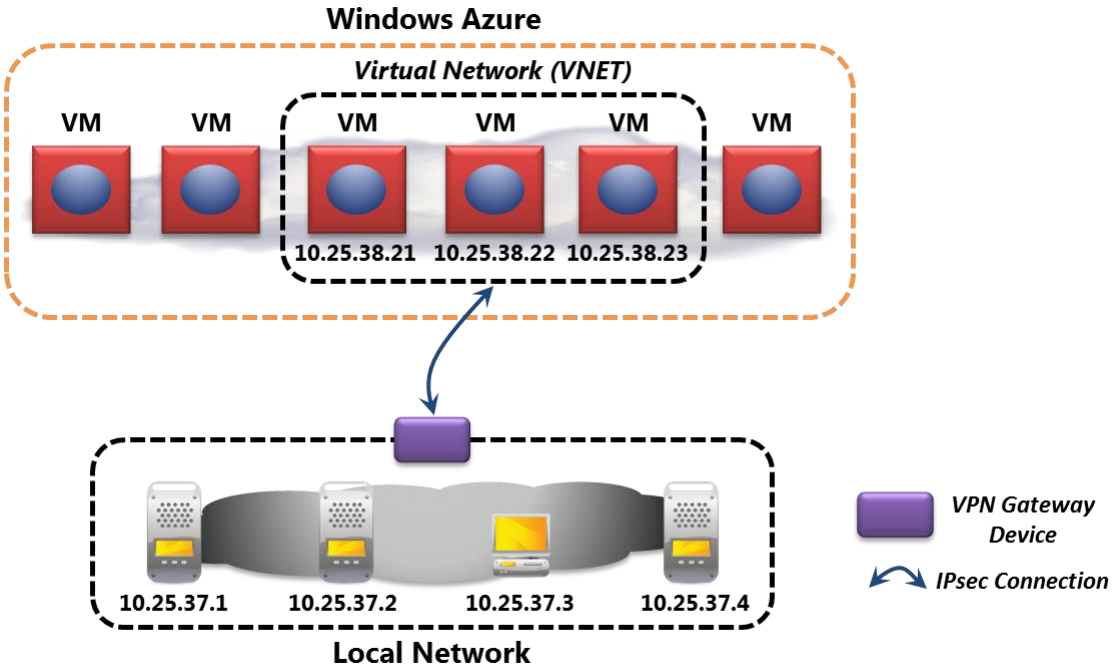

Figure 1: Windows Azure Virtual Network allows creating a virtual network in the cloud that's connected to your on-premises datacenter.

As the figure shows, Windows Azure Virtual Network lets you create a logical boundary around a group of VMs, called a *virtual network* or *VNET*, in a Windows Azure datacenter. It then lets you establish an IPsec connection between this VNET and your local network. The VMs in a VNET can be created using Windows Azure Virtual Machines, Windows Azure Cloud Services, or both. In other words, they can be VMs created using either Windows Azure's Infrastructure as a Service (IaaS) technology or its Platform as a Service (PaaS) technology.

Whatever choice you make, creating the IPsec connection requires a VPN gateway device, specialized hardware that's attached to your local network, and it also requires the services of your network administrator. Once this connection is in place, the Windows Azure VMs running in your VNET look like just another part of your organization's network.

As Figure 1 suggests, you allocate IP addresses for the Windows Azure VMs from the same IP address space used in your own network. In the scenario shown here, which uses private IP addresses, the VMs in the cloud are just another IP subnet. Software running on your local network will see these VMs as if they were local, just as they do with traditional VPNs. And it's important to note that because this connection happens at the IP level, the virtual and physical machines on both sides can be running any operating system. Windows Azure VMs running Windows Server or Linux can interact with on-premises machines running Windows, Linux, or other systems. It's also possible to use mainstream management tools, including System Center and others, to manage the cloud VMs and the applications they contain.

Using Windows Azure Virtual Network makes sense in many situations. As already mentioned, this approach lets enterprise users more easily access cloud applications. An important aspect of this ease of use is the ability to make the Windows Azure VMs part of an existing on-premises Active Directory domain to give users single sign-on to the applications they run. You can also create an Active Directory domain in the cloud if you prefer, then connect this domain to your on-premises network.

Creating a VNET in a Windows Azure datacenter effectively gives you access to a large pool of on-demand resources. You can create VMs on demand, pay for them while they're running, then remove them (and stop paying) when you no longer need them. This can be useful for scenarios that need fast access to a preconfigured machine, such as development teams building new software. Rather than wait for a local administrator to set up the resources they need, they can create these resources themselves in the public cloud.

And just as Virtual Network makes Windows Azure VMs appear local to on-premises resources, the reverse is also true: Software running in your local network now appears to be local to applications running in your Windows Azure VNET. Suppose you'd like to move an existing on-premises application to Windows Azure, for example, because you've determined that it will be less expensive to operate in the cloud. But what if the data that application uses is required by law to be stored on premises? In a situation like this, using Virtual Network lets the cloud application see an on-premises database system as if it were local—accessing it becomes straightforward. Whatever scenario you choose, the result is the same: Windows Azure becomes an extension of your own datacenter.

Windows Azure Connect

Sometimes, connecting your entire on-premises network to a group of Windows Azure VMs is the right thing to do. Windows Azure Virtual Network is designed to solve this problem. But what if you don't need a solution that's this general? Suppose instead that all you'd like to do is connect a single Windows Azure application—or even a single VM—to a specific group of computers on your local network. Addressing this problem is the goal of Windows Azure Connect, as Figure 2 shows.

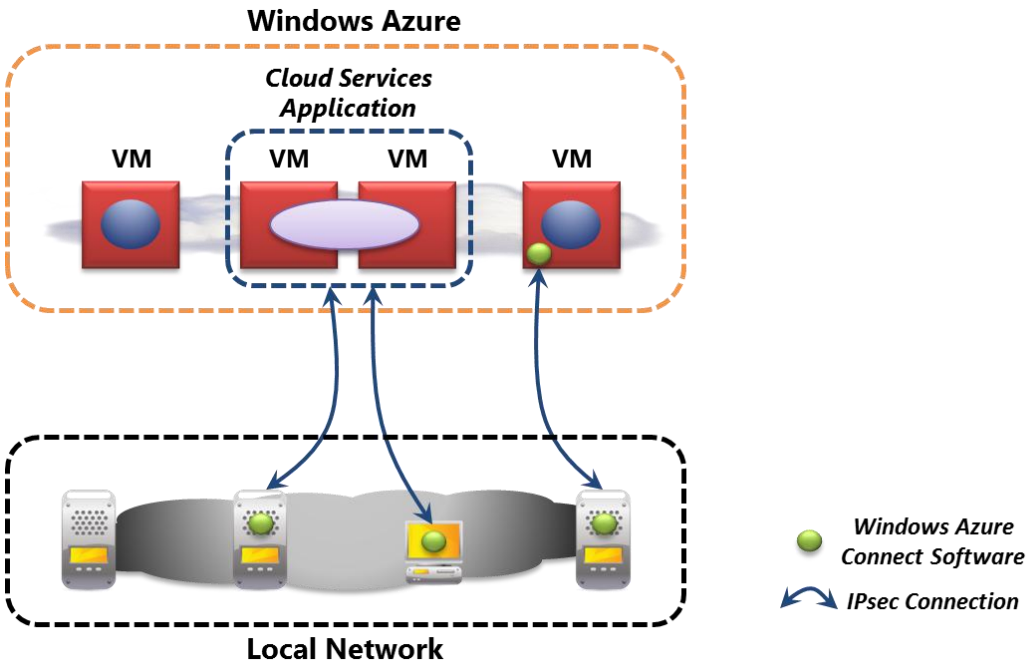

Figure 2: Windows Azure Connect links one or more VMs in Windows Azure with a group of on-premises machines running Windows.

Unlike Virtual Network, Connect doesn't require using a VPN gateway device, nor does it require the services (or approval) of a network administrator. Instead, anybody with administrative access to a Windows machine in the local network can install the required Windows Azure Connect software on that machine. Once this is done, the software can create an IPsec link with designated Windows Azure VMs.

As the figure shows, Connect doesn't link two networks together; the Windows Azure VMs retain whatever IP addresses they already have. Instead, it creates direct IPsec connections between specific on-premises Windows computers and specific Windows Azure VMs. (To work with existing firewall settings, Connect actually sends IPsec on top of an SSL connection.) For Cloud Services applications, you can choose one or more roles to connect to, and Windows Azure will make it possible to communicate with each instance in those roles. For VMs created using Windows Azure Virtual Machines, you can install the same Windows Azure Connect software used for on-premises computers.

Windows Azure Connect is useful in a variety of situations. An application running on Windows Azure might use Connect to link to an on-premises database system, for example, or a developer on the local network might use Connect to domain-join a cloud VM to an on-premises environment. While Connect isn't as general a solution as Virtual Network, it is significantly easier to set up. Developers can do it without bothering their network admins and with no extra hardware. Which approach is right for you depends on exactly what problems you need to solve.

Windows Azure Traffic Manager

Imagine that you've built a successful Windows Azure application. Your app is used by many people in many countries around the world. This is a great thing, but as is so often the case, success brings new problems. Here, for instance, your application most likely runs in multiple Windows Azure datacenters in different parts of the world. How can you intelligently route traffic across these datacenters so that your users always get the best experience?

Windows Azure Traffic Manager is designed to solve this problem. Figure 3 shows how.

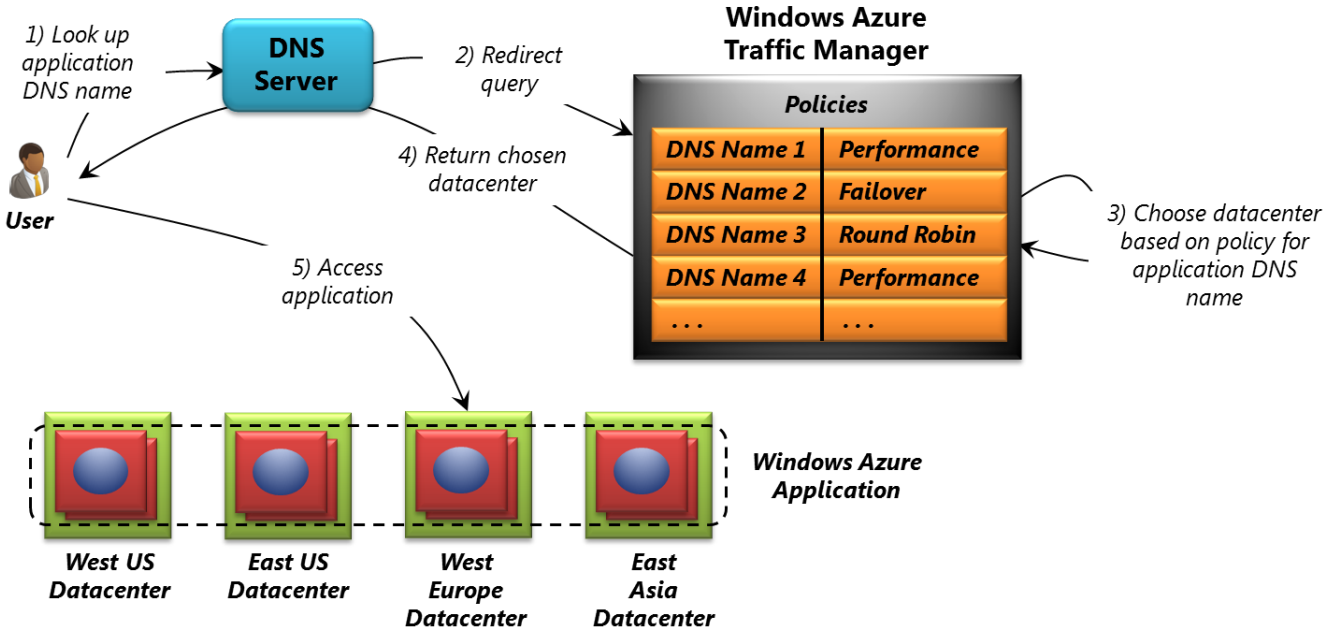

Figure 3: Windows Azure Traffic Manager intelligently directs requests from users across instances of an application running in different Windows Azure datacenters.

In this example, your application is running in VMs spread across four datacenters: two in the US, one in Europe, and one in Asia. Suppose a user in Berlin wishes to access the application. If you're using Traffic Manager, here's what happens.

As usual, the user's system looks up the DNS name of the application (step 1). This query is redirected to the Windows Azure DNS system (step 2), which then looks up the Traffic Manager *policy* for this application. Each policy is created by the owner of a particular Windows Azure application, either through a graphical interface or a REST API. However it's created, the policy specifies one of three options:

- Performance: All requests are sent to the closest datacenter.

- Failover: All requests are sent to the datacenter specified by the creator of this policy, unless that datacenter is unavailable. In this case, requests are routed to other datacenters in the priority order defined by the policy's creator.
- Round Robin: All requests are spread equally across all datacenters in which the application is running.

Once it has the right policy, Traffic Manager figures out which datacenter this request should go to based on which of the three options is specified (step 3). It then returns the location of the chosen datacenter to the user (step 4), who accesses that instance of the application (step 5).

For this to work, Traffic Manager must have a current picture of which instances of the application are up and running in each datacenter. To make this possible, Traffic Manager periodically pings each copy of the application via an HTTP GET, then records whether it receives a response. If an application instance stops responding, Traffic Manager will stop sending traffic to that instance until it resumes responding to pings.

Not every application is big enough or global enough to need Traffic Manager. For those that do, however, this can be a quite useful service.

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.