David Chappell

August 2011

THE MICROSOFT PRIVATE CLOUD

A TECHNOLOGY OVERVIEW

Sponsored by Microsoft Corporation

Contents

Introducing the Microsoft Private Cloud

Every organization wants to use its resources well. One way to do this is by running applications in virtual machines (VMs) rather than directly on physical machines. Virtualization has real value, and most organizations have adopted this approach in their datacenters. Useful as it is, however, the next step in the evolution of virtualization has become clear: It's private clouds.

A private cloud provides a more effective way to deploy, use, and manage VMs, applications, and other IT resources on hardware that's dedicated to a single organization. Microsoft's private cloud technologies, embodied in System Center 2012 and Hyper-V, make this possible. This paper provides a big-picture overview of this technology, explaining what it is and why it's useful.

What is a Private Cloud?

What exactly is a private cloud? And how is it different from traditional virtualization? Both questions have straightforward answers.

In a traditional virtualized environment, like those in most enterprises today, getting new resources requires human intervention. For example, if a developer needs a new VM, what happens commonly looks something like the process shown in Figure 1.

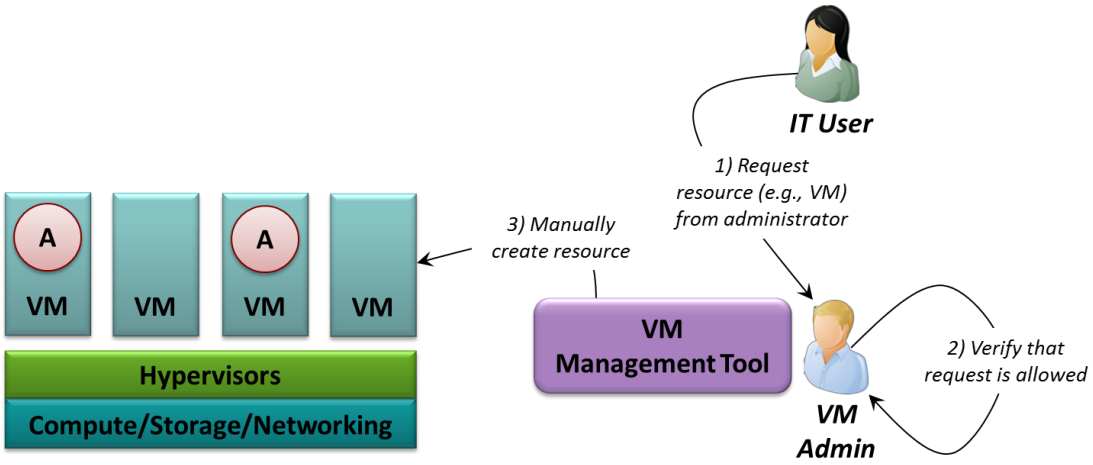

Figure 1: In traditional virtualization, an IT user requests resources from a person, then waits for a verification process to complete.

To get a new VM, application instance, or another resource, an IT user makes a request to the VM administrator (step 1). How this request is made varies; some organizations provide a Web portal to do this, while others rely on email or another less formal interaction. However it's done, the request causes the VM administrator to carry out some kind of process to determine whether the request should be granted (step 2). This might be simple: Perhaps the administrator can make his own decision based on available resources. Especially in larger organizations, however, the process might be more complex, requiring approvals from various people. The more complex the process, the longer it's likely to take, and so an apparently simple request for a new VM can take days or weeks to

approve. Assuming the request is granted, however, the VM administrator eventually creates the new resource using some VM management tool (step 3).

Why not automate this? Rather than require human intervention, why can't the IT user who needs a new resource allocate it herself from a pre-defined resource pool? This is the core idea of a private cloud, as Figure 2 shows.

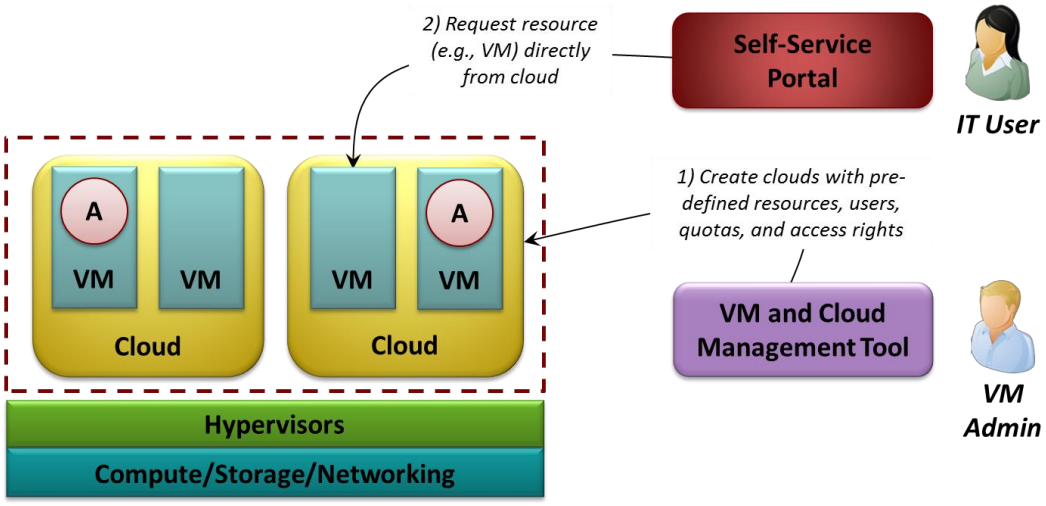

Figure 2: In a private cloud, an IT user requests a resource directly from a pre-defined pool.

Rather than requiring an approval process for each request made by an IT user, the VM admin can instead create one or more clouds in advance (step 1). An organization's finance department might have its own cloud, for instance, with another cloud for marketing and a third for the research group. Each cloud has a defined set of available resources and users, with pre-defined quotas that limit how much users of that cloud can consume. When an IT user needs a resource, such as a developer requesting a VM, she can now use a self-service portal to request this resource directly from a cloud to which she has access (step 2).

The benefits of doing this are apparent. For one thing, it's faster. Rather than waiting days or weeks, the IT user can have a resource allocated automatically in minutes. This approach is also likely to be cheaper, since there's no need to use the time of a highly skilled (and highly paid) VM administrator to carry out these simple requests. And especially for applications, this kind of automated installation reduces the chances for error in what can be a complicated process.

Getting these benefits requires some work, of course. Creating a private cloud requires up-front effort, such as deciding what resources each cloud should be allocated and what user quotas should be. An organization must also determine exactly which services each cloud will offer. A cloud used solely by a development group might provide only standard VMs, for example, while one used by a business unit might allow deploying any of a specific set of applications.

Another issue for many organizations is the loss of immediate control. While the VM admin still has ultimate power over the environment, he's no longer involved in the detailed allocation of new virtualized resources. Similarly, IT users no longer have a static set of physical or virtual servers that belong solely to them. Instead, they're allocated resources from the pool provided by a cloud.

These changes can be challenging, especially at first, and it's important to realize that adopting a private cloud isn't just a technology change—it also requires new attitudes and behaviors. Still, the benefits can be significant. What IT organization doesn't want to give its users faster service at a lower cost?

Private Clouds and Public Clouds

Before looking more specifically at Microsoft's private cloud technologies, it's worth examining the idea of cloud computing itself. The characteristics of this approach—the things that make a cloud a cloud—include the following:

- Pooled Resources: Rather than assigning fixed compute, storage, and networking resources to particular users, a cloud provides a resource pool that all users share. Which physical machine a VM runs on doesn't matter to a user—all she cares about is that the cloud meet the service level agreement (SLA) it promises. And because the physical resources are opaque to the cloud's users, those resources can be freely reconfigured as needed to optimize the cloud's service.

- Self-Service Access: Getting access to computing resources—VMs and more—without needing human approval for each request is a fundamental aspect of cloud computing.

- Elasticity: The set of resources a particular user has can grow and shrink over time. For example, a three-tier application might increase the number of Web server VMs it's running when an application's load increases, then decrease them when the load shrinks. From the point of view of a cloud's user, computing resources are elastic.

- Metered Use of Resources: Because resources can be allocated in a fine-grained way, such as per-VM per-hour, clouds can offer metered use. A user might be able to monitor the details of her resource use, for example, perhaps being charged only for what she uses.

Put simply, cloud computing allows self-service access to an elastic pool of IT resources. When this pool is provided by physical resources dedicated to a single user organization, it's viewed as a private cloud. When this pool is provided by physical resources shared by multiple user organizations, it's commonly viewed as a public cloud.

While both private and public clouds have all of the characteristics just listed, the idea of cloud computing first became visible in public clouds. One popular public cloud option, known as *Infrastructure as a Service (IaaS)*, offers VMs on demand. This same IaaS approach is what underlies most private clouds today (although more than just VMs can be provided, as already mentioned).

Yet it's misleading to think that organizations need to choose between private and public clouds. Some scenarios, such as supporting Internet-scale applications, are clearly a good fit for public cloud technologies such as Microsoft's Windows Azure. But private clouds are really just the evolution of today's virtualized datacenters. Any organization that is serious about virtualization today, regardless of its public cloud plans, should also be serious about private clouds. They're the next step on the virtualization path.

Creating a Private Cloud with System Center 2012

System Center 2012 allows creating and using private clouds. Customers and partners can use these technologies to offer traditional IaaS services, such as VMs on demand. This technology family also provides more, with support

for deploying multi-tier applications, monitoring and updating those applications, and automation services to make all of this more efficient. Figure 3 illustrates the primary technologies used to construct a Microsoft private cloud.

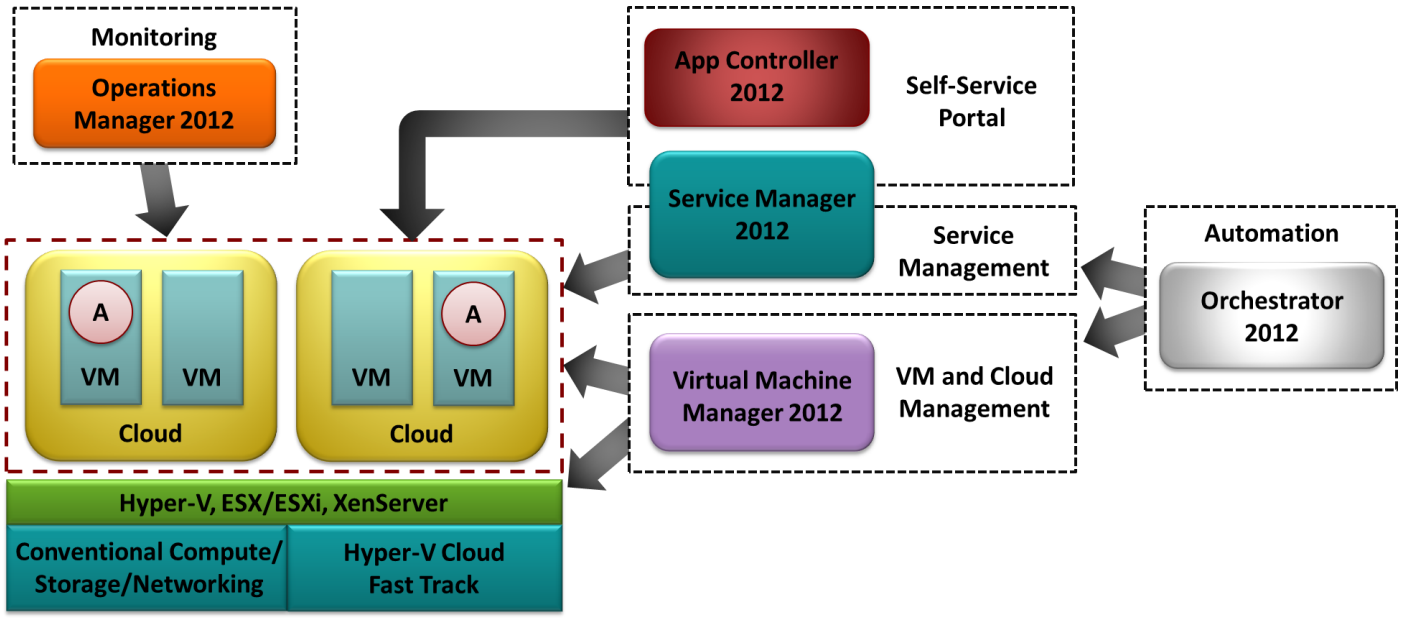

Figure 3: The Microsoft private cloud relies on several different System Center 2012 components and supports multiple hypervisors.

As the figure shows, the Microsoft private cloud supports three different hypervisors: Microsoft Hyper-V, VMware ESX/ESXi, and Citrix XenServer. It also supports conventional compute, storage, and networking hardware along with pre-packaged hardware configurations that conform to the Hyper-V Cloud Fast Track specification.

Building on this foundation, System Center 2012 can create a private cloud. The main components used to do this are the following:

- System Center Virtual Machine Manager (VMM) 2012, which provides the fundamental services for creating and managing clouds. It also provides the technologies used to deploy and update VMs and applications.
- System Center App Controller 2012, a self-service portal for requests made directly to a private cloud created with VMM 2012. (As described later, this portal can also be used with Windows Azure.)
- System Center Service Manager 2012, which provides automated IT service management and an associated self-service portal. If an organization wishes to define a process with human approvals for some private cloud scenarios, for example, Service Manager 2012 makes this possible.
- System Center Orchestrator 2012, providing a way to automate interactions among other management tools such as VMM 2012 and Service Manager.
- System Center Operations Manager 2012, which can monitor VMs, applications, and other aspects of a private cloud, then initiate actions to fix problems it detects.

All of these technologies depend on Windows Server 2008 R2 and Active Directory. It's also worth pointing out that the System Center 2012 suite as a whole includes more than this, with technologies for configuration management, data backup, and more. The focus here, however, is solely on the private cloud aspects of this family.

Using the Microsoft Private Cloud: Scenarios

The best way to get a sense of what the Microsoft private cloud provides is to walk through examples of how it can be used. This section looks at six scenarios:

- An administrator creating a cloud.
- A developer creating a virtual machine.
- An IT pro deploying an application.
- An administrator updating a deployed application.
- A developer creating a virtual machine with an approval process.
- System Center detecting and fixing a problem in a running application.

The goal is to provide a representative sample of how an organization can use the Microsoft private cloud and of the value it provides.

An Administrator Creating a Cloud

Before anybody can use a private cloud, someone must create it. In the Microsoft private cloud, this task is done primarily by a Virtual Machine Manager 2012 administrator. Figure 4 gives a high-level view of the process.

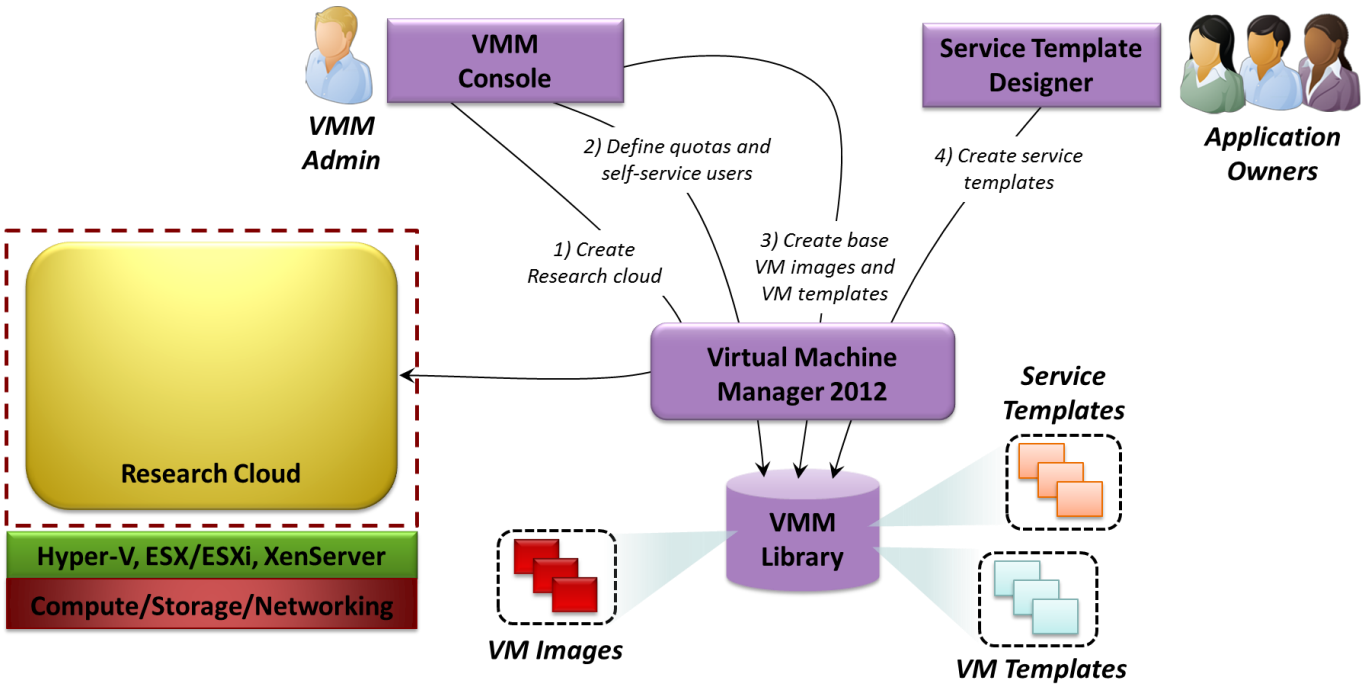

Figure 4: A VMM administrator creates a cloud and defines VM images and VM templates, while application owners create service templates for the cloud's applications.

Through the user interface provided by the *VMM console*, the VMM administrator creates a new cloud and gives it a name (step 1). In this example, the cloud provides IT resources that will be used by an organization's research group, and so the cloud's name is Research. When creating the cloud, the VMM admin specifies which physical resources are available to it, including servers, storage, and load balancers.

Next, the VMM admin defines quotas and self-service user roles for this cloud, information that's stored in the *VMM library* (step 3). Quotas let the admin specify limits on the total number of virtual machines that can be created in this cloud, on the amount of memory those VMs can use, on the total amount of storage available, and more. By creating self-service users, the admin defines exactly who has the right to create VMs, deploy applications, and perform other actions in this cloud. A cloud's users are actually defined in Active Directory, and they're typically specified as groups rather than individual users. For example, the Research cloud might have a Developers group whose members have the right to create VMs for their own use, and a BU IT Admin group containing administrators in this business unit who are allowed to deploy applications. The VMM admin can also define quotas for each self-service user, limiting the resources they can use. Different users can be allowed to consume different amounts of memory and storage, for example, or create different numbers of VMs. These limits prevent any one set of users from using too large a share of a cloud's resources.

Although it's not required, the VMM admin is also likely to define a base set of VM images and VM templates for this cloud (step 3). Despite their similar names, these two things are quite distinct:

- A *VM image* contains an operating system image that can be used to create a virtual machine. With Hyper-V and XenServer, these are stored in .vhd files, while ESX/ESXi uses .vmdk files.

- A *VM template* references a particular VM image, then defines other information needed to create a virtual machine using that VM image. Among other things, this information includes a hardware profile, specifying things such as the amount of memory the VM should have, and an operating system profile, which can define the name of the Active Directory domain the VM should belong to when it's created and more. Once VM templates exist, the cloud's users can create instances of new VMs from those templates.

In some situations, such as developers creating a development environment, creating VMs from VM templates might be sufficient. Yet a cloud created with VMM 2012 can do more than this: It can also work with complete applications. To allow this, the product introduces the notion of *service templates*. As described in more detail later, a service template can wrap together everything needed to deploy and run a multi-tier application. This includes the code for the application, descriptions of its database schemas, VM templates that describe the VMs it runs in, and more.

While it's possible for a VMM admin to create a service template for an application, it's more likely that an *application owner* will do this. This owner might be a developer who worked on the application, the person who deploys the application, or someone else. Whoever it is, she'll need to have knowledge of how the application is structured and how it should run. When a new cloud is created, it's likely that application owners will create some number of service templates for that cloud (step 4 in Figure 4). To do this, they use a *Service Template Designer* provided by VMM 2012. Users of the cloud can then deploy new instances of the applications defined by these service templates.

A Developer Creating a Virtual Machine

Once a cloud is created, using it is straightforward. Figure 5 shows perhaps the most basic scenario for using a private cloud: a developer requesting a virtual machine.

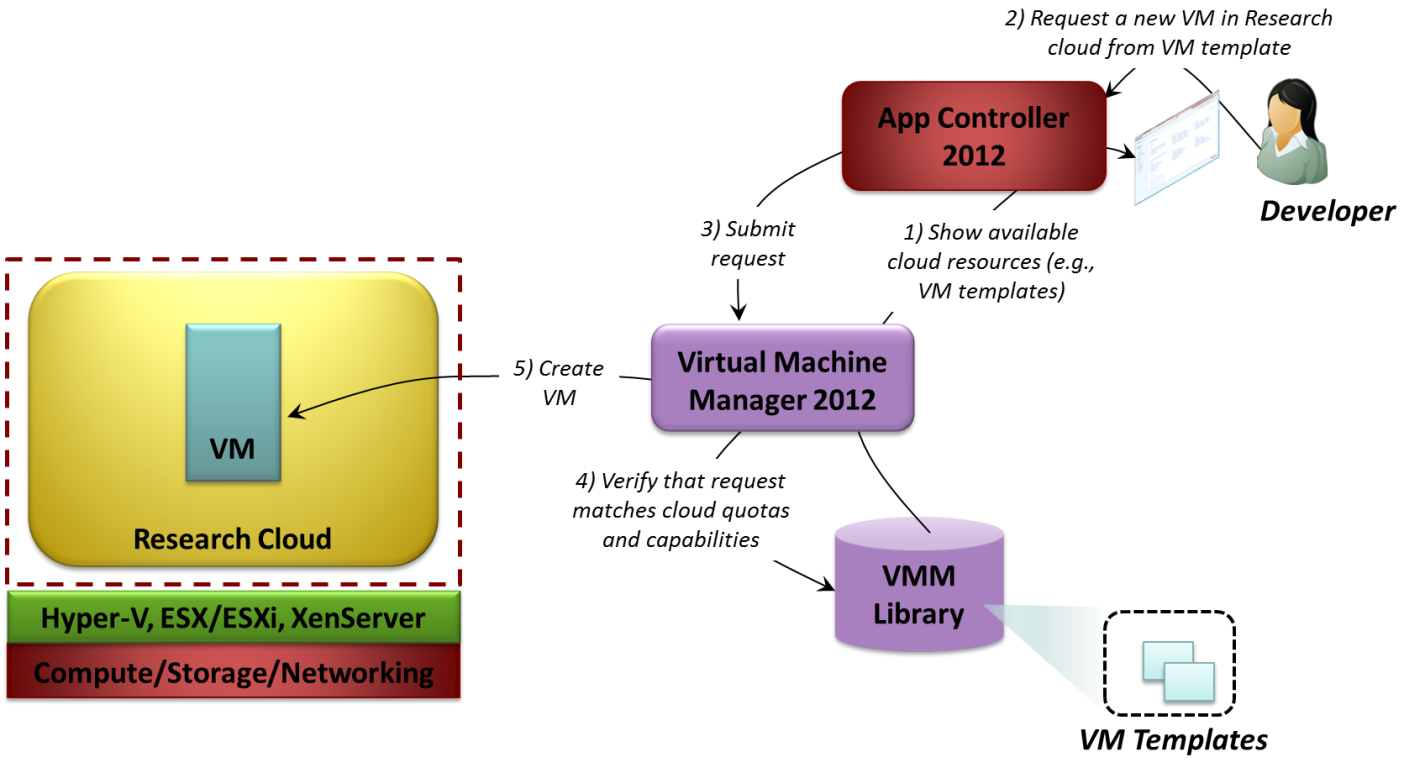

Figure 5: A developer can use App Controller 2012 to request a VM, then rely on VMM 2012 to create that VM.

As the figure shows, a developer can access the Research cloud through the self-service portal provided by App Controller 2012. Once she's logged in, the portal will show her all available resources in all of the clouds to which she has access (step 1). This includes any VMs she can create from VM templates available in the Research cloud. She chooses to create one of these VMs (step 2), and App Controller 2012 passes her request to VMM 2012 (step 3).

Before creating this VM, VMM 2012 makes sure that the request can be fulfilled (step 4). Doing this requires checking that it doesn't exceed either the quota for this cloud or the quota for the user making the request. VMM 2012 also makes sure that the request is within the capabilities of the cloud. (For example, a request might ask for a hardware resource that isn't provided in the Research cloud, such as high-speed storage.) If the developer's request passes all of these checks, VMM 2012 creates the VM she asked for in the Research cloud (step 5).

Notice that this entire process is automated. Because the private cloud was already defined, and because quotas had been specified for both the cloud and its users, there's no need for any approval process. Rather than waiting days or weeks, the developer gets her requested VM in minutes. Notice too that the developer creating the VM doesn't need to worry about what physical machine that VM is running on. VMM 2012 makes this decision based on its knowledge of the resources available for this cloud.

An IT Pro Deploying an Application

A traditional IaaS cloud provides VMs on demand. This is useful, since helping a developer get a VM more quickly is a good thing. But VMs are just a means to an end; the real business value of IT comes from applications. Letting an

organization deploy applications more effectively can be significantly more useful than just allowing faster access to VMs. Accordingly, the Microsoft private cloud allows deploying new applications as well as new VMs.

To understand why this is useful, think about how organizations typically deploy applications today. The process is often quite complicated—just describing it can take many pages. And the same application frequently needs to be deployed multiple times in development, test, staging, and production environments. Even once it's in production, updates commonly require deploying the application again.

Every time an application is deployed, the people carrying out the process have the opportunity to make a mistake. Given that one of the core benefits of a private cloud is increased automation, why not automate the application deployment process? By embedding this process in software, an organization can increase the odds of a successful outcome.

Doing this requires management software support, which is why VMM 2012 provides service templates. Each service template encapsulates everything required to deploy and run a new instance of an application. Just as a cloud's user can create new VMs on demand, he can also install and start new applications on demand. Figure 6 illustrates the process.

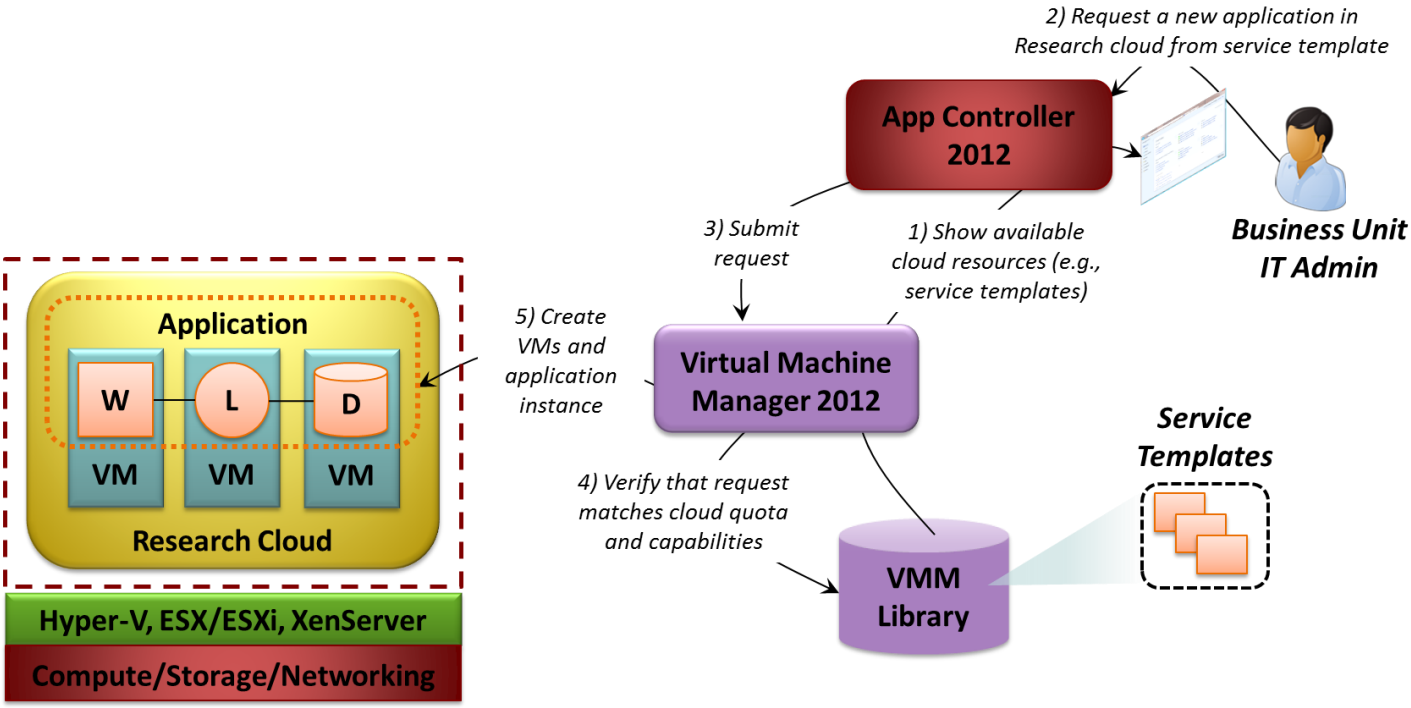

Figure 6: An administrator can request a new instance of a multi-tier application, then rely on VMM 2012 to deploy that instance.

As in the previous scenario, a user logged into App Controller 2012 sees a list of available cloud resources (step 1). In this case, however, that user is a business unit IT admin, not a developer, and his goal is to deploy an application rather than create a VM. To do this, he chooses an application from the list displayed by the portal (step 2), and this request is conveyed to VMM 2012 (step 3).

Choosing an application in App Controller 2012 in fact chooses a service template. VMM 2012 examines this service template to learn what's required to create an instance of the application it describes, then verifies that this request matches the cloud's quota and capabilities (step 4). If everything checks out, VMM 2012 will create one or more new VMs to run this application, install the application in those VMs, and start it running (step 5).

In this example, the application has three tiers—Web, business logic, and data—each running in its own VM. While this is common, it's not required; service templates are more general than this. It's even possible to define a service template that does nothing more than create a new VM (which can be quite useful, as described later). However they're applied, the goal of service templates is to provide a convenient way to package and deploy useful units of functionality.

Deploying applications through service templates also has other advantages. For example, a service template can specify that one or more of an application's tiers can be scaled out, with two or more VMs running in that tier. The template can then specify a load balancer to spread requests across those VMs, and the application owner can control how many VMs are running at any time. In a private cloud built using System Center 2012, using service templates to define and deploy applications is the recommended approach.

An Administrator Updating a Deployed Application

Deploying an application more quickly is all but certain to please that application's users. Yet an application (and the platform it runs on) is probably updated more often than it's deployed, and so making updates simpler also has significant value. Reflecting this, the Microsoft private cloud contains several technologies aimed at making it easier to update applications deployed from service templates.

For example, suppose a VMM administrator needs to apply patches to the copy of Windows Server 2008 R2 that's running in VMs used by a deployed application. With the Microsoft private cloud, the newly patched version of the operating system can be rolled out automatically to all applications that use it. Figure 7 illustrates how this process works.

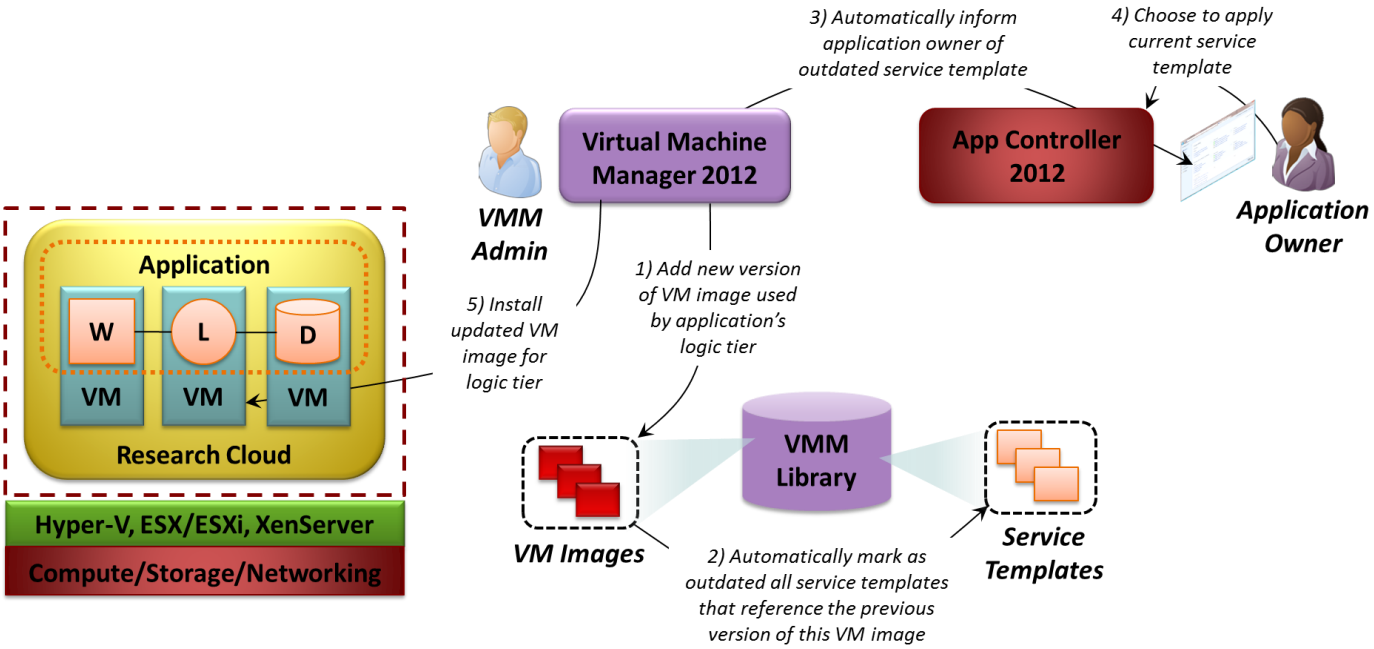

Figure 7: When a VMM administrator updates a VM image, that change can be propagated automatically to all applications that use the VM image.

Suppose the VMM administrator patches Windows Server 2008 R2 in a particular .vhd file, then imports this updated file as a new VM image into the VMM library (step 1). As mentioned earlier, each service template contains VM templates for the VMs it needs, and each VM template in turn contains a reference to a VM image. If, say, the logic tier of a deployed three-tier application uses this now-updated VM image, the new image needs to be deployed to that tier's VM. A deployed application is always linked to the service template it was created from, and so VMM 2012 can find all of the applications that use this service template.

To begin the application update process, VMM 2012 automatically marks as outdated all service templates that reference the previous version of the now-updated VM image (step 2). It doesn't immediately apply the updates to applications that use these service templates, however—making this kind of change without warning can sometimes cause problems. Instead, VMM 2012 uses the App Controller 2012 user interface to inform the application owner that the service template her deployed application uses is outdated (step 3). She can then choose to apply an updated service template now or at some later time (step 4). Once it gets approval from the application owner, VMM 2012 will automatically install the new VM image in the logic tier of this application (step 5).

This scenario is slightly simplified. (In fact, the service template itself also has an owner who must accept the change before the application owners are notified.) Still, it illustrates the key idea, which is that an update made to a single VM image can be automatically deployed to all applications that use that image. This automation makes updates faster and less error-prone. And because VM images can be updated separately from applications, it also lets organizations create and manage fewer of those images. Rather than using a distinct VM image for each application, administrators can reuse a smaller set of these images across multiple applications.

One last point to understand is that this kind of automated update is available only with service templates; it's not possible with VM templates. This is why it usually makes sense to wrap a VM template inside a service template—it allows automated updates to VMs as well as applications.

A Developer Creating a Virtual Machine with an Approval Process

One of the fundamental attractions of a private cloud is the ability for a user to get IT resources automatically—no human intervention is required. If a VMM administrator is comfortable with creating a cloud for a subset of his organization, defining services and quotas for that cloud, then letting people use the cloud as they wish, this approach can work well.

But what if he's not always willing to do this? Suppose there are some situations that still require IT users to go through a more formal approval process to get resources. This can't be done using App Controller 2012—this self-service portal is designed entirely for automated access. Instead, the Microsoft private cloud provides another option for doing this with Service Manager 2012.

The goal of Service Manager 2012 is to support IT service management in a broad sense. This includes implementing ITIL processes, such as change management and incident management, and it can also include processes for other things, such as allocating resources from a private cloud. For example, suppose an organization wishes to let developers allocate VMs from a private cloud, as shown earlier, but decides that these requests should sometimes require approval from the developer's manager. Figure 8 shows an example of how this can be implemented in the Microsoft private cloud.

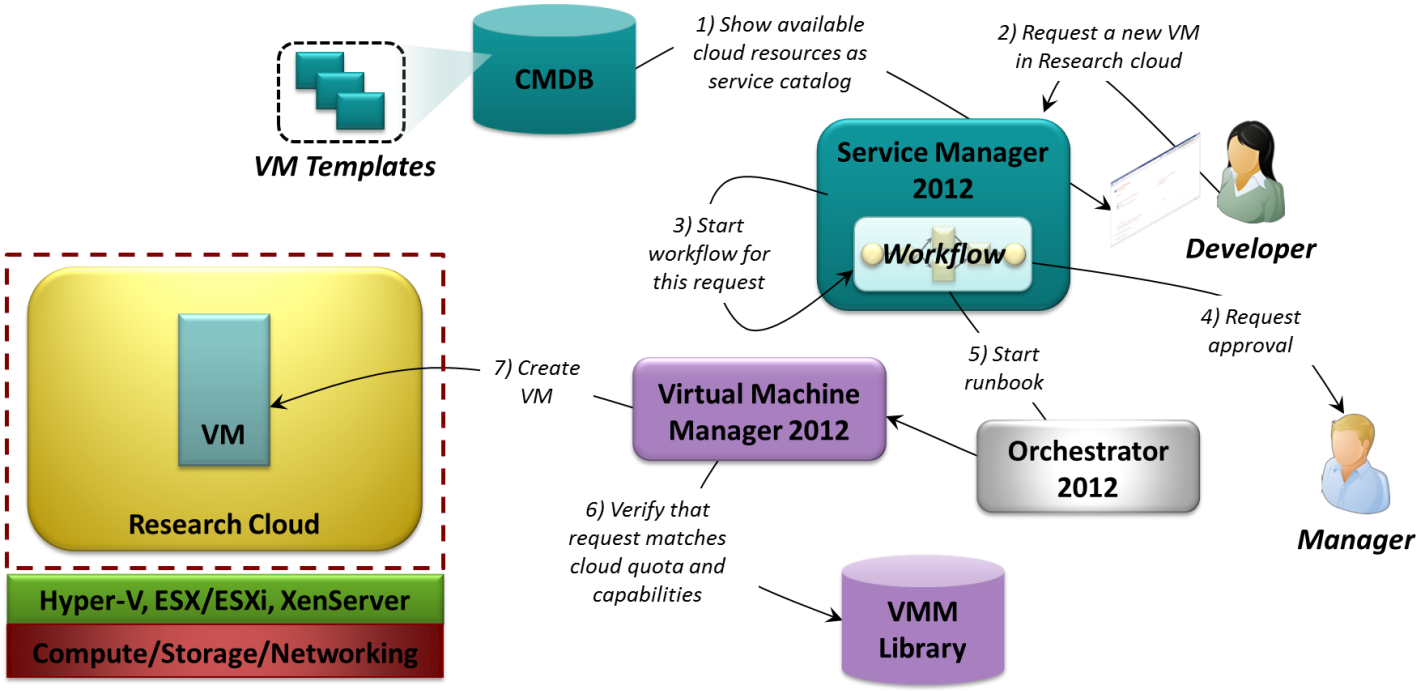

Figure 8: Using Service Manager 2012, a developer's request for a new VM can trigger a workflow that implements a process such as requesting approval from her manager.

As the figure shows, Service Manager 2012 maintains a *Configuration Management Database (CMDB)*. The CMDB is the repository for nearly all configuration and management-related information in the System Center 2012 environment. With the Microsoft private cloud, this information includes VM templates, which are copied regularly from the VMM library into the CMDB.

As the figure also shows, Service Manager 2012 provides its own self-service portal. Using the information in the CMDB, Service Manager 2012 can construct a *service catalog* that shows the services available to a particular user (step 1). In this case, that user—a developer—decides to create a VM in the Research cloud (step 2). Rather than passing this request directly on to VMM 2012, however, as App Controller 2012 does, Service Manager 2012 instead starts a *workflow* to handle the request (step 3). This workflow can do pretty much anything—it's custom logic—but in this example, it does two things. First, suppose that the developer's organization is charged by the central IT group for each VM she creates. Because of this, the workflow contacts the developer's manager to get his approval for this request (step 4). If the request is approved, the workflow then starts an Orchestrator 2012 *runbook* (step 5).

A runbook is essentially another kind of workflow. While a Service Manager workflow is designed to implement ITIL-style processes, however, a runbook is designed to interact directly with system management tools. Accordingly, Orchestrator 2012 provides a range of pre-built components for interacting with many other technologies (including non-Microsoft management tools). Among these are some designed to interact with VMM 2012. In this example, the runbook relies on Orchestrator 2012 components to ask VMM 2012 to create a new VM in the Research cloud (step 5). (Doing this relies on a PowerShell API exposed by VMM 2012.) As before, VMM 2012 checks that this request is within the user's quota and the cloud's capabilities (step 6), then creates the requested VM (step 7).

It's important to note that, unlike the scenarios shown earlier, this example doesn't necessarily show a built-in function of System Center 2012. Instead, Service Manager 2012, Orchestrator 2012, and the other components shown here make it possible for an organization to build its own custom workflow and runbook to accomplish this. Microsoft will provide solution accelerators, however, that give users of this technology predefined implementations of the most important private cloud scenarios. The key point is that automated resource allocation with App Controller 2012 isn't the only option in the Microsoft private cloud. Using Service Manager 2012, a private cloud can also handle requests that require human approvals or carry out other processes.

System Center 2012 Detecting and Fixing a Problem in a Running Application

Private clouds are an effective way to give IT users the computing resources they need. Yet while providing resources automatically is useful, it's not enough. Wherever possible, it also makes sense to automate management of those resources.

For example, one of the key aspects of the Microsoft private cloud is its support for applications, not just the VMs those applications run in. Various things can go wrong while an application runs, many of which can be fixed (or better yet, avoided) without human involvement. The components of System Center 2012 can work together to accomplish this, as Figure 9 shows.

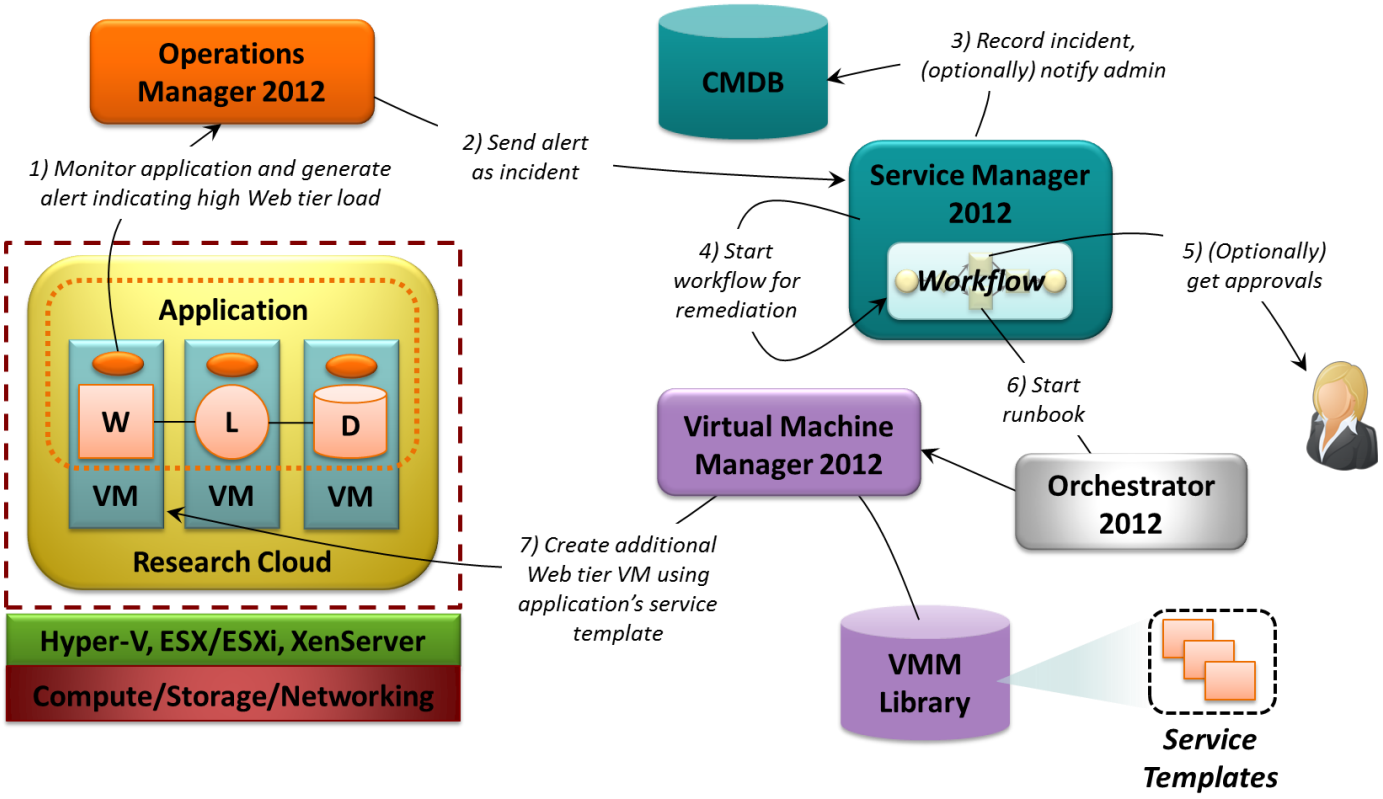

Figure 9: Operations Manager 2012, Service Manager 2012, and other System Center components can work together to automatically detect and fix problems with a running application.

In this example, a three-tier application has been deployed by VMM 2012 from a service template. As described earlier, this template allows requests from the application's users to be load balanced across two or more servers in its Web tier. Suppose that the number of incoming HTTP connections gets too high, forcing users to wait too long for responses. If the application is monitored by Operations Manager 2012, an Operations Manager agent in a Web tier VM (the orange ovals in Figure 9) can generate an *alert* indicating that the load on those servers has gotten too high (step 1). When it receives this alert, Operations Manager 2012 can send it as an *incident* to Service Manager 2012 (step 2). Service Manager 2012 then records the incident in the CMDB and (optionally) notifies someone via email (step 3). Next, Service Manager starts a workflow to remedy this problem (step 4). As in the previous example, this workflow might optionally request human approval before proceeding (step 5). In any case, the workflow starts an Orchestrator 2012 runbook to interact with VMM 2012 (step 6). This runbook uses the VMM PowerShell API to tell it to add another VM to the application's Web tier. To do this, VMM 2012 reads the application's service template to learn about this tier, then creates a new VM, installs the application's Web tier code in this VM, and starts it running (step 7). While people can be involved in this process, it's not required. The entire scenario, from discovering the problem to fixing it, can be entirely automated.

As with the previous scenario, the workflow and runbook required to do this might be entirely custom, with Service Manager 2012 and Orchestrator 2012 providing the foundation. Different organizations will implement this in different ways, but the main point remains the same: Managing applications well is an essential aspect of working with a private cloud.

Examining the Microsoft Private Cloud: A Closer Look at the Technology

Understanding the main scenarios for using the Microsoft private cloud provides a feeling for the technology and what it can do. Really understanding what's going on requires a slightly deeper look, however. This section delves a bit further beneath the surface of this technology suite.

Hardware: Compute, Storage, and Networking

Like all software, a private cloud depends on hardware: servers, storage, networks, and more. The Microsoft private cloud can run on traditional hardware configurations, including conventional or blade servers, a storage area network (SAN), and various load balancers. Many organizations are likely to deploy the software in this kind of environment.

Yet doing this requires validating that everything—hardware and software—works together, which can be complex and time consuming. One way to avoid this is to deploy the Microsoft private cloud using a pre-integrated package. Especially when an organization is deploying a private cloud on new hardware, choosing this approach is likely to be faster. To make this easier to do, Microsoft has defined the *Hyper-V Cloud Fast Track Architecture*. This specification defines a reference architecture designed to support a private cloud, including compute, storage, networking, and Microsoft software. A number of vendors provide products today that support the Fast Track architecture, including Dell, HP, IBM, Fujistu, Hitachi, NEC, and Cisco/NetApp.

Hypervisors: Hyper-V and More

The Microsoft private cloud supports three hypervisors: Microsoft Hyper-V, VMware ESX/ESXi, and Citrix XenServer. The private cloud aspects of System Center 2012 work identically across all three, even in environments that combine two or more of these options[1]. Still, it's worth looking in a bit more detail at Microsoft's hypervisor.

Hyper-V is in its third release today, and it supports the typical functions of a mature hypervisor. These include:

- *High availability (HA)*, which allows grouping servers together into a cluster, then letting a failed server's work be automatically shifted to another server in the cluster.
- *Live Migration*, which allows a VM to be moved from one physical server to another without any disruption to applications running in that VM or to the people using these applications. To perform maintenance on a physical server, for example, an administrator can use Live Migration to move all of that server's VMs to another machine, perform the maintenance, then move the VMs back to their original home. Users of applications running in those VMs won't notice any changes throughout this process.

Hyper-V also supports a range of guest operating systems, including several versions of Windows and Windows Server, SUSE Linux, Red Hat Enterprise Linux, and CentOS.

[1] Some aspects of System Center do have hypervisor-specific behaviors, however. For example, VMM 2012 can deploy a hypervisor onto a bare metal server, but only if that hypervisor is Hyper-V.

Creating a Cloud

As described earlier, an administrator creates clouds using Virtual Machine Manager 2012. The physical resources a cloud uses can come from either *host groups*, each of which can contain some combination of physical servers running Hyper-V, ESX/ESXi, or XenServer, or *VMware resource pools*, which contain only servers running ESX/ESXi. For each cloud, the administrator can specify its total capacity along with many details: what its VMs can look like (including options for processors, memory, disks, and network interface cards), whether the VMs use HA, and more. The administrator can also create different categories of storage, giving each one a different name. For example, Gold storage might be fast and expensive, while Silver storage is slower but cheaper. An IT user creating a VM in that cloud can then choose the storage option that best meets her needs by specifying just the category's name—she needn't be aware of the underlying details.

The VMM administrator also defines what the cloud's users are allowed to do. Along with per-user quotas, VMM 2012 provides a range of fine-grained options for specifying this. There are separate permissions for authoring a VM template or service template, for example, for starting a VM or application from those templates, and for stopping a VM or service. A VMM administrator can also create delegated administrators who have limited access to specific parts of the physical environment, such as a branch office.

Creating a cloud isn't enough to make it useful, however. It also needs VM templates and service templates that let users create VMs and applications. The next sections take a closer look at these fundamental concepts in the Microsoft private cloud.

Using VM Templates

A VM template defines the specifics of how a VM instance should look. Figure 10 illustrates this idea.

Figure 10: A VM template contains a link to a VM image together with the information required to create a running VM instance.

As the figure shows, every VM template contains a link to a VM image. This image, such as a .vhd file, contains the guest operating system that will run in this VM, together with any applications or other software that has been installed on the image. Creating a VM requires more than this, however, and so a VM template can contain more. The possibilities include:

- A hardware profile that describes the VM itself. For example, a VM template might specify that the VM has 1 CPU, 8 gigabytes of memory, and is attached to a logical network named "DevTest", as Figure 10 shows.

- A guest operating system profile that describes specifics about the OS running in this VM. In the example shown here, for instance, a VM created from this VM template will become part of the Active Directory domain "CorpDev" and have its administrator password set to "p@W".

VM templates originally appeared in VMM 2008, an earlier version of the product. With the addition of service templates in VMM 2012, however, standalone VM templates become much less attractive. For most private clouds, using service templates makes more sense, as described next.

Using Service Templates

Just as a VM template provides the information required to create an instance of a VM, a service template provides the information required to create an instance of a service. A service can be a multi-tier application, as shown earlier, but this isn't required—it can also be just a simple VM. No matter what it describes, however, a service template is constructed from the same basic components. Figure 11 shows a slightly simplified picture of what a service template can contain.

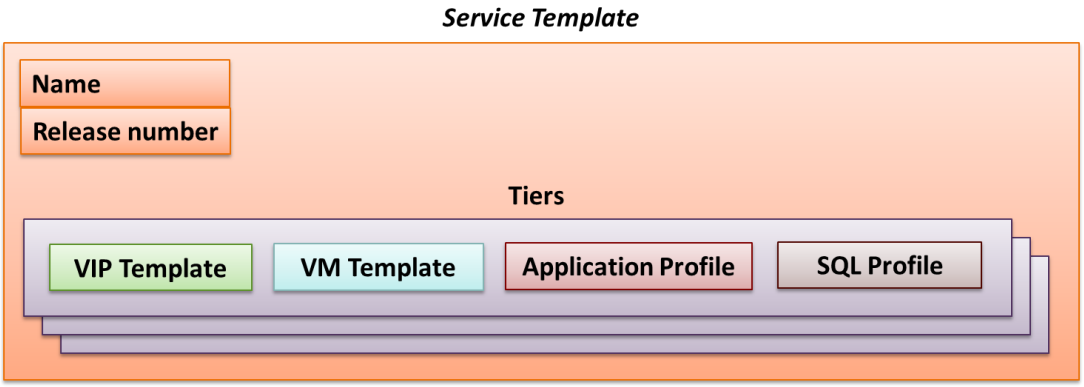

Figure 11: A service template can contain multiple tiers, each with a VM template, an application profile, and other information.

Every service template has a name and a release number. It also contains one or more tiers, each describing some part of an application or VM. A tier can contain some combination of four components:

- A VIP template, describing the characteristics of a virtual IP address that will be supported by a hardware load balancer connected to this tier. For example, a VIP template might specify that a particular virtual IP address requires support for SSL, sticky sessions, or other options.
- A VM template, describing the kind of VM this tier should use. This is a standard VM template, as just described, and it's the only mandatory component in a tier.
- An application profile, referencing the application code this tier runs (if any). This profile can also reference various kinds of scripts, such as scripts that run before and after the tier's application code is installed.
- A SQL profile, referencing schema definitions and other information for a SQL Server database.

Defining a Service Template for a VM

The simplest way to use a service template is just to wrap a VM template. Figure 12 shows how this looks.

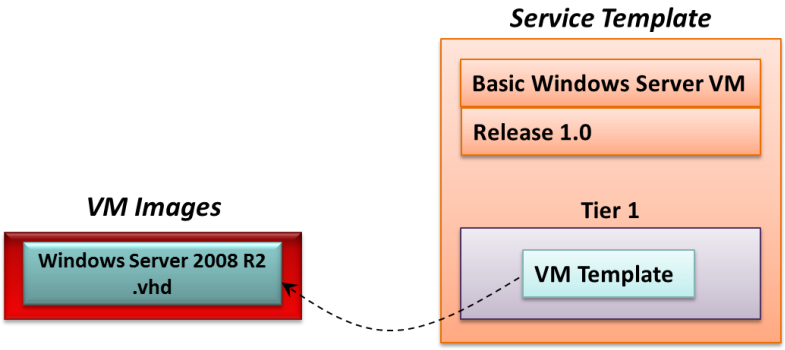

Figure 12: A simple service template can contain just a single tier with a VM template.

In this example, the service template is named "Basic Windows Server VM", and it's currently at release 1.0. This service template has only a single tier, and that tier contains only a VM template. As always, the VM template points to a VM image that the VM will use when it's created.

This raises an obvious question: Why do this? A service template wrapped around a VM template doesn't really add any extra information, so why bother? The answer stems from what happens when a new VM is created. If a VM is created from a VM template, no connection is maintained between the VM and that template. Changes to the VM template can't be automatically propagated to the VM itself. If that VM is created from a service template, however, such as the one in Figure 12, it maintains a connection to the service template. If the VMM administrator wishes to apply an updated VM image, for example, or perhaps increase the VM's memory, all he has to do is change the service template. As described earlier, those changes will propagate automatically to all VMs created from this service template. Because of this, Microsoft recommends always wrapping VM templates in service templates—the approach shown in Figure 12 is the new normal for VMM.

Defining a Service Template for an Application

Using service templates to create VMs is handy. Service templates can also be used, of course, to describe complete applications. Figure 13 shows a service template for a typical three-tier application.

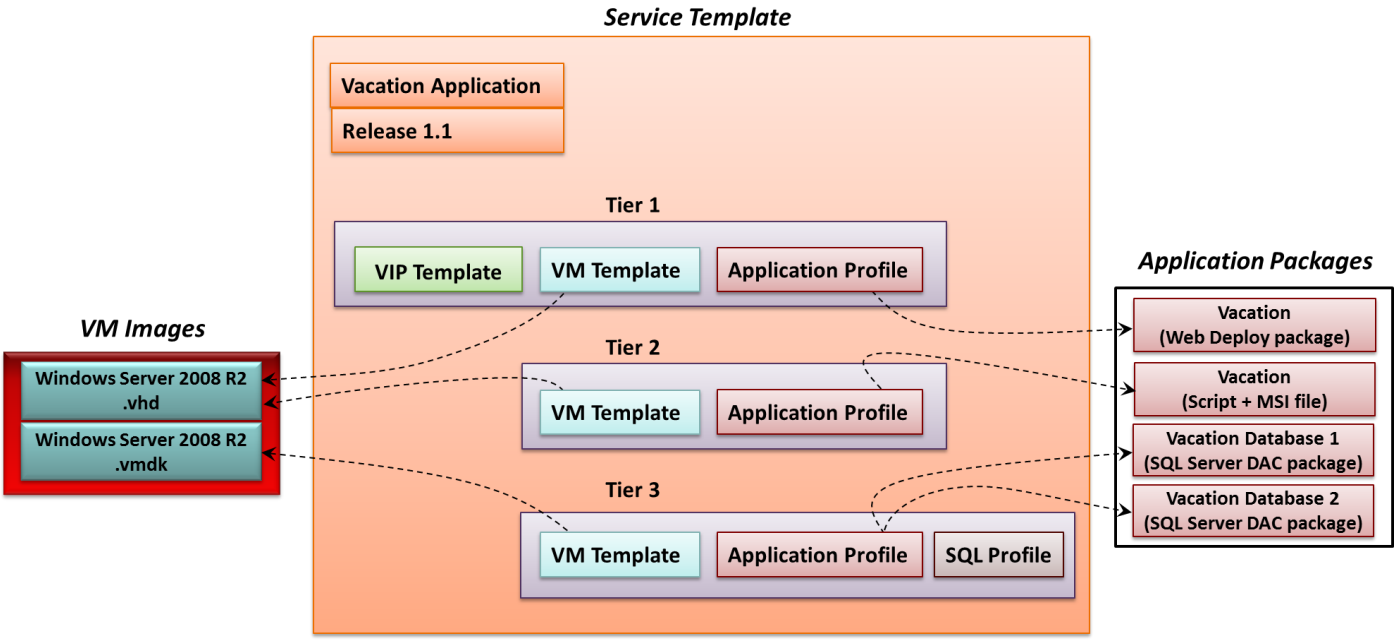

Figure 13: A service template can describe an entire three-tier application.

Once again, the service template has a name—"Vacation Application"—and a release number. Tier one in the application, the Web tier, includes a VIP template, a VM template that references a Windows Server 2008 R2 image, and an application profile that references a Web Deploy package. When an application is deployed from this service template, the Web Deploy package will be installed in each VM created using the tier's VM template. Those VMs will then be placed behind the virtual IP address specified in the VIP template.

Tier two contains a VM template that points to the same VM image as tier 1. It also contains an application profile that references a script capable of installing an MSI file with the code for this application's middle-tier business logic. When the application is deployed, this MSI file gets installed in a VM created using the VM template. In this case, there will be only a single instance of the middle-tier business logic, and so the tier doesn't define a load balancer—there's no VIP template.

Tier three contains a VM template that points to a VMware .vmdk image. It also includes an application profile that references two Data Tier Application (DAC) packs for SQL Server and a SQL profile component. When the application is deployed, the VM image it uses must have SQL Server installed. Once the VM is created, the SQL profile component is used to configure that SQL Server instance. The DAC packs are then used to configure the databases this application needs in SQL Server. (And although it's not described here, there's also an option that lets an application install DAC packs into an existing SQL Server instance.)

The tiers in a service template map directly to the tiers in a typical application, and so the concepts aren't hard to understand. Still, it's worth looking at one more example: an application that uses *Server App-V*, a technology that's new with VMM 2012. Figure 14 shows how this looks.

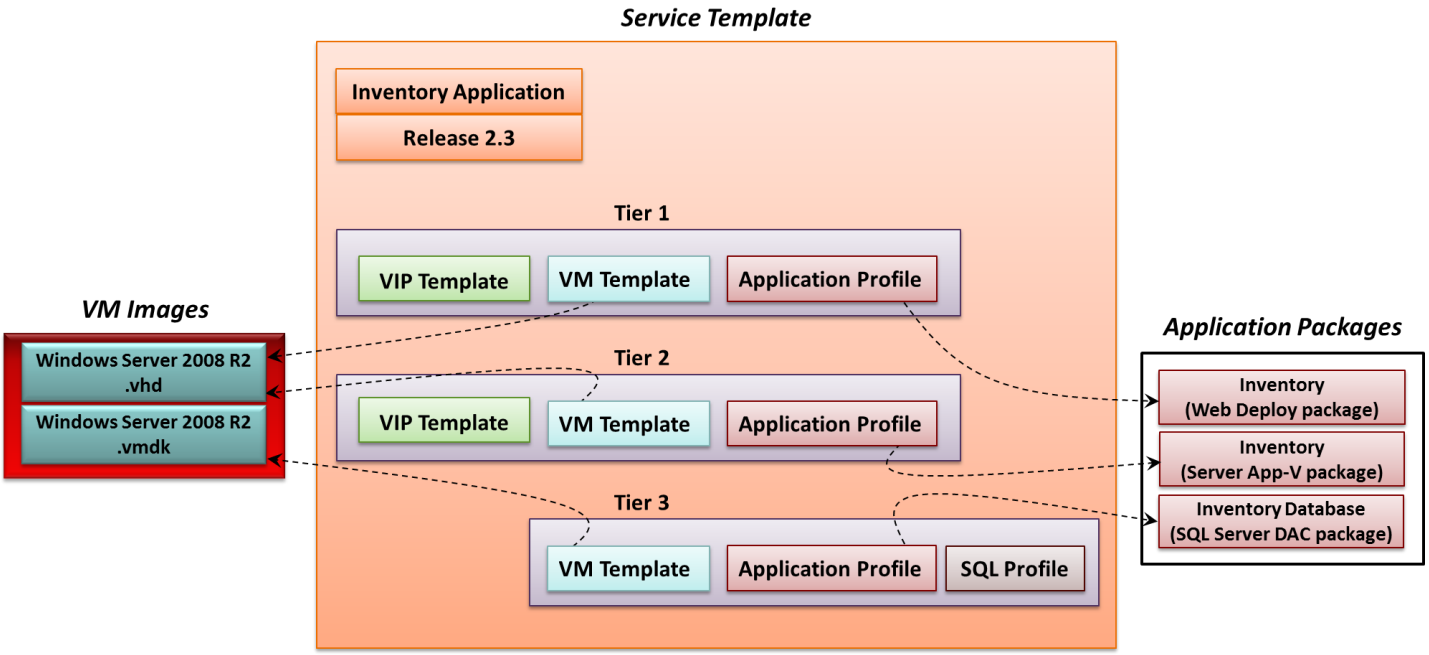

Figure 14: A service template that supports image-based update can use a Server App-V package for its business logic.

This service template is much like the previous example. It has a name, a release number, and three tiers. One difference is that tier two now includes a VIP template, which means that multiple instances of the middle tier business logic can run simultaneously behind a load balancer with user requests spread across them. Another difference is that the tier 2 application profile references a *Server App-V package* rather than a script and MSI file.

Understanding the role of Server App-V requires taking a closer look at how VMM 2012 uses service templates to apply updates. There are two options:

- *Conventional* or *in-place updates*: Updates are applied to the currently running VMs. This option can be used for most kinds of updates, including installing a new version of the application or changing the amount of memory allocated to this VM. In-place servicing can't be used to replace the VM image of a running application, however.

- *Image-based updates*: With this approach, VMM 2012 first creates a new virtual disk with a new VM image, such as a newly patched version of Windows Server, for the VM being updated. It installs the tier's application on top of this image, then replaces the tier's running VM boot disk with the disk containing this new VM image. While the VM being updated will be unavailable during this servicing operation, it nonetheless gets a new VM image while retaining its name, IP address, and other attributes. (In fact, the application update scenario shown earlier in Figure 7 used this option, since it replaced the VM image for a running application.)

Conventional update can be used with any kind of Windows application. Using image-based update, however, imposes a few requirements. To understand why this is true, think about what must happen when, say, a virtual machine containing the middle-tier business logic of a running application must receive a new VM image. First, the application's code must be installed in the new VM image, as just described. Since service templates contain a

complete description of the application, VMM 2012 can do this with no problem. And if the running application maintains no state within its VM, nothing more is required. But applications often make local changes within their VM, such as modifying the Windows registry or relying on local configuration files. If a running application has done this, making it run successfully on a new VM image requires moving this state as well.

VMM 2012 provides two options for doing this. One is to wrap the application code in a Server App-V package, as in Figure 14. Done through a process called *sequencing*, this lets a Server App-V agent running in the application's VM detect and track any local state changes the application makes. When this application is installed into a new VM image, the Server App-V agent saves the application's current state, including registry changes, config files, and more, then replicates this state in the disk containing the new VM image. This lets the application continue running where it left off with all of its accrued local state intact.

Not every application can be wrapped in a Server App-V package, however—there are some restrictions—and so VMM 2012 provides another option for doing image-based updates. For applications that can't use Server App-V, a service template's application profile can reference custom scripts that save and restore the state of a running application when an image-based update is performed. Rather than relying on the Server App-V agent to do this, the creator of the service template can instead supply scripts that carry out the same tasks.

Changing a VM image can still be done manually, of course—there's no requirement to use VMM 2012's image-based update. Why bother with this new approach? There are two main reasons:

- Because image-based update can effectively install a new VM image beneath a running application, it allows separating applications and VM images. This eliminates the need to have a separate VM image for each application that uses that image. Instead, an organization might choose to use the same small set of VM images for many applications, combining the two as needed with service templates. Managing fewer VM images is simpler, cheaper, and less error-prone.

- Temp files, unused configurations, and many other things can collect over time in a VM, slowing down its performance. Because image-based update installs a new copy of the application on top of a fresh VM image, any baggage the currently running VM has collected will no longer be present.

If an application's logic is wrapped in a Server App-V package, as with tier two of the application shown in Figure 14, it's a good candidate for image-based update. But what about the other tiers? Assuming the Web Deploy package used in tier one maintains no state, this tier can certainly use image-based update. Using image-based update with tier three, however, is unlikely. This tier maintains a great deal of state—the application's data—and so using image-based update here may well be impractical.

Updates without Downtime: Using Upgrade Domains

If the creator of the service template has enabled this option, VMM 2012 can create two or more VMs for a tier. When an application has multiple VMs running in a tier, those VMs can be divided into two or more *upgrade domains*. Doing this lets that tier remain continuously available to its users while it's updated. Figure 15 illustrates the idea.

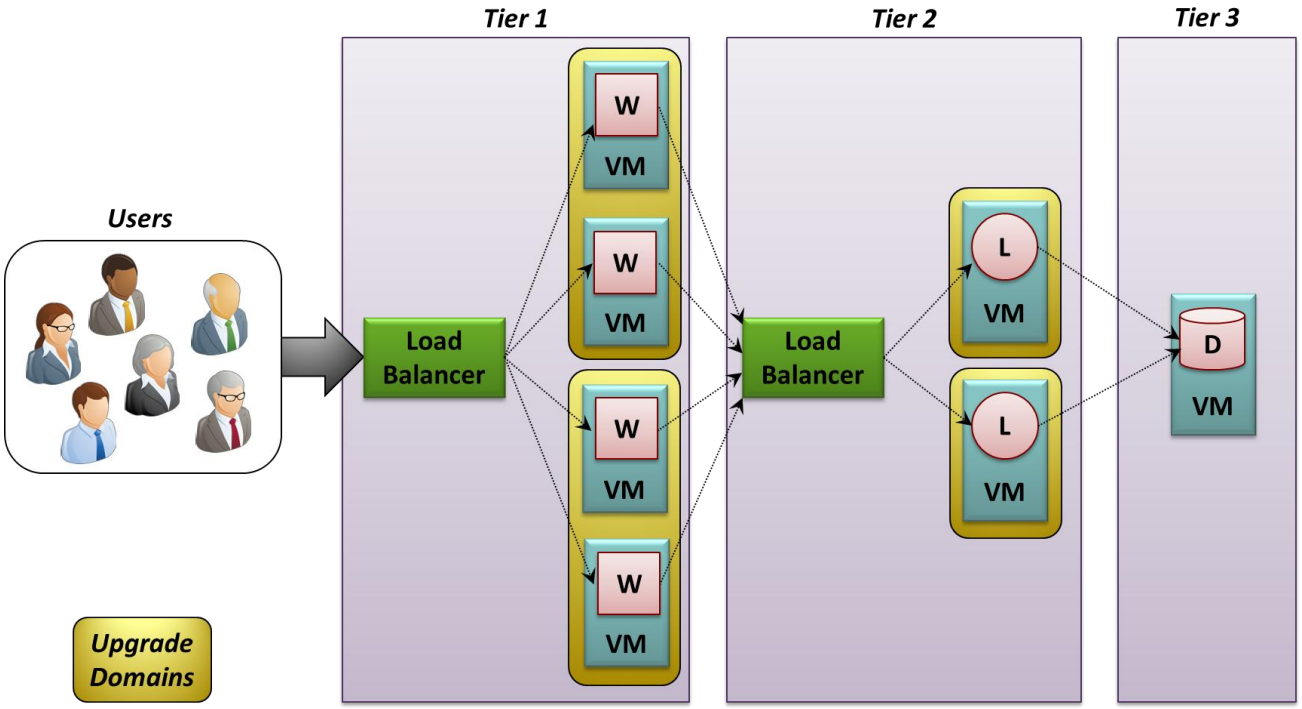

Figure 15: Grouping a tier's VMs into upgrade domains lets the tier be updated with no downtime.

The application shown here might have been created from the service template shown in Figure 14, since it has load balancers in tiers one and two. Tier one is currently running four VMs, grouped into two upgrade domains, while tier 2 has two VMs, each in its own upgrade domain. The function of an upgrade domain is implied by its name: The VMs in one upgrade domain can be updated independently from the VMs in another upgrade domain within that tier.

For example, suppose that VMM 2012 is used to increase the memory size of the VMs used by tier 1—the Web tier—in the application shown in Figure 15. To do this, it can remove from the load balancer only the VMs in one upgrade domain, apply the update, then add them back to the load balancer. Once they're running, VMM 2012 can then repeat this procedure with the VMs in the second upgrade domain. Since some of the tier's VMs remain running throughout the update process, the application itself need never go down.

But what about the data tier? As Figure 15 suggests, using upgrade domains here is atypical, yet the VM and the database it contains still need to be updated. The best solution is to use SQL Server clusters, updating one VM in the cluster at a time.

Creating Service Templates

Service templates are a central aspect of the value provided by the Microsoft private cloud. Once the concepts are clear, creating these templates is straightforward. As shown back in Figure 4, application owners do this using the Service Template Designer provided by VMM 2012. Figure 16 shows how this tool looks.

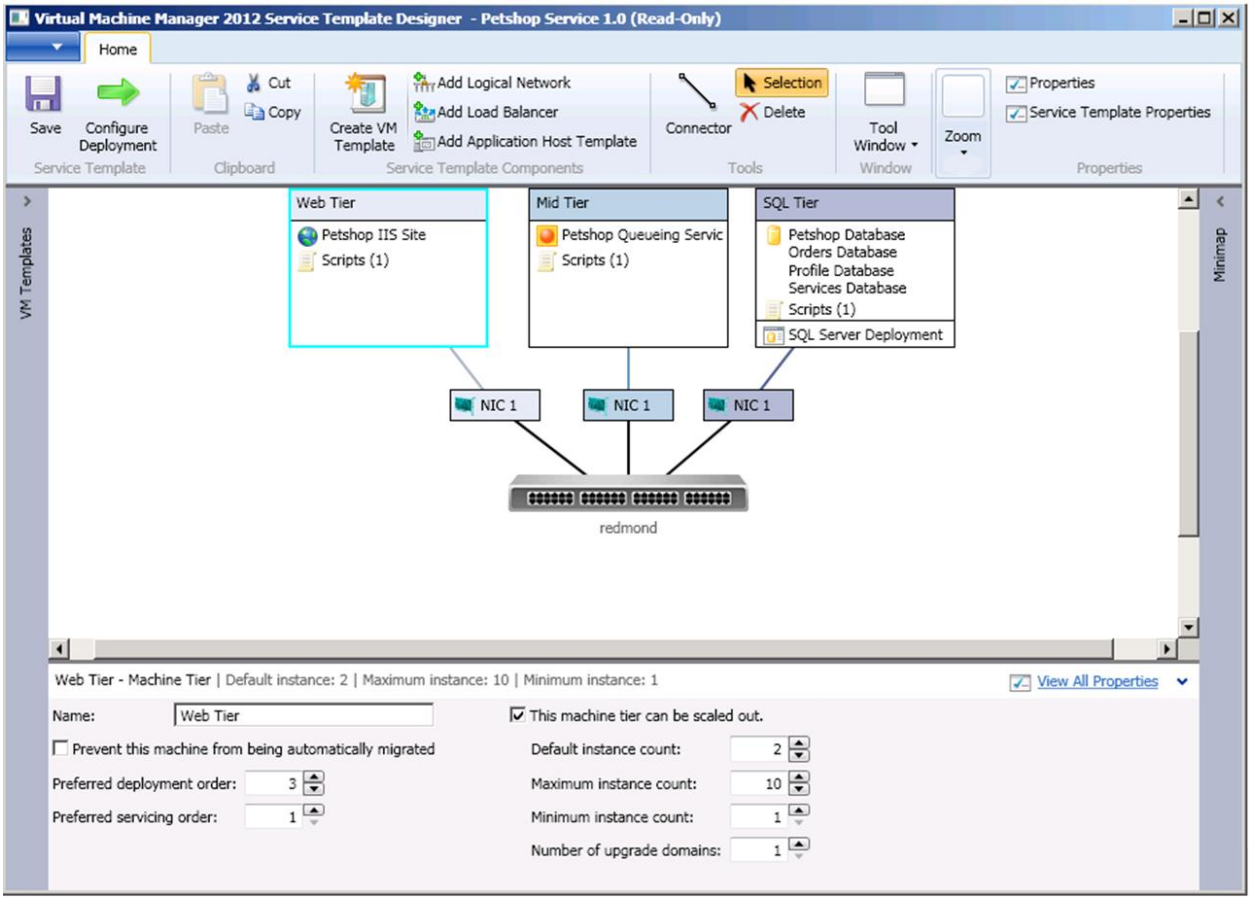

Figure 16: The Service Template Designer is a graphical tool for creating service templates.

At the top is a ribbon providing access to the Designer's basic functions. In the middle is a canvas on which a user can graphically assemble and define the tiers of her application. At the bottom is a pane for setting properties, such as the instance counts for each tier, the number of upgrade domains, and the order in which tiers should be deployed. From all of these settings, the Designer generates the service template, which can be exported and imported across different VMM 2012 systems as needed. The goal is to make life as simple as possible for the people who create and manage these templates.

Monitoring and SLAs

Whether or not an organization uses a private cloud, monitoring matters. Everybody who's responsible for a computing environment wants to know what's happening in their world. Yet private clouds, with their clear separation between a cloud's providers and its users, create new monitoring requirements. To understand why, think about the different monitoring needs of those two groups.

The cloud's providers create and maintain the infrastructure, and so they need to monitor physical components, such as servers and storage, as well as the software that runs on these components, such as operating systems and VMs. The cloud's users need to monitor the availability and performance of the VMs and applications they deploy. They're not concerned with the underlying infrastructure—they can't even see it. In the Microsoft private cloud,

VMM administrators are cloud providers, while application owners, IT pros who deploy applications, and developers are cloud users.

Operations Manager 2012 addresses the needs of both groups. To help VMM administrators, the product includes a broad set of functions. They allow monitoring physical components, such as servers and network equipment, along with software, including operating systems and VMs. Operations Manager also allows monitoring applications, providing real-time information about their execution. For example, when a VM is deployed from a VM template, that new VM is automatically added to the set of things that Operations Manager 2012 is monitoring. Similarly, when an application is deployed from a service template, Operations Manager 2012 is informed of this addition. This tool then applies its usual application monitoring functions to the new service, automatically creating a diagram for the application. Figure 17 shows an example of how this looks.

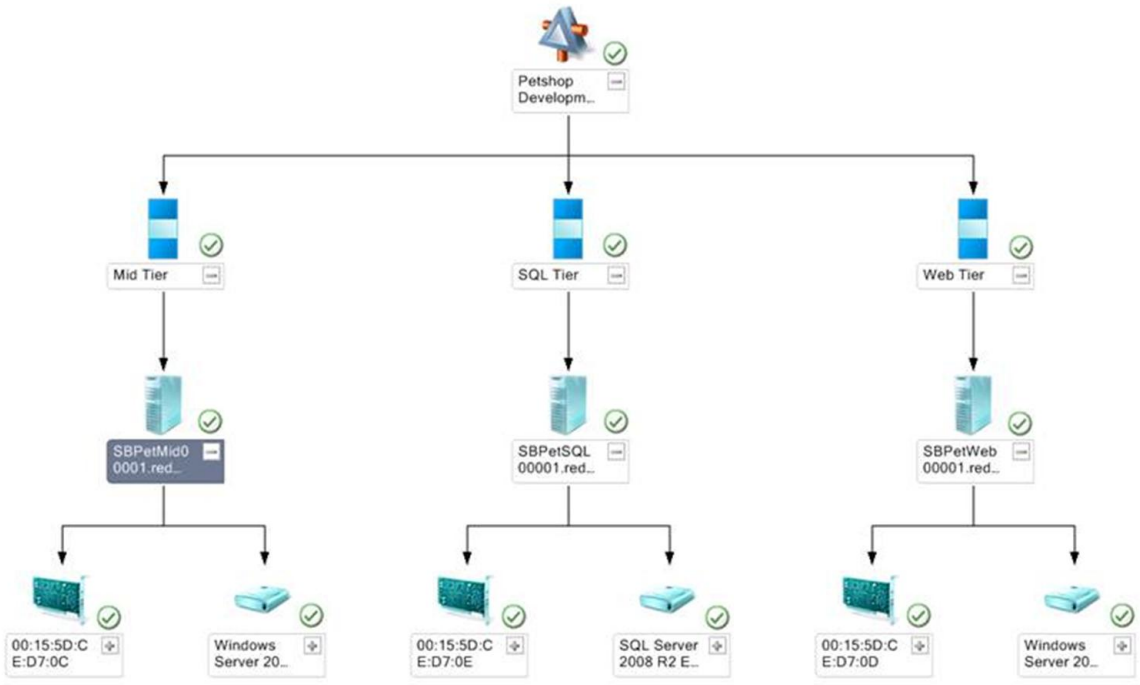

Figure 17: An application deployed from a service template is automatically made visible through the application monitoring capabilities of Operations Manager 2012.

As this example shows, an administrator can use this capability to examine the tiers in the application, look at their VMs, and more. If a new VM is added to or removed from a tier, this change also appears automatically in the Operations Manager 2012 view.

While all of this detail is essential for people who manage the infrastructure, it's too much for the private cloud's users. These users do care deeply about their applications, however. For example, suppose an application deployed using a service template is running too slowly. What's the problem, and who's responsible for correcting it? Answering these questions requires understanding what the application is doing.

To allow this, Operations Manager 2012 provides built-in support for detailed monitoring of .NET applications. For example, an application owner might keep tabs on traffic coming in and out of her application, monitoring things

like the end-to-end latency of a user request. If this latency gets too high, she can use Operations Manager 2012 to learn which method in which part of the application is taking the longest to execute. Armed with this information, an application owner can better determine whether the latency is caused by the application itself or by a problem in the cloud infrastructure. Since she can't directly access that infrastructure, being able to see application-level data is essential.

Another important aspect of monitoring is ensuring that service level agreements (SLAs) are met. Private clouds create a clear user/provider relationship between the infrastructure team and cloud users. SLAs are an important element of this relationship. Operations Manager 2012 supports this as well, providing information that can be used by both groups to track SLAs.

Using Private Clouds with Public Clouds

Both private and public clouds have value, and many organizations will use both. For organizations creating private clouds with System Center 2012, Microsoft provides two options for connecting to public clouds: working with a service provider that implements a cloud using System Center 2012 and using Windows Azure. This section looks at both approaches, beginning with Windows Azure.

Using the Microsoft Private Cloud and Windows Azure

Windows Azure provides a platform for Internet-accessible applications that run in Microsoft datacenters. Like a private cloud, it allows self-service access to an elastic pool of IT resources. Public clouds differ in some important ways from private clouds, however, and so even organizations with a private cloud might choose to build some applications on Windows Azure. Among the most common reasons to choose Windows Azure are these:

- Windows Azure runs in very large Microsoft datacenters located around the world. Because of this, it can be an attractive choice for an organization that wishes to build an Internet-scale application, especially one that will be accessed by users in many countries.

- Windows Azure provides true pay-as-you-go economics. While an organization's private cloud might charge an individual user only for the computing resources he consumes, the organization still owns the entire datacenter that provides this cloud—it bears all of the datacenter's costs all of the time. With a public cloud, however, a user stops paying entirely when he stops using the cloud's resources. This makes some scenarios, such as handling large but infrequent spikes in demand, more economical with public clouds.

- Windows Azure offers economies of scale that can sometimes make it less expensive than using an internal datacenter. This is likely to be even more true over time, as public cloud platforms become more widely used.

Public clouds also have drawbacks, of course. The most obvious is that data stored in Windows Azure is no longer in an organization's own datacenter, which can raise legal, regulatory, and security concerns. Yet because private and public clouds each have strengths and weaknesses, organizations should consider both. Going forward, the most common approach to cloud computing will almost certainly be some combination of the two.

Recognizing this, the self-service portal provided by App Controller 2012 can access both the Microsoft private cloud and Windows Azure. Figure 18 illustrates this idea.

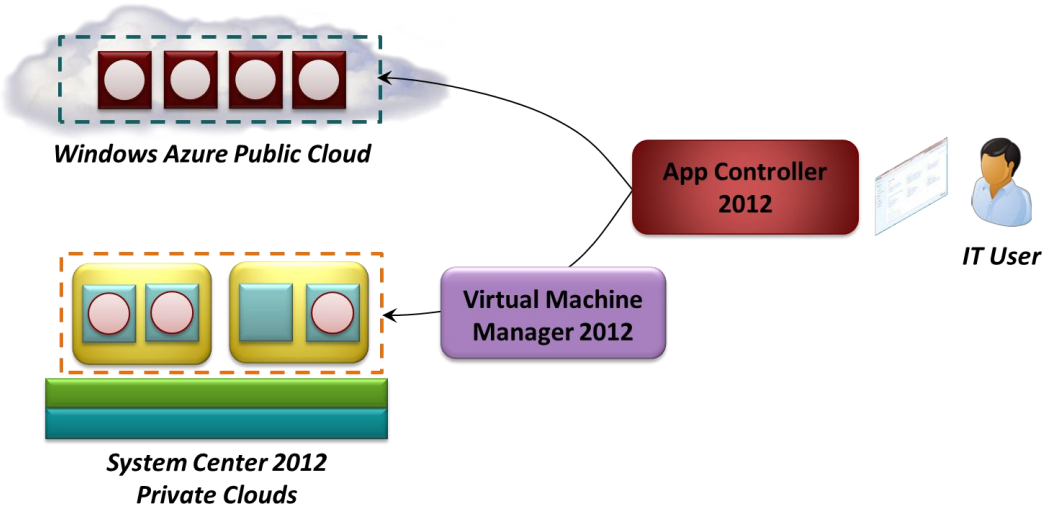

Figure 18: App Controller 2012 lets IT users access both System Center 2012 private clouds and the Windows Azure public cloud.

Through the App Controller 2012 user interface, an IT user can work with applications running on Windows Azure as well as VMs and applications in a System Center 2012 private cloud. For example, an administrator might use App Controller 2012 to manage a Windows Azure subscription or to check the status of a running Windows Azure application. He can then use the same interface to deploy a new application from a service template in a System Center 2012 private cloud. Both private clouds and the Windows Azure public cloud are displayed in a similar way, as Figure 19 shows.

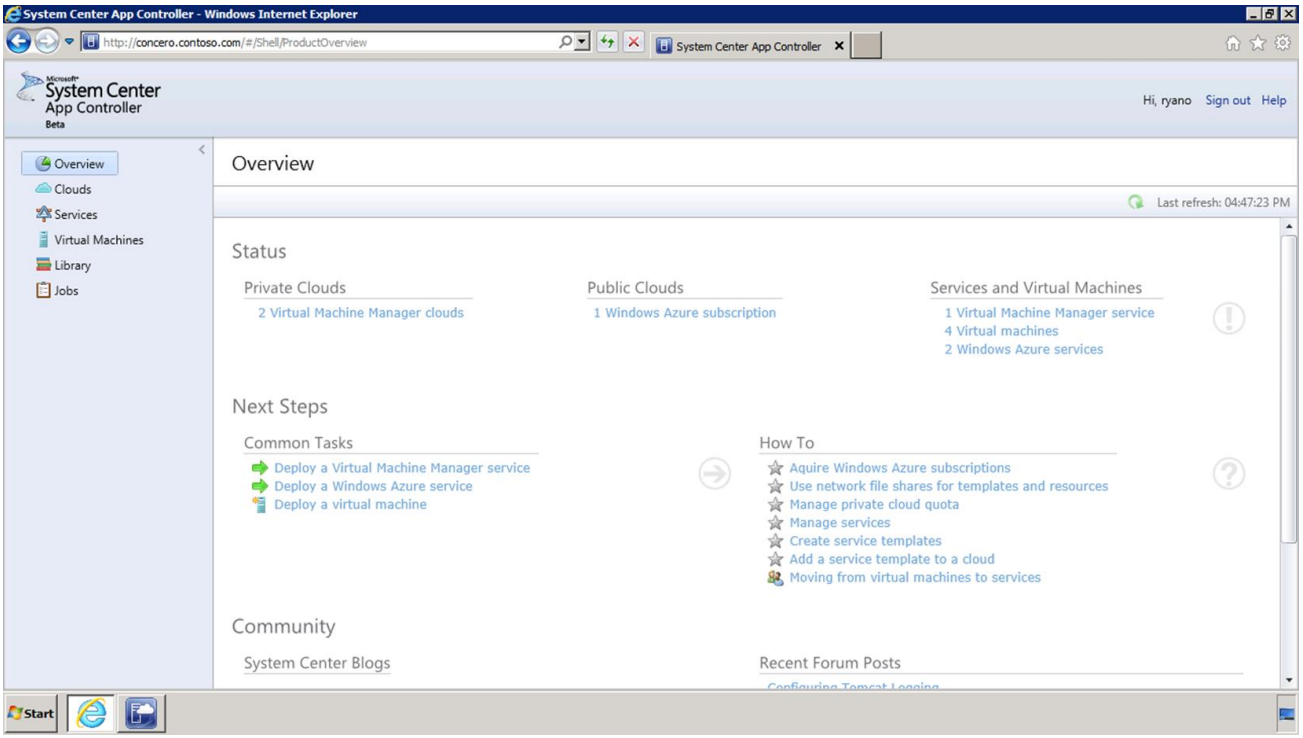

Figure 19: The App Controller 2012 user interface shows all clouds—System Center 2012 private clouds and the Windows Azure public cloud—that a self-service user has access to.

As this example shows, the App Controller 2012 interface gives a consistent view of private clouds created with VMM 2012 and Windows Azure public clouds. It also shows the services (i.e., applications) running on each one, together with VMs running in the private cloud. The interface provides straightforward access to common operations as well, such as creating a user role and deploying a VM in the private cloud.

It's important to recognize that the Microsoft private cloud and Windows Azure see applications differently. Both allow defining applications using a generalized service model, and both run the various components of an application in VMs. Both also provide scale-out, letting a developer or administrator increase or decrease the number of VMs an application is using, along with upgrade domains to minimize downtime during updates. But an application described with a VMM service template can't be directly deployed on Windows Azure, nor can a Windows Azure application be directly deployed on the Microsoft private cloud. Today, at least, the two models are different. Still, the two worlds share a good bit of technology, including development tools (such as Visual Studio), application infrastructure (such as the .NET Framework), and other things.

App Controller 2012 isn't the only link between the Windows Azure platform and the Microsoft public cloud. For example, Operations Manager provides a management pack that allows monitoring the health of applications running on Windows Azure. Using this option lets an administrator see both kinds of clouds using a common tool, something that can simplify life for organizations that use both.

Using the Microsoft Private Cloud and Service Provider Clouds

So far, the focus of this discussion has been on using the System Center 2012 technologies within an enterprise or public sector organization. This isn't the only option, however. The product suite can also be used by service providers (e.g., hosters) that wish to provide IaaS cloud services to their customers. To help them do this, Microsoft has established the *Hyper-V Cloud Service Provider Program.*

From a purely technical perspective, things work much the same with service providers as they do in a private cloud inside a single company. The big difference is that the cloud's users are now in a different organization than the people who provide the cloud's resources. This sharper divide between users and providers further increases the need for effective monitoring. Violating SLAs probably carries monetary penalties, for example, and the cloud is more likely to have explicit chargeback for its services. The technology is the same, but the business relationship necessarily changes with a service provider cloud.

It's also possible to link a private cloud within an organization with a cloud offered by a service provider. If the two clouds are domain joined, a single VMM 2012 instance can manage both. Domain-joined clouds also let users use App Controller 2012 to make requests across both environments. Whatever option an organization chooses, it's important to understand that the technology underlying the Microsoft private cloud can be used by service providers as well as within a firm's own datacenter.

Final Thoughts

Every organization wants to minimize its infrastructure costs, freeing up more IT dollars for innovation. Every organization also wants to make IT more responsive—taking days or weeks to provision a VM or deploy an application just isn't acceptable. Private clouds can help with both of these things. The technologies in System Center 2012, including Virtual Machine Manager, App Controller 2012, Service Manager, Operations Manager, and Orchestrator, let organizations create one or more private clouds. Working with multiple hypervisors over diverse hardware, these clouds can help lower costs and improve response times.

Server virtualization has proven to be a remarkably useful technology, saving money and improving service in enterprises and public sector organizations around the world. The next step on the virtualization road is to automate more of how we allocate, use, and update those VMs and the applications they support. The next step is private clouds.

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.