Realtime publishers

New Best Practices in Virtual and Cloud Management: Performance, Capacity, Compliance, and Workload Automation

Greg Shields

sponsored by vmware®

Copyright Statement

Chapter 4: Workload Automation in Virtualized and Cloud Environments

Properly constructed automation can be the virtual administrator's force multiplier.

There's no question that automation has become increasingly important in recent years. Much of that focus is a direct result of the sheer workload IT is expected to handle. There are simply too many servers, clients, and resources these days for each to be managed manually and individually.

You can argue that the increasing workload is a product of prior successes. As virtualization began enabling ever-faster creation of ever-more machines, IT excitement quickly turned to fear. That (incredibly warranted) panic centered on the effects of *virtual machine sprawl*. Suddenly awash in a sea of new computer instances, IT staff realized in hindsight that, "When a thing becomes easy to do, we do that thing."

Complicating those fears is the unbounded promise offered by public cloud computing providers. They tease, "Come to us when you have needs. Our resources are (effectively) unlimited." Happy words like burstability and elasticity get casually thrown around, while shallowly under the surface lie others more concerning: *consumption-based pricing* is one; *pay-as-you-go* is another.

The cloud indeed offers limitless resources, priced by the hour—or whatever pricing scheme you negotiate. But using those resources smartly requires some up-front intelligence. Are they even a good idea? When are they affordable? When will they break the bank?

Metrics are obviously the answer. Gathering those metrics are the very monitors we've discussed throughout the entirety of this book. Those metrics and monitors bring quantification to the variety of behaviors in a virtual environment. They illuminate intelligent decisions one should make when pondering what to do *with all these ever-increasing workloads*.

That said, metrics and monitors are but the intelligence. They provide the information. Actually making change in a measured and predictable manner requires a fourth new best practice: *workload automation*. Herein lies the key to bringing everything in your data center back under control: Not just any automation will do. The smart data center recognizes that intelligence and automation go together—*intelligent automation*. Asserting that control begins by letting the computers manage themselves.

Why to Automate

Few organizations consider virtualization without realizing they'll need some measure of automation to make it manageable and reach a greater level of efficiency and agility. That's an easy assumption, but it's only the first step. Look past mere virtualization and towards cloud management, and you'll quickly realize that automated *management* isn't the only driver.

Each chapter has introduced new management activities that tie into a black box approach (Figure 4.1). The first three chapters argue that performance, capacity, and configuration/compliance management benefit from the logic built into the box. The approach should now make sense. After all, the more your technology knows about itself, the better it is at managing itself. That's a fact that becomes increasingly important as a data center scales to the very large.

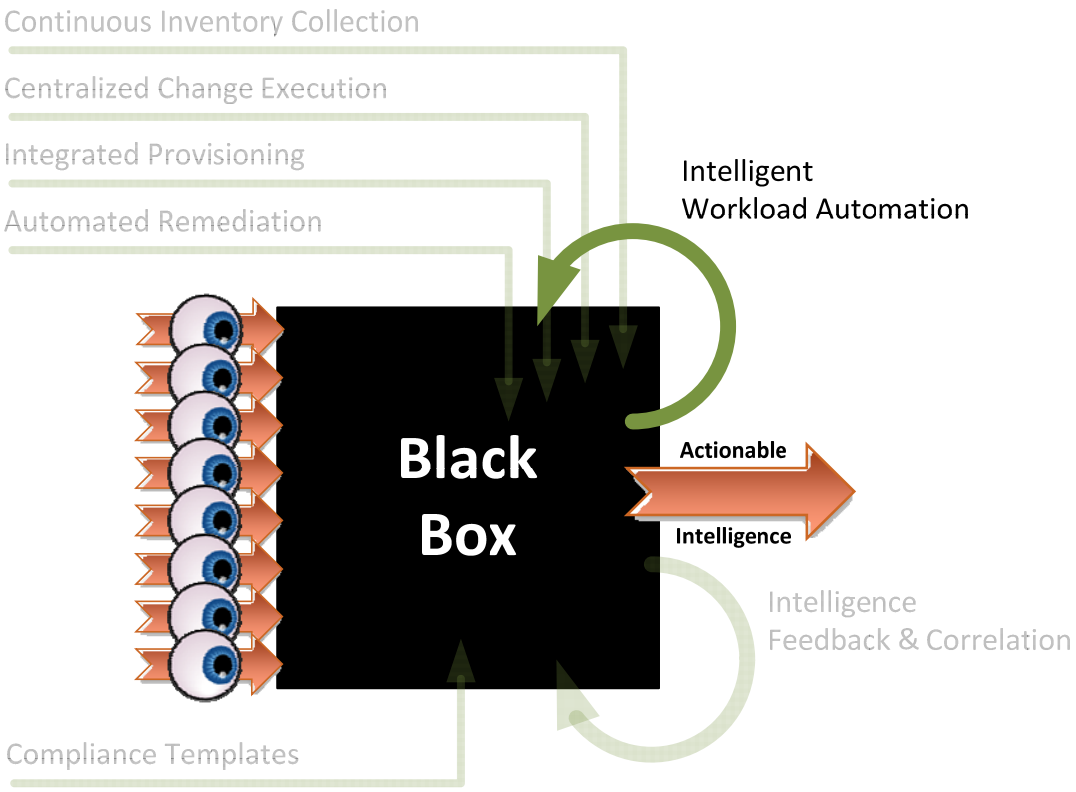

Figure 4.1: Intelligent workload automation.

This fourth and final chapter concludes the conversation by tying workload automation into the operations management experience. Critically important to recognize here is the *intelligence* the black box can add into each automation.

As organizations consider the tasks they've traditionally regarded as overhead, the term automation begins to take on broader meaning. In addition to day-to-day management, organizations begin to realize automated operations might be an option for other tasks they may have never considered. Here are three important terms: *provisioning*, *reprovisioning*, and *deprovisioning*.

Provisioning

Automations in the *provisioning* activity are perhaps the easiest to comprehend. After all, the point of a cloud management infrastructure is to accelerate the creation of new servers and services. What's important to recognize, however, is that provisioning in this context needn't be driven only by "creating new virtual machines." It can and often does also reference the ability to provision *entire services* on demand.

Doing so requires automations. But, as you've already learned, this method requires being smart about how those automations perform their tasks.

One critically important measurement is the impact of potential new servers and services on those already existing. The metrics captured inside the black box can ensure that creating a new service won't be an impactful event. You don't want to spin up 50 new high-load virtual machines in the middle of the day on a host running critical workloads. Automation enables a kind of situational awareness in places that are too complex to manage on your own.

Reprovisioning

The reprovisioning activity can be a bit less obvious, but is no less important. Reprovisioning focuses on the delivery of wholesale configuration changes that are the result of some other activity. For example, imagine your business suddenly falls under a new legal or regulatory requirement. That requirement must be immediately implemented across a large group of servers. Reprovisioning goes about updating those servers with the new requirement in a measurable and predictable way.

Workload Automation and Configuration Management Are Linked Activities

You might be asking yourself, "Isn't this merely configuration management with a different buzzword?" In a way, it is. Automating the delivery of new servers and services and later reprovisioning updates all involve changing an environment's configuration. Thus, configuration management is indeed involved.

That said, the *actual* act of *actually* delivering each change can be accomplished through a variety of means. You can click buttons manually. You can script the change. You can leverage tools that automate change delivery for you.

Workload automation strives to accomplish that delivery without requiring cumbersome and manual effort. As you'll learn in a minute, the new best practice leans on tools to assist.

Deprovisioning

A third activity is deprovisioning, which focuses on the activities in removing servers and services from an environment. This task might seem simple, but deprovisioning correctly can require far more planning than one might think. Indeed, you must occasionally delete a virtual machine when it is no longer relevant. Where workload automation becomes critical is when that decommissioning *needs to happen automatically and based on predetermined environmental conditions or as a course of life cycle management*.

An example can help here. Consider the situation where some IT service requires one or more Web servers. A single Web server might be necessary when user load is nominal. More than one Web server might become necessary when user activity exceeds a certain threshold. Later on, when activity returns to nominal, those extra Web servers are no longer necessary.

Workload automation enables IT to provision preconfigured Web servers to meet the increasing user load situation. It further enables the deprovisioning of those servers the moment they become unnecessary. This job is a configuration task, *but it is also a monitoring task*.

The key capability driving these decisions is the intelligence built into the black box. It monitors for performance, so it is aware of available capacity. That gives a well-constructed automation the intelligence it needs to deploy (and later decommission) the necessary resources.

It is worth restating here, "When a thing becomes easy to do, we do that thing." Workload automation in today's virtual and cloud environments has evolved past merely speeding up patch deployment or updating a few user accounts. The new best practice seeks to impact every aspect of a workload's life cycle, from creation through ongoing management and all the way to end-of-life decommissioning.

How Does One Automate?

You need to ask yourself, "I get it. How then do I automate?" The answer depends on your needs and the solution you're using.

Part 1: Scripting

Scripting and the use of scripts have long been the go-to approach for automation. In recent years, Windows PowerShell has evolved to become a primary script environment. VMware's vSphere offers robust scripting support via its PowerShell-based PowerCLI toolkit. Microsoft's Hyper-V uses it, as does Citrix's XenServer to a lesser extent.

Kits such as these enable administrators to quickly create scripts that perform a variety of tasks. Those scripts take time to create, debug, and perfect, but that time is commonly viewed as an investment. The scripter expects that the task once automated will take less subsequent manual effort to perform: *pay now, benefit later*.

At the end of the day, scripts *are* code. They're vastly extensible, but they can be a pain in the neck to generate. The code that follows (see Figure 4.2), for example, is a small portion of a much larger PowerShell script. In it, the New-VM cmdlet creates a virtual machine. The configuration of that virtual machine is based on the list of parameters that are supplied. Once created, the script's next cmdlet—Start-VM—then powers on the newly-created virtual machine.

```
New-VM -Name $VMName -OSCustomizationSpec $Customization -Template $Template -
VMHost $VMHost -Datastore $Datastore -RunAsync
Start-VM -VM $VMName -RunAsync
```

Figure 4.2: An excerpt from a PowerShell script.

There's incredible power here, but there can also be incredible problems. Getting value out of scripts first requires a significant investment in scripting. Administrators with the right expertise are also required, and not *every* administrator has the aptitude or the interest.

Pressing on through and learning from failures is a further requirement. Administrators without formal software development experience can generate code that is less maintainable, less robust, and/or less reliable than it needs to be. Their efforts are less reusable—less modular, which reduces the return on investment in creating scripts.

Scripts have another downside, too, in that they don't tend to be designed for accessibility by anyone other than a skilled administrator. Often, they're meant for use exclusively by the person who wrote them in the first place. As a consequence, they're often written with unavoidable quirks or compromises that limit their effectiveness in true automation. Poorly-constructed scripts might have to be manually run and monitored by that skilled individual or are not parameterized to a level where their reuse is cost effective.

Essentially, a poorly-conceived script isn't "shrink-wrapped," in that others can't just pick it up off a shelf and use it elsewhere.

Scripting's Biggest Problem: The Scripter?

A decade ago, I was a systems administrator for a large defense contractor, and was known for finding automation solutions for many of my job's manual tasks. The scripts I created worked great while I was responsible for them. They created problems after I left that employer. Many "automated" tasks were in fact completed by a small piece of code that no one else knew existed—until one day it ceased to function. I still get calls every so often when one of my little buried automations stops working.

In the end, I might have created more problems over time than I solved. Luckily, my employer still knows how to find me. Not every business is so lucky.

Most importantly, most scripts also lack intelligence. Although it's easy with a modern hypervisor to script the creation of a new virtual machine, there's more to automation than merely performing the task. When should the task be performed? On what hosts? For what reason? What happens next? Although a script can certainly be programmed to answer these questions—reaching out and checking schedules, verifying capacity, measuring current workload, and so forth—the sheer number of decision and data points can mean rapidly scoping a simple script into a major development effort.

Part 2: Objects and Runbooks

So if scripts are good but scripts are bad, then what's an automation-seeking virtual environment to do? One approach is to create an "overarching management solution" that focuses its energies on reusable management objects. Although each object is really a script at heart, its creation as an object needs to ensure that it possesses the necessary input, output, and processing components that facilitate its use within a greater framework.

Figure 4.3 shows an example of how three objects can be collected to enact a change. These objects don't necessarily eliminate the scripts themselves. Rather, they encapsulate them into specific units of functionality within the context of an overarching management solution. The Measure-Performance and Validate-Capacity functions in Figure 4.3 represent those functionality units.

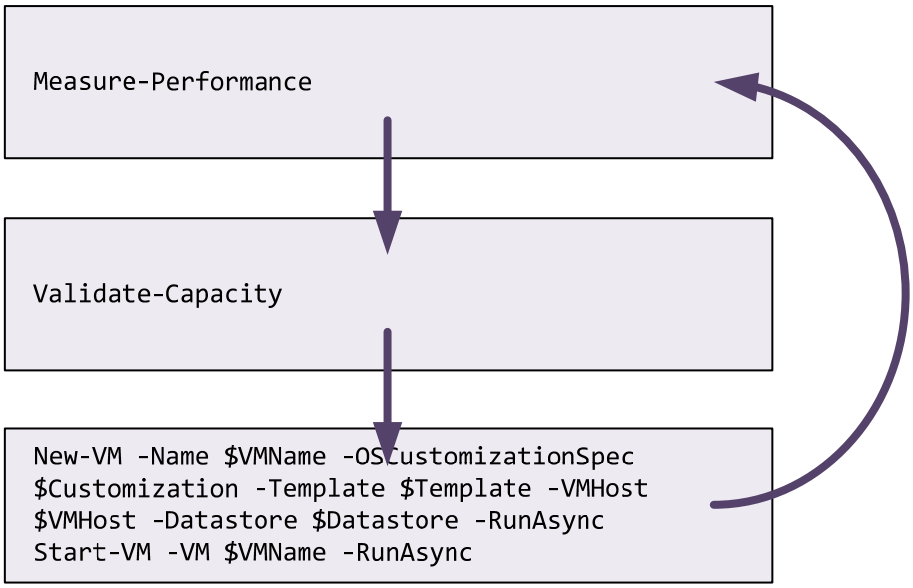

Figure 4.3: Wrapping scripts into management objects.

This approach aids in making scripts more production-friendly and more reusable. It lets organizations treat scripting automation as *a solution* rather than a series of break-fixes. The scripts themselves gain as well. Those generated within a larger framework tend to be more structured, more reliable, and more maintainable over time.

The framework has a name as well: *runbook automation*, or RBA. You can think of RBA as the environment in which all these automations are orchestrated (e.g., constructed), scheduled, and tested. The results of their actions can be reported on. Most importantly, the overarching RBA solution can take cues from the black box intelligence to help make more-informed decisions.

That situational awareness lets RBA tools be used as remediation systems for specific problems. Once tied into monitoring (Figure 4.4), a poor performance condition can trigger an RBA action to remediate the problem and alert the administrator.

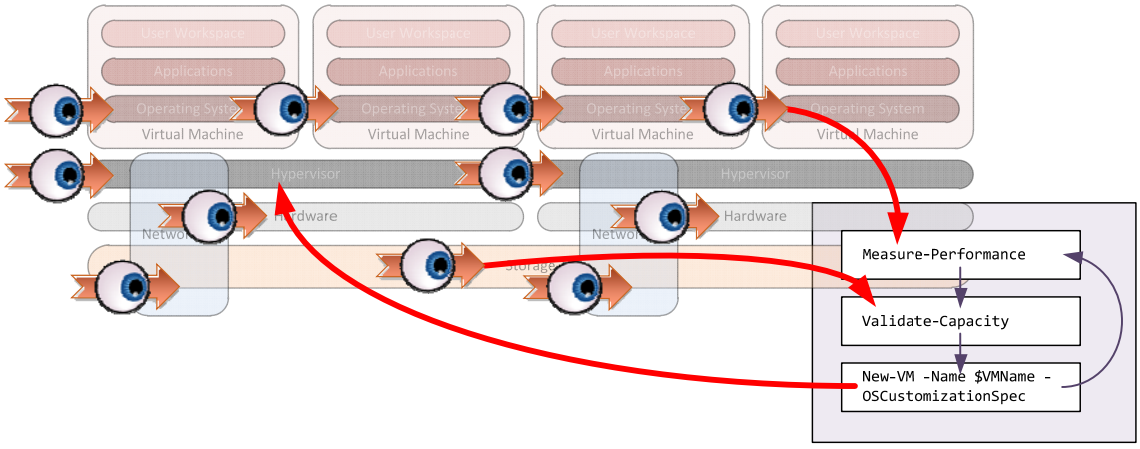

Figure 4.4: Integrating monitors into automations.

A key benefit of an RBA framework is that it needn't require specific functionality built into the technology it manages. For example, as long as the resource you're attempting to automate has scripting exposure of some form, the resource can be acted upon by objects in the RBA solution. Less-mature resources can be provisioned, managed, and de-provisioned through RBA tools, without needing any specialized automation "hooks" inside the resource itself.

RBA can very obviously facilitate back-end administrative activities, but smartly-designed ones can also expose those activities to front-end users. Herein lies the advantage in self-service. With enough intelligence built into the system, the provisioning, reprovisioning, and deprovisioning tasks become exceptionally well-suited for RBA (and, thus, self-service) execution. You can see why self-service—with the right user in mind—is quickly becoming the new best practice.

An Aside: Private Cloud to Hybrid Cloud

It is worth pausing this conversation for a quick tip-of-the-hat towards automation's role in managing resources that sit outside the local data center. An automation framework is useful for automating workload activities in a private cloud, but it is arguably more important when considered in the context of a hybrid cloud.

A key characteristic of hybrid cloud computing is burstability, which is the ability to augment and extend services wherever and whenever necessary. The public cloud portion of hybrid cloud is by definition a (virtually) unlimited pool of resources. Your cloud provider relationship makes available a reasonably unending capacity of resources that you can consume when your needs require.

Those resources don't come inexpensively, nor are they priced in ways that are familiar to traditional IT. Hybrid cloud computing's central hurdle is arguably its pay-as-you-use costing model. This model does wonders for eliminating capital expenditures and reducing the impact of unused resources, but it can be painful on the monthly bill when eagerness exceeds actual usage.

The problem is that data centers tend to always need more: more resources, more machines, and more services. Left unchecked, this tendency towards *always more* can quickly create a cost problem in the pay-as-you-use public cloud. One counters that problem with the resource usage quantifying assistance of the black box, or, in plainer English, "You've got to know what you're using if you're to know what it costs."

Another key aspect of a hybrid cloud environment is the seamless flexibility that exists between local and non-local resources. Figure 4.5 shows a representation of various workloads that could be burst into a connected provider's public cloud.

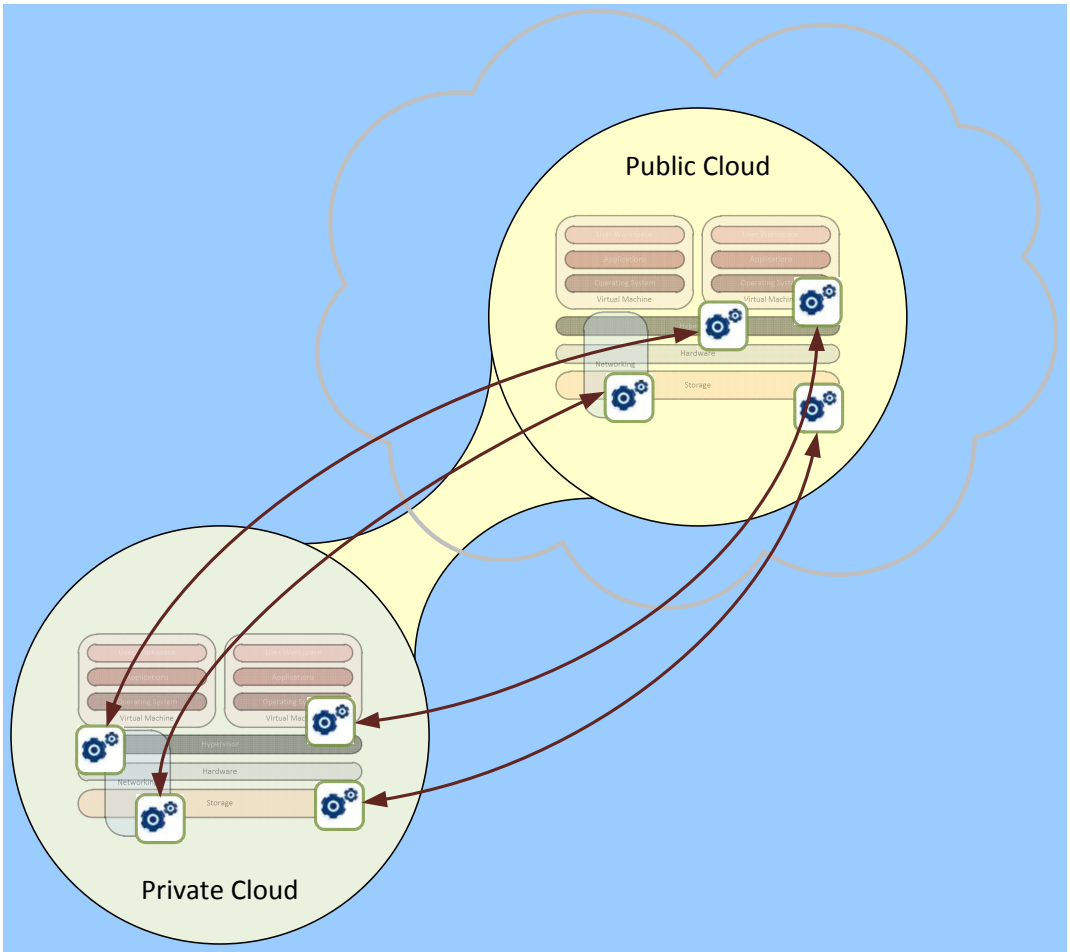

Figure 4.5: Workload automations extend into the hybrid cloud.

These workloads include activities such as:

- Live migrating virtual workload to free local resources
- Spinning up additional resources when demand dictates
- Powering down resources when demand decreases
- Relocating workloads when their usage patterns make more sense in the public cloud space

Important to recognize here is *the connection* between the private and public cloud entities. That connection facilitates the backward-and-forward flow of workload processing as demands change. A virtual machine that starts its life in the private cloud might later be better positioned in the public cloud (or vice versa) due to the conditions of the day. Alternatively, an IT service designed for the private cloud can burst onto public cloud resources when its user load increases.

Control of these activities happens via the very workload automations this chapter has focused on. Those automations facilitate the work in provisioning, reprovisioning, and deprovisioning resources irrespective of where they ultimately reside. What results is a kind of extended capacity up in the public cloud (Figure 4.6) that's available for when you need it.

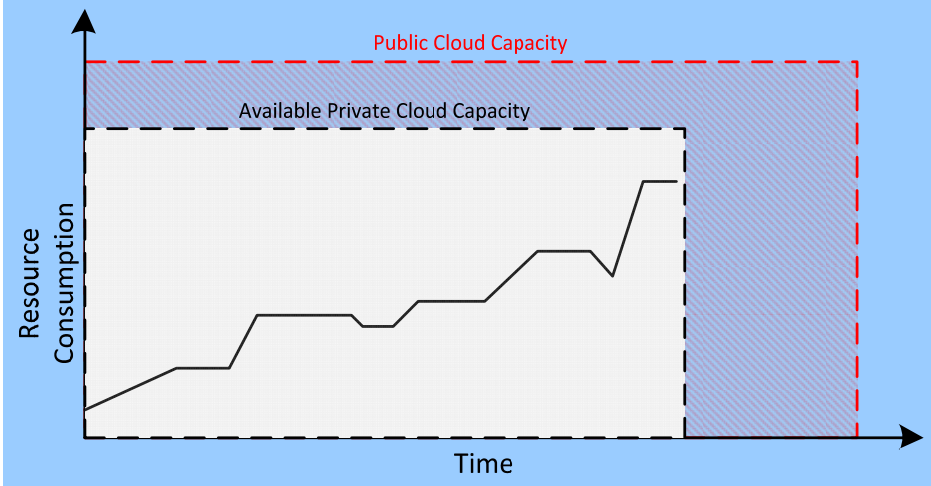

Figure 4.6: The public cloud delivers another pool of resources.

Part 3: Policies and Autonomous Management

Policy-based administration is generally held to be more automated, more proactive, and more desirable than standalone scripts or even RBA tools. In a typical policy-based administration setup, administrators define groups of desired configuration settings. The technology being administered—a hypervisor, for example—is configured by its vendor to read and understand these policies, and to *configure itself* to ensure their compliance.

This model is significantly different than scripting and RBA. Rather than giving the hypervisor the steps to reconfigure itself, you simply configure it with *the end configuration state you desire*. Armed with that information, it is then enabled to perform whatever is necessary under the hood to meet the requirement.

The policy-based approach offers certain distinct advantages, particularly in large and well-controlled environments. In these situations, when a desired configuration changes—such as some new operational requirement—one simply alters the top-level policies. Your resources can then adjust themselves accordingly. When paired with an enterprise-ready solution, the approach can scale to the near-infinite levels one expects in a public and/or hybrid cloud environment.

Important to recognize is that policies aren't scripts, nor are they configuration objects like those used in RBA. With policies, you're not running a script against 10,000 machines. Rather, you're communicating a new configuration target to those machines, then letting them intelligently reconfigure themselves.

Policy-based administration works well for static configuration items such as security settings or broad operational parameters. Policies can be used for *automation,* as well. For example, a policy-enabled hypervisor can work with policies that define contingent actions, such as what to do in various failure scenarios. They can also define instructions for when a host's resource consumption exceeds its capacity. Policies enable a virtual platform to automatically respond to operational conditions without having to wait for monitoring to notice the problem and alert an administrator who then runs a script or runbook to remediate the situation.

As you can imagine, a policy-ready hypervisor has to be a *smarter* hypervisor. That hypervisor must understand the impact of workloads, their resource usage, and their importance to your business operations. That hypervisor must understand something of a workload's history as well as its configuration. Armed with this intelligence, the underlying system can know what the workload *is intended to do*. It can then make smarter decisions—guided by policies—at every point in time. What this setup creates is an environment of *autonomous management,* which is arguably the ultimate form of automation.

Autonomous Management: An Example

Autonomous management might sound like a far-fetched concept, but its foundations are merely another approach in combining intelligence with actions. For example, suppose a virtual host recognizes that it is reaching capacity. It knows this because it has been configured to monitor for this exact situation. It also knows to analyze its list of policies to match what it sees with what it should do.

Your policies might prioritize demand so that highly critical workloads are left alone if at all possible. Other resources may be migrated to another host. Very low-priority workloads might be suspended entirely until the situation has resolved.

All of these actions can be initiated automatically because they are triggered by recognizable behaviors. By merely classifying workloads, one can implement policies to manage those workloads more effectively. Workloads with changing priorities can also be easily reclassified. Has a certain workload become more mission critical than it was in the past? No need to reprogram a bunch of scripts. Simply reclassify the workload and let policies handle the rest.

Part 4: Self-Service

The entirety of this conversation is designed to lead towards a fourth and final automation: *self-service*. The notion of self-service involves moving certain tasks from the responsibility of the virtual administrator onto the requesting individual. Self-service has gotten significant attention lately due in part to the increasing complexity and scale of many virtual and cloud environments. Simply put, IT environments are becoming so large and so well-automated that it can begin to make sense to *let users manage their own resources*.

Self-service needn't focus exclusively on servers. With the right automations in place, one can offer self-service for entire services that users need (Figure 4.7). Such self-service needn't necessarily expose the entire gamut of administrative controls, merely those that make sense for the self-serving user.

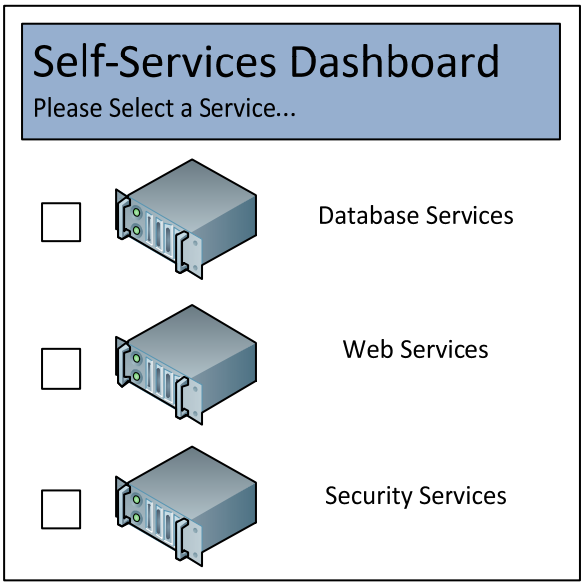

Figure 4.7: Self-services can be servers, but it can also be entire services.

Who Is the Self-Server?

For many in IT, the notion of *self-service* raises the hairs on the back of the neck. To this group, their understanding of self-service runs counter to the charter of the IT organization: managing computing assets.

What's interesting is that these people might be incorrectly considering self-service's end user. In many cases, that end user won't be the regular, run-of-the-mill person who happens to need computer resources to accomplish a job. It can be, but the everyday user still today relies on IT to actually manage the services they consume.

The intended user of self-service in most situations is usually *someone else in IT*. That's because self-service's automations directly align with a growing trend in IT.

> Consider for a minute your IT organization. You probably have people who manage the data center infrastructure. Others manage the applications atop that infrastructure. Self-service enables the first group to automate the delivery of computing resources to the second. It is a natural extension of what most IT organizations already do today: Someone needs a resource; a different person gives them that resource. *Self-service merely eliminates extra work for the middleman.*

Key to this offering is the creation of a *bounded experience.* One obviously can't allow (even trusted IT) individuals to start spinning up virtual machines whenever they need. Their experience requires controls: Control over *when* that happens, *where* the workload is hosted, *how many* resources they're allowed to consume, and so on.

These controls become the logical extension of self-service's separation of duties. The virtual administrator maintains environment performance and capacity; the requestor works within the boundaries they're given. Maintaining those boundaries happens via the same data being collected by the black box. With the right tools in place, the black box and the infrastructure itself helps to provide those boundaries. And, again with the right tools in place, freeing designated users to manage their own resources is absolutely becoming the new best practice.

The Right Tools

You can probably surmise by this point that the thesis behind each of these new best practices is that *you can't effectively do this unaided*. Even with all their benefits, today's virtual environments have become just too complex to recognize greatest benefit without assistance. Odds are good that you've already implemented your hypervisor of choice along with its management platform; odds are also good that management platform isn't enough. Also needed are the additional services one gets from the extra tools that integrate performance, capacity, and configuration and compliance management with workload automation.

Further, if you've struggled with understanding your virtual environment's fit into the greater cloud story, your challenges might be directly related to exactly those services you're missing. Effective tools like those you've learned about in this book are becoming ever-more necessary to first help you understand that fit, then take advantage of everything the cloud has to offer.