OpenStack in Action
by V. K. Cody Bumgardner

Chapter 9

brief contents

9 Architecting your OpenStack

This chapter covers

- Using OpenStack to replace existing virtual server platforms
- Why you should build a private cloud
- Choices to make when building your private cloud

In the first part of this book, you dipped your toes into OpenStack through the use of DevStack. The purpose of that part was to introduce you to the hows and whys of OpenStack and to pique your interest in a deeper understanding of how things work under the covers.

In the second part of the book, you undertook a manual deployment of several core OpenStack components. Although it's important that you understand the underlying component interactions that make up OpenStack, the second part of the book isn't a blueprint for OpenStack deployment. This level of understanding builds confidence in the underlying system through low-level exposure to the components and configurations, but isn't intended to encourage you to manually install components in a production environment.

This third and final part of the book covers topics related to deploying and utilizing OpenStack in production environments, specifically enterprise environments where the typical systems administrator might take care of a wide variety of both infrastructure and applications. Often in enterprise environments, systems engineers drive infrastructure design, deployment, and adoption. The chapters in this part of the book are intended to help you develop a successful OpenStack deployment for your environment.

This chapter covers decisions—architectural, financial, and operational—that you'll need to make as you plan your deployment. This chapter isn't a cookbook, but rather a starting reference for use when developing a successful architecture. For a prescriptive approach to developing your architecture, consult the "OpenStack Architecture Design Guide" (http://docs.openstack.org/arch-design). Once you determine the type of OpenStack deployment that's right for you, the online design guide can be a valuable asset for configuration and sizing.

Many enterprise systems people will approach OpenStack from the perspective of traditional virtual and physical infrastructure platforms. We'll first discuss using OpenStack as a replacement for existing virtual server platforms, covering strategic design choices that you need to make to get the most out of your OpenStack deployment.

9.1 Replacement of existing virtual server platforms

In the 2015 Gartner report "Magic Quadrant for x86 Server Virtualization Infrastructure," it was estimated that 75% of x86 workloads were virtualized, with VMware as the predominate enterprise vendor.[1] This section covers how OpenStack can be used as either a replacement for or augmentation of your existing VM environment. In addition, the section makes a case for thinking of OpenStack as more than a replacement for a traditional virtual server platform.

Likely your traditional virtual environment was designed to provide virtual machines in a way that operationally mimics physical machines. There's also a good chance that virtualization was introduced into your environment as an infrastructure cost-savings measure for existing workloads. As workloads were moved from physical servers to their virtual equivalents through a process known as Physical to Virtual (P2V), an exact clone of the physical server was made by the P2V tool. More often than not, physical and virtual servers operated on the same networks, which allowed the P2V migration to take place without service interruption. For many environments, the process of consolidating workloads on virtual servers resulted in significant financial savings. In addition to resources being used more efficiently, new capabilities like master images and virtual machine image snapshots became part of the software deployment and upgrade processes, mitigating many types of software and hardware failures. Despite many of the new capabilities that virtual environments offered users, system

[1] See Thomas J. Bittman, Philip Dawson, Michael Warrilow, "Magic Quadrant for x86 Server Virtualization Infrastructure" (14 July 2015), www.gartner.com/technology/reprints.do?id=1-2JGMVZX&ct=150715.

administrators were still managing both the OS and application levels of virtual machines, much in the same way they had physical machines.

If your intent is to treat your virtual environment the way you do a physical environment, as described in the previous paragraph, then the benefits of OpenStack in your environment will be limited. This is to say, if your operational practice is to manually deploy virtual services through a central group without automation, you must evaluate how cloud frameworks such as OpenStack can be effectively adopted in your environment.

Suppose you've been using VMware vSphere as your server virtualization platform, and you're interested in adopting OpenStack as a VMware replacement for cost-reduction purposes. If you think of OpenStack as simply a "free" alternative to VMware, then you might be heading down the wrong path. Although in most cases OpenStack can be deployed in a way that provides feature parity with many competing virtual environments, this has to be done in a way that's compatible with your operational practices. Consider again the previous VMware replacement example, where you wanted to move all existing VMware workloads to OpenStack. Although OpenStack Storage can deal with VMDK (VMware image format) files, there's no graphical Virtual to Virtual (V2V) migration tool like what's provided by VMware, and there probably shouldn't be.

Now, consider the process most often used to build VMware-based machine images. Typically, using a desktop client, a user virtually mounts a CD or DVD from their workstation to virtual hardware and performs an install as they would with a physical machine. However, the ability to remotely mount CD and DVD images doesn't exist on the OpenStack Dashboard. One shouldn't consider these things as an indication that OpenStack is incomplete, but as an indication that it's intended to be used differently than traditional virtual server environments.

Where might OpenStack be a good replacement for VMware and other commercial hypervisors? To answer this, you must first consider strategically how you want to interact with your infrastructure resources. Table 9.1 lists the possible impact of OpenStack based on your infrastructure management strategy.

Table 9.1 OpenStack impact based on environment

Environment	Description	Impact
Siloed and manual VM	Hardware management is siloed and resources are shared. Virtual hardware is manually provisioned by IT staff to end users, where end users are responsible for OS-level operation.	Low to negative
Siloed and automated VM	Hardware management is siloed and resources are shared. Virtual infrastructure is deployed using automated methods by IT staff, where end users are responsible for application-level operation.	Medium

Table 9.1 OpenStack impact based on environment *(continued)*

Environment	Description	Impact
Application-specific backend	Hardware is dedicated and managed by the cloud framework. Infrastructure and application deployment are automated by IT staff for a specific application.	High to very high
Private cloud	Hardware is dedicated and managed by the cloud framework. Applications and standard (size and OS) VMs are provided to end users using automated self-service methods.	Very high

In the case of the *siloed and manual VM* listed in the table, there's a central group manually provisioning virtual machines without automation, and they will likely view OpenStack as either incomplete or unnecessary. Without the addition of automation, OpenStack doesn't necessarily provide them with anything that can help them do their current jobs.

The *siloed and automated VM* environment is similar to the manual environment, with the exception that at least some level of infrastructure orchestration is being used by the IT department managing the infrastructure. For instance, departments that make use of dynamic automated provisioning as part of a request workflow are considered part of this group. Much like their manual counterparts, organizations that fall into this category often evaluate OpenStack as a direct cost-saving replacement for whatever they are currently using. Although it's true that OpenStack can lead to cost savings, the operational and business processes of these organizations must change in order to take full advantage of the framework.

Now, suppose the central group manually provisioning the virtual machines is repositioned as infrastructure or application resource consultants, as in the *application-specific backend* scenario.

Suppose that through this strategic shift, not only is automation used in the infrastructure, but it's also used for application-level provisioning. Suppose further that resources are allotted to tenants, and departmental-level personnel are able to provision their own resources. That would be the *private cloud*, in which the central group is enabling departmental agility by brokering services, not prescribing them. In many respects, operating in this way allows you to change your thinking about the role of infrastructure in your environment. Automation on application and infrastructure levels removes the need for P2V and V2V tools, so there's no need to move images around. In this mode of operation, infrastructure resources are more transient and function more as an application capability than a static allocation.

The real value of OpenStack is in the automation and platform abstractions provided by the framework. The following subsections will discuss architectural considerations that you must take into account as you develop your OpenStack design.

9.1.1 Making deployment choices

If you're used to supporting virtual server platforms like VMware vSphere or Hyper-V, you might need to rethink how you purchase and support hardware. Although it's less common than in the past, physical resources in the enterprise, like network switches and central pools of storage, can be shared between physical and virtual resources. Even if a resource is exclusively assigned to a virtual server platform, you'd typically think of provisioning resources to be used by the platform, not of the platform itself managing the resources. For instance, it's common to assign a new VLAN or to make a shared logical unit number (LUN) available to a group of hypervisors. But if you need to create new VLANs or new shared LUNs, the administrators of those systems would need to go through their own provisioning process. It's also very common for the "network person" to do all network configurations, the "storage person" to do all storage assignments, and the "VM person" to tie the resources together with a physical server to produce VMs. Each person in this process has to perform manual provisioning steps along the way, often without understanding how their resource plays into the complete infrastructure.

Deploying OpenStack from slivers of shared central infrastructure is generally the wrong path to take. OpenStack detects, configures, and provisions infrastructure resources, not the other way around. Even if your shared central infrastructure provides multi-tenant (not to be confused with OpenStack tenants) operation, which would isolate OpenStack automation, you'd still need to consider the effects of making OpenStack-provisioned resources dependent on shared resources. For instance, software upgrades for reasons outside of OpenStack operations could impact services. In addition, resource utilization outside the scope of OpenStack resources could impact performance while providing no indication to OpenStack services that problems exist.

In many cases, virtual environments aren't designed to leverage the benefits of an infrastructure that can be managed programmatically. In these cases, the operational practices will have been developed around vertical management of siloed resources like compute, storage, network, load balancers, and so on. In contrast, OpenStack was designed with the complete abstraction of physical infrastructure in mind. In general, you can save yourself lots of trouble by assigning resources exclusively to OpenStack, and through plug-ins and services letting the framework manage resources, instead of the other way around.

In the following subsections, it's assumed you wish to augment or add new services using OpenStack to manage your resources. In your environment, you want to leverage the management capabilities of OpenStack, and you even want OpenStack to manage your hardware, but you want the end product to resemble what you are providing now. Specifically, you're willing to change your operational and deployment practices for the sake of efficiency, but your primary interest is to deploy VMs, just as you might be doing now with VMware or Microsoft. Section 9.2 discusses taking a more progressive approach to IaaS.

9.1.2 What kind of network are you?

If you want to take advantage of OpenStack, but you don't want OpenStack to manage L3 services such as routing, DHCP, VPN, and the like, you must evaluate your options based on management of L2 (switching) services.

For example, figure 9.1 shows a VM directly connected to a public L2 network. This example isn't specific to OpenStack and would be representative of similar network deployments for many virtual server platforms, including VMware vSphere and Microsoft Hyper-V. In this network deployment scenario, the job of the hypervisor is to direct L2 network traffic to a switch, which is typically a physical switch outside the control of the hypervisor. Unlike many of the network examples in this book, there's no concept of an "internal" or hypervisor network, because no L2 services are being provided by the virtual server platform. In this deployment type, all L3 services are provided by systems outside the hypervisor. As you might imagine, separating the majority of network services from the virtual server platform limits the benefits of the platform, but the simplification isn't without benefit. Based on your IT strategy, existing resources, and support structure, this mode of operation might be best for you.

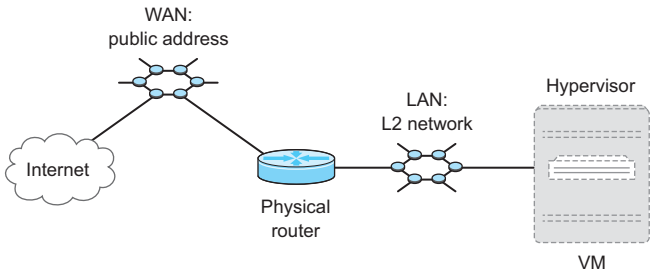

Figure 9.1 L2 network with VM and hypervisor

The majority of this book has focused on OpenStack Networking (Neutron) providing L2 and L3 services. As discussed in previous chapters, Neutron was built to manage complex networks and services within OpenStack environments, not simply to push L2 traffic to external networks. But the OpenStack Compute (Nova) project, which predates Neutron, does provide basic L2 services. If you want to limit your OpenStack deployment to L2 services only, you'll want to use Nova for networking, not Neutron.

Network hardware

If you're using Nova for networking, there's very little reason to worry about the integration of OpenStack with network hardware. Early on in the OpenStack project, hardware vendors were writing drivers for Nova integration in their hardware, but most development has since moved to Neutron.

A Nova network can operate in three different topologies: flat, flat DHCP, and VLAN.

In a *flat* topology, all network services are obtained externally from OpenStack. You can think of a flat topology working the same way as your office or home network

connection works. When you connect your computer to the flat network, your computer relies on the existing network for services such as DHCP and DNS. In this mode of operation, OpenStack is simply connecting VMs to an existing network, just as you would a physical machine.

The *flat DHCP* topology is similar to the flat topology, with the exception that OpenStack provides a DHCP server to assign addresses to VMs.

The *VLAN* topology works in the same way as the flat topology, but it allows for *VLAN* segmentation of the network based on *VLAN ID*s. Simply put, in the flat network all VM traffic is sent to the same L2 network segment, whereas with the VLAN topology, you can assign a specific L2 network segment to a particular VM.

The next section covers choices in storage.

9.1.3 What type of storage are you?

If you're coming from a traditional virtual server platform environment, you may have no management integration between your hypervisor and storage subsystem. If you're using VMware vSphere, you'll typically attach large shared host volumes to your hypervisors, as shown in figure 9.2 SharedVolume.

In the figure, you can see a single host volume shared across all hypervisors. A shared host volume is formatted with a cluster-aware filesystem that allows hypervisor A to use the same underlying host volume to store data for VM A, as hypervisor B would for VM B. If VM B was to be migrated to hypervisor A, no stored data would need to be transferred, because the data is already accessible by hypervisor A. In this scenario, VM volume management is done on the shared-host volume level, so from the

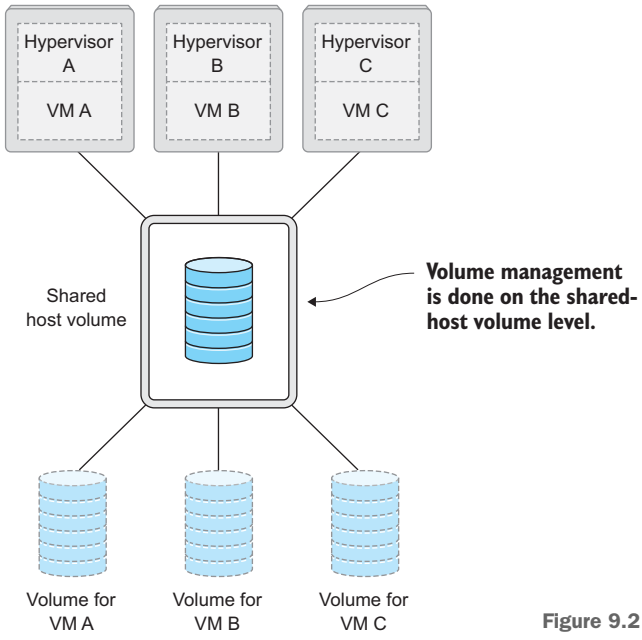

Figure 9.2 Shared volume across hypervisors

standpoint of the underlying storage subsystem, nothing is managed beyond the large shared host volume attached to the hypervisors.

In contrast, Microsoft Hyper-V promotes a "shared-nothing" model, where each hypervisor maintains its own storage and the storage for its VMs. An independent host volume model is shown in figure 9.3.

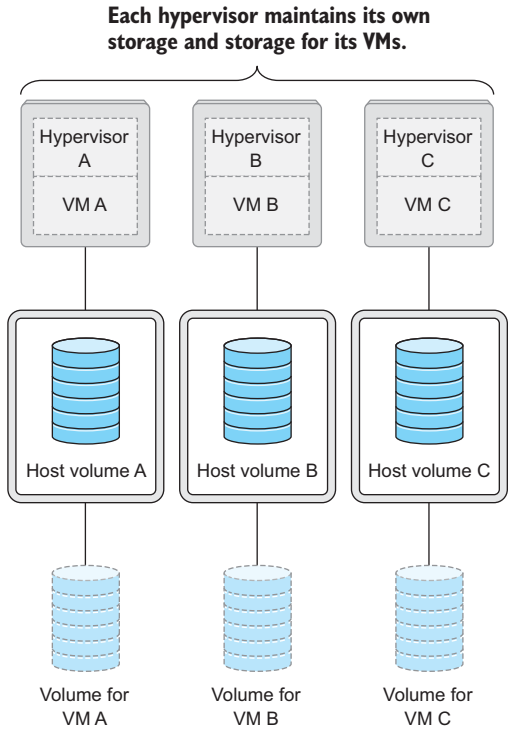

Figure 9.3 Independent host volumes

In this figure, hypervisor A uses host volume A to store data for VM A. If VM B were migrated to hypervisor A, volume information would need to be migrated to the new hypervisor. The benefit of this shared-nothing architecture is that the failure domain is reduced, but the migration costs are increased. As with the shared–host volume model, the hypervisor is managing volumes for the VMs it maintains, so the storage subsystem is still not actively managed as part of the virtual server platform. This is not to say that there are no storage vendor integrations with vSphere and Hyper-V, simply that a high level of integration is not fundamental to their operation.

As discussed in previous chapters, there are two types of storage in the OpenStack world: object and block. OpenStack Swift provides object storage services, which can be used to provide backend storage for VM images and snapshots. If you've been working as a virtual server platform administrator, you might not have ever worked with object storage. Although object-based storage is very powerful, it's not required to provide virtual infrastructure and is outside the scope of this book. In contrast, block storage is a required VM component and is covered in several chapters.

The majority of this book has been devoted to working with block storage using OpenStack Block Storage (Cinder). Using the OpenStack Compute service (Nova), it's possible to boot a VM without using Cinder. But the volume used to boot the VM is *ephemeral*, which means that when the VM is terminated, data on the VM volume is also removed. In contrast to ephemeral storage is *persistent* storage, which can be detached from a VM and reassigned to another VM. The relationship between the hypervisor, persistent VM volume, Cinder, and the storage subsystem is shown in figure 9.4.

As shown in the figure, the VM, not the hypervisor, communicates directly with an underlying storage subsystem. In contrast, both in the case of VMware vSphere and Microsoft Hyper-V, VM storage is provided by the hypervisor. This fundamental

difference in operation forces a higher level of storage subsystem management for OpenStack. Where other virtual server platforms might manage VM volumes on the hypervisor level, Cinder interfaces with hardware and software storage systems to provide functions like volume creation, expansion, migration, deletion, and so on. A list of storage systems and supported functions can be found on the Cinder support matrix wiki: https://wiki.openstack.org/wikiCinderSupportMatrix.

Aside from a few corner cases, most OpenStack deployments will use Cinder to manage volume storage. But the question remains, what type of storage hardware or software platform should Cinder manage?

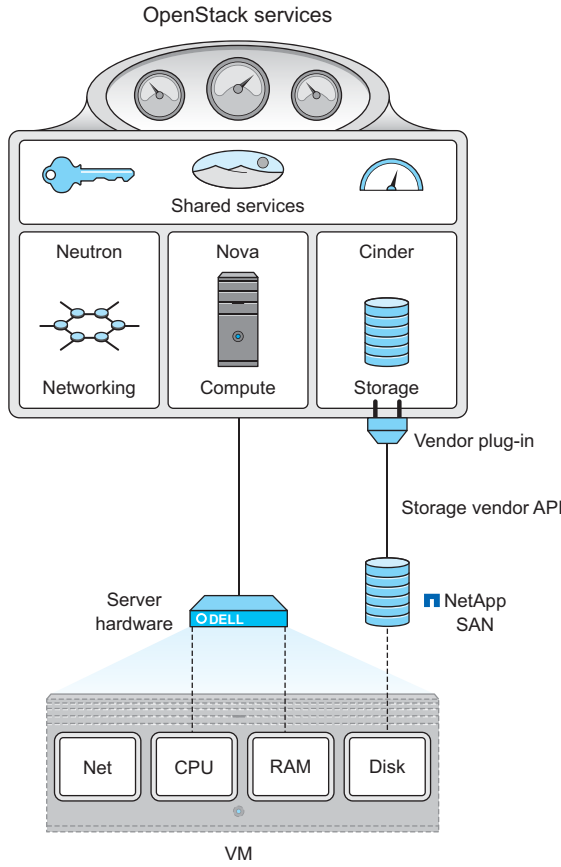

Figure 9.4 OpenStack VM volumes

The question of the underlying storage subsystem depends on the intersection of several factors, including what your current storage vendors are, whether you're willing to dedicate a storage system to OpenStack, and what your tolerance is for risk in your environment. For instance, suppose you wish to mimic the operations of vSphere or Hyper-V, and your storage is provided by a large centralized storage area network (SAN). In this case, you might not want Cinder to communicate directly with your shared central system, but you do want to use storage from this system. In this scenario, you could abstract the underlying storage subsystem by attaching independent volumes directly to compute or storage nodes, similar to what was shown in figure 9.3. You'd then use LVM to manage your independent host volume. LVM would then be managed by Cinder, thus abstracting the underlying storage subsystem. Managing an independent volume using LVM doesn't invalidate the storage model shown in figure 9.4; in fact, LVM was used as the underlying storage subsystem for Cinder in chapter 7. There are, of course, other options where centralized storage software and hardware can be used directly, but LVM is a common choice for people using shared central services.

In the next section, we'll cover choices in servers.

9.1.4 What kind of server are you?

The past few sections have covered choices you need to make in providing network and storage resources for your VM. For the most part, the choices that you make regarding networking and storage are based on your current and intended future state of operations. In neither case was there any mention of changing the underlying configuration of network or storage hardware and software in a way that was fundamentally different from what you're doing now. In fact, everything we've discussed in this chapter so far has described an architecture that can be used to mimic a traditional virtual server platform environment. When it comes to OpenStack Compute (Nova), the support matrix (https://wiki.openstack.org/wiki/HypervisorSupportMatrix) doesn't list server hardware vendors, it lists supported hypervisors.

> **Bare metal and containers**
>
> Although it's outside the scope of this book, using OpenStack to provision bare-metal servers and LXC/Docker containers is also an option. The use of containers in the OpenStack environment is especially interesting for those who'd like to use OpenStack to provide applications.

Commoditization in the server hardware market around the x86_64 instruction set all but guarantees that if you purchase a server with an Intel or AMD x86_64 processor, any hypervisor will work. Although some hardware configurations and vendors provide advantages over one another, OpenStack Compute is hardware agnostic. The real question you face is what hypervisor you want to use.

You must consider your motivations for deploying OpenStack in the first place. If your intent is to replace an existing commercial virtual server platform in a way that mimics the operation of that platform, then you likely won't want to maintain the license cost of the commercial hypervisor.

> **FREE VERSIONS OF VMWARE ESXI AND MICROSOFT HYPER-V** Recently VMware and Microsoft have released free versions of their core visualization platforms. The changes in licensing have generated a great deal of community interest in using these hypervisors with OpenStack. But there are drawbacks, including limited initial community support compared to KVM.

Based on OpenStack user surveys, KVM is the hypervisor used in the vast majority of OpenStack deployments, and most community support will be based around using KVM. In summary, you should select server hardware based on current business practices and use KVM until you have a reason to use something else.

This section covered the architectural decisions around deploying OpenStack to replace an existing virtual server environment. The next section covers the architecture for deploying a greenfield environment for an on-premise private cloud.

9.2 Why build a private cloud?

OpenStack is used in some very large public cloud services, including DreamHost DreamCompute (see www.openstack.org/marketplace/public-clouds/). These companies make use of OpenStack projects, along with their own custom integration services, to manage resources on a much larger scale than most enterprise customers. The ratio of servers to administrators varies wildly, based on the size and complexity of an organization's infrastructure and related workloads. For example, it's common to have a 30:1 or lower ratio of physical servers to administrators for small and medium-sized business, whereas a medium to large enterprise might have 500:1 virtual servers to administrators. But when you consider that the Amazons and Googles of the world achieve ratios of 10,000:1 physical servers to administrators, you can start to appreciate the infrastructure management efficiencies that large-scale providers have developed.

When an enterprise provides public cloud–like services from resources that are exclusive to the enterprise, we call this a *private cloud*. By adopting technologies and operational practices born out of large-scale providers for private clouds, enterprises are able to develop hybrid cloud strategies based on workload. So why do private clouds exist? Why aren't all workloads in the public cloud? A detailed study of IT strategy concerning public, private, and hybrid clouds is beyond the scope of this book. But this section presents several arguments for deploying a private cloud for your enterprise and for adopting a hybrid cloud strategy.

9.2.1 Public cloud economy-of-scale myth

Cloud computing is often described as the computational equivalent of the electrical power grid. Considering that the economic definition of *utility* is the ability of a commodity to satisfy human wants, it's easy to see how cloud computing gained this reputation.

But there's a fundamental flaw in this power-grid comparison. The power grid, much like cloud computing, produces a commodity that must be consumed in real time, but this is where the comparison ends. There's a vast difference between the economies of scale related to the production of the commodities. Bulk power generation from nuclear facilities are orders of magnitude more cost effective than what could be produced by a cloud of consumer-grade power generators. On the computer side, there isn't any quantum or other type of computer capable of producing computational power with a cost benefit greater than commodity clusters, so the economies of scale aren't comparable. In fact, the profit margins across commodity servers is so low that the difference between what enterprise and public cloud providers pay for the same hardware is negligible.

This is not to say that there aren't advantages for large-scale providers. For instance, a large diverse group of workloads balanced across many resources should be more efficient than its small-scale unoptimized counterpart. But enterprise customers can take advantage of the same fundamental components as public providers, and often near the same price point.

9.2.2 Global scale or tight control

Public cloud providers offer a wide variety of services beyond IaaS, but for the sake of argument, we'll restrict our evaluation of public cloud to IaaS.

Consider IaaS as providing VMs comprised of discrete components (CPU, RAM, storage, and network). Public cloud consumers are unaware of the physical infrastructure used to provide public IaaS offerings. Specifically, users have no way of knowing if they're paying for the latest and greatest or yesterday's technology. To complicate matters further, the consumer has no way to determine the level of oversubscription for a particular type of shared service. Without knowing the underlying platform and the number of shared users, there's no passive, quantitative way to measure IaaS value between public providers. Consider a case where provider A has a cost per unit of X and an oversubscription of 20:1, whereas provider B has a cost per unit of 2X and rate of 10:1. The total cost is clearly the same, but in the eyes of the customer, provider B is simply twice the cost of provider A.

In many industries, service level agreements (SLAs) are defined so consumers can evaluate the expected quality of a service provider. Typically, public cloud SLAs are based on uptime, not performance. No doubt large consumers of public cloud resources develop performance SLAs with their public providers, but this isn't common in the small business to medium enterprise space. Without a quantitative approach, you're left with qualitative evaluation based on active measurements. But although there's no shortage of benchmarks in the computing industry, an accepted workload measurement standard for cloud services has yet to emerge.

In the absence of clearly defined SLAs and verification methods, it's difficult to compare the value of public cloud offerings between providers. In addition, value comparisons can change over time as workloads change across providers. For many workloads, the benefits of global on-demand IaaS far outweigh resource performance variability. But for other workloads, the tightly controlled performance provided by a private cloud is necessary.

9.2.3 Keeping data gravity private

Dave McCrory coined the term *data gravity* to describe how applications and other services are drawn to sources of data, analogous to the attraction of physical objects in the universe proportional to their mass. Public cloud providers recognize this phenomenon and typically make it much more financially attractive to move data into their services than out. For instance, there's no charge to move data into an Amazon EC3 service, but there is a tiered pricing structure to move data out of the service. A similar pricing structure exists for Amazon EC2 instances and other IaaS provider offerings, to entice users to move data into a specific cloud provider and keep it there.

Cloud providers can exploit the data gravity phenomenon through their transfer-rate pricing structure to create cloud vendor lock-in for consumers. Consider the case where an organization determines that based on the unit price of resources (not accounting for transfers) it's cost effective to move all of its storage and related computation to a public cloud provider. Even if the majority of data was generated

outside the cloud provider, there would be no transfer penalty to continuously add data to storage maintained by the public provider. Now suppose this organization wants to utilize a secondary public cloud provider for redundancy. Although the new provider might not charge transfer fees for incoming data, the existing provider will charge for outgoing data, which could greatly increase the cost.

The same holds true if you want to process information locally or if you want to take advantage of processing on a lower-cost provider. When the majority of your data is maintained in the public cloud, it's hard for services to escape the force of data gravity pulling you to your data provider.

Keeping the majority of your data in a private cloud allows you to move data in and out of public clouds as needed. For many workloads and organizations, the ability to consume services from several providers, including local resources, outweighs the benefits of pure public cloud offerings.

9.2.4 Hybrid moments

The principle of *pay as you go* is a key differentiator between public and private clouds. It's easy to understand the economic benefits of the spending 1 hour on 1000 computers versus using 1 computer for 1000 hours when timely information is essential. But at first glance, the economics of purchasing cloud services doesn't seem viable when you assume 100% service usage 24/7/365.

The idea of usage-based pricing allows for the redirection of capital that would otherwise be committed to infrastructure investment to be repositioned in other strategic investments. Large public cloud providers have a natural tolerance for load spikes for specific workloads. Given the diversity of workloads over a wide range of clients, it's unlikely that all workloads for all clients will experience a simultaneous resource peak need. Given the natural elasticity of the cloud, much of the capacity risk related to operating a private cloud is transferred to the public cloud provider. Private clouds must be built for the peak usage, regardless of peak duration, and this involves extra costs. In most cases, peak workloads exceed average workloads by a factor of 5 to 1.

The public cloud has been adopted broadly across the enterprise, but rarely exclusively. Based on enterprise surveys, public cloud services are adopted for specific strategic workloads, whereas private clouds provide services for a more diverse range of services.

The majority of IT organizations have adopted a hybrid cloud strategy, making use of both public and private cloud resources. The real economic benefits of the public cloud in the enterprise can be realized if organizations find the right balance between private and public cloud services.

For the service provider, OpenStack can provide project components that can be used to construct massively global clouds of resources. For the enterprise, the OpenStack framework can be used to deploy private cloud services. From an integration standpoint, API compatibility between public and private OpenStack-based providers allows the enterprise to optimally consume resources based on workload requirements.

9.3 Building a private cloud

This book has focused on approaching OpenStack from the enterprise perspective not as a virtual server platform replacement, but as a cloud management framework. The previous section covered the benefits of deploying a private cloud. In addition, the benefits of adopting a hybrid cloud strategy, where resources can be managed based on a common OpenStack API control set, were covered.

This section ties together what you've learned in previous chapters and prepares you for the rest of the chapters in part 3 of this book.

9.3.1 OpenStack deployment tools

The deployment tool you choose will be based on your existing vendor relationships, current operational strategy, and future cloud direction. There are three approaches you can take when deploying OpenStack.

The first approach, a manual deployment, was covered in part 2 of this book. Manual deployments offer the most flexibility but have obvious problems at scale.

The second approach is to use general orchestration tools, such as Ansible, Chef, Juju, Puppet, and Vagrant, which are used to deploy a wide range of systems and applications. Being well versed in a collection of general orchestration tools allows you to deploy not only OpenStack, but also applications using OpenStack resources. The drawback of these systems is that each tool has its particular role to play, so you end up using a wide range of general-purpose tools, which constitutes a training and operational challenge for the organization adopting this strategy.

The third approach, using a standalone OpenStack deployment and management tool, is a familiar approach for those in the enterprise. OpenStack deployment platforms like HP Helion, Mirantis Fuel, and Red Hat RDO not only provide easy-to-use tools for deploying OpenStack, they also provide their own validated OpenStack versions and deployment methods. You can think of this the same way you think of Linux distributions. Like the Linux kernel, there's one OpenStack source repository (with many branches) for community development. Enhancements and fixes recognized by Linux and OpenStack communities make their way into the respective code repositories. But as in the Linux community, vendors validate community work for specific use cases, for which they provide support. Just as you don't technically pay for the Linux kernel when you pay for a supported Linux distribution, you aren't paying for OpenStack when you purchase a commercially supported

Research, big data, and OpenStack

For those who deal in the area of research computing, new consolidated infrastructure management options are emerging. Traditional high-performance computing (HPC) niche vendors like Bright Computing and StackIQ are getting into the Hadoop and OpenStack game. These vendors, and many others, are adapting their HPC deployment and management platforms to provide a holistic management view across HPC, Hadoop, and OpenStack deployment.

OpenStack distribution. Commercial OpenStack vendors typically provide community supported versions of their deployment tools—one such tool, Mirantis Fuel, is covered in chapter 11.

In many IT shops, the title "systems programmer" is still used to describe roles that haven't had much to do with programming since the days of the mainframe. In some organizations, however, the systems programmer role has been reborn as part of the DevOps movement (referring to systems administrators who also write code and scripts). A systems administrator who is accustomed to manually double-clicking their way through VM and application deployments isn't likely to be comfortable with general orchestration tools that they have to code or script together. On the other hand, someone with automation experience will feel very restricted with only a standalone OpenStack deployment tool.

Based on the strategic direction of your organization, you should pick an approach that not only deploys OpenStack but that is sustainable. For some this approach will involve purchasing a commercially supported distribution and assigning existing resources to work with the supporting vendor. Other organizations will choose to develop DevOps teams that will not only be able to deploy OpenStack but also orchestrate resources and applications across private and public cloud providers.

9.3.2 Networking in your private cloud

In section 9.1.2 we discussed Nova networking. When using Nova networking, your choice of networking hardware isn't of great importance, because OpenStack does very little to manage your network. However, throughout most of this book, networking is discussed in the context of OpenStack Networking (Neutron). When using Neutron, your choice of network hardware and software is very important, because OpenStack will be managing many aspects of your network.

At the time of writing, few vendor-provided L3 (router) services exist (see https://www.openstack.org/marketplace/drivers/). Because L3 services will likely be provided by OpenStack, the focus of this discussion will be on choices related to L2.

From a Neutron L2 perspective, you have two choices. Your first choice is to use a community- or vendor-provided monolithic network plug-in. These plug-ins are considered monolithic because all L2 OpenStack services must be implemented by the driver, as shown in figure 9.5.

Initially, monolithic plug-ins were the only way to integrate vendor hardware and software with OpenStack Networking. Several of these plug-ins have been developed for vendor hardware, including Arista, Cisco, Melinox,

Neutron Distributed Virtual Routing (DVR)

One of the many goals of the Neutron DVR subproject is to provide distributed routing with compute nodes, integration with routing hardware, and routing service migration between nodes. Although the project is fairly new, the DVR project will likely serve as the primary integration point for most advanced L3 vendor services.

VMware, and others. The issue with this approach is that the plug-in code must be modified with subsequent OpenStack releases, even if nothing has changed on the vendor side. The effort involved in separating out vendor-specific code from OpenStack code led to the second choice in Neutron L2 networking, the modular Layer 2 (ML2) plug-in, previously covered in chapter 6 and shown in figure 9.6.

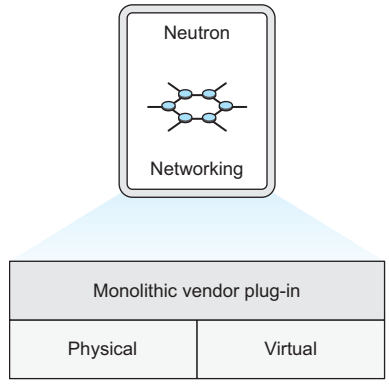

Figure 9.5 Monolithic plug-in architecture

The ML2 plug-in framework allows communities and vendors to provide L2 support much more easily than with monolithic plug-ins. The majority of vendors, even those who had previously developed monolithic plug-ins, are now adopting the ML2 plug-in by writing mechanism drivers for their specific technologies.

Based on the OpenStack user survey, Open vSwitch (OVS) is the most commonly used network driver (interface between standalone hardware and software packages and OpenStack) used in OpenStack deployments. Due to its popularity, OVS was used as a network driver in both parts 1 and 2 of this book. Specifically, using ML2 terminology, the ML2 plug-in was configured to use the GRE type driver and OVS mechanism driver. By using a combination of an overlay (GRE) type driver and a software switch (OVS), we simplified the switch hardware configuration down to simple connectivity between compute and network nodes. The hardware configuration is simple in this context, because OVS is providing virtual switching (traffic isolation on the OVS level), so you only need to worry about making sure that OVS switches on separate servers can communicate with each other.

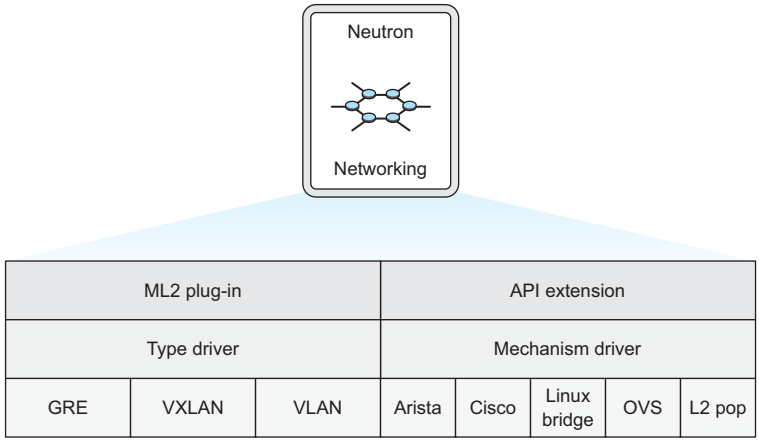

Figure 9.6 ML2 plug-in architecture

There are many benefits to using network overlays like GRE and VXLAN, including scale and flexibility. But there are performance costs related to using network overlays and software switches (OVS) in general. In chapter 11 the VLAN type driver will be used with the OVS mechanism driver. OVS is still the network driver, but instead of overlays connecting OVS instances, OVS makes use of a range of VLANS. The VLANS used by OVS in the chapter 11 example must be manually configured on the switch. In this context, some switching load is offloaded to the hardware switch and some remains in OVS.

The next progression in offloading network load from software to hardware is to use a mechanism driver that deals with all L2 operations on the hardware device (this could be a hybrid hardware and software device). In this configuration, OpenStack Networking operations are translated by the network driver into vendor-specific operations. This doesn't mean that you have to use VLAN as your type driver when using a hardware vendor mechanism driver. In fact, there are many vendor-managed types, including very powerful VXLAN type drivers, that are offloaded in hardware.

As with most things in OpenStack and technology, generalized solutions (in software) come at the cost of performance (using dedicated hardware). You must determine if the performance of software switching and overlay is acceptable, or if your private cloud can benefit from the performance gained through tight integration of OpenStack Networking and vendor hardware.

9.3.3 Storage in your private cloud

In chapter 7 you walked through the deployment of the OpenStack storage node using Cinder. The purpose of the storage node was to provide block storage to VMs. Just as Neutron uses network drivers to communicate with underlying software and hardware network resources, Cinder uses storage drivers (see www.openstack.org/marketplace/drivers/) to communicate with storage resources.

In chapter 7 an LVM-configured volume was managed by Cinder. In that example, an LVM storage driver was used by Cinder to interface with the underlying LVM subsystem. Where did the storage device that was used by the LVM volume come from? As discussed in section 9.1.3, the LVM volume could have been a local disk, or it could have been provided from an external source like a SAN. From the standpoint of LVM, and by relation Cinder, as long as the device shows up as a block storage device in the Linux kernel, it can be used. But this is similar to OVS using network hardware as a physical transport. Through the abstraction of the underlying storage device by LVM, you're losing many of the advanced storage features the underlying storage subsystem might offer. Just as with OVS, OpenStack is unaware of the capabilities of the underlying physical infrastructure, and storage functions are offloaded to software. Luckily, there are many Cinder storage drivers for OpenStack, including drivers for storage systems provided by Ceph, Dell, EMC, Fujitsu, Hitachi, HP, IBM, and many others. As with OpenStack Networking, integrating OpenStack Storage with hardware and software storage subsystems through the use of vendor storage drivers allows OpenStack to make use of the advanced features of the underlying system.

Based on OpenStack user surveys, the Ceph storage system is used in the majority of OpenStack deployments. Due to its popularity in the OpenStack community and its inclusion in many standalone OpenStack deployment tools, chapter 10 is dedicated to walking through a Ceph deployment.

Like vendor decisions related to OpenStack Networking, storage decisions need to be based on your current capabilities and future direction. Although it's extremely popular with the OpenStack community, building out support for a Ceph storage cluster might not be the right choice if the rest of the storage in your enterprise is EMC. Likewise, many advanced storage features previously found only in high-end arrays are now found in Cinder or are not needed because of some other aspect of private cloud operation, so purchasing a high-end array might not be necessary.

As you continue through the remaining chapters in part 3 of this book, think about the type of environment you want to construct. For some, a purpose-built system with deep vendor integration will be the best fit. For others, a flexible general-purpose deployment will be the right choice. Regardless of the path you take, make sure OpenStack is the right tool for the job and that your organization is well positioned to take advantage of the benefits of the OpenStack framework.

9.4 Summary

- If you intend to limit the use of your virtual infrastructure to what you can do with physical infrastructure, the benefits of OpenStack in your environment will also be limited.
- Systems administrators who are happy with manually provisioning infrastructure can view OpenStack as either incomplete or unnecessary.
- Systems administrators, developers, consultants, architects, and IT leadership interested in the benefits of cloud computing can view OpenStack as a disruptive technology for the enterprise.
- Users wishing to use OpenStack as a replacement for traditional virtual server infrastructure will find Nova networking comparable to their existing environment, whereas those building a private cloud will likely use Neutron networking.
- Users wishing to use OpenStack as a replacement for traditional virtual server infrastructure will find LVM-based storage comparable to their existing environment, whereas those building a private cloud will likely use Ceph or another vendor-specific directly attached VM storage system.
- By adopting technologies and operational practices born out of large-scale providers for private clouds, enterprises can develop hybrid cloud strategies for best-of-breed solutions.