CHAPTER 1

Cloud Computing Basics

In this chapter we go through some basic concepts with the purpose of providing context for the discussions in the chapters that follow. Here, we review briefly the concept of the cloud as defined by the U.S. National Institute of Standards and Technology, and the familiar terms of IaaS, PaaS, and SaaS under the SPI model. What is not often discussed is that the rise of cloud computing comes from strong historical motivations and addresses shortcomings of predecessor technologies such as grid computing, the standard enterprise three-tier architecture, or even the mainframe architecture of many decades ago.

From a security perspective, the main subjects for this book—perimeter and endpoint protection—were pivotal concepts in security strategies prior to the rise of cloud technology. Unfortunately these abstractions were inadequate to prevent recurrent exploits, such as leaks of customer credit card data, even before cloud technology became widespread in the industry. We'll see in the next few pages that, unfortunately for this approach, along with the agility, scalability, and cost advantages of the cloud, the distributed nature of these third-party-provided services also introduced new risk factors. Within this scenario we would like to propose a more integrated approach to enterprise security, one that starts with server platforms in the data center and builds to the hypervisor operating system and applications that fall under the notion of *trusted compute pools,* covered in the chapters that follow.

Defining the Cloud

We will use the U.S. government's National Institute of Standards and Technology (NIST) cloud framework for purposes of our discussions in the following chapters. This provides a convenient, broadly understood frame of reference, without our attempts to treat it as a definitive definition or to exclude other perspectives. These definitions are stated somewhat tersely in *The NIST Definition of Cloud Computing*[1] and have been elaborated by the Cloud Security Alliance.[2]

[1]Peter Mell and Timothy Grance, *The NIST Definition of Cloud Computing*. NIST Special Publication 800-145, September 2011.
[2]*Security Guidance for Critical Areas of Focus in Cloud Computing*, Cloud Security Alliance, rev. 2.1 (2009).

The model consists of three main layers (see Figure 1-1), laid out in a top-down fashion: global essential characteristics that apply to all clouds, the service models by which cloud services are delivered, and how the services are instantiated in the form of deployment models. There is a reason for this structure that's rooted in the historical evolution of computer and network architecture and in the application development and deployment models. Unfortunately most discussions of the cloud gloss over this aspect. We assume readers of this book are in a technology leadership role in their respective fields, and very likely are influential in the future direction of cloud security. Therefore, an understanding of the dynamics of technology evolution will be helpful for the readers in these strategic roles. For this purpose, the section that follows covers the historical context that led to the creation of the cloud.

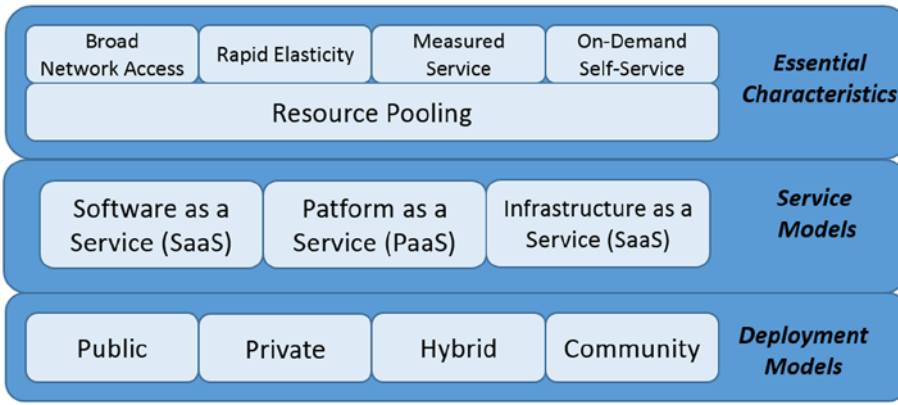

Figure 1-1. *NIST cloud computing definition*

The Cloud's Essential Characteristics

The main motivation behind the pervasive adoption of cloud use today is economic. Cloud technology allows taking a very expensive asset, such as a $200 million data center, and delivering its capabilities to individual users for a few dollars per month, or even for free, in some business models. This feat is achieved through *resource pooling*, which is essentially treating an asset like a server as a fungible resource; a resource-intensive application might take a whole server, or even a cluster of servers, whereas the needs of users with lighter demands can be packed as hundreds or even thousands to a server.

This dynamic range in the mapping of applications to servers has been achieved through virtualization technology. Every intervening technology and the organizations needed to run them represent overhead. However, the gains in efficiency are so large that this inherent overhead is rarely in question. With applications running on bare-metal operating systems, it is not unusual to see load factors in the single digits. Cloud applications running on virtualized environments, however, typically run utilizations up to 60 to 80 percent, increasing the application yield of a server by several-fold.

Cloud applications are inherently distributed, and hence they are necessarily *delivered over a network*. The largest applications may involve millions of users, and the conveyance method is usually the Internet. An example is media delivery through Netflix, using infrastructure from Amazon Web Services. Similarly, cloud applications are expected to have automated interfaces for setup and administration. This usually means they are accessible *on demand* through a *self-service* interface. This is usually the case, for instance, with email accounts through Google Gmail or Microsoft `Outlook.com`.

With the self-service model, it is imperative to establish methods for *measuring service*. This measuring includes guarantees of service provider performance, measurement of services delivered for billing purposes, and very important from the perspective of our discussion, measurement of security along multiple vectors. The management information exchanged between a service provider and consumers is defined as *service metadata*. This information may be facilitated by auxiliary services or *metaservices*.

The service provider needs to maintain a service pool large enough to address the needs of the largest customer during peak demand. The expectation is that, with a large customer base, most local peaks and valleys will cancel out. In order to get the same quality of service (QoS), an IT organization would need to size the equipment for expected peak demand, leading to inefficient use of capital. Under some circumstances, large providers can smooth out even regional peaks and valleys by coordinating their geographically disperse data centers, a luxury that mid-size businesses might not be able to afford.

The expectation for cloud users, then, is that compute, network, and data resources in the cloud should be provided on short order. This property is known as *elasticity*. For instance, virtual machines should be available on demand in seconds, or no more than minutes, compared to the normal physical server procurement process that could take anywhere from weeks to years.

At this point, we have covered the *what* question—namely, the essential characteristics of the cloud. The next section covers service models, which is essentially the *how* question.

The Cloud Service Models

The unit of delivery for cloud technology is a *service*. NIST defines three service models, affectionately known as the SPI model, for SaaS, PaaS, and IaaS, or, respectively, software, platform, and infrastructure services.

Under the *SaaS* service model, applications run at the service provider or delegate services under the service network paradigm described below. Users access their applications through a browser, thin client, or mobile device. Examples are Google Docs, Gmail, and MySAP.

PaaS refers to cloud-based application development environments, compilers, and tools. The cloud consumer does not see the hardware or network directly, but is able to determine the application configuration and the hosting environment configuration.

IaaS usually refers to cloud-based compute, network, and storage resources. These resources are generally understood to be virtualized. For simplicity, some providers may require running pre-configured or highly paravirtualized operating system images. This is

how a pool of physical hosts is able to support 500 or more virtual machines each. Some providers may provide additional guarantees—for instance, physical hosts shared with no one else or direct access to a physical host from a pool of hosts.

The bottom layer of the NIST framework addresses *where* cloud resources are deployed, which is covered in the next section.

The Cloud Deployment Models

The phrase *cloud deployment models* refers to the environment or placement of cloud services as deployed. The quintessential cloud is the multi-tenant *public cloud,* where the infrastructure is pooled and made available to all customers. Cloud customers don't have a say in the selection of the physical host where their virtual machines land. This environment is prone to the well-known noisy and nosy neighbor problems, with multiple customers sharing a physical host.

The *noisy neighbor* problem might manifest when a customer's demand on host resources impacts the performance experienced by another customer running on the same host; an application with a large memory footprint may cause the application from another customer to start paging and to run slowly. An application generating intense I/O traffic may starve another customer trying to use the same resource.

As for the *nosy neighbor* problem, the hypervisor enforces a high level of isolation between tenants through the virtual machine abstraction—much higher, for instance, than inter-process isolation within an operating system. However, there is no absolute proof that the walls between virtual machines belonging to unrelated customers are completely airtight. Service-level agreements for public clouds usually do not provide assurances against tenants sharing a physical host. Without a process to qualify tenants, a virtual machine running a sensitive financial application could end up sharing the host with an application that has malicious intent. To minimize the possibility of such breaches, customers with sensitive workloads will, as a matter of practice, decline to run them in public cloud environments, choosing instead to run them in corporate-owned infrastructure. These customers need to forfeit the benefits of the cloud, no matter how attractive they may seem.

As a partial remedy for the nosy neighbor problem, an entity may operate a cloud for exclusive use, whether deployed on premises or operated by a third party. These clouds are said to be *private clouds*. A variant is a *community cloud*, operated not by one entity but by more than one with shared affinities, whether corporate mission, security, policy, or compliance considerations, or a mix thereof.

The community cloud is the closest to the model under which a predecessor technology, *grid computing,* operated. A computing grid was operated by an affinity group. This environment was geared toward high-performance computing usages, emphasizing the allocation of multiple nodes—namely, computers or servers to run a job of limited duration—rather than an application running for indefinite time that might use a fractional server.

The broad adoption of the NIST definition for cloud computing allows cloud service providers and consumers alike to establish an initial set of expectations about management, security, and interoperability, as well as determine the value derived from use of cloud technology. The next section covers these aspects in more detail.

The Cloud Value Proposition

The NIST service and deployment models—namely public, private, and hybrid—get realized through published APIs, whether open or proprietary. It is through these APIs that customers can elicit capabilities related to management, security, and interoperability for cloud computing. The APIs get developed through diverse industry efforts, including the Open Cloud Computing Interface Working Group, Amazon EC2 API, VMware's DMTF-submitted vCloud API, Rackspace API, and GoGrid's API, to name just a few. In particular, open, standard APIs will play a key role in cloud portability, federation, and interoperability, as will common container formats such as the DMTF's Open Virtualization Format or OVF, as specified by the Cloud Security Alliance in the citation above.

Future flexibility, security, and mobility of the resultant solution, as well as its collaborative capabilities, are first-order considerations in the design of cloud-based solutions. As a rule of thumb, de-perimeterized solutions have the potential to be more effective than perimeterized solutions relying on the notion of an enterprise perimeter to be protected, especially in cloud-based environments that have no clear notion of inside or outside. The reasons are complex. Some are discussed in the section "New Enterprise Security Boundaries," later in this chapter. Careful consideration should also be given to the choice between proprietary and open solutions, for similar reasons.

The NIST definition emphasizes the flexibility and convenience of the cloud, enabling customers to take advantage of computing resources and applications that they do not own for advancing their strategic objectives. It also emphasizes the supporting technological infrastructure, considered an element of the IT supply chain managed to respond to new capacity and technological service demands without the need to acquire or expand in-house complex infrastructures.

Understanding the dependencies and relationships between the cloud computing deployment and the service models is critical for assessing cloud security risks and controls. With PaaS and SaaS built on top of IaaS, as described in the NIST model above, inherited or imported capabilities introduce security issues and risks. In all cloud models, the risk profile for data and security changes is an essential factor in deciding which models are appropriate for an organization. The speed of adoption depends on how fast security and trust in the new cloud models can be established.

Cloud resources can be created, moved, migrated, and multiplied in real time to meet enterprise computing needs. A trusted cloud can be an application accessible through the Web or a server provisioned as available when needed. It can involve a specific set of users accessing it from a specific device on the Internet. The cloud model delivers convenient, on-demand access to shared pools of hardware and infrastructure, made possible by sophisticated automation, provisioning, and virtualization technologies. This model decouples data and software from the servers, networks, and storage systems. It makes for flexible, convenient, and cost-effective alternatives to owning and operating an organization's own servers, storage, networks, and software.

However, it also blurs many of the traditional, physical boundaries that help define and protect an organization's data assets. As cloud- and software-defined infrastructure becomes the new standard, the security that depends on static elements like hardware, fixed network perimeters, and physical location won't be guaranteed. Enterprises seeking the benefits of cloud-based infrastructure delivery need commensurate security and

compliance. Covering this topic is the objective for this book. The new perimeter is defined in terms of data, its location, and the cloud resources processing it, given that the old definition of on-premise assets no longer applies.

Let's now explore some of the historical drivers of the adoption of cloud technology.

Historical Context

Is it possible to attain levels of service in terms of security, reliability, and performance for cloud-based applications that rival implementations using corporate-owned infrastructure? Today it is challenging not only to achieve this goal but also to measure that success except in a very general sense. For example, consider doing a cost rollup at the end of a fiscal year. There's no capability today to establish operational metrics and service introspection. A goal for security in the cloud, therefore, is not to just match this baseline but to surpass it. In this book, we'd like to claim that is possible.

Cloud technology enables the disaggregation of compute, network, and storage resources in a data center into pools of resources, as well as the partitioning and re-aggregation of these resources according to the needs of consumers down the supply chain. These capabilities are delivered through a network, as explained earlier in the chapter. A virtualization layer may be used to smooth out the hardware heterogeneity and enable configurable software-defined data centers that can deliver a service at a quality level that is consistent with a pre-agreed SLA.

The vision for enterprise IT is to be able to run varied workloads on a software-defined data center, with ability for developers, operators, or in fact, any responsible entity to use self-service unified management tools and automation software. The software-defined data center must be abstracted from, but still make best use of, physical infrastructure capability, capacity, and level of resource consumption across multiple data centers and geographies. For this vision to be realized, it is necessary that enterprise IT have products, tools, and technologies to provision, monitor, remediate, and report on the service level of the software-defined data center and the underlying physical infrastructure.

Traditional Three-Tier Architecture

The three-tier architecture shown in Figure 1-2 is well established in data centers today for application deployment. It is highly scalable, whereby each of the tiers can be expanded independently by adding more servers to remove choke points as needed, and without resorting to a forklift upgrade.

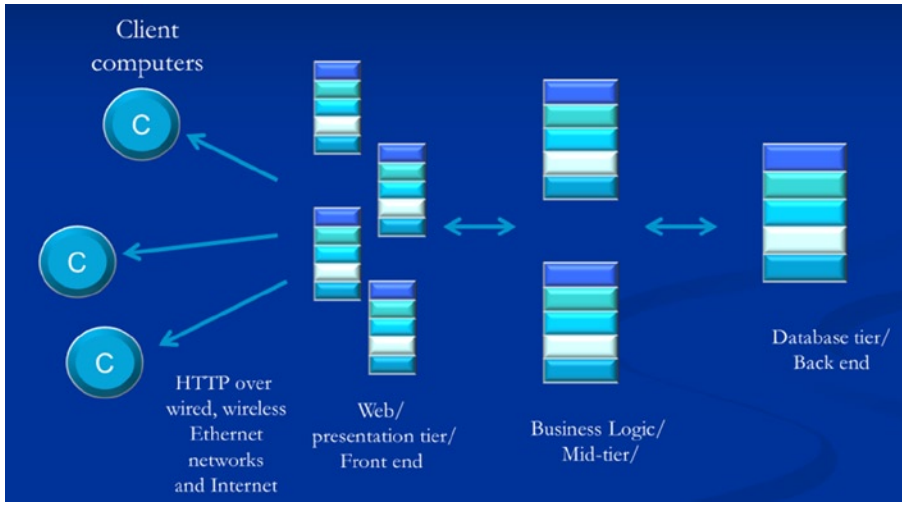

Figure 1-2. *Three-tier application architecture*

While the traditional three-tier architecture did fine in the scalability department, it was not efficient in terms of cost and asset utilization, however. This was because of the reality of procuring a physical asset. If new procurement needs to go through a budgetary cycle, the planning horizon can be anywhere from six months to two years. Meanwhile, capacity needs to be sized for the expected peak demand, plus a generous allowance for demand growth over the system's planning and lifecycle, which may or may not be realized. This defensive practice leads to chronically low utilization rates, typically in the 5 to 15 percent range. Managing infrastructure in this overprovisioned manner represents a sunk investment, with a large portion of the capacity not used during most of the infrastructure's planned lifetime. The need for overprovisioning would be greatly alleviated if supply could somehow be matched with demand in terms of near-real time—perhaps on a daily or even an hourly basis.

Server consolidation was a technique adopted in data centers starting in the early 2000s, which addressed the low-utilization problem using virtualization technology to pack applications into fewer physical hosts. While server consolidation was successful at increasing utilization, it brought significant technical complexity and was a static scheme, as resource allocation was done only at planning or deployment time. That is, server consolidation technology offered limited flexibility in changing the machine allocations during operations, after an application was launched. Altering the resource mix required significant retooling and application downtime.

Software Evolution: From Stovepipes to Service Networks

The low cost of commodity servers made it easy to launch application instances. However, little thought was given to how the different applications would interact with one another. For instance, the information about the employee roster in an organization

is needed for applications as diverse as human resources, internal phone directory, expense reporting, and so on. Having separate copies of these resources meant allocating infrastructure to run these copies, and running an infrastructure was costly in terms of extra software licensing fees. Having several copies of the same data also introduced the problem of keeping data synchronized across the different copies.

■ **Note** Cloud computing has multiplied the initial gains in efficiency delivered by server consolidation by allowing dynamic rebalancing of workloads at run time, not just at planning or deployment time.

The initial state of IT applications circa 2000 ran in stovepipes, shown in Figure 1-3 on the left, with each application running on assigned hardware. Under cloud computing, capabilities common across multiple stacks, such as the company's employee database, are abstracted out in the form of a service or of a limited number of service instances that would certainly be smaller than the number of application instances. All applications needing access to the employee database, for instance, get connected to the employee database service.

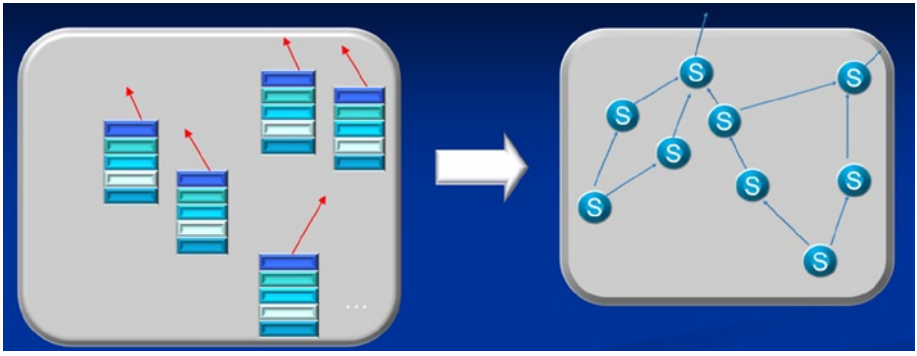

Figure 1-3. *Transition from stovepipes to a service network ecosystem*

Under these circumstances, duplicated stacks characterizing stovepiped applications now morph into a graph, with each node representing a coalesced capability. The capability is implemented as a reusable service. The abstract connectivity of the service components making up an application can be represented as a network—a *service network*. The stovepipes, thus, have morphed into service networks, as depicted on the right side of Figure 1-3. We call these nodes *servicelets*; they are service components designed primarily to be building blocks for cloud-based applications, but they are not necessarily self-contained applications.

With that said, we have an emerging service ecosystem with composite applications that are freely using both internally and third-party servicelets. A strong driver for this application architecture has been the consumerization of IT and the need to make existing corporate applications available through mobile devices.

For instance, front-end services have gone through a notable evolution, whereby the traditional PC web access has been augmented to enable application access through mobile devices. A number of enterprises have opened applications for public access, including travel reservation systems, supply chain, and shopping networks. The capabilities are accessible to third-party developers through API managers that make it relatively easy to build mobile front ends to cloud capabilities; this is shown in Figure 1-4. A less elegant version of this scheme is the "lipstick on a pig" approach of retooling a traditional three-tier application and slapping a REST API on top, to "servitize" the application and make it accessible as a component for integration into other third-party applications. As technology evolves, we can expect more elegantly architected servicelets built from the ground up to function as such.

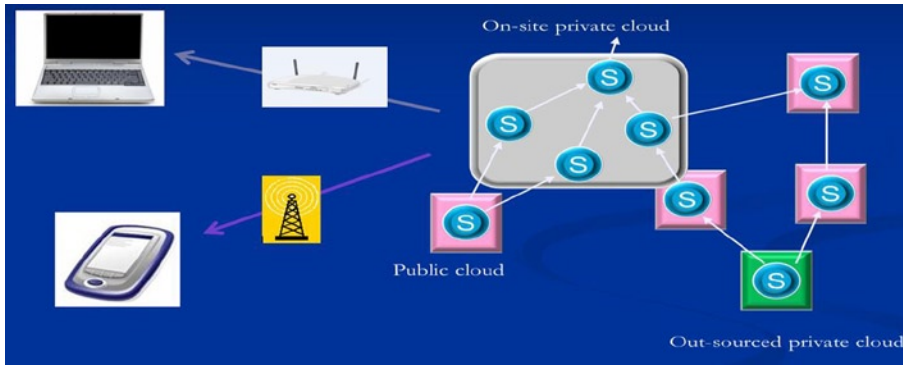

Figure 1-4. *Application service networks*

So, in Figure 1-4 we see a composite application with an internal API built out of four on-premise services hosted in an on-premise private cloud, the boundary marked by the large, rounded rectangle. The application uses four additional services offered by third-party providers and possibly hosted in a public cloud. A fifth service, shown in the lower right corner, uses a third-party private cloud, possibly shared with other corporate applications from the same company.

Continuing on the upper left corner of Figure 1-4, note the laptop representing a client front end for access by nomadic employees. The mobile device on the lower left represents a mobile app developed by a third-party ISV accessing another application API posted through an API manager. An example of such an application could be a company's e-commerce application. The mobile app users are the company's customers, able to check stock and place purchase orders. However, API calls for inventory restocking and visibility into the supply chain are available only through the internal API. Quietly, behind the scenes, the security mechanisms to be discussed in the following chapters are acting to ensure the integrity of the transactions throughout.

In this section we have covered the evolution of application architecture from application stovepipes to the current service paradigm. IT processes have been evolving along with the architecture. Process evolution is the subject of the next section.

The Cloud as the New Way of Doing IT

The cloud represents a milestone in technology maturity for the way IT services are delivered. This has been a common pattern, with more sophisticated technologies taking the place of earlier ones. The automobile industry is a fitting example. At the dawn of the industry, the thinking was to replace horses with the internal combustion engine. There was little realization then of the real changes to come, including a remaking the energy supply chain based on petroleum and the profound ripple effects on our transportation systems. Likewise, servicelets will become more than server replacements; they will be key components for building new IT capabilities unlimited by underlying physical resources.

■ **Note** An important consideration is that the cloud needs to be seen beyond just a drop-in replacement for the old stovepipes. This strategy of using new technology to re-implement existing processes would probably work, but can deliver only incremental benefits, if any at all. The cloud represents a fundamental change in how IT gets done and delivered. Therefore, it also presents an opportunity for making a clean break with the past, bringing with it the potential for a quantum jump in asset utilization and, as we hope to show in this book, in greater security.

Here are some considerations:

- *Application development time scales are compressing,* yet the scope of these applications keeps expanding, with new user communities being brought in. IT organizations need to be ready to use applications and servicelets from which to easily build customized applications in a fraction of the time it takes today. Unfortunately, the assets constituting these applications will be owned by a slew of third parties: the provider may be a SaaS provider using a deployment assembled by a systems integrator; the systems integrator will use offerings from different software vendors; IaaS providers will include network, computing, and storage resources.

- *A high degree of operational transparency is required to* build a composite application out of servicelets—that is, in terms of application quantitative monitoring and control capability. A composite application built from servicelets must offer end-to-end service assurance better than the same application built from traditional, corporate-owned assets. The composite application needs to be more reliable and secure than incumbent alternatives if it's to be accepted. Specific to security, operational transparency means it can be used as a building block for auditable IT processes, an essential security requirement.
- *QoS constitutes an ever-present concern and a barrier;* today's service offerings do not come even close to reaching this goal, and that limits the migration of a sizable portion of corporate applications to cloud. We can look at security as one of the most important QoS issues for applications, on a par with performance.

On the last point, virtually all service offerings available today are not only opaque when it comes to providing quantifiable QoS but, when it comes to QoS providers, they also seem to run in the opposite direction of customer desires and interests. Typical messsages, including those from large, well-known service providers, have such unabashed clauses as the following:

> "Your access to and use of the services may be suspended . . . for any reason . . ."
>
> "We will not be liable for direct, indirect or consequential damages . . ."
>
> "The service offerings are provided 'as is' . . . "
>
> "We shall not be responsible for any service interruptions . . . "

These customer agreements are written from the perspective of the service provider. The implicit message is that the customer comes as second priority, and the goal of the disclaimers is to protect the provider from liability. Clearly, there are supply gaps in capabilities and unmet customer needs with the current service offerings. Providers addressing the issue head on, with an improved ability to quantify their security risks and the capability of providing risk metrics for their service products, will have an advantage over their competition, even if their products are no more reliable than comparable offerings. We hope the trusted cloud methods discussed in the following chapters will help providers deliver a higher level of assurance in differentiated service offerings. We'd like to think that these disclaimers reflect service providers' *inability,* considering the current state of the art, to deliver the level of security and performance needed, rather than any attempts to dodge the issue.

Given that most enterprise applications run on servers installed in data centers, the first step is to take advantage of the sensors and features already available in the server platforms. The next chapters will show how, through the use of Intel Trusted Execution Technology (TXT) and geolocation sensors, it is possible to build more secure platforms.

We believe that the adoption, deployment, and application of the emerging technologies covered in this book will help the industry address current quandaries with service-level agreements (SLAs) and enable new market entrants. Addressing security represents a baby step toward cloud service assurance. There is significant work taking place in other areas, including application performance and power management, which will provide a trove of material for future books.

Security as a Service

What would be a practical approach to handling security in a composite application environment? Should it be baked-in—namely, every service component handling its own security—or should it be bolted on after integration? As explained above, we call these service components *servicelets,* designed primarily to function as application building blocks rather than as full-fledged, self-contained applications.

Unfortunately, neither approach constitutes a workable solution. A baked-in approach requires the servicelet to anticipate every possible circumstance for every customer during the product's lifetime. This comprehensive approach may be overkill for most applications. It certainly burdens with overwrought security features the service developer trying to quickly bring a lightweight product to market. The developer may see this effort as a distraction from the main business. Likewise, a bolted-on approach makes it difficult both to retrofit security on the servicelet and to implement consistent security policies across the enterprise.

One possible approach out of this maze is to look at security as a horizontal capability, to be handled as another service. This approach assumes the notion of a virtual enterprise service boundary.

New Enterprise Security Boundaries

The notion of a security perimeter for the enterprise is essential for setting up a first line of defense. The perimeter defines the notion of what is inside and what is outside the enterprise. Although insider attacks can't be ruled out, let's assume for the moment that we're dealing with a first line of defense to protect the "inside" from outsider attacks. In the halcyon days, the inside coincided with a company's physical assets. A common approach was to lay out a firewall to protect unauthorized access between the trusted inside and untrusted outside networks.

Ideally, a firewall can provide centralized control across distributed assets with uniform and consistent policies. Unfortunately, these halcyon days actually never existed. Here's why:

- A firewall only stands a chance of stopping threats that attempt to cross the boundary.
- Large companies, and even smaller companies after a merger and acquisition, have or end up having a geographically disperse IT infrastructure. This makes it difficult to set up single-network entry points and it stretches the notion of what "inside" means.

- The possibility of composite application with externalized solution components literally turns the concept of "inside" inside out. In an increasingly cloud-oriented world, composite applications are becoming the rule more than the exception.
- Mobile applications have become an integral part of corporate IT. In the mobile world, certain corporate applications get exposed to third-party consumers, so it's not just matter of considering what to do with external components supporting internal applications; also, internal applications become external from the application-consumer perspective.

The new enterprise security perimeter has different manifestations depending on the type of cloud architecture in use—namely, whether private, hybrid, or public under the NIST classification.

The *private cloud* model is generally the starting point for many enterprises, as they try to reduce data center costs by using a virtualized pooled infrastructure. The physical infrastructure is entirely on the company's premises; the enterprise security perimeter is the same as for the traditional, vertically owned infrastructure, as shown in Figure 1-5.

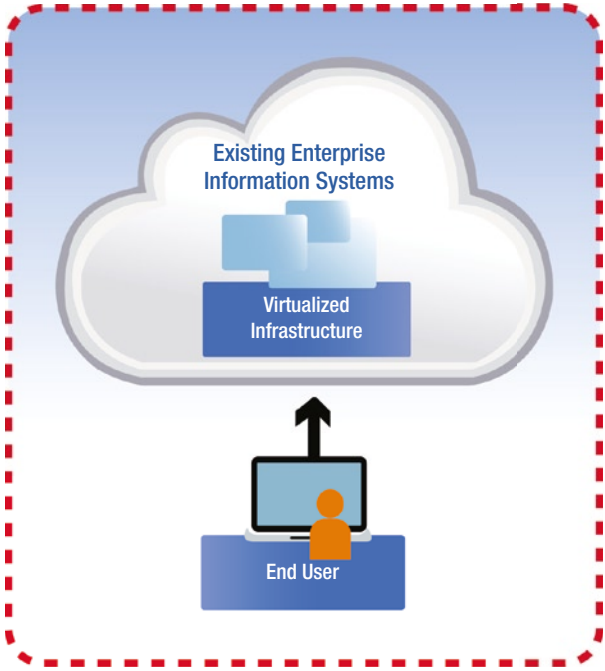

Figure 1-5. *Traditional security perimeter*

The next step in sophistication is the *hybrid cloud*, shown in Figure 1-6. A hybrid cloud constitutes the more common example of an enterprise using an external cloud service in a targeted manner for a specific business need. This model is hybrid because the core business services are left in the enterprise perimeter, and some set of cloud services are selectively used for achieving specific business goals. There is additional complexity, in that we have third-party servicelets physically outside the traditional enterprise perimeter.

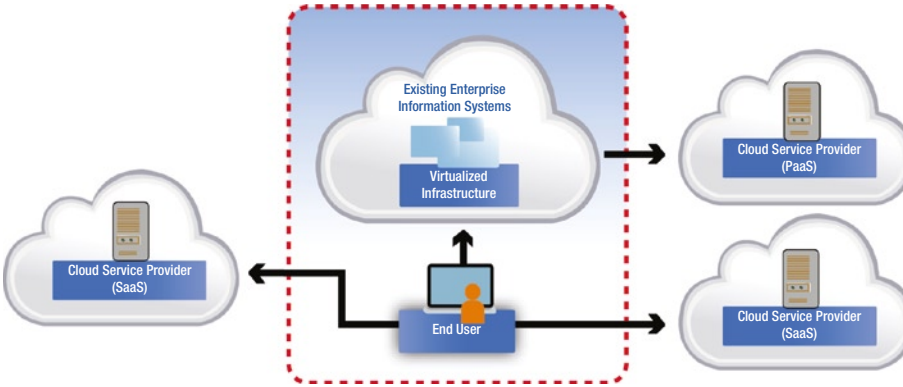

Figure 1-6. *Security perimeter in the hybrid cloud*

The last stage of sophistication comes with the use of *public clouds*, shown in Figure 1-7. Using public clouds brings greater rewards for the adoption of cloud technology, but also greater risks. In its pure form, unlike the hybrid cloud scenario, the initial on-premise business core may become vanishingly small. Only end users remain in the original perimeter. All enterprise services may get offloaded to external cloud providers on a strategic and permanent basis. Application components become externalized, physically and logically.

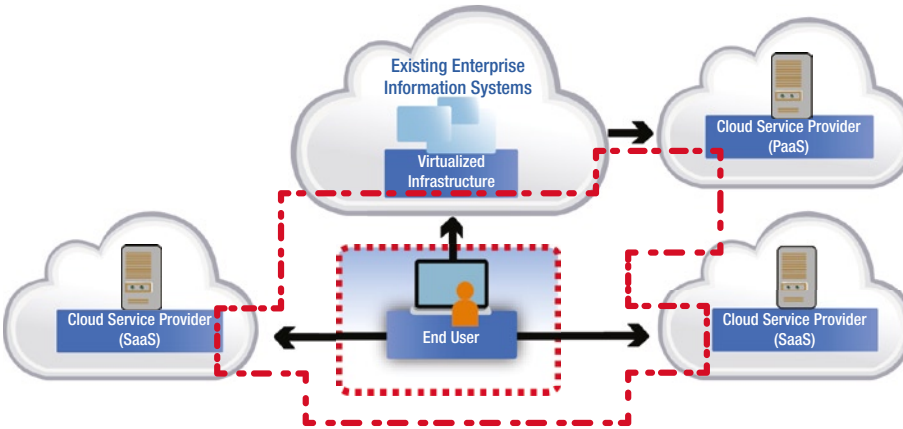

Figure 1-7. *Generalized cloud security perimeter*

Yet another layer of complexity is the realization that the enterprise security perimeter as demarcation for an IT fortress was never a realistic concept. For instance, allowing employee access to the corporate network through VPN is tantamount to extending a bubble of the internal network to the worker in the field. However, in practical situations, that perimeter must be semipermeable, allowing a bidirectional flow of information.

A case in point is a company's website. An initial goal may have been to provide customers with product support information. Beyond that, a CIO might be asked to integrate the website into the company's revenue model. Examples might include supply-chain integration: airlines making their scheduling and reservation systems, or hotel chains publishing available rooms, not only for direct consumption through browsers but also as APIs for integration with other applications. Any of these extended capabilities will have the effect of blurring the security boundaries by bringing in external players and entities.

■ **Note** An IT organization developing an application is not exclusively a servicelet consumer but also is making the company become a servicelet provider in the pursuit of incremental revenue. The enterprise security boundary becomes an entity enforcing the rules for information flow in order to prevent a free-for-all, including corporate secrets flying out the window.

If anything, the fundamental security concerns that existed with IT delivered out of corporate-owned assets also apply when IT functions, processes, and capabilities migrate to the cloud. The biggest challenge is to define, devise, and carry out these concepts into the new cloud-federated environment in a way that is more or less transparent to the community of users. An added challenge is that, because of the broader reach of the cloud, the community of users expands by several orders of magnitude. A classic example is the airline reservation system, such as the AMR Sabre passenger reservation system, later spun out as an independent company. Initially it was the purview of corporate staff. Travel agents in need of information or making reservations phoned to access the airline information indirectly. Eventually travel agents were able to query and make reservations directly. Under the self-service model of the cloud today, it is customary for consumers to make reservations themselves through dozens of cloud-based composite applications using web-enabled interfaces from personal computers and mobile devices.

Indeed, security imperatives have not changed in the brave new world of cloud computing. Perimeter management was an early attempt at security management, and it is still in use today. The cloud brings new challenges, though, such as the nosy neighbor problem mentioned earlier. To get started in the cloud environments, the concept of trust in a federated environment needs to be generalized. The old concept of inside vs. outside the firewall has long been obsolete and provides little comfort. On the one hand, the federated nature of the cloud brings the challenge of ensuring trust across logically and geographically distributed components. On the other hand, we believe that the goal for security in the cloud is to match current levels of security in the enterprise, preferably by removing some of the outstanding challenges. For instance, the service abstraction

used internally provides additional opportunities for checks and balances in terms of governance, risk management, and compliance (GRC) not possible in earlier monolithic environments.

We see this transition as an opportunity to raise the bar, as is expected when any new technology displaces the incumbent. Two internal solution components may trust each other, and therefore their security relationships are said to be implicit. If these components become servicelets, the implicit relationship becomes explicit: authentication needs to happen and trust needs to be measured. If these actions can't be formalized, though, the provider does not deliver what the customer wants. The natural response from the provider is to put liability-limiting clauses in place of an SLA. Yet there is trouble when the state-of-the-art can't provide what the customer wants. This inability by service providers to deliver security assurances leads to the brazen disclaimers mentioned above.

Significant progress has been achieved in service performance management. Making these contractual relationships explicit in turn makes it possible to deliver predictable cost and performance in ways that were not possible before. This dynamic introduces the notion of service metadata, described in Chapter 10. We believe security is about to cross the same threshold. As we've mentioned, this is the journey we are about to embark on during the next few chapters.

The transition from a corporate-owned infrastructure to a cloud technology poses a many-layered challenge: every new layer addressed then brings a fresh one to the fore. Today we are well past the initial technology viability objections, and hence the challenge *du jour* is security, with security cited as a main roadblock on the way to cloud adoption.

A Roadmap for Security in the Cloud

Now that we have covered the fundamentals of cloud technology and expressed some lingering security issues, as well as the dynamics that led to the creation of the cloud, we can start charting the emerging technology elements and see how they can be integrated in a way that can enhance security outcomes. From a security perspective, there are two necessary conditions for the cloud to be accepted as a mainstream medium for application deployment. We covered the first: essentially embracing its federated nature and using it to advantage. The second is having an infrastructure that directly supports the security concerns inherent in the cloud, offering an infrastructure that can be trusted. In Chapter 2, we go one level deeper, exploring the notion of "trusted cloud." The trusted cloud infrastructure is not just about specific features. It also encompasses processes such as governance, assurance, compliance, and audits.

In Chapter 3, we introduce the notions of *trusted infrastructure* and *trusted distributed resources* under the umbrella of *trusted compute pools* and enforcement of security policies steming from a hardware-based root of trust. Chapter 4 deals with the idea of *attestation*, an essential operational capability allowing the authentication of computational resources.

In a federated environment, location may be transparent. In other cases, because of the distributed nature of the infrastructure, location needs to be explicit: policies prescribing where data sets and virtual machine can travel, as well as useful *ex post facto* audit trails. The topic of geolocation and geotagging is covered in Chapter 5. Chapter 6 surveys security considerations for the network infrastructure that links cloud resources.

Chapter 7 considers issues of identity management in the cloud. And Chapter 8 discusses the idea of identity in a federated environment. The latter is not a new problem; federated identity management was an important feature of the cloud's predecessor technology, grid computing. However, as we'll show, considerations of federation for the cloud are much different.

Summary

We started this chapter with a set of commonly understood concepts. We also observed the evolution of security as IT made of corporate-owned assets to that of augmented with externalized resources. The security model also evolved from an implicit, essentially "security by obscurity" approach involving internal assets to one that is explicit across assets crossing corporate boundaries. This federation brings new challenges, but it also has the possibility of raising the bar in terms of security for corporate applications. This new beginning can be built upon a foundation of trusted cloud infrastructure, which is discussed in the rest of this book.