O'REILLY®

Compliments of ORACLE

Migrating Applications to the Cloud

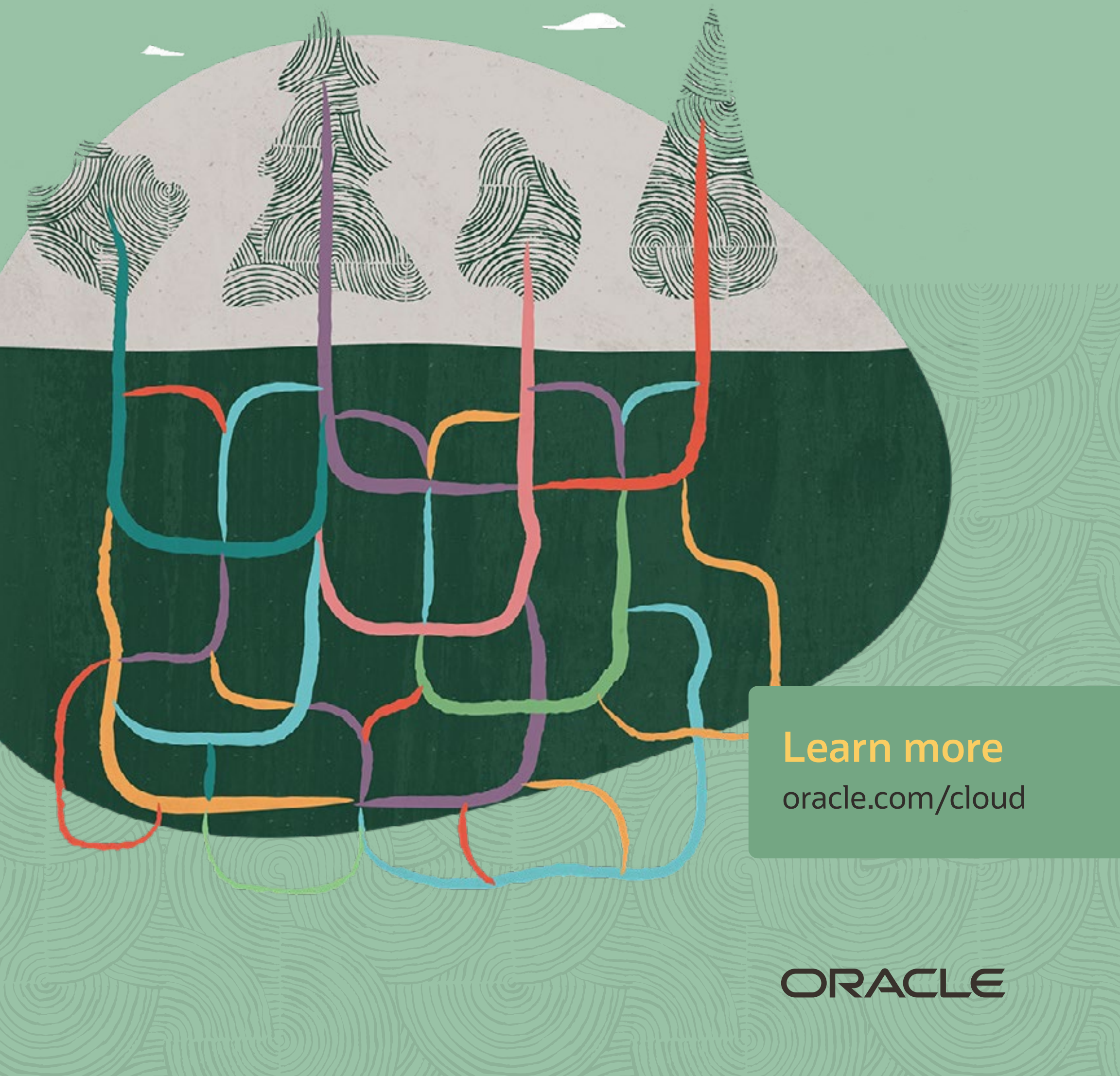
Securing the Value of Digital Transformation for Your Business

Steve Swoyer

REPORT

Oracle Cloud Infrastructure

Delivering advanced levels of automation, performance, scaling and security for your workloads.



Learn more

oracle.com/cloud

ORACLE

Migrating Applications to the Cloud

Securing the Value of Digital Transformation for Your Business

Steve Swoyer

Beijing · Boston · Farnham · Sebastopol · Tokyo O'REILLY®

Migrating Applications to the Cloud
by Steve Swoyer

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Acquisitions Editor: Andy Kwan
Development Editor: Melissa Potter
Production Editor: Christopher Faucher
Copyeditor: nSight, Inc.
Proofreader: Abby Wheeler
Interior Designer: David Futato
Cover Designer: Randy Comer

April 2021: First Edition

Revision History for the First Edition
2021-03-23: First Release

978-1-098-10277-7

[LSI]

Table of Contents

Introduction

Imagine a scenario in which hundreds of millions of people radically increase their consumption of different kinds of goods, products, and services. Demand at this scale is the IT equivalent of an extinction event: customer-facing systems—websites, mobile apps, and call centers—are overwhelmed, as are the core IT systems and packaged applications that power their sales and marketing, inventory management, supply chain management, and procurement processes, just to name a few. If these systems are overwhelmed, so, too, are the database, application server, and other middleware systems that they depend on, along with the custom apps that knit together their business processes.

It is a recipe for disruption, downtime, and customer dissatisfaction all the way down.

The pandemic exposed the essentially reactive posture of on-premises infrastructure; *reactive* in the sense that on-premises infrastructure, unlike cloud infrastructure, was not designed with elasticity and resiliency foremost in mind. It provided a fresh demonstration, if one were needed, of the inherent limitations of this legacy infrastructure model. It exposed the fact that conventional IT infrastructure lacks the practically limitless capacity of the public cloud, the pervasive automation that is characteristic of that environment, the abstraction afforded by the software-defined virtualization of resources, and, not least, the flexibility and adaptability permitted by the pooling and sharing of virtualized resources.

The COVID-19 pandemic demonstrated that conventional IT infrastructure—the on-premises status quo—is holding business back. It highlighted the usefulness of the public cloud as a critical support for business continuity and business expansion in the midst of global disruption and upheaval. At a more basic level, however, it underscored the viability of cloud concepts and the usefulness of cloud infrastructure as a general frame for thinking about and "doing" IT, regardless of context. It was a proof of concept of what advocates of converged infrastructure—a model in which virtualized compute, storage, and network, resources are pooled and shared as needed—have long argued: that the cloud must come to the enterprise…and that the enterprise must come to the cloud.

CHAPTER 1
The Cloud and Business Transformation

This leads us to one of the less obvious facts about on-premises-to-cloud migration, especially with respect to the task of migrating custom-built business applications: the goal of migration is not just to move these apps to the cloud but to transform the organization and its operations. To say this, however, is to put the proverbial cart before the horse; in order to transform the business and its operations, organizations must sometimes transform not only their applications and software architectures (e.g., by shifting apps, services, and their workloads into the cloud) but their custom-built apps and services, too. (In Chapter 5 and the Conclusion, this book addresses the different kinds of transformations—from refactoring or redesigning apps and services to scrapping and rebuilding them from scratch—that an on-premises-to-cloud migration may entail.)

It is necessary to unpack this idea in order to identify a few of the ways in which cloud migration can improve and transform an organization.

In the first place, the migration process gives the organization a chance to address problems with its software infrastructure and with the business functional areas, business processes, and workflows that this infrastructure supports. Maybe the organization is hitting a wall with respect to the capacity limits imposed by its on-premises IT resources. Maybe it is hamstrung in its ability to pursue new business opportunities by the crippling balance of its technical debt. Cloud migration does not magically fix these problems; however, it gives an organization a long-overdue opportunity to address them.

In the second place, the cloud is not just a means of driving down costs or of fixing problems, but of transforming IT operations—and, in the process, transforming business operations, too. The migration process gives business and IT stakeholders an opportunity to talk about what is new, different, and advantageous about the cloud. For example, what business benefits should the organization expect to realize by relocating its business applications to the cloud? How do cloud features such as elasticity, redundancy, security, and the ability to rapidly provision new resources translate into business benefits? What new types of use cases, practices, and users can the business enfranchise by migrating applications to the cloud? The logistics of the cloud open up new opportunities for exposing business functions to customers, partners, suppliers, and other consumers; the cloud's colocality with useful machine learning (ML), artificial intelligence (AI), automated security, data integration, software development, and other services makes it easier for developers to incorporate advanced features into the apps and services they build.

The combination of these benefits makes it possible for the business to develop new products and services, pursue new initiatives, firm up new kinds of partnerships, and pursue other new opportunities for innovation.

This ebook will explore not only the *what* and *how* of migrating business applications to the cloud but, most importantly, the *why*. Enterprises are under enormous pressure to do something strategic about cloud migration. The early cloud use cases—which tended to take the form of tactical migrations, with an emphasis on reducing or eliminating especially costly apps or services—have played out. The priority now is to develop a rational, prospective, purposeful cloud migration strategy; to balance and rationalize investments in cloud infrastructure over and against investments in on-premises infrastructure, and to hash out a strategy in which the cloud—be it the on-premises private cloud, the hyperscale public cloud, the public cloud marketplace, or (a recent innovation) the on-premises public-private cloud—is at least on equal footing with traditional on-premises IT infrastructure.

It is likely that most cloud migration strategies will privilege cloud infrastructure (in one or more of its forms) over and against conventional on-premises IT infrastructure. But the goals, priorities, and content of each of these strategies—the *why*—are and will continue to be wholly unique to each organization. It is up to each organization itself to determine the *why* of cloud migration. This ebook is a resource that would-be migrators can use to formulate their own answers to this question.

Plain Truth About Cloud Migration

Cloud migration is not and probably cannot be a turnkey process; nothing worthwhile ever is.

Homogeneous migrations that entail like-for-like deployments in both the on-premises data center and the cloud are more straightforward than heterogeneous migrations; migrating from Vendor X's packaged enterprise resource planning (ERP) suite to its equivalent cloud service is akin to "lifting and shifting" application and user data from the on-premises environment to the cloud. In such cases, it is usually sufficient to refactor custom-built business apps and services while continuing to use the same packaged application.

On the other hand, *replatforming* (i.e., migrating from Vendor X's ERP suite to Vendor Z's cloud ERP service) is usually more complicated. Customers should expect to redesign—or to rebuild from scratch—custom-built apps, or (alternatively, but not ideal) to create new services that facilitate interoperability between these legacy assets and the new cloud architecture.

The good news is that cloud vendors have a demonstrable interest in facilitating heterogeneous migration scenarios. In practice, cloud vendors offer partially automated tools and services, along with packaged consulting services, that aim to simplify heterogeneous migrations.

This is the good news. The bad news is that not all custom-built applications are designed to integrate and interoperate with packaged ERP suites. In very large organizations, some applications—usually older apps—may have been written in obsolete languages. They might consume data from applications and services that run on mainframes, proprietary systems, and other long-lived platforms. Another complication is that these applications are usually, but not always, written to frameworks and software development kits (SDKs) that have been provided by the software vendors from which the organization licensed its suites. These frameworks and SDKs depend on middleware components (application servers and application- or function-specific databases) that are likewise supported and/or developed by that same vendor.

These applications must “migrate,” too—even if they do not actually move to the cloud. Migration is as much a process of facilitating coexistence between software that lives in the on-premises environment and software that lives in the (public or private) cloud as it is of moving applications and data to the cloud. The irrefragable fact is that a portion of IT infrastructure will remain in the on-premises data center. Some IT infrastructure will be instantiated in conventional nonvirtualized systems, some in enterprise private clouds, and some in public-private clouds. Cloud migration must accommodate these resources, too. This entails managing not only service dependencies across both contexts, but also managing and enforcing information security policies and security dependencies. (As we’ll see, hybrid cloud implementations pose unique security implications of their own.) It likewise involves managing and enforcing consistent data management, data governance, data retention, and other policies.

To get a sense of everything involved here, let’s look more closely at the enterprise software stack.

Minimizing Your Technical Debt

If you imagine that an enterprise software stack comprises a kind of archaeological site, with different objects located at different strata that correspond to different (more or less disruptive) technological shifts over time, you have a feel for what application migration entails: the deeper the IT archaeologist digs, the less familiar the objects they encounter. As they dig their way down to the mainframe systems that constitute the ground floor of the enterprise and its IT history, they are increasingly likely to encounter strange hacks (called “kludges”) and accumulations of software detritus (called “cruft”).

Collectively, these artifacts comprise the technical debt of the organization. Each year, organizations allocate huge portions of their IT budgets—upwards of 50%—to pay down the interest on this debt.[1] Cloud migration is an opportunity to pay down, permanently, at least some of the principal on this debt.

The good news is that cloud providers have devoted a great deal of time and effort to addressing these and similar issues. For example, all providers that market integrated on-premises ERP-like applications suites also market these same suites—or, rather, their equivalents—as cloud services. And most of these providers offer features (for example, service- or platform-specific SDKs or frameworks) and expose functions that subscribers can use to transplant their custom-built, on-premises apps into the cloud context. Some third-party vendors also offer cloud-migration services; some of them are well-known purveyors of enterprise software, some are upstart cloud-first players.

Corralling Security Risk

In the on-premises environment, organizations have a responsibility to safeguard the integrity and consistency of their applications and data. They need to be careful about how they store sensitive data, not only to protect against possible data breaches but to comply with applicable data privacy laws.

Relatedly, organizations have a responsibility to comply with complicated data retention and data deletion statutes and regulations that vary across regions, political unions, and even states within countries. Lastly, organizations need to be able to safeguard access to

1 See CEB Consulting, *Key Findings from the IT Budget Benchmark: 2015–2016*, 2015, 17. CEB found that IT spending on maintenance declined from 63% in 2011 to 57% in both 2014 and 2015. As maintenance costs decreased in 2014 and 2015, CEB showed organizations allocating a slightly larger share of IT spending (33% in both years, as against 32% or less from 2011–2013) to "business opportunity and innovation." (2016 is the last year for which CEB provided data; it was acquired by Gartner Inc. in 2017.) A useful comparison is the US government, which in 2019 allocated fully 80% of its IT budget to "Operations and Maintenance." (A large chunk of this spending was used to support and maintain legacy systems.) In other words, only about 20% of federal IT spend was directed to what budget analysts described as "Development, Modernization, and Enhancement." See *US Government Publishing Office, Efficient, Effective, Accountable: An American Budget—Analytical Perspectives*, February 2018, 223, for more information.

their physical assets—not just server, storage, and network hardware, but the business campus (with its buildings, doors, windows, etc.), too.

In the cloud, many of these requirements either go away or are transformed in ways both subtle and substantive. Cloud providers encrypt all data by default, a practice that is not consistently applied in the on-premises enterprise. Similarly, a cloud best practice is to isolate sensitive from nonsensitive data—another practice that is inconsistently applied in the on-premises enterprise. Cloud database, data integration, data quality, identity and access management, encryption, and similar services incorporate built-in mechanisms for enforcing data governance between and among disparate regions, countries, states within countries, political unions, etc. And once applications and data move to a cloud service, the subscriber is no longer on the hook for securing the underlying hardware and software resources. This responsibility devolves to the cloud provider.

The On-Premises Capacity Conundrum

Scalability is always constrained in the on-premises environment by hard constraints that are less a function of the limitations of physical hardware—in almost all cases, nonvirtualized on-premises hardware is more performant than virtualized cloud resources—but of practical limitations with respect to data center floor space, the density of hardware resources (a maximum number of servers and storage arrays), and, most importantly, IT budget dollars. The upshot is that few organizations have a limitless pool of resources upon which to draw to support applications and workloads, to say nothing of the (ever-increasing) data volumes associated with them.

Cloud resources, by contrast, are virtually inexhaustible; in most cases, for most workloads, subscribers can cost-effectively provision cloud resources to easily exceed the performance of their on-premises environments—with headroom for expansion to accommodate aperiodic spikes in demand. This last point is critical. Vanishingly few organizations are able to respond to drastically changing conditions or unexpected changes in demand by turning on (or rapidly purchasing, configuring, and provisioning) sufficient on-premises compute, storage, and network resources. Converged infrastructure helps in this regard, but, confronted with massive spikes in demand or—just as important—with rebound effects such as the Jevons paradox, it is also insufficient.

To the extent that organizations purchase extra on-premises capacity, they usually do so for two reasons: first, as a means to provision (or augment) existing server, storage, and other sorts of resource pools; or, second, as a hedge against expected future growth. In extreme cases, however, massive aperiodic spikes in demand could easily outstrip the capacity of pooled converged resources. (Many organizations experienced sustained spikes in demand during the recent pandemic event.) Another problem, which contributes to the first, is the Jevons paradox; that is, the tendency for demand to increase in lockstep with an expansion of capacity or as a result of improvements in efficiency. If roads and highways are good examples of this, so, too, is IT infrastructure.

The economics of cloud obviate these concerns. For one thing, providers (not subscribers) are responsible for purchasing, operating, and maintaining cloud infrastructure resources. For another, the cloud leasing model permits subscribers to deduct the total amount of cloud spending for the fiscal year in which it occurs. Conversely, in almost all cases, an organization must depreciate or amortize the cost of its on-premises IT resources over a period of several years.

Most importantly, the cloud permits a resource elasticity that simply has no analogue in the on-premises environment. If a subscriber requires additional capacity, it can rapidly spin up resources in the cloud environment. Another critical difference is that, in the on-premises enterprise, acquiring additional capacity means acquiring some fixed amount of compute, memory, storage, and network resources.

In the cloud, by contrast, virtual resources can scale independently of one another. Best of all, resources in the cloud can be turned on and off—or paused and resumed—as needed. Some cloud providers charge subscribers only for the capacity that they actually use when they use it.

This is a radical departure from the status quo in the on-premises data center. That being said, all departures have costs and benefits, and cloud migration is no different. Before we explore these costs and benefits in detail, it is necessary to clarify what cloud migration is. What does it mean to "migrate" an application, service, or (even) business process to the cloud? For example, has an application or service “migrated” even if, the data or services it depends on continue to live in the on-premises data center? The next chapter explores these and other related questions.

CHAPTER 2

What Is Cloud Migration?

Cloud migration describes the process of moving some or all of an organization's business applications from the on-premises data center to virtual cloud infrastructure or to cloud services of some kind.

Cloud "infrastructure" takes any of several forms. The most well-known is that of the *public cloud*, which describes a cloud infrastructure service that is accessible via the internet and which is usually owned and operated by a "hyperscale" cloud provider. (*Hyperscale* describes the ability of a cloud architecture to scale elastically in response to changing demands—for example, to seamlessly allocate new compute or storage resources as required.) Another common type of cloud infrastructure is the *private cloud*, which is installed in an organization's on-premises data center.

Deployments that span both public and private cloud contexts are called *hybrid* clouds. Today, hybrid clouds constitute the single most common deployment scenario, although cloud-to-cloud (called *multicloud*) and cloud-to-cloud-to-on-premises (*hybrid multicloud*) deployments are also common.[1]

1 See *Nutanix Enterprise Cloud Index: Enterprises Embark on Hybrid IT Journey*, 2020. In this report, 86% of respondents identify hybrid cloud as their "ideal" operating model. Also, in the *2020 Denodo Global Cloud Study*, hybrid cloud deployments comprise just under half (42%) of all cloud deployments, with 18% of respondents pursuing public cloud deployments, 17% private cloud, and 11.1% multicloud. N.B.: Both Nutanix and Denodo are software/services vendors, which—as regards the impetus, if not the results of their studies—could have a bearing on their findings.

One of the most common cloud migration scenarios entails moving a portion of an organization's on-premises applications, services, and data to a public cloud *infrastructure as a service* (IaaS), which is roughly analogous to the on-premises data center. Other common variations are cloud *software as a service* (SaaS) and *platform as a service* (PaaS); these are application and platform services that aim to mask much of the complexity involved in configuring, managing, and maintaining IT resources.

This chapter will explore not only what cloud migration is but also the different types of cloud migration scenarios. It likewise addresses a few of the considerations—both common and esoteric—that would-be adopters should keep in mind as they plan their on-premises-to-cloud migrations.

Relocating Packaged Business Applications to the Cloud

A tendency over the last 15 years has been for organizations to consolidate most of their on-premises business software onto a single vendor's ERP stack or "suite." Today, in virtually all large organizations, on-premises business applications live in a single ERP suite and are licensed as modules. An organization might license modules for finance, HR, inventory, sales, CRM, and other functions.

Migration usually does not entail moving *all* on-premises software and services to the off-premises cloud (i.e., to a public cloud infrastructure service). Rather, a hybrid deployment that spans both the on-premises environment and the public cloud is the most common cloud deployment option.

Hybrid deployments take several different forms. The most common hybrid scenario distributes business applications, middleware, and services between conventional resources that live in the on-premises data center and virtual resources that live in the public cloud. Many hybrid scenarios also involve an on-premises private cloud; that is, the subscriber owns and operates on-premises resources, along with virtualized resources that live in a private cloud. Another variation is the *on-premises data center with full public cloud capability*: a scenario in which a cloud vendor owns and operates virtual cloud infrastructure or services in the subscriber's on-premises data center.

Relocating Custom Business Applications to the Cloud

The typical on-premises packaged enterprise applications system consists of modules for ERP, finance, human resources (HR), supply chain management (SCM), sales and marketing, and other function areas. These packaged applications are textbook exemplars of the familiar 80/20 rule: out of the box, they aim to deliver at least 80% of what the average customer requires; the rub, however, is that it is the customer's responsibility to supply the missing 20%.

To help with this, software vendors offer application frameworks, SDKs, and an assortment of middleware services (e.g., application servers, application-specific database services, etc.) that customers use to build their own business applications. Traditionally, these apps ran in the context of an application server and exploited vendor-specified APIs to exchange data with the core modules of (for example) an ERP suite.[2] A large organization might have built hundreds, perhaps even thousands, of custom applications, as a means to extend the capabilities or functionality of an ERP suite, to support unique or unorthodox implementations of certain business processes, or to support custom processes that crisscross several different business functional areas.

That was the goal, anyway. The reality was usually a little bit (or a lot) messier. In large organizations, it almost never happens that all essential business applications and middleware services are written to a single standard framework or SDK. Some critical business apps and services live on third-party systems, such as mainframes or minicomputers. But some business apps and middleware were built using third-party frameworks or SDKs. This includes software built on top of collaborative suites, directory services, rapid application development (RAD) environments, and so on. (Of these, some proportion were likely written in languages, or designed to exploit runtimes, that are no longer actively developed or supported via long-term maintenance contracts.) Some custom-built software was

2 In the last decade, on-premises functions (and, in some cases, entire applications) have gradually shifted to the cloud, too. The upshot is that some definite proportion of an organization's custom-built applications are already exchanging data with cloud services.

designed for vestigial distributed software architectures—such as component object model (COM) or common object request broker architecture (CORBA)—that have been deprecated in favor of modern simple object access protocol (SOAP) or RESTful design patterns. And in some cases, too, software was written to frameworks or SDKs exposed by other (i.e., third-party) ERP suites.

Maintaining Interoperability Between Cloud and On-Premises Resources

Large organizations usually discover that a portion of on-premises resources cannot be moved to the cloud. In some cases, for example, an organization might be barred by statute or regulation from moving certain kinds of applications—or, more precisely, the data associated with them—to the cloud.

This is one reason that hybrid cloud deployments are so common.

Applications and data tend to stay on-premises for practical or logistical reasons, too. Most large IT organizations are home to a heterogeneous mix of hardware, with racks of 64-bit Intel servers sharing data center space with newfangled ARM servers and the occasional RISC-Unix system, while mainframe systems, black-box proprietary systems, and even obsolete minicomputer systems lurk in dimly lit corners, closets, and raised-floor rooms throughout the data center. The problem is that the applications that live in these systems are compiled to run on different types of CPU instruction set architectures: PA-RISC, POWER, Itanium, MIPS processors, mainframe CMOS processors, function-specific ASICs and FPGAs, and so on. These systems are costly to operate and maintain. But they, along with the applications and services they host, remain in place because they are essential; the organization is not able to transition off of them. Cloud does not—cannot—magically fix this problem.

After all, the easiest way to migrate a mainframe application or service to the cloud is to relocate it to… a mainframe cloud infrastructure service. This is because mainframe CPUs and chipsets are neither virtualized nor emulated in the largest or most common cloud infrastructure services. This is also true, to varying degrees, of RISC-Unix servers, midrange computers, minicomputers, and similar systems.

In most cases, then, these applications and services, along with their data, will remain in the on-premises data center unless and until the organization can wean itself off of its dependence on them. Cloud providers offer subscribers different kinds of tools, features, and services to accommodate them. One new innovation is that of the on-premises public-private cloud—essentially, a dedicated public cloud owned, operated, and managed by the cloud provider, which is installed in the subscriber's own data center. Another option is to host public cloud services in a local "zone" or region, as opposed to hosting them in a regional data center located hundreds or potentially thousands of miles away.

Not all cloud providers support deployments of this kind. But they give would-be adopters additional options with which to accommodate applications, services, and workloads that cannot be moved from the on-premises data center. These services are likewise ideal for high-performance, low-latency, and data residency use cases (as described in the section "Migrating Low-Latency Applications" on page 36).

Ensuring Continuity of Business Processes Between Cloud and On-Premises Resources

Packaged and custom-built applications tend to exchange data with an assortment of infrastructure systems, applications, and services. To migrate some or all of the functions of that ERP system to the cloud is to impact these other infrastructure resources, too. The more important point is that all of these IT infrastructure resources—the ERP system, along with its constitutive modules, complementary apps and services, etc.—serve as a substrate for critical business processes. In practice, the uninterrupted operation of processes such as customer account creation, order creation, credit checking, order fulfillment, etc., depends on the reliability, availability, and performance of the IT infrastructure that undergirds them.

Why does this matter? In the first place, these systems usually host applications, services, and data that are useful to business applications. When these apps move to the cloud, the organization must figure out how to integrate them with on-premises resources that do not always expose RESTful APIs.

In the second place, the applications, services, and data that live in these systems are usually embedded in or consumed by business processes—the same processes that the ERP suite itself is designed to support. In other words, to migrate an ERP system or some of its constitutive functions to the cloud is to impact, and possibly to disrupt, the business processes that depend on those resources.

So applications do not just need to get data from on-premises applications, services, repositories, etc., they need to exchange data with these infrastructure resources, too—sometimes in real time. For example, a sales order nominally originates in sales and marketing. But the process that creates this order—along with a new customer account, if applicable—is rarely confined just to sales and marketing. It might traverse the finance area (should the business offer financing to this customer?), along with, of course, inventory (is the product in stock?). In the background, the process might exchange data with several legacy systems as well as with third-party (external) services.

If the organization expects to move this application to the cloud, it must plan for how to manage interoperability—primarily, data exchange—with on-premises resources, too. It must anticipate how relocating core ERP, customer relationship management (CRM), SCM, HR/human capital management, and other apps (including critical business intelligence and analytic apps and services) to the cloud will impact the business processes in which these apps are embedded, the workflows they provide, and the decision-making activities they support. It must attempt to anticipate the complexity of the migration, too; that is, whether migration involves the straightforward rehosting of custom-built applications and services in the cloud environment or, conversely, whether migration entails a nontrivial replatforming of apps and services, rearchitecting assets to perform as expected in the new cloud environment. (See Chapter 5 for more information about rehosting and replatforming.)

Migration Is What You Make of It

The process of migrating business applications to the cloud is roughly analogous to that of relocating a growing business and its headquarters from a small- or medium-sized town to a large metropolitan or megalopolitan area—a supercity, teeming with people and businesses of all kinds, replete with cultural, economic, educational, and other resources, boasting amenities of every conceivable kind, many in seemingly limitless quantities. But migrating systems and applications—or relocating people and operations—is rarely the sole purpose of a migration or relocation; rather, these purposes are adjunct to some greater purpose. Simply transplanting a business, its people, and its operations into a new ZIP code accomplishes next to nothing. The business must make the most of its new home, avail itself of the opportunities its new location affords, and make use of resources and amenities that are, in effect, colocal with it. This is the promise and the challenge of cloud migration.

CHAPTER 3

The Benefits of Cloud Migration

Cloud migration is not a zero-sum calculus—an either/or choice between on-premises infrastructure, converged or otherwise, and public cloud infrastructure. Rather, it has a pragmatic both/and dimension. Looked at this way, migration gives an organization an opportunity to use public cloud infrastructure as a means to complement or extend on-premises resources and to offset extreme spikes in demand. Other, less concrete benefits include improved flexibility, greater agility, and business resilience. For example, a business that can pragmatically negotiate unexpected or unpredictable conditions is by definition a more agile and resilient competitor.

But cloud migration—and migration to the PaaS cloud, especially—levels the playing field for the business, too. Cloud providers incur the responsibility of not only maintaining but investing in and enhancing their platforms. IT, and IT-driven innovation, is a core competency for cloud service providers. In the last decade alone, hyperscale cloud platforms have been the focus of thousands or even tens of thousands of years of human effort. Cloud service providers have introduced features and capabilities that not only simplify the nitty-gritty of IT operations, but—more importantly—deliver benefits for businesses and their customers. These include:

Potentially faster time-to-market
: IT is all cloud service providers do, all the time. One consequence of this is that cloud service providers are constantly delivering new features and new capabilities. Useful features—such as ease-of-use and ease-of-access enhancements—benefit

not only the IT people who manage cloud resources but the businesspeople (along with customers, partners, and other stakeholders) who use and consume these resources.

Improvements in application performance and reliability
: Upgrade cycles in the cloud occur more frequently; providers regularly replace hardware with new, more powerful, more efficient kits. The upshot is that—even allowing for the virtualization penalty that is a feature, not a bug, of cloud infrastructure—applications and services should perform and behave more predictably in the cloud. Don't forget that cloud architecture is designed for resilience at massive scale; because underlying hardware resources are abstracted by software, problems with these resources can be abstracted—architected around, so to speak—with software, too.

Other improvements include greatly simplified IT infrastructure (subscribers are not responsible for maintaining the hardware and infrastructure software that power cloud services) and improved collaborative capabilities. Public cloud services, in particular, are designed for remote access by default: employees, customers, partners, etc., can access resources from anywhere.

Cloud Infrastructure Breaks the Mold

In a *nonvirtualized* system, compute, memory, storage, and network resources are tightly coupled. The easiest way to increase compute capacity (i.e., "scale out") is to add another server, with its fixed complement of memory, storage, and other resources. Adding new servers is not overly difficult, but it takes time and is not cost-effective. Upgrading existing servers (i.e., "scaling up") is not only time-consuming but usually entails some downtime.

Cloud infrastructure solves this problem. For one thing, cloud architecture constitutes a definitive break with a number of the rules and conventions that have long dominated enterprise computing, particularly with respect to the scaling of systems. In the same way, however, cloud is consistent with accepted ideas with respect to the usefulness of hardware virtualization, system and software automation, and resource maximization. Cloud's emphasis on large-scale virtualization, its dependence on automation and orchestration technologies, and the priority it gives to maximizing the utilization of compute, storage, and other resources—each of these can be traced back to modern mainframe and minicomputer systems.

This chapter will discuss what cloud is and why it is different, as well as introduce and explain key concepts—such as resource decoupling, elasticity, and resilience—that differentiate cloud infrastructure from nonvirtualized on-premises infrastructure.

Elasticity and Pay-for-What-You-Use

Cloud decouples compute, storage, and other resources from one another; this means subscribers can scale out (or scale up) compute capacity independent of storage capacity, and vice-versa.[1] It means subscribers can start, stop, pause, or resume cloud capacity as required—for example, in response to changing demands. This is possible because cloud's compute, storage, and network resources are *virtualized*. Virtual processors can be switched on or off as needed, virtual storage can be added or subtracted, and so on. As far as subscribers are concerned, the provisioning of virtual hardware resources happens entirely in software. It is—quite literally—a matter of configuring a few parameters, clicking a few buttons, and waiting, usually about 30 seconds, for the cloud service to do its thing. In this way, cloud is a radical implementation of software-defined everything; compute, memory, storage, and network resources—even infrastructure itself—are defined and managed by software.

Abstraction via software is what enables the elasticity that makes cloud a transformative option for business customers. Elasticity means that cloud providers charge subscribers based solely on what they use (e.g., ~$0.02 per GB, or ~$100 per month for a virtual

1 For all practical purposes, the term "cloud infrastructure" describes a combined model that emphasizes different kinds of virtualization, software automation, hardware density (from the cloud provider's perspective), and—a function of all of these—resource pooling. That said, several providers offer nonvirtualized "cloud" capacity, too; that is, bare-metal systems running nonvirtualized operating system, database, middleware, application, etc., software. This model is analogous to that of the early application service providers (ASP), which exposed nonvirtualized operating systems, apps, middleware, etc., as "services" that (in the background) were configured, hosted, and managed by the ASP. In practice, bare-metal schemes such as this are useful in cases in which a legacy application, service, workload, etc., will not perform predictably in virtualized infrastructure.

server with 8 CPUs and 16GB of RAM) or on the basis of other, more granular metrics. And abstraction via software also permits cloud providers to offer on-demand capacity, which subscribers can lease temporarily (on a per-minute or hourly basis) to accommodate unexpected demands. Some providers also offer “spot” discounts on capacity.[2]

ML and AI Are at Home in the Cloud

Business applications that live in the cloud are colocal with cloud-based data science, ML, AI, and data integration services. Some of these services are developed and managed by cloud providers. More, notionally, will be developed and managed by a subscriber’s own data scientists, ML/AI engineers, data engineers, and other users. Furthermore, business apps that live in the cloud are likewise colocal with cloud compute and storage resources, which are practically limitless in terms of available capacity. They are colocal, too, with compute engines of all kinds, from general-purpose data processing engines (such as Spark) to SQL query engines and stream processing engines to engines optimized for time-series processing, graph processing, and different kinds of ML processing.

This is beneficial for two reasons. The first is that the cloud model is more consonant with how data scientists, ML engineers, etc., do their work. The second is that the colocality with cloud data science, machine learning, data integration, AI, and other services makes it easier for developers to embed these services in business applications—or to call them from other services, for that matter.

Why is this important? In the first place, the bulk of the data scientist’s work involves obtaining data, preparing data, and performing operations on data, usually in different data processing engines and often by scheduling multiple operations as part of a data flow or “pipeline.” Data scientists, ML engineers, etc., design complex data pipelines that schedule and orchestrate operations on across different data processing engines, as well as move and consolidate data into different repositories, usually preparatory to additional processing. Cloud is useful for precisely this use case.

2 Cloud providers make money by maximizing utilization across all of their virtual resources. If some proportion of these resources is idle, providers offer temporary discounts—so-called “spot” pricing—to encourage uptake.

In the second place, the cloud is not just a playground for hosted ML, AI, and data integration services; rather, data scientists, ML/AI engineers, and data engineers can create their own services—be they function-specific services triggered by events (such as application calls), or ensembles of services that perform complex tasks. Cloud is, by definition, an *as-a-service* paradigm. To this end, most cloud providers offer services, features, and tools designed to simplify the hosting, orchestration, and management of subscriber services. Cloud developer services have the potential to transform how organizations create and maintain apps and services. (For more on this subject, see Chapter 5.)

Cloud-Native Is Native to the Cloud

You may be familiar with the term "cloud-native," which describes an approach to designing and building software that uses container virtualization (i.e., application-level virtualization) and container orchestration to deliver cloud applications and services. Cloud-native design principles are often associated with microservices development, although there is no necessary relation between the two. So microservices are optional, but most cloud-native designs do require the use of a container orchestration platform—such as Kubernetes—to schedule and manage containers.

The logic behind cloud-native design principles is relatively simple: it says that because cloud infrastructure is radically different, it requires a no-less-different way of designing and delivering software. But cloud-native technologies, methods, concepts, and patterns are new and potentially confusing, especially from the point of view of conventional software development. Cloud-native technologies, such as containers and Kubernetes, pose different types of learning curves, both with respect to configuring, deploying, and maintaining these assets (e.g., container design tools and container orchestration platforms) and with respect to using them in practice.

Cloud services provide an easy entrée for organizations that are interested in experimenting with production-ready cloud-native applications and services. Most hyperscale providers offer PaaS-like Kubernetes offerings, for example, along with complementary cloud-native services. (Hyperscale providers offer both KaaS and CaaS—respectively, Kubernetes as a service and containers as a service—cloud PaaS offerings.) These services make it easier to get started with Kubernetes, containers, and cloud-native design because they eliminate the bootstrapping steps—acquiring/provisioning hardware resources, installing and configuring Kubernetes, deploying a container design and management environment, etc.—and also expose ease-of-use features, wizards, and similar guided, self-service features. Another difference with cloud services (versus on-premises infrastructure) is that in the on-premises context, customers would be responsible for maintaining these platforms, too—updating them, upgrading them, resolving broken dependencies, and so on. In the cloud, these responsibilities shift to the service provider. In the same way, and for the same reasons, cloud services should make it easier for organizations to experiment with function-as-a-service (FaaS) or serverless computing, too. (Chapter 5 of this book discusses microservices, Kubernetes, and FaaS in more detail.)

Shared Security Responsibilities in the Cloud

Almost all organizations will discover that security in the cloud environment compares favorably to that of their on-premises data centers. In on-premises environments, data that lives in relational databases and other repositories is not consistently encrypted by default; data that is in-flight may not be encrypted by default, internal systems are not always sufficiently hardened, security patches and firmware updates are not always applied in a timely manner, and password requirements are not always enforced. User accounts are not always updated or deleted—as, for example, when a user changes her role or leaves the organization.

By contrast, most prominent cloud PaaS and IaaS services are certified for compliance with Cloud Security Alliance (CSA), National Institute of Technology Standards (NIST), Federal Risk and Authorization Management Program (FedRAMP), Payment Card Industry Data Security Standard (PCI DSS), and a slew of similar cloud security standards.[3] Providers regularly conduct vulnerability scans, audits, penetration tests, and other kinds of security testing on their systems and the services that they expose. Providers typically encrypt all of the data that is stored (i.e., at "rest") in the cloud; most also encrypt in-flight data—data in transit. Elsewhere, almost all

3 Most business application services boast compliance with dozens of public sector and industry-specific security standards.

providers enforce common-sense security standards, such as password-length, password-aging, and password-reuse requirements, along with multifactor authentication. And most cloud implementations strictly separate virtual machines from one another in order to prevent customers from accessing one another's data. Some cloud services—SaaS and PaaS services, specifically—automatically apply security patches, too. The combination of all of these factors helps to make the cloud a hardened, secure environment for business applications and (just as important) for the data associated with them.

When applications move to the cloud, the infrastructure resources they depend on—the context in which they expect to live—must move with them. The good news is that cloud services usually implement more or less standard versions of open protocols (LDAP, Kerberos, SAML, OAuth, OpenID, etc.) and offer migration services designed to simplify these issues. Thanks to the way single sign-on (SSO) is designed, applications usually do not have to be modified to get them to work in the new cloud context. Instead, IT teams focus on migrating user accounts and data to the user repository in the cloud and configuring the cloud SSO service. The bad news is that cloud migration poses different kinds of security and compliance challenges for custom-built as opposed to packaged applications.

Think about it this way: a vendor that markets both on-premises and cloud versions of its ERP suite has a deep understanding of the idiosyncrasies of its stack; it has the ability to anticipate most of the problems that subscribers will encounter during the migration process. This is also true, to a limited extent, of heterogeneous migrations—that is, migrations that involve replatforming as opposed to rehosting—because cloud providers have an incentive to make it as easy as possible for new subscribers to migrate from their competitors' products to their own services.

This is not necessarily true with respect to custom-built business apps, however. These apps may be designed to exploit proprietary features (for example, vendor-specific directory services and SSO schemes and vendor-specific extensions to security protocols), and will need to be redesigned to work in the cloud. Custom-built apps may also expect to establish stateful connections to stateful resources, which can pose security and performance challenges in the cloud environment. These are not in any sense show-stopping issues. They do underscore the need for subscribers to critically assess their application portfolios—first, to determine which applications must remain on-premises (and, if so, in which kind of on-premises context); second, to identify problems with cloud-bound apps; and third, to develop mitigations or solutions for these problems.

Multicloud for Redundancy

Many organizations employ a multicloud strategy as a hedge against different kinds of risk. One obvious source of risk is that of *cloud service-provider lock-in*: an organization that migrates a large proportion of its on-premises IT resources to a single cloud service provider ipso facto places a large proportion of its proverbial eggs in a single cloud basket. But another source of risk is that of *service outage*: what happens if a provider experiences a catastrophic outage of some sort—for example, failures that cascade across data centers and affect an entire region?

Other risks that organizations can hedge against using multicloud include a service provider going out of business or getting acquired by a competitor.

In multicloud deployments, an organization distributes its IT resources across two or more cloud infrastructure services. There are different ways of doing this. In most cases, the organization uses some combination of load-balancing servers and application servers to distribute demand between different cloud contexts. (This includes the on-premises private cloud, in addition to public cloud infrastructure services.) An organization can opt to architect for redundancy across clouds—that is, run the same PaaS/SaaS software, along with the same IaaS operating systems (OS), databases, and middleware resources in *all* cloud contexts—or split resources between contexts.

Another consideration is that cloud providers—even cloud hyperscale providers—tend to be friendlier with some competitors than with others. It is not unusual for a provider to offer enticements to subscribers that host their applications in a friendly competitor's infrastructure service or consume its PaaS or SaaS offerings. One example is that a cloud provider might charge less for outbound data transfers to a friendly competitor's cloud services. Cloud-to-cloud arrangements can result in significant savings for subscribers, especially in multicloud scenarios.

Cloud and Business Continuity/Disaster Recovery

Organizations opt for multicloud for other reasons, too—business continuity/disaster recovery (BC/DR) planning, for one. Classic BC/DR splits resources between a “hot” site (traditionally, the on-premises data center) and a “cold” site (traditionally, a geographically separate data center). In the past, however, “cold” sites were cold because few organizations could afford to maintain geographically separate, always-on IT infrastructures. But the cloud by definition is an always-on IT infrastructure service; with respect to cloud apps and data, then, splitting BC/DR across cloud infrastructure services helps hedge against catastrophic failure in the cloud context, too.

One obvious problem is that not all on-premises systems, applications, workloads, etc., are able to move to the cloud. Many types of legacy and/or proprietary systems and applications, for example, resist cloud migration; they run on obsolete hardware—such as explicit legacy kit (minicomputer systems, certain kinds of mainframes) —or they are written to deprecated or obsolete application architectures. Certain types of sensitive applications and, especially, data, cannot move to the cloud, either. To the extent that an organization cannot (or, in the case of core mainframe systems, does not want to) divest itself of its dependence on these resources, it must make alternate arrangements for BC/DR. The upshot is that multicloud is a viable option for cloud-to-cloud BC/DR and a possible option for more demanding BC/DR scenarios.

All this being said, cloud migration does not necessarily simplify BC/DR planning; rather, it gives subscribers new flexibility for BC/DR, especially for failing over on-premises applications. (Again, legacy systems will continue to pose problems in connection with BC/DR planning. Organizations should not expect to fail over workloads running on a mainframe or minicomputer system to just any cloud infrastructure service.) One obvious concern is that a subscriber’s PaaS/SaaS provider—or the core cloud infrastructure provider on which that provider hosts its service—could experience a series of cascading failures that impact multiple regions, as happened with a prominent cloud provider just a few years ago. A region-wide outage would affect not only the upstream cloud provider’s own IaaS, PaaS, and SaaS services but those providers that host SaaS and PaaS offerings on its cloud platform. A multiregion failure severely tests the BC and DR use cases; it is extremely unlikely, to be sure, but it poses precisely the kind of problem that BC/DR practitioners must anticipate and for which they must control. After all, in the event—unlikely or not—that a region-wide outage does occur, a subscriber could be ruined not only by loss of revenue but by massive litigation on the part of its shareholders.

Another problem is that failover between dissimilar cloud infrastructure providers—or from PaaS/SaaS services that are offered in multiple cloud infrastructure platforms—is still not a turnkey thing; to the extent that failover scenarios are officially supported, this tends to vary on a provider-by-provider basis. For example, a PaaS SCM provider that offers the same version of its service across multiple hyperscale cloud infrastructure platforms might support failing over (or "pushing") workloads running from one cloud platform to another. However, the ease or extent to which these scenarios can be automated also tends to vary on a provider-by-provider basis.

CHAPTER 4

Building a Business Case for Cloud

The inconvenient truth about cloud resources is that they are inexpensive… until they are not. A large organization that fails to properly anticipate the resources its applications will consume in the public cloud could incur significant monthly charges. This is not just a problem of resources running wild in the hyperscale public cloud. Rather, it stems from an essential difference between the way software performs in the public cloud as distinct to the on-premises data center. As a general rule, software that runs in the public cloud usually requires more resources to perform the same operations than software running in on-premises systems. The good news is that the economics of the cloud model make it possible—and, for many purposes, cost-effective—to solve most, but not all, scaling problems.

Estimating Costs

Cloud service providers offer different kinds of fixed-pricing models. The basic rule is that they charge customers a predetermined price for cloud resources, sometimes for the duration of a lease period, sometimes on a rolling basis (i.e., per minute, hour, day, week, etc., of use).

So far, so good. But different cloud providers offer different types of fixed-price leases, and some fixed-pricing models are more favorable to certain use cases than others.

For example, some cloud providers offer a subscription pricing option in which customers are charged for a definite reserved amount of compute and storage capacity. (Note: This fixed price typically does not include region-to-region data transfer costs or outbound data transfers that leave the provider's cloud infrastructure.) So an IaaS subscriber that reserves 32 virtual servers and a total of 100TB of cloud storage pays a fixed monthly fee for this capacity, regardless of whether it uses all of it. This subscriber must pay a per-GB fee for outbound data transfer, too. Or a PaaS database subscriber pays for a definite reserved amount of compute power and a definite reserved amount of storage. The benefit to the customer is that costs are predictable and—over a long enough lease period—comparatively low. The caveats are obvious, however; the customer must have a high level of trust in the cloud provider as well as an excellent understanding of the IT resources it will require in the cloud.

Some cloud infrastructure providers charge strictly on the basis of usage. To refer to the earlier example, rather than the subscriber leasing a definite fixed amount of capacity at all times (that is, 32 virtual servers and 100TB of cloud storage), it pays only for the capacity it actually uses during the time period that it uses it. In all hosting regions, subscribers will pay more for cloud capacity during the business workday than at night, for example, because they use fewer resources (and so are charged less) after hours. The advantage of usage-based pricing is that subscribers pay only for what they use; the drawback is that usage-based pricing is typically more expensive than subscription pricing. In usage-based pricing schemes, too, subscribers pay a fixed price for outbound data transfer costs.

These are high-level descriptions that mask a great deal of variation and complexity. For example, most hyperscale providers also charge for cloud capacity on a per-service basis: a customer signs up for a relational database service, along with data warehouse, general-purpose compute, general purpose storage, and one or more cloud data integration services. Each of these is exposed (and billed) as a distinct service; each has its own fixed (per-minute, per-hour, per-month) charge. And, again, all providers charge more for data egress than ingress. The problem is that egress charges can be especially difficult to anticipate and/or account for in budgeting. Some subscribers will be surprised if/when the cost of a cloud service exceeds their initial projections. In such cases, data egress charges are often a material contributing factor.

Planning the Migration

An important consideration is that core ERP, SCM, HR, and CRM systems support critical business processes. As we have seen, common business processes layer over and consume functionality from these systems. The upshot is that to move some or all of these systems to the IaaS or PaaS cloud is to impact the business processes that depend on them. It is essential, then, that the migration plan accounts for this impact. In this sense, on-premises-to-cloud migration is not just an IT-driven exercise but, rather, requires organization-wide alignment. The organization must develop and formalize a detailed migration plan to avoid business disruption.

Having said this, migrating packaged business applications from the on-premises data center to a cloud service can be a relatively straightforward process; at a bare minimum, the subscriber *rehosts* —or lifts and shifts—the application and user data associated with these applications to the cloud environment.

This description elides an enormous amount of complexity, however. As noted, most organizations prefer to custom-build business applications that run on top of an ERP, SCM, HR, or other system and its constitutive application modules. For example, a custom-built sales application might exploit functions that are exposed by the ERP system's finance modules. Or a custom-built inventory application might consume functions exposed by the ERP system's finance and sales/marketing modules, as well as consume core functionality provided by a separate SCM system. Another example is a custom application that an employee can use to schedule time off, schedule half-days, or—depending on the organization—adjust their work schedule. These custom-built applications are typically designed to do specific things. To cite a few common use cases, an organization might design custom-built apps in order to extend the capabilities of the core ERP, SCM, HR, or other system's functional areas, or to provide functions that are tailored to a specific business process in a specific organization. Some of these custom-built apps are relatively new, while some are a decade or even two decades old. Is the process of migrating custom-built applications relatively straightforward, too? Is it as simple as lifting and shifting them, too? The simple and unequivocal answer is: it depends.

Prioritizing the Applications and Data to Migrate to the Cloud

Organizations must identify and prioritize candidate applications and/or workloads for cloud migration, as well as determine how rehosting or moving applications and data to the cloud will impact critical business processes and their associated workflows. But they must also determine what, if anything, needs to be done to these apps to permit them to thrive in the cloud.

Along with this, there are several different tactical approaches to cloud migration. These are:

Rehost
: Moving application and user data as-is from the on-premises data center to the cloud.

Replatform
: Adapting applications to thrive in the cloud.

Refactor
: Reengineering applications to scale better, behave more reliably, and improve security in the cloud environment. Some proportion of an organization's custom-built applications will need to be refactored to behave as expected in the cloud.

Rebuild
: Redesigning and rewriting applications from scratch.

Replace
: Some on-premises applications perform functions that are provided by cloud services. It is a safe bet these cloud services are better (more scalable, reliable, and secure) than custom-built apps.

Chapter 5 will explore the challenges involved in refactoring, rebuilding, and replacing applications.

Migration Strategies: Rehosting

Rehosting, otherwise known as lifting and shifting, is straightforward enough: the organization transplants its existing applications and data to the cloud environment. On-premises applications are either virtualized in the IaaS cloud or exposed as SaaS and/or PaaS cloud services. On-premises data is either replicated in virtual databases in the IaaS cloud or exposed via a PaaS cloud service. On-premises storage is configured in different ways: first by reduplicating direct-attached and network storage (such as SAN and NAS) resources, and second by re-creating on-premises network storage, such as NFS and SMB shares, object storage, etc. It is usually practicable to lift and shift a specific vendor's packaged ERP suite from the on-premises environment to that same vendor's cloud service. This is the fastest and most direct route to migrate an on-premises ERP suite to the cloud. What's more, the vendors who develop and maintain these suites usually offer quasi-automated tools and an assortment of migration options, including the on-premises public-private cloud, traffic-prioritized VPN connections between the on-premises environment and the cloud, consulting assessment and migration services, etc. On-premises-to-cloud migration is a hugely disruptive change in this market, with the result that vendors are anxious to retain their existing customers—as well as to poach customers from competitors. This is why most also offer tools, services, and consulting to encourage users of competitive ERP suites to migrate to their cloud services.

Operational applications (i.e., business applications) do not typically generate extremely large data volumes. For this reason, most subscribers will opt to transfer application and user data over the public internet to the cloud environment. Cloud providers offer several options to accelerate this process. For example, some offer secure, traffic-prioritized connectivity between their cloud "edge" locations and the customer's on-premises data center. Some also bundle this connectivity with best-in-class data replication and data synchronization tools.[1] These technologies feature data connectors that are optimized for different kinds of ERP suites, databases, messaging

1 In this context, "best-in-class" is not marketing boilerplate; these are cloud providers that acquired third-party vendors that once marketed standalone data replication and data synchronization products. Some built robust data replication and data synchronization features into their core relational database products.

middleware, mainframe data sources, and a range of vertical-specific data sources. By using a combination of data replication and data synchronization, subscribers can replicate on-premises application and user data to the cloud, as well as synchronize cloud and on-premises resources on an ongoing basis. This ensures the integrity of critical business data along with the continuity of business operations.

Rehosting is the fastest and, notionally, the most direct method of moving on-premises software and data to the cloud. Because of the issues discussed above, organizations usually combine rehosting with other methods, such as replatforming, refactoring, and, in some cases, redesigning software. It is worth emphasizing that the decision to refactor or redesign software is not contingent on a decision to rehost. After all, some custom-built applications will move to the cloud without modification of any kind.

Migration Strategies: Replatforming

Replatforming describes the process of adapting applications to exploit new features, functions, etc., that are specific to the cloud. Replatforming is not a cloud-only play, however. Software that stays in the on-premises data center could benefit from replatforming, too; it can invoke cloud features or functions, exchange data with cloud services, call event-driven cloud services, persist data to cloud storage, and so on. Replatforming is something to think about in connection with all applications, regardless of whether or not they actually move to a cloud context.

Replatforming is an opportunity to make the most of cloud migration. Think of an online sales app that schedules the creation and fulfillment of a web order by invoking functions across different systems that correspond to different business functional areas—sales and marketing, finance, inventory, shipping, etc. This app might likewise make calls to external services (i.e., services exposed by credit-scoring agencies, suppliers and resellers, shipping companies, etc.). In some cases, the app itself might have been written when these external services either did not exist or were not yet exposed as RESTful services. So developers built applications or services that implemented these functions.

In the cloud, external services are complemented by services that either reduplicate most of the functions that applications require (e.g., authentication services, certain kinds of data transformations) or extend this functionality in useful ways. For example, in fulfilling a new sales order, the organization might need to determine if it should offer financing to a customer, especially a new one. This is an analytic question, one that—in the on-premises environment—might be answered by calling an external service to perform a credit check and triggering a stored procedure that runs one or more analytic models in a data warehouse. It is possible to replicate this scheme in the cloud, but it is likewise possible to extend it in useful ways by, for example, invoking an ML service. The colocality of resources in the cloud makes it easier to perform other useful analytic-driven actions at the same time; not just name, address, and payment verification, but also fraud detection and product upsell or cross-sell. If the application is redesigned to expose a RESTful interface, it becomes easier to exchange data with other cloud resources, including partner or supplier systems, third-party resellers, etc.

Migration Strategies: The On-Premises Public-Private Cloud

Cloud infrastructure software uses virtualization to decouple hardware resources from one another. This is a condition of the possibility for the elasticity (i.e., the ability to start or stop, scale up or scale down resources as needed) that is the cloud model's defining characteristic.

But virtual resources in the public cloud are *nonlocal* with one another, too; virtual CPUs and other virtually instantiated hardware resources communicate with one another over network transport, not via local computer buses. So, too, do applications and services. In most cases, the virtual "disks" that a virtual server reads data from and writes data to are actually implemented in object storage. In all cases, network transport comprises the biggest bottleneck in the public cloud. Sometimes, for especially demanding applications, this bottleneck is just too constraining.

This is one of the problems that the on-premises *public-private cloud* aims to address. For event-driven and other types of low-latency applications, performance in the on-premises private cloud may be indistinguishable from that of the on-premises data center. It could, notionally, be better.

The on-premises public-private cloud has other useful applications, too. For one thing, it gives enterprises a viable option for realizing the benefits of cloud in connection with sensitive applications and data that they are unable (or unwilling) to move to the cloud. One benefit of cloud migration is that subscribers are no longer responsible for servicing, maintaining, and upgrading on-premises hardware and software. Another is that cloud subscription services are recorded as OpEx, not CapEx. The on-premises public-private cloud delivers on both of these benefits. The cloud infrastructure provider owns and manages the hardware. If necessary or applicable, it services, replaces, or upgrades this hardware. It exposes the same ease-of-use and ease-of-configuration facilities and wizards in its public-private cloud systems as in its public cloud infrastructure software.

Migrating Packaged Business Applications

The easiest applications to migrate to the cloud are packaged business applications, such as those integrated into an ERP suite. These include:

Financial applications

In large enterprises, accounting and financial management functions are usually provided by a "module" exposed as part of an ERP-like ERP suite. This module also supports the workflows and business processes associated with these functional areas.

The cloud is already a popular destination for financial applications—and for ERP-like suites in general.[2] For one thing, financial applications that move to a cloud service benefit from ML, AI, data transformation, and other colocal services. For another, most cloud ERP-like suites now integrate planning and forecasting capabilities into their finance modules, too.

2 See Chris Pang, John Kostoulas, Neha Gupta, and Supradip Baul, *Market Share: Enterprise Resource Planning, Worldwide 2019* (Gartner Research, May 29, 2020). The report cites 8.8% ERP market growth year-to-year from 2018–2019 and lists a prominent cloud-first vendor among the top three ERP vendors overall.

Human resource information system (HRIS)

In large enterprises, human capital management functions are performed by an HR module that lives in an ERP-like ERP suite. HR systems contain highly sensitive data: the names and social security numbers of employees and their dependents, as well as information about employee salaries, work visas, and sometimes pertinent medical information, too. Even though there usually is no regulatory or statutory reason restricting an HRIS system to the on-premises environment, some organizations prefer to keep it in place.

That said, the cloud is already a popular destination for HRIS applications—a trend that should accelerate over the coming years as companies embed ML and AI services into HR functions.[3]

Customer relationship management (CRM)

CRM was one of the first applications to take off in the cloud, and CRM remains a popular SaaS and PaaS cloud offering. Many large businesses already host their core CRM functions in a cloud service of some kind; however, according to market watcher Gartner Inc., cloud CRM spending accounted for just 20.4% of overall CRM spending in 2019.[4] The upshot is that a clear majority of organizations still own and operate on-premises CRM systems.

All ERP suite vendors offer cloud PaaS suites that are at least equivalent to their on-premises ERP suites. But the cloud teems with cloud-first ERP service providers, too, the majority of which also offer ERP PaaS offerings. The thing is, ERP systems are notoriously

3 See KPMG International, *The Future of HR 2019: In the Know or in the No*, 2018. According to this report, almost one-third (32%) of HR executives invested in cloud HR services in 2017 and 2018, while 49% invested in HR (or "human capital management") software over the same period. For both 2019 and 2020, majorities said that they expected to prioritize HR-related investments in predictive analytics (60%) and enhanced process integration (53%). Almost half (47%) expected to prioritize HR-related AI investments. On balance, executives say that more than one-third (36%) of their HR functions already incorporate AI-like capabilities.

4 See Robert DeSisto, *Software as a Service: Uncertainties Revealed* (Gartner, 2019). DeSisto's comments are specific to *SaaS* CRM, although it is not clear that Gartner distinguishes between SaaS and PaaS in this space. He notes that a single cloud CRM vendor—namely, the earliest and most prominent—accounts for about half of the entire cloud market, as well as almost 15% of all enterprise SaaS spending.

difficult to configure and, in some cases, onerous to change. Cloud migration does not magically fix this, and this has implications for migrating custom business applications, which is addressed in the next section.

In general, homogeneous (like-to-like) migrations are more direct than heterogeneous (like-to-unlike) migrations. This is certainly the case if migrating from physical on-premises infrastructure to the virtual IaaS cloud. But cloud providers, both established ERP-like suite vendors and upstart providers, are especially keen to entice users of on-premises ERP-like suites to migrate to their *PaaS* offerings.

The upshot is that the average ERP-like PaaS provider offers migration tools, features, and services; some even partner with regional and national services firms to offer consulting engagements of one kind or another. However, large cloud providers are able to offer features (e.g., on-premises public-private cloud configurations) that are beyond the reach of most upstarts. In any case, customers should critically evaluate all migration tools, features, services, etc.

Migrating Low-Latency Applications

As a general rule, application performance is inversely related to latency; the lower the latency, the better the performance of the application. In this context, "latency" is an expression of the amount of time it takes for a program to exchange messages (data) with another program.

Latency is not necessarily a function of (higher or lower) data throughput but, rather, of faster or slower input/output performance (in this context, a low-latency application is able to rapidly exchange messages with other applications) and of consistent message delivery. Some applications in specific verticals—finance and telecommunications, for example—have extremely demanding latency requirements. Certain kinds of mainframe applications (some of which are also used in the finance and telco low-latency use cases) likewise have extremely demanding latency requirements, as do a subset of decision-support workloads—for example, those associated with real-time data processing.

Organizations will find it challenging to migrate these applications to the cloud—assuming (with respect to applications that live in mainframes and to proprietary telco systems) that it is even possible to move them. But other types of applications, especially event-driven applications, tend to perform better and more reliably in low-latency contexts, too. This does not mean, however, that they must remain in the on-premises environment—or, more precisely, that these applications cannot move to a cloud context of some kind. In the public cloud, hyperscale providers usually offer different kinds of low-latency options that may (or may not) suit the requirements of a specific application. Again, a growing number of cloud providers now offer on-premises versions of their cloud services, making it possible for an organization to deploy public-private cloud services in its own data center. Not all providers offer this option, and those that do tend to use dissimilar product names to describe the on-premises public-private cloud, but the concept is that of a hardware and software combination that is owned and managed by the provider but that is installed in the customer's own data center.

Another option for latency-sensitive applications is a cloud-based high-performance computing (HPC) service, which typically offers comparatively low-latencies (priced at a premium) relative to standard cloud services. Most hyperscale providers offer HPC services, but PaaS HPC offerings are available, too. These services are nominally designed for scientific computing—their métier is parallel processing on massive data sets—and are normally used to support demanding applications in specific verticals, including oil and gas, financial services, and pharmaceuticals. The tools and features they expose—and their overall feature sets—may be less IT-friendly than conventional cloud computing services. That said, HPC services give organizations another option for hosting low-latency apps and workloads.

Application migration is in no sense a turnkey proposition, but cloud providers do offer a large, and growing, assortment of tools, features, and services (in the form of the on-premises public-private cloud, the availability of "edge" or local public cloud services and the emergence of dedicated HPC cloud services) to ease the transition. Providers can do just so much, however. In many cases—and in basically all large-scale application migration efforts—subscribers must shoulder their share of the work, too. For all practical purposes, this means refactoring, redesigning, and sometimes scrapping older, custom-built applications. The next chapter will explore this process in more detail.

CHAPTER 5

Refactoring, Redesigning, and Rebuilding Applications

Many custom-built applications will move to the cloud without modification, others will require reengineering of some kind, and some will need to be scrapped and rebuilt from scratch.

To reengineer software without changing its design or adding new external functions is to *refactor* it. Refactoring aims to improve the performance, reliability, security, etc., of software. Analogically, refactoring is akin to replacing a set of lower-quality car parts from a suspect manufacturer with better-quality parts from a trusted manufacturer. You might "refactor" your car when you replace its stock suspension with a stiffer performance suspension; the design of the car itself does not change, but the driving experience itself might become better, more enjoyable, for the driver.

Software cannot always be refactored, however. It might be written in a language that is obsolete or little-used or that poses a greater security risk (given the application for which it is designed, the performance that is required of it, etc.) than higher-level languages. It might exploit frameworks, runtime environments, etc., that are obsolete or which—because of concerns about reliability, performance, and security—the organization is loath to replicate in the cloud. Lastly, it might just be poorly designed—or, rather, designed in accord with ideas or patterns in software architecture that have been discredited. (Cargo-cult thinking is as common in software architecture as in any other domain.) In such cases, developers determine to scrap the application and rewrite it from scratch, preferably using languages, tools, libraries, services, etc., that are exposed in the cloud environment.

Refactoring for Cloud Migration

In some, if not most, cases, developers must refactor or reengineer custom-built applications to get them to run as expected in the cloud environment. For applications that are designed in accordance with a vendor-supplied framework and SDK, this can be a surprisingly straightforward process.

Applications that are good candidates for refactoring include:

Resource-intensive applications
: Poorly designed software consumes a disproportionate share of available resources. In the cloud environment, resource-intensive applications will incur excess charges.

Applications that scale poorly, have spotty availability, etc.
: An application that scales poorly in the on-premises data center will behave exactly the same way in the cloud.

Applications that depend on custom-built services
: For example, an application might call other services—or another custom-built application—to perform a credit check or query a third-party service to determine the availability of a product. Developers can refactor applications to call third-party services directly or, alternately, develop coarse/finely grained services that provide essential functions. Legacy apps and services that used to provide these functions can be retired.

Under certain conditions, it is cost-effective to refactor these applications to address performance, availability, security, and other issues. This is the case if an application implements certain kinds of functions in software—data transformations, for example—that are better implemented by modern standalone libraries, cloud services, or microservices. In the same way, the application might depend on deprecated (or obsolete) functions, runtime features, or software libraries. Refactoring it to exploit modern implementations could reduce its consumption of CPU, memory, or network resources, as well as improve scalability, availability, and security. On its own, refactoring the application to use the fit-for-purpose libraries could go a long way to addressing performance, reliability, security, and other issues.

Redesigning or Rebuilding Applications for Cloud Migration

As a basic rule, if an application is written to and compliant with a vendor-supplied framework or SDK—even an older, supported framework, provided it has an upgrade path to a current version—it is an ideal candidate for refactoring. Java Enterprise Edition (Java EE) or Jakarta Enterprise Edition (Jakarta EE) applications designed to run in a Java application server are good examples of this. Refactoring applications written for older, supported Java/Jakarta EE frameworks to run in the cloud is usually a relatively straightforward process. However, applications written to frameworks or standards that are no longer supported will probably need to be redesigned and rewritten from scratch. Ditto for applications written in languages that are deprecated or difficult to maintain or that (as with low-level languages such as C and C++) could pose heightened security risks in the cloud environment.

One solution to this problem is to migrate custom-built apps to the cloud by reinstating (and, if necessary, refactoring) them in a virtual context (e.g., an on-premises private cloud, the IaaS public cloud). The long-term goal should be to retire these problem custom-built apps and to replace them with apps designed in accordance with modern tools, concepts, and methods. Hosting them in a virtual context is only a band-aid—a kludge—and, over time, amounts to throwing bad money after good.

A common-sense practice is to maintain these resources in situ in the on-premises data center and to prioritize redesigning them (and rewriting them from scratch) for the cloud environment. Again, to migrate obsolete, unstable, or poorly performing software to the cloud is to perpetuate technical debt—to squander time and effort (and to take on new technical debt) maintaining legacy code in the cloud.

Microservices, Serverless, and Other Approaches to Migrating Applications to the Cloud

For better and worse, we are accustomed to designing with "monolithic" applications that incorporate hundreds or thousands of functions. Microservices architecture is a design pattern that says that an application can be *decomposed* into its constitutive functions; that is, granular units that correspond to simple, specific (i.e., primitive) functions. The basic idea is that microservices can be "orchestrated" in ensemble combinations that approximate the features of a monolithic application. This logic is squarely in the tradition of minimalist software development philosophies, like Unix, which hold that programs should be designed to perform specific functions—and that developers should create new programs in order to support new specific functions. It is a refinement of the same logic that undergirds a similar but different paradigm—namely, service-oriented architecture or SOA—with the qualification that microservices design aims to produce stripped-down programs (ideally, functional primitives) that do very basic things. Software monoliths are notoriously difficult to understand and maintain; microservices, like core Unix system programs, aim to be simple and easy to maintain.[1]

The salient point is that microservices need not be an all-or-nothing design pattern. An organization need not commit to the strict definition of microservices architecture in order to make use of microservices-like concepts and methods; microservices need not be stripped down to absolutely basic functional primitives. For many potential adopters, the formal concept of the microservice as such is less useful than the germ of the idea, which says, in the first place, that larger applications can be composed out of multiple constitutive services. Implicated in this is the idea that larger applications—including monolithic apps—can invoke external services (including microservices) to improve or to extend their own functions. In the same way, and for the same reasons, developers need not design and instantiate all of their "applications" as composable microservices.

1 As with all analogies, this one breaks down to the extent that there are specific challenges bound up with building, managing, and especially scaling distributed microservices that are just not applicable to Unix system software development.

To sum up: microservices-like concepts are applied in dissimilar software patterns and regimes—for example, as a means to create less granular services or even in the context of monolithic development.[2] As discussed in Chapter 3, microservices are also strongly associated with cloud-native design. This association is not absolute, however; an organization can develop and use microservices without practicing cloud-native design.

With respect to the task of migrating custom-built applications to the cloud, cloud-native design principles and microservices design patterns can be useful for several reasons. First, developers can use the tools, technologies, frameworks, etc., that they prefer to work with to design simple services or granular microservices that perform specific functions. These services or microservices can be called from different kinds of apps and/or other services, including monolithic apps. For example, a microservice that performs a specific kind of data transformation—converting strings encoded in one standard (Unicode) to another (Python)—is relatively simple to create and instantiate. Second, an application might call several services and microservices to perform a sequence of elaborate transformations. Third, there is no hard-and-fast rule that services and/or microservices must be instantiated in containers and managed by a container-orchestration platform. For example, developers can write either services or microservices in Java and run them in an application server, queue them to run in Spark (or another compute engine), or design them to run in a serverless (i.e., FaaS) context.

Speaking of which, some hyperscale providers also host serverless computing services. Serverless abstracts the logic of microservices to the nth degree. Serverless applications do not "live" anywhere, per se; they are not compiled into executable files, libraries, etc. Unlike conventional apps and containers, they do not expect to run in an operating system context, virtual or otherwise. Typically, they consist of application logic that is managed by a FaaS runtime. The canonical serverless "application" is an event-driven function; it is triggered in response to an event—an application call, for example—of some kind. Serverless functions come into being, do their thing, and usually terminate and go away—all in the space of a few

2 See Steve Swoyer and Mike Loukides, *Microservices Adoption in 2020* (O'Reilly, 2020) for more information.

milliseconds. From the point of view of on-premises-to-cloud application migration, the same type of simple, function-specific service that a developer might design according to a microservices pattern can also be designed as serverless function.

Overall, it is easier to experiment with serverless computing and microservices architecture—including containers and container orchestration—in the cloud context. The necessary infrastructure, so to speak, is already there, and the cloud service provider, not the subscriber, is on the hook for managing it, updating it, etc. For that matter, the cloud is colocal with development services of all kinds. Hyperscale providers offer different kinds of HTTP-based services that aim to simplify the design, deployment, and hosting of RESTful apps. These services work with popular integrated development environments (IDE) as well as with other cloud services, cloud-based or standalone text or source-code editors, interactive notebooks, and so on. One benefit for developers is that they are fully managed; the cloud provider assumes responsibility for patching and upgrades, as well as for the availability and continuity of the apps hosted on the service. Most hyperscale providers also offer different kinds of database services, which are essential for software development.

Hyperscale and other providers also offer so-called "low-code" development services. These are in the tradition of RAD tools, although—unlike RAD tools—they are typically designed for less-skilled programmers. The apps themselves are hosted and managed by a low-code development platform (LDCP) service. The low-code development experience marries guided self-service capabilities with model-driven logic. Like microservices, FaaS, and other patterns or methods, low-code is not a software development panacea; rather, it is one of a number of modern developer-oriented services that are available to use—or experiment with—in the cloud environment.

Conclusion

Cloud migration is inevitable. This does not mean that all on-premises IT resources will move to the public cloud. It does mean that a majority of on-premises resources will move to a cloud context of some kind, be it an on-premises private cloud, a public cloud infrastructure service, public PaaS or SaaS services, or some melding of the two, such as an on-premises public-private cloud.

Enterprise applications are no different. The functions that on-premises ERP, SCM, HR, CRM, and other systems provide will move—in whole or in part—to ERP, SCM, HR, CRM, and other systems in the cloud. Count on it. So it is that the impending migration of these systems and the applications they support to a cloud context comprises a once-in-a-generation opportunity for vendors of every stripe—and not just for established on-premises players, but also dozens of different cloud-only providers. Under normal circumstances, it is staggeringly difficult to convince a large organization to move off of its core enterprise applications stack onto a comparable stack that is provided by another vendor. The systems themselves are so complex and so thoroughly interpenetrated with business operations as to constitute contextually immovable objects.

But cloud migration is an exception; it is a once-in-a-generation opportunity for transformation—for customers and vendors alike.

Cloud Is a Model of Shared Opportunity—and Responsibility

Cloud's cost advantages are impossible to ignore, as are its advantages with respect to IT agility and flexibility, to say nothing of its role as a catalyst for business transformation. The logic of cloud, with its emphasis on the abstraction, decoupling, and pooling of resources, has already helped purge IT of prejudices and shibboleths that had constrained its—and the business's—freedom of action since the earliest days of information technology.

To be sure, problems such as technical debt do not disappear in the cloud; in the cloud, as in the on-premises enterprise, an organization that designs and maintains applications or services of its own will incur technical debt, full stop. However, the interest payments on this debt are, in a sense, halved. The cloud model is one of shared responsibility. The cloud service provider, not the subscriber, is on the hook for securing, managing, and maintaining infrastructure hardware and software; subscribers have their own responsibilities with respect to defining (and enforcing) best practices with respect to information security, data governance, and other essentials.

In the same way, it is incumbent upon subscribers (or architects and developers) to refactor and, if necessary, rewrite apps and services to comport with whatever changes cloud providers introduce. (In the cloud, as in the on-premises enterprise, to introduce a new API and to deprecate an existing API is to break stuff.) These problems do not disappear in the cloud. But a big chunk of the problems that are bound up with the software development life cycle in the on-premises environment do cease to be problematic.

And, yes, there are risks. We have seen this on the consumer side, as, for example, when a cloud provider has eliminated a (usually free) cloud service it once touted with a surfeit of hype or fanfare. But there are good reasons to believe that things will be different in the case of enterprise-oriented cloud services, starting with the leasing contracts, service-level agreements (SLAs), and even terms-of-service that cloud providers offer to their enterprise customers. Other risks include the possibility that a cloud service provider could fail, that it could get acquired by one of its competitors, or that—owing to a combination of factors—the cloud hosting experience could change for the subscriber. But these risks are commensurate with similar risks in the on-premises environment, where vendor lock-in, the acquisition of one software vendor by another, the elimination of favored products or product features, etc., are known risks. In both contexts, organizations employ strategies to hedge against risk and to mitigate the fallout of unexpected events. Cloud does not give subscribers license to shrug off real-world risks.

Overlooked Factors for Success in Cloud Migration

Organizations must consider several important factors prior to actually grappling with the nitty-gritty of migrating apps to the cloud. For example, in the case of PaaS and SaaS cloud services, is the service offered across more than one cloud service provider? Is the service offered across multiple national or international regions? Is the same version of the same cloud service available in two or more regions in the same country? In regions around the world? Does the cloud service provider guarantee the same experience (i.e., security controls, standards compliance, performance, features, etc.) across all regions? Even among hyperscale cloud providers, the availability of services and the consistency of the service experience still tends to vary from region to region. (This is true even of hyperscale cloud providers in the United States; not all services are consistently available in all regions.) Most hyperscale providers usually operate more than one data center—sometimes called "[availability] zone"—in each region. Regional coverage is important. For one thing, the closer a subscriber is in geographical proximity to an availability zone in a region, the faster and more consistent the delivery of their services. For another, the availability of infrastructure and services across regions is important for BC/DR; if a specific region experiences a catastrophic failure, service delivery should fail over to an adjacent region. Providers also usually offer edge locations (i.e., local caches of data and content) in major metropolitan areas in all of the regions in which they operate data center facilities. Local caching helps reduce latencies for frequently accessed data; it also enables cloud providers to meet tighter SLAs.

Speaking of which, SLA options vary widely across cloud service providers. In addition, large customers contemplating significant multiyear contracts have considerable leverage when it comes to negotiating SLAs, up to and including less aggressive versions of the multilevel SLAs that are typical in the enterprise. At a bare minimum, cloud SLAs should clearly stipulate that subscribers own their application and use data as well as their custom-built apps. Some providers offer customer-based SLAs, which guarantee that all subscribers will have the same levels of performance and availability for a specific type of cloud service. Hyperscale providers offer customer-based service-level tiers that specify performance and availability criteria and are usually warranted for specific latency baselines. Some cloud providers offer customer-friendly terms for data ingest (with respect to initial bulk loading and ongoing bulk data movement) and data egress (i.e., outbound data) especially for data transfers between and among intracloud regions and/or supported third-party cloud infrastructure services. With respect to the performance of critical systems (such as databases, data warehouses, and enterprise applications) in the cloud, PaaS providers tend to offer SLAs that are more customer-friendly than those of IaaS SLAs. The reason for this is simple: in the PaaS model, the provider itself is responsible for managing and scaling the service (as well as the underlying virtual resources that power the service), which makes it possible for PaaS providers to support more granular performance/availability criteria. In the IaaS model, by contrast, subscribers are required to configure, maintain, and scale virtual resources and software.

Cloud Infrastructure Is Evolving—and Improving—at a Breakneck Pace

The issue of SLAs gets at something else, too: namely, the inexorable improvement of cloud infrastructure services, especially of PaaS infrastructure. Cloud providers continue to focus on the weak points of the cloud paradigm, especially with respect to performance and to the sub-optimal latencies of cloud resources. The availability of low-latency cloud PaaS services, along with the emergence of dedicated HPC cloud services, are good examples of this. But the PaaS cloud especially continues to evolve, and improve, in other ways, too. Today, for example, PaaS services expose GUI-based wizards and guided self-service features that aim to simplify (or, if practicable, automate) tasks that consume a disproportionate amount of time or require significant labor in the conventional data center environment. PaaS cloud services also make use of ML-powered, rule-driven automation to monitor performance, detect issues, and trigger one or more predefined actions. A large PaaS provider might host thousands or even tens of thousands of subscribers; this gives it a huge data set with which to train the ML models used to power rule-driven AI remediations. This same data will also hold valuable clues as to generalizable tasks, processes, or use cases that cloud providers can automate (via rule-driven AI) or simplify (via ease-of-use wizards).

The point is that the distinctive strengths of the PaaS model—the number and variety of its ease-of-use features and amenities and the sophistication of its ML-powered, rule-driven automations—will improve over time. The PaaS cloud of 24 months from now will boast lower latencies, superior elasticity, more granular provisioning capabilities (that is, with respect to the allocation or deallocation of compute, memory, storage, and other resources), and improved "smart" automation features. Count on it.

The Cloud Is a Different Kind of Playing Field

As a kind of thought experiment, imagine the cloud as an impossibly massive sporting complex for hosting different kinds of services. Analogously, it supports an inexhaustible supply of soccer, baseball, football, field hockey, and lacrosse fields; a no less inexhaustible supply of basketball, tennis, and volleyball courts; limitless capacity for ice-skating, ice hockey, and curling rinks; potentially billions of gallons of capacity for swimming, diving, and similar aquatic sports.

The point is that all sorts of people with different interests, needs, priorities, pursuits, and—yes—passions can collaborate in and/or make use of both the capacity and the amenities of the cloud environment. This book focused on the challenges and the opportunities associated with migrating on-premises business applications and services (both prepackaged and custom-built) to the cloud. Once an organization transplants these apps and services into the cloud environment, they "live" in an essentially transformed context—a context in which other apps and services are at once adjacent and, as it were, available: callable, invokable, easily consumed.

The thing is, the terms of use (so to speak) of these services differ radically from those of their on-premises equivalents. Many of the responsibilities that complicate the creation, deployment, management, and scaling of apps and services in the on-premises environment (particularly with respect to ongoing maintenance) disappear in the cloud. To cite one example: many cloud providers expose their ML, AI, and developer services as managed platforms. This means that—in contradistinction to a conventional application server, a service-orchestration platform, or any similar platform in the on-premises context—cloud subscribers are not responsible for provisioning the infrastructure hardware or maintaining the infrastructure software that hosts and orchestrates the services they develop. This is the stuff of radical transformation. The efficacy, agility, and scalability of prior service-oriented regimes were always already constrained by the requirement that IT itself purchase, operate, and maintain infrastructure hardware and software. And the maintenance of infrastructure software, in particular, was a huge pain point (and source of technical debt) for IT on an ongoing basis.

Final Thoughts

The stresses induced by the pandemic demonstrated the essential fragility—the conspicuousness—of conventional IT infrastructure. Compared to the plasticity of cloud infrastructure, on-premises IT infrastructure is analogous to a proverbial glass house. By turns intricate, involute, and baroque—sporadically elegant, even—it is inescapably brittle. If or when conditions change, conventional IT infrastructure has little margin for shrinking or stretching; stress it too much, and it must shatter.

This is not to reject the viability of on-premises IT infrastructure; it is, rather, to make two points: first, that on-premises IT infrastructure must evolve to mirror cloud infrastructure (e.g., by emphasizing pervasive automation, increased hardware density, and large-scale virtualization, especially with respect to the abstraction and pooling of resources); second, that a focus solely on cloud migration misses this point. Cloud migration is not a one-and-done, one-way-only affair; it is conceivable, after all, that workloads and processes that move to the public cloud could, under certain conditions, come back again—for example, to run on virtualized resources in the context of on-premises converged infrastructure. (In its most basic sense, "converged infrastructure" describes a model in which compute, storage, network, and other resources are pooled such that they can be allocated dynamically as needed.) It is likewise conceivable that workloads and processes could shift back and forth between different cloud contexts: the public cloud, the on-premises private cloud, and on-premises environments with public cloud capabilities.

To return to the prior example, converged infrastructure is inclusive of both on-premises and public cloud infrastructure. The practical benefits of convergence—the improved scalability, availability, and resilience of IT resources—are the same regardless of context. These benefits are strongly associated with cloud service providers, but the outsized success of the public cloud obscures the important point that cloud infrastructure is an organic outgrowth—an evolutionary enhancement—of the IT organism itself. What we are describing, then, is less a process of migrating business operations to the cloud than one of incorporating the cloud into business operations. Both the public cloud and the on-premises enterprise are sites of convergence; both converge upon, and with, one another.

An organization is not riven in half if it extends its IT infrastructure and operations to the cloud context; rather, it becomes a new, different whole. The cloud (i.e., cloud concepts, cloud-like infrastructure) has a home in the on-premises enterprise, too. A cloud strategy must keep this in mind. The point of extending one's operations to the cloud—public or otherwise—is not (just) to slough off legacy, costly, or unreliable resources; nor is it (just) to better position oneself to navigate shocks or disruptions. And it is not (just) adjunct to a strategic effort to improve or transform the business. These are all benefits of cloud (i.e., of the incorporation of cloud concepts and cloud-like infrastructure into the organization and its operations). But the reason for doing this is that cloud concepts and cloud-like infrastructure are an organic part of, are bound up with, the growth and development of the enterprise itself.

About the Author

Steve Swoyer is a writer, researcher, and analyst with more than 20 years of experience. His research focuses on business intelligence, data warehousing, and analytics, as well as edge issues in data science, machine learning, and artificial intelligence. Steve enjoys researching and writing about emerging trends and potentially transformative ideas in systems design and architecture. As an analyst and researcher, Steve explores trends in distributed systems architecture, cloud native architecture, and other emergent subject areas. He is a recovering philosopher with an abiding interest in ethics, philosophy of science, and the history of ideas.