A SHORT INTRODUCTION TO CLOUD PLATFORMS

AN ENTERPRISE-ORIENTED VIEW

DAVID CHAPPELL

AUGUST 2008

SPONSORED BY MICROSOFT CORPORATION

COPYRIGHT © 2008 CHAPPELL & ASSOCIATES

CONTENTS

DEFINING TERMS: WHAT IS A CLOUD PLATFORM?

The coming shift to cloud computing is a major change in our industry. One of the most important parts of that shift is the advent of *cloud platforms*. As its name suggests, this kind of platform lets developers write applications that run in the cloud, or use services provided from the cloud, or both. Different names are used for this kind of platform today, including *on-demand platform* and *platform as a service (PaaS)*. Whatever it's called, this new way of supporting applications has great potential.

To see why, think about how application platforms are used today. When a development team creates an on-premises application (i.e., one that will run within an organization), much of what that application needs already exists. An operating system provides basic support for executing the application, interacting with storage, and more, while other computers in the environment offer services such as remote storage. If the creators of every on-premises application first had to build all of these basics, we'd have many fewer applications today.

Similarly, if every development team that wishes to create a cloud application must first build its own cloud platform, we won't see many cloud applications. Fortunately, vendors are rising to this challenge, and a number of cloud platform technologies are available today. The goal of this overview is to categorize and briefly describe those technologies as they're seen by someone who creates enterprise applications.

CLOUD PLATFORMS IN CONTEXT: THREE KINDS OF CLOUD SERVICES

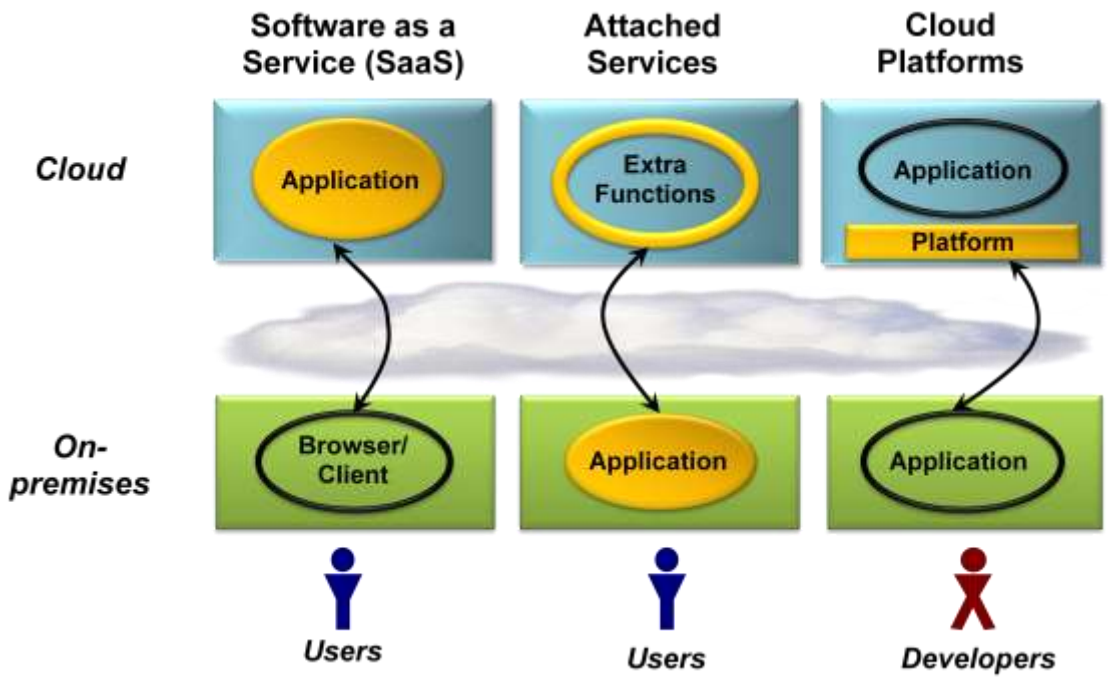

Figure 1: Cloud services can be grouped into three broad categories.

To get a grip on cloud platforms, it's useful to start by looking at cloud services in general. As Figure 1 shows, services in the cloud can be grouped into three broad categories. Those categories are:

- Software as a service (SaaS): A SaaS application runs entirely in the cloud (that is, on servers at an Internet-accessible service provider). The on-premises client is typically a browser or some other simple client. The most well-known example of a SaaS application today is probably Salesforce.com, but many, many others are also available.

- Attached services: Every on-premises application provides useful functions on its own. An application can sometimes enhance these by accessing application-specific services provided in the cloud. Because these services are usable only by this particular application, they can be thought of as attached to it. One popular consumer example of this is Apple's iTunes: The desktop application is useful for playing music and more, while an attached service allows buying new audio and video content. Microsoft's Exchange Hosted Services provides an enterprise example, adding cloud-based spam filtering, archiving, and other services to an on-premises Exchange server.

- Cloud platforms: A cloud platform provides cloud-based services for creating applications. Rather than building their own custom foundation, for example, the creators of a new SaaS application could instead build on a cloud platform. As Figure 1 shows, the direct users of a cloud platform are developers, not end users.

Understanding cloud platforms requires some agreement on what the word "platform" means in this context. One broad way to think about it is to view a platform as any software that provides developer-accessible services for creating applications. The next section looks at this idea in a bit more detail.

A GENERAL MODEL FOR APPLICATION PLATFORMS

Our experience with application platforms today comes mostly from on-premises platforms. A useful way to think about cloud platforms is to see how the services an application developer relies on in the on-premises environment translate to the cloud. To help do this, Figure 2 shows a general model that can be applied to both worlds.

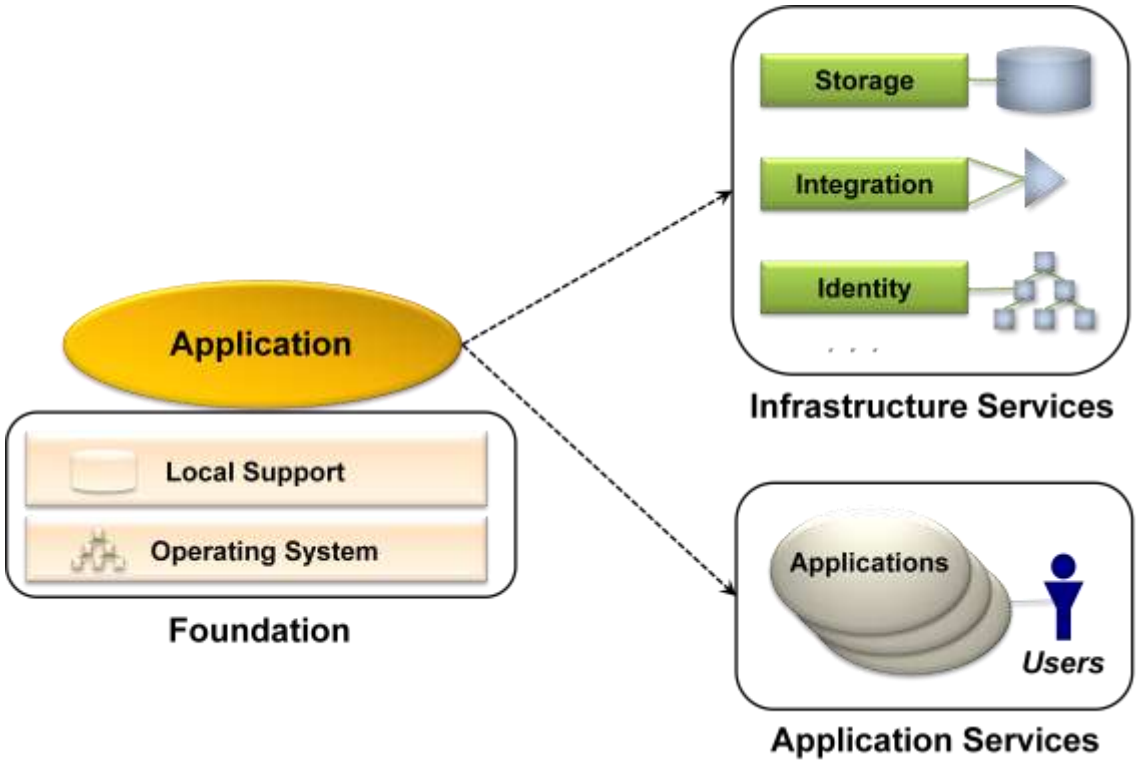

Figure 2: A modern application platform can be viewed as having three parts.

Whether it's on-premises or in the cloud, an application platform can be thought of as comprising three parts:

- A *foundation*: Nearly every application uses some platform software on the machine it runs on. This typically includes various support functions, such as standard libraries and storage, and a base operating system.
- A group of *infrastructure services*: In a modern distributed environment, applications frequently use basic services provided on other computers. It's common to provide remote storage, for example, integration services, an identity service, and more.
- A set of *application services*: As more and more applications become service-oriented, the functions they offer become accessible to new applications. Even though these applications exist primarily to provide services to end users, this also makes them part of the application platform. (It might seem odd to think of other applications as part of the platform, but in a service-oriented world, they certainly are.)

And while they're not shown in Figure 2, development tools are another important part of this story. Modern tools can help developers build applications using all three parts of an application platform.

To make this abstract model more concrete, think about how it fits with today's most popular on-premises platforms. The on-premises foundation looks like this:

- Operating system: The dominant choices are Windows, Linux, and other versions of Unix.

- Local support: Different technologies are used depending on the style of application. The .NET Framework and Java EE application servers provide general support for Web applications and more, for instance, while other technologies target specific kinds of applications. For example, Microsoft's Dynamics CRM product includes a platform designed for creating a particular type of business application. Similarly, different kinds of storage are used for different purposes. Raw byte storage is provided by the file systems in Windows, Linux, and other operating systems, while more structured storage is provided by a range of database technologies, including the Oracle DBMS, MySQL, Microsoft SQL Server, and IBM DB2.

For on-premises infrastructure services, typical examples include the following:

- Storage: Like storage in the foundation, infrastructure storage comes in various styles. A remote file system might provide simple byte-oriented storage, while a Microsoft SharePoint document library provides a more structured remote storage service. Applications can also access a database system remotely, allowing access to another kind of structured storage.

- Integration: Connecting applications within an organization usually depends on a remote service provided by some integration product. A message queue is a simple example of this, while more complex scenarios use products such as IBM's WebSphere Process Server, Microsoft's BizTalk Server, and others.

- Identity: Providing identity information is a fundamental requirement for most distributed applications. Common on-premises technologies that address this include Microsoft's Active Directory and other LDAP servers.

On-premises application services, the third category shown in Figure 2, vary widely across different organizations. The reason for this is simple: Different organizations use different applications, which in turn expose diverse services. One way to think about these applications in the on-premises platform is to divide them into two broad categories:

- Packaged applications: This includes business software such as SAP, Oracle Applications, and Microsoft Dynamics, along with a myriad of other off-the-shelf products. While not all packaged applications expose services to other applications, more and more of them do.

- Custom applications: Many organizations have a large investment in custom software. As these applications increasingly expose their functionality through services, they become part of the on-premises application platform.

When it's described like this, the on-premises application platform can seem quite complex. The truth, though, is that this platform has evolved over time. In the early days of computing, the application platform consisted of nothing more than an on-premises foundation. (Think of MVS and IMS on an IBM mainframe, for example.) In the 1980s and 1990s, as distributed computing spread, on-premises infrastructure services were added, with remote storage, integration, and identity becoming common. Today, with the advent of service-oriented applications, on-premises application services have become part of the platform. The next step in this evolution is clear: providing cloud versions of all three.

FROM ON-PREMISES PLATFORMS TO CLOUD PLATFORMS

Along with describing on-premises platforms, the general model just described can also be used to think about cloud platforms. And since on-premises and cloud platforms can be used together, it's important to understand how the two work in concert. Figure 3 illustrates this new world.

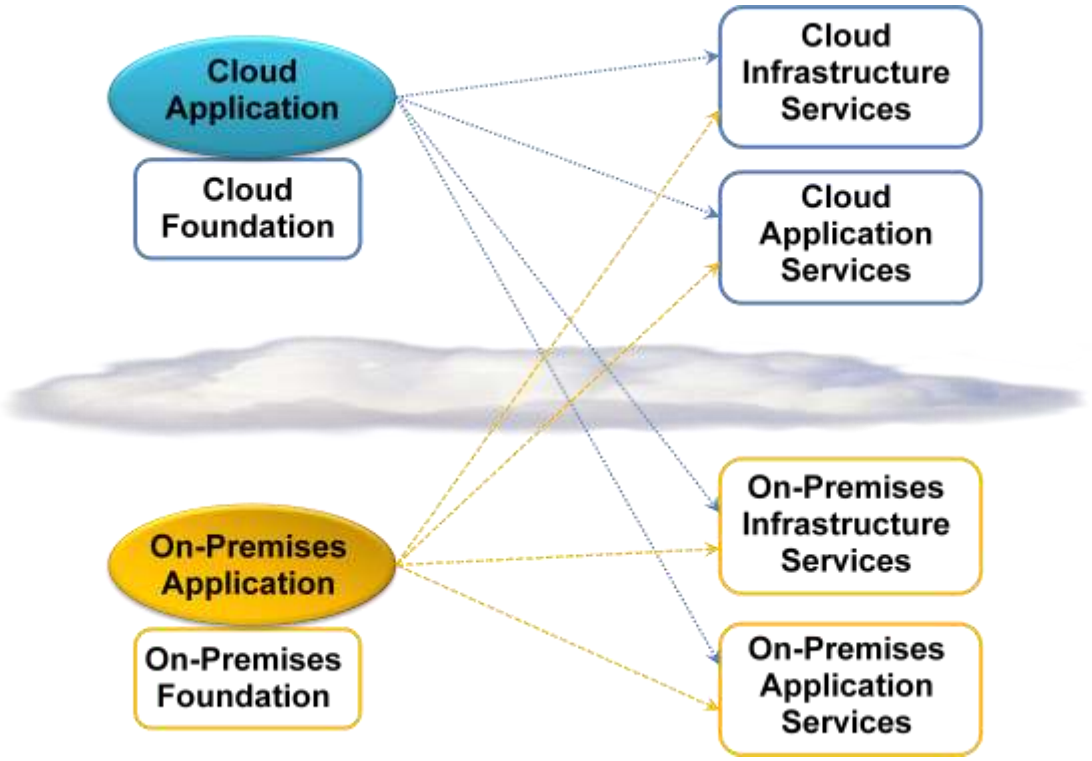

Figure 3: On-premises platforms and cloud platforms can be viewed in similar ways, and they can also be used together.

As the figure shows, a cloud application can be built on a cloud foundation, just as an on-premises application is built on an on-premises foundation. Both kinds of applications can access infrastructure and application services provided on-premises and in the cloud. Just as on-premises platforms support today's applications, cloud platforms provide services for the applications we're likely to build tomorrow.

EXAMINING CLOUD PLATFORMS

Understanding cloud platforms means looking at each of their parts: the cloud foundation, cloud infrastructure services, and cloud application services. This section walks through these three areas, using some of today's most visible cloud platform technologies as examples.

Before we begin, one important note: While it's useful to look at on-premises platforms and cloud platforms through the same lens, the two aren't identical. When platform functions move into the cloud, they sometimes change in significant ways. For example, on-premises platforms are designed to support (at most) enterprise-scale applications. Applications that run in the cloud, by contrast, can potentially operate at Internet scale, which requires handling many more simultaneous users than any enterprise application. While the same kinds of platform functions might be needed in both cases, achieving this high

scalability can force a cloud platform to provide them in a quite different way. In what follows, expect to see some differences from the on-premises world.

CLOUD FOUNDATION

Like their on-premises cousins, cloud foundations provide the basic local functions an application needs. These can include an underlying operating system and local support. Yet how cloud platforms provide these functions differs from what we're used to, as this section shows.

Operating System

From a platform point of view, an operating system provides a set of basic interfaces for applications to use. By far the most well-known example of an operating system in the cloud today is Amazon's Elastic Compute Cloud (EC2). EC2 provides customer-specific Linux instances running in virtual machines (VMs). From a technical perspective, it might be more accurate to think of EC2 as a platform for VMs rather than operating systems. Still, a developer sees an operating system interface, and so viewing it in this light makes more sense here.

Each development team is free to use whatever local support it likes in this VM—Amazon doesn't care. The creators of one application might choose a Java EE app server and MySQL, for example, while another group might go with Ruby on Rails. EC2 customers are even free to create many Linux instances, then distribute large workloads across them in parallel, such as for scientific applications. While the service EC2 provides is quite basic, it's also very general, and so it can be used in many different ways.

Local Support

In an on-premises platform (and in EC2), a developer can mix and match parts of the foundation as she sees fit. Choosing to use the .NET Framework on Windows doesn't mandate using a particular database, for example. Similarly, an on-premises application using the .NET Framework is free to access the underlying Windows operating system, as is an application built on a Java EE server.

The local support functions in today's leading cloud foundations don't work this way. Instead, a cloud local support technology typically includes its own storage, and it hides whatever the underlying operating system might be. A developer choosing to build on a particular local support option must accept the limitations it imposes.

There are good reasons for these limitations, of course. One of the things that makes cloud computing so attractive is its potential for scalability, but to make an application built on a cloud foundation handle Internet-size loads requires limiting it in some ways. By making the local support functions more specialized, a cloud platform provider has more freedom to optimize the application environment. Accordingly, each set of local support functions in cloud foundations today focuses on supporting a particular kind of application.

For example, Google's AppEngine provides local support for running Python Web applications. Along with a standard Python runtime, AppEngine also includes a hierarchical data store with its own query language. Another example of a cloud platform providing local support is Force.com, offered by Salesforce.com. Rather than targeting general Web applications, however, Force.com is aimed at creating data-oriented

business applications. Toward this end, it provides its own focused support for data storage. And rather than adopt an existing programming language, this platform's creators invented their own, a language called Apex.

Microsoft also provides local support for applications in the cloud as part of its CRM Live offering. Based on the Dynamics CRM platform mentioned earlier, this technology targets data-oriented business applications, much like Force.com. And like both Force.com and AppEngine, it includes both run-time application support and a data store. Microsoft has also talked about its plans to go further in this area, with a platform that will support standard .NET development languages and tools. The intent, Microsoft says, is to allow portability of both applications and developer skills between the company's on-premises foundation and its cloud foundation.

CLOUD INFRASTRUCTURE SERVICES

Whether they run on-premises or in the cloud, some applications don't need anything beyond a foundation. Still, many can benefit from distributed storage, common identity, and other infrastructure services. We're accustomed to having these services provided on-premises today, but analogous services are also provided in the cloud.

As Figure 3 showed, cloud infrastructure services can be accessed by applications running on either an on-premises foundation or a cloud foundation. Initially, the most common users of cloud infrastructure services will be on-premises, because there aren't yet many applications built on a cloud foundation. Over time, expect this to change, as more and more cloud-based applications also use cloud infrastructure services.

Storage

Applications commonly use some kind of local storage, which is why storage is part of both on-premises and cloud foundations. Remote storage is also useful, however, as the popularity of this service in the on-premises world shows. Accordingly, it's reasonable to expect that providing a storage service in the cloud will be attractive for many applications.

As with on-premises platforms, remote storage in the cloud comes in different styles. For example, Amazon's Simple Storage Service (S3) provides basic unstructured remote storage. The model it exposes to developers is straightforward: *objects*, which are just bunches of bytes, are stored in *buckets*. Applications can create, read, and delete objects and buckets. Objects can't be updated, however—they can only be entirely replaced. This is another example of how platform services must change to support Internet-scale usage, something that Amazon is clearly focused on. This simple but limited storage service is much easier to make scalable than a more fully featured offering would be. The trade-off is clear: Application developers get cheap storage in the cloud, but they might need to do more work in their applications to use it effectively.

Another approach to cloud storage is to support more structured data. In Microsoft's SQL Server Data Services (SSDS), for example, a *container* includes one or more *entities*, each of which holds some number of *properties*, as shown in Figure 4. An application can issue queries against a container's data with operators such as ==, !=, <, >, AND, OR, and NOT.

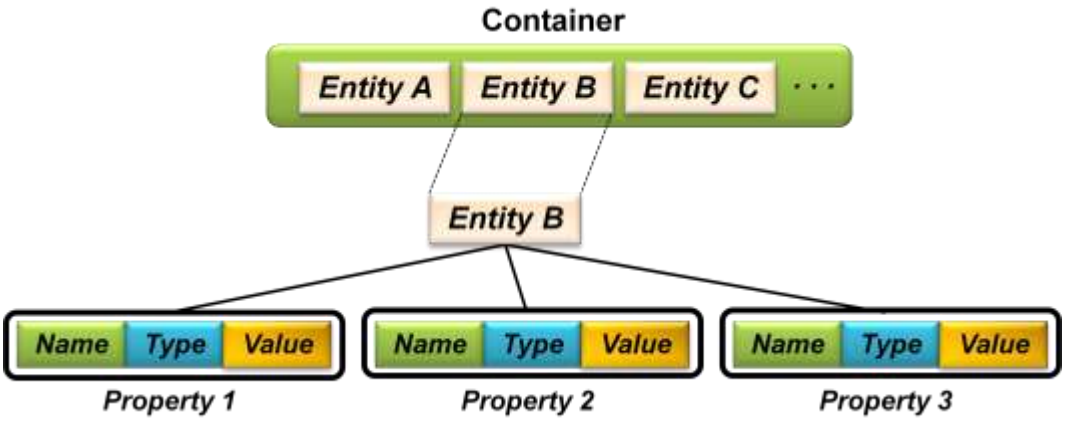

Figure 4: In SQL Server Data Services, a container holds entities with properties.

It's important to note that this isn't a relational database, and the query language isn't SQL. Once again, we're seeing an illustration of how application platform technologies change when they're moved into the cloud. This simpler approach is easier to use than a relational database—there's no need to define a schema up front—and it's also easier to make scalable.

Amazon's SimpleDB provides one more example of the value of structured storage in the cloud. The way SimpleDB organizes information is similar to SSDS—it's a hierarchy of *domains*, *items*, and *values*—and it also provides a non-SQL query language. Like SSDS, no up-front schema definition is required, and so the approach provides a mix of flexibility and scalability.

Integration

Is there any application left that doesn't talk to at least one of its fellows? Connecting applications has become a staple of computing, and vendors have provided a plethora of on-premises infrastructure services to do it. These range from relatively simple technologies like message queues to quite complex integration servers.

As integration services move into the cloud, a range of technologies is also appearing. For example, Amazon's Simple Queue Service (SQS) provides just what its name suggests: a straightforward way for applications to exchange messages via queues in the cloud. Yet SQS once again illustrates what happens when a familiar on-premises service is recast as a cloud service. Because SQS replicates messages across multiple queues, an application reading from a queue isn't guaranteed to see all messages from all queues on a particular read request. SQS also doesn't promise in-order, exactly-once delivery. These simplifications let Amazon make SQS more scalable, but they also mean that developers must use SQS differently from an on-premises message queuing technology.

BizTalk Services provides another example of cloud-based integration. Rather than using message queuing, BizTalk Services implements a relay service in the cloud that lets applications communicate through firewalls. Cloud-based integration, such as connecting applications in different organizations, typically requires traversing firewalls, and so solving this problem is important. BizTalk Services also provides simple workflow support along with a way for an application to register the services it exposes, then let those services be invoked by any other application that has permission to do so.

Going forward, expect to see more integration services offered in the cloud. Given the importance of integration as an on-premises service, it shouldn't be surprising to see its functions become part of the cloud infrastructure.

Identity

Whether an application runs on-premises or in the cloud, it typically needs to know something about its users. Toward this end, the application commonly demands that each user provides a digital identity, a set of bytes that describes that user. Based on what these bytes contain and how they're verified, the application can determine things such as who this user is and what they're allowed to do.

Many on-premises applications today rely on an on-premises infrastructure service, such as Active Directory, to provide this identity information. When a user accesses a cloud application, however, or an on-premises application accesses a cloud service, an on-premises identity usually won't work. And what about an application built on a cloud foundation? Where does it get its identity information?

An identity service in the cloud can address these issues. Because it provides a digital identity that can be used by people, by on-premises applications, and by cloud applications, a cloud identity service can be applied in many different scenarios. In fact, one indication of the importance of this kind of identity service is the number of cloud identity services available today. Accessing Amazon cloud services such as EC2 or S3 requires presenting an Amazon-defined identity, for instance, while using Google AppEngine requires a Google account. Microsoft provides Windows Live ID, which can be used for Microsoft applications and others, while BizTalk Services also offers its own identity service, which can be federated with others. Developers don't have complete freedom—cloud platforms are frequently tied to a particular identity provider—but the need for identity as a cloud service is clear.

CLOUD APPLICATION SERVICES

What's the difference between an application service and an infrastructure service? To answer this question, think first about the obvious distinction between applications and infrastructure: Applications are designed to be used by people, while infrastructure is designed to be used by applications. It's also fair to say that infrastructure usually provides a general, relatively low-level service, while applications provide more specific, higher-level services. An infrastructure service solves a broad problem faced by many different kinds of applications, while an application service solves a more targeted problem. And just as it's possible to identify different kinds of infrastructure services, it's also possible to distinguish different categories of application services, as this section illustrates.

SaaS Application Services

Users in most enterprises today rely on both purchased and home-grown applications. As these applications expose their services to remote software, they become part of the on-premises platform. Similarly, SaaS applications today frequently expose services that can be accessed by on-premises applications or by other cloud applications. Salesforce.com's CRM application, for example, makes available a variety of services that can be used to integrate its functions with on-premises applications. As organizations begin to create their own SaaS applications running on a cloud foundation, those applications will also expose services. Just as packaged and custom on-premises applications today are

part of the on-premises platform, the services exposed by packaged and custom SaaS applications are becoming part of the cloud platform.

Search

Services exposed by SaaS applications are useful, but they're not the whole story. Other kinds of cloud application services are also important. Think, for example, of search engines such as Google and Live Search. Along with their obvious value to people, why can't they also offer cloud application services?

The answer, of course, is that they can. Microsoft's Live Search, for example, exposes services that allow on-premises and cloud applications to submit searches and get results back. Suppose a company that provided a database of legal information wanted to let customers search both its own data and the Web in a single request. They could accomplish this by creating an on-premises application that both searched their proprietary data and, via the Live Search application service, the entire Web. It's fair to say that not many applications are likely to need this kind of service, but that's one reason why it's most accurate to think of search as an application service rather than an infrastructure service.

Mapping

Many Web applications today display maps. Hotel Web sites plot their locations, retailers provide store locators, and more. The people who create these applications probably don't have the time, interest, or budget to create their own mapping database. Yet enough applications need this function to justify creating a cloud application service that provides it.

This is exactly what's done by mapping services such as Google Maps and Microsoft's Virtual Earth. Both provide cloud-based services that application developers can use to embed maps in Web pages and more. And as with search, these mapping services are adjuncts to existing Web sites that target users directly, i.e., they're cloud application services.

Other Application Services

Many other application services are available today. In fact, almost any Web site can expose its functionality as a cloud service for developers to use. Photo-sharing sites such as Google's Picasa and Microsoft's Windows Live Photo Gallery do this, for example, as do online contacts applications such as Google Contacts and Microsoft's Windows Live Contacts. One big motivation for exposing services is to make it easier to create mash-ups that exploit the functions of diverse Web applications.

Vendors sometimes group cloud application services together under a common umbrella. The services for accessing information in Google Contacts, Picasa, and more are all part of the Google Data APIs, for instance. Similarly, Microsoft groups several of its services together under the Live Platform brand, including Live Search, Virtual Earth, Windows Live Contacts, Windows Live ID, an Alerts service, a specialized storage service called Application-Based Storage, and several more.

The line between cloud infrastructure services and cloud application services can sometimes be fuzzy. General cloud storage services such as S3 and SSDS are clearly infrastructure, for example, as are cloud identity services. A mapping service such as Google Earth is just as clearly application-centric—only certain kinds of apps need it—as is a service like Live Search. But an Alerts service might be considered

infrastructure, since it's more generally useful, and Windows Live ID is definitely infrastructure, even though Microsoft views both services as part of its Live Platform.

Cloud platforms are a relatively new area, and so it shouldn't be surprising that defining a firm taxonomy is challenging. However you choose to view them, it's clear that cloud application services have an important role to play. Knowing what's available in the cloud should be a core competency today for everyone who designs and builds software.

CONCLUSION

A new kind of application platform doesn't come along very often. But when a successful platform innovation does appear, it has an enormous impact. Think of the way personal computers and servers shook up the world of mainframes and minicomputers, for example, or how the rise of platforms for N-tier applications changed the way people write software. While the old world doesn't go away, a new approach can quickly become the center of attention for new applications.

Cloud platforms don't yet offer the full spectrum of an on-premises environment. For example, business intelligence as part of the platform isn't common, nor is support for business process management technologies such as full-featured workflow and rules engines. This is all but certain to change, however, as this technology wave continues to roll forward.

Cloud platforms aren't yet at the center of most people's attention. The odds are good, though, that this won't be true five years from now. The attractions of cloud-based computing, including scalability and lower costs, are very real. If you work in application development, whether for a software vendor or an end user, expect the cloud to play an increasing role in your future. The next generation of application platforms is here.

ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps software professionals around the world understand, use, and make better decisions about new technology.