InfoWorld
SPECIAL REPORT
SEPTEMBER 2009

Cloud Computing Deep Dive

Deep Dive Articles

Multimedia at InfoWorld.com

- InfoClipz: Cloud Computing Explained in 3 Minutes

"Cloud versus Cloud" QuickTime tours:

- Amazon EC2
- AppEngine
- GoGrid
- AppNexus

EXCLUSIVE BOOK EXCERPT

Selecting the Right Cloud

A step-by-step architectural approach for savvy enterprise adoption

By David Linthicum

THERE ARE MANY PATTERNS, OR CATEGORIES, IN THE WORLD of cloud computing that you can use to meet the needs of your enterprise architecture. Some solve specific problems, such as security-as-a-service or testing-as-a-service, and some provide complete platforms, such as platform-as-a-service or infrastructure-as-a-service. They all have trade-offs and different problems that each solves. However, you must consider them all in light of your architecture.

So, the categories of service are storage, database, information, process, application, platform, integration, security, management/governance, testing, and infrastructure. The figure shows how they relate.

You can further break them down into fine-grained solutions, or those providers who solve very specific problems that alone cannot be considered a platform. Or into coarse-grained providers, or those that, unto themselves, are a complete platform.

Thus, fine-grained services include storage, database, information, process, integration, security, management/governance, and testing. And coarse-grained services include application, platform, and infrastructure.

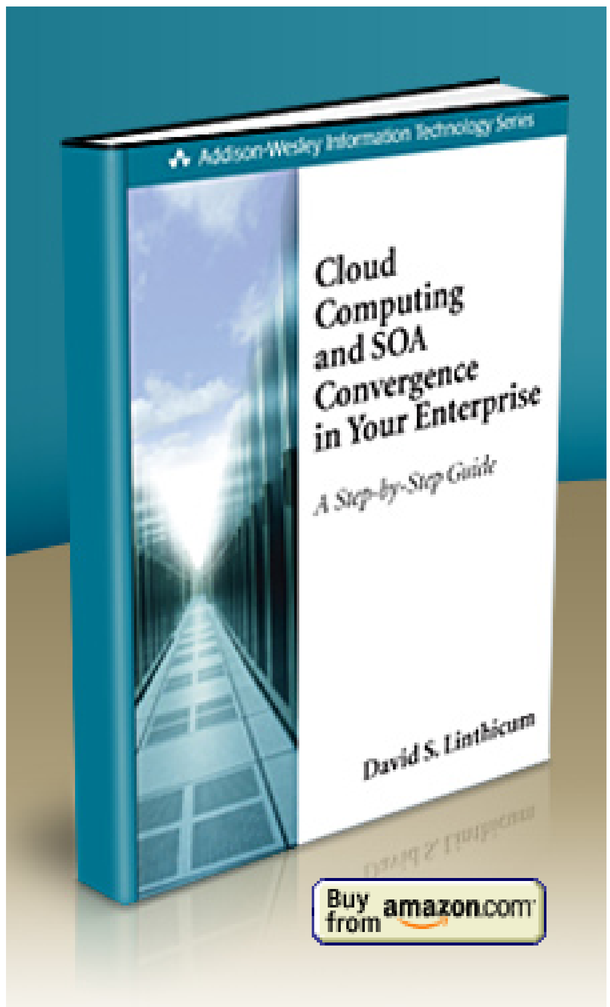
It's helpful to do this breakdown, because one coarse-grained cloud computing provider can actually be made up of many fine-grained resources. For example, a single platform-as-a-service provider could offer storage, database, process, security, and testing services.

However, while it may seem easier to use a coarse-grained cloud computing solution because it provides many fine-grained resources, the decision is really a matter of the requirements of your architecture. You may find that selecting many fine-grained cloud computing solutions is a much better fit for your architecture, when considering your requirements and/or the ability to mesh effectively with the on-premise portion of your architecture. Thus, it is useful to think of the candidate cloud computing provider categories by architectural component:

- For processes, the service components are application, platform, infrastructure, process, and integration.
- For data, the service components are application, platform, infrastructure, storage, database, and information.
- For services, the service components are application, platform, infrastructure, and information.

To make this point clearer, consider this provider arrangement based on one possible architectural categorization:

- Processes: process service via Appian Anywhere.
- Data: infrastructure service via Amazon.com's Elastic Computing Cloud (EC2) and database service via Amazon Simple DB.
- Services: infrastructure service via Amazon EC2.



You might store your data in Amazon Simple DB, as well as on the Amazon EC2 platform. Then, you might build and/or host the services on the Amazon EC2 platform, say using an application server it provides on-demand within that platform. Finally, you could use Appian Anywhere as the platform where those processes live. Keep in mind that the processes are connected to the services, and the services are connected to the data. You're just selecting the target platforms here.

As a more complex example involving more cloud computing providers:

- Processes: process service via Appian Anywhere and application service via Salesforce.com.
- Data: infrastructure service via 3Tera Cloudware and Amazon EC2, and database service via Amazon Simple DB.
- Services: infrastructure service via Amazon EC2 and 3Tera Cloudware, application service via Salesforce.com, and platform service via Salesforce.com's Force.com.

Or, you could have a simple arrangement using a single cloud computing provider:

- Processes: process service via Amazon EC2.
- Data: infrastructure service via Amazon EC2.
- Services: infrastructure service via Amazon EC2.

Moreover, you need to consider the other core components of the architecture including security, testing, and governance, which can be deployed on-premise or in the cloud, depending on your needs.

The purpose of this exercise is to is to illustrate the number of architectural options you have, and how you can mix the options to form your final architecture, using as many or as few as you need to address the requirements of the architecture and thus the business.

THE PROCESS OF MOVING TO THE CLOUDS

Figure 2 depicts the high-level process you can use to find the right cloud computing category or categories, and finally cloud computing providers to move your processes, services, and data that were selected as good cloud computing candidates.

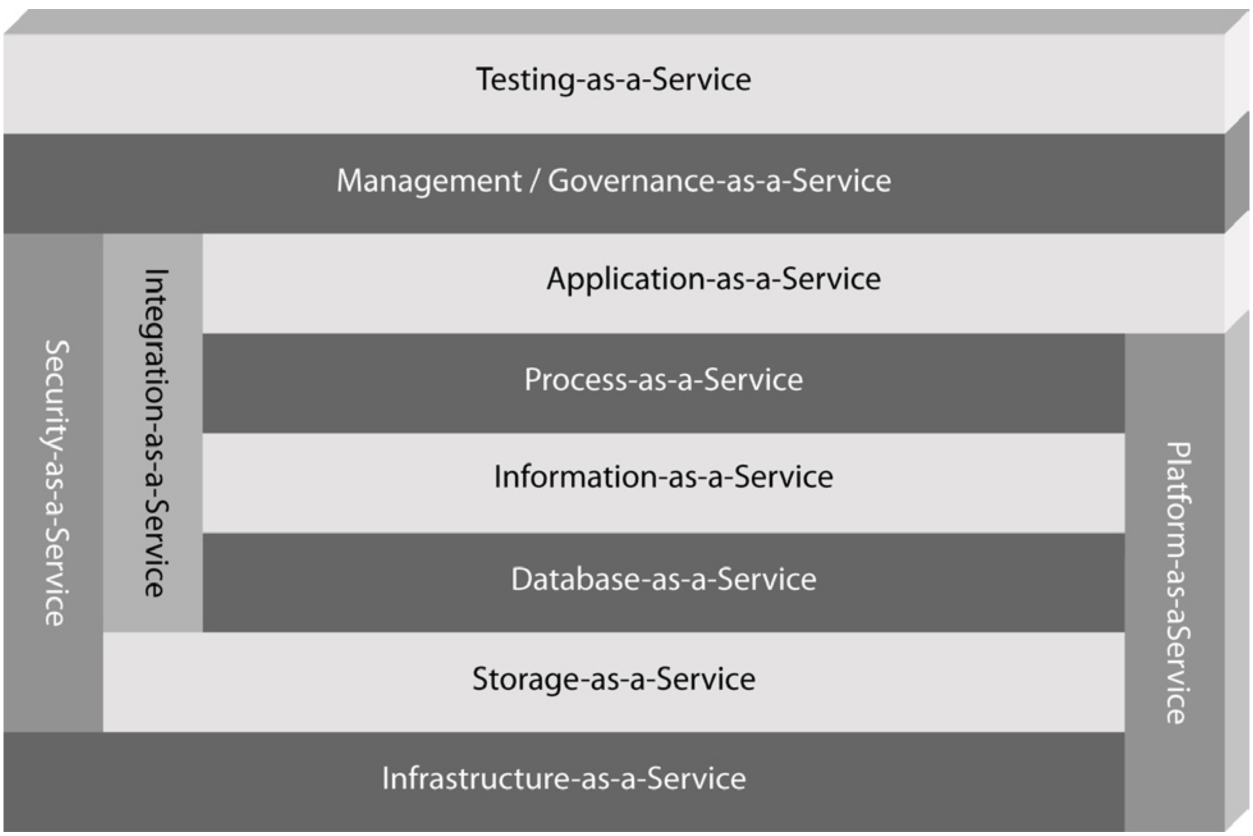


Figure 1: The patterns or categories of cloud computing providers allow you to use a discrete set of services within your architecture.

The core steps are:

1. List the candidate platforms.
2. Analyze and test the candidate platforms.
3. Select the target platforms.
4. Deploy to the target platforms.

Let's look at each step in more detail.

STEP 1: LIST THE CANDIDATE PLATFORMS

This is pretty simple, considering the information already presented above. You need to list any and all cloud computing platforms that may be a fit for your "to be" architecture. This requires that you understand what's out there, as well as the categories they exist in, and then what they do.

The fact of the matter is that there are no hard-and-fast rules around what defines a cloud computing solution. Thus, you'll find that many software providers — no matter if they have a true cloud computing solution or not — have a tendency to say they do have one. For example, some software vendors claim that, because they can be downloaded over the Web to an on-premise computing system,

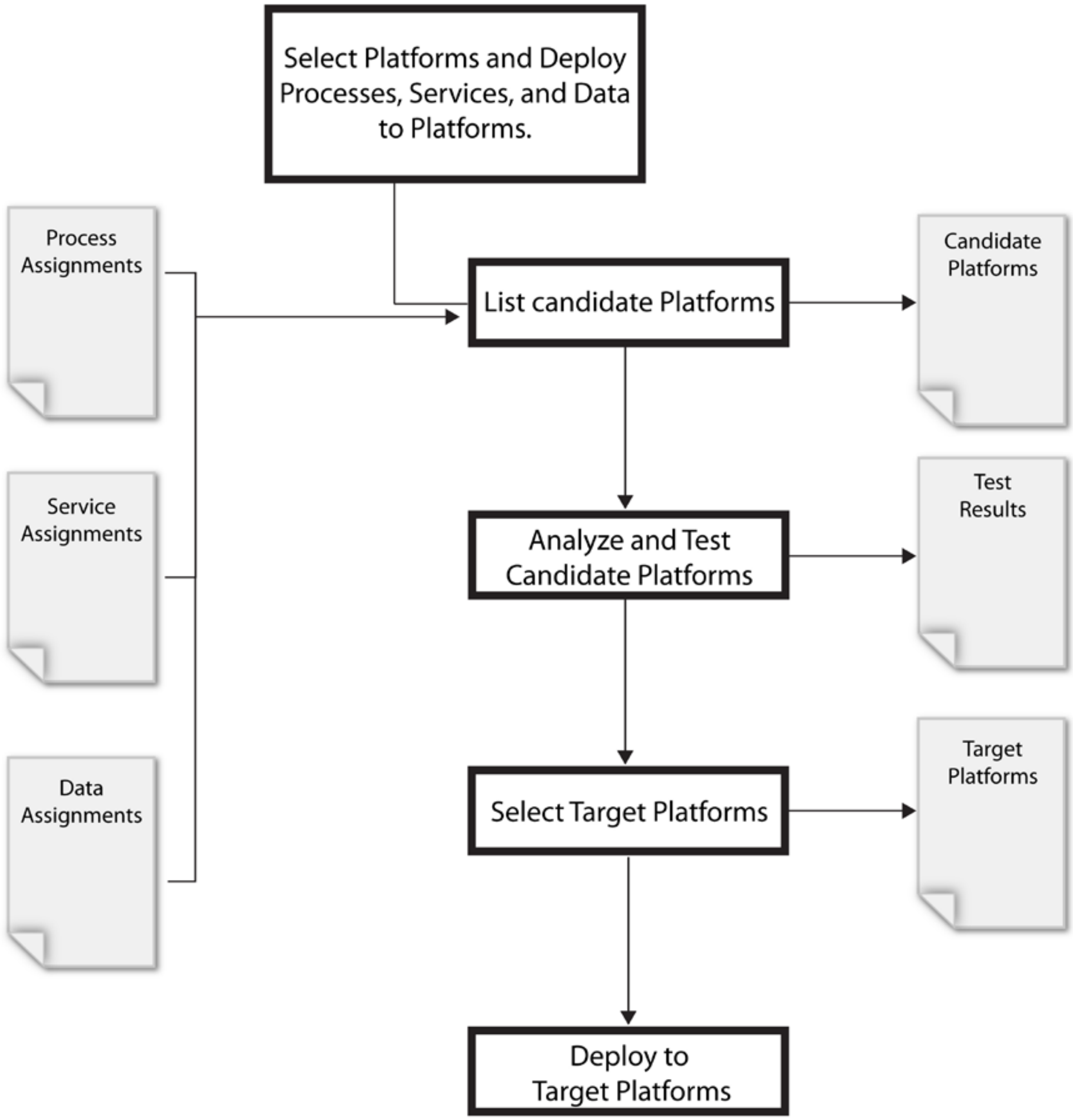


Figure 2: This step in the process is all about taking the process, service, and data requirements and mapping them to the right technology.

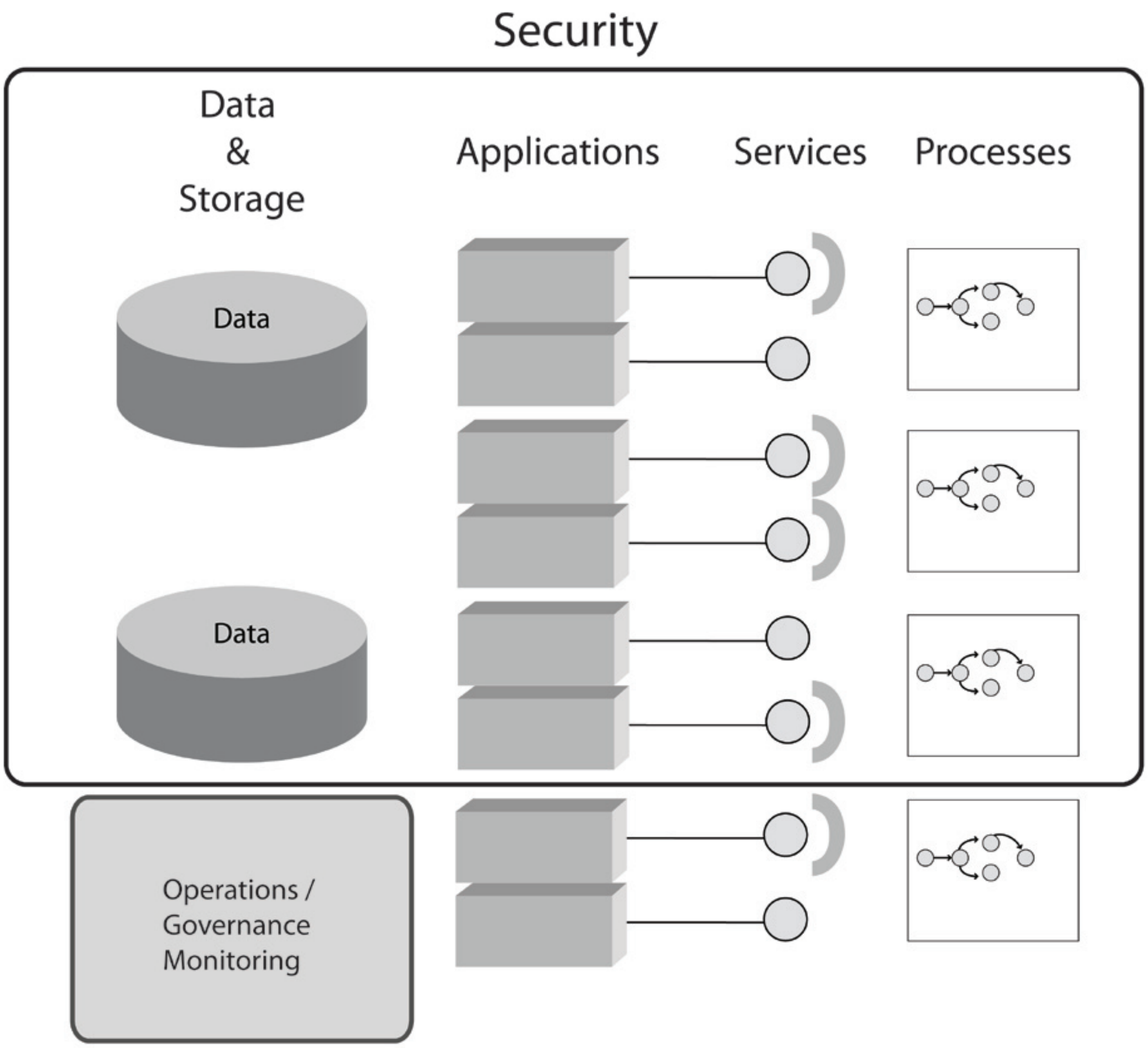


Figure 3: The core architecture requires you to find places for storage, data, operations, governance, security, services, and processes.

they are an on-demand or cloud computing platform. They are not. Therefore, this step is about separating the wheat from the chaff, not just tossing together a list.

Mastery of cloud computing is as much about keeping up with the market space as it is understanding what each vendor provides.

Thus, you need to answer two key questions:

1. What categories do you need?

2. Which cloud computing providers in these categories should appear on the list?

The categories that you'll use depend on the final logical architecture and the requirements you've identified through this process. However, there are some generalizations I can talk about here, including the fundamental layers that you'll require, and what to look for within each layer (see Figure 3). They are storage, database, processes, services, security, governance, and management.

Storage: The cloud service that will allow you to store, share, and manage file systems in support of parts or of all of the architecture. You typically use storage-as-a-service for this, either from a cloud computing provider that only provides storage-as-a-service or as part of an infrastructure-as-a-service provider.

What you need to look out for here are capacity and performance. Capacity is your ability to scale your storage needs to support your architecture; performance is your ability to move files to and from the cloud computing service at a speed that supports the business. Performance problems are the most likely issue here, so make sure you do your testing.

Database: This is the storage and retrieval of data, either by using a platform, database, or infrastructure service.

THE 11 CLOUD COMPUTING CATEGORIES

As cloud computing emerges, there is a lot of discussion about how to define cloud computing as a computing model. Maturity models have been published and debated, and providers clearly have a model for their own products.

In attempting to better define cloud computing, I have come up with a stack of sorts, which I believe makes logical sense. The stack considers each component of cloud computing and how they interact. While clearly this could be much more complex, I don't think it needs to be.

While you can debate the individual components, there are 11 major categories or patterns of cloud computing technology:

Storage-as-a-service (a.k.a. disk space on demand): As you may expect, this is the ability to use storage that physically exists at a remote site but is logically a local storage resource to any application that requires storage. This is the most primitive component of cloud computing, and it is a component or pattern that's used by most of the other cloud computing components.

Database-as-a-service: This provides the ability to use the services of a remotely hosted database, sharing it with other users, and having it logically function as if the database were local. Different providers have different models, but the power is to use database technology that would typically cost thousands of dollars in hardware and software licenses.

Information-as-a-service: This refers to the ability to consume any type of information, remotely hosted, through a well-defined interface such as an API. Examples include stock price information, address validation, and credit reporting.

Process-as-a-service: This refers to a remote resource that can bind many resources together, such as services and data, whether hosted within the same cloud computing resource or remotely available, to create business processes. You can think of a business process as a meta-application that spans systems, leveraging key services and information that are combined into a sequence to form a process. These processes are typically easier to change than applications, and thus provide agility to those who use these process engines that are delivered on-demand.

Application-as-a-service (a.k.a. software-as-a-service): This is any application that is delivered over the platform of the Web to a user, typically accessing the application through a browser. While many people associate application-as-a-service with enterprise applications such as Salesforce SFA, office automation applications are indeed applications-as-a-service as well, such as Google Docs, Gmail, and Google Calendar.

Platform-as-a-service: This is a complete platform — including application development, interface development, database development, storage, and testing — delivered through a remotely hosted platform to subscribers. Based on the traditional time-sharing model, modern platform-as-service providers provide the ability to create enterprise-class applications for use locally or on-demand for a small subscription price or for free.

Integration-as-a-service: This is the ability to deliver a complete integration stack from the cloud, including interfacing with applications, semantic mediation, flow control, and integration design. In essence, integration-as-a-service includes most of the features and functions found in traditional EAI technology, but they are delivered as a service.

Security-as-a-service: As you may have guessed, this is the ability to deliver core security services remotely over the Internet. While the typical security services provided are rudimentary, more sophisticated services are becoming available such as identity management.

Management-/governance-as-a-service: This is any on-demand service that provides the ability to manage one or more cloud services. These are typically simple things such topology, resource utilization, virtualization, and uptime management. Governance systems are becoming available as well, such the ability to enforce defined policies on data and services.

Testing-as-a-service: This is the ability to test local or cloud-delivered systems using testing software and services that are remotely hosted. It should be noted that while a cloud service requires testing unto itself, testing-as-a-service systems can test other cloud applications, Web sites, and internal enterprise systems, and they do not require a hardware or software footprint within the enterprise.

Infrastructure-as-a-service: This is actually datacenter-as-a-service, or the ability to remotely access computing resources. In essence, you lease a physical server that's yours to do with as you will, and for all practical purposes it is your datacenter, or at least part of a datacenter. The difference with this approach versus more mainstream cloud computing is that instead of using an interface and a metered service, you're getting access to the entire machine and the software on that machine. In short, it's less packaged and more akin to hosting.

What you need to consider here include the ability for the cloud-delivered database to support the features and functions you require for your architecture — including the use of stored procedures and triggers, the function of the API, adherence to standards, and performance.

Within the case of infrastructure-as-a-service, the cloud computing providers typically allow you to use name-brand databases, such as Oracle or MySQL. However, the database-as-a-service providers typically use a database that's homegrown and thus tends to be proprietary.

Performance comes into play here as well. Most on-premise and traditional applications are data I/O-bound, so you'll find that similar performance problems may exist here. Consider the overhead of I/O on a multitenant platform, as well as the latency that can occur when you send large amounts of data between your enterprise and the cloud computing provider over the Internet. This fact could also lead you to make a case for placing the database closer to the processes and services that use that database — a core tenet of architecture when you consider performance and the reliability of databases.

Processes: These can exist on process, platform, application, and infrastructure providers, for the most part. There are a few issues you need to consider here.

When using process providers, processes are all they do. Thus, you'll need to bind the other architectural components (typically services and data) to those processes. The data and service assets exist either within systems that are on-premise or with other cloud computing providers, so you'll have to make sure that integration occurs — and that it is reliable.

Application-as-a-service providers typically don't provide a platform for you to create your own processes, but allow you to use pre-built processes on their platform. This is handy because, for example, you won't have to create a custom fulfillment process for your business as you can just use the provider's. However, as with the process-as-a-service, the processes are isolated and thus must be linked back with other on-premise and cloud computing-delivered systems that are part of your architecture.

When considering infrastructure-as-a-service providers and platform-as-a-service providers, you are typically dealing with platforms that provide the "complete stack," including storage, database, processes, applications, services, development, and testing. These processes are just a component of those platforms.

It may seem tempting to use "complete stack" providers because they do indeed provide one-stop-shopping for cloud computing. However, you'll have to make tradeoffs such as loving the application development features of one platform-as-a-service provider, but hating the way its product manages processes or the process engine it provides. In many cases, it may be better to use other cloud computing providers or even on-premise software to address processes, trading simplicity for complexity, but leveraging a process engine that is the right fit for the architecture.

Services: Generally speaking, services (such as Web services) can live on most cloud computing platforms. However, only a few cloud computing providers (including platform, process, and infrastructure service) offer the capabilities to create and host services, By contrast, application and information service providers offer access to pre-built services they host, but you cannot change these services.

The most common issue here is performance, because services such as Web services (no matter if you use REST or SOAP) have a tendency to cause performance problems if the platform hosting the service can't provide enough computing resources to the service, or if there are too many services that saturate the platform and the network. Again, you need to test for performance by actually using the services, and then adjust your platform, the number of services you use, and how those services are designed to optimize the performance of your architecture.

Security: This is not a platform nor a piece of software that exists on-premise or on cloud computing platforms. If done right, it should be a systemic attribute of the architecture, no matter how much of it is on-premise or delivered via cloud computing. Thus, you should address security by creating a strategy and a model to secure your architecture, based on the requirements you identified. Then you select the proper approach and the enabling technology. This effort is typically around identity management and the standards that support identity management.

With the increasing interest in identity management, in support of more complex and distributed architectures such as SOA (service-oriented architecture) and SOA using cloud computing, there's been a rise in the need for standards to better define this space. These standards all aim to bind together identity management systems in all organizations into a unified whole, allowing for everyone to be known to everyone else, securely.

So, why do we need identity management? It's a fact that services are not for internal use anymore, as is the case when using cloud computing. Those who use services (consumers) or produce services (providers) need to be known to each other, else you risk invoking malicious or incorrect behavior, which could cost you dearly. This is clearly the case for cloud computing.

Governance: This brings its own set of issues when considering architecture and cloud computing. While there are governance systems that are cloud-delivered (and they work well for some types of architecture), governance systems that implement, manage, and enforce policies are both runtime in nature and are typically on-premise.

Issues to look out for here again include performance, because in some instances executing policies could cause latency issues. Also important is the governance mechanism's ability to govern resources, which are typically services that are cloud-delivered. This means having the ability to track remote services within the governance technology's repository, as well as monitor those services during runtime.

Management: A widely distributed and complex architecture, such as SOA using cloud computing, requires a management technology that can see both systems that are on-premise, which most do, and those that are cloud computing-based, which only a few do well. Moreover, you should check if the cloud provider has an interface that lets management technology talk to it.

The core idea here is to provide a management platform that sees all on-premise and cloud computing-based systems at the "working or not working" level, so you can at least see if a system is down and how that status affects other systems in the architecture. However, it's preferable to have a management system that can see systems at more granular levels — such as services, processes, data, storage — so it's much easier to diagnose the issues and spot troubles before they happen.

Management and governance are clearly linked, and have very similar patterns.

STEP 2: ANALYZE AND TEST THE CANDIDATE PLATFORMS

Once you've selected the candidate cloud computing platforms, you need to make sure that they live up to the requirements you established. You do this through some deep dives into each candidate platform you selected and then through testing.

Cloud testing is a bit different than on-premise testing, in that you're actually testing the generic capabilities of the cloud computing platform. Specifically, you'll look at how that cloud computing platform supports the requirements of the architectural components, including services, data, and processes, before actually deploying the components on those platforms. It's a validation exercise before you do the deployment.

The use of performance modeling and performance testing is helpful here. Modeling creates a simulation of how the system should perform under different types of loads; typically light, medium, and heavy. Performance testing means you actually do the testing to determine how the architecture performs under stress.

This means modeling the architecture, including how the information flows and the services that are invoked, and how that affects the different computing resources, both on-premise and cloud-based. While not perfect, you should have a general idea as to what performance you can expect from the cloud computing platforms, and how things such as decreasing processing power or expanding bandwidth should affect overall performance.

While proving the performance models, you should use performance testing —how well and how fast the complete architecture, both on-premise and cloud-based, supports the business. Moreover, you should measure how the system performs during an ever-increasing storage, database, process, and service processing load. This also identifies the potential bottlenecks, such as network, database, and

services that you can choose to accept, try to work around, or forgo (which means you must look elsewhere).

STEP 3: SELECT THE TARGET PLATFORMS

Once you go through all of the analysis — including a service-, process-, and data-level understanding of your problem domain — and have considered both security and governance, compiled a list of candidate systems, and complete the validation testing, it's time to pick the cloud computing platforms.

You'll find that this step is pretty easy considering that any issue around the platform's ability to meet the requirements of the architecture, and thus the business, should be understood by now.

It's likely that the final selection of the suite of target cloud computing platforms is very different than what you first envisioned, but if you've done your homework and followed each step, they should be the proper platforms for your architecture.

Also worth mentioning is the ease of switching from cloud computing platform to cloud computing platform, if for some reason you make the wrong call or, more likely, some business event occurs with the cloud computing platform, such as the cloud computing provider going out of business or a merger or acquisition that changes or removes that platform. Of course, the ability to switch cloud computing providers depends on their use of standards and your ability to find another provider that offers similar characteristics and features.

The business issues are more important if you're looking to create a SOA using cloud computing, because you're completely dependent on the cloud provider to stay in business. Thus, you need to carefully consider:

- the viability of the provider, and the likelihood that it will continue to support your cloud computing platforms
- its ability to recover from hardware, software, and network failures — dynamically and with minimum downtime
- the service-level agreements (SLAs) and a meeting of the minds between you and the cloud provider as to what service levels need to be supported for your architecture
- a complete understanding of the policies of the cloud computing provider, and what denotes a violation (in some instances, cloud computing providers have just cancelled accounts due to policy violations, without notice)

STEP 4: DEPLOY TO THE TARGET PLATFORMS

This the "just do it" step, meaning that you actually port code, migrate data, and create new services, processes, and databases, as well as test and validate that all services, databases, and processes are working correctly.

You should use an approach centered around migration and development over time, rather than a "big bang" approach. Thus, you should select which components of the architecture should move, or be created on the cloud computing platforms, in the order of most important to least important.

As you move these architectural components to the cloud computing platforms, make sure they are functioning correctly and have been properly tested before moving on to the next architectural component. While the pressure may be on to make the "big switch," the reality is that this evolutionary approach avoids problems and does not overwhelm those who deploy services, data, and processes to the cloud computing platforms. Also, this approach provides the value of learning as you go, so your knowledge of how to make cloud computing platforms work for your architecture will increase significantly as you move through this process.

EMBRACE THE NEW "CLOUDY" PLATFORMS

The activities outlined in this article are some of the most fun you'll have around cloud computing: actually moving systems to the clouds and making those systems work for the business. It's doing rather than planning or analyzing, but it's also the trickiest of all the activities and thus carries the most risk.

In addition, you know that the cloud computing platforms are moving targets, so as the hype and the market heats up, new providers appear weekly and existing providers try to pack in as much functionality as they can to capture the market.

Cloud computing platforms are very easily changed, because they don't require the distribution of software to enterprises, which results in ongoing activity: constant upgrades, bug fixes, and other changes to the platform. Hopefully, these changes move the overall system in better directions and not break your architectural components that exist on these platforms (that is, fail to ensure backward compatibility).

David Linthicum is InfoWorld's Cloud Computing blogger.

CONTEXT

What cloud computing really is

The next big IT trend sounds nebulous. Let us clear the air.

By Eric Knorr and Galen Gruman

CLOUD COMPUTING IS ALL THE RAGE. "IT'S BECOME THE PHRASE du jour," says Gartner senior analyst Ben Pring, echoing many of his peers. The problem is that (as with Web 2.0) everyone seems to have a different definition.

As a metaphor for the Internet, "the cloud" is a familiar cliché, but when combined with "computing," the meaning gets bigger and fuzzier. Some analysts and vendors define cloud computing narrowly as an updated version of utility computing: basically virtual servers available over the Internet. Others go very broad, arguing anything you consume outside the firewall is "in the cloud," including conventional outsourcing.

Cloud computing comes into focus only when you think about what IT always needs: a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. Cloud computing encompasses any subscription-based or pay-per-use service that, in real time over the Internet, extends IT's existing capabilities.

Cloud computing is at an early stage, with a motley crew of providers large and small delivering a slew of cloud-based services, from full-blown applications to storage services to spam filtering. Yes, utility-style infrastructure providers are part of the mix, but so are SaaS (software as a service) providers such as Salesforce.com. Today, for the most part, IT must plug into cloud-based services individually, but cloud computing aggregators and integrators are already emerging.

As a metaphor for the Internet, "the cloud" is a familiar cliché, but when combined with "computing," the meaning gets bigger and fuzzier.

InfoWorld talked to dozens of vendors, analysts, and IT customers to tease out the various components of cloud computing. Based on those discussions, here's a rough breakdown of what cloud computing is all about:

1. SAAS (SOFTWARE AS A SERVICE)

This type of cloud computing delivers a single application through the browser to thousands of customers using a multitenant architecture. On the customer side, it means no upfront investment in servers or software licensing; on the provider side, with just one app to maintain, costs are low compared to conventional hosting. Salesforce.com is by far the best-known example among enterprise applications, but SaaS is also common for HR apps and has even worked its way up the food chain to ERP, with players such as Workday. And who could have predicted the sudden rise of SaaS "desktop" applications, such as Google Apps and Zoho Office?

2. UTILITY COMPUTING

The idea is not new, but this form of cloud computing is getting new life from Amazon.com, Sun, IBM, and others who now offer storage and virtual servers that IT can access on demand. Early enterprise adopters mainly use utility computing for supplemental, non-mission-critical needs, but one day, they may replace parts of the datacenter. Other providers offer solutions that help IT create virtual datacenters from commodity servers, such as 3Tera's AppLogic and Cohesive Flexible Technologies' Elastic Server on Demand. Liquid Computing's LiquidQ offers similar capabilities, enabling IT to stitch together memory, I/O, storage, and computational capacity as a virtualized resource pool available over the network.

3. WEB SERVICES IN THE CLOUD

Closely related to SaaS, Web service providers offer APIs that enable developers to exploit functionality over the Internet, rather than delivering full-blown applications. They range from providers offering discrete business services — such as Strike Iron and Xignite — to the full range of APIs offered by Google Maps, ADP payroll processing, the U.S. Postal Service, Bloomberg, and even conventional credit card processing services.

4. PLATFORM AS A SERVICE

Another SaaS variation, this form of cloud computing delivers development environments as a service. You build your own applications that run on the provider's infrastructure and are delivered to your users via the Internet from the provider's servers. Like Legos, these services are constrained by the vendor's design and capabilities, so you don't get complete freedom, but you do get predictability and pre-integration. Prime examples include Salesforce.com's Force.com, Coghead, and Google App Engine. For extremely lightweight development, cloud-based mashup platforms abound, such as Yahoo Pipes or Dapper.net.

5. MSP (MANAGED SERVICE PROVIDERS)

One of the oldest forms of cloud computing, a managed service is basically an application exposed to IT rather than to end-users, such as a virus scanning service for e-mail or an application monitoring service (which Mercury, among others, provides). Managed security services delivered by SecureWorks, IBM, and Verizon fall into this category, as do such cloud-based anti-spam services as Postini, recently acquired by Google. Other offerings include desktop management services, such as those offered by CenterBeam or Dell Everdream.

6. SERVICE COMMERCE PLATFORMS

A hybrid of SaaS and MSP, this cloud computing service offers a service hub that users interact with. They're most common in trading environments, such as expense management systems that allow users to order travel or secretarial services from a common platform that then coordinates the service delivery and pricing within the specifications set by the user. Think of it as an automated service bureau. Well-known examples include Rearden Commerce and Ariba.

7. INTERNET INTEGRATION

The integration of cloud-based services is in its early days. OpSource, which mainly concerns itself with serving SaaS providers, recently introduced the OpSource Services Bus, which employs in-the-cloud integration technology from a little startup called Boomi. SaaS provider Workday recently acquired another player in this space, CapeClear, an ESB (enterprise service bus) provider that was edging toward B-to-B integration. Way ahead of its time, Grand Central — which wanted to be a universal "bus in the cloud" to connect SaaS providers and provide integrated solutions to customers — flamed out in 2005.

Today, with such cloud-based interconnection seldom in evidence, cloud computing might be more accurately described as "sky computing," with many isolated clouds of services which IT customers must plug into individually. On the other hand, as virtualization and SOA permeate the enterprise, the idea of loosely coupled services running on an agile, scalable infrastructure should eventually make every enterprise a node in the cloud. It's a long-running trend with a far-out horizon. But among big metatrends, cloud computing is the hardest one to argue with in the long term.

Eric Knorr is InfoWorld's editor in chief; Galen Gruman is an InfoWorld executive editor.

CONTEXT

The 9 myths of cloud computing

Don't let the hype put you on the wrong path

By Robert L. Scheier

WHEREVER YOU TURN, SOMEONE'S READY TO TELL (OR SELL) you something related to cloud computing. Cutting through the myths is essential to deciding whether, when, and how the cloud is right for you. Here's our top list of myths.

MYTH NO. 1: THERE'S ONE SINGLE "CLOUD"

There are at least three forms of "cloud computing," each with different benefits and risks. They are 1) "infrastructure as a service" (bare-metal virtual servers available on demand from the likes of Amazon's Elastic Compute Cloud); 2) Web services providers, or "platform as a service," which are APIs or development platforms that let customers create and run apps in the cloud; and 3) software as a service, applications such as Salesforce.com's CRM software that users access over the Internet with little or no code running on their own machines.

The type of application you're running and the kinds of data you're generating also make a big difference in whether — and how — to move to the cloud. Which leads to:

MYTH NO. 2: ALL YOU NEED IS YOUR CREDIT CARD

If you're a lone developer with time to burn, configuring a virtual bare-metal server from the command prompt may be no problem. But if you have a business to run, installing and configuring the OS, multiple applications, and database connections could get in the way of generating revenue. And if you're big enough to have any standards for security, data formats, or data quality, someone has to do that work, too.

Some vendors imply that a business user "can just go in and buy a development server in 15 minutes that's as good as the one it would take their IT department three or four days to provision," says Michael Kollar, chief architect at Siemens IT Solutions and Services North America, which virtualizes about 2,500 servers to provide cloud-based application services to internal users as well as external customers. However, he says, that cloud-based server may not be secure, meet corporate standards, or be integrated into the wider IT environment.

For example, even a Web server thrown up in the cloud for a short-term marketing campaign might need to meet corporate security and data format standards. That's because the customer data it gathers is subject to the same corporate and legal standards as "real" IT systems, says Kollar, and it must be usable by corporate analytic or customer tracking systems.

Many infrastructure-as-a-service players also can't meet the needs of enterprise applications. Phil Calvin, founder and CTO of Sitemasher, tried to find a cloud provider to manage the servers he now manages himself in a collocation facility. However, he says, "we couldn't find anyone to scale our standard servers" on demand. Nor could the cloud vendors provide the low-latency performance he requires or do global load balancing across datacenters.

Amazon.com recently announced a public beta of new features that include auto-scaling, monitoring, and load balancing. In a blog post, cloud management vendor RightScale said the new capabilities were a step in the right direction but appeared to lack necessary capabilities such as configuration management and life cycle management.

MYTH NO. 3: THE CLOUD REDUCES YOUR WORKLOAD

In the long run, maybe. But to get started, you have to figure out which model of cloud computing is right for you; which applications or services are best suited to it; and how to ensure the proper levels of security, compliance, and uptime. And remember, monitoring the performance of any vendor takes extra time.

"When you're running production applications, there's a lot of thinking that goes on in terms of redundancy, in terms of reliability, in terms of performance and latencies," says Thorsten von Eicken, CTO and founder of RightScale. Before moving applications to the cloud, customers need to ensure those requirements are met, he says, calling it "wishful thinking" that cloud-based systems automatically manage themselves.

In addition, not all apps are right for the cloud. Those relying on clustered servers, for example, aren't good fits for cloud environments where they share resources with other customers, says James Staten, a principal analyst at Forrester Research. That's because they require identical configuration of each server and large dedicated bandwidth among servers, which can't always be guaranteed by a cloud vendor. Again, thinking through these issues requires work, at least up front.

Seamless integration is easier if you are running the same platforms in both the public and private cloud.

MYTH NO. 4: YOU CAN SEAMLESSLY BLEND YOUR PRIVATE "CLOUD" (YOUR VIRTUALIZED DATACENTER) WITH PUBLIC PROVIDERS

Some cloud evangelists hold out the promise of the best of both worlds: the control provided by an in-house datacenter and the low cost and flexibility provided by the cloud, with the ability to drag and drop applications, storage, and servers among them as needed.

But it's not yet that easy, at least for a complex multi-tier application that depends on internal databases and that serves thousands of users with ever-changing access rights.

"Currently, it takes a lot of footwork, and a lot of manual reconfiguration, and lots of engineering effort" to move applications among public and private clouds, says Staten. And even then, "we're still in the 'I hope it works' phase." Seamless integration is easier if customers are running the same platforms in both the public and private clouds, he says, but for the typical, more complex environments standards efforts such as the Open Virtualization Format are still "very basic" attempts to ease interoperability.

The key requirements, says Siemens' Kollar, are a security infrastructure that can span both environments, secure and cost-effective ways to either replicate data or access it across the public and private clouds, and orchestration software to ensure that services are working as required and proper steps taken to repair them if they aren't.

Renata Budko, vice president of marketing at virtualization management vendor HyTrust, says the best candidates for movement are those with relatively few modules and tiers, that are relatively "stateless" (not overly dependent on the timing and sequence of processing events), and those with relatively few user profiles to track. "If it's an internal cloud, you can access the policy database within the same cloud," she says, while customers may be reluctant to host sensitive security data in an external cloud or allow external access to their internal security data.

Having said that, beware of:

MYTH NO. 5: YOU WON'T EVER BE ABLE TO SEAMLESSLY BLEND YOUR PUBLIC AND PRIVATE CLOUDS

Vendors are scrambling to provide such seamless blending. Kollar, for example, expects to provide it to his customers within 12 to 18 months. Until it's widely available, RightScale's Von Eicken recommends standardizing configurations, data models, and automated deployment policies for both public and private clouds. That allows you to take advantages of the public cloud when it makes sense today, while building a foundation to do more sharing of public and private resources as the technology, standards and processes mature.

MYTH NO. 6: THE CLOUD ALWAYS SAVES YOU MONEY

McKinsey & Co. recently released a hotly contested white paper claiming customers are only likely to save money when running specific platforms, such as Linux, in the cloud. For an entire datacenter, the report says, you're better off staying in-house.

McKinsey declined to comment, but in a blog posting, Google Apps senior product manager Rajen Sheth said that the study erred by only considering the savings of using low-cost servers in a highly redundant architecture. It neglected, he says, the additional money customers save by using "the same scalable application server and database that Google uses for its own applications" and not having to purchase, install, maintain, and scale their own databases and application servers.

Another wild card, say Staten, is that under current licensing and support models, customers could pay significantly more to their commercial software vendors by deploying their software in the cloud than they would internally.

MYTH NO. 7: A CLOUD PROVIDER CAN GUARANTEE SECURITY

Even if a cloud provider has every security certification in the book, that's no guarantee your specific servers, apps, and networks are secure. When it comes to, say, compliance with the credit card industry's PCI DSS (Payment Card Industry Data Security Standard) a retailer or credit card processor is audited on how well their servers and applications are deployed on the platforms provided by a cloud vendor such as Amazon or Google. "If you set up your applications badly," says Staten, "it doesn't matter how secure the platform you're running on is."

Securing Siemens' cloud environment required looking at IT "from the outside in" and securing every conceivable path by which a user could access critical information, says Kollar. Securing each platform was not a significant challenge, he says, but ensuring all the needed security technologies worked together was.

Staten says it may require "architect-to-architect" sitdowns to assure a vendor hasn't, for example, cut costs "by simply giving each customer their own table space in the same database," as that would allow any customer to see any other customer's data.

In the cloud world, it's easier than in the physical world to assign new network interface cards to a virtual machine that might link it to an insecure network, says HyTrust's Budko. An organization's existing firewalls would have no way of knowing the new NIC exists and that it needs to monitor traffic through it, she says. Potential threats like that make it important to independently assess, rather than blindly trust, a cloud vendor's security infrastructure.

MYTH NO. 8: IF YOU'RE RUNNING VMS, YOU'RE DOING CLOUD COMPUTING

Virtualization — creating logical servers or storage that span multiple physical devices — is one of the requirements of cloud computing. But having VMs doesn't mean you have cloud computing.

To reap the full benefits of virtualization, IT or its cloud providers also must provide the ability to grow or shrink capacity as needed, provide pay-as-you-go pricing, and let users easily provision new servers and storage themselves as needed.

Letting users do some of the work of ordering virtual servers (especially those preconfigured for specific tasks) is a key money-saving goal of some cloud customers.

But such self-service doesn't automatically happen just because you're running software such as VMware Infrastructure 3. Siemens, for example, had to make "a significant investment" in developing a standard catalog of virtual servers and related services users can order as needed from its private cloud, says Kollar.

MYTH NO. 9: CLOUD COMPUTING IS ABOUT TECHNOLOGY

Technology makes cloud computing possible, but realizing cost savings and flexibility also requires that you have the right processes.

The virtualization that underlies cloud computing "is very dynamic and allows a very high rate of change," says Budko, as customers move data and applications among physical devices. "What's missing is the ability to manage it smoothly," avoiding a sprawl of unused or underused virtual machines that soak up electricity, cooling, and management time and possibly create security risks — just as unmanaged physical servers do.

Using standardized processes in the cloud can, on the other hand, increase efficiency. Using the Information Technology Infrastructure Library (ITIL) management framework in combination with technologies such as virtualization, Siemens has reduced its IT management and administration task by 25 to 35 percent, says Kollar.

THE TRUTH ABOUT THE CLOUD

What's the takeaway? That the cloud isn't a magic wonderland of carefree computing, but a complex resource that requires understanding and hard work to manage correctly. And that's no myth.

Robert L. Scheier is a freelance writer.

DEPLOYMENT

Cloud options that really help IT

Extend your datacenter infrastructure while saving big bucks

By Mel Beckman

BACK IN 1991, BEFORE THE INTERNET WAS A BIG DEAL, OHIO State University technologist Jerry Martin signaled the nascent Internet's value with an official standards document entitled "There's gold in them thar networks!" (formally called the RFC1290 paper). Although simmering as an academic tool for years, the Internet had not yet triggered a significant paradigm shift for commercial computing. Martin's formal proclamation was an early push to business, which eventually embraced Internet commerce wholeheartedly.

Cloud computing promises a similar, if not equivalent, kick in the paradigm, by shifting fundamental IT infrastructure from on-site, hands-on servers, disks, and networks to off-site, ephemeral cycles, bits, and bandwidth. That transition hasn't happened yet, but many pundits see it as inevitable. The main barrier is the cloud's unproven reliability — IT is loathe to put all the corporate computational jewels in a vapor-lined basket.

If the cloud isn't yet ready to take on traditional business tasks, does it have value to IT? Yes, it turns out. The cloud is full of resources that IT can use for its own purposes, from help-desk ticketing to disaster recovery.

As with early Internet adopters, IT shops have found the nascent cloud full of golden nuggets worth mining. The three primary cloud services identified in InfoWorld's analysis of cloud developments (see previous article) — infrastructure services, software as a service (SaaS), and development platforms as a service — provide a slew of labor- and cost-saving options for harried IT managers.

THE CLOUD'S MANY USEFUL TOOLS FOR USE BY IT ITSELF

Many an IT project starts with a month-long equipment acquisition timeline, followed by another month of installation, configuration, and setup. This front-end burden is often the kiss of death for smaller tasks. Two of the salient features of infrastructure cloud services — instant provisioning and scaling — head this problem off at the pass.

At its most basic level, infrastructure cloud providers sell the nuts and bolts of IT on a pay-as-you-go basis: server CPU cycles, storage gigabytes, and bandwidth megabits per second. These cloud services give customers the ability to launch self-contained application environments — servers, storage, and network connectivity — in minutes. Providers like Amazon.com, IBM, and Sun Microsystems deliver this utility-computing capability in the form of raw servers that you configure and manage yourself.

By themselves, these infrastructure components leave a lot to be desired. Yes, they save you the time and expense of capital equipment deployment, but you're stuck with the same configuration and integration chores as before. Worse, you have to perform these tasks remotely, and you carry the burden of bandwidth bottlenecks and strange new security risks. For steady-state workloads that can't take advantage of the cloud's rapid scaling capabilities, the effort hardly seems worth the trouble.

But the cloud value proposition changes dramatically when you factor in preinstalled, preconfigured virtual appliances, supplied by an army of third-party developers and conveniently delivered as ready-to-boot virtual disk images. We're not talking about major line-of-business applications such as CRM here, but IT-centric tools that frequently fall off the budget due to deployment costs. Help-desk ticketing, network management, vulnerability assessment, and enterprise knowledgebases are just a few of the applications you can spin up in the cloud in just minutes.

These applications fall into three broad categories: unsupported free open source software (FOSS), supported FOSS, and full commercial offerings. In the unsupported FOSS category are popular network administration tools such as Nagios, Cacti, and MediaWiki. Third-party cloud enablers such as JumpBox sell these same FOSS apps and dozens of others as support subscriptions for just a few hundred dollars a year. Virtual appliance migration tools such as Citrix's Kensho and RPath's rBuilder provide physical-to-virtual (P2V) migration engines that let you move most any FOSS appliance to an infrastructure service such as Amazon's EC2.

Not all of these applications benefit from offsite hosting, but some definitely need it. For example, Tenable Network

Security's Nessus vulnerability assessment tool by definition resides outside your network, where it simulates hacker attacks to ferret out any border security weaknesses. But often the initial savings in time and labor are enough to justify even simple cloud-basing projects.

A number of hybrid service products are appearing as well — a cross between FOSS and commercial software, offering both customer-managed cloud deployment and vendor-managed SaaS. Kayako offers its line of help-desk portal products as purchasable software — including the source code — and as a fully managed hosted service. Clients are free to move their data between items, and thus can start out with the managed service for less than $50 per month and migrate to a self-managed cloud deployment when their needs warrant.

GAIN CHEAPER EMERGENCY PREPAREDNESS WITH CLOUD-BASED DISASTER RECOVERY

A down economy and constricting budgets tend to force spending cuts in areas that don't contribute directly to the bottom line. One of the first cuts many organizations make is to expensive disaster recovery services. You might think such economies ill-advised, but the conventional wisdom is that your enterprise's existence trumps business continuity concerns. The $5,000-per-month hot site that never gets used represents a job or two, and thus becomes an attractive target.

But it need not be that way. Infrastructure virtualization theoretically lets you replicate your business processes in the cloud, where they can lie dormant at very low cost until you need them in a disaster. The emphasis here is on "theoretically." Moving physical applications to the cloud and keeping cloud-resident data reasonably up-to-date requires considerable skill and finesse. You trade "instant failover" for dramatically lower monthly costs, but keep the peace of mind that comes from knowing your business DNA is safely archived in a distant state or country.

The skills needed for cloud disaster-recovery implementation are within the abilities of most IT technologists, but if your company is small and consultant-dependent, you'll have to get outside help. Consulting firms are stepping up to the plate, creating cloud-oriented disaster-recovery service packages that handle the headaches for less-sophisticated users, while still reaping the bulk of cloud economies of scale.

One constraint of such services is a client's local Internet connection speed. But speeds are increasing as costs plummet, especially as fiber connectivity options penetrate business markets; most are adequate for nighttime backup synchronization. One consultancy that offers a cloud-based disaster recovery service, CompuVision, uses a 100Mbps Internet service center to provide fast data transfers during an outage, for example.

RUN YOUR APP DIRECTLY ON A CLOUD TO LOSE INFRASTRUCTURE HASSLES

A few cloud providers — Microsoft and Google among them — foresee application development moving straight to the cloud, bypassing the traditional server-OS-storage platform. Although not yet ready for prime time, Microsoft's Azure aims to leverage the skill set of existing .Net developers to let them code, test, and deploy applications without concern for the OS or hardware on which they run. InfoWorld's Test Center drive of Azure finds its architecture well conceived but concludes that it's too soon to predict its role as a major cloud offering.

Google's much more lightweight App Engine, also only available in beta but slightly more baked than Azure, focuses on a much smaller audience: Python developers. Billed as a thin layer of Web-enabled Python with fat Internet connectivity and automatic performance scaling, this is an easier tool for most developers to get their arms around.

Software engineering consultant Denny Bollay has examined both Amazon's EC2 and App Engine: "EC2 is fine for what it is, but someone has to play system administrator, a chore that software engineers don't want. App Engine looks like a nice first cut at a streamlined cloud application platform environment, but it has issues like cost prediction and vendor lock-in. What I really am looking for is a cross between Amazon's nonproprietary cloud and Google's cloud compiler with BigTable database. And I'd like to see data providers in the mix, delivering real-time streams of weather, stocks, news, and the like that I can process on the fly in App Engine or its equivalent. Cloud-seeding, as it were."

Although Microsoft's Azure supports open Web application standards, such as REST and AJAX, App Engine has spawned a fledgling open source community with actual FOSS App Engine components. Many of these are variations on the Google-supplied (and FOSS) Gaeutilities and

Continued on page 21

DEPLOYMENT

Test Center: Cloud versus cloud

A guided tour of Amazon, Google, AppNexus, and GoGrid

By Peter Wayner

WHO WOULDN'T WANT TO LIVE IN A "CLOUD"? THE TERM IS A perfect marketing buzzword for the server industry, heralding images of a gauzy, sunlit realm that moves effortlessly across the sky. There are no suits or ties in this world, just toga-clad Greek gods who do as they please and punish at whim, hurling real lightning bolts and not merely sarcastic IMs. The marketing folks know how to play to the dreams of server farm admins who spend all day in overgrown shell scripts and impenetrable acronyms.

To test out these services, I spent a few days with them and deployed a few Web sites. I opened up accounts at four providers, configured some virtual servers, and sent Web pages flowing in a few hours. Our choice of four providers wasn't as scientific as possible because there are a number of new services appearing, but I chose some of the big names and a few new services. Now, I can invoke Joni Mitchell and say I've looked at both sides of these services and offer some guidance.

The first surprise is that the services are wildly different. While many parts of Web hosting are pretty standard, the definition of "cloud computing" varies widely. Amazon's Elastic Compute Cloud offers you full Linux machines with root access and the opportunity to run whatever apps you want. Google's App Engine will also let you run whatever program you want — as long as you specify it in a limited version of Python and use Google's database.

The services offer wildly different amounts of hand-holding, and at different layers in the stack. When this assistance works and lines up with your needs, it makes the services seem like an answer to your prayers, but when it doesn't, you'll want to rename it "iron-ball-and-chain computing." Every neat feature that simplifies the workload does it by removing some switches from your reach, forcing you into a set routine that is probably but not necessarily what you'd prefer.

After a few hours, the fog of hype starts to lift and it becomes apparent that the clouds are pretty much shared servers just as the Greek gods are filled with the same flaws as earthbound humans. Yes, these services let you pull more CPU cycles from thin air whenever demand appears, but they can't solve the deepest problems that make it hard for applications to scale gracefully. Many of the real challenges lie at the architectural level, and simply pouring more server cycles on the fire won't solve fundamental mistakes in design.

By the end of my testing, the clouds seemed like exciting options with much potential, but they were far from clear winners over traditional shared Web hosting. The clouds made some things simpler, but they still seemed like an evolving experiment.

AMAZON ELASTIC COMPUTE CLOUD

Amazon was one of the first companies to launch a product for the general public, and it continues to have one of the most sophisticated and elaborate set of options. If you need CPU cycles, you can spin up virtual machines with Elastic Compute Cloud (EC2). If it's data you want to store, you can park objects of up to 5GB in the Simple Storage Service (S3). Amazon has also built a limited database on top of the S3, but I didn't test it because it's still in a closed beta. To wrap it up, your machines can talk among themselves with the Simple Queue Service (SQS), a message-passing API.

All of these services are open to the Web and accessible as Web services. There's a neat demo for the SimpleDB that is just a pile of HTML running in your browser while querying the distant cloud. The documentation is extensive, and Amazon makes it relatively easy to wade through the options.

The ease, though, is relative because almost everything you do needs a command line. Amazon built a great set of tools with sophisticated security options for sending orders to your collection of machines in the sky, but they all run from the command line. I found myself cutting and pasting commands from documentation because it was too easy to mistype some certificate file name, for example.

Unix jockeys will feel right at home in this world because the virtual machines at your disposal are all versions of Linux distros like Fedora Core 4. After you grab one off the shelf, you can install your own software and create a custom instance that can be loaded relatively quickly if there's space available in the cloud.

It's hard to go into enough detail about all of the offerings described here, but Amazon is the most difficult because it has the most extensive solutions. Amazon is thoroughly committed to the cloud paradigm, rethinking how we design these systems and producing some innovative tools.

■ Watch InfoWorld's QuickTime video of Amazon

GOOGLE APP ENGINE

Google's App Engine is a polar opposite of Amazon's offering. While you get root privileges on Amazon, you can't even write a file in your own directory with the App Engine. In fact, it's not even clear that you get your own directory, although that's probably what's happening under the hood. Google ripped the file write feature out of Python, presumably as a quick way to avoid security holes. If you want to store data, you must use Google's database.

The result of all of these limitations is not necessarily a bad thing. Google has stripped Web applications down to a core set of features and built up a pretty good framework for delivering them. I was able to write a simple application with several hundred lines of Python (cutting and pasting from Google's documentation) in less than an hour. Google offers some nice tools for debugging the application on your own machine.

Deploying this application to the cloud should have taken a few seconds, but it was held up by Google's insistence that I fork over my cell phone number and wait around for a text message that tests the number. When my message didn't show up for several hours after retrying, I switched to a friend's phone and finally activated my account.

Google insists on linking your App Engine account to both your cell phone and your Gmail account because — well, I don't know. I think it's to track down the scammers, spammers, pharmers, phishers, and other fraudsters, but it starts to feel a bit creepy. Maybe it will help customer service and allow them to field support requests with answers like, "Your cell phone shows you filed this report from a location with a liquor license. Your e-mail suggests you're coding while waiting for Chris to get off of work. We suggest going home, sleeping this off, and then it will take you only a few seconds to find the endless loop on line 432 of main.py. BTW, Chris is lying to you and is really out with someone else."

The best users for the App Engine will be groups, or most likely individual developers, who want to write a thin layer of Python that sits between the user and the database. The API is tuned to this kind of job. In the future, Google may add more features for background processing and other services such as lightweight storage, but for now, that's the core strength of the offering.

■ Watch InfoWorld's QuickTime video of App Engine

GOGRID

GoGrid refers to itself as the "world's first multiserver control panel." GoGrid's offerings aren't functionally different from Amazon's EC2, but using the old term "control panel" seems to be a better description of what's going on than the trendier term "cloud." You start up and shut down load balancers in much the same way as relatively ancient tools like Plesk and cPanel let you add and subtract services.

While GoGrid offers many of the same services as Amazon's EC2, the Web-based control panel is much easier to use than the EC2 command line. You point and click. There's no need to cut and paste information because little pop-up boxes show the way, by suggesting available IP addresses, for example. The system is intuitive, and it takes only a few minutes to build up your network. A simple ledger on the left keeps track of the costs and helps you manage the budget.

GoGrid also has a wider variety of OS images ready to go. There is the usual collection of CentOS/Fedora and common LAMP stacks. If you need Windows, you can have Windows Server 2003 with IIS 6.0, and Microsoft SQL Server is available at extra cost. There are also images with Ruby on Rails, PostgreSQL, and the Facebook application server. These make it a bit easier to start up.

While GoGrid offers many of the same features as Amazon's EC2, it doesn't provide more cloudlike services for storing information in a shared way like SimpleDB. This can make it a bit harder to start up and shut down servers without a bit of grief. The startup notes for the service point out that the only way to stop paying for a server is to delete it, and that means losing all of the data on it.

There's no simple way to build custom images at this moment, but the documentation says GoGrid is working on a way to turn any running server into an image that can be restarted later. If you're going to be expanding and contracting your network as the traffic ebbs and flows, you'll have to come up with some tools of your own to add and subtract these servers.

- Watch InfoWorld's QuickTime video of GoGrid

APPNEXUS

If you like the idea of the cloud but aren't sure if you want to leave behind the old trustworthy world of Unix, cron jobs, and other tools, then AppNexus is a service that aims to be a bit more transparent. The company has taken a big, industrial-sized server farm with the best load-sharing tools and storage boxes and found a way to let you buy it in small portions. AppNexus provides a number of command-line abstractions that let you turn servers on and off, but they also let you drill down into the file system.

The main functions of the AppNexus cloud are similar to Amazon's EC2. You log in through a command line and boot up images of Linux distributions. AppNexus says it can rebuild images from other sources like Amazon's EC2 by replacing the kernel with a version that's more aware that it is running in a virtual environment. Then it just takes a few key clicks on a command line to set up a load balancer.

One open question in the world of cloud computing is where the abstraction occurs; that is, where do the walls between the machines become blurred and it all starts to look a bit cloudy? Amazon's SimpleDB hides the storage behind a software wall and gives you access to it through some Web service call. AppNexus is working at a lower level by building in a cluster of Isilon IQ X-Series storage clusters into its cloud.

This gives you the option of simply mounting the storage and sharing the data across your cluster of servers — if you consider that simple. Instead of working with abstract keys, you use real file names as the keys. The cluster handles the rest of the work.

A better solution is to use what AppNexus calls its CDN, or Content Delivery Network. The storage cluster has its own set of HTTP servers built in, and you can automatically begin serving static data from your files. Just write the files to the /cdn directory and they become available. AppNexus will distribute this storage cloud to multiple datacenters, making it simpler to serve up the static data from the closest location.

- See InfoWorld's QuickTime video of AppNexus

THE FINE PRINT

One of the ways to go truly insane is to read the terms of service for these clouds. While the people who wrote the old co-location contracts could try to imagine the data as living on a single server that was in a certain box owned by a certain person and residing in a certain jurisdiction, all bets are off with a cloud. The whole point is that it isn't confined to one box, one building, or even one country.

Some of the service agreements are very specific and clear. GoGrid, for instance, spells out numerical thresholds for standard values such as latency, jitter, and packet loss for the six continents. If the cloud doesn't meet them, GoGrid promises to give you service credits for 100 times the amount lost.

Other terms are deliberately murky. You might consider it fairly capricious for Amazon to demand the right to terminate your account "for any reason" and "at any time," but the company also carefully reserves the right to terminate your account for "no reason" too. In other words, "It's not you, honest. It's me. No. I take that back, it's not even me. It's just over between the two of us. No reason."

Google's terms seem more generous, indicating it will terminate accounts only if you breach the terms of the agreement or do something unlawful. But Google does reserve the right to "pre-screen, review, flag, filter, modify, refuse, or remove any or all Content from the Service." I want to say that the terms seem more reasonable than they were when I read them several weeks ago, but I can't be sure. And it doesn't matter too much because new terms apply whenever Google wants to change them, and you signify your acceptance by continuing to use the service.

If you think it's hard to work through the legal rules when a server is in one state and a user is in another, imagine the right answer when your virtual server could migrate within a cloud that might encompass datacenters spread out across the globe. Amazon's terms, for instance, prohibit you from posting content that might be "discriminatory based on race, sex, religion, nationality, disability, sexual orientation, or age." It sounds like Amazon is worried that part of the cloud might touch down in a municipality that forbids things like this.

It almost seems scary to mention this fact, but New York is insisting that Amazon charge sales taxes because Amazon pays a commission to Web sites that do business in the state. What does this mean for applications hosted by Amazon? Do you owe sales tax if your application touches down in a part of the cloud that's in New York? Do you owe income tax?

I wanted to make some allusion to Schrödinger's cat and imply that we can't know where the computation occurs in the cloud, but then I slowly realized that this is far from true. Cloud servers have log files too, and these log files can produce insanely detailed analyses of who might owe which taxes. Major league athletes already hire tax attorneys to compute their share of income earned in each stadium, and some people are suggesting that Web companies aren't paying enough to support the local fire trucks and orphanages. Say good-bye allusions to Joni Mitchell; it's time to start invoking Warren Zevon's "Lawyers, Guns, and Money."

CRASHING THE CLOUD METAPHOR

The legal worries are just part of the details that aren't so certain. One of the biggest dangers is reading too much into the cloud metaphor. While it's largely true that these services are very flexible ways to build up a network of machines, they are far from perfect. What happens if a server or a hard disk crashes in the middle of an operation? Often the same thing that happens when a generic server kicks the bucket: Your data might disappear and then it might not.

An instance of a machine from Amazon's EC2 looks just like a normal machine because after you strip away the hype, it is just another version of Linux running on a chip that probably speaks 8080 machine code and writes data to a spinning platter. If you write something to a good old file in the Unix file system, the cloud metaphor won't protect it. It will stay there until the machine dies. If you shut down the server to save some cash when traffic is low, that's the same thing as dying. That means you can't really scale up and down without a savvy plan for migrating data.

In other words, MySQL in the cloud works just like it does on a generic server. Everything could be lost in a poof unless you start up several instances and mirror them with each other. The magic of the cloud metaphor can't remove this fundamental rule.

If you want something to survive a crash, you've got to put it into the cloud's data stores. These are great services, but they're not cheap. One friend of mine used to back up his disks to Amazon's S3 until he started getting bills for more than $200 a month. He bought a hard disk and kept it on his desk.

The price is higher because the service level is higher. Amazon wants people to be able to trust the data store, and that means providing a level of service that would make a bank happy. Sharing data across servers takes time and careful coding. Google cautions users to be careful writing to its data store because it can be expensive. If you're someone who likes to keep lots of log files just in case, you'll probably pay much more to store them in the cloud than you would in a regular file. Alas, Google doesn't have regular files.

One of the trickier details is trying to understand the prices. GoGrid, for instance, likes to say that its Intel Xeon servers are more powerful than its competitors. Google doesn't even sell server time per se; it just bills you for CPU megacycles, a squirrelly metric. Amazon EC2 has regular-sized machines and bigger ones that are a bit more expensive. When costs change, the companies often lower their prices. But they also raise them when a service turns out to be more expensive to provide than they thought. This complexity will have you scratching your head for a long time because it's hard to know what things will end up costing. That box from Sun may not scale up and down, but the bill isn't going to change with every hit on your Web site.

BEST AND WORST

After working through these systems, I tried to imagine the best and worst applications for these clouds. One of the best fits might be some kind of reservation system for weekend events like concerts. While there might be a small amount of the load at any time, the crunch would come each Friday afternoon when people realize they have no weekend plans. The cloud's ability to spin up more servers to handle this demand would fit this perfectly. The service might also take real reservations and sell tickets in advance, a service that would demand the higher qualities of service offered by the shared data stores.

The worst possible application might be something like RedSoxYankeesTrashTalk.com or any Web site filled with an endless stream of mostly forgettable comments trolling for reactions from the rival fans. While there might be a slight peak around game time, I've found that sites like this keep rolling along even late at night during the off-season. And such a site would certainly attract First Amendment proponents who would look for ways to write a single sentence that could zing all seven of Amazon's protected targets of discrimination.

Furthermore, there would be no reason to pay for high-quality storage because I'm sure that even the participants wouldn't notice if their comments disappeared by mistake. For fun, read Amazon's terms on getting your data back after they shut you down. While I would probably write the same thing if it were my cloud, there are plenty of examples of applications that are better off on their own.

These examples aren't perfect, of course, but neither is cloud computing. After a few weeks of building up some machines and hearing from people who've used the services, I'm pleasantly confused and filled with curious and optimistic questions. Will these clouds be large enough to handle the Internet equivalent of the Thanksgiving weekend traffic jams? Will the cloud teams be able to find a way to offer simple options that are priced correctly for the serious and not-so-serious data wrangler? Will they ever find an adequate meter for computation time?

I suspect the only people who know the answers to these questions today are living in the real clouds where they went after a life ministering to the IBM mainframes. If we could get those guys back here today, we might be able to get this cloud thing up and running smoothly. We just have to convince Intel to build a chip that understands IBM 360 binaries.

Peter Wayner is a freelance writer and InfoWorld contributing editor.

Cloud Options That Really Help IT

Continued from page 16

provide various computational widgets that simplify App Engine development. Others, such as Nuages, cpedialog, and KGPL, are full-blown Web applications that you can run as is or use as a starting point for your own apps.

CLOUD COMPUTING'S CAVEATS EMPTOR

Cloud computing has some attractive low-hanging fruit for IT shops, but you should take care to count the cost before deploying in today's cloud marketplace. Some cloud computing risks are easily discerned: reliability, security, and performance. It's too soon to put mission-critical apps in the cloud unless you do the necessary homework to ensure adequate failover mechanisms, and that any sensitive data meets the ethical and legal standards for which you're accountable. Thoughtful preparation can keep you out of the cumulus-granite, but you should select applications that can tolerate a modicum of outages. Some will occur as a result of your own human error, but others will be disturbances in the clouds themselves.

A second potential pitfall is cost containment. Cloud providers are in the business of selling services, not aiming to minimize your expenses. It's your responsibility to closely track costs, and if you don't keep an eye on metered services, you can find a hefty bill in your inbox. Cloud purveyors don't make cost tracking easy. Amazon, for example, provides an excruciatingly detailed log of every CPU minute consumed, data byte stored, and megabyte transferred, but it provides no cost calculations for those statistics. You get a lump sum bill for each Amazon service you use — EC2, S3, and so on — with no detailed explanation of charges.

The second driver of unexpected cloud expense is the cloud's own ease of use. Spinning up a server — or 10 — only takes a minute. But servers stay spinning, and clocking dollars, until you turn them off. Third-party cloud management services like Rightscale and Elastra can automate the cost accounting process, as well as set hard spending limits. But you pay for that convenience — a minimum of $500 per month for Rightscale's auto-scaling cloud management console, for example.

As long as you keep these precautions in mind, there's no reason not to leverage cloud services to shorten your IT hit list today.

Mel Beckman is a freelance writer, as well as a network engineering consultant.