Clouds in My Condo

By Bennett Reddin and Bradley Rhine, Tilting Windmills

Cloud computing is the new techno-magic Oz, orbited by an El Dorado of Big Data, the Barsoom of Social, and the Xanadu of Gamification. It seems magic, or at least illusion; the 18 year old daughter sees the cloud as a place to access her music (and her father’s account to pay for the music) while the spouse sees the cloud as the place where gmail wonderously appears, and the sales force sees the cloud as, well, SalesForce, on which they operate their business, while the technologist views the cloud as a place to provision hardware resources on the cheap. Somewhere in this ephemeral domain, Amazon knows what we’re going to buy next, LinkedIn notifies us of every minute movement our business peers make, and Facebook keeps us forever glued to Farmville. So, how are we to pierce the veil, to make informed decisions about where to allocate our precious budgets, and to understand how all of this fits together?

With so many simultaneous and competing perspectives of cloud computing, we’re all too easily relegated to the role of spectator at the magic show, waiting for the “Man Behind the Cloud” to present yet another trick to dazzle. As vendors, we’re all too willing to stand behind that curtain and claim “cloud-enabled,” “cloud-capable,” and more, waiting for the dazzled spectators to purchase these advanced wonders.

One approach to understanding cloud computing is the analogy of condominiums—as home ownership compares to condos, so on-premise systems compare to cloud systems. Home ownership has many advantages, including the freedom to have children, pets, or even a vegetable plot in the yard, not to mention a woodshop or project car in the garage. Such ownership comes with responsibilities as well, from yard work to individual utility bills, plus maintenance and repairs. With condominium life though, plumbing (with unlimited hot water for showers!) is provided, along with common heating & air conditioning. When these services break, a building engineer takes care of repairs for all the tenants. Tenants, though, don’t have the freedom to garden, or run noisy woodshop tools—some limitations come along with the conveniences of the shared tenancy model of living.

Cloud computing holds many parallels to condos—significant capabilities are present out-of-the-box with cloud delivery, from access to hardware and networking, to extension and configuration of complete functionality in the software housed in the cloud. Limitations to that configuration capability exist, of course, leading to recommendations to follow a vendor’s practices in implementation. However, the simplicity, power, and cost of cloud delivery are often considered as outweighing any configuration limitations.

The formal definition of cloud computing is quite straightforward though, allowing us to shift from spectators in the show to directors guiding our own destiny. The National Institute of Standards and Technology (NIST) defines the essential characteristics, service models, and deployment models of cloud computing:

**Essential Characteristics of Cloud Computing**

**On-Demand Self-Service** – a consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed, automatically and without requiring human interaction with each service provider.

**Broad Network Access** – Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops and workstations).

**Resource Pooling** – The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

**Rapid Elasticity** – Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

**Measured Service** – Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service. Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

These essential characteristics are summarized as “Utility Computing,” where the applications and processing needed by a consumer are as available as we currently experience electricity. Today we can find a socket for our appliances, plug them in ourselves, and expect that the necessary power is available—despite how much or how little power is needed. We’re not concerned with whether the power itself is generated in the next town, or the next state, and we know that the power usage is measured at the meter so that we can monitor and pay by our own usage.

In the same manner, cloud computing gives us the ability to take part in processes from the smallest connected cell phone to the most powerful workstation, scaling the distant processing power from individual
employee address lookups to running payroll for thousands of employees. And we're allocated that payroll processing power only as we need it; we're not paying for massive hardware power when it's idle. (In fact, someone else may well be using that hardware when our processing needs are idle.)

**Service Models**

NIST defines three service models for cloud computing; and we've become familiar with many "as-a-Service," brandings:

**Software-as-a-Service (SaaS)** – The capability provided to the consumer to use the provider’s applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client (Web browser) or program interface. The consumer doesn’t manage or control the underlying cloud infrastructure, with the possible exception of limited user-specific application configuration settings.

**Platform-as-a-Service (PaaS)** – The capability provided to the consumer is to deploy, onto the cloud infrastructure, consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure, but has control over the deployed application and possibly over configuration settings for the application hosting environment.

**Infrastructure-as-a-Service (IaaS)** – The capability provided to the consumer is to provision processing, storage, and network resources where the consumer is able to deploy and run software, including operating systems and applications. The consumer doesn’t manage or control the underlying cloud infrastructure, but has control over the processing, storage, and applications installed.

As functional consumers, we typically have visibility only to SaaS. We access application functionality and configure capabilities in Salesforce or Workday, with no control, or concern about the development underpinnings and hardware through which the functionality is delivered to our browser or mobile device.

Delivering this SaaS functionality requires some sort of development platform underneath. Most HCM vendors have proprietary approaches to developing application functionality, primarily so they can always certify delivery adhering to common business rules across client instances. In contrast to these “Private PaaS” capabilities, we can look to examples of Google’s AppEngine or Salesforce’s AppExchange to see where public PaaS is available to extend the capabilities of a vendor’s offerings.

The hardware and network components on which SaaS, and potentially PaaS, run (including software to manage these resources) is properly called Infrastructure-as-a-Service. Again, many HCM vendors operate on private IaaS that is housed in their own data centers, but public IaaS is available in offerings like Amazon Web Services and Rackspace.

There’s a great deal required behind the scenes to bring this functionality to us as consumers. It’s easy to start with the familiar analogy of an iceberg, with a tremendous amount of “volume” hidden from us to execute on this delivery. However, there are myriad choices vendors can make in technology – what’s behind the curtain isn’t a monolith, identical from vendor to vendor. Were we to apply a monolithic definition to the entire delivery chain, no vendor today could be called “True SaaS.” It’s more accurate to provide a complete summary of an offering, that illuminates the spectrum of answers:

“HCMVendor provides SaaS HCM, via a multi-tenant delivery model, built on a publicly available development platform, hosted in private co-located data centers worldwide.”

-OR-

“Acme ASP provides SaaS HCM, via a single-tenant delivery model, built on the proprietary BigERP platform, hosted in private co-located data centers.”

-OR-

“ERP Vendor provides premise-based HCM, via a single-tenant delivery model, built on the proprietary BigERPvendor platform, hosted in the client’s own datacenter.”

This description framework allows several questions to be answered in a single response:

1. Is the application hosted on-premise, SaaS, or can it be deployed in either mode?
2. Is the application deployed in a single or multi-tenant mode?
3. Is the application built using a consumer-available development platform (PaaS)?
4. What hardware and network infrastructure is used to deliver the application (IaaS)?

There’s a simple assertion to this framework: “If the software is hosted by the vendor outside the client premises, it qualifies as SaaS.” Of the example statements above, the third, premise-based implementation, is clearly indicated as “not-SaaS.”

We’ve demonstrated over several decades now that premise-based software, especially ERP, is costly to maintain in licenses, maintenance, hardware, and human resources. Initially, moving the software, hardware, and support resources offsite was a huge chasm to cross. With early ASP offerings, vendors could leverage some economies of scale to reduce the cost of delivery, when individual enterprises couldn’t make such reductions. Even the reduction to “one Oracle database serving several clients” was a significant cost savings that the vendor could pass on to enterprises, with no reduction in service capabilities.

Once the chasm was crossed, the driving force behind cloud evolution and maturity became economics. The computing industry has undertaken Herculean efforts to mature its software architectures, network and hardware capabilities in order to deliver the greatest amount of functionality and configurability at the lowest cost possible. The results thus far lead to the inevitable conclusion that mature and cost-efficient SaaS applications are provided “via a multi-tenant delivery model.”

Tenancy, simply put, describes the level of sharing between application components in a SaaS offering. For instance, ASP vendors in the 1990s would provision one running version of software installed and customized for each client—a single-tenant, no-sharing model of deployment (Figure 3). While this model serves customization needs for clients, it incurs tremendous overhead for vendors to maintain all the disparate customizations that have been delivered to clients, especially in the face of ongoing upgrades to the core software offering. A step forward in deployment evolution had identical software (no customizations, but only minimal configuration capabilities) provisioned one-to-one for each client, reducing management overhead, but still accomplishing little to reduce the level of data center expenditures (Figure 4).

Modern SaaS architectures strive for a high degree of sharing, with all client data stored in one database, and all client business processes driven from a single configurable version of the application code (Figure 5).

---

Figure 2. The Premise-based to SaaS Economy Spectrum.

Figure 3. A single-tenant, no-sharing Model of Deployment.

Figure 4. Identical Software, No Customizations, Minimal Configuration Capabilities.

Figure 5. Modern SaaS Architectures.
To summarize, tenancy solutions span a spectrum, where generally speaking:

- There’s greater capability for customization, but little economy of scale at the single-tenancy end of the spectrum; and
- With less customization, but greater economy of scale the farther toward multi-tenancy a vendor chooses to execute.

The most sophisticated multi-tenant architectures have all clients connect to a single version of application functionality running on resources that scale automatically to meet processing demands (Figure 6). This deployment model allows the vendor to service an open-ended number of clients by adding server resources as needed, while also minimizing operational and programming overhead by maintaining only one version of software that is always current and known.

An early implementation vendor for the Abra HR system marketed the catchphrase, “It’s not magic,” playing on the vendor name. Cloud computing in HRIS isn’t magic, but still holds tremendous potential for HR information delivery in our demanding business environments.

Endnotes

1 The NIST Definition of Cloud Computing, Special Publication 800-145, Peter Mell & Timothy Grance
2 Wired magazine 4.12 (December 2004) “Mother Earth Mother Board” Neal Stephenson, originally, “It behooves wired people to know a few things about wires – how they work, where they lie, who owns them, and what sorts of business deals and political machinations bring them into being.”

About the Authors

Bennett M. Reddin is founder and chief technologist of TiltingWindmills, applying his HRIS expertise extending over 25 years from early PC-based HR systems to current service-oriented architectures. Alternating between development and implementation, he has designed several HR and Payroll architectures for service bureaus and ERP vendors, then deployed those systems, improving toolsets and practices for implementation in the process. In the role of systems architect for consulting and software vendors, Reddin is designing HRIS to meet the needs of the 21st century enterprise. Consulting directly to these enterprises, he is helping them lay the strategic and technology foundations for emerging innovation in HRIS. Frequently contributing to HR strategy and technology publications, he can be contacted at Bennie@TiltingWindmills.com.

Bradley Rhine spends his days (and some nights) building and supporting “always on,” cloud-based services for the HR, Payroll and Benefits business domains. His work spans front-end applications, security infrastructure, database, data integration and analytics, network protocols, operating systems and HA/DR/BC. As CIO for Advantec, he incrementally deployed an open source, SaaS HR platform, while maintaining 24/7 operation of the legacy system. His previous experience includes director of development, TriNet; CEO and co-founder, Cogentes; VP Systems Engineering, Ceridian; and VP at Concur Technologies. Earlier, he built intelligence data handling systems for the U.S. Department of Defense. He can be contacted at Bradley@TiltingWindmills.com.