

# InformationWeek

## 7 Cloud Computing Myths Busted

**Amazon, Google, Microsoft and others are investing aggressively in the cloud, even as critics point to security, reliability, and compatibility issues. We cut through the fog.**

By [Serdar Yegulalp](#)  
[InformationWeek](#)

November 14, 2009

What is it about "the cloud" that has people, well, getting their heads up in the clouds over it? Almost no other IT innovation in recent memory has engendered this much enthusiasm -- and furor, and confusion, and outright misunderstanding.

The cloud isn't exclusively a cure-all or a calamity in progress; neither is it a savior or sinner. It's a new tool for solving emergent problems, and like every new hammer in someone's hands it can make everything look like a nail.

In this piece we'll examine many of the current myths -- good, bad, and bogus -- about cloud computing. Many are borne by simple ignorance or inexperience. Others are legitimate criticisms in the guise of gripes. And some are entirely too on target, and need to be nipped in the bud by prospective cloud-creators before they get bitten by them.

### 1. Compatibility Issues

*Myth: Cloud computing is too proprietary.*

At present, no two clouds are alike -- both in nature and in IT. Amazon's cloud platform is nothing like Google's, which is nothing like Microsoft's, which is nothing like and you can insert the name of any other [up-and-coming cloud provider](#) here.

And yet "proprietary" has not proved to mean "useless" -- not by a long shot.

Think back to the early days of the personal computer. The first wave of PCs were all from different makers, used different hardware, and weren't remotely cross-compatible. Programs written for the Apple II weren't assumed to have any interchangeability with the Atari, the Amiga, or even the IBM PC itself.

What few common platforms that existed -- e.g., CP/M -- were largely for the sake of porting and running existing applications to those platforms, rather than for creating a crossbar of compatibility among them. None of this stopped a remarkable amount of development from taking place -- and the various platforms were able to compete heavily based on their differences.

Granted, the situation today is totally unlike that. People expect a great deal more cross-compatibility as a matter of course -- between devices, between applications, between platforms and environments. What's most proprietary about the platforms isn't so much the way they work on the inside as the fact that talking

to each cloud, getting data into and out of each cloud, and managing functionality within each cloud are all done differently.

The proprietary nature of the first wave of cloud computing platforms is, for lack of a better way to put it, a necessary evil. And maybe even not all that evil in the first place, when it grants you access to platforms like Linux (Amazon.com) and languages like Python (Google), which on their own terms are as open as they get. Things could be made less proprietary outside clouds and among clouds, although odds are [the standards](#) that will exist between clouds will develop more as a consequence of what people are actually using (e.g., EC2) rather than something drafted in the abstract.

## 2. Privacy Concerns

*Myth: Cloud computing is the end of privacy as we know it.*

Privacy fears over cloud computing can be seen as an outgrowth of privacy concerns in general, with cloud computing just being the bogeyman / whipping boy of the moment. That said, there's solid reasons to be skeptical -- or, if you're a creator of cloud-based services, to be cautious.

What makes cloud computing such a fierce target for privacy advocates is not only the newness of the technology, since every freshly minted technology is a possible privacy suspect. It's also the fact that cloud computing, on the face of it, can cause a huge degree of aggregation across multiple IT spheres. When you have many disparate things suddenly all under one roof, it translates into "single point of failure" and "all your eggs in one basket." It's not your data anymore, either; it's someone else's, and whatever happens will happen on his watch. There's a chance that provisions about your data security aren't even in the contract you signed.

It's difficult to write off such concerns as mere paranoia -- not when most data leaks and theft happen from *within* organizations as inside jobs, rather than from outside attacks. Worse, people who store their data on other people's systems might not have the law on their side when expectations of privacy become a legal issue.

Cloud providers need to be proactive about this, early in the lifecycle of their services. They should spare no expense to make it clear to their customers -- and, by extension, their customers' customers -- that data and process security can be protected from outside attacks and internal theft, and where they stand vis-à-vis the law whenever possible (if only by making their terms of service as explicit as possible).

Providing security for user data actually isn't the hard part; there are plenty of examples of how this could be implemented. Mozy, the online backup service provider, addresses questions of privacy by allowing the customer to provide his own high-grade encryption key for his data. The backed-up data cannot be read by anyone else, Mozy included. If you leave the service, you take the key with you; the data becomes unreadable by default.

What's going to prove more difficult is crafting a forward-looking policy for privacy -- figuring out how much you can guarantee, or expect, in the cloud. The more explicit you can be, the more usage cases you can cover, and the more proactive you can be about everything you didn't think of the first time, the better.

### 3. Reliability

*Myth: Cloud computing is not reliable.*

File this one under "guilty until proven innocent." The cloud's been acquiring a leaden lining as of late -- a bad reputation for being questionably reliable. When T-Mobile's Sidekick [service crashed](#) recently and lost a ton of user data, criticism flew in all directions -- much of it aimed at clouds-in-the-abstract. The story has a happy ending -- everyone's data appears to have been recovered -- but does anyone want to sit through an experience like that again?

Because no two clouds are built the same way, people are quickly discovering "cloud" does not automatically equal "dependable," "redundant," or "safe." What's more, no customer cares why the data is missing -- and people remember your one failure far more than they do your five hundred days of flawless uptime.

The biggest frailty with clouds isn't uptime, but data protection. If nothing else, it's a sign that having only one kind of data protection -- either as an end user or as a provider of backup services -- isn't enough. The cloud needs to have multiple, concentric levels of data protection, from individual disk mirroring to robust file systems, to differential remote backups (whether to tape or another disk somewhere), all working in concert -- and to allow for data portability as another escape route, which is the perfect lead-in for the next point.

### 4. Migration

*Myth: Cloud computing is a one-way street.*

Sadly, there's more than a grain of truth to the complaint that once you get things into the cloud, it's a chore and a half to get them out again. This gripe mostly comes from the mouths of those who trust their data to a cloud-based service, only to find they have very limited options when it comes to migrating back out later -- a totally justified complaint. Anyone in the business of building a cloud-based system should [start thinking](#) now about how customer data is going to be migrated in -- and back out -- of the cloud.

Said outwards migration isn't just about data formats, but also the possibility that a customer may need to have his data exported to some physical carrier -- tapes, hard drives, what have you. It's open for debate as to whether the cost of such an outwards migration should be assumed by the customer, the provider, or split between them. But the debate itself should exist, as a way to make it clear to all that the customer's choice of cloud shouldn't ever be a dead end; it should serve its customers and not oppress them.

### 5. Big Deal

*Myth: Cloud computing is just virtual computing by another name. It's nothing we don't already have.*

This objection seems more out of simple ignorance of what clouds are meant to do. Clouds aren't just fancy virtual computing environments -- they make use of virtual computing along with many other technologies to accomplish things that aren't possible with individual pieces of iron.

Some of this confusion probably arises from the way certain cloud providers present themselves to the developer. Instances in [Amazon's EC2](#) are described as analogues to real-world systems with fixed amounts of memory, storage, and computing power. The reason for this is simple: The only way to

access the cloud that makes sense to developers right now is to present them with variations of hardware with which they are already familiar.

That doesn't mean the cloud will always appear to them like that. As the cloud evolves, and distributed / parallel computing along with it, so will new ways to describe the computing resources available through it -- descriptions that aren't limited to being echoes of real-world hardware. It's not impossible to envision a future cloud-computing platform where everything from the programming language to the development environment itself understands that the resulting code is going to be deployed in an elastic fashion.

Among the more vocal of the naysayers was Oracle's own Larry Ellison, most likely because for him the cloud represents a step away from Oracle's business model and toward computing models such as [SaaS](#) or IaaS (Information-as-a-Service). That hasn't stopped Oracle from creating its own "Application Grid" system, which allows Oracle databases to be deployed into the Amazon EC2 cloud. Clearly Oracle doesn't out-and-out hate the cloud; it just wants it on its own terms. Who doesn't?

## 6. Network and Storage Constraints

*Myth: Cloud computing is too hidebound by network or storage constraints to be useful.*

Well, yes and no. This depends entirely on the scale and scope of what you have in the cloud, how I/O bound it is, how scalable the processes involved are, and your chosen implementation for all of it.

I/O, into and out of the cloud, is still by and large, the main choke point, and will remain that way until we all have terabit-fiber-to-the-curb connections -- which at this rate will arrive shortly after [Godot](#) finally shows up. In the meantime, the smart thing to do is to build your cloud architecture -- its front *and* back ends -- in such a way that you only move the data that needs to be moved. Failing that, you can move around only the *work* that needs to be done on the data, which while more computationally difficult and programmatically challenging, is a bit more flexible than moving *all* the data.

One side of this issue that hasn't been discussed as much is how network speeds constrain the back end as well as the client-to-server connection. Bandwidth on the back end (and backbone) isn't as limited as it is to the client, but data densities are growing faster than network speeds -- we're talking petabytes of data that may need to be synchronized among multiple hosts.

What's needed to offset is not a bandwidth breakthrough, but smarter use of the existing networks via innovations such as [low-bandwidth file systems](#). The most important thing is to assume nobody else will do this for you: you have to make it happen, and set an example for others.

If nothing else, these issues should provoke more conversation on the subject of how to most efficiently use the network, and what sorts of clients are worth pairing with the cloud as a server / back end.

## 7. Scalability

*Myth: It's too difficult to make use of the cloud's scalability.*

This is a common complaint from the programmer's side: Cloud computing is hard to program for effectively. Sure, it's easy enough to get something up and running, but getting it to scale -- that's another story entirely. Ask the folks at Twitter, who started off their service as a Ruby on Rails application and are now incrementally rewriting the service in [Scala](#).

Much of this frustration comes down to the still-nascent state of parallel programming. Writing applications -- and by extension, algorithms -- that scale across multiple cores is difficult. The most commonplace

languages like PHP weren't designed to scale -- they were built to make it possible to deploy something quickly and easily. The aforementioned Scala is a language specifically designed for making an app scale (hence the name), and its close interoperability with Java means existing Java programmers have less work to do. That said, learning another language -- to say nothing of implementing it in a broad-scale setting -- is never a trivial measure, so this complaint is typically a reflection of the effort involved to get the work started on the right foot.

Scaling the most urgently needed operations is even tougher, since most of those very operations don't lend themselves to being distributed in this fashion. Consequently, it's easier to look at the cloud and feel the whole thing is a solution that just creates its own new set of problems.

Consider the Apache Software Foundation's [Hadoop](#) used by Yahoo, and Google's [MapReduce](#) technologies, used to access and aggregate results from large sets of data. The data stays distributed across your different nodes; it's the computation that's sent out to different nodes and then pulled back into a result.

Unfortunately, MapReduce only works on certain kinds of operations -- it's not an all-purpose solution, so the problem often has to be contorted into a shape that matches one of the existing solutions. Another solution-under-wraps is [Swarm](#), a framework for Web applications that allow computations rather than data to be scaled across multiple nodes. The concept and its execution are still in their infancy, but it deserves an ongoing look as a possible weapon in the arsenal.

The myths, the hype, the difficulties and the promise of the cloud all seem to come in the same package. Consequently, anyone involved in making the cloud into more than just a buzzword should stick his or her neck out as far possible (see James Urquhart's [Cloud Computing Bill of Rights](#) to fulfill the promise and dispel the mythology. Confront the very real issues that go hand-in-hand with cloud technology, and become part of the solution whenever possible.

If not you, who else will?