White Paper

10 Ways to **Think Like a Cloud**

Rules of the new on-demand architecture

by **Alistair Croll**

Introduction

Cloud computing represents a significant shift in IT strategy. Now that applications run on virtual infrastructure—independent of the underlying machinery—they are portable and therefore give third-party providers the opportunity to sell computing as a utility.

This on-demand model of computing is different from its predecessor in a variety of ways, many of which aren't obvious to the casual observer. To really understand the implications of cloud computing, we need to look at how it's being used by early adopters and at the design patterns and lessons they're learning.

In this report, we'll consider 10 fundamental shifts in IT mindset that IT professionals need to undergo in order to "think like a cloud." Some of these concepts will take years to find their way into mainstream organizations that still rely on mainframes and traditional bare-metal computing, but eventually they'll affect everyone. If you take these patterns to heart, you'll be far ahead of the average enterprise in your IT thinking, and better equipped to thrive in a utility computing world.

ADA—Patience. Business value does not result in quick adoption

Any big, fundamental change in thinking takes time to catch on, regardless of how promising it might be. For example, the advent of object-oriented programming. Between 1985 and 1995, NASA tried to roll out a programming language called ADA. On the surface, it looked wonderful:

- Code re-use increased by 300 percent
- The cost of systems dropped by 40 percent
- Bug counts fell 62 percent
- Development cycles were trimmed by 25 percent.

So it was a wild success, right?Unfortunately not. Less than 20 percent of the software produced by the organization was written in ADA, despite these advantages. Many developers simply couldn't comprehend the concepts of object-oriented programming. Others resisted, wanting to stay with procedural languages like FORTRAN for which they had libraries and tricks they'd collected over time.

Proponents of the language didn't help either: they promised too much, too soon, and avoided underlying problems such as the lack of environments and tools for experimentation. Object-oriented programming eventually won, but not before a whole generation of developers' skills became obsolete.

If this sounds familiar, it should. It's the path down which many enterprises are proceeding, and the results are likely to be the same: only some enterprise IT professionals will really make the switch to cloud computing. Others may pay it lip-service, then return to the fixed equipment and bare-metal mindset with which they feel more comfortable.

Private clouds make everybody angry

One way to define cloud computing is by the technologies that make it possible: virtualization, automation, auditing and accounting, a service-oriented architecture, self-service interfaces and the separation of computing from underlying hardware.

For many people, that definition means private clouds are simply "IT done properly." That's because the other definition of cloud computing is a business model: the cloud provider is a third-party organization, a utility like the power company, delivering when-you-need it compute cycles without upfront investment. To this group, "private clouds" are an oxymoron.

Many enterprise IT professionals see the term as an attempt by hardware manufacturers to sell them new equipment. Others are convinced they're already running clouds in-house simply because they've virtualized their servers. And third-party providers see private clouds as the fabrication of luddites, delaying their inevitable loss of control over their infrastructure.

In other words, private clouds make everyone angry. The debate gets in the way of reasoned discourse, so for the purpose of this document, we're going to ignore it. The 10 patterns outlined here apply whether you're using a third-party cloud, or whether you're running on a private cloud delivered as a service by your internal IT team.

Without further ado, then, here are 10 ways to think like a cloud.

1 Designing for failure

One of the fundamental tenets of the public cloud is that we design for failure. It's hard for many IT professionals to truly embrace this. We're used to calculating Mean Time Between Failure (MTBF) and buying redundant power supplies on high-availability hardware.

Public clouds don't work like that. If you walk through Google's data centers, you'll see thousands of cheap servers

info@cloud.ca
@cloud_dot_ca
420 Guy Montreal, QC H3J 1S6
1-888-796-8364

attached with Velcro™ to their racks, with batteries on each server rather than in a central location. Google knows servers are cheap, and breakable, and designs its processes accordingly.

What does this mean in practice? Consider, for example, that you were building a web browser but designing for failure. You'd assume that every time you clicked on a link, the server from which you'd received the previous page was gone. As a result, you'd do a DNS lookup to get a new IP address each time. This would be time-consuming, but you'd know that you were always getting a web server that was functional. On the other hand, if you assumed that the server was relatively stable, you'd cache the IP address and keep using it, and only look for another server when the first one stopped responding.

This kind of trade-off—cautious, assuming it's broken, versus fast, assuming it's working—is at the root of many architectural decisions. The difference is that cloud-thinking architects design the system cautiously, then look for opportunities to optimize; but traditional IT architects design the system to its specifications, then try to catch problems when they happen.

You want a Service Level Agreement (SLA) from your cloud provider that can help you determine the right cloud for your workload. This is the subject of much contention and controversy since cloud pundits believe the best SLA is an architecture that assumes failure but the business case is not always there. The cost of re-engineering your code or hiring talent that can design for failure may be difficult to justify. If you build an application that lives in three availability zones, Amazon will give you an astonishingly robust SLA for data. But the application you deploy atop it has to take advantage of that underlying redundancy and you increase the complexity of operating the application.

2 Machines are almost free

You need to treat machines as if they're free, and let innovators self-govern their usage. Many IT organizations are concerned-about enabling self-service for developers thanks to costly experiences with virtual machine (VM) sprawl and limited internal capacity. Enabling reporting on resource utilization per environment and per project moves the accountability from the operators to the integrators and the developers. This runs counter to decades of IT frugality, but it's essential. When you realize that machines are simply a convenient unit of measure—not a physical component—many cumbersome IT processes become faster and easier.

For example:

- Data analysis gets easy, because you can use a thousand machines for an hour, rather than one machine for a thousand hours.
- Fixing a machine that's misconfigured or infected doesn't happen any more. You simply roll back to a previous copy. In fact, you may as well give employees a fresh machine every day.
- You don't need maintenance windows. At one time, you needed to schedule downtime for an application. Now that machines are free, you can take a snapshot of a system, upgrade the snapshot, and switch over to the new system.[1]
- When you want to test an application, you can clone it exactly, rather than testing on a smaller, simulated system.
- Business intelligence and analytics are suddenly fast and affordable. Because machines are free, you can index data on many dimensions, slashing the time it takes to build and query data warehouses. When you're done, just turn them off.

There are many other consequences of free machines. Every time you perform a task, you need to ask yourself, "how would I do this if I really had more machines than I'd ever need?"

3 Big data is what clouds are for

We're awash in data sets. Companies hoard warehouses of data in-house, and public information is growing each year as we click, share, like, tweet, and swipe everything we do. In recent years, we've found new algorithms to speed up the processing and correlation of all this data; and cloud computing makes its analysis possible.

In many ways, big data gives clouds something to do. Tying together internal and public data sets can give businesses new insight—something web marketers quickly learned as they started to instrument their online storefronts. Now, sales and marketing are demanding fast, accurate updates on every aspect of their business.

Cloud computing has unlocked a huge demand for visibility and accountability with customers, suppliers, and nearly every part of a business. Where once a quarterly report was sufficient, now non-technical employees want real-time access to information.

It's essential to understand that cloud computing and data-driven businesses go hand-in-hand. It's not enough to be cloud-literate; you also need to be conversant in Big Data technologies like Hadoop, Cassandra, Bigtable, MongoDB, Riak, CouchDB, Ceph, and dozens of others.

1 Okay, it's not really this easy. But it's dramatically faster and easier than the previous approach.

info@cloud.ca
@cloud_dot_ca

420 Guy Montreal, QC H3J 1S6
1-888-796-8364

4 Disaster recovery is automatic

Microsoft's Jim Gray, after considerable research, declared that "compared to the cost of moving bytes around, everything else is free." A cloud platform needs to be able to launch, pause, and remove an application on demand. It also needs to be resilient to error, outage, or attack. Since the cloud doesn't know where or when a workload will be needed, most true clouds store multiple copies of data—including copies of virtual machines—in many places. This is known as sharding data.

Consider for a minute that Google's App Engine stores four copies of data; and Amazon allows companies to store information in several "availability zones" around the world. The underlying architecture of clouds makes them redundant, which translates to "free" disaster recovery.

In the past, having a DR strategy meant more than doubling the cost of infrastructure—two copies of everything, plus the technology to keep them in sync. With clouds, that changes. You only pay for that second copy when you need it (admittedly, often at a premium) and connecting it to your primary location is getting easier every day. Soon, even that will change: when the cloud is the primary location, it can be automatically configured in a redundant manner.

5 It'll be right eventually

Traditional design of data sources relies on them being accurate. As developers, we want to know that the content of a database is correct—when I check my bank balance, it had better be accurate—but that correctness comes at a high price.

When several users are reading and writing data to a database at the same time, conflicts can occur. Database Administrators go to great lengths to make sure this doesn't happen, because it leads to data corruption. The bluntest instrument at their disposal is simply locking down the database so only one person can write to it at a time, which ensures that no conflicts occur.

Imagine for a minute that Facebook worked this way. When you wanted to write something on your wall, Facebook would make all the other users wait; when you were done, they'd be free to read and comment. Obviously, this isn't how Facebook works—but it underscores a significant challenge in large-scale data systems.

When you think like a cloud, you look for "eventual consistency." Many large-scale, multi-user systems favor performance and parallelism at the expense of immediate accuracy. When you use Twitter, for example, you may not be seeing all of the messages from people you follow; but come back in a couple of minutes and the list will be complete. That's acceptable because of the nature of the data.

In fact, banks do this too. Your bank balance has a little disclaimer that transactions will be posted at the end of the next business day; funds you deposit are on hold for several days in order for checks to clear; and so on. Eventual consistency is all around us: sometimes almost is better.

Your bank balance has a little disclaimer that transactions will be posted at the end of the next business day; funds you deposit are on hold for several days in order for checks to clear; and so on. Eventual consistency is all around us: **sometimes almost is better.**

The accuracy that traditional architects crave comes at a high cost, limiting the ability of an application to scale elastically, undermining performance, and making it harder to keep several copies of information. Architects who think like clouds know that eventual consistency is the key to scaling and avoiding bottlenecks.

6 The new capacity equation

For decades, IT has dealt with three basic metrics:

- The **demand** for an application (users, requests per minute, or whatever else is consuming a resource.)
- The **capacity** of that application (number of servers, number of threads, megabits per second of bandwidth, or whatever is consumed.)
- The **performance** of that application (the time it takes to download something, the time the page takes to load, or some other measure of user experience.)

info@cloud.ca
@cloud_dot_ca
420 Guy Montreal, QC H3J 1S6
1-888-796-8364

This is a gross oversimplification, but it's the fundamental tradeoff that IT makes: capacity comes from how many resources we have at our disposal. Bad user experience hurts the business, frustrating customers and making workers unproductive. We can express this as:

$$\text{performance} = \left(\frac{\text{demand}}{\text{capacity}} \right)$$

Clouds change this equation forever, because with an elastic cloud platform, capacity is (virtually) unlimited, provided you're willing to pay for it.

$$\text{performance} = \left(\frac{\text{demand}}{\infty} \right)$$

In other words—and again, this is a massive oversimplification—you can have any performance you want, as long as you're willing to pay for it. Cloud application designers will be evaluated on metrics like cost per user-second: how much does it cost to service a user's request in a second.

This complicates discussions around performance SLAs significantly. The provider will deliver good performance, for a fee. Different cloud stacks will have different performance characteristics, and a company will save or waste money depending on how efficient its applications are.

7 Having your cake and eating it too

The two dominant models of cloud computing are Infrastructure as a Service (renting virtual machines by the hour) and Platform as a Service (pay-as-you go computing environments in which you ignore the underlying infrastructure entirely.) As Table 1 shows (see next page), each approach has benefits and drawbacks.

Most of the utilities we rely on today follow the PaaS model: a constrained offering in which the user abdicates much of their decision-making ability. When you use electricity, you don't get to say which generator produces it—nor would you want to. A cell phone bill doesn't tell you which towers were used for your calls. Because cloud computing is relatively new, however, we still cling to metaphors like the Virtual Machine because they give us convenient units of measure we understand.

The problem, however, is the simplicity promised to developers by PaaS does not cost-effectively scale in production. To tune production for efficiency and avoid lock-in, you need the control and breadth of options that IaaS can provide.

Cloud vendors are trying to balance the two. Amazon's Elastic Beanstalk is a pre-configured, automated, self-scaling platform built from several of its services. If you don't touch the underlying pieces, it scales like a PaaS; but if you start to customize it, you lose the automatic elasticity of the system. Microsoft's Azure straddles the IaaS/PaaS world with a blend of code execution and virtual machines.

When we really understand cloud offerings, we know about the tradeoffs between control and ease of use. Unfortunately, many IT professionals aren't thinking like clouds yet, believing instead that they can enjoy the turnkey elasticity of PaaS with the portability and architectural opinions that IaaS gives them.

8 DevOps is how we manage clouds

Cloud computing is the result of a long evolution in IT:

- Machines were inefficiently used, sitting idle for much of the time.
- Virtualization made it possible to more efficiently consume computing resources by letting a single physical machine run many virtual machines.
- Virtual machines were easier to start, stop, and copy—making them popular with IT operators.
- To handle the inevitable sprawl of easily-created machines, and to reduce human error, we built automation tools.
- With automation in place, it was easy to give the end users of those machines access to self-service consoles where they could request and control them, taking IT out of the loop.
- Once portable, self-service computing was accepted, third-party providers could offer competing services.

For decades, this transformation has been underway. It's had a number of other consequences, though. One is that the cycle time for releasing and tweaking an application has diminished dramatically, with some companies publishing code updates several times a day. As a result, the old notion of writing software and "throwing it over the wall" for someone else to operate has disappeared.

info@cloud.ca
@cloud_dot_ca
420 Guy Montreal, QC H3J 1S6
1-888-796-8364

Table 1 Comparing IaaS and PaaS

	IaaS	PaaS
Examples	Amazon EC2; Rackspace Cloud; cloud.ca; Terremark; Gogrid	Force.com, Google App Engine, Heroku, Engineyard
Choice of application and OS environment	Flexible: Any OS you want	Limited: Use only the languages it supports
What limits you?	Machines: Number and type of VMs you're using	Nothing (but governors kill long-running processes)
How you grow	Manually or scripted: By adding more machines, configuring load-balancers, defining auto-scaling	Automatically: It just happens
Storage options	Many options: file system, object, key-value, RDBMS, or build your own on a VM	Constrained: Use the storage API y ou're given (i.e. Bigtable)
Portability & lock-in	High portability: Portable if the machine image is relatively standard; some manage-ment tool standardization	Low portability: Rewrite the app and transform the data if you want to take it elsewhere
Billing & reporting	Billing and reporting on your infrastructure utilization	Billing and reporting on the application utilization, including function calls
Efficiency & optimization	Can be fined tuned and at scale this option is proven to be up to 10X more efficient	Optimized for the speed of development instead of the efficiency of resource utilization

info@cloud.ca
@cloud_dot_ca

420 Guy Montreal, QC H3J 1S6
1-888-796-8364

Instead, developers are now working closer with the operators of their code, putting hooks into the applications themselves that make the applications self-regulating. In the future, we will code the infrastructure as well as the application. One test of this is the tenth-story test: If you drop a random piece of infrastructure from a tenth-story window, will your application survive?

This is known as the DevOps movement. With roots in large-scale web operations and the Agile software development model, DevOps is finding its way into enterprises. Tools like Chef, Puppet, and Ansible let developers write pre-defined, self-healing clusters of functionality. These are then deployed, and use their control of the underlying infrastructure to grow, shrink, and adapt to changing operating conditions.

9 Clouds have data gravity

The late Jim Gray, who talked about the cost of moving data around, spent much of his life analyzing the costs of computing. He claimed that the future of computing was "hairy, smoking golf balls," by which he meant computers would be small, generate a lot of heat, and be bristling with connections.

This has important consequences for clouds. Data is the center of gravity of a cloud. Amazon launched its S3 service for large object storage—atop which many of its other offerings are built—roughly five months before it introduced the EC2 offering for which it is perhaps best known. Storage is the hardest thing to move; in many ways, a cloud is a central storage system, with a set of surrounding services that can do things to that data.

This is a critical concept when thinking like a cloud. It's not about where machines live—they're trivial, ephemeral, and relatively tiny. It's where the data live, and how you can access that data. In the long game, data is the basis for lock-in (just think about moving all your pictures from Flickr or Picasa, or your social graph from Facebook or LinkedIn, to get a sense of how strong the gravity really is.)

10 The moat doesn't matter anymore

Many security models rely on topography. On the left of the firewall is safety; on the right, dragons and horrors. We speak about security using places: the demilitarized zone, the local area. Cloud computing and portable workloads are quickly making this model less reliable, because perimeters are malleable and quick to change in an on-demand model.

Cloud security is all about workloads. Each application has to know who has permissions, what it can do, and what environments it will run on. It may even change its behavior depending on how it's being used, encrypting data in one place but not in another or restricting what information can be retrieved depending on circumstances.

Castle walls don't work anymore when the villagers are roaming the countryside. Thinking like a cloud means thinking about the applications, rather than having a false sense of security that the application is safe because of where it's located.

Conclusions

Cloud computing upends many of the assumptions we make about IT. We've looked at a few areas, from the cost of infrastructure to the negotiation of service levels to the ability to scale and survive outages. Thinking like a cloud will be critical for IT professionals who want to thrive in an increasingly on-demand world where utility computing is the norm.

Castle walls don't work anymore when the villagers are roaming the countryside. Thinking like a cloud means thinking about the applications, rather than having a false sense of security that the application is safe because of where it's located.

info@cloud.ca
@cloud_dot_ca

420 Guy Montreal, QC H3J 1S6
1-888-796-8364