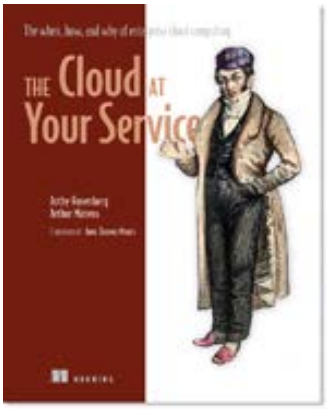
MANNING PUBLICATIONS



Cloud at Your Service
The when, how, and why of enterprise cloud computing
By Jothy Rosenberg and Arthur Mateos

Choosing a cloud vendor is a momentous decision. In this article, based on chapter 8 of Clout at Your Service, we look at the business considerations important in choosing a cloud provider by examining the key criteria for evaluating various vendor offerings.

Choosing a Cloud Vendor

Choosing a cloud vendor is a momentous decision. Let's explore the two most critical aspects of making that decision: the business considerations and the technical operational considerations.

Business considerations

When choosing a public cloud provider, you're often able to purchase services in an on-demand fashion. The advantage of this sort of arrangement is the ability to stop using it at any time. You can experiment with the services at little cost, and if the quality and reliability of the service leaves something to be desired, you can choose to go elsewhere. If you're running an application requiring significant investment that has high business criticality, or if you're considering an annual contract to lower costs, it probably makes sense to proceed cautiously. In this case, the selection and evaluation of a potential cloud provider bears some similarity to how you choose a traditionally outsourced service, such as web hosting or collocation.

The questions you should research and ask regarding the basic facts about a business providing cloud services should be along the lines of the questions you ask any outside vendor:

- Financial viability:
  - How long has the vendor been around?
  - Are they financially stable? Is it a public company or a well-financed privately held company?
  - Is it profitable?
- Operational viability
  - Does it have the requisite core assets, such as multiple data centers, and is it reasonably located?
  - Can it provide references of satisfied customers?
  - Does it have an operating history as a service provider?
- Contractual viability
  - Are its operations audited and in compliance with best practices for service based operations (SAS 70 Type II)?
  - What are its SLAs, and how are they enforced?

SAS 70 Type II Compliance

SAS 70 is an acronym for Statement of Auditing Standards, developed by the American Institute of Certified Public Accountants. Being SAS 70 Type II compliant means that a service organization has the appropriate infrastructure and controls in place to handle and process its customer's data in a satisfactory manner. SAS Type II certification is a costly process and represents a significant investment by a cloud provider. It involves a six-month data-collection process followed by a lengthy and comprehensive audit; it concludes with a detailed report produced by an independent accounting firm.

At a minimum, before entrusting a cloud provider with your business-critical applications, you should be comfortable that it is in a strong financial position and has a good operational track record and good operational assets, such as appropriate data-center facilities and network connectivity. When business issues have been satisfactorily addressed, the next step is to evaluate the technical considerations germane to cloud operations, which we'll talk about next.

Technical operational considerations

Many of the issues often cited as barriers to general cloud adoption end up being issues that you need to deal with when managing a deployed cloud application. In this section, we'll look at the main technical operational issues you'll encounter when you put your business in the cloud. Let's start with a discussion of availability and performance, which are traditionally the bread and butter of operational management. Next, we'll discuss elasticity and scale, operational security and compliance, and, finally, issues around interoperability and platform compatibility.

Availability and performance

Most people who work in the IT industry, or who are responsible for running IT operations, have an intuitive feel for what availability and performance mean. The simplest definitions are as follows:

- *Availability*—Whether an application performs its design function.
- *Performance*—How fast or slow the application is.

To proceed further, let's be more specific and provide precise definitions of these terms. Let's start with availability.

In the context of applications, whether delivered via the cloud or not, it's important to measure the availability as experienced by the intended user of the application—was the end user able to get what they came for? Most applications perform several business functions or services, and you can measure the availability of each. For any specific application, the starting point is to determine what the important business processes or services are and measure the availability associated with them. For example, consider an e-commerce site that sells books. A myriad of business services constitute such a site, but four are the primary ones important for generating revenue:

- *Search*—Find a book.
- *Browse*—Look at the description of a book.
- *Shopping cart*—Select a book for purchase.
- *Purchase*—Buy the book.

To measure the availability of the e-commerce site, you need to determine whether each of these business services works properly when you try to use it. You can attempt to measure these services independently and assign an availability measure to each. Alternatively, you could define a composite business service that looked at the total availability as being able to perform each of these business services. The product of the individual availability measures would be the result for each business service measured independently.

One way to define availability is as the number of successful uses of the application divided by the number of attempts to use the application:

$$Availability = \frac{(Total\ \#\ of\ successes)}{(Total\ \#\ of\ attempts)}$$

Although technically correct, this definition has some drawbacks. In practice, it's generally straightforward to measure the number of successes in using an application by looking, for example, at log files or other records of activity. The number of tries is less straightforward to measure. Whereas you can measure the number of errors that caused an availability issue when the application is more or less running properly, in the event that the system is down, you have no way of knowing how many users tried to use the application when it wasn't available.

The traditional way of defining availability looks at what percent of the time a given application or service is able to service users successfully over a given duration. We can sum up the definition of availability in the following equation:

$$Availability = \frac{(Total\ time\ the\ service\ is\ usable)}{(Total\ duration\ of\ measurement\ period)}$$

The availability of a system is often measured in *9s*, which describes the percent value of availability. *Three 9s* refers to 99.9% availability, and *five 9s* refers to 99.999% availability—truly a high bar for reliability.

"To the 9s": measures of application availability

Service-level agreements (SLAs) on availability are often measured in 9s. This describes the target percent of unplanned availability to be achieved, typically on a monthly or annual basis. Each 9 corresponds to a 10-fold decrease in the amount of downtime. For an important application, such as email or a CRM system, three 9s might be a reasonable target, whereas critical services such as public utilities would tend to target five 9s. The following table describes the amount of acceptable downtime per year for the corresponding level of availability:

# of 9s	SLA target	Maximum downtime per year
2	99%	3 days, 15 hours, and 40 minutes
3	99.9%	8 hours and 46 minutes
4	99.99%	52 minutes and 36 seconds
5	99.999%	5 minutes and 16 seconds
6	99.9999%	31.56 seconds

You should measure performance, like availability, as experienced by the end user, for each specific business process important for an application. You can measure the performance for each individual transaction but track aggregates over a given time interval to understand the overall performance of the system. Typically, you measure the average of the performance at a specific percentile level—such as 95th or 99th percentile over a period of time—to develop a baseline expectation for the delivered performance.

The availability and performance of applications in a cloud environment are due to three primary factors. The first factor involves the performance and reliability of the underlying hardware and software offered by the cloud vendor. The robustness of the infrastructure depends on how redundant and well provisioned the cloud vendor's data centers are. Cloud infrastructures are built with huge numbers of commodity servers and are designed with the expectation that the individual components in the system might fail. If a particular server running several instances in a cloud goes down, it doesn't impact the overall advertised availability of the cloud. But if the particular instance that goes down is the one supporting your application, the failure is indeed a problem for you. The best way to deal with this sort of occurrence is to plan for it. You can adopt two possible strategies:

- *Design with the possibility of failure in mind.* This strategy is included in the inherent design of the application and uses techniques such as horizontal scalability and balancing between multiple instances.
- *Plan to fail fast but recover quickly.* Make sure you have a good handle on detecting the occurrence of a failure and can react quickly to bring up an alternate instance to take over immediately.

The second factor relates specifically to the application's design and robustness as well as its quality. This is the realm of the application developer and is fundamentally no different from traditional application development except for the ability to design applications in the context of the resources available in a cloud environment. In the early days of computing, when memory and computing capacity were expensive, it was necessary to build applications that were stingy in their consumption of these precious resources. As CPU and RAM became less expensive, this requirement could be relaxed. In the same way, with the advent of the cloud, the ability to shard and horizontally scale with less regard to cost means you can trade off stringent quality control against the ability to dynamically distribute the load. But don't take this idea to imply the extreme, where the code is so unreliable that it crashes repeatedly.

Rather, in a cloud environment you have an additional degree of flexibility and freedom as tradeoffs between code quality and dynamic scaling, to reach a desired level of robustness.

Another factor that comes into play is somewhat different from traditional application development. It relates to the fact that, by definition, the application's end users are necessarily connecting to the infrastructure via the public internet. The network connectivity between the cloud infrastructure and the client used to access the application affects the cloud application's performance and availability for end users.

The availability of the application can suffer for a variety of reasons. The first involves the loss of connectivity between the cloud provider's data center and the internet—or, equivalently, the loss of connectivity between the client side and the internet. The best way you can deal with the former is to choose a cloud provider with data centers that have multiple connections to the internet backbone . In the event of an interruption to one, you'll still have connectivity. Similarly, if end users are accessing the application from within your corporate network, redundancy of connection with multiple different network access providers can improve application availability. In the case of applications accessed by those outside the corporate network, such as consumers of an e-commerce application, providing such redundancy isn't possible.

Don't forget, for a cloud-based application, availability is subject to the macro conditions of the internet at large. A recent error at Google related to DNS for its free Google Analytics offering caused general slowdowns and unavailability for several hours on the internet. For a cloud application serving an intranet audience, such an availability hit would be avoidable for a traditional deployment. For an internet facing application, you probably would have a good case for getting a free pass on the performance and availability hits taken for an event such as this, because "everyone else in the world" is also down.

The last point to mention with regard to network effects relates to the geographic dependence of performance on cloud applications. In a study on cloud performance, Gomez measured significant variations in the response time experienced by end users in different geographies. Figure 1 shows the performance of a simple web application deployed at a sample cloud provider, as experienced by end users from 18 different geographies.

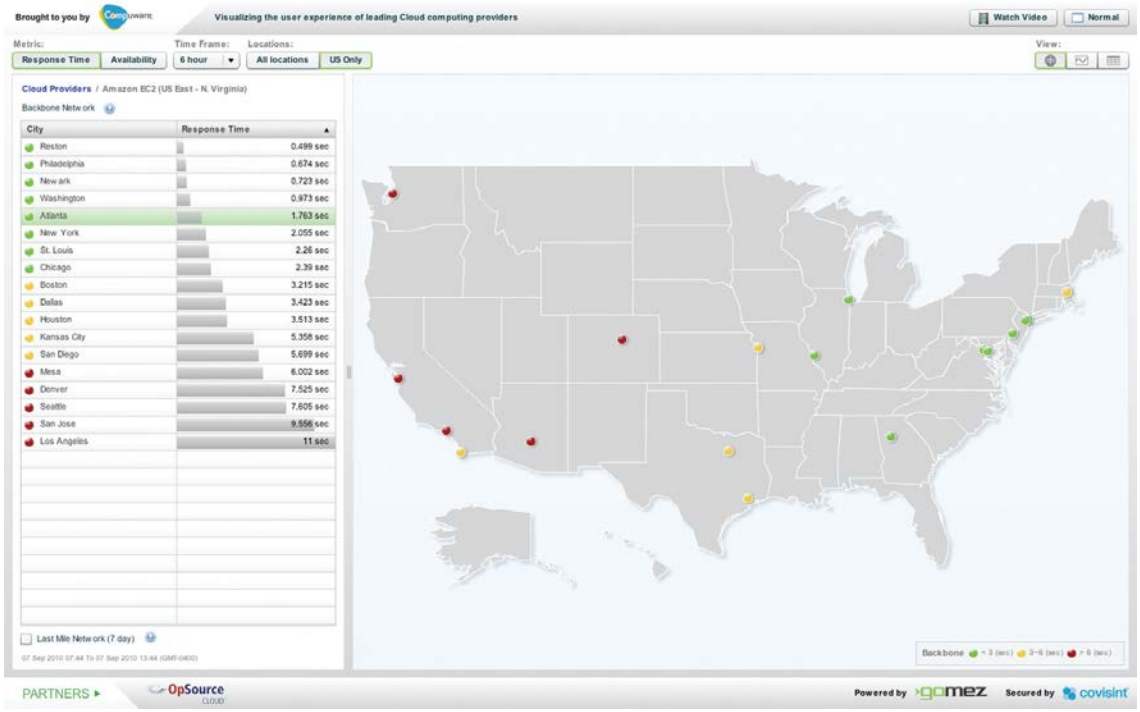


Figure 1 A comparison of the average time it takes to access the same web page for an application running in a sample cloud provider from 18 different cities across the U.S., as measured by Gomez

If the primary users of your application are in Virginia or the mid-Atlantic, for example, Amazon in VA may be an ideal choice. For an internet facing application, you can expect a wide variation in the performance experienced by end users from their points of access around the world.

To improve performance, you could look to deploy the application in multiple cloud data centers. Another approach would be to use content delivery network (CDN) technologies such as those provided by Akamai and Limelight.

Elasticity and scale

In the architecture and design of traditional applications, an important consideration relates to the scale at which it must run. When deploying the application, you provision the appropriate amount of hardware and system capacity to meet expected demand. In a cloud world, one of the primary assumptions is that an infinite resource pool is available. As long as the cloud application is appropriately designed using sharding and other appropriate horizontal-scaling techniques, you can reach a certain scale or capacity by bringing online more instances as the situation dictates. The problem then comes down to being able to bring resources online quickly enough for a given situation.

The term that describes the attribute of a cloud system to respond with rapidity or velocity in response to spikes in demand is *elasticity*. A typical application that may stretch the elasticity of a cloud is a media website that gets spiky load due to a news event, such as a natural disaster. You can drive peak load s to a website in less than a minute. The elasticity of a cloud can vary based on the underlying cloud technology as well as intrinsic boot times of OS images.

The elasticity of Amazon EC2 and a cloud built using the open-source Eucalyptus platform have been measured by looking at the time it took to launch one and then eight virtual images by the team that created Eucalyptus. In figure 2, you can see that on average, it took less than 20 seconds to launch 1 new instance on EC2 and less than 25 seconds for 8 new instances. For most applications, this level of elasticity is sufficient to scale up in times of peak demand, provided you're able to detect the need for the extra computing power quickly enough.

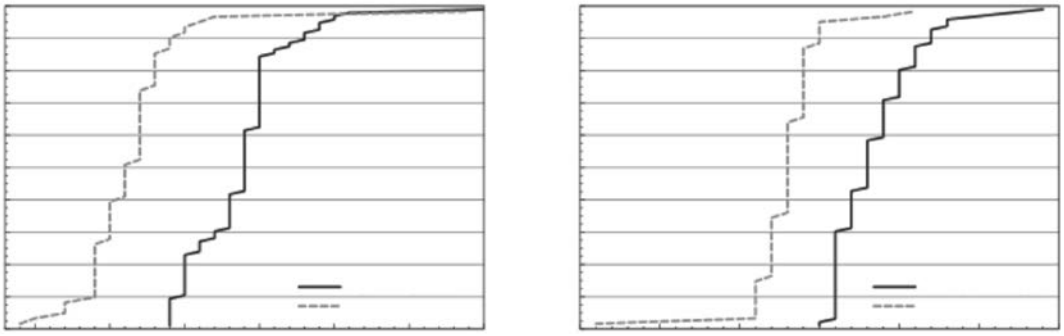


Figure 2 Elasticity of EC2 and a private cloud deployment based on Eucalyptus. The left graph shows the cumulative probability as a function of time for an instance to be available after the request has been made to start it. The right graph shows the cumulative probabilities of the time it takes before eight instances are available. Source: Daniel Nurmi et al, "Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems," UCSB Computer Science Technical Report Number 2008–10, August 2008.

In addition to the time taken to start a base image, you also need to factor in the time required for the application to initialize and be ready for work. The general load on the cloud system can affect the time required to start new instances. It can be subject to variability due to the volume of traffic requests within the cloud for various administrative functions.

Operational security and compliance

Security- and compliance-related issues tend to top the list in any survey on the primary inhibitors to widespread adoption of the cloud for business critical applications.

The two main compliance standards often cited as potentially problematic for a cloud style deployment include the Payment Card Industry Data Security Standards (PCI DSS) for e-commerce related businesses and the Health Insurance Portability and Accountability Act (HIPAA) that governs privacy within the healthcare industry. Both are very detailed specifications on the proper way to handle data in a safe, secure, and auditable manner. A cloud provider, assuming it has deployed the appropriate base security architecture, maintains sufficient physical infrastructure, and has put in place the proper processes and procedures, can provide the necessary base for a generically secure infrastructure. SAS 70 Type II certification, as we discussed earlier, is a measure of whether a cloud provider has all of these necessary ingredients in place, and can be seen as a prerequisite for PCI DSS or HIPAA.

In addition, the cloud application must be deployed with the appropriate security standards, including correctly configured firewalls, appropriate encryption of sensitive data, appropriate practices for key management, and so on. None of these practices are much different than those required for an internally hosted compliant application. The primary difference, and regulatory hurdles that often can't be overcome, are strict requirements for such things as direct inspection of hardware infrastructure that may not be practical within the context of a cloud environment.

Interoperability and platform compatibility

Interoperability and platform *compatibility* refer to how easy or difficult it is to switch between different providers. These factors are often cited as primary blockers to cloud adoption. But it's probably arguable whether these are major practical issues worth worrying about. In the simplest public cloud adoption scenario, where you choose one provider, are happy with it, and stick to it, interoperability and compatibility never come into play. Also, when choosing something more complex than a basic IaaS provider (such as PaaS or SaaS), it's your choice. You make the implicit choice of trading richer functionality for portability.

Several potential levels of interoperability exist. The simplest forms involve provisioning and management interoperability. More complex forms may include the ability to copy instances from one cloud provider to another or even migrate between them dynamically in real time. You can argue that the only kinds of interoperability that really matter are command and control. Next most important are a common framework for monitoring.

Interoperability of VMs and also the ability to move them dynamically sounds cool but is probably impractical in a public cloud-type deployment because it would take too much time to move stuff around. The good news is that the current state of interoperability isn't bad.

Let's consider a scenario where you need to switch from one provider to another. This case requires, at some level, provisioning compatibility, but not much more. For a one-time move, a little friction and some manual processes are probably acceptable.

Let's look at another scenario in which you need multiple simultaneous providers. This is rare, but it can be driven by two potential business requirements. The first relates to applications that require more capacity for short bursts than is available from any given provider. Here, at a minimum, once again, you need provisioning.

The second potential use case, which is even rarer today, is the ability to arbitrage the IaaS providers. Again, this can only work in an IaaS world because for services to be tradable, they (by definition) need to be easy to switch between, and that must happen at minimum cost. You can consider arbitrage between like services on several factors. Some of these may include cost or performance. The final case is a hybrid private-public cloud scenario. In these scenarios, you'll see a similar requirement regarding interoperability of provisioning. As of today, two primary standards are used for provisioning: Amazon-style APIs and VMware.

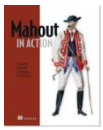
For a public cloud provider, interoperability is a two-edged sword. Interoperability with your competitors means it's easy for their customers to leave them and come to you. But the flip side is that it can work in reverse. In a world of perfect interoperability and functional equivalence between vendors, only two factors remain for a cloud vendor to compete on: the price of the offering and the quality and reliability of the service they provide. SLAs are documents that cloud providers publish, detailing their level of service. They also serve as contractual commitments to delivery of that service level. In the next section, we'll look in detail at the SLAs provided by several major cloud vendors.

Summary

A cloud application is different from a conventionally deployed application in the level of control that's ceded to a third party. In a traditional deployment of an application with a colocation provider, you cede control over physical security, power, and network connectivity to a third party, but you retain control over the server hardware, operating systems, and application software. In a cloud deployment, you extend the ceding of control to include the server hardware and the operating system (through the virtualized environment). In comparison, this should be a good thing, because there's less to worry about as long as the provider's systems operate at a sufficient level of reliability.

Here are some other Manning titles you might be interested in:

[The Joy of Clojure](#)
Michael Fogus and Chris Houser



[Mahout in Action](#)
Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman

[Hadoop in Action](#)
Chuck Lam

Last updated: October 19, 2011