CHAPTER 3

Cloud Computing Worst Practices

When you come to a fork in the road, take it.

—Yogi Berra, Hall of Fame baseball player

The U.S. Army invested $2.7 billion in a cutting-edge cloud-based solution with the goal of communicating real-time information from various sources to assist in battlefield operations in Iraq and Afghanistan. The system failed to deliver and actually hindered operations instead of helping. As one person put it, "Almost any commercial software product out there would be better." Cloud computing can create huge competitive advantages if applications and services are correctly architected to satisfy the business requirements. This chapter discusses the nine common mistakes that companies make in the cloud. At the end of the discussion for each common mistake, recommendations for avoiding these mistakes are given.

Avoiding Failure When Moving to the Cloud

Historically, many companies fail when it comes to implementing new and transformational technologies. There are many causes of failure. Sometimes companies fail because they don't fully understand or embrace new technologies. Sometimes they rush into development mode and forgo the necessary architecture and design steps. Sometimes they have unrealistic expectations like too-aggressive due dates, too large of a scope, not the right people, and many other reasons. The next few sections focus on the top reasons why companies moving to the cloud might fail.

Migrating Applications to the Cloud

A common misperception about cloud computing is the notion that migrating existing applications to the cloud is a simple solution that drives down costs. The reality is usually the complete opposite. In fact, very few applications are good

candidates to move to the cloud in their current architecture. Legacy software is architected to run within the corporate firewalls of the company. If the software was built several years ago, there is a high probability that the software is highly dependent on the physical hardware it runs on and possibly even on the technology stack it was written on. This is often referred to as being a *tightly coupled* architecture, because the software cannot function properly if it is separated from its physical environment. Cloud computing architectures require a *loosely coupled* architecture. As mentioned in Chapter 2, elasticity is a key component of cloud computing. For software to be truly elastic, meaning it is able to be scaled up and down as needed, it must be independent of its physical environment.

Most legacy architectures were never intended to be built in a manner where the system automatically scales as the number of transactions increases. Traditional scaling techniques often rely solely on vertical scaling. Vertical scaling is accomplished by increasing the existing hardware either by adding more CPUs, memory, or disk space to the existing infrastructure or by replacing the existing infrastructure with bigger and more powerful hardware. Vertical scaling is known as *scaling up*. Vertical scaling typically does not require software changes beyond changing configurations to allow the software to leverage the new infrastructure as long as the same type of infrastructure is used.

With this type of scaling strategy, architects often forgo designing their software to be independent of the infrastructure. For example, if an application is built on an IBM iSeries computer, the software is typically written in ways to take full advantage of the proprietary infrastructure, thus becoming tightly coupled to the hardware. Migrating an application like that would take a major reengineering to remove the dependencies from the iSeries so that the application can become elastic in the cloud. For a system to be elastic it must be able to handle unanticipated and sudden spikes in workloads.

If elasticity is not the reason for moving the application to the cloud and the company just does not want to manage and maintain the infrastructure anymore, then what the company most likely needs is a hosting solution. Hosting is not the same thing as cloud computing. Hosting does not provide for the five characteristics of cloud computing: broad network access, elasticity, measured service, on-demand self-service, and resource pooling. Hosting is simply renting or buying infrastructure and floor space at a hosting provider's facility. Think of migrating to a hosting facility as forklifting an application from site A to site B. Migrating an application to the cloud is much more involved than that.

Scaling in the cloud can be done with vertical scaling but is mostly accomplished through automated horizontal scaling. Horizontal scaling is accomplished by adding additional infrastructure that runs in conjunction with the existing infrastructure. This is known as *scaling out*.

Horizontal scaling is often done at multiple layers of the architecture. Some common horizontal scaling methods are to add nodes by server farm type (Figure 3.1), by customer type, and by application domain type (Figure 3.2).

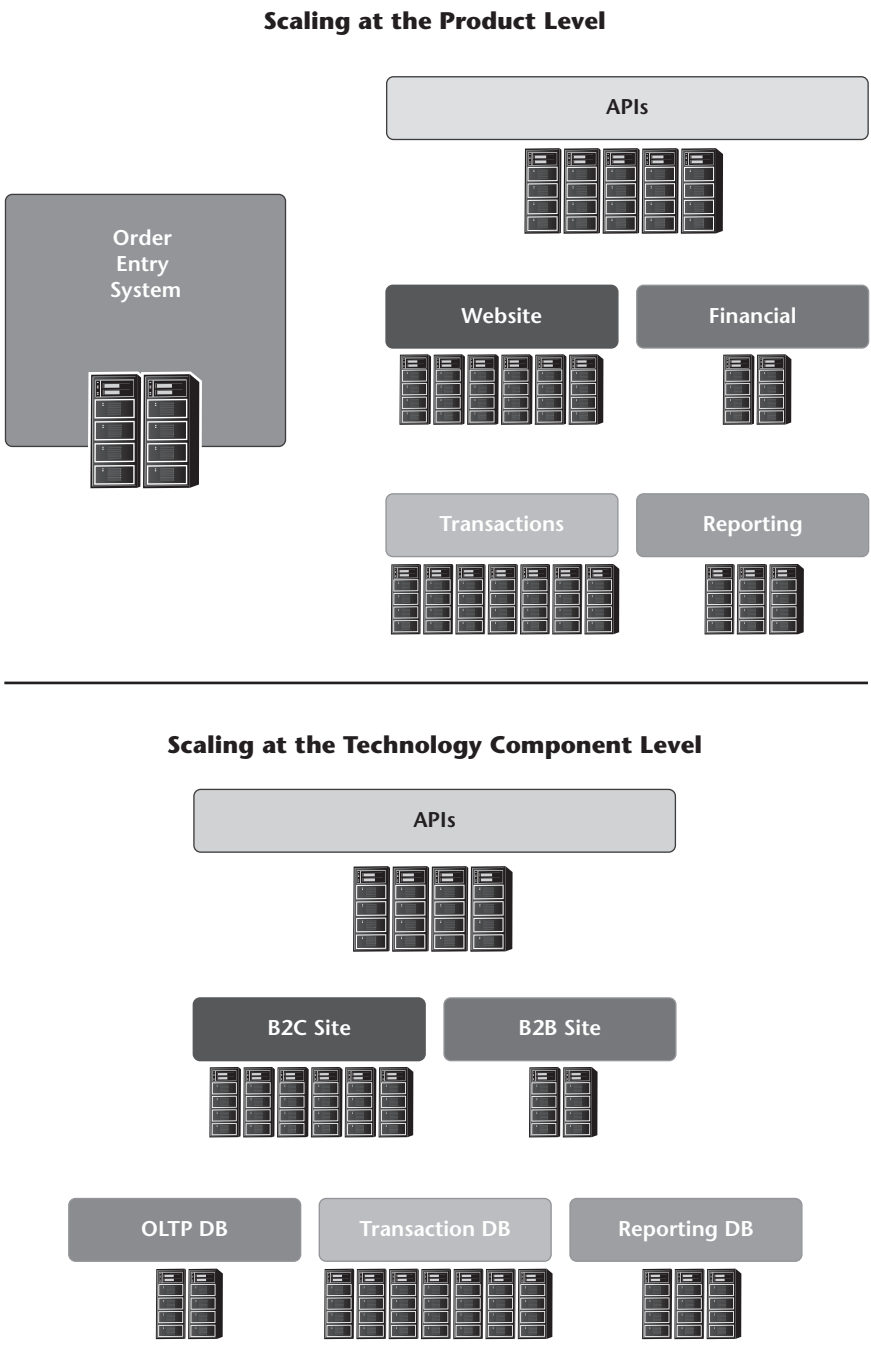

Figure 3.1 Scale by Server Farm Type

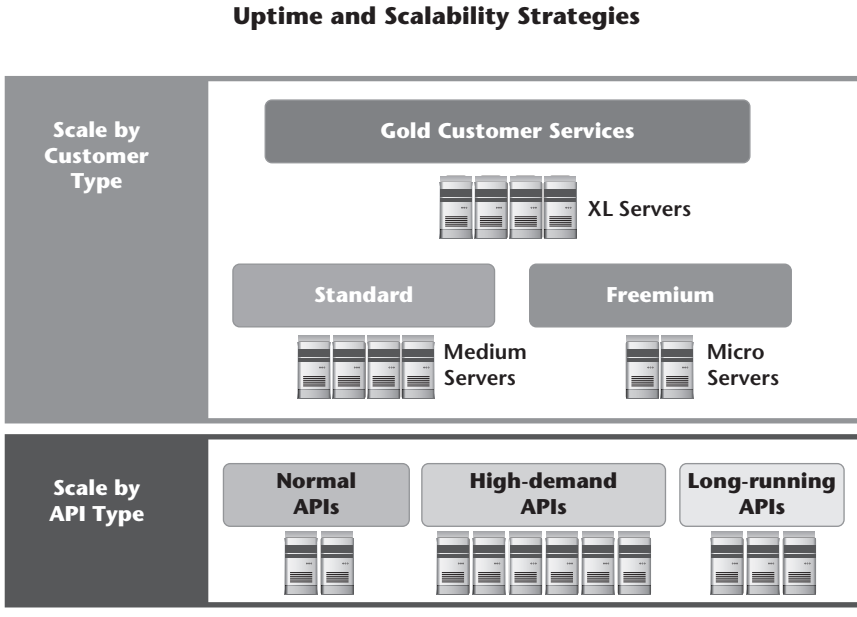

Figure 3.2 Scale by Customer Type

We will discuss these design patterns in detail in Chapter 4 ("It Starts with Architecture").

Another challenge for legacy applications is whether the system design is stateful or stateless. Cloud services are stateless. A *stateless* service is a service that is unaware of any information from previous requests or responses and is only aware of information during the duration of the time that the service is processing a given request. Stateless services store the application state on the client, not the server, and therefore do not have a dependency on the infrastructure. For example, if a loan service receives a request to evaluate the credit rating of a customer applying for a loan, the service has no record of any information about the customer until it receives an incoming message (usually as an XML or JSON document). After it processes the document, determines the credit score, and responds back to the requestor, it does not store any of the information within the session and no longer knows anything about the customer.

We will discuss application state in more detail in Chapter 6 ("The Key to the Cloud") to explain why stateless architectures are better suited for the cloud than stateful architectures. The work required to change the underlying architecture from maintaining state to being stateless is often not feasible and a total replacement of the application is a more realistic

approach. Companies that migrate legacy stateful applications to the cloud will likely be disappointed with the end result if they expect to reap all of the benefits of cloud computing.

In summary, unless an on-premises application was architected as a collection of loosely coupled services to be accessed by other technology and infrastructure-agnostic services, then migrating to the cloud either will take a major reengineering effort, will not reap many of the benefits of the cloud, or might be completely unfeasible.

Recommendation: First, make sure the architects have a keen understanding of the differences between stateless and stateful design patterns. Understand if an application is a good candidate for migrating to the cloud or if hosting or a rewrite is a better option.

Misguided Expectations

Another common mistake is that companies often take on cloud computing initiatives with inflated expectations. There are many impressive success stories in recent years as companies large and small have embraced the cloud. On April 9, 2012, Facebook CEO Mark Zuckerberg posted on Facebook that his company had just purchased start-up Instagram for its innovative mobile photo-sharing platform for $1 billion. At the time it was purchased, the entire company consisted of only 13 people and had over 100 servers running in the Amazon cloud supporting over 30 million users. In its first year of existence with only 3 engineers, the platform went from 0 to 14 million users with over 150 million photos and several terabytes of traffic.

Another star of the cloud is Netflix. By late 2012, Netflix laid claim to the fact that almost 29 percent of all Internet traffic flowed through its streaming video platform to televisions, computers, and devices to consumers in North America, bypassing YouTube and even HTTP. The team at Netflix is a poster child for how to run an innovative culture that pushes the envelope and uses the cloud as a competitive advantage.

Both Instagram and Netflix are outliers. Instagram had the luxury of starting from scratch and architecting for the cloud out of the gate. Netflix made a business decision to put all of its chips in the cloud and hired and trained an incredible engineering team who continue to be pioneers in cloud computing. Neither company represents a normal Fortune 500 company or an established small or medium-size business looking to leverage cloud services. Many organizations have a very complex enterprise consisting of numerous vendor and proprietary solutions ranging from mainframe technologies, to midsize computers, n-tier architectures, and every other architectural pattern that was popular at one time or another. Starting with a blank slate or getting

an initiative from the CEO to re-platform the entire product line with a new cloud-based architecture is not the norm for most companies. Often management, or even the architects, are so starstruck by the success of companies like Netflix and Instagram that they expect similar results, an outcome they most likely can't achieve even if they do a good job architecting. Setting an expectation for the outcomes for a cloud computing initiative should be based on the business case in support of the initiative, not what other companies have achieved. Cloud computing is only part of the success these companies had. A bigger part of their success was their vision, the talent within their team, and their ability to execute.

One of the biggest misguided perceptions of cloud computing is that cloud initiatives will greatly reduce the cost of doing business. That may be true for some initiatives but not all of them; after all, cost is not the only reason to leverage the cloud. Even if a company has a good business case for reducing costs in the cloud, it takes more than cloud computing to achieve the cost reduction. Companies need to design with cost in mind. The cloud can be cost-effective if the architecture effectively optimizes its usage of cloud services. In order to optimize costs, the architecture must monitor the usage of cloud services and track costs.

Cloud services are a metered service where the cost is calculated on a pay-as-you-go model much like utilities such as electricity and water are in our homes. In legacy on-premises data centers, purchased infrastructure becomes a sunk cost, which depreciates on the books over the course of the several years. To plan for surges in traffic and growth over time, a company must overbuy so that there is excess capacity and usually redundancy for fail over at another location. These monies are paid in advance and much of this infrastructure may sit idle most of the time. A correctly architected cloud-based equivalent solution would allow a system to scale up and down as needed to align costs with revenues, the infamous pay-as-you-go-model. The key word there is *correctly* architected. If there are flaws in the architecture and the cloud services consumed are not appropriately turned off when not needed, the cloud can become a very expensive proposition. On the flipside, if the architecture does not scale up sufficiently or does not design for failure, the end result can be outages and poor performance, resulting in lost customers and revenue.

Not every problem is one that needs to be solved by cloud computing. For example, I once had a client call me and ask what cloud provider he should move his server to. When I asked him what problem he was trying to solve he said he had a code repository on a single server and wished to migrate it to the cloud. The asset cost him about $3,000 in hardware and software plus annual maintenance (if he was even paying for it). If he moved the server to the cloud at a 50-cents-an-hour rate, he would pay that rate every hour from this point forward. At 50 cents an hour, the server would cost $12 a day and for

the year it would cost $4,380. To make matters worse, he would continue to pay that each year where his on-premises sunk cost of $3,000 was a one-time cost. Since his application did not need to scale up and down, was already paid for, and was not causing any issues, there was no business case for it to move to the cloud and it certainly was not going to be cheaper in the cloud. I proposed two solutions to him. Solution 1: Do nothing. Solution 2: Replace the on-premises solution with a Software as a Service (SaaS) equivalent solution. There are many SaaS-based code repositories that charge a minimal amount of money per month and do not require any hardware or software to manage.

Recommendation: Set realistic expectations. Break cloud computing initiatives into smaller deliverables in order to deliver business value sooner and allow the team to learn along the way. Do not try the big-bang approach where the team goes away for several months or a year with the hope of delivering a large number of new cloud services. Understand the pros and cons of each cloud service model. Design to optimize, monitor, and audit cloud service consumption and implement a governance process to enforce proper consumption patterns. Each monthly bill from cloud service providers should be closely monitored to ensure costs are within expectations.

Misinformed about Cloud Security

Security is another area where expectations are often off base and there are two camps. The first camp believes in the myths that cloud computing is catastrophically insecure and data cannot be placed in a public cloud for any reason. People in this camp refuse to consider public clouds and often resort to building their own private clouds. If security and infrastructure are not a core competency, then building private clouds based out of fear might not be the best use of company money and time. The second camp is the group that believes that security is taken care of for them by the cloud vendors. This camp then proceeds to deploy software and services with gaping security holes into the cloud, where cyber-criminals welcome them in with open arms.

The cloud is not only an enabler for enterprises but it is a great enabler for cyber-criminals as well for two reasons. First, cloud computing is still very immature and lacking standards at this time. There are not a lot of engineers with years of hands-on experience securing applications in the cloud. The end result is that many cloud services are being deployed by today's corporations without the necessary security and controls and are very vulnerable to all kinds of attacks and breaches. The second reason why the cloud is an enabler for cyber-criminals is that the cloud vendors are a huge target because they house compute resources and data for a large number of companies. The cloud providers typically provide high levels of perimeter security, but

it is up to the companies deploying their services to build the appropriate level of application security. For example, an Infrastructure as a Service (IaaS) cloud provider like Amazon Web Services (AWS) has world-class secure data centers, white papers on how to build highly secure services on its platform, and provides a suite of application programming interfaces (APIs), making it easier to design for security. However, it is up to the architects building the software on AWS to encrypt the data, manage the keys, implement good password policies, and so forth.

The truth about security and the cloud is quite simple. With the proper security architecture, the public cloud *can be* more secure than most on-premises data centers. Unfortunately, very few corporations know enough about the security requirements in the cloud necessary to architect for it, and many others do not have skill sets internally to build the appropriate level of security. A recent report on security breaches from Forrester declares that 75 percent of all security breaches are inside jobs. Of the inside jobs 63 percent were not caused by intent. Examples of causes are lost, stolen, or misplaced assets such as thumb drives, disks, documents, devices, laptops, and so forth. Architects should sort through all the myths and hype about cloud security and research factual information.

Security is not something one buys; security must be planned and designed into software. Many of the security best practices that have been applied in data centers for years should be applied within the cloud, as well. We will discuss how to design for security in Chapter 9 ("Security Design in the Cloud"). What is important to note here is that deploying or leveraging cloud services requires additional steps to provide the appropriate level of security necessary to comply with regulatory constraints and to pass audits such as HIPAA, SOC-2, PCI DSS, and others. With the proper investment in infrastructure and application security, cloud services can be more secure than on-premises solutions, especially in organizations whose competency is not security.

Most non-Fortune 500 companies simply do not have the staff, expertise, and budget to build and maintain the appropriate levels of security to keep up with the increasing number of threats. For most cloud providers, security is a core competency and they invest heavily in talent and budget dollars to produce best-of-breed security solutions. Leveraging security as a service from a cloud computing vendor that excels at security can allow companies to obtain higher levels of security than they have achieved in the past in their own data centers. The trick is knowing what the security risks are and addressing those risks with a combination of technology, process, and governance.

Recommendation: Start by making sure the architects, the product team, and the security professionals have a broad understanding of cloud security, regulatory controls, and auditing requirements, which we will cover in detail in Chapter 9. If necessary, bring in an independent third party to perform

an assessment and to perform audits prior to deployment and ongoing after deployment. Design for security up front. Do not try to plug the holes later.

Selecting a Favorite Vendor, Not an Appropriate Vendor

A common mistake many companies make is they don't thoroughly evaluate the cloud vendors and simply select vendors that they are familiar with. For an obvious example of this worst practice go to any .NET shop and the odds that it has selected Microsoft Azure are incredibly high. That is not to say that Azure is not a good technology, but it may not be the right tool for the job. In Chapter 5 ("Choosing the Right Cloud Service Model") we will discuss the business use cases that make sense for each service model. Azure is a Platform as a Service (PaaS). The fact that a company writes .NET code should not override the technical requirements that determine whether the best cloud service model is SaaS, PaaS, or IaaS. In fact, there are several PaaS solutions that support .NET development.

The same applies to Google App Engine. Google's PaaS supports Python development. Instagram is a Python shop. Had it defaulted to Google's PaaS due to its choice of Python as its stack, it might not have been able to achieve the scalability that it achieved on AWS. This by no means is a knock on Google or a declaration that AWS is any better than Google. Simply put, for scaling requirements like Instagram's, an IaaS provider is a better choice than a PaaS. PaaS providers have thresholds that they enforce within the layers of their architecture to ensure that one customer does not consume so many resources that it impacts the overall platform, resulting in performance degradation for other customers. With IaaS, there are fewer limitations, and much higher levels of scalability can be achieved with the proper architecture. We will revisit this use case in Chapter 5. Architects must not let their loyalty to their favorite vendor get in the way of making the best possible business decision. A hammer may be the favorite tool of a home builder, but when he needs to turn screws he should use a screwdriver.

Recommendation: Understand the differences between the three cloud service models: SaaS, PaaS, and IaaS. Know what business cases are best suited for each service model. Don't choose cloud vendors based solely on the software stack that the developers use or based on the vendor that the company has been buying hardware from for years.

Outages and Out-of-Business Scenarios

When leveraging cloud services there should be an expectation that everything can and will fail. It doesn't matter which cloud service model a company

is relying on; everything fails at some point. It is no different from the power that runs our houses. All houses will encounter a power outage at some point, whether it is a brief flicker or an outage that lasts for hours or days. The same applies to cloud computing. A best practice is to design for failure. In Chapter 13 ("Disaster Recovery Planning") we will discuss best practices for dealing with failures for each service model. Here are a couple of examples where companies did not plan for failure.

PaaS provider Coghead was an early market mover in the database-as-a-service space. Database-as-a-service products create huge speed-to-market advantages through the automation of database administration tasks and by providing autoscaling capabilities. Customers leverage these services that allow them to focus more on their applications and less on database management, thus providing greater agility. Customers that leverage this type of service must understand that they are signing up for vendor lock-in and giving up some levels of control. Lock-in is nothing new in the world of databases. Companies have been buying Oracle, SQL Server, and DB2 licenses for decades and are used to being locked into a vendor. The difference with on-premises computing is that if the vendor goes away, the company still can use the software for as long as it wants. In the cloud, when the vendor goes away, the service often goes with it. In 2009, SAP purchased Coghead and gave customers eight weeks to get rid of their dependencies on Coghead because it was shutting down the service. Many customers never considered this scenario and were sent scrambling for the next eight weeks or more to recover from the shutdown of their database technology. A best practice for companies leveraging SaaS or PaaS database technologies is to ensure that they have access to the data outside of the service provider, whether it is snapshots of database backups, a daily extract, or some method of storing recoverable data independent of the service and the provider.

By now, everyone has seen the issues created by outages from major IaaS and PaaS providers like AWS, Rackspace, Microsoft, and Google. Even companies like Netflix, which has built a service called the Chaos Monkey, whose job is to break things in production to test real-time recovery of the overall platform, is not immune to outages.

When a provider like AWS has a service outage within one of its availability zones, a large number of its customers' websites and services go offline until AWS resolves the issue. Most of these customer outages could have been easily avoided had the customer anticipated and designed for failure. Many customers only deploy to a single zone and have no plan for recovery when that zone is impacted. A provider like AWS provides a service level agreement (SLA) of 99.95 percent per zone. An SLA of 99.95 percent equates to 20 minutes and 9 seconds of downtime a month or roughly 4 hours a year. In 2011, a study by Emerson Network Power reported the impact of downtime on the average business equated to a loss of $5,000 a minute or $300,000 an hour.

Apply that number to the 4 hours of downtime a year predicted by the AWS SLA and the average business is looking at $1.2 million of lost business a year! What company in its right mind would not build redundancy across the availability zones knowing the potential losses of a single zone failure? AWS provides multiple zones within a region and multiple regions across the globe. Customers have the opportunity to achieve SLAs far beyond the 99.95 percent by building cross-zone redundancy and/or cross-region redundancy. Everything fails eventually. If AWS takes a hit for two hours, who is really to blame when the customer's website goes down because it did not design for failure? Don't blame the cloud provider; blame the architecture.

Recommendation: When choosing a cloud service model and cloud service providers, understand the risks and points of failure and design for failure. Understand the cloud providers' SLAs, data ownership policies, and thoroughly examine all legal binding documents and agreements. Each cloud service creates some level of vendor lock-in. Study the cause and effect of lock-in and make decisions at the service level. The cloud is not an all-or-nothing proposition. There is no need to be all in or all out. Choose wisely; architect even more wisely.

Underestimating the Impacts of Organizational Change

For a start-up beginning with a blank sheet of paper and building new systems, the cloud has few barriers. For established organizations with existing infrastructure and IT staffs with limited experience in the cloud, the impacts of organizational change should not be underestimated. Change goes way beyond IT, though. Business processes, accounting principles, human resource incentive programs, and legal processes have never had to deal with data and services existing outside of the enterprise. Procurement processes change from buying physical assets and software licenses to paying for virtual infrastructure and on-demand software. Capacity planning changes from the science of forecasting usage in the future to the science of real-time autoscaling. Security now takes center stage and securing data outside of the corporate firewall presents new challenges. The list goes on. Organizations that think of cloud computing as just another IT thing are in for a big surprise.

For some companies, their first attempt at cloud computing might be a proof-of-concept, a research and development exercise, or maybe a low-risk initiative like storing some noncritical data with a cloud storage provider. Small initiatives like these do not create large amounts of organizational change, which lowers the risk of failure. It is when companies take on larger projects or projects that create more risks that they need to implement a plan to manage organizational change.

AEA CASE STUDY: Dealing with Change

To give a common example of how organizational change can impact cloud computing initiatives, let's reintroduce our fictitious online auction company Acme eAuctions (AEA).

AEA built an internal customer relationship management (CRM) system nearly 10 years ago to support its on-premises web-based auction platform. Several members of the team that worked on the application are still employed at AEA today and are very proud of their work. However, as the years have passed the application has become increasingly expensive to manage and maintain and does not provide many of the features that a modern CRM tool has today, such as a mobile application, integration with third-party tools, social networking, and more. AEA has decided to move to a SaaS-based CRM solution and the IT team is fighting the decision. They are quoting articles and blog posts about problems with security and uptime in the cloud. The security team is struggling with the concept of putting customer data in the cloud. The SaaS vendor could have the solution turned on and configured in one day, but the effort to export the data out of the legacy system and into the new SaaS-based system is blocking the project. What should be a quick-and-easy win for IT has become a battle between the business and IT. How could AEA have avoided this conflict and the long project delays that the conflict has created?

The conflict is not a technology problem; it is a people problem. We have seen this pattern over and over though the years, whether people were resisting moving off of the mainframe to the client-server model, or pushing back on the enterprise use of the Internet, or refusing to embrace changes in business processes. The bottom line is when people are *told* to change, often their first response is to resist that change. We will discuss strategies for managing change and discuss how AEA addressed the organizational issues to get the project moving forward in Chapter 15 ("Assessing the Organizational Impact of the Cloud Model").

Recommendation: If possible, start with smaller, lower-risk initiatives as the early candidates for cloud computing projects. If the projects are risky and large in size, do not underestimate the impacts of organizational change. Assign somebody as the change leader. That person can be internal or external. Create a sense of urgency, form and empower a team to own the project, create a vision of the future state, and drive that message throughout the organization over and over using every communication mechanism possible (town hall meetings, blogs, newsletters, meetings, posters, etc.).

Skills Shortage

Enterprises often don't have the required expertise to build cloud-based solutions. The average medium-to-large company that has been in business for more than a few years typically has a collection of applications and services spanning multiple eras of application architecture from mainframe to client-server to commercial-off the-shelf and more. The majority of the skills internally are specialized around these different architectures. Often the system administrators and security experts have spent a lifetime working on physical hardware or on-premises virtualization. Cloud architectures are loosely coupled and stateless, which is not how most legacy applications have been built over the years. Many cloud initiatives require integrating with multiple cloud-based solutions from other vendors, partners, and customers. The methods used to test and deploy cloud-based solutions may be radically different and more agile than what companies are accustomed to in their legacy environments. Companies making a move to the cloud should realize that there is more to it than simply deploying or paying for software from a cloud vendor. There are significant changes from an architectural, business process, and people perspective. Often, the skills required to do it right do not exist within the enterprise.

Many legacy architectures have applications that depend on state, as mentioned earlier in this chapter. Building software for the cloud requires developing stateless applications. The secret to well-architected cloud services is to fully understand and embrace the concepts of RESTful services.* Many companies claim that they have service-oriented architectures (SOAs) but many implementations of Representational State Transfer (REST)–based SOA are nothing more than just a bunch of web services (JABOWS). Building RESTful services correctly requires exceptional engineers who know how to architect services in a manner that leverages the virtual compute resources in the cloud. If the developers building solutions in the cloud do not correctly handle the application state, do not provide the correct level of abstraction, and do not apply the right level of granularity, the cloud experience for that company will not be a pleasurable one. Another area where skills tend to fall short is in application security. Companies have built software for years that has lacked the appropriate level of application security. The applications often get by because the perimeter security might be good enough to keep most attacks out. Writing software to run in the cloud outside of the corporate

*Thomas Earl states that "statelessness is a preferred condition for services and one that promotes reusability and scalability." These are fundamental characteristics of cloud services.

firewall requires developers and architects who are highly knowledgeable about application security. On the system administration side, development skills are a higher priority in the cloud. Since many manual management tasks are now available as cloud services, administrators need to develop software in conjunction with the application development team. This also requires that administrators use similar application development tools and processes and become part of the release management lifecycle. In many companies application development and system administration do not work closely together and have entirely different skill sets. We will discuss the working relationship between development and operations in detail in Chapter 14 ("Leveraging a DevOps Culture to Deliver Software Faster and More Reliably").

Deploying, monitoring, and maintaining software in the cloud can be drastically different from how an organization currently handles those tasks. Whether developers, administrators, help desk staff, scrum masters, or whatever their role, these people need to have a solid understanding of cloud computing to excel in their jobs. They will need to have a broad knowledge of networking, security, distributed computing, SOA, web architectures, and much more. This change is no different from what happened when organizations shifted from mainframes to client-server architectures. People need to learn new methods in order to make the transition. At the same time, the company needs to keep the lights on and maintain the legacy systems. It is hard to find people who know all the different architectures from the old legacy approaches to the new cloud-based approaches. It can be a good strategy to bring in new staff with experience in the cloud and to help transform the organization and train the existing staff along the way. Trying to move to the cloud with people who are not experienced in cloud architectures will most likely not produce the best results.

Recommendation: Evaluate the current staff and identify skill gaps based on the project requirements. Plug those skill gaps with experienced talent, either full time or contract resources. Make sure the existing staff learns from the experienced people along the way. If only experienced people from the outside get to work on these initiatives, there could be a lot of resentment from within. Attend meet-ups and conferences where practitioners present (beware of vendor pitches), and reach out to local organizations that have had success in the cloud. Encourage team members to take courses, read blogs, and collaborate through their networks to learn about the other people's experiences.

Misunderstanding Customer Requirements

Sometimes IT people neglect the business side of the equation and build the cloud solution that is best for IT. It is critical that part of the service model selection is based on customers' needs. For example, if a company is building an

SaaS solution that is processing credit card information, the business requirements will be drastically different than if the company is building a sports news website. If a company is processing patient health records or top secret government information, the business requirements around security and privacy are far greater than if the company is building a streaming video product.

The details of the business requirements should drive which type of cloud deployment model (public, private, hybrid) and which type of service model (IaaS, PaaS, SaaS) to architect the solution on. If a company is building consumer-facing websites where users voluntarily exchange their personal data for a free service (Facebook, Twitter, Instagram, etc.), the company can easily justify putting everything in a public cloud. If a company is selling to enterprises such as retail establishments, hospitals, and government agencies, there is a very good chance that some customers will demand that at least some of the data is either in a private cloud or does not leave their premises. It is critical to know what the end-user requirements are when it comes to things like security, privacy, integration, regulatory constraints, and so forth. We will discuss these patterns in detail in Chapter 4 ("It Starts with Architecture"). If a company is building SaaS solutions, it should expect customers to demand the highest levels of security, privacy, and auditability requirements. SaaS software has to be secure, reliable, scalable, and configurable or many customers won't buy it. It is key to understand customers' expectations up front so their needs can be built into the underlying architecture from the start.

Recommendation: Understand the business requirements and customer expectations of cloud computing before selecting cloud service models and cloud types. Requirements drive the decisions; the decisions should not drive the requirements. Get clarity on the product definition and requirements, perform a security and regulatory assessment of the requirements, and add the gaps to the overall product backlog. Have a list of frequently asked questions handy that answers all of the questions and concerns that the typical customer will have for the cloud-based solution.

Unexpected Costs

One of the promises of cloud computing is that the pay-as-you-go model greatly reduces the cost of IT infrastructure. This only holds true if the software is architected and managed in a way that optimizes the use of cloud services. One of the powerful things about cloud computing is how quickly services or computing resources can be turned on, but if the process of consuming cloud resources is not closely governed, then the monthly costs can skyrocket, earning you a visit to the CFO's office to explain why you just destroyed her monthly forecast.

Each cloud service model brings a unique set of challenges for controlling costs. SaaS companies typically charge per user or by tier. For example, GitHub, a SaaS-based code repository, has tiers that range from free for public repositories, a Micro tier for up to five repositories for a minimal monthly fee, and several other tiers all the way to the Platinum plan that allows over 125 repositories for $200 a month.* Often these tools are left ungoverned and a company that should be using the Silver plan with 20 repositories at $50 a month is paying $200 for the Platinum and has no idea what repositories exist, who manages them, and which ones are even still being used. Some SaaS solutions charge monthly by user; others are transaction based. Transaction-based SaaS solutions like an e-mail campaign management tool charge per e-mail sent. Some companies use the e-mail tool's APIs and integrate them into their products. If they don't build in safeguards to protect from erroneous code like an infinite loop or a miscalculation, the end results could be a monthly bill that is thousands of dollars higher than expected. Make sure these risks are identified and throttles are built into the system to protect against such scenarios.

PaaS solutions have their share of challenges as well, when it comes to optimizing costs. One of the big benefits of PaaS is that the platforms handle scaling during peak times. The promise of PaaS is that it allows developers to focus on business requirements while the platform handles the infrastructure. Just as in the previous SaaS example, controls must be put in place to ensure that software bugs or even unexpected excessive workloads do not result in the PaaS consuming huge amounts of infrastructure and running up an enormous bill for the month.

IaaS solutions are even more critical to govern closely. It is so easy to deploy a virtual compute resource that without the right controls in place a company can quickly get into a "server sprawl" situation where hundreds of servers are running in various developer and R&D environments and the meter is running on them 24/7. To make matters worse, many of these servers will have dependencies on them that make them hard to shut off. One of my clients was a start-up with a small group of developers who managed everything manually. It worked well because each person was responsible for his own area and it was easy to track all of the virtual servers and what their dependencies were. The small start-up was purchased by a big company and many people were added to the team. No time was allocated to build in processes around managing compute resources and almost overnight the number of servers quadrupled. To make matters worse, nobody was really aware of the impact until finance noticed that the bill was starting to get enormous.

*GitHub's cost as of May 2013.

Unfortunately, a number of months had passed before this was noticed, and many of the compute resources had dependencies within the software lifecycle. Each project had one-to-many development, test, quality assurance, and stage environments, and they did not all have the same versions of patches and updates on them. When finance demanded that the team reduce costs, it took several months to take inventory and implement a plan to consolidate and standardize environments. To this day, that client believes that the cloud is more expensive than on-premises computing and is looking for ways to do less in the public cloud. The real issue with costs was a lack of governance, not the cost of cloud computing in public clouds.

The most expensive part of cloud computing usually has nothing to do with the cloud at all. Often, companies underestimate the effort it takes to build software in the cloud. In February 2013, KPMG International published a survey taken by 650 executives worldwide on the adoption of cloud computing. The survey found that one-third of the respondents discovered that costs related to their cloud computing initiatives were higher than they expected. The respondents pointed to a lack of in-house skills and complexities with integrating to existing systems as some of the main reasons for the inflated costs. With the right architecture and governance, cloud computing can drive down costs substantially within an organization, but there are no shortcuts or magic pills that will guarantee that success. At the end of the day it comes down to architecture and planning. In Chapter 4 ("It Starts with Architecture"), we will discuss this in more detail.

Recommendation: Understand the costs of each cloud service model and establish the appropriate levels of governance and software controls to optimize and monitor costs. For companies with legacy solutions, don't underestimate the effort required to integrate with legacy architectures and the costs of training existing employees or hiring experienced engineers.

Summary

The reality is that implementing cloud initiatives can be much more challenging than many people are led to believe. Companies that successfully implement cloud initiatives can reduce IT costs, improve speed to market, and focus more on their core competencies. Start-ups have the luxury of building from scratch for the cloud, but established companies have both legacy solutions and legacy IT shops that may require significant changes in order to have success moving to the cloud. Companies attempting to build in the cloud should become aware of the worst practices mentioned in this chapter and pay attention to the recommendations. The marketing hype from the vendors makes cloud computing initiatives seem incredibly easy. Don't be fooled by

their PowerPoint slides. It is critical that a company fully understands the pros and cons of each service model and the implications pertaining to key issues such as security, privacy, data ownership, regulations, costs, impact of organization change, and much more.

References

Bennett, C., and A. Tseitlin (2012, July 30). "Chaos Monkey Released into the Wild." Retrieved from http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html.

Cloud Security Alliance. 2011. "Security Guidance for Critical Areas of Focus in Cloud Computing v3.0." General format, retrieved from https://cloudsecurity alliance.org/guidance/csaguide.v3.0.pdf.

Erl, Thomas. 2008. *Service-Oriented Architecture: Concepts, Technology, and Design*. Boston, MA: Prentice Hall.

Hill, S., and R. Wright (2013, February). "The Cloud Takes Shape: Global Cloud Survey: The Implementation Challenge." Retrieved from http://www.kpmg.com/Global/en/IssuesAndInsights/ArticlesPublications/cloud-service-providers-survey/Documents/the-cloud-takes-shapev2.pdf.

Hoff, T. (2012, April 9). "The Instagram Architecture that Facebook Bought for a Cool Billion Dollars." Retrieved from http://highscalability.com/blog/2012/4/9/the-instagram-architecture-facebook-bought-for-a-cool-billio.html.

Johnson, R. (2011, July 5). "Army's $2.7 Billion Cloud Computing System Does Not Work." Retrieved from http://www.rawstory.com/rs/2011/07/05/armys-2–7-billion-cloud-computing-system-does-not-work/.

Linthicum, David S. 2009. *Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide*. Boston, MA: Addison-Wesley.

n.a. (2012, September 27). "Cyber Security: 75% of Data Breaches Are Inside Jobs." Retrieved from http://www.theinformationdaily.com/2012/09/27/75-of-data-breaches-are-inside-jobs.

Preimesberger, C. (2011, May 13). "Unplanned IT Downtime Can Cost $5K per Minute: Report." Retrieved from http://www.eweek.com/c/a/IT-Infrastructure/Unplanned-IT-Downtime-Can-Cost-5K-Per-Minute-Report-549007/.

Wainewright, P. (2009, February 3). "Coghead's Demise Highlights PaaS Lock-out Risk." Retrieved from http://www.zdnet.com/blog/saas/cogheads-demise-highlights-paas-lock-out-risk/668.