

# CLOUD MIGRATION

eMag Issue 33 - October 2015



## ARTICLE

The Cloud-Migration Checklist

## INTERVIEW

Anatomy of a Cloud Migration Program:

## ARTICLE

Transitioning to Cloud-Native Applications

## The Cloud-Migration Checklist

*Are you in the process of moving applications to a public cloud? You're not alone. 451 Research says that 46% of 2015 IT budgets are going towards off-premises systems, with that number expected to climb to over 50% within the next three years. In this article, we'll explore four stages in a cloud migration lifecycle and the questions to answer before exiting each one.*

## Anatomy of a Cloud Migration Program: Q&A with Tim Beerman

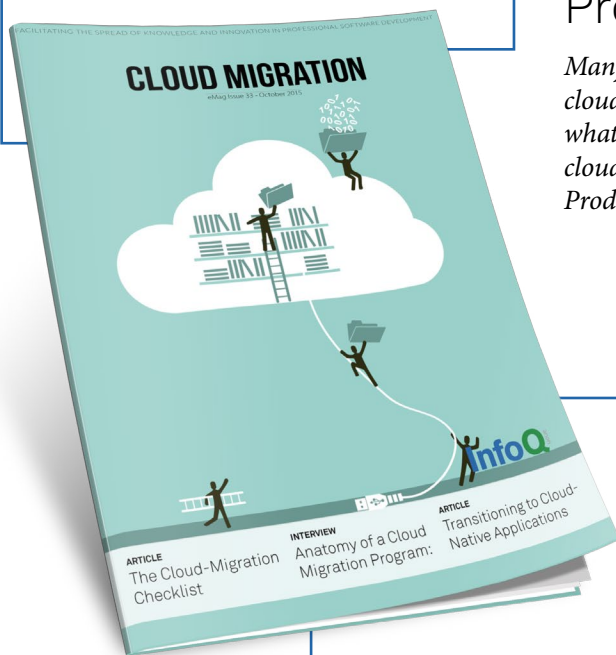
*Many cloud providers offer services to onboard new customers into the cloud. What advice can they give us on how to prepare for a migration, what pitfalls to avoid, and what types of apps are the best fit for the cloud? To learn more, InfoQ reached out to Tim Beerman, the VP of Product Strategy and Development at CenturyLink.*

## Transitioning to Cloud-Native Applications and Beyond

*Enterprises have continued to accelerate their adoption of cloud infrastructures. As this shift continues, it is important to understand what this means to applications that run in cloud environments.*

## Why You Should Definitely Migrate Existing Apps to the Cloud

*In this article we'll explore various benefits of migrating your existing apps to the cloud in detail so that you can make an informed decision.*



### FOLLOW US



facebook.com  
/InfoQ



@InfoQ



google.com  
/+InfoQ



linkedin.com  
company/infoq

### CONTACT US

GENERAL FEEDBACK [feedback@infoq.com](mailto:feedback@infoq.com)

ADVERTISING [sales@infoq.com](mailto:sales@infoq.com)

EDITORIAL [editors@infoq.com](mailto:editors@infoq.com)



Conference for Professional Software Developers  
[www.qconlondon.com](http://www.qconlondon.com)

## Tracks for QCon London 2016

- **Architectures You've Always Wondered about** - Case studies from: Google, LinkedIn, Alibaba, Twitter, and more...
- **Disrupting Finance** - Technology advances in finance (blockchain, P2P, Machine Learning, API's)
- **Data Streaming @ Scale with Spark** - Big data, fast-moving data. Practical implementation lessons on Real-time Data with Spark
- **Close to the Metal** - Get efficiency back into your code, concepts like: cache efficient algorithm and lock free data structures
- **Modern Native Languages** - Modern native languages: Safe efficiency with Go, Rust, Swift
- **Containers (in production)** - Real-world lessons on scalability and reliability in production container deployments
- **Modern CS in the real world** - Real-world Industry adoption of modern CS ideas
- **Full Stack Javascript** - Level up Javascript with topics like Angular, React/ReactNative, Node, Mongo/Couch/Other, Falcor, GraphQL, etc
- **DevOps & CI/CD** - Lessons/stories on optimizing the deployment pipeline
- **Security, Incident Response & Fraud Detection** - Master-level classes on building security into your system and responding to incidents when things go wrong.
- **Head-to-Tail Functional Languages** - Free-range Monads, Tackling immutability, tales from production, and more...
- **Data Science & Machine Learning Methods** - A developer's data science and machine learning toolkit
- **Optimizing You** - Keeping life in balance is always a challenge. Learning lifehacks
- **Microservices for Mega-Architectures** - Practical lessons on Microservices success.
- **Architecting for Failure** - Your system will fail. Take control before it takes you with it
- **Modern Agile Development** - Revisiting Agile today and tackling challenges we are seeing in the wild
- **21st Century Culture** - New ways to organise technology companies and workplace culture





**RICHARD SEROTER** is the VP of Product for CenturyLink, a Microsoft MVP for Integration, Pluralsight trainer, lead InfoQ.com editor for cloud computing, frequent public speaker, and author of multiple books on application integration strategies. Richard maintains a regularly updated blog on topics of architecture and solution design and can be found on Twitter as @rseroter.

## A LETTER FROM THE EDITOR

Countless surveys show that companies are steadily moving applications from on-premises to cloud environments. While it's easy to get excited about deploying brand new apps to the cloud, the reality is that most companies have on-premises data centers full of existing applications that could benefit from a cloud migration.

These migrations often seem straightforward -- just pick up the virtual machines and move them into a cloud! Inevitably, complexities surface and cast doubt on the entire endeavor. There's so much more to a cloud migration than a basic lift-and-shift exercise.

How do you assess your application portfolio and identify the candidates for cloud? What apps should you avoid migrating? How can you decide on the right cloud environment for a given app? Is it better to move apps incrementally or via a "big bang" exercise? What pitfalls should you look out for? InfoQ decided to investigate this topic further and share our findings. In this eMag, you'll find practical advice from leading practitioners in cloud. Discover new ideas and

considerations for planning out workload migrations.

This eMag has four major topics:

- 1 A cloud migration checklist that takes you through the multiple phases of a migration project, with key considerations for each phase.
- 2 Guidance on transitioning to a cloud-native mindset and deploying apps that can take the most benefit of a cloud environment.
- 3 A look at the types of applications that make sense for the cloud.
- 4 An interview with a team that regularly migrates complex customers to cloud environments.

As you read this eMag, you may see slight variations for your own use case, but hopefully much of the guidance resonates and provides actionable data to help you be more successful in your transition.

# The Cloud-Migration Checklist



by Richard Seroter

Are you in the process of moving applications to a public cloud? You're not alone. In its fourth-quarter 2014 "Voice of the Enterprise: Cloud Computing", 451 Research said that 46% of 2015 IT budgets would go towards off-premises systems, with that number expected to climb to over 50% within three years.

Organizations still purchase cloud services in order to save money, but increasingly identify strategic goals like "time to market", "improved availability", and "establishing new revenue streams" as primary drivers of adoption. A recent survey that showed that only 27% of participants were extremely satisfied with their cloud-migration experience indicates that there's clearly room to discuss how to introduce cloud services into your IT portfolio and successfully transition apps or data to the new environment.

To be sure, any migration is never truly "done". As you constantly hire new services to solve business problems, the cycle

starts all over, potentially with different technologies and objectives. While the below checklist may help you start your journey, it's important to personalize it to your particular situation!

## Assessment

There is no "one size fits all" cloud service. Many organizations try to identify a preferred cloud environment before understanding how that cloud matches their organization's maturity, culture, and application portfolio. What questions should you ask of yourself and the candidate providers?

- **Are you the right fit for this cloud?** There are countless providers of cloud services,

and not all of them fit your specific needs. One cloud may offer thousands of virtual servers in seconds, which is wonderful if you are among the organizations that need such a capability. Another could deliver white-glove managed services that may or may not be relevant to your mode of operation. Consider what matters most to your organization, and actively try out a handful of providers that fit the bill. Don't just blindly choose a leader in a particular cloud domain and accidentally choose a provider that hampers your ability to deliver services successfully. ►



- **What are your capacity demands?** Based on the type of workloads you're considering for the cloud, different providers may bubble to the top. Is your application portfolio bursting with modern, cloud-native applications that capitalize on thousands of cheap, ephemeral servers or containers? Do you need a relatively slow-growing pool of a few hundred durable compute resources? Consider your application landscape and make sure that providers can support your upper bounds of compute, storage, and network throughput.
- **What are the actual costs?** The cost of cloud is rarely the easily calculated number you see in the price list. Make sure to model real-life scenarios and look for special charges the provider may apply based on geography, backup storage, consumed bandwidth, API calls, and more. Also, don't forget to factor in costs for onboarding programs, support plans, and net new environments (e.g. performance testing, staging) that didn't exist on premises.
- **Where are the gaps, and are they still gaps after migration?** It's likely that you'll finish your first assessment of cloud providers and have a list of perceived or actual deficiencies. It's important to talk about those with the provider and see if there are plans to close those gaps or viable workarounds to those gaps. One possible workaround is that the infrastructure or application pattern that caused the gap is refactored to fit more natively into the cloud's defined model.
- **What's the fit for your existing tools?** If you've been

in business for more than a month, you probably have an existing set of tools and processes that you've grown accustomed to using. If you can't easily adapt your tools to the candidate cloud environments, make sure you're comfortable replacing the status quo with the toolset supported by the cloud vendor.

## Planning

Congratulations, you've picked a cloud provider for a particular business domain. Now comes the hard part: planning a migration! What are the most important things to consider when plotting out the migration strategy?

- **What are the candidate apps/workloads/environments?** Ideally, the first applications you move to the cloud aren't the trickiest, most business-critical ones. But either way, make sure that you do a comprehensive inventory of applications and indicate those that are the best fit — strategically or tactically — for a cloud environment.
- **What's the topology of the application architecture?** There's a good chance that your application architecture will differ in the cloud. Cloud servers, data services, and networks behave differently than their on-premises counterparts, so you may need to evolve your reference architecture and deployment process in order to fit in this new world.
- **What possible performance bottlenecks are being introduced?** Does moving to the cloud guarantee an instant boost in application performance? Absolutely not. If you take a monolithic application and distribute its parts among cloud services, you

may introduce unforeseen latency. If your applications have integration points with on-premises systems that do not migrate to the cloud, you may also see performance problems due to long-distance connections. Cloud servers come in all shapes and sizes, so make sure that you give your applications the necessary CPU, memory, and disk horsepower to meet or exceed their historic performance levels.

- **What are the hybrid-integration plans?** It's unrealistic to think that you'll move an entire application portfolio to the cloud at once, or maybe ever. Treat the cloud as a logical extension to your existing landscape and consider how you will extend your current data, network, and identity to the cloud.
- **Have you identified the first adopters?** A migration to the cloud can be disruptive, and it's important to find people within the organization who are eager to help define new standards and guide the organization through this important transition.
- **How will users access the environment?** Your co-workers probably already have too many passwords to manage. The last thing you want to do is add an entirely new set of complex credentials necessary to access a critical set of cloud services. Look at single sign-on (SSO) mechanisms offered by your cloud vendors and make this an upfront aspect of any project to adopt a cloud service.
- **How will you train staff?** It's important for your team to become deeply familiar with these new cloud services. Consider each audience (e.g. project managers, developers, architects, system

administrators) and tailor materials that help staff get comfortable in the cloud.

- **What internal processes have to change in order to capitalize on the new service?** Your documented procedures for hardware requisition, change management, testing, and deployments may not apply in the same way when using cloud services. If you attempt to overlay current processes on more agile, self-service environments, you may lose all the benefits that you were after in the first place. Do an honest assessment of the changes that you need to make.
- **How will you deploy updated code, data, and configurations to the environment?** Your new cloud service may work with your existing system-administration tools, or it may not. If you're using modern configuration-management tools like Chef or Ansible, you may find extensions that work seamlessly with your cloud destination. Make sure that you do a thorough simulation of the deployment and configuration process and identify any areas that need improvement.
- **What's the plan for operating this service after migration?** The migration is only the start of that application's new life in the cloud. What happens when a change is needed? How do you troubleshoot a problem? What monitoring hooks will you use to measure key performance indicators? Look at the full lifecycle of a cloud workload and consider all the care-and-feeding activities.
- **Do you have a strategy for cutover?** If your application

users can tolerate a prolonged downtime while a migration occurs, you may pursue a simple, yet disruptive cutover plan. However, if you want to minimize downtime, you will have to consider strategies where you set up the new environment and keep it in sync with the primary environment until the time comes to steer all traffic to the new instance.

- **How will financial processing occur?** Yes, you'll probably have to pay for your cloud usage. How will you do that? Do you plan on charging each business unit that consumes cloud resources or simply pay the entire bill out of a general fund? Make sure to think through the process of incurring charges, getting an invoice, and making a payment.
- **Have you completed a small-scale pilot that flexes all the above concerns?** Above all, don't go into a migration with only paper prototypes. Physically set up applications in the target cloud and do trial runs. Get familiar with the interface, capabilities, and constraints so that you develop hands-on expertise.

## Migration

With proper planning, the migration itself should be uneventful. To be sure, surprises will invariably surface when the actual migration is underway but, if you've answered the above questions, your team should be equipped to handle the unexpected.

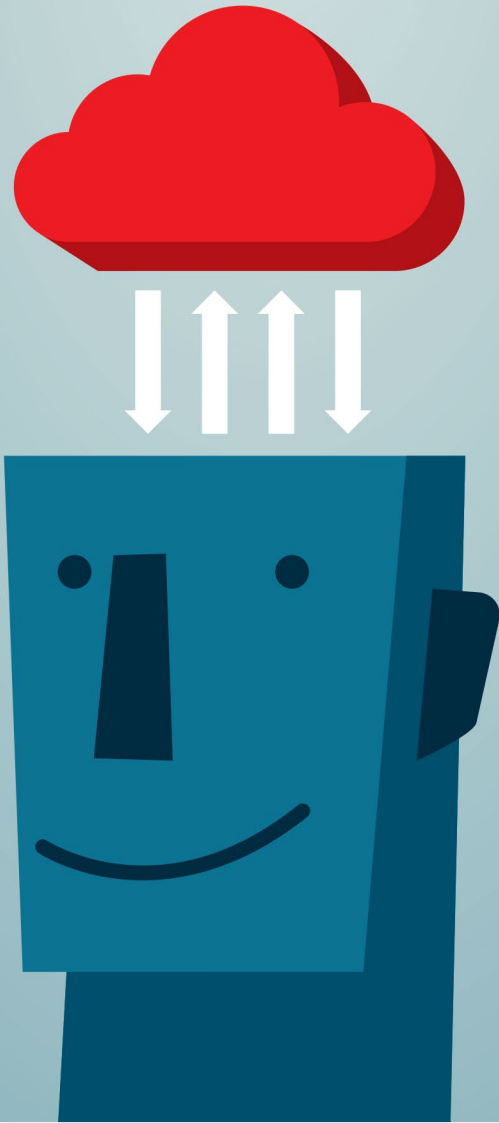
- **How are you distributing the apps and data to the cloud environment?** There are a handful of ways to get your applications and data up to a cloud location. For moderately sized workloads, you can often use simple copy commands and pass

this data over your Internet connection. For large data sets, however, you might be incurring significant bandwidth charges from your cloud provider and long transfer times. In those cases, it's better to either compress the data and copy it to a staging location in the target cloud before transferring it to the final destination or to physically ship drives to the cloud provider (if they support that).

- **What security controls are in place during transit?** During the migration, you may end up using staging servers or temporary object storage repositories. Make sure you've thought through the data and access security considerations, especially for sensitive data.
- **Migrate virtual machines.** Moving entire VMs is one way to migrate an application to the cloud. There can be unexpected side effects, depending on how the VM was attached to the on-premises network, the domain it was part of, the types of drives used, and more. While a "lift and shift" of VMs often appears to be the easiest route to the cloud, it's often more complex than expected. Additionally, that sort of approach doesn't force you to reconsider your application architecture and strategic ways to refactor your application for the cloud.
- **Migrate data.** Your data may move to a database-as-a-service environment, or a self-managed database instance. Make sure you know which tools are provided by the vendor and what limitations exist regarding data volume or structure. ►

*Stand out  
of the crowd*

Read **InfoQ**<sup>ueue</sup>  
content  
written by  
expert peers



- **Migrate applications.** If your application deployment tool can natively point to cloud infrastructure, containers, or application platforms, you're in good shape. You're also probably in the minority! It may take some reconfiguration to get your on-premises tools to deploy code to the cloud, or you might evaluate new tools to make the process more seamless.
- **Recreate metadata.** Companies often say that they want portability in the cloud and seek to avoid lock-in with a particular vendor. Good luck with that. While assets like virtual machines and application code can move relatively freely between clouds, environment metadata is often very provider specific. Account structures, users, permissions, policies, load balancers, and the like vary from cloud to cloud. Make sure you know how to set up the supporting configurations for your particular cloud destination.
- Validation
- Once the migration is completed, it's critical to verify that all aspects of the application are performing as expected.
- **Is the application reachable?** A simple test, but it's vital to check out all the various application services and make sure that users can access them and that internal components can communicate without error.
- **Did all the data make its way into the environment?** Through automation — or, at worst, a manual spot check — verify that both transaction data and reference data successfully transferred to the cloud. If you have complex data relationships, one failed migration can cause a cascading problem.
- **Can administrative tools access the cloud environment?** You hopefully verified this during your planning phase, but here's one last chance to validate that all of the management tools can see the cloud application and monitor it with no issue.

### Summary

Every day, organizations are successfully adopting cloud services and breathing new life into their IT portfolio by migrating applications to a more agile environment. Unrealistic expectations are a leading source of migration frustration. The best way to avoid this angst is to seriously evaluate your organization's goals and readiness, and answer practical questions about the planned migration process. ■



# Anatomy of a Cloud Migration Program: Q&A with Tim Beerman



by Richard Seroter



**Tim Beerman** is vice president of product strategy and development at CenturyLink, focusing on platform migration and onboarding strategy. Having joined CenturyLink in 2008, Tim has held senior leadership positions and has been responsible for world-class infrastructure products and application solutions that serve as the backbone of the company's managed hosting initiatives. Most recently, he drove the strategy to implement CenturyLink's managed-services suite of products in the core CenturyLink Cloud, laying the foundation for a broad range of future managed services under a single unified platform.

Many cloud providers offer services to bring new customers into the cloud. What advice can they give us on how to prepare for a migration and what pitfalls to avoid? To learn more, InfoQ reached out to Tim Beerman, a VP of strategy and development at CenturyLink.

## What things besides migration are part of cloud "onboarding"?

**Tim Beerman:** "Onboarding" is a very broad term that can mean a lot of different things to different audiences. In general, a successful platform onboarding program will accomplish three primary goals, all culminating in a delightful customer experience that extends throughout the

customer relationship: 1) general platform introduction and expectation/goal setting (i.e. What does the customer want to accomplish through onboarding?); 2) account configuration and hands-on training to accomplish the goals defined in step 1; and 3) a thorough closure session where the customer understands how to engage the platform support model and what tools and KBs are available to them for continued learning and updates.

I like to use the analogy that a good onboarding program should "teach the customer how to fish" so they learn how to be self-sufficient using the self-service capabilities available to them. Of course, different customers may have different requirements based on the scale of their operation, so the onboarding program should be able to accommodate as appropriate versus a "one size fits all" approach. ►

---

**Which aspects of migration can be faster than expected? Slower?**

---

Obviously, there are a lot of variables that come into play in a migration and it is largely dependent on the complexity of the environment. The one aspect of migrations that can often be slower than expected is the up-front planning. Many people believe that they can just pick up an image and start the migration process and turn everything up on the destination platform. There are a lot of things to consider as you start the process, including such things as licensing, IP addressing, performance differences in how the existing application may perform in the destination, etc. Careful planning and consideration of these elements up front can save a lot of time and effort on the back end. One thing that can often be faster than most people expect is the actual build-out of the destination site. With proper use of available automation and tools, the general configuration goes quite quickly depending on the migration method employed.

---

**Are you seeing that organizations start migrating on their own first and then get help, or start out being guided by others?**

---

Many customers migrating to a cloud are doing so for the self-service flexibility offered by cloud platforms. As a result, many choose to migrate on their own, especially if good documentation and migration tools such as image import are readily available and easy to use on the destination platform. This gives them a great opportunity to really understand the platform that

they will be operating within. That being said, this is where a strong onboarding program really shines as it provides an avenue for customers to learn the ins and outs of their new platform with some professional guidance as they get started.

---

**What types of apps do you discourage a company from migrating?**

---

Applications that have a transactional reliance on data sources (i.e. databases, mainframes, etc.) that are not proximate to the cloud location may not be good candidates for migration. Also, some applications that are designed for physical server implementations versus virtual are not good candidates for cloud migration as HA schemes may not work properly, performance may be impacted due to such things as storage types (FC vs. iSCSI or NFS), or licensing may not be compatible. This, of course, may not be an issue if your cloud provider offers physical servers as part of their cloud platform (shameless plug :)).

---

**Do you typically move an entire system (e.g. web, app, database servers) to the cloud or just a subset? If a subset, how do you avoid negative performance impact?**

---

This largely depends on the overall size and complexity of the environment and latency between source and destination. Whenever possible, it is best to move the entire stack — and if latency between locations is an issue, it becomes a requirement. If latency is not a big issue and it becomes necessary to break it up, then it

is best to move the app and application tier first (the web tier is usually pretty straight forward) because if any problems are encountered that are not a result of the separation, it is easier to fail that tier back to the source versus re-synching the database.

How does your team validate that a migration was successful?

Validation is an important final step in the migration process. In fact, throughout the entire migration process, certain steps have their own validations such as checksums on data transfers for individual components. Usually, once the new environment is built out, both environments will run in parallel (one in production and the other for testing) for a given period of time to ensure the application works properly and performs as expected before the final cutover. Once satisfied, we work with the customer on a final data synchronization and cutover of the production environment while keeping the original in place for further validation period. Of course, smaller or less complex environments may not need this diligence and cutover and validation can happen very quickly. In all cases, we rely on the customer to give the thumbs up that they are happy in their new home and support them as necessary to make that determination.

---

**How does migration to a SaaS or PaaS platform differ from moving to an IaaS environment?**

---

SaaS and PaaS migrations have their own unique challenges, but the primary difference from IaaS is that all the hardware is abstracted so you are largely just dealing with environment configurations and data transfer. If

a customer is migrating a SaaS environment, they are typically going from one SaaS provider to another that provides a similar service. In this case, the most complex piece is mapping the data elements from one SaaS application to another to facilitate a seamless data transfer where possible. Some SaaS providers provide tools to make this easier, especially if they are trying to capture market from one of their competitors. PaaS can be much simpler, especially if you are staying within the same PaaS framework as you switch providers (e.g. Ruby, LAMP, etc.). In this case, it is largely just a data transfer, although you want to carefully consider any performance or billing differences based on your new provider.

---

**What's the biggest misunderstanding that you have to address when first talking to a customer about migration?**

---

For the most part, customers go into a migration with eyes wide open since it can be an intrusive process to their business. The biggest challenge is that many customers assume that cloud is completely redundant and that if a component fails, it will seamlessly be picked up by another process or virtual machine. Customers often have this impression based on how they may have built internal clouds using common hypervisors configured for those needs. A public cloud platform is built for scale and to meet the needs of the many and as such may perform differently and handle component availability differently, so setting expectations is key as the migration process begins. In many cases, we recommend that customers look at their applications as part of the migration and see if they can

be modified to take advantage of cloud flexibility with features such as autoscale, geographic diversity, etc. This is not a trivial task, but educating customers on the possibilities allows them to make choices during migration or as fast-follower enhancements to ensure their cloud migration will be a long-term success.

---

**What existing tools do most companies want to keep using when switching their apps from on premises to the cloud?**

---

Common tools that customers want to keep are things like their ticketing system and monitoring systems specific to their applications to minimize impact to their current operations processes. This is where it is important for customers to understand the API capabilities of their new cloud platform, as they can often program elements of their new cloud to interact with their existing systems.

---

**What technique do you see companies use when performing a cutover? Do they accept downtime or try techniques to stay continuously online?**

---

This is largely dependent on the customer application and their overall business model. Mature, "born in the cloud" companies very often have developed their application as a distributed architecture and can often stand up new capacity on a target platform and perform a migration with very little to no downtime. While this is an evolving trend, it is still the exception rather than the rule.

For everyone else, it is often cost prohibitive to plan and execute a zero-downtime mi-

gration if the application was not built with that in mind (think IP addressing, geographic load balancing, data synchronization, etc.). That being said, with many of the tools available, it is possible to plan and execute a cutover well within established maintenance windows. Tools such as RackWare, Racemi, and others allow customers to establish a relationship between each virtual machine in the target and source locations and have the entire virtual machine (applications and data) replicated to the migration destination without impacting production. The replication relationship can even be broken on purpose to test the application in the new location and then re-established to do a final synch before production cutover.

---

**Operations departments have the (undeserved?) reputation as being blockers to cloud adoption, but is that your experience? Is there one group that is most likely to slow down a migration to cloud?**

---

Operations departments are usually very protective of the applications they are responsible for maintaining and are rightly wary of changes that may affect their established processes, procedures, and expertise. They may also fear that they will lose control or jobs, as some of their current functions are no longer necessary. As such, I would not characterize them as blockers, but as a key stakeholder in the migration planning process. It is important that the operations teams fully understand how the new cloud environment can improve many of their processes because of the ability to plug into robust APIs for better real-time data on their environment and even to drive more automation ►

of mundane tasks, allowing them to focus on being more proactive in supporting their business. This is also a great opportunity for forward-thinking operations teams to learn new skills and for companies to evaluate a broader DevOps approach to supporting their applications, but that could be a whole different interview. :)

---

### **What's the most complex technical challenge you encounter during migrations?**

---

There can be lots of complexities in the overall migration process, but the most complex technical challenges arise when migrations are between dissimilar architectures. When migrating from one cloud environment to another, things are relatively straightforward since the application is typically optimized for the cloud already. You just need to account for changes in cloud implementation. The real challenges appear for customers that are moving to a cloud for the first time, especially if their app is not currently virtualized. The biggest considerations typically include the following:

- License agreements for many commercial “off the shelf” applications may be different in the cloud or not supported at all.
- IP addressing in the cloud can often be different and hard-coded addresses within the application could require code modifications.
- Often, existing application platform components such as clustering protocols are not supported in cloud environments.
- Current application to Infrastructure dependencies may need to undergo an architec-

ture transformation for cloud deployments.

- Some application platform components and underlying middleware and operating systems could have direct binding with the current server hardware platform that may need to be addressed.
- Performance baselines for a cloud environment are often different and can typically be improved over existing infrastructure, however application system architecture needs to be revised to take advantage.
- If moving from an on-premises data center to a cloud, user functionality and the impact of network latency sometimes need to be addressed by a redesign of the application architecture.
- Transformation of proprietary implementation of existing load balancers and security controls may need to be redesigned for the new cloud environment.
- Transformation of services/systems-deployment lifecycle management tools and processes such as monitoring, backup and recovery, automation (config, build, patch, and deploy), ITIL, etc. may also need to be addressed.

---

### **What should an organization do to make itself migration friendly?**

---

Perhaps the biggest thing an organization can do to make itself migration friendly is to begin the process well informed and with an open mind. Be truly knowledgeable about the applications they are targeting for migra-

tion. Often, I see an organization embark on a migration and its people don't fully understand all the relationships their application needs to operate effectively. This can include things like data sources, other application dependencies that are not targeted for migration, and ports and protocols necessary to enable this communication. It is very frustrating for customers to start a migration and then find that it will not work as planned because of an unknown dependency.

Organizations should also be well informed about the capabilities and potential limitations of their chosen cloud platform. This is not limited to the technical capabilities, but also includes things like SLAs, billing models, how you get support, and if premium support models are available and necessary to meet operational expectations.

Finally, approach the migration with an open mind. Moving to a cloud platform opens up a lot of new possibilities for how a company approaches application design and future development, and this often requires a new way of thinking and operating. If an organization approaches a migration from a perspective that everything HAS TO work just like it did before, it often ends up disappointed or left wanting. ■



# Transitioning to Cloud-Native Applications and Beyond



**Ross Jimenez** believes all great technology starts with great people. He has been hacking the Web for 20+ years and is a veteran technology leader who has held numerous positions at Hewlett-Packard, Compaq, and Sandia National Laboratories. He currently is the engineering director at CenturyLink Innovation Labs.

Enterprises have continued to accelerate their adoption of cloud infrastructures. As this shift continues, it is important to understand what this means to applications that run in cloud environments.

Traditionally, applications relied on redundant and highly optimized infrastructure — not software architecture — to maintain high availability and reliability. The result was infrastructure that was more often than not severely underutilized. This excess capacity is essentially wasted money. Given the needs of modern organizations to decrease costs, increase efficiencies, and improve flexibility, this is unacceptable and has driven the move to the cloud.

## Cloud-native applications

Architects have a greater responsibility to design an application for the cloud in a way that increases uptime, fault tolerance, and scalability. Although many cloud providers talk about the reliability of their services, the true paradigm shift is that responsibility for uptime is moving to the application due to its ability to be aware of and control the infrastructure it is relying on. This

is fundamental to what makes an application cloud native.

Architects should first and foremost design cloud-native applications with failure in mind. Netflix has been touted for its cloud-native applications and the company's employees often blog about how they [design for failure](#). There are many patterns and practices to use not only for recovery from unexpected failures, but to make sure an application degrades gracefully as the unexpected happens. ►

Netflix provides a great [chest of tools](#) that simulate failures and latency, which can help build efficient cloud-native applications.

A great place to begin to understand how to design cloud-native applications is by reading and learning about [12-factor methodology](#). The 12 factors can be applied using any language in the creation of modern software that is delivered as a service.

Let's review a few of the 12-factor principles that will help in designing cloud-native applications. All processes should be [stateless and share nothing](#). Data, be it for state or other reasons, limits the distribution and scalability of a process, which means that you should never implement a process that expects a persistent piece of data. Instead, all data should persist outside the process in an attached service such as a database or other data store.

Another fundamental principle is that [dependencies should be explicitly declared and isolated](#). In practice, that means the application is environmentally agnostic and doesn't expect some tool or library to exist out-

side what is explicitly declared. Aligned with dependency declaration and isolation is the concept of a [config file that stores anything that is distinct to an execution environment](#), like development versus production credentials. Anything from connection strings to credentials to unique hostnames should be stored in some type of config file, distinct from code that uses it. These are just a few examples of patterns and practices that should be understood and implemented in all modern applications to make them suitable for deployment in the cloud.

Prior to starting that next project, make sure you and your team know what makes a 12-factor application.

### The future of modern applications

Several emerging trends and technologies have been picking up momentum in software engineering. We at [CenturyLink Labs](#) have been [working on containerization](#) and the role that containers play in the ways we will design, distribute, and deploy software in the future. Containers are becoming increas-

ingly popular because once an application is containerized, it is very easy to package, ship, and run anywhere. Unlike traditional virtualization, containers provide lightweight virtualization at the application level and thus are much better at squeezing every last ounce of utilization from available capacity.

Another great aspect of containers is they can start and stop in sub-second timeframes, making it easy to both scale out and recover from failures simply by starting up new containers. The ability to quickly start and stop is the core 12-factor principle called [disposability](#), which containers make tremendously easy.

This is a great paper from IBM on the [performance of virtual machines and Linux containers](#) that has a ton of detailed data. Read it if you're interested in comparing and contrasting containers and VMs.

The promise of containers is that they will deliver increased speed, cost savings and flexibility. They also open up new ways of thinking about software architectures like [using ephemeral containers to accomplish asynchronous processes](#) and workflows. If you find this area of containers interesting, be sure to check out the CenturyLink Labs [dry.it project](#).

If you have yet to learn about containers, check out various resources like [Docker](#) and the [CenturyLink Labs blog](#). Consider doing a POC within your company to get some practical hands-on experience.

### Immutable infrastructure and deployments

Another much-discussed, emerging practice that containers make easier is the concept of immutable infrastructures and deployments, which is the prac-



tice of never modifying the infrastructure or the application that is running. Instead, the application or infrastructure is always destroyed and recreated from scratch. But why would you want to do this?

Fundamentally, it is very simple: it is always harder to make and keep track of changes than it is to create new things. It simply is a matter of understanding and controlling all variables in a complex system. For small applications, this might not be a big deal, but it is compounded in modern, complex applications that are constantly evolving with frequent new features and patches. It can be nearly impossible to track if humans are making manual changes. The important thing is to keep the system in a known state, specifically one that has been thoroughly tested, so that you can easily recreate it regardless of the environment you're in.

If you haven't yet invested heavily in configuration management tools (Puppet, Chef, Ansible, etc.) you might want to brush up on immutable infrastructure and deployment practices, as it might be an easier way to keep things in a known state and help with the deployment speed, stability, and testability of your applications.

## Microservices

A final trend to discuss is the emergence of microservice-based architectures. Microservices is a service model in which every minor capability of an application is split into a distinct, self-contained program (called a service). Most modern web applications would have well over a dozen microservices. Containers again make this easier to accomplish and provide a great level of encapsulation.

However, microservices are not a panacea for distributed ap-

plication architectures, and they often only play a role in certain situations, specifically when a large team and the complexity of many people working on the same codebase starts to impact productivity. If you have this problem, it might be a reason to start refactoring your architecture to microservices. Instead of adding your next feature to your codebase, break it out as a microservice and slowly evolve your application away from its monolithic or macro-service orientation toward being a microservice application.

As with anything, if you break things up in smaller chunks, it is easier to troubleshoot, change, and iterate — especially using different people to focus on and own the specific capability that each microservice provides.

A great talk on the benefits and pitfalls of microservices can found on InfoQ: [“Scaling Gilt: from Monolithic Ruby Application to Distributed Scala Micro-Services Architecture.”](#) As with all new architecture paradigms, there are tradeoffs that need to be understood. Martin Fowler's blog post, [“Monolith-First”](#), talks about the many pitfalls of starting new applications with a microservices approach.

## Learn and explore

It is important to understand the tradeoffs of any new technology, pattern, or practice. Nothing is free, and although some technologies or practices might improve things in one area, they might make other areas more complex. Many of the new trends discussed still do not have the support of mature tooling. Enterprises should constantly test, explore, and learn by doing real-world POCs and having dedicated individuals whose role it is to understand emerging tech-

nologies and their potential benefits and impacts.

Cloud-native applications, application containers, immutable infrastructures, and microservices are all destined to have substantial impacts on the creation, delivery, and operation of applications, and should move the needle in improving the fundamental cost, efficiency, and flexibility of any business. ■

# Why You Should Definitely Migrate Existing Apps to the Cloud



**Basant Singh** is a software developer with 12 years of experience in application and database architecture. For the past five years he has applied his programming skills to power startups beyond the MVP stage. Currently, he is passionately gearing up to work on his next project on Go programming and Docker. He has built applications for Fortune 500 clients across the globe while working for a multinational corporation. Blogs: Techno-Pulse and GolangPro. Twitter: @SinghBasant.

Do your customers or users really care whether your application is hosted in a tier-one data center or the public cloud? Probably not. But they do care about their experience while using your service on a regular basis. Customers expect quick and reliable service that doesn't succumb to spikes in traffic and consistently performs great. They also expect you to update your service deliveries with the latest technology developments.

In business parlance, this translates to:

Is the infrastructure of your existing application (say, a healthcare app) agile and flexible enough to let you quickly roll out a new feature like alerting a patient when it's time to take pills with a voice reminder or other stimuli on a wearable device such as a smartwatch?

Let's explore various benefits of migrating your existing apps to the cloud so that you can make an informed decision.


## Elastic capacity

Look at these headlines:

- Servers crashed and tickets sold out for Big Bang's '2015 World Tour' in Seoul
- Flipkart hosts biggest-ever sale, website 'crashes'

- China corruption website crashes

The above occurrences are quite common and often grab more media attention than they deserve. Although the first time an app crashes due to popularity is more a sweet than sour experience, you simply can't ignore a crash.

Can your app survive an unprecedented spike in traffic? 



To answer this question, you may need to ask your infrastructure provider how quickly it can add capabilities to take care of the spike. Is your provider capable of provisioning (or de-provisioning) resources within minutes?

Unlike traditional service providers, cloud services are designed with built-in tools to facilitate rapid elasticity, which ensures that:

1. extra resources like virtual machines (VMs) responsively serve users in cases of peak traffic;
2. extra resources are de-provisioned (VMs released back to the cloud) as soon as the crowd is gone, which saves a lot of money; and
3. your application has on-demand, instant access to infinite resources.

At any time, cloud services should be capable of offering you the proper capability based on your workload, neither over-provisioning nor under-provisioning and performing quickly and automatically.

A traditional infrastructure provider may take hours or commonly days to react to a change in workload, as it requires manual intervention for this kind of response.

However, do not confuse elasticity with scalability. Elasticity is implicitly achieved by deploying on a cloud infrastructure whereas an application is scalable by virtue of its architecture. An application is scalable when it has the ability to accommodate more load when extra power (RAM, CPU) is added to a physical machine (vertical scaling or scaling up) or extra physical machines are added (horizontal scaling or scaling out) to the existing resource pool.

Moving to a cloud service doesn't always guarantee scalability. An application build

on standard-edition Oracle or SQL Server databases that do not support partitioning (a prerequisite for a scalable architecture) may require huge re-engineering efforts to scale. NoSQL databases are highly scalable. Similarly, you may host your application on a multicore server but the cores are of little use if your application has not implemented multithreading. This is why Google has come up with a new language, Go: to make concurrency and multithreading easier.

Not all apps require elastic capability. For example, a small business running an internal human-resources app for a stable company size of 50-odd employees might not have a good use case to migrate to the cloud just for the sake of elasticity.

### Cost reduction

Cloud hosting saves money through a reduction in capital expenses, as you don't need to buy IT infrastructure. Cloud providers do not require any long-term commitment.

You also only pay for what you use. Metered by usage is one of the essential attributes of a cloud service. The service plans are based on number of hours for pre-configured VMs and on gigabytes for bandwidth, data transfer, storage, etc. Configure and provision it right for maximum savings.

In the traditional data-center approach, most of the time you over-provision and keep paying for something that you may use once in a blue moon.

### Agility: Faster time to market

As an entrepreneur, you must look into the future, spot an opportunity, make a quick decision to build a new feature in your application, and release it to market to the surprise of your customers and competition. The

automated infrastructure and a quick turnaround time of a cloud service provider helps you progress more quickly than the bureaucratic request, procurement, and installation practices of a traditional service provider.

Have you noticed how e-commerce businesses are gradually moving from mobile-first to app-only strategies? Is this a suitable strategy for you and, if so, are you agile enough to quickly respond to this kind of a technology shift?

### Focus on your business

As most IT chores can be automated by opting for the cloud, managing IT through traditional ways simply wastes valuable time and energy, as it doesn't directly contribute to your core business.

Avoid the distractions, and align and engage your workforce in higher-value activities. Time is the new money.

### Increased availability, reliability, and performance

If not in cloud-service SLAs, where can you find genuine three 9s and four 9s (i.e. 99.99%) availability? Although traditional data centers may claim to match those figures, most of them don't have the appropriate measuring and monitoring services required to accurately audit their SLAs.

To increase availability and reliability, cloud services offer multiple availability zones (AWS, Rackspace), a federated network of public cloud providers, and multi-hypervisor design to approach that elusive 100% availability!

Many existing applications with users in a specific geographic region have reported decreased network latency by selecting and migrating to a cloud provider in the appropriate location. ►

## Disaster recovery

A variation of Murphy's law is that of there is a worst time for something to go wrong, it will happen then.

Can your application withstand a disaster like a network/power outage, fire, flood or other physical damage to the building that houses your server resources?

A disaster recovery plan (DRP) requires meticulous planning and design to recognize a failure and respond quickly to mitigate any loss to the business. DR is a tricky subject; most businesses either don't have a DRP or have one that is bound to fail due to minor gaps in the plan that they have missed because they never rehearsed a disaster scenario. They don't realize this until they are actually stricken by that rare disaster!

A battle-hardened DRP is critical for business continuity. Cloud is resilient by design and you can choose the right set of tools and configurations so that your business survives a disaster.

## The cloud is secure

Complete security is an illusion. Cloud services are in no way less

secure than any of the existing systems in place. In fact, cloud service providers are known for their security innovations. It is likely that they implement better physical and logical security practices than a standalone, on-premises data center. Many cloud providers conform to ISO, PCI DSS, EU Model Clauses, and other global security certifications.

But not all applications require bank-grade security, do they? If you handle highly sensitive data or your app is subjected to specific security and privacy regulations (such as HIPAA and HITECH) you can opt for hybrid cloud services (which are recommended for healthcare apps).

Questions of cloud security are often based more on fear, uncertainty, and doubt than in reality.

## A few other benefits you can't ignore

Apart from the benefits that you instantly receive once you migrate to the cloud, there are intangible benefits as well. Public [cloud services are green](#) be-

cause resource pooling is one of their essential characteristics.

Cloud services empower you to efficiently work from anywhere with collaboration and efficient document control. While migrating may require a little re-engineering here and there, it also opens opportunities to think of implementing some cutting-edge technologies like Docker (container-based virtualization that has seen unprecedented adoption) for easier deployment. Also, with IoT taking the center stage in 2015, you may need to integrate microservice architecture in your existing apps to address the near-future demands of your customers.

## Which applications are suitable for migration to cloud services?

As a rule of thumb, most applications developed during last seven years — n-tier or the popular three-tier web applications, batch processes, and back-end services — might be a good fit for migration. Older applications may require more re-engineering effort. It all depends on the architecture of the existing web application.

Various factors can make an existing application incompatible with or may require tremendous effort for cloud migration:

1. a tightly coupled architecture with hard-coded configurations;
2. a database that is not supported by the cloud; the app requires specific hardware to function such as hardware-based encryption, mainframes, etc.;
3. dependencies on third-party apps; and
4. licensing issues.

Here's a ready reference that might help you decide:





<b>Packaged Applications</b> E-mail, collaboration, and productivity apps	<b>Healthcare Apps</b>	<b>Legacy Apps</b>
<b>Seasonal Applications</b> Payroll processing, tax filing, school/college admissions, specific bidding apps, event management, etc.	<b>Stringent Compliance</b> HIPAA - HITECH, US-EU Safe Harbor, EU General Data Protection Regulation (GDPR), etc.	<b>Database Support</b> Microsoft SQL Server versions before 2008 R2, MySQL versions before 5.1, and Oracle database versions before 11g are not fit for cloud migration. They will require moves to the supported versions of the databases. For Microsoft Azure, Microsoft may provide limited or no technical support for any version of SQL Server earlier than SQL Server 2005. Azure VM images (templates) are available from SQL Server 2008 R2.
<b>Apps with Expected Traffic Spikes</b> E-commerce apps during Black Friday or other peak shopping times, royal weddings, sports sites, ticket booking, etc.	<b>Banking Apps</b> PCI DSS compliance, security concerns, reliance on mainframes	<b>Licensing Issues</b> Your existing licenses can be used in the cloud only with a license mobility option like "bring your own license" (BYOL). For most legacy apps, this option may not be available.
<b>Apps with Unexpected Traffic Spikes</b> Flash crowds due to mention in authoritative sources, like the Chinese website on sharing info about corruption that crashed shortly after launch.	<b>Geopolitical Constraints</b> Most cloud data centers are located in North America or Western Europe. By design, data in the cloud is borderless unless configured with restrictions. Your country may have specific data regulations you need to consider before migration.	<b>ERP Systems</b> Architecture bottlenecks such as tight coupling, hard-coded configuration, dependency on third-party apps, or specific hardware-based encryption.
<b>Startup Apps/Online Games</b> Size up or down based on your success or failure.		
<b>Big-Data (Analytics) Apps</b>		
<b>Supercomputing Needs</b> High bandwidth, enhanced networking, and high-performance computing (HPC).		
<b>Microservices</b>		
<b>PoC, Development, and QA</b>		
<b>Backup/Archiving/ Storage</b>		

## Action speaks louder than words

Here's a list of organizations that migrated their existing applications to the cloud. It also mentions the benefits they acquired after migration.

Serial#	Organization	The Challenge	Migrated to	Benefit
1.	<a href="#">Animoto</a> American video and slideshow maker.	Originally launched the app on its own servers. The Facebook application went viral in April 2008 with nearly 750,000 people signing up in three days.	AWS	At the peak, almost 25,000 people tried Animoto in a single hour! To meet this demand, the app scaled up from 50 EC2 instances (i.e. virtual servers) to 3,500 instances.
2.	<a href="#">redBus</a> Indian online bus-ticketing service.	The infrastructure of its traditional data center could not effectively handle processing fluctuations, which reduced productivity. Procurement and upgrades of server configurations took a long time.	AWS	Overall cost benefit of about 30%-40%. Reduced latency by about a factor of four. Traffic to redBus tripled due to reduced latency.
3.	<a href="#">Expedia</a> Online travel company.	The company had to run Expedia Suggest Service (ESS) in locations physically close to customers to enable a quick and responsive service with minimal network latency.	AWS	Decreased average network latency from 700 milliseconds to less than 50 milliseconds
4.	<a href="#">Xiaomi</a> Chinese high-end smartphone maker.	Needed reliability and connection speeds that would satisfy users in various countries and could seamlessly expand as the company grew.	AWS	30% faster download speeds. Shortened service delivery cycle and quick launch of the app-download center. Superior access speed with reduced up-front investment cost.
5.	<a href="#">Aucor</a> Finnish web-design company.	Its virtual, private servers worked nicely on a small scale, but when the number of projects grew, it needed too many servers and full-time system administrators.	GAE	Can handle over 70,000 requests per second without users noticing a thing.
6.	<a href="#">Khan Academy</a> American non-profit educational organization.	Maintained a website for its growing video library for several years, but experienced limitations as traffic increased.	GAE	Ability to support 3.8 million unique visits each month, along with 1.5 million practice questions served and answered every school day. Stores a growing collection of 2,000-plus videos. A system that easily handles usage surges.

## Conclusion

With tens of thousands of apps already migrated and functioning in the cloud, there's no doubt that the cloud is real and has lived up to industry's expectations. Irrespective of the location and domain, majority of the companies around the world

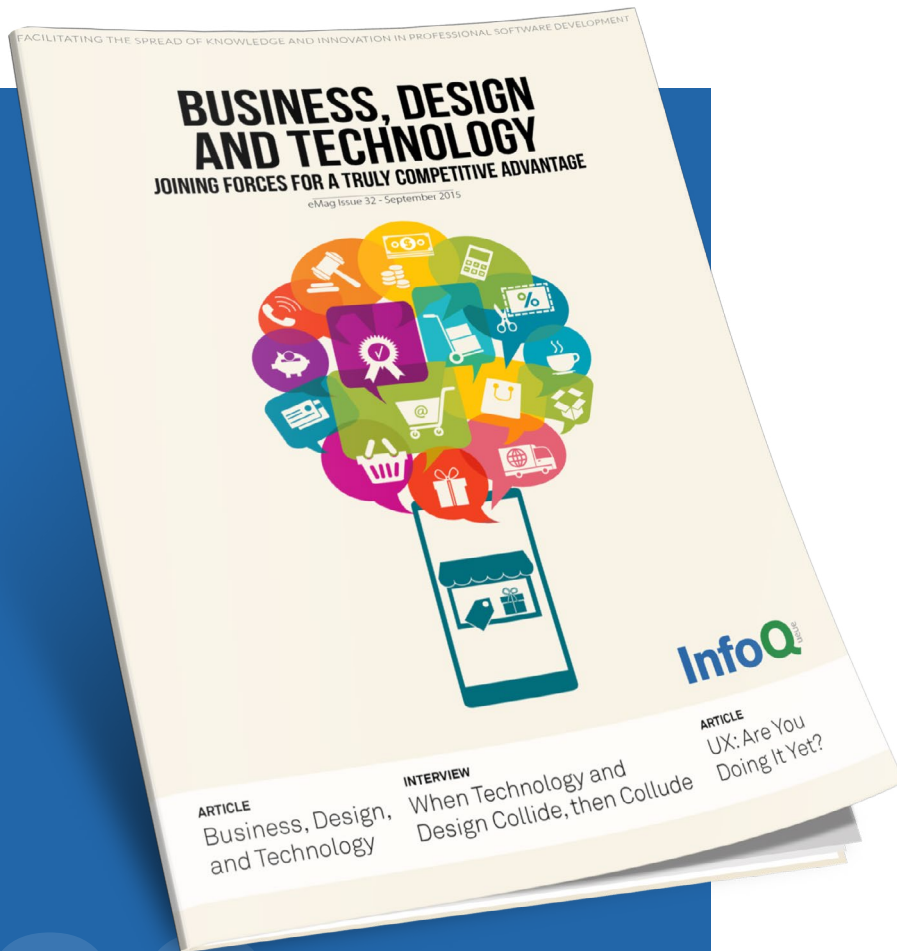
are using cloud services in some form or the other.

Back in 2008, when inflated expectations for the cloud peaked, Larry Ellison, CEO of Oracle [said](#), "The computer industry is the only industry that is more fashion-driven than women's fashion. Maybe I'm an idiot, but

I have no idea what anyone is talking about. What is it?" The IT behemoths who ignored the trend are now lagging behind, with declining fortunes. They are now scrambling for the cloud — playing catch-up! Plan your move and act before it's too late. ■



# PREVIOUS ISSUES



ARTICLE  
Business, Design,  
and Technology

INTERVIEW  
When Technology and  
Design Collide, then Collude

ARTICLE  
UX: Are You  
Doing It Yet?

## Business, Design and Technology: Joining Forces for a Truly Competitive Advantage

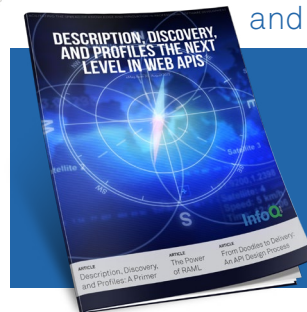
This eMag offers readers tactical approaches to building software experiences that your users will love. Break down existing silos and create an environment for cross-collaborative teams: placing technology, business and user experience design at the core.

## Architectures You Always Wondered About



In this eMag we take a look at the state of the art for the microservice architectural style in both theory and practice. Amongst others Martin Fowler talks about Microservice trade-offs, Eric Evans explores the interplay of Domain-Driven Design, microservices, event-sourcing, and CQRS, Randy Shoup talks about Lessons from Google and eBay, and Yoni Goldberg describes Gilt's experience.

## Description, Discovery, and Profiles



This eMag focuses on three key areas of "meta-language" for Web APIs: API Description, API Discovery, and API Profiles. You'll see articles covering all three of these important trends as well as interviews with some of the key personalities in this fast-moving space.

## QCon New York 2015 Report



This eMag provides a round-up of what we and some of our attendees felt were the most important sessions from QCon New York 2015.