# Five Options for Migrating Applications to the Cloud: Rehost, Refactor, Revise, Rebuild or Replace

Jeff Woods

Gartner The Future of IT Conference

October 4-6, 2011
Centro Banamex
Mexico City, Mexico

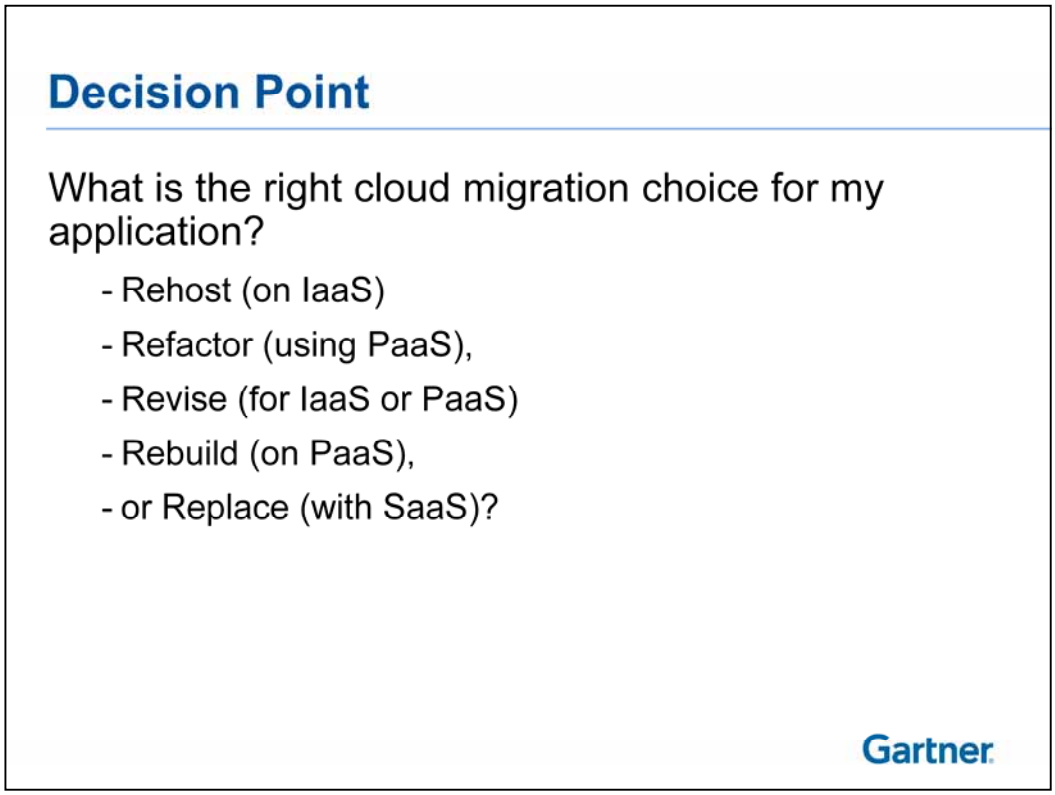
Notes accompany this presentation. Please select Notes Page view.



Gartner

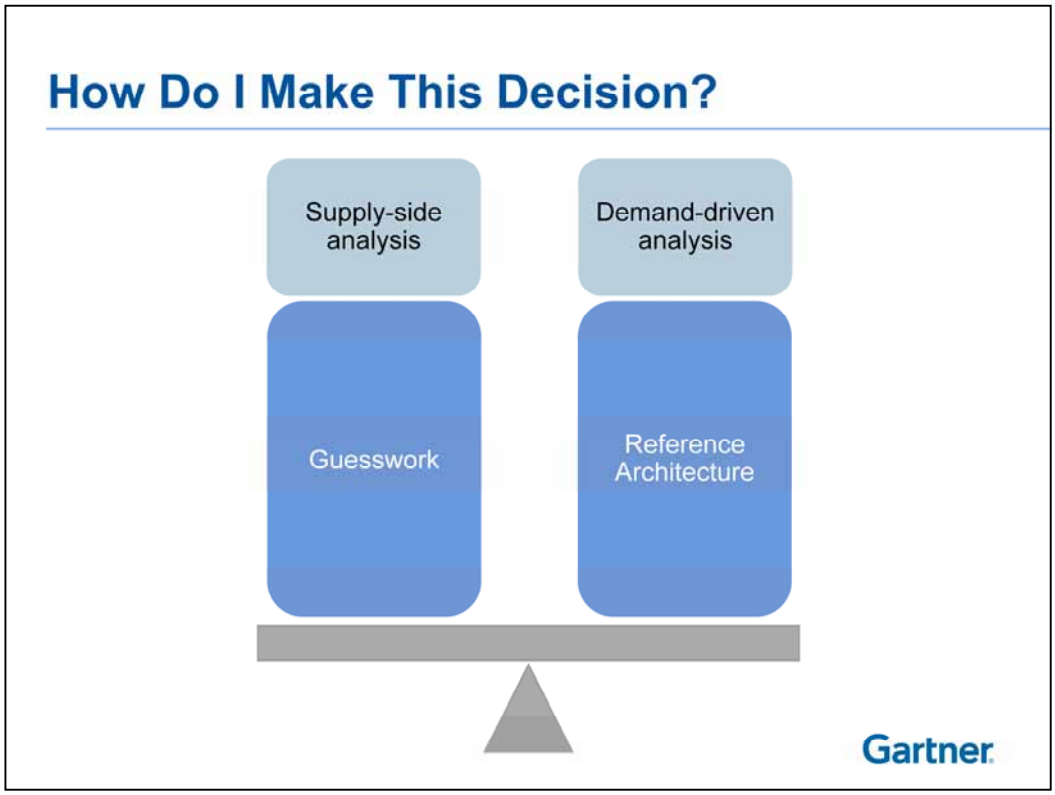

Like most things, cloud computing is about matching demand with supply. We know the supply is there. That is what the cloud providers are telling us. But what about the demand? This year we talked to many clients who have had instructions from senior IT executives similar to what we see above. The software delivery team must have an answer to strategy decisions about cloud computing, to figure out what the directive to "move some apps to the cloud" means. 2009/2010 was about what cloud computing is (i.e., what's public, private, internal and external), so I'm not going to cover that today — you've had enough of that. 2011 is about how do we do it — what are the options for moving apps and data to the cloud? You can rent a storage unit into which you move some of your applications. It doesn't mean you've dealt with them in a meaningful way. And now you have less control over your mess, the external applications are more difficult to combine with the ones at home.

*Image Credit: (the_matt) [http://www.flickr.com/photos/the_matt/1440298280/](http://www.flickr.com/photos/the_matt/1440298280/)*

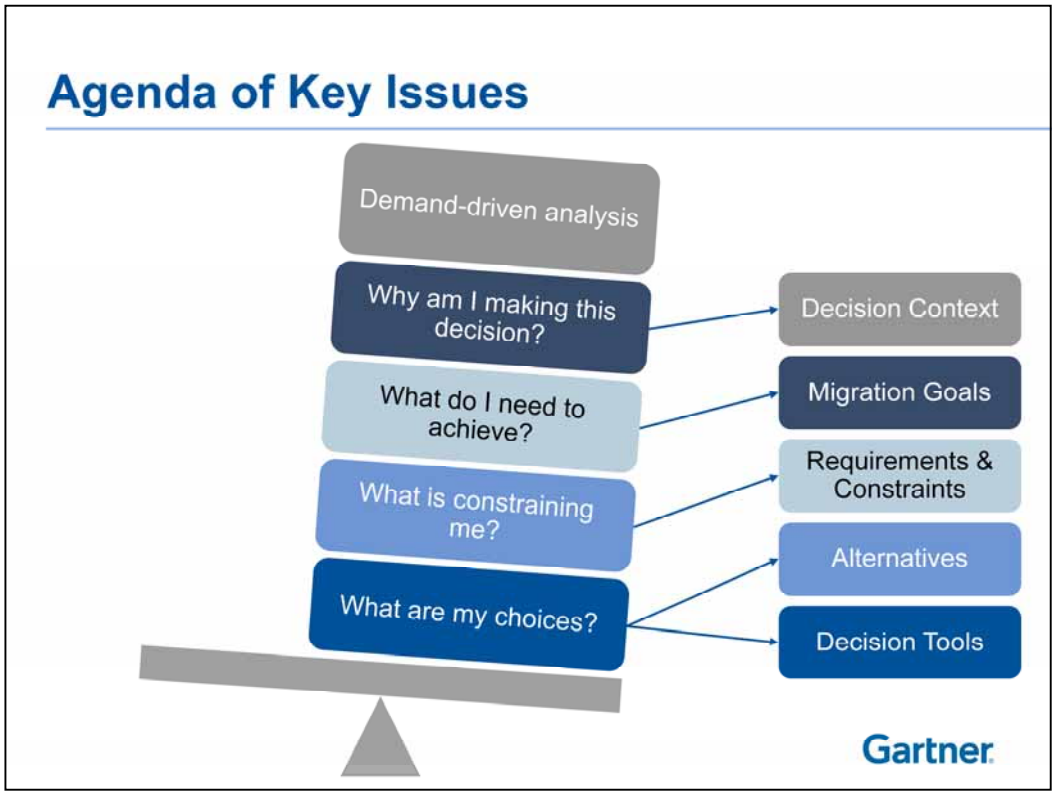



A decision point is part of the reference architecture component of IT1 research. The audience is a knowledgeable architect tasked with making a decision. Decision points classify technologies or mechanisms that address a common set of requirements. They deal with problems where customers are trying to make complex decisions based on repeatable criteria. This is not binary world: It should be something that informed people disagree about.

The decision in scope for this decision point is not solely a question of migration, but is truly one of optimization. Another way to state the question to be decided is: Which cloud platform and migration techniques offer the chance to optimize the application's contribution to stated and implied business and IT goals? Those business and supporting IT goals, described next, should be driving a migration decision, not a rush to experiment with new toys.




If you haven't seen any Burton IT1 research, this is how we work, we're demand-driven analyst team. Every decision starts with customer requirements and context. A decision point is a decision-making tool and not a foolproof guide or scientific method. It helps you make an explicit choice between reasonable alternatives. Decision points are not used to choose between two products, but to choose between types or categories of a product. They won't help you decide between two cloud platform providers, but will help decide which of IaaS, PaaS or SaaS you need. Of course, this decision is not a binary one — there is no right answer — just a best fit stretching over possibly conflicting requirements.



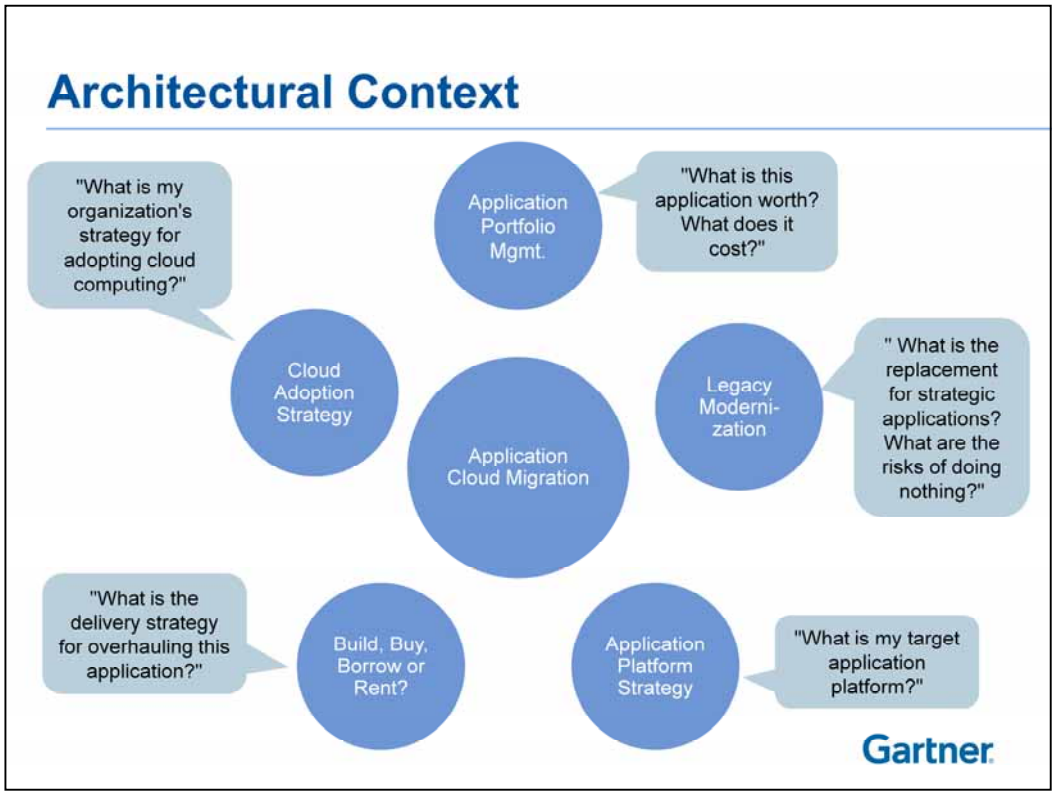

The chart illustrates the key issues that shape the rest of this presentation.

First, ask "Why am I making this decision?" The resulting "decision context" outlines the business and architecture environment that will lead you to a decision. The context is shaped by three assumptions:

1.This is not a decision about whether or not to migrate applications to the cloud. That decision has already been made. This is a "how do I do it" execution decision, not a "should I do it" strategy decision. It implies that many security, availability and trust issues have already been answered.

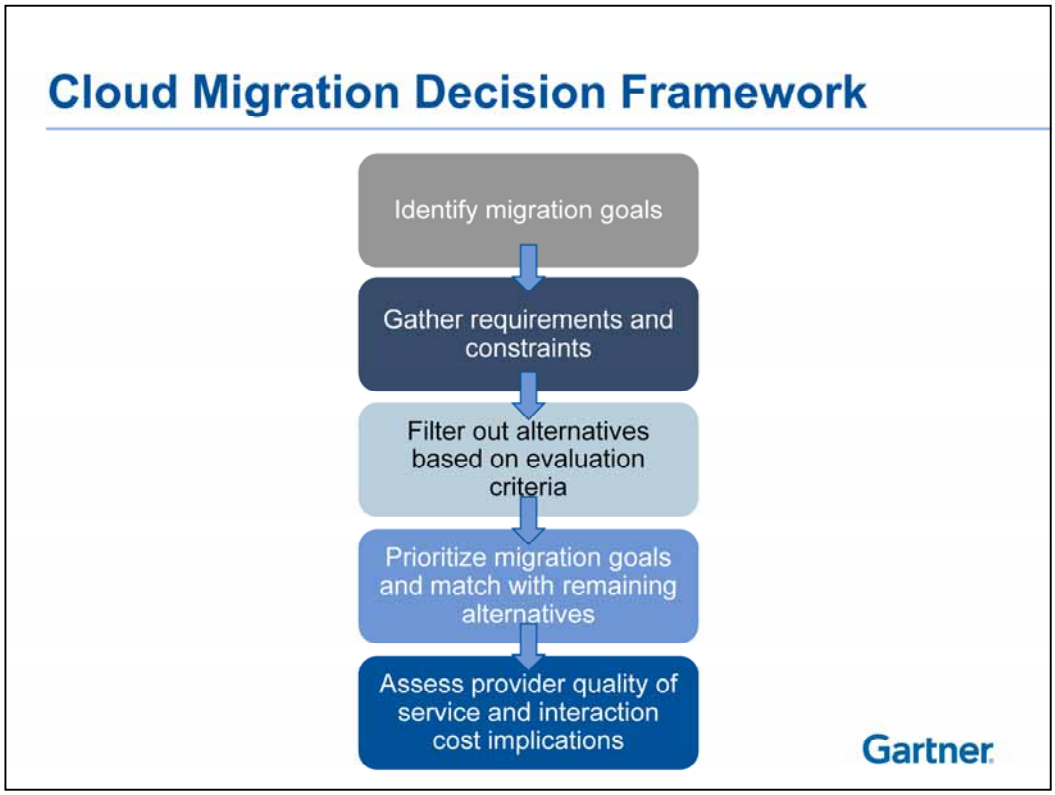
2. The application already exists in some form. This is not a "greenfield" project. Several of the alternatives involve abandoning existing systems, but the requirements are well-known. Options share the "re-" prefix, because migration is involved. This application is for production use, not for development or testing.

3. This decision is not really a question of migration but of optimization. Which platform and migration techniques offer the chance to optimize the degree to which the application contributes to stated and implied business and IT goals? Those business and supporting IT goals, described next, should be driving a migration decision, not a rush to experiment with new toys.

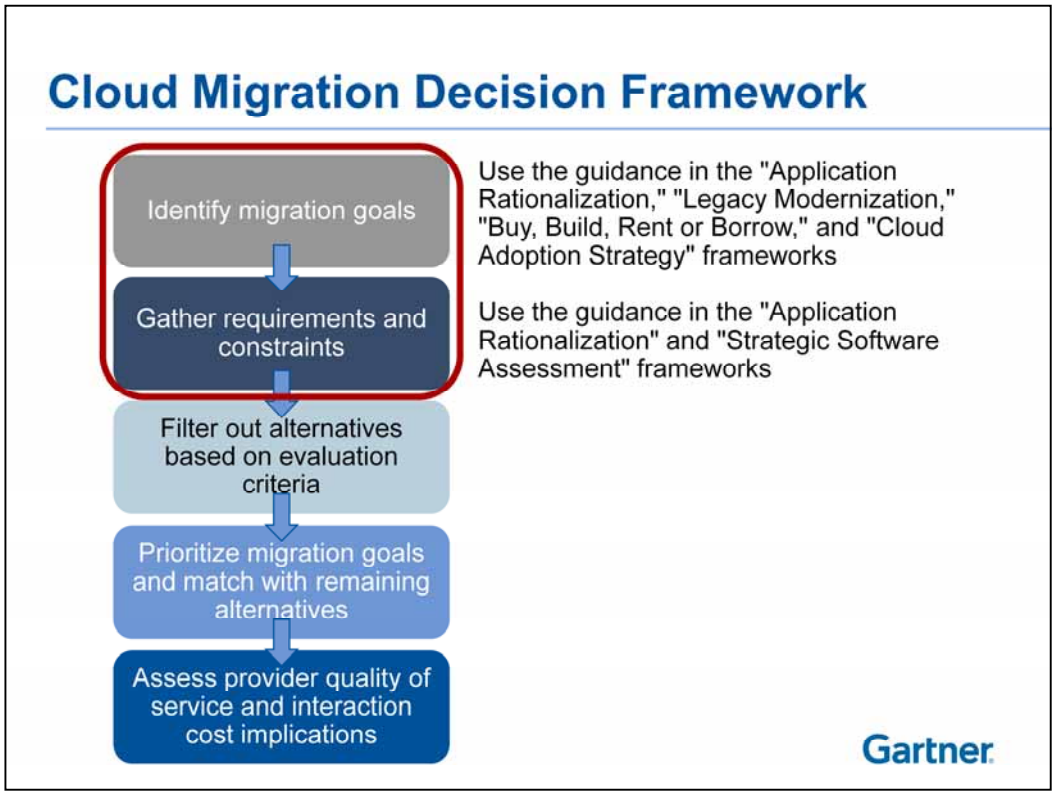



Choosing the optimal application migration option is a decision that cannot be made in isolation. Any cloud migration decision is, in essence, an application or infrastructure modernization decision and needs to be made in the larger context of related application portfolio management (APM) and infrastructure portfolio management (IPM) programs. The decision to migrate an application to one cloud tier or another is, in reality, a small issue shaping the IT portfolio and project management machine. Successful organizations will apply existing Gartner frameworks guiding the activities in the above chart before they decide how to approach the cloud-specific sourcing issues required by an application's modernization approach. Use the guidance provided in the following Gartner IT1 frameworks:

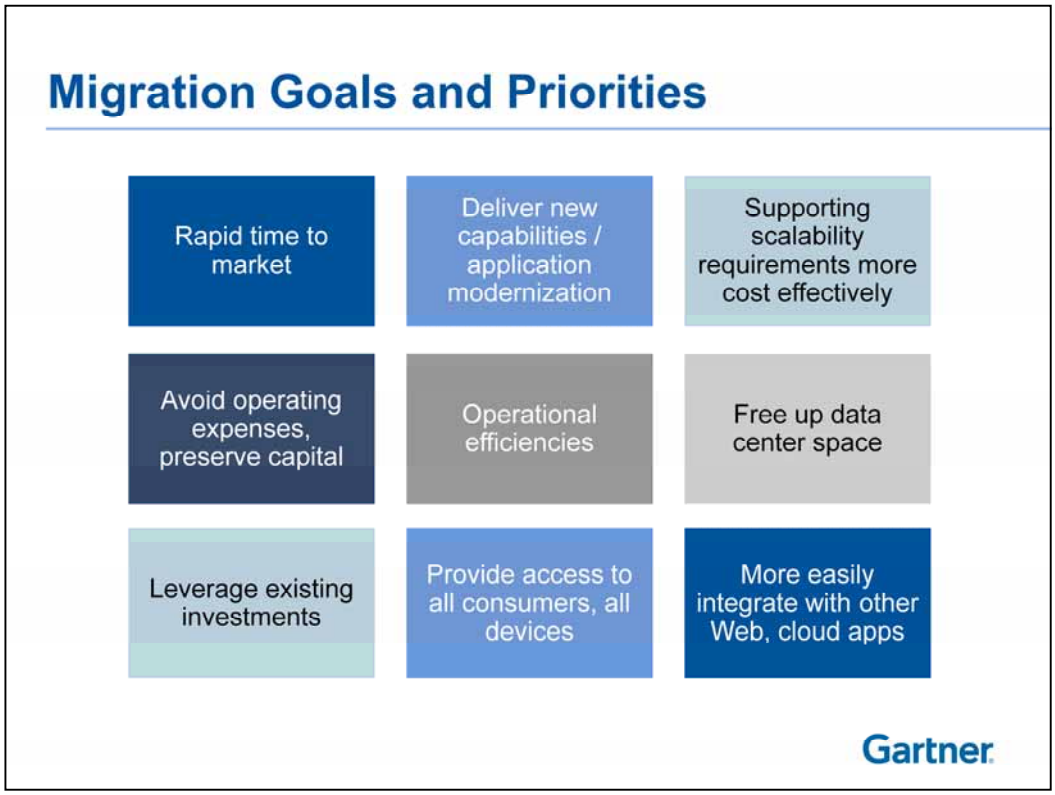
- "Application Rationalization: Burning Fat and Building Muscle" (G00203228)
- "Strategic Software Assessment Framework" (G00205755)
- "SWLegMod: A Framework for Legacy Software Modernization" (G00203866)
- "Buy, Build, Rent or Borrow: Choosing Custom Development Software, Open Source Software, Commercial Off-the-Shelf Software, or Software as a Service" (G00203291)
- "Building a Solid Cloud Adoption Strategy: Success by Design" (G00203990)
- "Application Platform Strategies for the 2010's" (G00203785)
- "2011 Planning Guide: Application Platform Strategies" (G00209998)



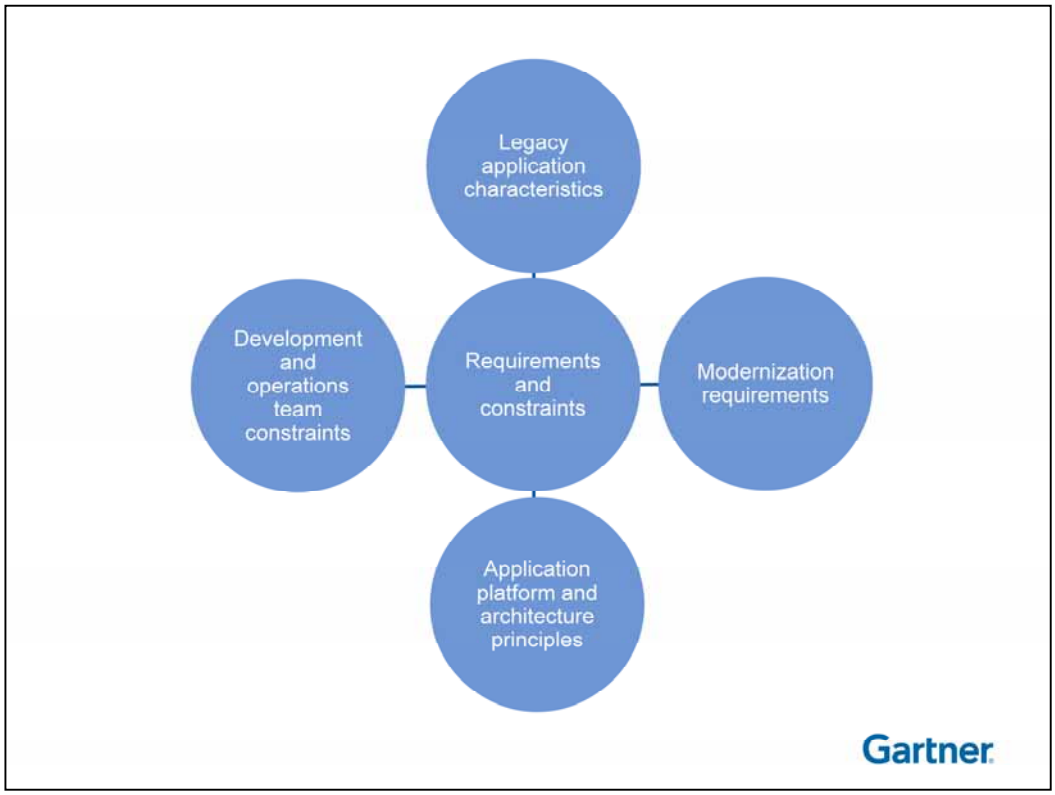

These phases show how we are going to make this cloud migration decision in a systematic way. First, identify migration goals. Second, gather requirements and constraints: Use the guidance outlined in the first two documents listed above. Third, filter out alternatives based on evaluation criteria: Use the "decision filters" to rule out alternatives. Forth, prioritize migration goals and match with remaining alternatives: Use the "alternatives assessment tool." And last, step is to assess provider quality of service and interaction cost implications: Use the QoS and interaction requirements list as a starting point for a migration cost model.



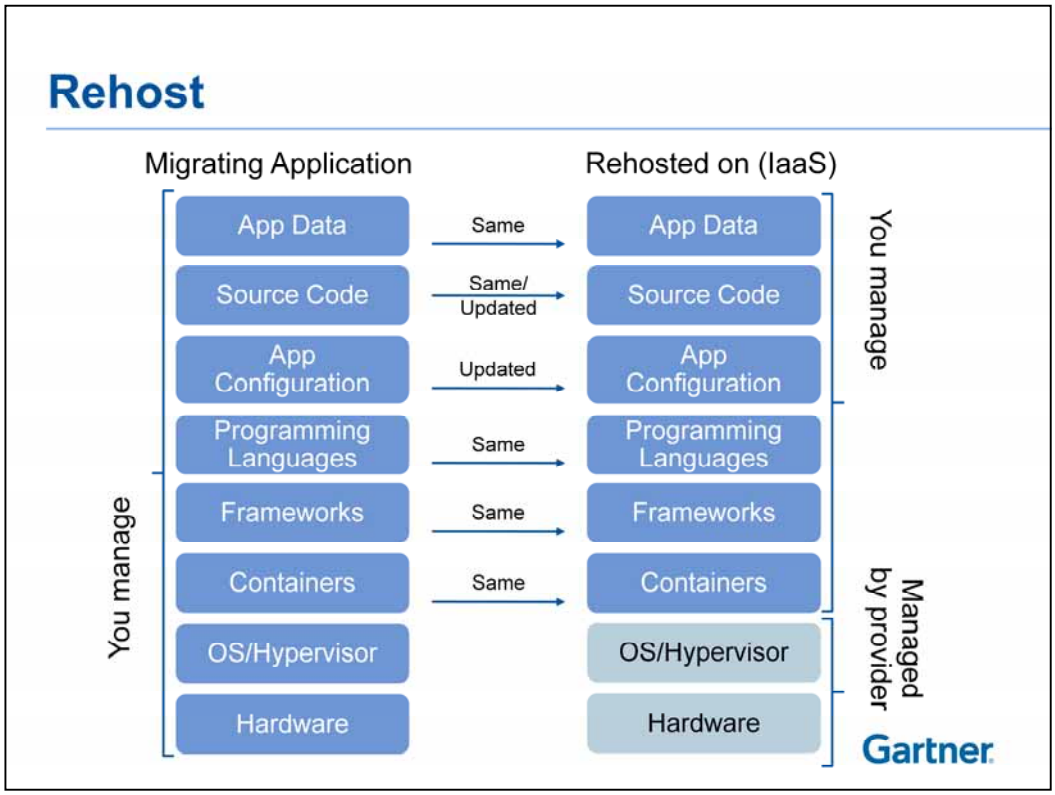

Your organization's objectives for cloud migration will direct alternative choices. Those goals are influenced by a number of often-conflicting factors. The company's cloud adoption and legacy modernization strategies will dictate relative priorities of the goals on the following slide.




The company's cloud adoption and legacy modernization strategies will dictate relative priorities of their application migration goals.




"Requirements and constraints" describes the various requirements that may apply to the specific situation. In some cases, these requirements may be in conflict (e.g., ease of use and flexibility versus high performance). Organizations may need to prioritize their requirements to make an appropriate decision, and tradeoffs and compromises are likely. They also reference the constraints on solutions (e.g., existing skill sets of development and operations staff). The organization must catalog specific architecture characteristics of the application that constrain the possible choices. "Legacy application characteristics" include application pattern, business logic complexity, data parallelism, licensing restrictions. The structured program for legacy software modernization (SWLegMod) and strategic software assessment framework (SSAF) frameworks will determine application modernization requirements. Decision makers should consider implementing modernization requirements such as scalability, security, integration requirements, code base value, and core capabilities. Architects considering application migrations will require varying levels of control over programming languages, frameworks, runtime infrastructure, and deployment topologies. The desire to reduce lock-in and domain specificity are crucial requirements here.




**Definition:** "Rehost" implies redeployment of the application to a different hardware environment and changing the application's infrastructure configuration. It is similar to deploying a Java application server on a Linux x86 server instead of Solaris SPARC-based server.

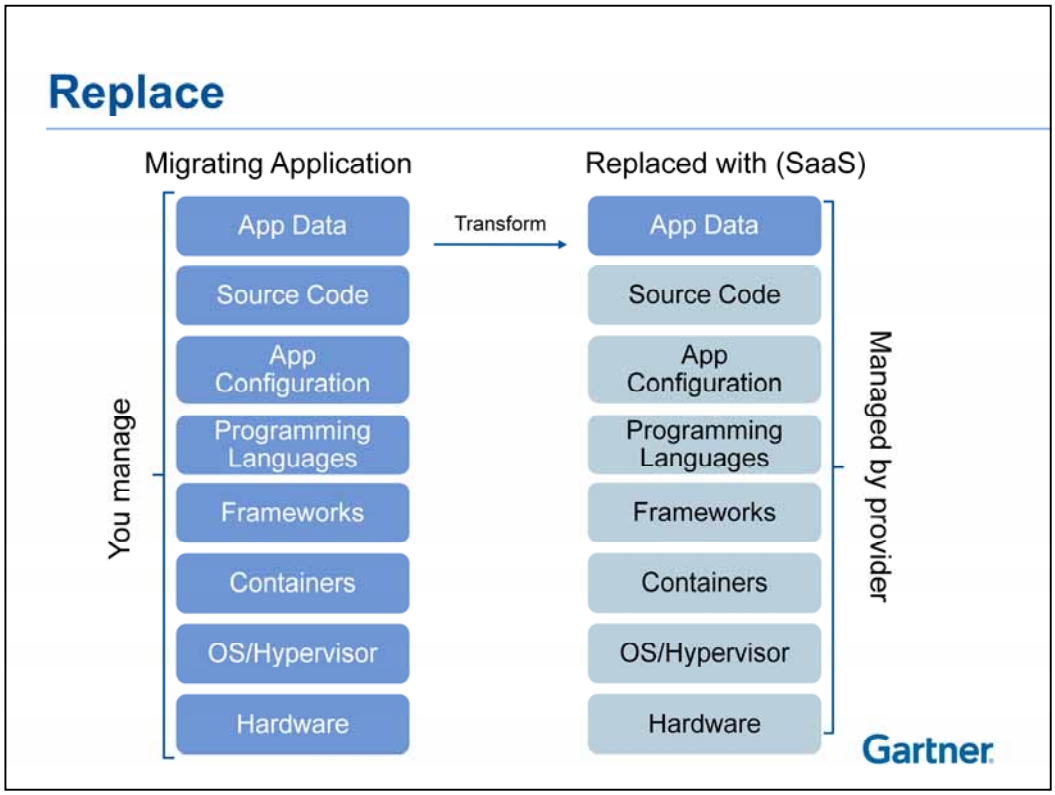
**What services am I consuming?** Virtual machines (VMs) as a service or operating systems (OSs) as a service, as well as block or volume storage as a service.

**Audience:** System administrators.

**Examples:** Deploying an existing Java EE container with a Java EE application on EC2 Linux instances from Amazon Web Services (AWS), backed by Elastic Block Store (EBS) for persistent VM images.

**Advantages:** Can work with systems where code modifications are impossible (e.g., COTS, code that cannot be rebuilt).

**Disadvantages:** If done without changes to application architecture scalability benefits may be lost. Assembling a complete application stack is necessary, choose from one off the peg, or engage a third-party provider to assemble one. VM image format and management API lock-in is a risk.




**Definition:** "Replace" means to discard an existing application (or set of applications) and use commercial software delivered as a service to satisfy those business requirements. Typically, existing data requires migration to the SaaS environment. Application data import/export is achieved with an API or configuration/admin console.

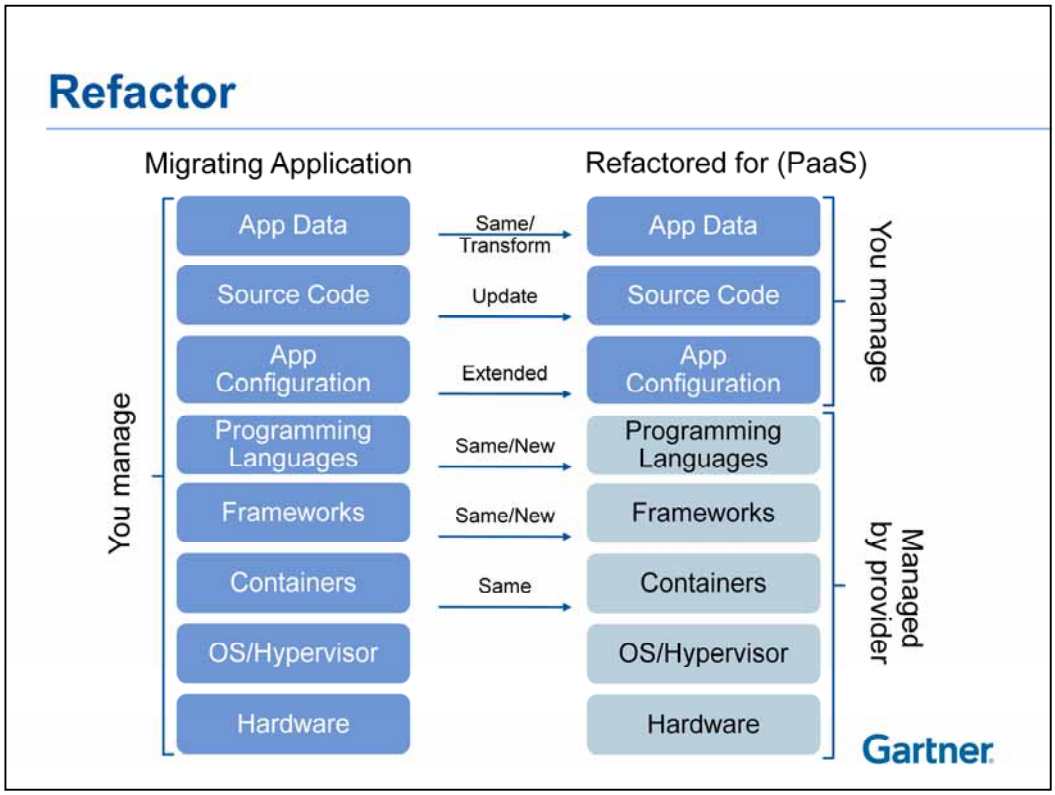
**What services am I consuming?** A complete, turnkey application solution (i.e., the IT organization does not build solution but may configure and integrate it). Users access SaaS via a user-centric interface, such as a Web browser or a mobile device.

**Audience:** Users — business users or IT. Developers or DBAs perform initial configuration.

**Examples:** Salesforce CRM, SugarCRM for customer records management, Workday for HR processes, Omniture for Web analytics, LiveMeeting for Web conferencing.

**Advantages:** Unlike custom-built software or single-tenant COTS, SaaS providers can monitor and leverage group behavior within a single tenant (one department or company), or across many tenants.

**Disadvantages:** Possible data lock-in; unless the plan includes a firm schedule for retiring replaced applications, adding more complexity to the applications portfolio landscape; incompatible process, policy or data models. Can be difficult to customize or reconfigure; and can be difficult to integrate with existing systems and processes, but no more and no less than integrating on premises applications.




**Definition:** "Refactor" describes running your applications (usually Web applications) on the cloud provider's infrastructure. You must make application code or configuration changes to connect the application to new infrastructure services. This is similar to linking in a new database driver, identity management system, or authentication module. It is also similar to moving a Java EE application from IBM WebSphere to Red Hat JBoss (the same type of container, some different frameworks and configuration). Necessary changes vary from none to widespread code changes to invoke new APIs. Refactor is "backward-compatible" PaaS. Existing programming models, languages and frameworks can be used and extended.

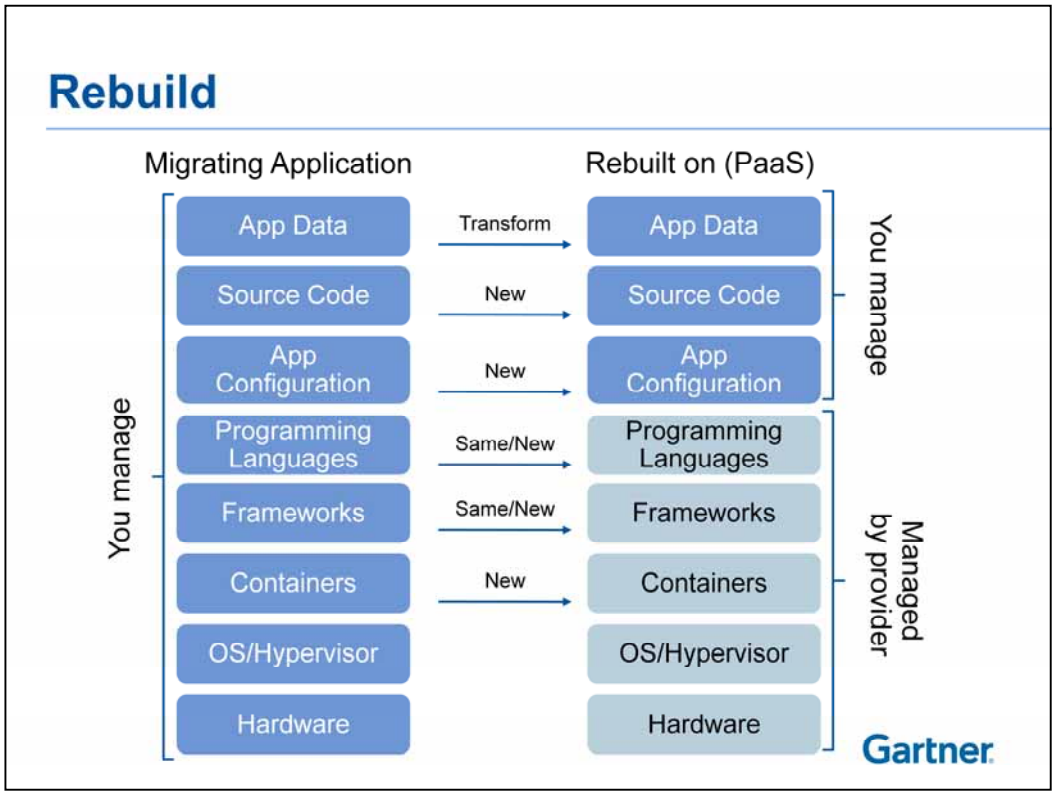
**What services am I consuming?** Providers' supply cloud-based frameworks and management tools that allow developers to take advantage of cloud characteristics of provider's infrastructure.

**Audience:** Professional developers.

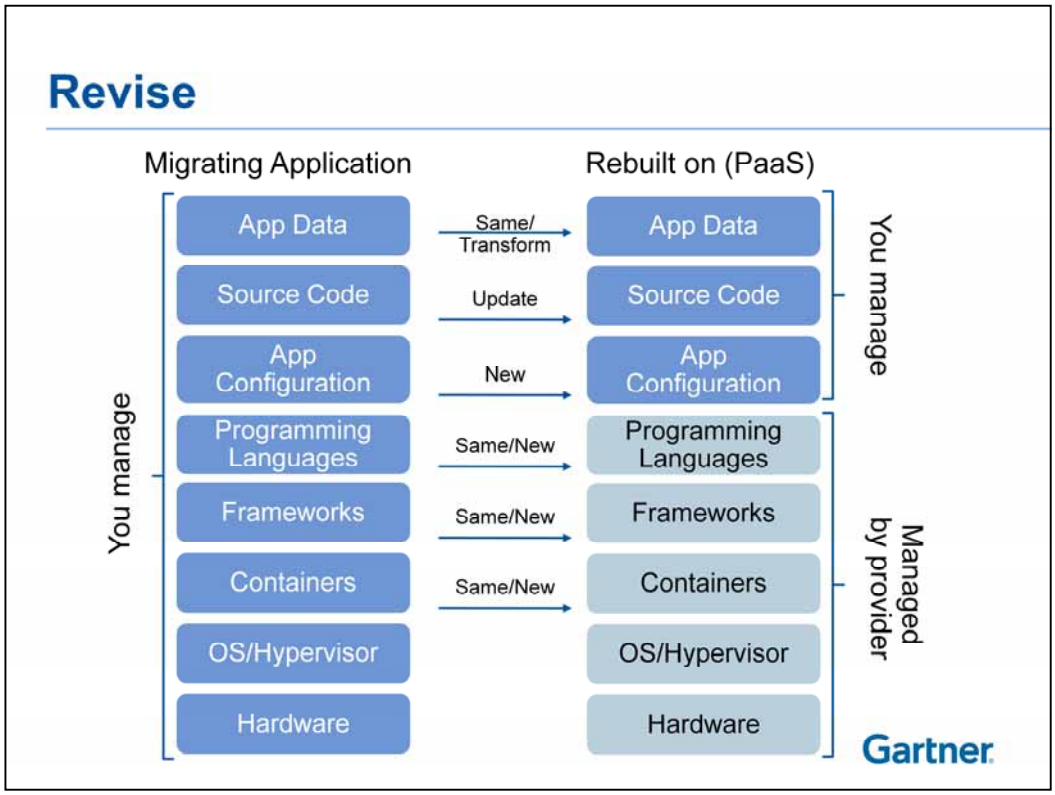
**Examples:** Deploying Ruby on Rails Web application to Heroku (including linking a monitoring service based on New Relic) deploying Active Server Pages for .NET (ASP.NET) application to Windows Azure.

**Advantages:** Reuses languages, frameworks and containers that developers have invested in, thus leveraging code the organization considers strategic.

**Disadvantages:** Immature PaaS offerings: Missing platform capabilities, transitive risk when PaaS providers build on IaaS providers, and framework lock-in.




**Definition:** "Rebuild" your solution on a provider's application platform, while discarding code for an existing application. Rebuild requires rearchitecting the application for a new container (e.g., from Java to .NET). "Forward-compatible" or incompatible PaaS tends to result. Not all existing programming models, frameworks, and languages will be retained.
**What services am I consuming?** An externally managed application platform to build and run applications.
**Audience:** Professional and "citizen" developers.
**Examples:** Building a force.com application for order management; Modernizing a C and FORTRAN financial risk calculation application by redesigning it in C#, then using Windows Azure platform libraries and tools to deploy it to Microsoft's cloud.
**Advantages:** Developer productivity is improved with tools that allow application templates and data models to be customized, metadata-driven engines, and communities that supply prebuilt components. Transparent automatic scalability. Allows non-professional developers the opportunity to develop and deploy simple applications into production. Multitenancy means the provider manages upgrades and patches. **Disadvantages:** Lock-in and PaaS immaturity are the primary disadvantages. Abandoning familiar programming languages and frameworks means that second sourcing strategies to mitigate lock-in risk may not work.




**Definition:** "Revise" means to modify or extend the existing codebase to support legacy modernization requirements, then use rehost or refactor options to deploy to the cloud. The scale of changes encompasses major revisions to add new functionality or to rearchitect the application for the cloud.

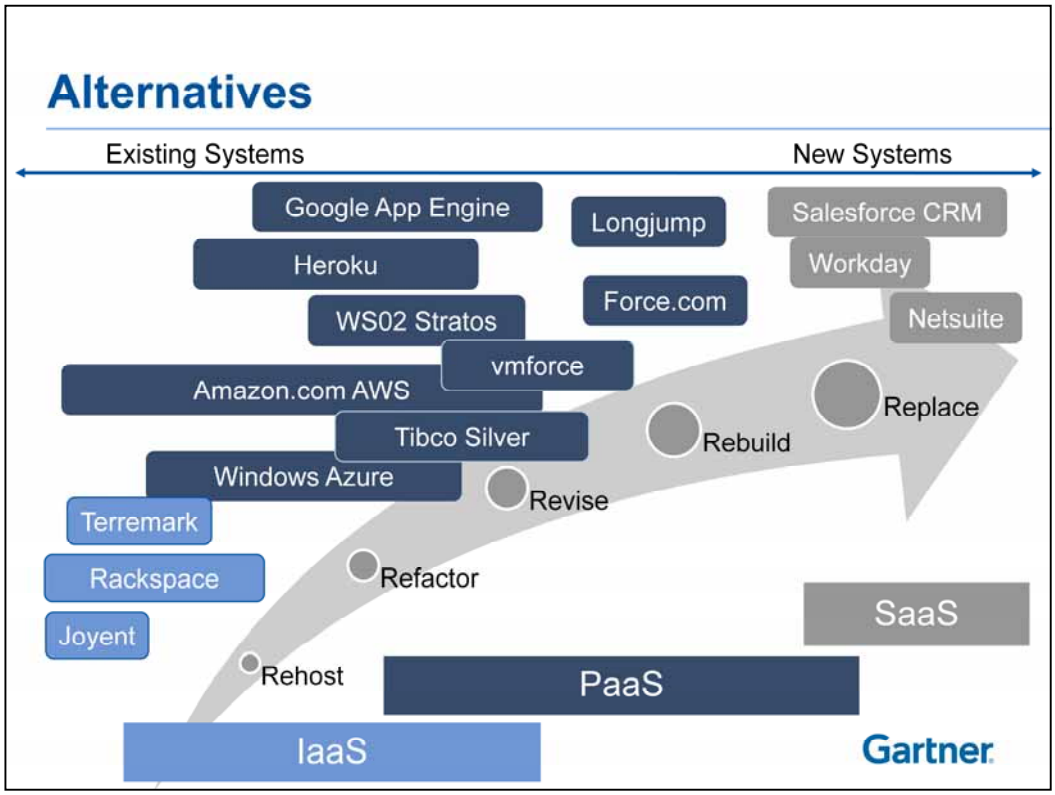
**What services am I consuming?** Either VMs as a service (IaaS) or frameworks and management tools that allow developers to take advantage of the cloud characteristics of a provider's infrastructure (PaaS)

**Audience:** Professional developers.

**Examples:** Redesigning a monolithic Java application, decomposing the functions into smaller parallel chunks, and then deploying on Rackspace Cloud Servers.

**Advantages:** Taking advantage of a valuable codebase by enhancing it to meet other cloud adoption or legacy modernization goals. Compared with the other migration alternatives, revise puts the organization in a position to optimize the application to leverage the cloud characteristics of providers' infrastructure.

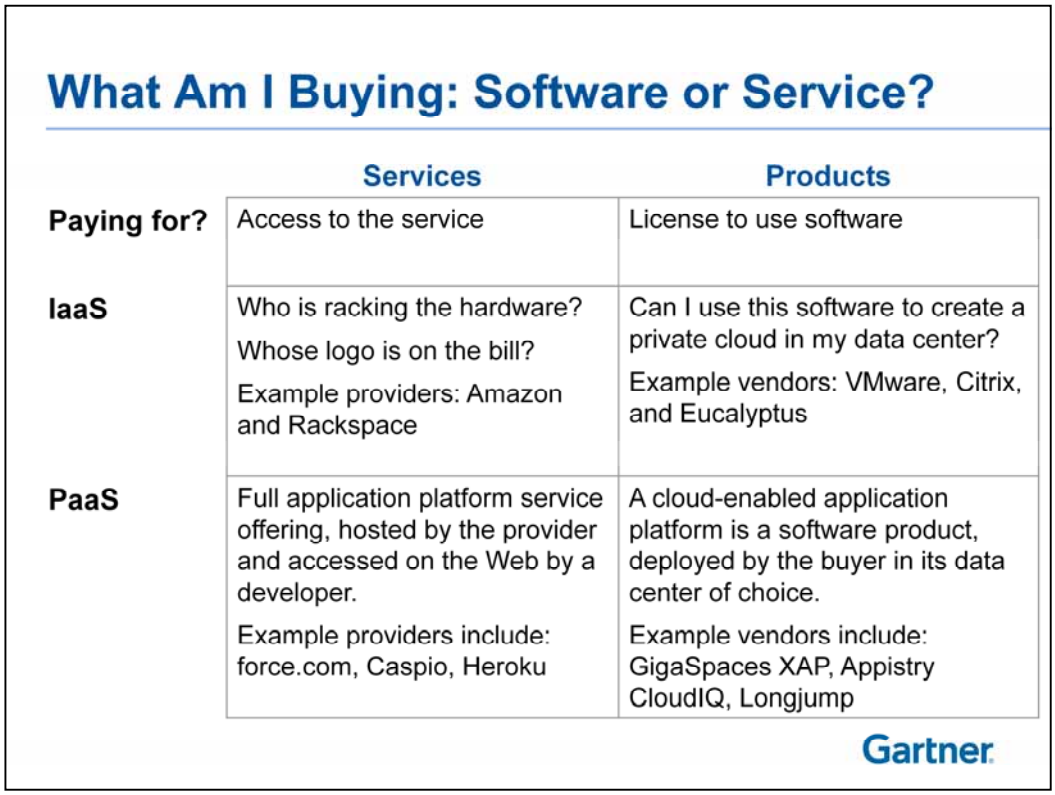
**Disadvantages:** The downside of revising an application is that kicking off a (possibly major) development project will require upfront expenses to mobilize a development team. Depending on the scale of the revision, revise is the option likely to take most time to deliver its capabilities. Immature PaaS offerings will also slow down progress.




The rehost, refactor, revise, rebuild, and replace cloud migration alternatives can be mapped to the cloud tiers. Cloud model tiers overlap with five migration options (i.e., not one-to-one mapping). The further you go up this spectrum the less you are considering working with existing applications.

Service providers crowd the cloud computing market. The above chart illustrates the approximate place on the spectrum of options for a selection of the best-known cloud providers. Because comparing provider offerings is challenging, this decision point presents migration options as a choice between five alternative actions. This decision point will not lead readers toward a choice of particular service provider, rather to an appropriate migration strategy.

Figure shows the rehost, refactor, revise, rebuild and replace cloud migration alternatives mapped to the cloud tiers and providers. Much overlap exists in current cloud providers offerings — and how they market themselves. Cloud model tiers overlap both with providers and the five migration options (i.e., not a one-to-one mapping).

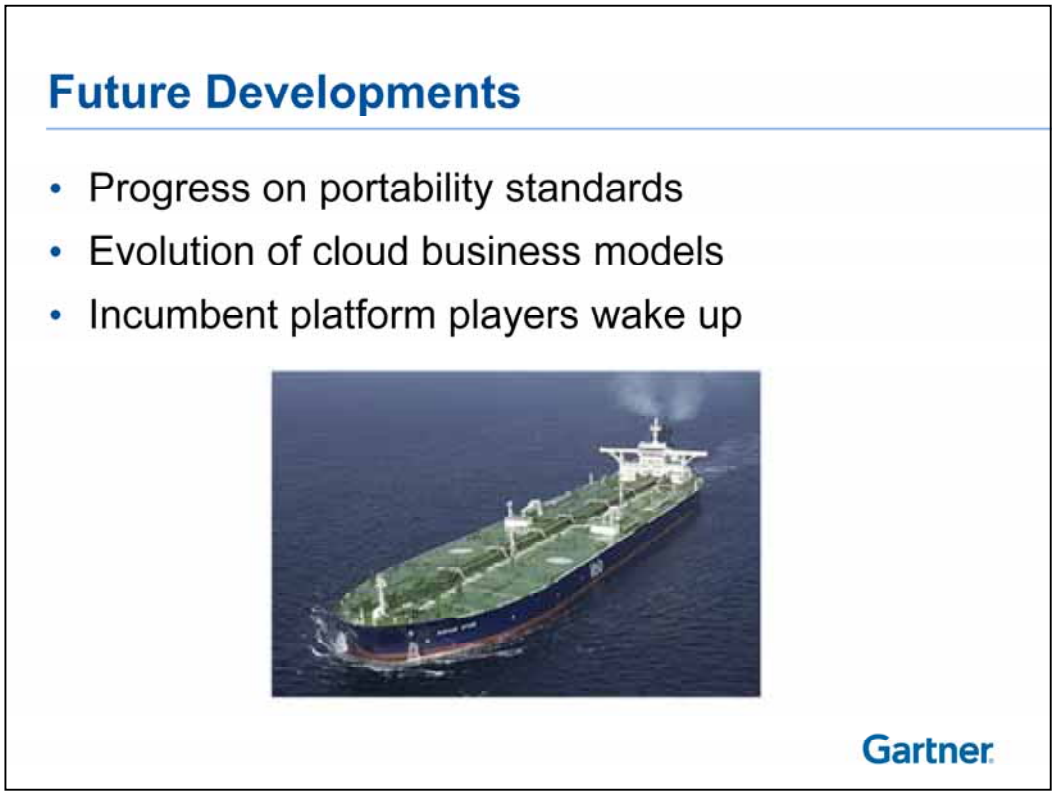


## What Am I Buying: Software or Service?

| | Services | Products |
|---|---|---|
| **Paying for?** | Access to the service | License to use software |
| **IaaS** | Who is racking the hardware?<br>Whose logo is on the bill?<br>Example providers: Amazon and Rackspace | Can I use this software to create a private cloud in my data center?<br>Example vendors: VMware, Citrix, and Eucalyptus |
| **PaaS** | Full application platform service offering, hosted by the provider and accessed on the Web by a developer.<br>Example providers include: force.com, Caspio, Heroku | A cloud-enabled application platform is a software product, deployed by the buyer in its data center of choice.<br>Example vendors include: GigaSpaces XAP, Appistry CloudIQ, Longjump |

Gartner

So far, we have made no distinction between public and private cloud options. Vendors have deliberately made this difficult to do. The distinction can be clearly made when considering what a purchaser is paying for: With products, you buy the software; with services: you pay for access to the service. This table summarizes this distinction, giving examples at IaaS and PaaS tiers. The IaaS tier is probably more familiar. For IaaS, someone else is racking the hardware. Their software creates new instances, configures them, supports you, and bills you per the amount consumed. Contrast that with software (and appliance) products that enable both internal IT organizations and external providers to build for creating private IaaS in a data center of your choice. One complexity with PaaS is a cloud enabled application platform (CEAP) may be provided as a VM image and hosted by a IaaS partner.

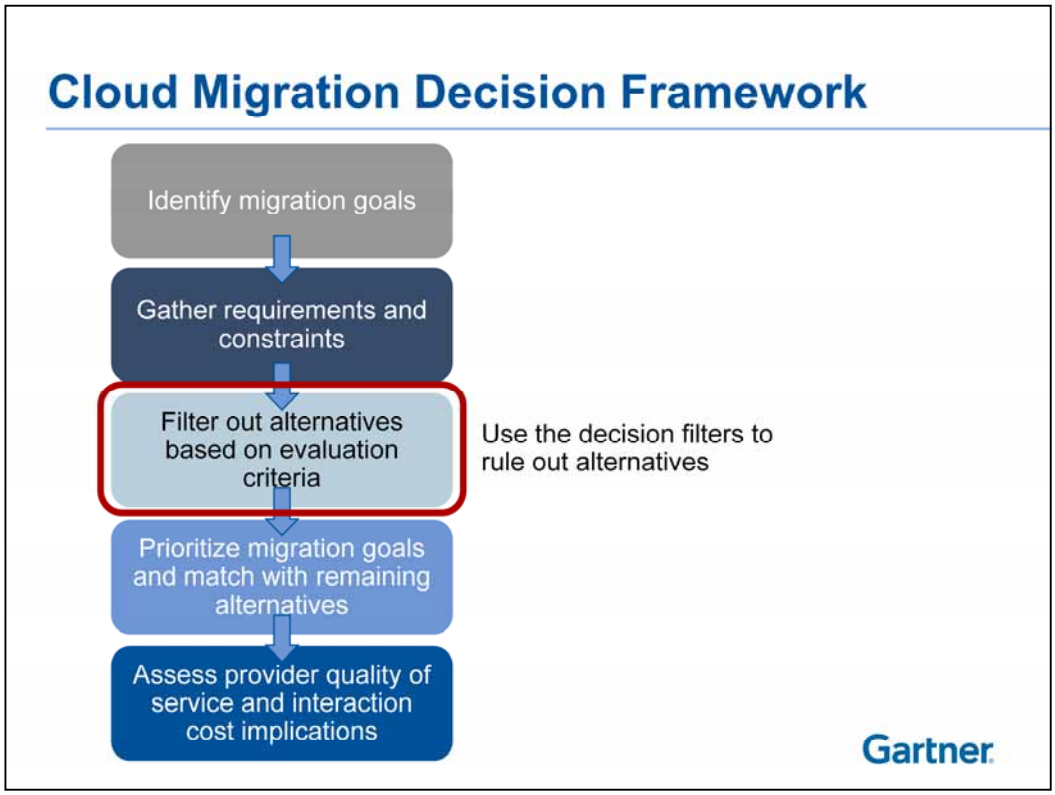



Emerging technologies don't quite qualify as current viable alternatives. Market or standardization issues that are likely to impact decisions within the next one to three years include:

**Progress on portability standards is required.** Portability is required to cover VM portability, (e.g., OVF, DTMF), open cloud initiatives, code portability (e.g., VMware "open PaaS"/Cloud Foundry), and data portability: (e.g., OData, Gdata), and Java EE 7.

**Cloud business models will evolve.** Today's offerings are prototypes: pricing, SLAs, T&Cs will all change. Today's models influence consumption patterns (e.g., things like data movement that were considered "free" [incorrectly]) are now explicitly priced. Private cloud offerings will continue to mature, offering many of the cloud characteristics without the regulatory and contractual risks.

**Incumbent platform players wake up.** Superplatform vendors have large turning circles. They have vested interests in keeping customers on their existing platforms. Two inevitable consequences are an increased interest in open source, and pure-play vendors will be acquired or squeezed out of the market.




**Before we dive into the formal decision making logic, a quick Recap of our assumptions**

IT decision makers above our pay-grade have already decided to migrate this application to cloud.

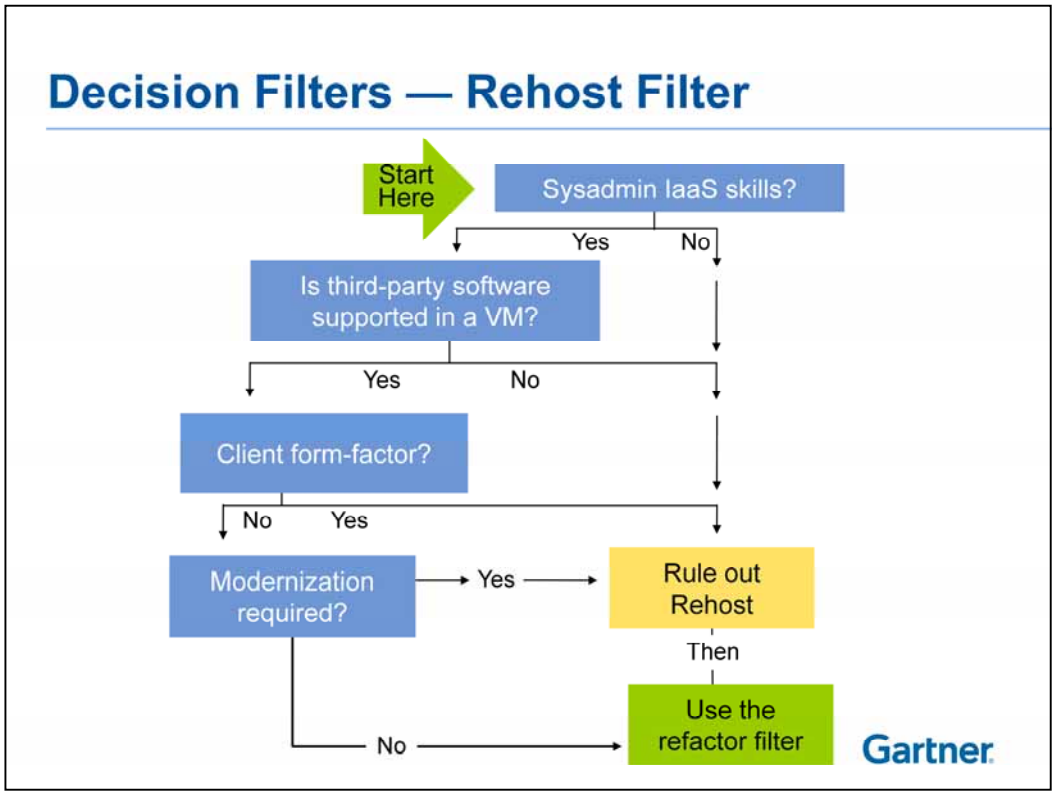
Quality of service, trust, user support, security and confidentiality risks will be manageable. Any outsourced option will add operational complexity to the application environment. These are the most pressing adoption barriers, but they are covered elsewhere.

Prior to using this decision tool, organizations should apply the APM guidance frameworks described in the "Architectural Context" section for each application and service. The frameworks will arm the organization with detailed information on the architecture characteristics, criticality and value of the application to the enterprise. Architects have already identified the characteristics of the application that makes it suitable for cloud migration, such as the following:

• The application is relatively isolated (there are few dependent applications).

• The traffic pattern is bursty — it is unlikely that the economics make sense if the app is in constant use 24/7, 365 days (e.g., a high "duty cycle").

For each of the alternatives, one or more factors will rule them out. Use the decision filters that follow to narrow the migration options that should be considered further for the applications under evaluation.




Conditions ruling out rehosting the application on IaaS concern operations team skills, modernization requirements, licensing restrictions and application form-factors as illustrated.

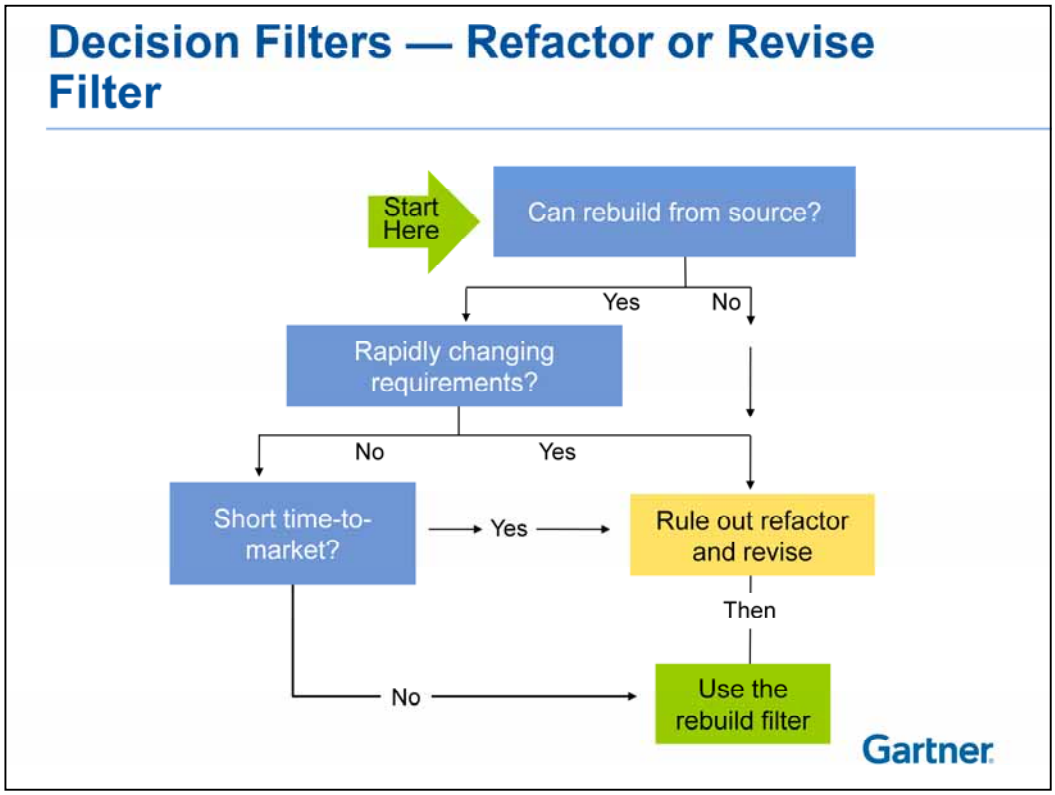
• Can the system administrator team assemble a VM image with a full application stack and configure hardware?

• Can third-party software be run in a VM, and is it supported by the vendor?

• Does the application have a fat-client form-factor?

• Does the application require modernization?

Rehost options can be ruled out when licensing compatibility for third-party software cannot be extended to a VM, or if the architecture doesn't fit (e.g., fat-client architecture).




Conditions that rule out refactoring the application for PaaS with new application services or revising it, concern development team skills time-to-market, and requirements volatility:

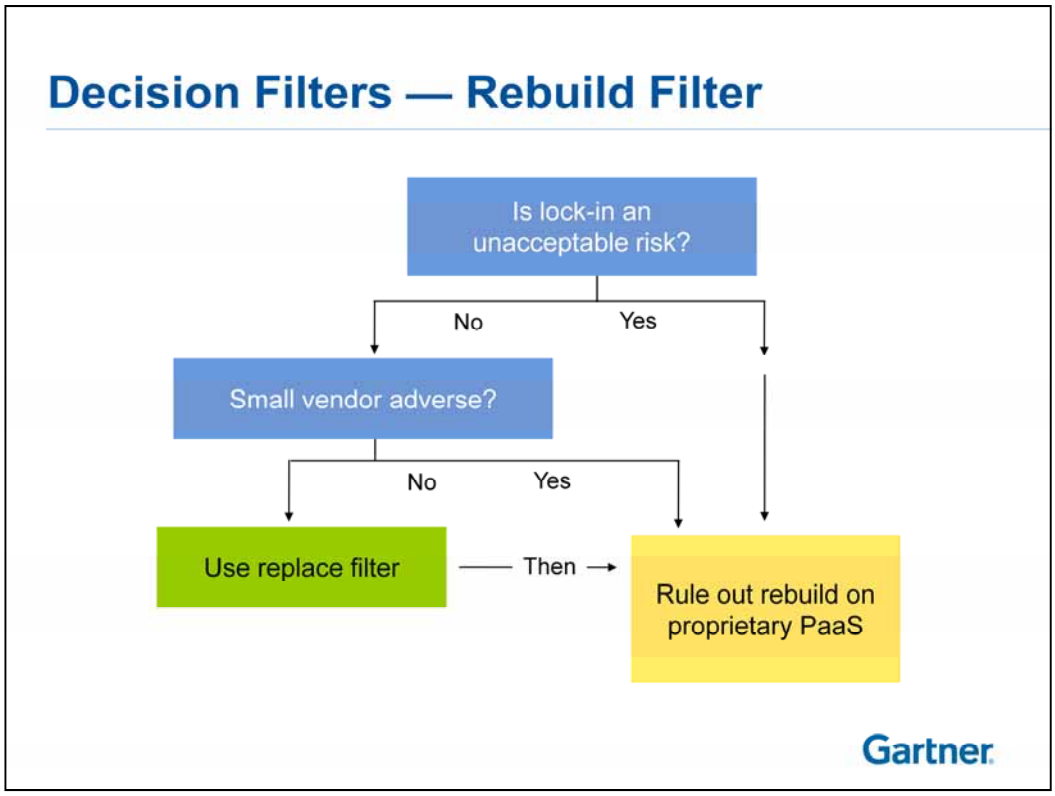
• Can the development team recompile and repackage the application from existing source code?

• Are the demands for this function or business requirements rapidly changing?

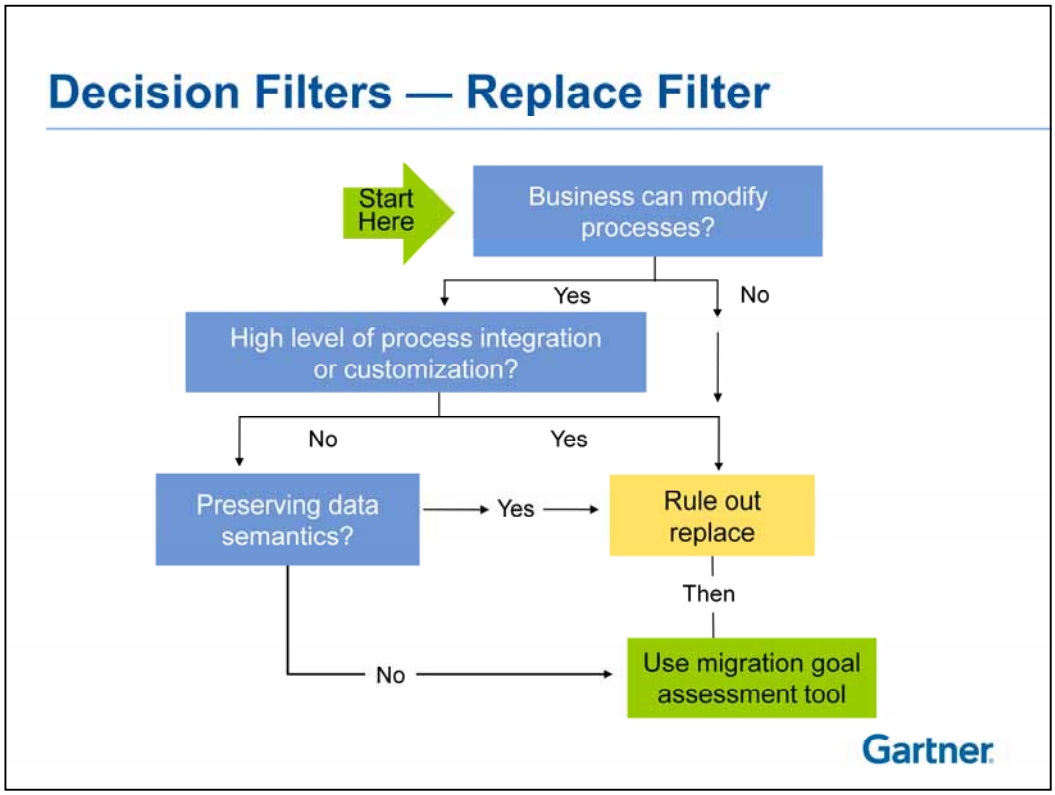
• Is the application required with an extremely short time-to-market deadline?

Refactor is not an appropriate option, and can be ruled out when an application cannot easily be adopted to meet provider's sandbox restrictions) or when skills or infrastructure to rebuild existing code are unavailable (e.g., build scripts or dependencies cannot be located and fixed, or developers cannot be assigned the task).

Revise should be ruled out if time to market is the overriding priority, project scope creep is an unacceptable risk, or the code base is no longer valuable to the organization.

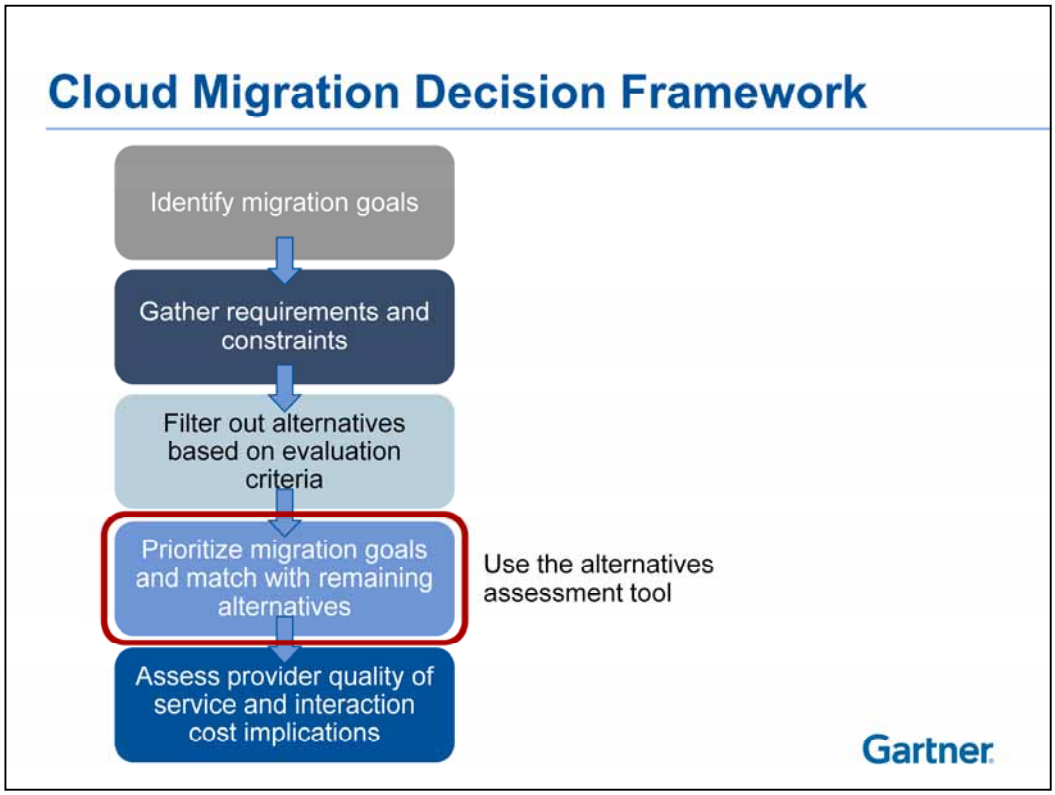



Rebuilding an application for a proprietary PaaS should be ruled out when code or framework lock-in is considered an unacceptably high risk, or if vendor sourcing strategy rules out smaller vendors. If rapid time-to-market for the application is paramount, rebuilding complex functionality for a new container is not a good choice.

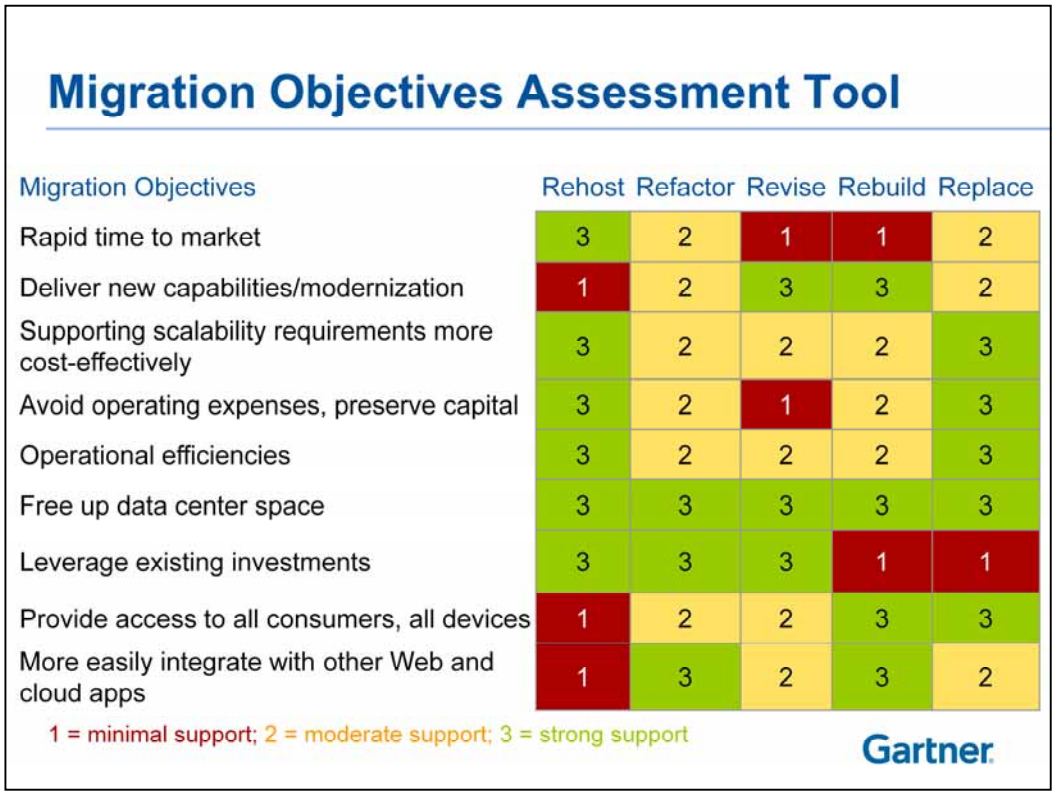



Conditions that rule out replacing the application with SaaS concern the level of competitive advantage afforded by the processes supported by the application, the business users' willingness to modify their business processes, the anticipated level of customization or integration with on-premises systems required, and the organization's attitude to data lock-in.

SaaS is best avoided when the organization is unsure about the level of integration and customization to the core feature set they will require. Enterprises sometimes overlook crucial configuration and ecosystem requirements, which limit the APIs and partner solutions used for integration work. SaaS can lose its long-term value if the organization has little or no flexibility over the data model, semantics, locations, sources, or security, or when data switching costs are exorbitant. Some SaaS vendors will charge enterprises for extracting data from their services. SaaS is also ruled out if the company is not willing or able to modify its business processes.

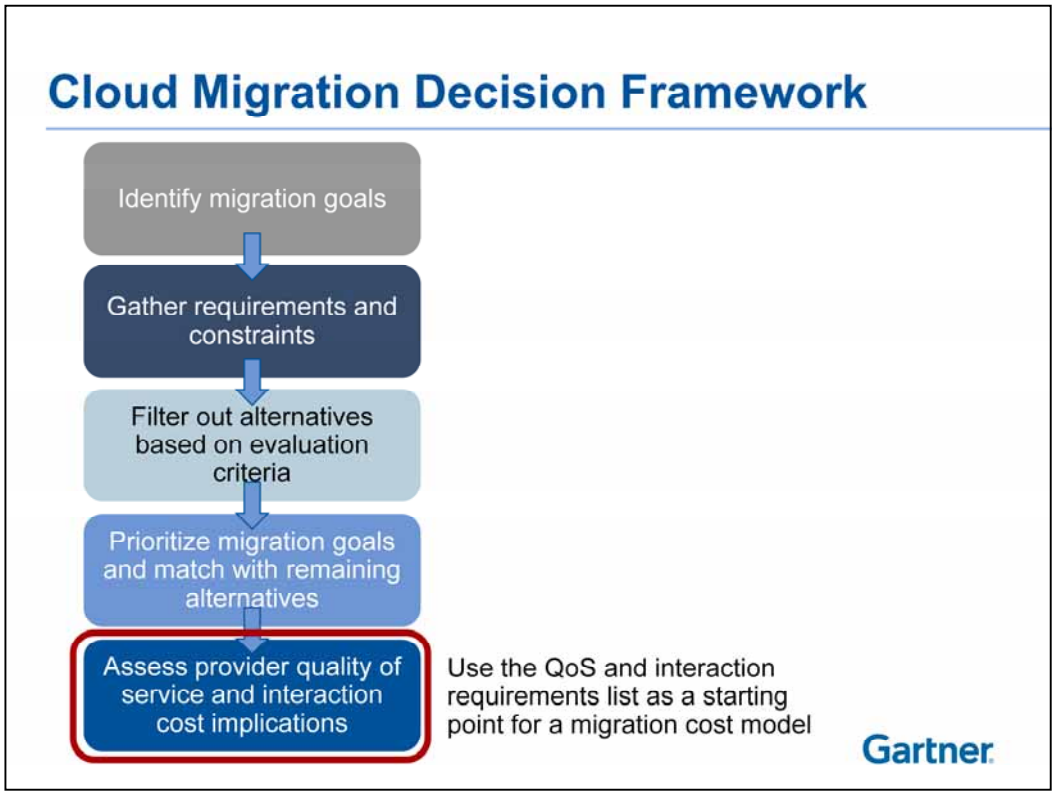



Use the migration goals assessment tool to identity the alternative that best fits the migration goals for this application.



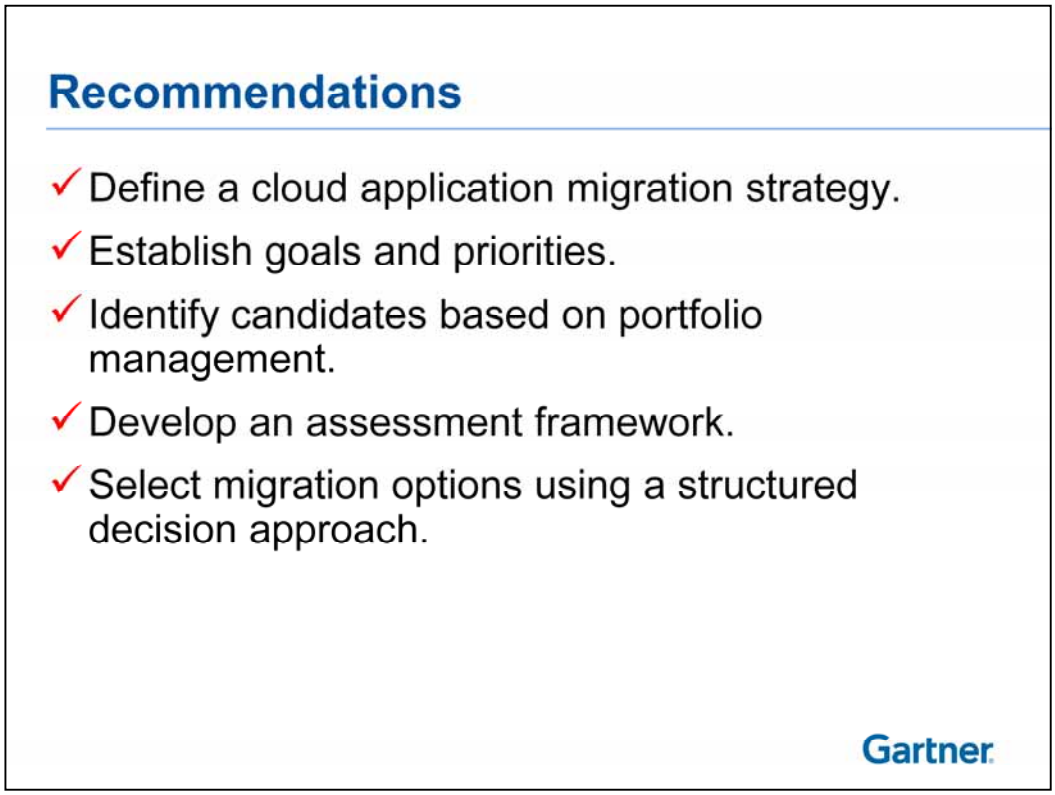

Migration objectives prioritization comes down to our clients asking themselves "why are we having this cloud migration conversation?" Part of the answer is asking the more specific question "What do we need for this application to give us a better outcome?" That outcome could be any combination of: We need it to be cheaper to run, we need it to have additional functionality, we need it to scale better than it does now, we need to get it off this OS or hardware that is EOL, or we need to support our field salespeople on their iPhones.

We use the migration goals assessment tool to identity the alternative that best fits the migration goals for this application. The tool allows the project team to assign weights to the migration goals and visually compare them to the capabilities supported by the alternatives, using Kiviat diagrams. The decision tool is implemented as an Excel spreadsheet, which is available as a separate download here.




The decision may be close, if more than one migration alternative closely matches the project's priorities. Choosing between the applicable options becomes a prioritization and a sourcing decision, requiring teams to analyze RFQs from providers, including factors such as vendor preference and cost. Architects should evaluate vendors on their capability to deliver the requirements discussed in the "Requirements and Constraints" section.

The following application interaction requirements will impact design (and execution cost) of a modernized or cloud-migrated application: Low-level interaction patterns (e.g., application "chattiness"), volume and the nature of data requirements, user migration requirements, latency constraints, I/O requirements, and ISV licensing requirements. Cloud providers may use a Services Provider License Agreement (SPLA) from ISVs (e.g., IBM or Microsoft,). In a SPLA, the price of the software license is included in the pay-as –you-go (PAYG) compute instance price. However, specific SPLA may not allow consumers to transfer internal software licenses to the cloud.



## Related Gartner Research

- **Building a Solid Cloud Adoption Strategy: Success by Design**
  Drue Reeves (G00203990)
- **Market Profile: Platform as a Service 2011**
  Richard Watson (G00209842)
- **2011 Cloud Computing Planning Guide: The Shift To Hybrid IT**
  Drue Reeves (G00210316)
- **Data Center Sourcing: Cloud, Host, Co-Lo, or Do It Yourself**
  Richard Jones (G00206662)
- **2011 Planning Guide: Application Platform**
  Anne Thomas Manes (G00209998)
- **Gartner Reference Architecture for Multitenancy**
  Yefim Natis et al. (G00205983)

For more information, stop by Gartner Solution Central or email us at solutioncentral@gartner.com.

Gartner