Decision Point for Choosing a Cloud Migration Strategy for Applications

Published 20 November 2018 - ID G00361356 - 62 min read

By Analysts Traverse Clayton

Supporting Initiative is Application Architecture and Platforms for Technical Professionals

When moving application workloads to cloud platforms, organizations must choose among five distinct strategies: rehost, revise, rearchitect, rebuild and replace. This research helps technical professionals weigh the trade-offs of each and select the right one for their goals and priorities.

Overview

Key Findings

- Not all applications will benefit from a migration to the cloud. This framework serves as only one part of an overall application portfolio assessment. “Retain” and “retire” are two additional strategies organizations should consider prior to deciding to migrate an application to the cloud.
- One Gartner survey shows that 75% of organizations plan on rearchitecting their custom-built applications for the cloud. This strategy requires a greater commitment from both development and operations, but offers them the improved scalability, resiliency and flexibility of the cloud.
- The rehost strategy is the most direct path to the cloud, but also the least beneficial. Rehost offers limited opportunity to improve the application’s runtime characteristics and provides no opportunity to redesign the underlying architecture to leverage cloud capabilities.
- The replace strategy, through SaaS, is the fastest way to transition to consuming a service with a cloud-native architecture. This option comes with the highest risk of cloud provider lock-in, especially if customization is also required.

Recommendations

Technical professionals responsible for migrating applications to cloud:

- Choose a migration strategy on an application-by-application basis rather than defining a strict policy for migration across the application portfolio. This gives you the flexibility to make the best decision for each application while adhering to your organization’s cloud strategy and principles.
- Create a detailed technical, operational and business profile of the application before choosing a migration strategy. Don’t take shortcuts based on assumptions — you’ll lose time and money.
- Choose a rearchitect or rebuild strategy when the lifetime of your application stretches beyond the foreseeable future and you have the budget, time and resources to pull it off. Don’t ignore the opportunity to use cloud migration as an impetus to modernize legacy applications.
- Maximize portability by choosing a platform that works across multiple IaaS providers, using containers as the unit of deployment and implementing a cloud-agnostic CI/CD toolchain.

Decision Point Question

How do we choose a cloud migration strategy for our application?

Organizations of all sizes have cloud strategies, and are selecting and prioritizing applications for cloud migration. The key remaining question is, “What is the best way to move them?”

The decision in scope for this Decision Point is not only an issue of migration, but also one of optimization. Another way to state the question is:

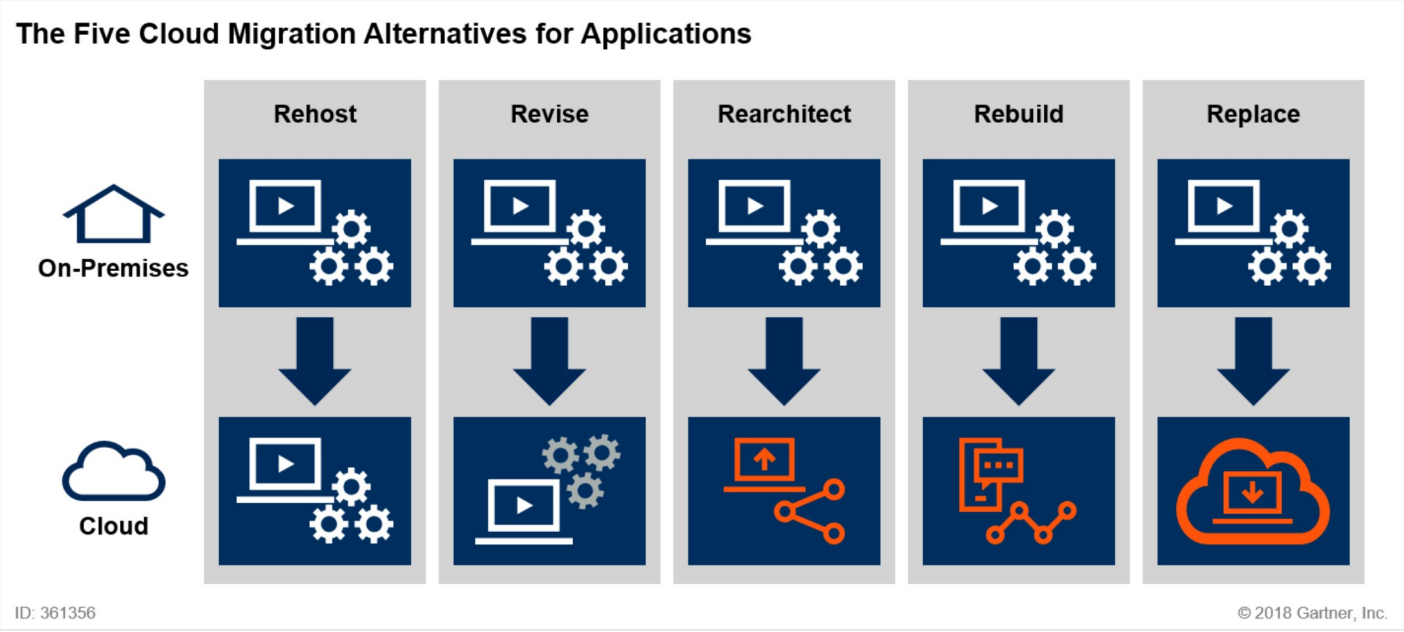
> **"Which cloud platform and migration techniques offer the chance to optimize the application's contribution to deliver the desired business outcomes and IT goals?"**

The starting point for the decision is either a COTS or a custom-developed application. However, these scenarios are not that cut-and-dried. For instance, a custom-developed application is not always written in 3GL. There are scenarios where a COTS application has extensions in 3GL, 4GL, a proprietary language or all of these. Typically, there is a direct SaaS or xPaaS equivalent for that extension. Another example is a legacy workflow application that can be migrated to an updated bpmPaaS platform.

Decision Overview

Migrating an application to the cloud requires you to consider myriad factors in order to select the best fit from five alternatives (see Figure 1).

Figure 1. Each Cloud Migration Alternative Requires New Platforms, Time and Resources



Source: Gartner (November 2018)

- **Rehost:** In this alternative, you "lift and shift" your application from its current physical or virtual environment onto a cloud infrastructure as a service (IaaS) or container as a service (CaaS) platform. While doing so, you avoid any modifications to the system other than those required to adapt to the hosting environment itself. Although this is the least-ambitious and least-beneficial alternative, it is also the quickest to implement to "get to the cloud." However, the application will not be cloud-native.
- **Revise:** In this alternative, you modify your application so that it can begin to take advantage of cloud capabilities for elasticity and minimized resource use. This strategy is focused on reducing operational overhead through the use of managed cloud services (e.g., database PaaS). You might opt to use IaaS or CaaS for the application, but at this stage, you could also choose certain platform as a service (PaaS) capabilities.

  Revise is primarily about tinkering around the edges of the application to optimize it while leaving the source code primarily untouched. For example, you may switch a self-managed Oracle Database to Amazon RDS for PostgreSQL and then modify the application configuration connection string to point to the RDS instance. In general, you are optimizing the infrastructure and backing services of the application, which requires reconfiguring the application, the system and the application dependencies while implementing no or very minor changes to the code.

- **Rearchitect:** In this alternative, you materially alter the application so that you can shift it to a cloud-optimized architecture, making heavy use of cloud-native capabilities. The rearchitect alternative is an in-depth undertaking requiring changes to culture, technology, people, processes and platforms. Rearchitect is appropriate for migration projects targeting cloud-native application platforms with a choice of hcaPaaS, fPaaS and serverless.

- **Rebuild:** The rebuild alternative proposes that you start from scratch, jettisoning existing code that you've accumulated over the years. This alternative prescribes cloud-native application platforms, including hcaPaaS, hpaPaaS, fPaaS, and serverless. Why would you want to do this? There are multiple reasons, including:
  - Enable professional developers to focus their efforts on delivering code
  - Enable nondevelopers to deliver applications through point-and-click styles of development
  - Write-off technical debt
  - Simplify the application (which might then enable you to use high-productivity tools)
  - Adopt new architecture paradigms (e.g., EDA) that are better-suited for the current business requirements
- **Replace:** In this alternative, you're giving up on either a COTS software package or a custom application altogether, and instead opting for commodity SaaS. While this is a common pattern for replacement of aging on-premises COTS solutions, you shouldn't immediately rule it out for your custom software applications. SaaS has low barriers to entry in terms of technical and capital constraints, making it entirely possible that SaaS alternatives for your application are available off the shelf.

This Decision Point makes crucial assumptions:

- **This is not a decision of *whether* to migrate applications to the cloud.** Your organization has already made that choice. This is a "how do we do it" execution decision, not a "should we do it" strategy decision. There are valid strategies for retaining or retiring the application that need to be considered prior to making the choice to migrate it to cloud.
- **Your organization has already analyzed and can manage cloud operations.** These include security, network, availability and trust issues that accompany the move to cloud computing.
- **You wish to move an application that already exists in some form.** It could be custom in-house software or commercial off-the-shelf (COTS) software. This is not a "greenfield" project. Several of the alternatives involve abandoning existing systems, but you already have a baseline of the requirements for the applications.
- **You are moving the application for production use.** You're not moving development and/or testing to the cloud while maintaining production in a legacy hosting arrangement.
- **Your organization has already assessed the applicationin question.** You have determined your goals for the migration.

Table 1 outlines some of the key differences affecting scope, complexity and level of effort introduced by each of the alternatives. The balance of this document advises you how to make the decision.

Table 1: Comparison of the Migration Strategies

↓	Rehost ↓	Revise ↓	Rearchitect ↓	Rebuild ↓	Replace ↓
Programming Language	No change	No change	No change	No change/new	N/A
Source Code	No change	Lightly updated	Updated/new	New	N/A
Application Configuration/Metadata	No change/updated	Extended	Extended	New	N/A
System Configuration	No change/updated	Updated/new	Transformed	New	N/A
Build, Packaging and Configuration Scripts	No change/new	Updated/new	Transformed	New	N/A
Frameworks	No change	No change	No change/new	New	N/A
Runtime Environment	No change	No change	No change/new	New	N/A
Application Data	No change	No change/transformed	No change/transformed	Transformed	Transformed

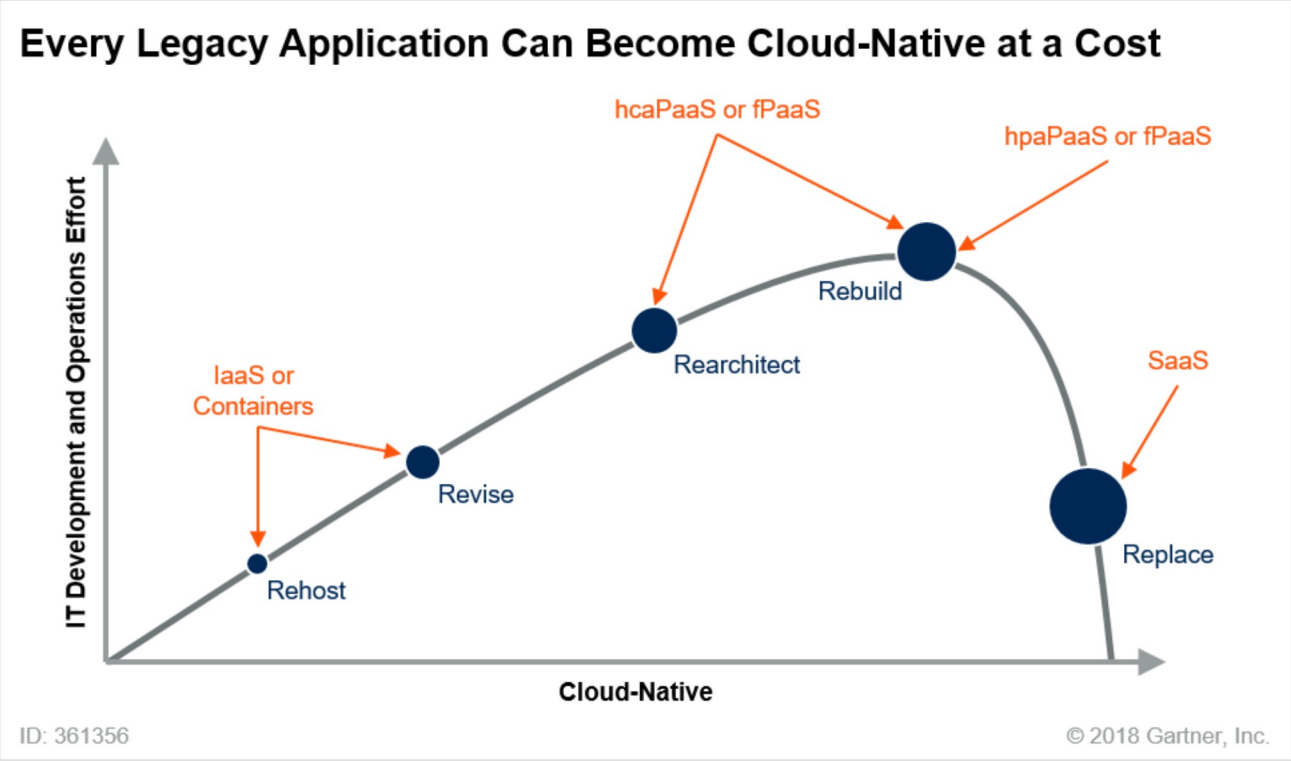
↓	Rehost ↓	Revise ↓	Rearchitect ↓	Rebuild ↓	Replace ↓
Hosting Hardware	New	New	New	New	New

N/A = not applicable

Source: Gartner (November 2018)

The scope, complexity and level of effort associated with each migration strategy vary, but roughly follow the curve illustrated in Figure 2 below. The more investment you put into changing the application, the more benefits the application can reap from cloud computing. Not every application needs to be rebuilt from the ground up, and ROI goals vary from application to application. In addition, the platform you are targeting will impact the level of effort and complexity of the migration strategy you select.

Figure 2. Scope, Complexity and Effort Associated With the Five Migration Strategies

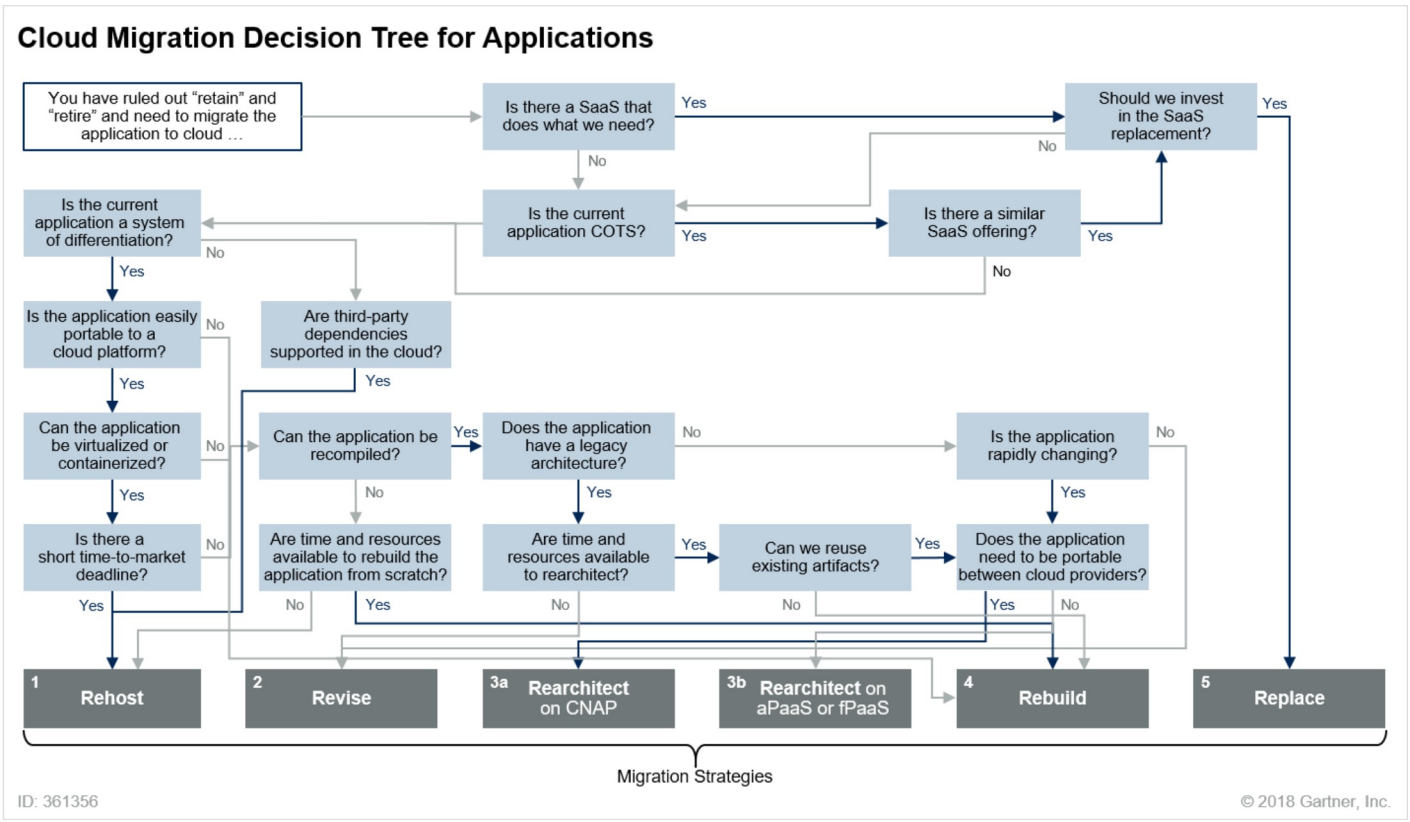


Source: Gartner (November 2018)

Decision Tool

The decision tool, a spreadsheet available for download with this research, guides you to a recommendation for a cloud migration strategy for a given application. Different applications may require different strategies. The recommendation is driven by your understanding and selection of the migration goals, application characteristics, development and operations skills, and other considerations discussed in the Requirements and Constraints section of this document. You will need to apply some of these considerations yourself to set the appropriate filters in the decision tool. Use Figure 3 and Table 2 to help you. Only attempt to use the decision tool after reading and thoroughly digesting the rest of this Decision Point research.

Figure 3. Filtering the Options in the Decision Tool



CNAP = cloud-native application platform

Source: Gartner (November 2018)

Table 2: Explanation of Flowchart Labels

Flowchart Label ↓	Full Question ↓
Is there a SaaS that does what we need?	■ Whether you have a custom-built or COTS application, the first question to ask is whether there is a SaaS offering that can replace your application.
Should we invest in the SaaS replacement?	■ If there is a SaaS replacement available, you will need to determine whether investing in it is appropriate.
Is the current application COTS?	■ Is this a custom-developed or COTS application?
Is the current application a system of differentiation?	■ Does this application contribute directly to revenue streams as a core differentiation from competitors in the market? ■ Do teams require agility to adopt new technologies and innovate to compete?
Is there a similar SaaS offering?	■ Does either the existing COTS vendor or another vendor in the market have a similar offering delivered as a SaaS?
Is the application easily portable to a cloud platform?	■ For example, is anything other than a standard x86 hardware architecture required for this application? ■ Does your application require UDP multicast?

Flowchart Label ↓	Full Question ↓
Are third-party dependencies supported in the cloud?	■ Does the application have one or more third-party dependencies that are all supported by the cloud provider? If not, select rehost as an alternative option. ■ Can third-party software be run in a VM, and is it supported by the vendor? ■ Will the application have vendor support for virtualization?
Can the application be virtualized or containerized?	■ Can the system administration team assemble a VM image with a full application stack and configure the hardware? ■ Can the system administration team assemble the application into Docker containers?
Does the application have a legacy architecture?	■ Does the application have a legacy architecture not built for the web? For example, does the application have a “fat client” form factor? ■ Is the front end delivered as a desktop application?
Can we reuse existing artifacts?	■ Is there code that can be reused, and is it worth investing in refactoring?
Are time and resources available to rearchitect?	■ Do you have the time, resources and appetite to rearchitect the application?
Can the application be recompiled?	■ Can the development team recompile and repackage the application from source code?
Are time and resources available to rebuild the application from scratch?	■ Do you have the time, resources and appetite to rebuild the application from the ground up?
Is the application rapidly changing?	■ Are the demands for this application or its business requirements rapidly changing?
Is there a short time-to-market deadline?	■ Does the application have an extremely short time-to-market deadline?
Does the application need to be portable between cloud providers?	■ Is code or framework lock-in an unacceptable risk? ■ Does the application need to be portable across cloud providers?

Source: Gartner (November 2018)

A Note on Retain and Retire

This research makes the critical assumption that you have already decided to migrate your application to the cloud. With that said, migrating your application to the cloud is not always the right approach (regardless of what the cloud providers tell you). You should consider retain and retire strategies prior to deciding to migrate an application to the cloud.

The Alternatives section ahead details each cloud migration strategy, including its definition, appropriate cloud platforms, the audience and examples. Selecting the appropriate migration strategy is dependent upon the principles, requirements and constraints of your company. Each application you migrate has specific legacy characteristics, modernization requirements, costs to consider, and constraints around personnel, including development and operational skills.

Principles, Requirements and Constraints

When you choose a migration option, you operate under certain principles. These principles are unique to your organization. Make sure you fully understand where your organization stands, because these principles directly impact the criteria you value as “most important” in the decision of where to migrate an application.

Principles

The following architectural principles impact the cloud migration execution decision:

- **Degree of autonomy:** Because of pay-as-you-go (PAYG), metered billing and self-service characteristics, application migration to cloud providers can support a team, a departmental or a business unit autonomy strategy. This principle impacts the migration decision, because some alternatives can be used more opportunistically, without central IT support. The higher the degree of autonomy, the more difficult it is to change traditional IT culture to adapt to this principle. An example of this is using high-productivity application platforms (see "Architecting Low-Code Cloud Applications With High-Productivity aPaaS" (https://www.gartner.com/document/code/350052?ref=grbody&refval=3893681)).
- **Closed versus open solutions:** Organizations favoring solutions based on open standards (e.g., Kubernetes, Docker, Cloud Foundry and Apache Kafka) have a reduced choice of services from their cloud provider, because many cloud provider services are closed/proprietary. Each choice of open versus closed application component presents switching costs related to the services offered by the cloud provider. Each component of an application needs to be considered in the context of closed versus open solutions. These components include virtualization, containers, code, application runtime and data.
- **Single vendor versus best-of-breed:** The organization's sourcing principles may dictate the use of single vendors over best-of-breed vendors and influence the decision when multiple migration options meet an application's requirements. Incumbent platform vendors may not have offerings corresponding to all alternatives.
- **Platform domain specificity:** Cloud platforms can be domain agnostic (or "generic") and used to build applications for any domain (e.g., Microsoft Azure or Amazon Web Services). Alternatively, they may have a specific domain bias, or "flavor" (e.g., Salesforce or ServiceNow). Applications deployed on a domain-specific platform extend the domain-based features or templates, and are often described as "adjacent applications." For example, Salesforce Lightning Platform applications can easily extend and leverage Salesforce CRM data and functionality (see "Choosing a Platform for Building and Extending Salesforce Sales Cloud Applications" (https://www.gartner.com/document/code/327175?ref=grbody&refval=3893681)). Organizations that have already made investments in a provider's services, such as G Suite for Business or Salesforce CRM, may favor creating adjacent applications to extend their investments (see "2019 Planning Guide for CRM and Customer Experience" (https://www.gartner.com/document/code/361504?ref=grbody&refval=3893681)).
- **Lock-in, portability, multicloud and interoperability:** Portability would not be a consideration if an architect could migrate an application, along with its data, to a cloud provider and leave it there for its lifetime. In reality, ignoring platform lock-in is risky, so architects must consider a multisourcing strategy for multiple facets of cloud portability:
  - VM portability covers VM image format and management APIs for hardware IaaS.
  - Container portability covers the container image format (e.g., Docker and CRI-O) and the container orchestrator (e.g., Kubernetes) delivered as a CaaS (e.g., Google Kubernetes Engine [GKE], Amazon EKS and Azure Kubernetes Service [AKS]).
  - Application code, runtime and platform portability determines how easy it is to take an application deployed on one PaaS offering and deploy it to an alternative platform (i.e., another PaaS offering or an on-premises solution).
  - Data portability measures whether an organization can easily retrieve its data and move it to an alternative (cloud-based or in-house) application in a reasonable time, at a reasonable cost and with no loss in data semantics.
  - For more information about designing for portability, see "A Guidance Framework for Architecting Portable Cloud and Multicloud Applications" (https://www.gartner.com/document/code/302796?ref=grbody&refval=3893681) and "Assessing the Strengths and Weaknesses of High-Value IaaS and PaaS Multicloud Use Cases." (https://www.gartner.com/document/code/336677?ref=grbody&refval=3893681)

  Cloud providers are not entirely interchangeable. You can select technologies and design solutions with portability being the highest priority, but there will continue to be significant differences between cloud providers. Every solution you build in a cloud provider will have specific dependencies on that provider's APIs, services and tooling. You must consider the migration of all of these dependencies when pondering a move to an alternative cloud provider. As part of solution planning, estimate what a migration to a new provider involves, and include that as part of your exit strategy (see "Designing a Public Cloud Exit Strategy" (https://www.gartner.com/document/code/355037?ref=grbody&refval=3893681)).

Requirements and Constraints

Your decision on how to move a given application to the cloud depends on more than just the core principles upon which you manage the middleware and application portfolio in the organization. The decision also hinges on the specific characteristics that apply to the individual application under your consideration. You should consider how your organization values the following migration goals and priorities, legacy application characteristics, modernization requirements, development and operations skills constraints, and migration cost factors.

Migration Goals and Priorities

Your organization's objectives for cloud migration will direct the choice of an alternative. A number of often-conflicting factors influence those goals.

Note that moving applications to the cloud won't automatically fix bad processes that hamper operational efficiency. Migrating to the cloud takes work, and there is no magic formula that does it for you. You have to put in the work to get the benefits of migration.

Your cloud adoption and legacy modernization strategies will dictate the relative priorities of the following goals (see Figure 4):

Figure 4. Cloud Migration Goals

Source: Gartner (November 2018)

- **Deliver rapidly to market:** Organizations may face deadlines related to infrastructure support end of life, regulatory compliance or seasonal business opportunities, requiring a rapid application migration. Teams will favor alternatives that allow migration with minimal architecture impact and without kicking off a development project.

  For migration projects that include development of new functionality, reducing the volume of code needed to express a capability is another way to make development teams more productive. Cloud platforms vary in the degree that they provide or support productivity features, such as customizable application templates and data models, metadata-driven engines, and prebuilt components. (Examples of prebuilt components include tools like data analytics, application performance monitoring and data stream processing, supplied either directly or via a community application store.)

  Deliver new application capabilities: Legacy-modernization strategy may dictate that the application admits new functional requirements. More maintainable software drives agility so that systems can accommodate rapid, cost-effective changes to functional and nonfunctional system requirements (e.g., capacity). More usable software reduces the cost and time to recoup implementation investment.

  Another objective for revision work is optimizing the application architecture for cloud infrastructure. For example, if the application is not scalable

on its current infrastructure, it won't scale any better in the cloud without significant revision. There are also additional services available to application developers for big data, AI/ML, IoT, analytics and cutting-edge capabilities that would be otherwise difficult to take advantage of with current on-premises hardware and organizational constraints.

- **Scale elastically:** Organizations may require better (or more cost-effective) resources to meet variable demand, such as peak seasons or end-of-month processing. This capacity either is not available (due to capital constraints) or can only be made available in the traditional data center more slowly than customer demand requires.
- **Limit operating costs:** Operating costs include head count (both development and operations) and infrastructure costs (including data center power and cooling). Instead of buying compute and storage upfront, organizations switch to a pay-as-you-go model with cloud computing, ensuring that they don't buy any capacity that they don't need. This consumption-based pricing also enables IT organizations to perform better operational budget forecasting.
- **Preserve capital:** By shifting to an operating-expense-driven model of compute and storage consumption, the organization can preserve capital that it otherwise would have thrown into expensive data center investments (and associated depreciation). Improving your budgeting processes in this manner can enable the IT organization to better manage its priorities. See "Calculating and Comparing Data Center and Public Cloud IaaS Costs." (https://www.gartner.com/document/code/334805?ref=grbody&refval=3893681)
- **Wring out operational inefficiencies:** A complex IT portfolio impedes an organization's ability to respond to new demands and business opportunities. Is the primary goal of cloud migration the ability to install, provision and secure new infrastructure quickly? Or, does the enterprise want to reduce or reassign development or operations head count assigned to this application?
- **Consolidate infrastructure or constrain expansion:** For IT departments running out of power, cooling and physical space in their data centers, migrating or retiring applications frees up these precious resources for applications that must reside on-premises. This in turn enables the consolidation of remaining applications within the existing infrastructure.
- **Leverage existing investments:** To maximize their ROI of funds already sunk into IT, organizations want to leverage existing investments in development and operations skills, experience, tooling, infrastructure and deployed applications. Portfolio management activities must evaluate the net value of these investments. For example, is the application's codebase valuable? Alternatively, is the organization willing to modify its business processes to use migration options that encapsulate prepackaged processes and capabilities?
- **Open multichannel access:** Migrating an application to the cloud can enable wider secure access to the application for all employees, external partners and customers, regardless of desired form factor or location. For example, better support for mobile users is a common feature of some innovative PaaS and SaaS solutions.
- **Compose and integrate solutions:** Does the team desire to integrate the application more easily with other cloud, SaaS and web applications?

Legacy Application Characteristics

You must catalog specific architecture characteristics of the application that constrain the possible choices. Aspects of architecture that influence cloud migration decisions include:

- **Application pattern:** Architects should determine which of the following processing and interaction patterns characterize the application. Cloud platforms often have a bias that makes it easier (or impossible) to host particular patterns. See "Decision Point for API and Service Implementation Architecture" (https://www.gartner.com/document/code/351010?ref=grbody&refval=3893681) for more details on these application architecture patterns:
  - Web application (broken into the familiar three-tier or Model-View-Controller [MVC] pattern)
  - Modern (sometimes called single-page) web application
  - Legacy desktop application
  - Service-oriented architectures and APIs, including microservices and miniservices
  - Integration-centric architectures orchestrated by ESB or integration platforms
  - Batch-processing systems or systems that have regularly executed background processes
  - Event-driven architecture
  - Process-centric architectures relying heavily on BPM or RPA
  - Analytics- and BI-centric applications, including ML-based solutions

- **Application complexity:** Applications can have complexity in the codebase, configuration, integration and data management components, and this complexity must be addressed. For example, applications with complex or highly specialized business logic are more costly and risky to replicate, and require extensive testing to ensure logical consistency.
- **Data parallelism:** Data will migrate more successfully onto horizontally scalable cloud infrastructure if it either:
  - Exhibits high locality of reference (meaning it has an affinity to a specific location)
  - Is already partitioned (e.g., nonrelational, horizontally sharded key-value stores like MongoDB and Apache Cassandra)
- **Application licensing restrictions:** Server virtualization software has matured to the point that, from a technology perspective, most applications can efficiently run inside VMs or containers. There are exceptions to this (e.g., when an application relies on IP multicast, which is not supported by most virtualization configurations). But, many application vendors have not matched that technical maturity with licensing and support policies offering virtualization or containerization compatibility. For the application under consideration, architects should determine whether the vendor supports third-party COTS components or dependent libraries that run on a VM or container.
- **Demand life span and volatility:** An application may have an extremely limited life span if, for example, business demand is transient or the application satisfies one-off computing requirements. Variable and vacillating business demand can also influence whether the requirements warrant the attention of a development team, rather than prototyping several commercial software options or simply migrating the existing application as-is while demand stabilizes.

Modernization Requirements

For the revise and rearchitect alternatives, modernization of the existing codebase is an important aspect to the migration. Most codebases will require at least a moderate degree of refactoring in order to take advantage of the requirements in this list. This is especially true for rearchitecting. The lines of code that need to be refactored can directly impact your migration costs, adding time and resources to your overall project.

Decision makers should consider implementing the following modernization requirements when evaluating cloud migration options:

- **Scalability:** Applications must cope with additional concurrent users and increased data volume or transaction frequency. Scalable applications satisfy these additional demands by incrementally adding hardware resources. An application that is already horizontally scalable is a better fit for migration onto infrastructure that offers the ability to elastically scale resources in response to demand. Determining the degree to which the components are autonomous and stateless and finding the most critical processing bottlenecks, are required prework for this decision. Do the architects need to change the application architecture to remove latency or parallelism bottlenecks? Scaling purely by throwing hardware at an inherently nonscalable application will become expensive in an elastic and PAYG environment. (See "How to Assess Your Application to Adopt Cloud-Native Architecture" (https://www.gartner.com/document/code/326583?ref=grbody&refval=3893681) and "A Guidance Framework for Architecting Highly Available Cloud-Native Applications." (https://www.gartner.com/document/code/308407?ref=grbody&refval=3893681))
- **Security and access management:** Enforcing consistent role-based access control and providing single sign-on for users require federation of identity provisioning and management systems between on-premises and cloud services. Web-based services increasingly require support for open authorization and distributed authentication frameworks such as OAuth (http://oauth.net/) and OpenID (http://openid.net/) (see "Modern Identity and APIs: Mobile, OpenID Connect, OAuth, JSON and REST" (https://www.gartner.com/document/code/277246?ref=grbody&refval=3893681)). Teams that want to federate more easily with other web services and SaaS will favor providers that support interoperable identity management mechanisms. Additionally, business-critical applications require enforcement of policies related to auditing, data encryption, patching, security monitoring, data retention, data integrity, privacy and confidentiality.
- **Transactional integrity:** Online transaction processing (OLTP) systems can have stringent requirements regarding distributed transactions and support for a two-phase commit (2PC) protocol when updating multiple data sources to ensure immediate and strong consistency. Application semantics may also require that operations be completed in a strict order. Variable network latency, variable service provider availability and unpredictable service levels mean that applications with 2PC requirements are not suitable models for cloud platforms. Architects should modify such applications to move transactional integrity semantics from protocol to application semantics. For more information on transactional integrity, see "Eventual Consistency and Its Implications: Can You Trust Your DBMS?" (https://www.gartner.com/document/code/276734?ref=grbody&refval=3893681)
- **Integration requirements:** How an individual application relates to others in the application portfolio has significant impact on its migration prospects. Picking up and moving an application with multiple dependencies, point-to-point integration and complex interdependencies will be challenging, if not impossible. To leverage cloud computing beyond trivial use cases, organizations should plan to create an IT environment without boundaries. This requires a hybrid architecture spanning (or "cohabiting") internal, private and public cloud implementations. Modernization requirements may include implementing better interfaces before shifting the application to the cloud.

  Migration options vary in the degree to which integration with on-premises applications is possible. For example, can the migrated application

appear to be on a private subnet of the organization's network? Or, is the migrated application entirely cut off from on-premises applications, except for the provider's proprietary import/export interfaces and limited APIs? For more information on integration across the cloud, see "How to Integrate Cloud-Hosted, SaaS and On-Premises Applications." (https://www.gartner.com/document/code/353947?ref=grbody&refval=3893681)

- **Preserving codebase value:** Organizations with valuable codebases that provide competitive advantage will favor migration options that leverage this existing asset. This requirement is of particular interest if your organization plans to capitalize on what's being called the "algorithm economy."
- **Segmenting noncore (or context) capabilities:** Developing new code, or actively maintaining code that implements standardized or commodity processes, provides little competitive advantage. Requirements for shared business services, such as HR, finance or horizontal applications (such as web conferencing), are well-understood and riper for "buy" migration options than specialized competencies. Adopting some of the alternatives will depend on business users' willingness to modify business processes they regard as "specialized." Department-specific applications may also have limited scope, making them less risky to prototype on more innovative cloud platforms.

Development and Operations Skills Constraints

Externalizing an application by migrating it to a cloud platform implies handing over some operational control to the provider's staff. Examining skills profiles for the remaining development and operations teams will determine how much or little control is desirable to hand to the provider. Aspects to consider include:

- **Development team skills profile:** Modernizing legacy software sometimes requires that a development team recompile or repackage the existing system reliably from the source code. Given layers of version control sediment, this is not always a straightforward task, and may require specialist knowledge or technical folklore.

  Taking advantage of scalable infrastructure requires the development team to have concurrent programming, workload decomposition and data partitioning experience. The development team also requires experience in debugging distributed systems.

  Resolving the tension between familiarity and innovation is a key to choosing between migration alternatives. Innovative teams may be amenable to learning new languages, or to relearning how to build and deploy applications using new frameworks and containers. By contrast, teams relying on the familiar may favor migration rather than optimization and move their traditional application platforms onto platforms as similar to the status quo as possible. For more detail, see "The Renaissance Developer: Skills Guidance for Modern Application Programmers." (https://www.gartner.com/document/code/300861?ref=grbody&refval=3893681)

- **Operations team profile:** The migration project should determine whether the enterprise presently has skilled resources to efficiently install, manage and maintain the application. A key consideration is whether the system administrators and network engineers are well-versed in "infrastructure as code" and familiar with modern automation tooling to provision and maintain cloud environments. See "The Cloud Engineer: An Evolutionary Role for a New IT Era," (https://www.gartner.com/document/code/322973?ref=grbody&refval=3893681) "Analyzing the Role and Skills of the I&O Professional in DevOps" (https://www.gartner.com/document/code/354968?ref=grbody&refval=3893681) and "Strengthen Your DevOps Capability With Platform Ops." (https://www.gartner.com/document/code/354920?ref=grbody&refval=3893681)

  Another consideration is whether the database administrator (DBA) teams can use and navigate the cloud tooling to determine:

  - How data will be persisted, replicated and managed
  - How the data packages that the application stack depends on will be migrated

  Cloud platforms offer very low-level tooling, automation and environment templating that are aimed at experienced system administrators. For more information on infrastructure as code and continuous infrastructure automation, see "To Automate Your Automation, Apply Agile Practices and DevOps Tools to Infrastructure and Operations." (https://www.gartner.com/document/code/355121?ref=grbody&refval=3893681)

- **New skills:** Externalizing to a cloud environment requires IT solution teams to acquire new skills or upgrade rusty ones, such as vendor management, capacity planning, metrics and cost estimation. Also falling into this category is DevOps, which focuses on improved collaboration and cooperation between the development and operations sides of the organization. DevOps requires a heavy investment in automation skills.

  For more information on agile, DevOps and measuring their success, see "Solution Path for Achieving Continuous Delivery With Agile and DevOps" (https://www.gartner.com/document/code/342688?ref=grbody&refval=3893681) and "Choose the Right Metrics to Drive Your Agile, DevOps and Continuous Delivery Initiatives." (https://www.gartner.com/document/code/320805?ref=grbody&refval=3893681)

Migration Cost Factors

Migrated applications have to meet adequate service levels. Breaking down your existing application's SLAs into their constituent measurements and baselining them accurately are important premigration steps. Engineers tend to ignore the need to measure this data, because budgeting for on-premises applications can capitalize resources, such as server or storage, to make the cost of those resources disappear as "sunk cost." With cloud deployments, all computing resources are measured, may be explicitly charged for, and cannot be hidden with accounting techniques. Before migrating an application to a cloud environment, perform iterative design optimization cycles to prove SLAs are achievable and to reduce cloud computing resource costs. (See "Comparing Tools to Track Spend and Control Costs in the Public Cloud," (https://www.gartner.com/document/code/323083?ref=grbody&refval=3893681) "How to Manage Public Cloud Costs on Amazon Web Services and Microsoft Azure" (https://www.gartner.com/document/code/336254?ref=grbody&refval=3893681) and "Calculating and Comparing Data Center and Public Cloud IaaS Costs." (https://www.gartner.com/document/code/334805?ref=grbody&refval=3893681))

The following application requirements will impact design (and execution cost) of a modernized or cloud-migrated application:

- **Low-level interaction patterns:** For example, the level of "chattiness" of the application's distributed components impacts the need for API usage and data movement.
- **Data requirements:** Architects must assess the volume of data the application generates and/or needs to store, and the nature of the required storage. For example, the application may explicitly require file or block storage services and include specific performance requirements, retention policies or compliance restrictions.
- **Code-refactoring requirements**: Legacy codebases are not well-tested and have poor code coverage. For the revise strategy, the code refactoring is geared primarily toward optimizing the application to use cloud services over self-managed services. For the rearchitect strategy, refactoring the codebase is a significant undertaking to transform the architecture into a cloud-native one. This code refactoring involves heavy lifting and the potential to rewrite major portions of the application, translating directly into cost.
- **Computational requirements:** Architects must assess the requirements for their application based on its workload profile, including CPU, memory footprint, and extended capabilities like GPU, FPGA and dedicated ML processors (e.g., TPUs).
- **User requirements:** Quantitative analysis of the number of users (including concurrent users) accessing the application is important capacity- and migration-planning data.
- **Latency:** Interactive applications are sensitive to network and processing latency for optimal usability. Latency due to the distance between users and a remote service provider requires you to consider regional proximity. The goal should be to select a provider that can guarantee an acceptable response time.
- **Security:** Architects must determine the sensitivity of a migrated application and whether it requires special handling of network, data or other types of security. Some migration options may make custom encryption impossible, while others allow for a wide variety of approaches. (See "Implementing Cloud Security Monitoring and Compliance Using Amazon Web Services" (https://www.gartner.com/document/code/351316?ref=grbody&refval=3893681) and "Comparing Security Controls and Paradigms in AWS, Google Cloud Platform and Microsoft Azure." (https://www.gartner.com/document/code/343562?ref=grbody&refval=3893681))
- **Data transfer:** Architects must assess the network bandwidth and speed required to support the ingress and egress data rates — before migration. Other data transfer considerations include whether the application requires a specific transport or network protocol for communicating internally or for exposing APIs to provide access to external clients. (See "Architecting an Amazon Web Services Account Governance and VPC Design Strategy" (https://www.gartner.com/document/code/347966?ref=grbody&refval=3893681) and "Best Practices for Amazon VPC and Azure VNet." (https://www.gartner.com/document/code/337223?ref=grbody&refval=3893681))
- **Independent software vendor (ISV) licensing requirements:** Cloud providers may use a Services Provider License Agreement (SPLA) from ISVs (e.g., IBM or Microsoft). In an SPLA, the price of the software license is included in the PAYG compute instance price. However, the specific SPLA may not allow consumers to transfer internal software licenses to the cloud.

Alternatives

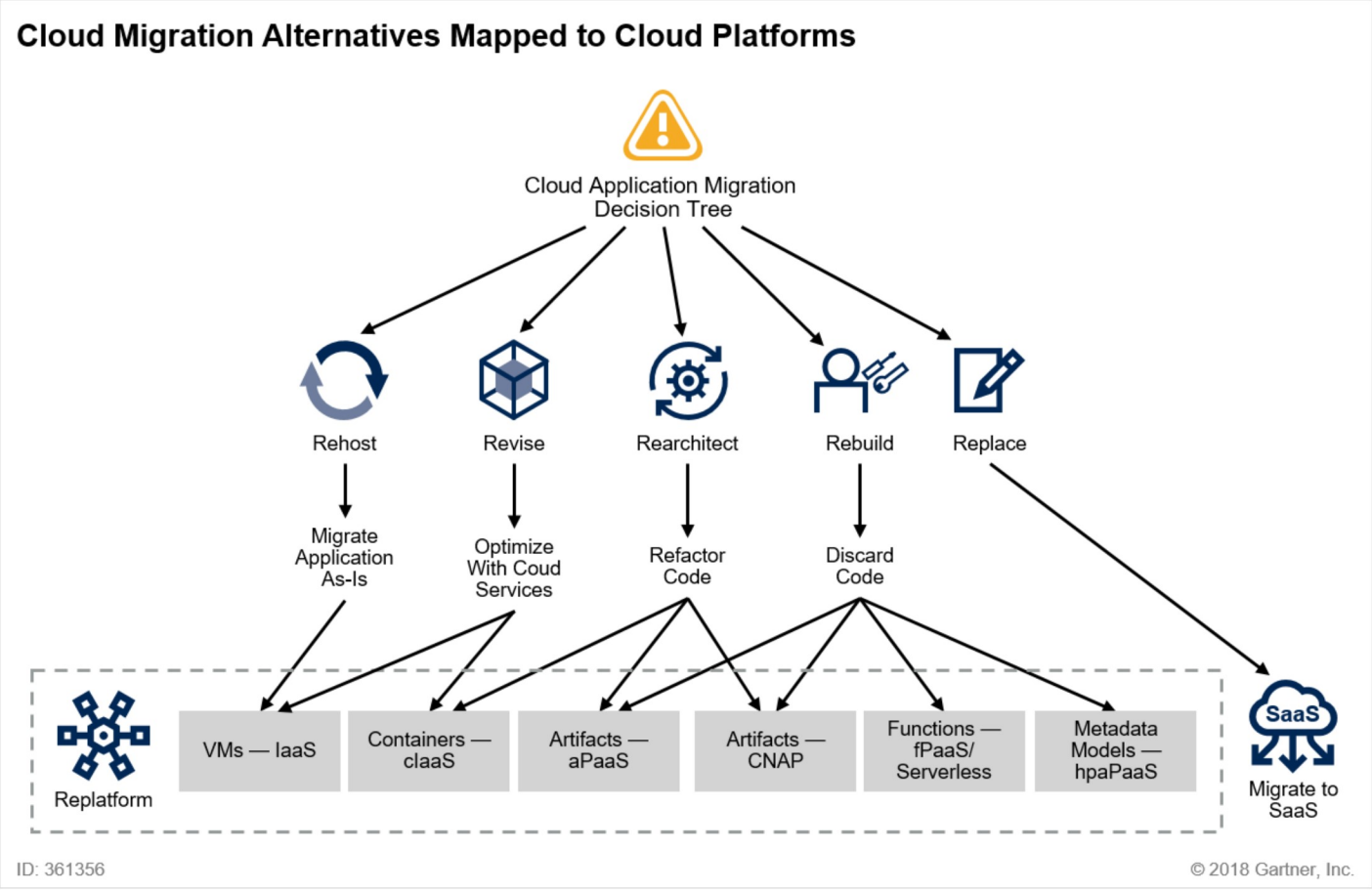
The cloud computing market is dominated by a small group of first-tier cloud providers, including Amazon Web Services, Microsoft Azure and Google Cloud Platform, with several followers, including Alibaba Group, Oracle, IBM and DigitalOcean. Each provider has essentially the same platform options, including IaaS, CaaS and hcaPaaS, which are discussed below in further detail.

This Decision Point presents cloud migration strategies as a choice between five alternative actions. However, it does not go into the specifics of migrating the application to a specific cloud platform on a specific cloud provider. This Decision Point leads readers to an appropriate migration strategy rather than toward a particular service provider. The alternative migration strategies considered in this Decision Point are explained and differentiated in this section. No matter which alternative you choose, make sure you pair your choice with a clear strategy to decommission the existing version or instances of the application. Many cloud migration projects have been derailed by an inability to move the last few stragglers off of the legacy solution. The bullets below briefly describe the five options, and Figure 5 maps them to the appropriate cloud platforms:

- **Rehost:** Leave the application intact and "lift and shift" it into the cloud provider's platform.
- **Revise:** Replace self-managed application components with the cloud provider's equivalent services. This is primarily an optimization step.
- **Rearchitect:** Take full advantage of cloud computing characteristics by:
  - Redesigning the application to be more testable, resilient and scalable
  - Refactoring the codebase
  - Utilizing the cloud provider's services for application capabilities
  - Running on a new platform
  - Introducing CI/CD automation

  This is the first strategy that is cloud-native.
- **Rebuild:** Fundamentally rethink the architecture, and leverage an entirely new cloud platform to build, package, deploy, run and operate the application in a cloud-native fashion.
- **Replace:** Select an alternative SaaS application that provides you with equivalent functionality that you are not running or building yourself, but rather consuming from a cloud provider.

Figure 5. Cloud Migration Alternatives Mapped to Cloud Platforms



Source: Gartner (November 2018)

As shown in Figure 5, the mapping between the migration strategies and the destination cloud platforms is not straightforward. It is increasingly a spectrum of compute platform options spanning from VMs on IaaS to metadata-defined models in hpaPaaS. This is reflected by the positioning of the providers and by the repetition of the largest ones that offer multiple tiers of service today. For example, Microsoft, which began with primarily an application PaaS focus, has dramatically expanded into IaaS (e.g., Azure Virtual Machine Scale Sets and Azure Resource Manager). By contrast, the most widely adopted IaaS provider, Amazon, continues to decorate its platform with a variety of PaaS capabilities (e.g., Amazon RDS and AWS

Lambda). fPaaS is the latest application platform offering from the cloud providers (see "How to Build Cloud-Native Applications Using Serverless PaaS" (https://www.gartner.com/document/code/368464?ref=grbody&refval=3893681) and "Adding Serverless Computing and fPaaS to Your Cloud-Native Architecture Toolbox" (https://www.gartner.com/document/code/318340?ref=grbody&refval=3893681)).

Cloud-native application platforms (CNAPs) like Pivotal Cloud Foundry and Red Hat OpenShift are becoming increasingly popular as an option to develop and operate applications independently of the underlying cloud (or on-premises) infrastructure, thus reducing cloud vendor lock-in and increasing portability. (See "Comparing Leading Cloud-Native Application Platforms: Pivotal Cloud Foundry and Red Hat OpenShift." (https://www.gartner.com/document/code/370391?ref=grbody&refval=3893681)) This approach also enables multicloud support by allowing developers to target the same operating environment using multiple cloud providers. It is important to note that, when you are using a CNAP as your target migration platform, a key success factor is to have a strong platform ops capability in place. (See "Strengthen Your DevOps Capability With Platform Ops." (https://www.gartner.com/document/code/354920?ref=grbody&refval=3893681))

For details on the pros and cons of the various target platforms, see "Selecting a Cloud Platform for DevOps Delivery With Microservice Architecture." (https://www.gartner.com/document/code/320951?ref=grbody&refval=3893681)

Rehost

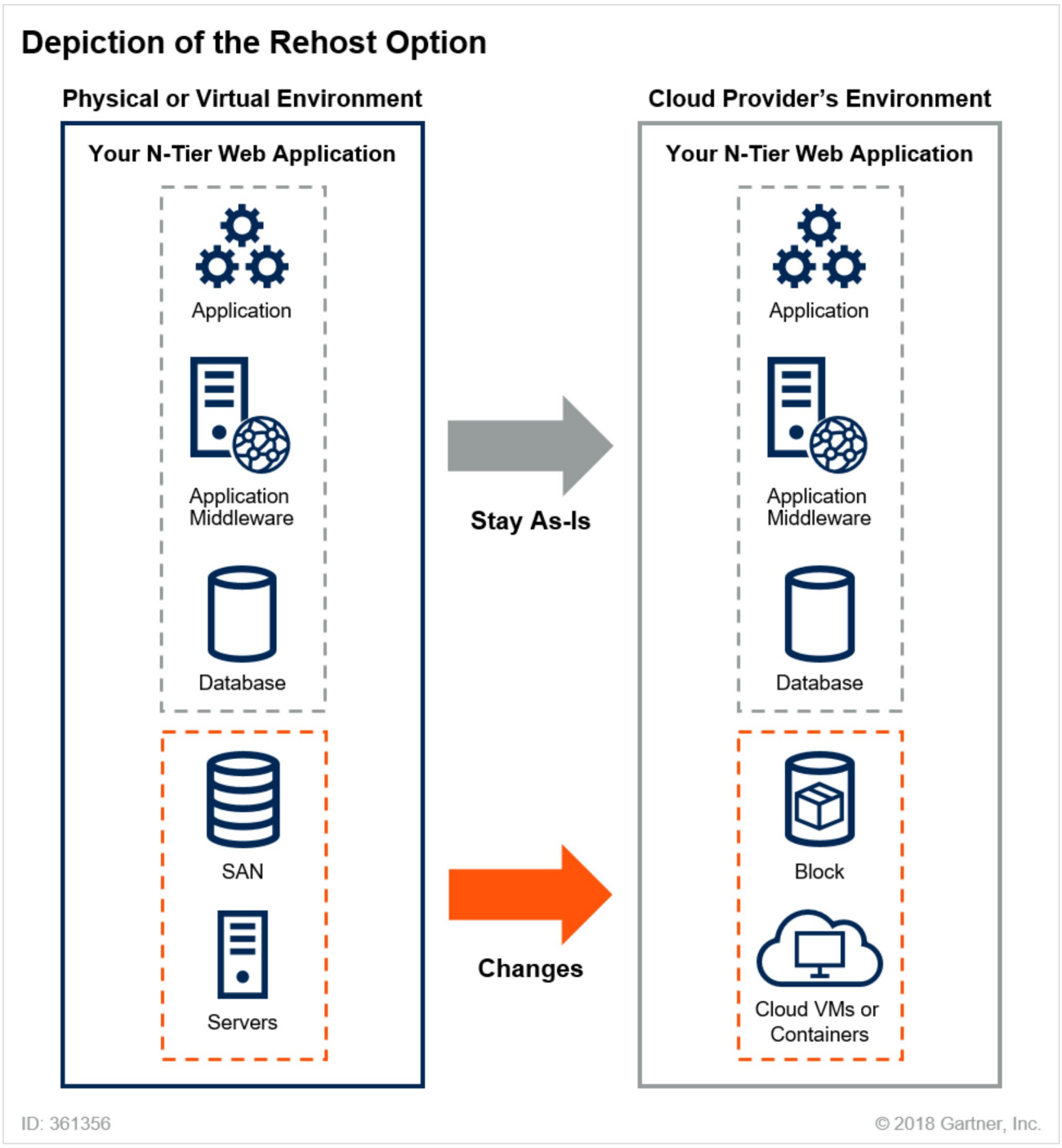
Table 3 concisely defines the rehost migration option, and Figure 6 depicts it graphically.

Table 3: Defining the Rehost Migration Alternative

The Rehost Migration Strategy ↓	
Definition	■ You redeploy the application to a different hardware environment. ■ You change the application's infrastructure configuration. ■ To use an analogy, this strategy is akin to redeploying a Java application server on a Linux x86 server instead of a Solaris SPARC-based server.
Cloud Platform Options	■ VMs on IaaS ■ Containers on CaaS
What Services Are We Consuming?	■ VMs as a service ■ Containers as a service ■ Block or volume storage as a service ■ Environment configuration templating (optional)
Audience	■ Cloud architects ■ System administrators ■ Operations staff ■ DevOps teams
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, possibly declared using AWS CloudFormation templates

Source: Gartner (November 2018)

Figure 6. Depiction of the Rehost Option



Source: Gartner (November 2018)

Rehost migration can work with systems where code modifications are impossible, but where the application can be reconfigured to run on different hardware infrastructure. If we assume the application and its dependencies can run in a VM, this "lift and shift" strategy is possible not only for in-house-developed systems, but also for COTS systems and other code that cannot be rebuilt. Rehosting an application without making changes to its architecture can deliver a cloud migration solution over a relatively short timeline, using the cloud provider's migration tools. Scenarios where a rehost can be a good option include:

- If you need to immediately exit a data center due to a lease expiration, M&A activity, a spinoff, or a threat of natural disaster, then IaaS can be a good option.
- A stateless web application that can be effectively load-balanced over multiple instances can be a good fit, assuming it does not have dependencies on local resources in the host OS. IaaS can provide elastic scalability if the application or data is already parallelizable.
- Rehosting can be a good option for replicating an existing software development life cycle (SDLC) with a large, complex delivery pipeline that is

built on on-premises assumptions. Rehosting can be an advantage for teams with mature software development and operations processes.

- Rehosting is beneficial when you are rearchitecting an application that has key dependencies on applications that do not require a sophisticated level of engineering. For instance, let's say you have Application A, which is getting an overhaul, but it has a dependency on Application B, which doesn't require any cloud capabilities. In this instance, a rehost of Application B would be more cost-effective than building an integration point between the two. Simply having Application B closer in proximity to Application A provides advantages in this case. There may never be a reason to do anything to Application B beyond rehosting it.

The disadvantages of choosing the rehost option include:

- "Moving without improving"
- Using IaaS as "rented virtualization"
- Low-level, do-it-yourself (DIY) nature of the service

The primary advantage of using IaaS for rehosting — that teams can migrate systems quickly without modifying their architecture — also becomes its primary disadvantage. If a team rehosts an application using IaaS without making changes to the application's architecture, the application will not benefit from the cloud characteristics of the infrastructure (e.g., elastic scalability). Rehosting may afford efficiency and operations improvements in the same way as moving applications from existing servers into a new data center with server virtualization and consolidation. However, neither activity makes application architecture improvements. That said, where project goals align well, rehosting to IaaS is an excellent option for fast time to market and a high level of organizational control. Examples of such projects include mandates to quickly shut down data centers or to implement a simple disaster recovery strategy.

Rehosting Using Containers

The traditional unit of compute service offered by cloud providers is either a VM instance or a container (e.g., Docker) that runs on CaaS (e.g., Google Kubernetes Engine [GKE], Amazon EKS or Azure Kubernetes Service [AKS]). Before the application's execution environment can be reproduced, the team needs to create a VM or container image configured with a complete stack that encompasses all the application's moving parts and dependencies (e.g., application server). Providers manage image marketplaces that allow teams to pick the closest image "off the rack." Alternatively, teams can build their own images.

One show-stopping issue for rehost migrations is whether third-party applications or dependent libraries are licensed, are supported and will work for deployment in a VM or container (e.g., UDP multicast doesn't work).

Developers who deploy applications to the cloud, on IaaS or CaaS, find that they have to do their own system administration, learn new configuration tooling, write configuration scripts and patch their systems. Unless the migration strategy simultaneously brings along application development, data management, security, identity and operations, developers can find themselves swimming in elastic Internet Protocol (IP) numbers and hacking Domain Name System (DNS) registries just to keep their applications running.

Each cloud provider has different APIs for management and configuration, and several competing VM image and storage formats exist, but none is the winner. For example, the largest IaaS player, Amazon, has not open-sourced its VM image format, thus leaving users constrained to using its Amazon Machine Image (AMI) format and APIs, and its CloudFormation environment declaration format. The downside of potentially getting locked into any vendor's tooling is that scripts, playbooks and artifacts developed to automate rehosting an application on one provider will not work with another. Effort spent assembling VM images will be wasted if the team decides to switch providers.

Lock-in is a concern in the cloud/no-cloud decision, but it is not a disadvantage of IaaS or CaaS when compared with the other platform options. Any migration to the cloud involves lock-in concerns, but of the five alternatives, rehost has the least lock-in concern. To mitigate this lock-in concern, there are options for automating your infrastructure builds using third-party tooling like Red Hat Ansible or HashiCorp Terraform (see "Assessing Terraform for Provisioning Cloud Infrastructure" (https://www.gartner.com/document/code/328206?ref=grbody&refval=3893681)). See "Evaluation Criteria for Cloud Infrastructure as a Service" (https://www.gartner.com/document/code/355118?ref=grbody&refval=3893681) for greater detail on the IaaS market. For details on infrastructure automation, see "How to Automate Server Provisioning and Configuration Management" (https://www.gartner.com/document/code/355120?ref=grbody&refval=3893681) and "To Automate Your Automation, Apply Agile Practices and DevOps Tools to Infrastructure and Operations." (https://www.gartner.com/document/code/355121?ref=grbody&refval=3893681)

When to Rehost

The rehost alternative is appropriate when the primary goal is to leverage existing software investments. Rehosting on IaaS or CaaS is most appropriate for the following scenarios:

- Applications that do not require any modernization, but that simplify integration for other applications that are being rearchitected or rebuilt in the cloud provider.
- Short-lived, transient applications, where the hardware is not available or worth additional investment.
- Organizations that lack capital to procure enterprise IT infrastructure (e.g., startups or new lines of business).
- Compute-intensive applications that are already built for parallelism, that have independent datasets, and for which load balancing already increases scalability and availability. Some applications built on clustering technology are designed for parallelism, but they are not good candidates for rehosting because they require high-performance interprocess communications (IPC) and multicast UDP.

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

Revise

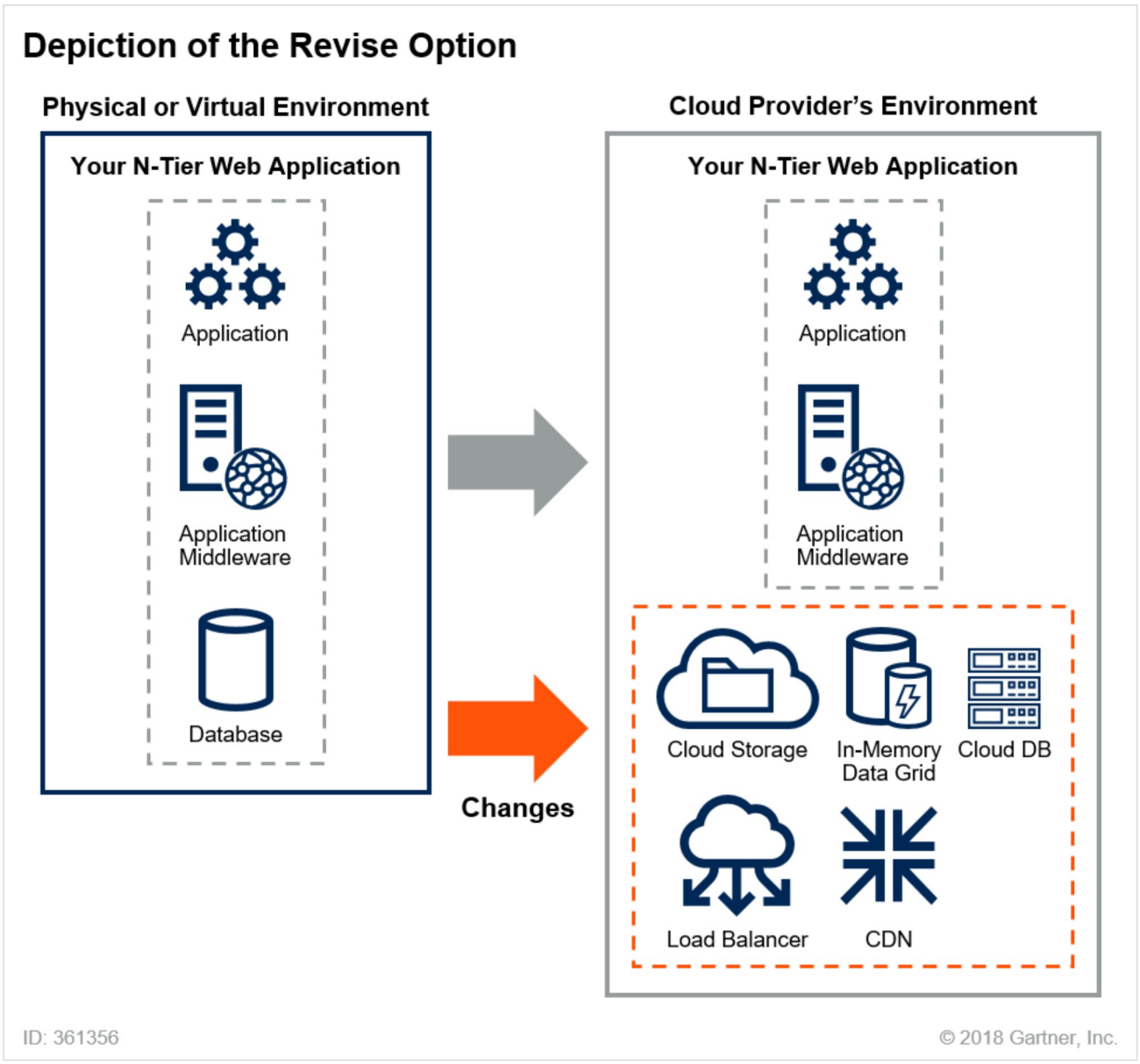
Table 4 concisely defines the revise migration option, and Figure 7 depicts it graphically.

Table 4: Defining the Revise Migration Alternative

Revise Migration ↓	
Definition	■ You run your applications (usually web applications) on a cloud provider's infrastructure. ■ You make application code or configuration changes to connect the application to new infrastructure services. ■ This alternative is analogous to linking in a new database driver, identity management system or authentication module. It is also analogous to moving a Java EE application from IBM WebSphere to Red Hat JBoss (same type of container, but some different frameworks and configuration). ■ Necessary changes are contained to configuration and minimal code modifications. ■ Existing programming models, languages and frameworks can be used and extended.
Cloud Platform Options	■ IaaS ■ CaaS ■ hcaPaaS
What Services Are We Consuming?	■ Provider-supplied cloud-based frameworks and management tools that allow developers to take advantage of the cloud characteristics of the provider's infrastructure
Audience	■ Cloud architects ■ System administrators ■ Operations staff ■ DevOps teams
Examples	■ Redeploying an existing Ruby on Rails web application to Heroku or an existing ASP.NET app to Microsoft Azure App Service (after doing a thorough analysis of the application to ensure it follows specific application patterns for hcaPaaS, such as 12-factor). While possible, the majority of applications require a rearchitecture to deploy to an hcaPaaS service. ■ Replacing a third-party load balancer with the cloud provider's load balancer (such as AWS Elastic Load Balancing). ■ For an application hosted on Amazon EC2, replacing a cloud-hosted Oracle DBMS instance with Amazon Relational Database Service (RDS).

Source: Gartner (November 2018)

Figure 7. Depiction of the Revise Option



Source: Gartner (November 2018)

The primary advantage of the revise strategy is blending familiarity with innovation. The team can take advantage of the providers' cloud services, which have baked-in best practices for cloud architecture. One example of the revise strategy is migrating from a relational DBMS to a dbPaaS. You are swapping out application components with cloud services that ultimately reduce your operational overhead while retaining the core of the application as-is with little to no code changes.

The primary disadvantage of the revise strategy is that the core of your application does not take on any modernization effort. The code is left intact and cannot take advantage of cloud resources for elastic scalability. Retaining this legacy application architecture has the potential to hold back development teams, inhibit scalability and availability, and introduce bottlenecks in your application.

When to Revise

Revising a modified application onto a cloud provider's platform is optimal when leveraging an existing, mature development skill set and codebase is paramount and code portability is a concern. Application fit is crucial: When the application fits a standard pattern (e.g., n-tier web application) or can be easily modified to work within the providers' restrictions, revising can be a productive and cost-effective execution environment.

Revising is not an appropriate option and can be ruled out when:

- An application cannot easily be adapted to meet the provider's restrictions

- Skills or infrastructure to rebuild existing code is unavailable (e.g., build scripts or dependencies cannot be located and fixed, or developers cannot be assigned the task)

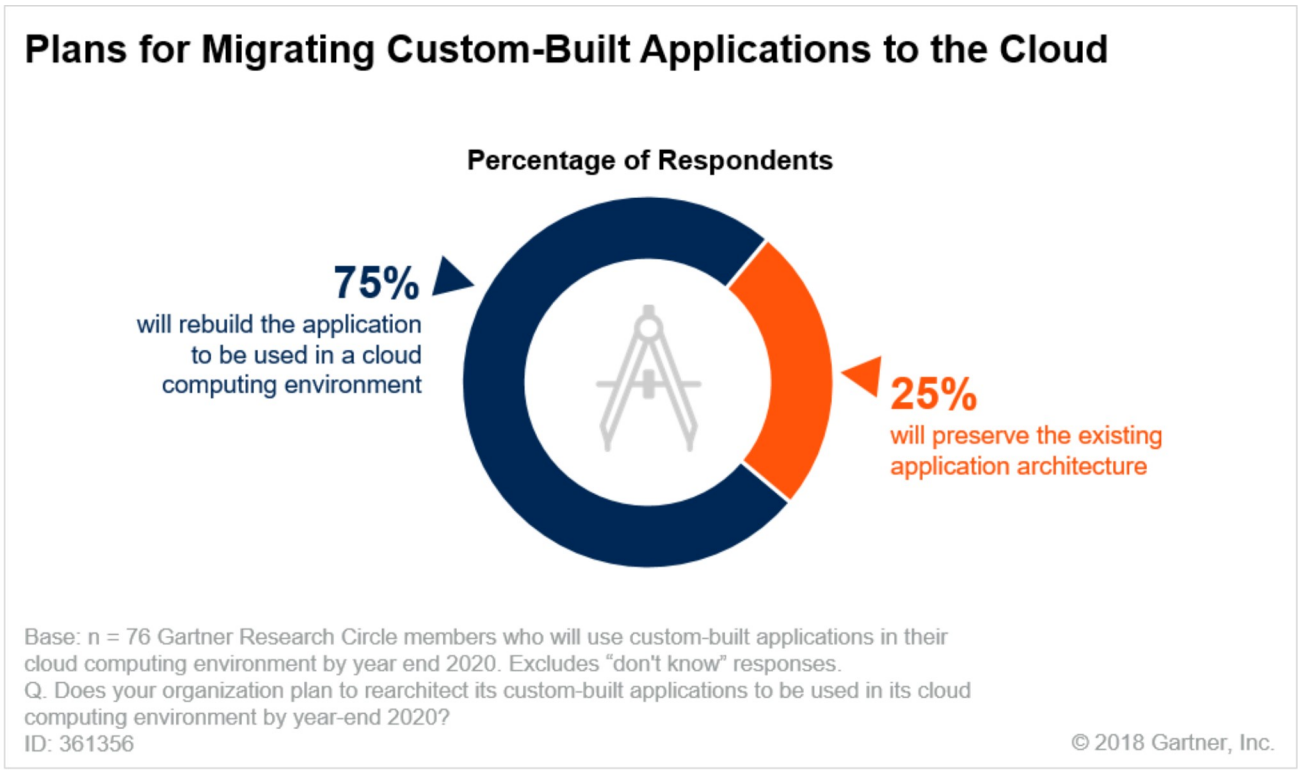
Rearchitect

In the 2017 Gartner Public Cloud Adoption Strategies Survey, [1] we asked respondents who plan to use custom-built applications in their cloud environment the following question:

"Does your organization plan to rearchitect its custom-built applications to be used in its cloud computing environment by year-end 2020?"

Figure 8 below illustrates that three-quarters of the respondents plan to rearchitect those migrated applications.

Figure 8. How Do You Plan on Migrating Your Applications to Cloud?



Source: Gartner (November 2018)

Table 5 concisely defines the rearchitect option, and Figure 9 depicts it graphically.

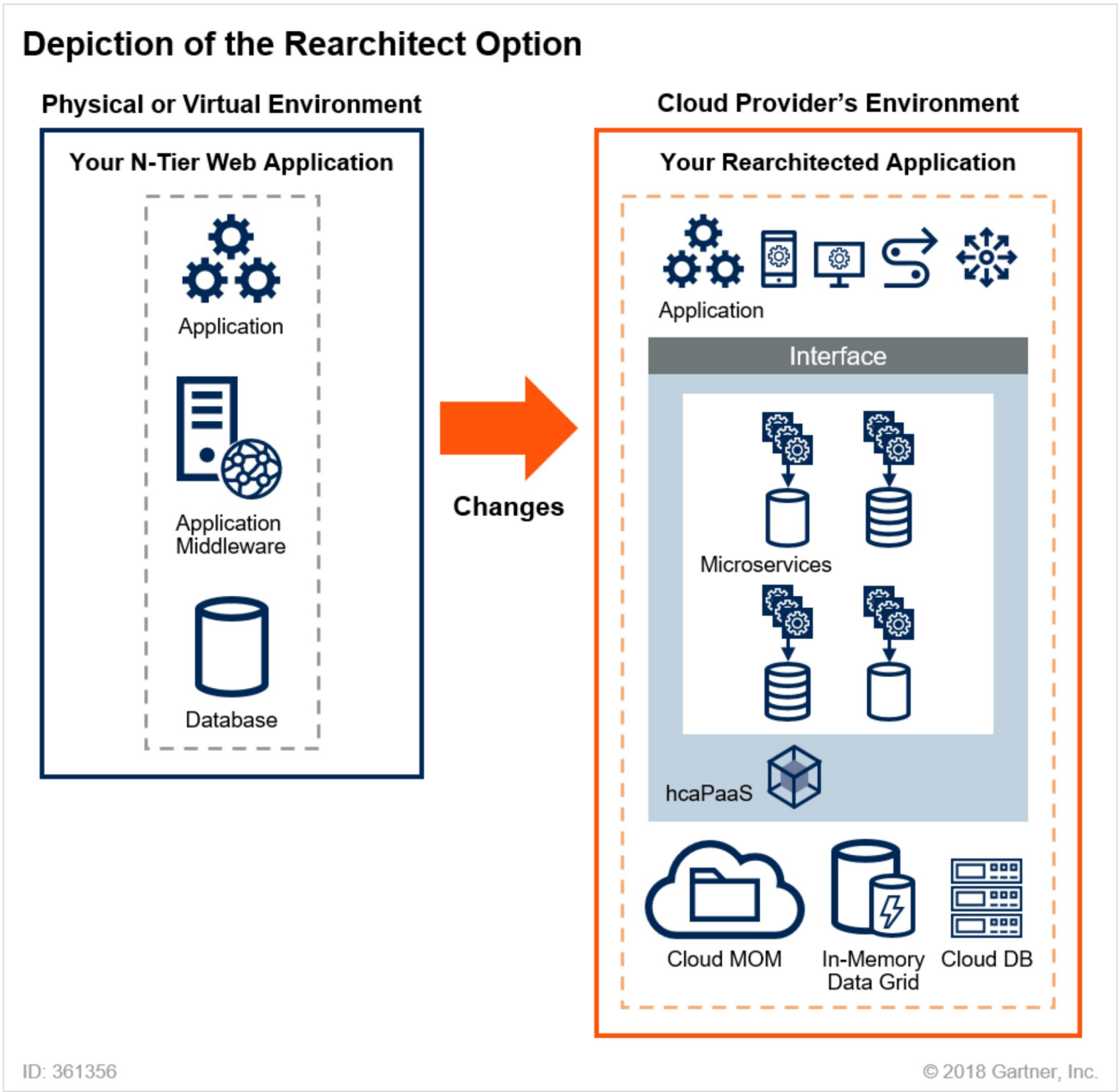
Table 5: Defining the Rearchitect Migration Alternative

Rearchitect Migration ↓	
Definition	■ You modify or extend the existing codebase to support modernization requirements, and then use the rehost or revise option to deploy to the cloud. ■ The scale of changes encompasses major revisions to add new functionality or to rearchitect the application for the cloud.
Cloud Platform Options	■ CaaS ■ aPaaS ■ fPaaS
What Services Are We Consuming?	■ Frameworks and management tools (CaaS, PaaS and fPaaS) that allow developers to take advantage of the cloud characteristics of a provider's infrastructure

Rearchitect Migration ↓	
Audience	■ Cloud architects ■ Developers
Examples	■ Redesigning a monolithic Java application, decomposing the functionality into smaller services to run in parallel, and then deploying on Amazon EC2 Container Service

Source: Gartner (November 2018)

Figure 9. Depiction of the Rearchitect Option



Source: Gartner (November 2018)

By choosing the rearchitect alternative, application owners can take advantage of a valuable codebase, enhancing it to meet other cloud adoption or legacy modernization goals. This is the first step toward a cloud-native architecture for the migrated application. Enabling elastic scalability, dynamic reconfiguration and creative billing options most likely requires substantial changes to the application architecture.

Compared with the other migration alternatives, rearchitect puts the organization in a position to optimize the application to leverage the cloud characteristics of providers' infrastructure.

The downside of rearchitecting an application is that kicking off a (possibly major) development project will require upfront expenses to engage a development team. Determining the length of time to rearchitect an application depends on the scale of the work and on the scope of functional requirements versus the changes needed to make those requirements work in the cloud. While rearchitect is the option likely to take the most time to deliver its capabilities, there are exceptions where rearchitecting can take less time than rebuilding, if the organization can reuse a large majority of the codebase. Of course, if the organization has already decided to undertake substantial functionality changes to the application, then the cost of this downside can be baked into the overall project budget for reworking the application. To assess your application for rearchitecture, see "How to Assess Your Application to Adopt Cloud-Native Architecture." (https://www.gartner.com/document/code/326583?ref=grbody&refval=3893681)

A Note on hcaPaaS

The rearchitect and rebuild strategies are heavily dependent upon abstractions beyond that of a VM. One of the primary goals is to spend less time on operations and more time on developing code. An hcaPaaS is a common service to replatform on. However, it has some known disadvantages that include missing capabilities, transitive risk and framework lock-in. In the hcaPaaS market, some of the capabilities that developers depend on in existing platforms can be missing from hcaPaaS offerings. Architects should verify that an hcaPaaS supports the required frameworks and APIs. Other PaaS offerings may not be so restrictive, such as those based on the open-source Cloud Foundry framework (e.g., IBM Cloud Foundry or Heroku), but applications may require revisions in other architecture aspects to fit into their runtime environments.

Each provider has its own model for workload scaling in hcaPaaS. For example:

- *Heroku has dynos, slugs and workers.*
- *Google App Engine for Java has automatic scaling using a servlet abstraction.*
- *Microsoft's Azure App Service platform uses instances.*

"Backward-compatible" hcaPaaS means developers can reuse languages, frameworks and containers they have invested in (e.g., hosting a .jar file on Google App Engine), thus leveraging code the organization considers strategic.

Some hcaPaaS providers depend on other cloud providers for infrastructure (e.g., Heroku uses Amazon Web Services), creating a transitive risk for the consumer. One common concern of hcaPaaS consumers is that of code and framework portability. Some hcaPaaS providers do not provide an on-premises product option in addition to hosting an application on their own infrastructure (e.g., Salesforce Lightning Platform or Google App Engine). However, other hcaPaaS providers do (e.g., Pivotal Container Service on GKE, Red Hat OpenShift on Azure, IBM Cloud Private and Microsoft Azure Stack). Code written for a provider's specific framework (e.g., Azure Service Bus routing or Google App Engine data store) will only work against the provider's APIs, regardless of what language the code is written in (e.g., C#, Java, golang or Node.js).

The major trade-off with hcaPaaS is that of productivity for portability. hcaPaaS is less portable than IaaS or CaaS. With hcaPaaS, the team relies heavily on a proprietary application runtime and tooling, as well as on strong dependencies to cloud services (such as decision services and AI/ML services).

When to Rearchitect

Rearchitect is an appropriate option when the application needs a major revision to incorporate new capabilities, or when the benefits of leveraging cloud characteristics justify the investment required.

Rearchitect should be ruled out if "time to cloud" is the overriding priority, project scope creep is an unacceptable risk, or the codebase is no longer valuable to the organization.

Rebuild

Table 6 concisely defines the rebuild migration option, and Figure 10 depicts it graphically.

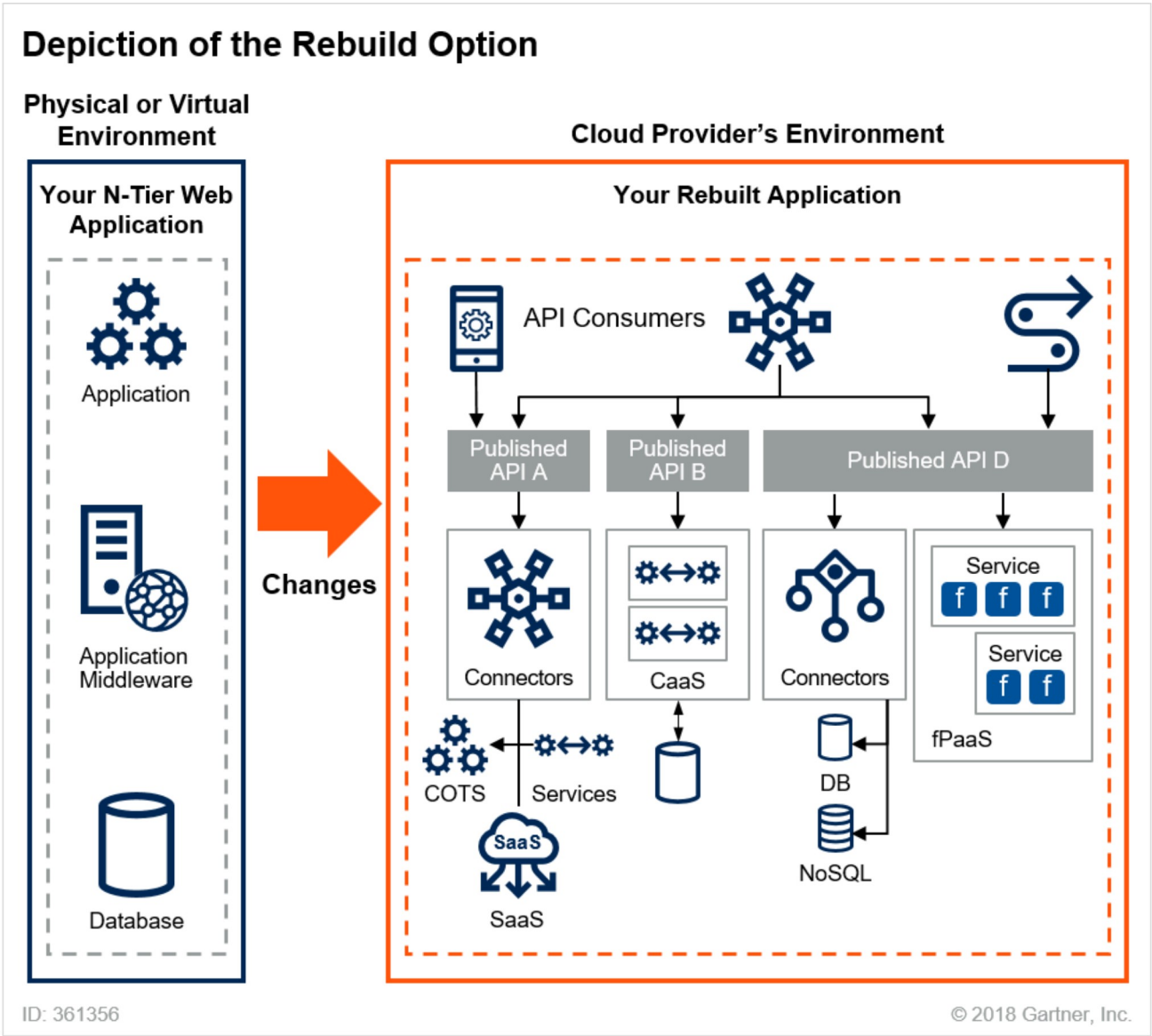
Table 6: Defining the Rebuild Migration Alternative

Rebuild Migration ↓

Rebuild Migration ↓	
Definition	■ You rebuild your solution on a cloud provider's application platform. ■ You discard code for an existing application. ■ The rebuild alternative requires rearchitecting the application for a new application container. ■ Not all existing programming models, frameworks and languages will be retained. ■ This option is analogous to moving from Java to .NET. However, you are more likely to be programming for an application container at a higher level of abstraction, or for a simpler application container or runtime like .NET Core or Node.js.
Cloud Platform Options	■ One or a combination of hcaPaaS, fPaaS and hpaPaaS
What Services Are We Consuming?	■ A cloud application platform for building and running applications
Audience	■ Cloud architects ■ Developers (including "citizen developers," for simple applications)
Examples	■ Building a modern banking application that can be accessed through multiple channels, packaged in containers, and deployed and run on Kubernetes with Istio service mesh and Prometheus for monitoring. ■ Modernizing a C and FORTRAN financial risk calculation application by redesigning it in C#, and then using .NET Core on Azure Service Fabric decomposed as microservices. See "Assessing .NET Core for Modernizing .NET Applications" and "Assessing Microsoft Azure Service Fabric for DevOps and Microservice Architecture." ■ Building a Salesforce Lightning Platform application for order management. See "Choosing a Platform for Building and Extending Salesforce Sales Cloud Applications." ■ Rebuilding parts of an ERP through simple workflows using an hpaPaaS like Mendix, OutSystems, Progress Rollbase or Microsoft PowerApps.

Source: Gartner (November 2018)

Figure 10. Depiction of the Rebuild Option



Source: Gartner (November 2018)

Although rebuilding requires giving up the familiarity of existing code and frameworks, the advantage of rebuilding an application is access to innovative features in the provider's runtime environment. Developer productivity is improved with:

- Tools that allow application templates and data models to be customized
- Metadata-driven engines
- Communities that supply prebuilt components

Rebuild enables adoption of new platforms and new architecture paradigms.

Mesh app and service architecture (MASA) is an architecture pattern being adopted by enterprises to enable more agility in their applications. This pattern allows applications to interact with more channels, applications and services through APIs and open standards. Gartner's "Adopt a Multigrained Mesh App and Service Architecture to Enable Your Digital Business Technology Platform" (https://www.gartner.com/document/code/338573?ref=grbody&refval=3893681) describes MASA as follows:

> **"Multigrained MASA incorporates numerous application architecture innovations and best practices to deliver delightful, extensible, flexible, agile, scalable, reactive, resilient and integratable applications."**

Other architectural models that are open include streaming, event-driven, microservices and serverless architectures, which are supported through various cloud platforms and cloud provider services. Microservices are fine-grained services that implement an individual feature and that are independently deployable. Microservices not only increase agility, but also require more complexity in operations to support discoverability, communications and operations of many services. See "Solution Path for Using Microservices Architecture Principles to Deliver Applications." (https://www.gartner.com/document/code/347054?ref=grbody&refval=3893681)

There are variations of cloud application platforms that include fPaaS, hcaPaaS, hpaPaaS, CaaS and CNAPs. In general, these offerings allow applications to transparently and automatically scale to comply with the provider's framework and container model. Platforms that exploit multitenancy automatically upgrade every tenanted application, simultaneously freeing the consumer from patching, upgrading dependencies and migrating users between versions. For more information on cloud application platforms, see:

- "Adding Serverless Computing and fPaaS to Your Cloud-Native Architecture Toolbox" (https://www.gartner.com/document/code/318340?ref=grbody&refval=3893681)
- "Selecting a Cloud Platform for DevOps Delivery With Microservice Architecture" (https://www.gartner.com/document/code/320951?ref=grbody&refval=3893681)
- "Evaluation Criteria for Public Cloud Application Platform as a Service" (https://www.gartner.com/document/code/370407?ref=grbody&refval=3893681)
- "Comparing Leading Cloud-Native Application Platforms: Pivotal Cloud Foundry and Red Hat OpenShift" (https://www.gartner.com/document/code/370391?ref=grbody&refval=3893681)
- "How to Build Cloud-Native Applications Using Serverless PaaS" (https://www.gartner.com/document/code/368464?ref=grbody&refval=3893681)

hpaPaaS offers capabilities to define applications through metadata-driven or canvas-driven (i.e., "point and click" programming) approaches, rather than code. Thus, hpaPaaS gives nonprofessional developers (i.e., citizen developers) the opportunity to develop and deploy simple applications (see "Architecting Low-Code Cloud Applications With High-Productivity aPaaS" (https://www.gartner.com/document/code/350052?ref=grbody&refval=3893681)). With hpaPaaS platforms, lock-in is the primary disadvantage. If the provider makes a pricing or technical change that the consumers cannot accept, breaches SLAs or fails catastrophically, the consumers are forced to switch, potentially abandoning some or all of their application assets.

> **Both rearchitect and rebuild require greater commitment on the part of the development and operations teams to master both the patterns of cloud architecture and the cloud provider's specific support for them.**

When to Rebuild

This Decision Point presumes the rebuild option targets cloud-native PaaS environments that range from high-control to high-productivity. High-control environments, like Azure App Service or Amazon EKS, allow you to continue using standards-based programming languages and frameworks, with some limitations. By contrast, high-productivity environments, like Salesforce Lightning Platform, are totally proprietary.

Rebuilding an application on hpaPaaS services is most appropriate when rapid prototyping is crucial, or when the scope of an application is limited (e.g., limited by life span or limited to a specific audience). Another "sweet spot" for hpaPaaS is extending previous investment in a cloud platform (e.g., if the organization's customer data has already been migrated to Salesforce CRM, or if the company extensively uses G Suite as its productivity suite).

Rebuilding an application using one or a combination of CNAP, hcaPaaS, fPaaS and CaaS options is most appropriate when the development teams:

- Have complex delivery pipelines
- Need high control over the environment

- Have strict requirements for how the code is written and how the application interfaces are consumed

The following bullets explain when each platform is most appropriate:

- **Rebuilding on a CNAP** is necessary when you require platform-level portability between the cloud and on-premises, but do require platform ops capabilities.
- **Rebuilding on an hcaPaaS** is best when:
  - You are bound to the most common language runtimes (i.e., Node.js, .NET Core and Java).
  - You are not concerned about working with — and hard-coding your build, packaging and deployment workflows to — the cloud provider's APIs.
- **Rebuilding on an fPaaS** is appropriate for only the portions of your application that can fit within the constraints of an fPaaS. These constraints include time limits on sessions, CPU and memory limitations, and cost limitations. fPaaS is an ideal option for single-purpose code modules that focus on solving a single problem. fPaaS also works well when you are plugging directly into the cloud provider's eventing infrastructure to respond to, and act upon, events throughout the system. (See "Enhancing Operations Automation With Serverless Computing." (https://www.gartner.com/document/code/355825?ref=grbody&refval=3893681))
- **Rebuilding on CaaS** is appropriate when developers require a high degree of control and flexibility in configuring their target systems, including the OS, language runtime, frameworks and dependencies. Teams should have strong competency in containerization, building and deploying applications in containers, and the ecosystem of container technologies and tooling, including Docker, Kubernetes, Helm and container registries.

Rebuilding an application for a proprietary PaaS should be ruled out when the risk of vendor lock-in is considered unacceptably high. If rapid time to market for the application is paramount, then rebuilding complex functionality for a new application runtime is not a good choice. Even when you target a rebuild with standards-based languages, the degree of development required makes the project akin to starting over. You should expect to gain little, if any, reuse of existing code. In other words, if reuse is a high priority for your project, rebuilding may be a less-than-ideal alternative.

Replace

Table 7 concisely defines the replace migration option, and Figure 11 depicts it graphically.

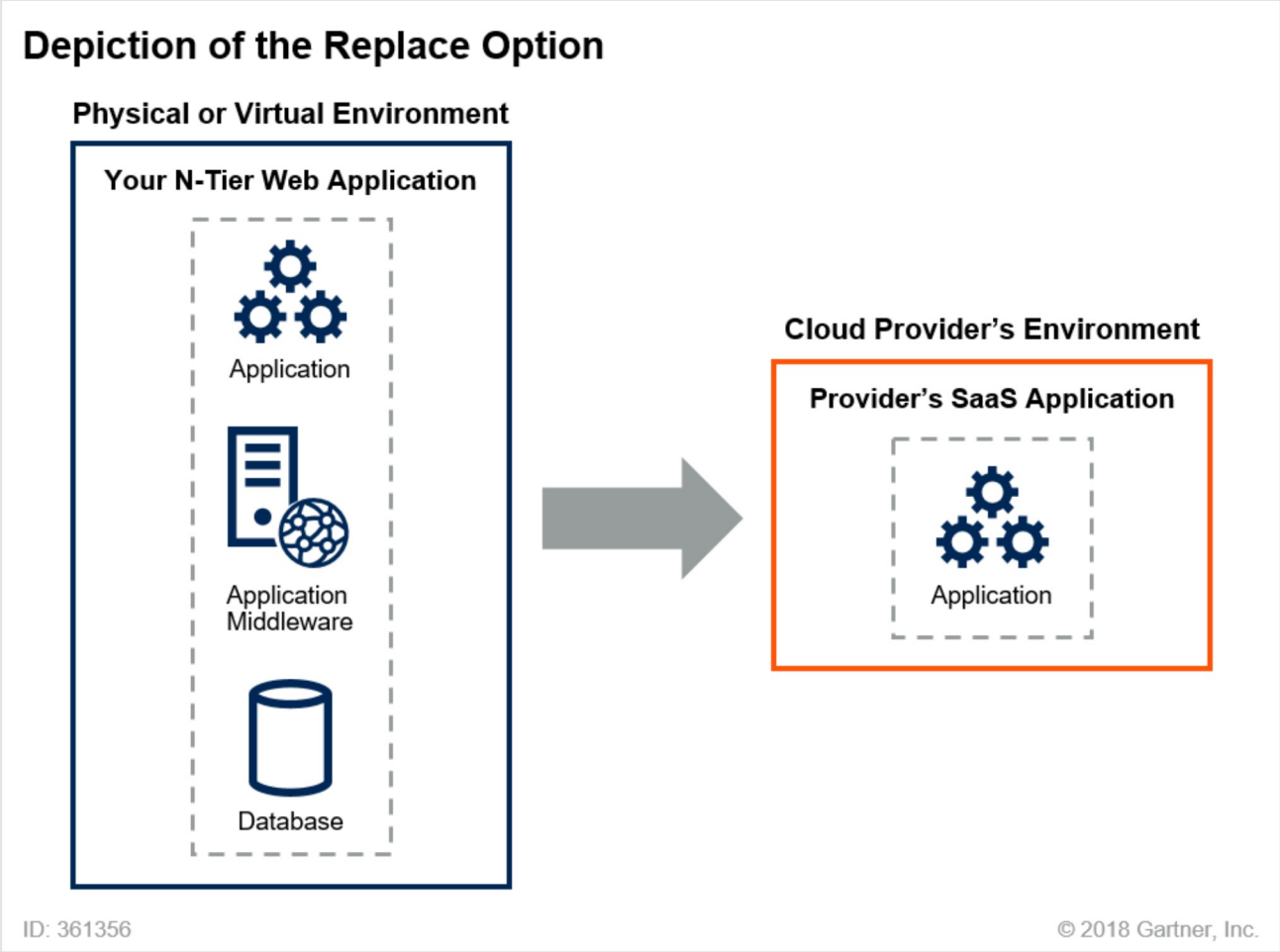
Table 7: Defining the Replace Migration Alternative

Replace Migration ↓	
Definition	■ You discard an existing application (or set of applications) and use commercial software delivered as a service (SaaS) to satisfy those business requirements. ■ Typically, existing data requires migration to the SaaS environment.
Cloud Platform Option	■ SaaS
What Services Are We Consuming?	■ The service is a complete, turnkey application solution (i.e., the IT organization does not build the solution, but may configure and integrate it). ■ Users access SaaS via a user-centric interface, such as a web browser or a mobile device. ■ Application data import/export is achieved with an API or a configuration/admin console.
Audience	■ Audience includes cloud architects and end users from both business units and IT. ■ Developers or DBAs perform initial configuration.

Replace Migration ↓	
Examples	■ Salesforce CRM and Microsoft Dynamics 365 for customer records management ■ Workday for HR processes ■ Adobe Analytics and Live Meeting for web conferencing ■ Office 365 for a productivity suite

Source: Gartner (November 2018)

Figure 11. Depiction of the Replace Option



Source: Gartner (November 2018)

Unlike custom-built software or single-tenant COTS software, SaaS providers can monitor and leverage group behavior within a single tenant (one department or company) or across many tenants. Although the benefits for the provider are obvious, there are both advantages and disadvantages for the customer. Customers benefit from better-informed product decisions that are driven by substantial data (rather than the focus groups and leading customer strategies of old). On the flip side, customers give up any notion of privacy in how they use the application, even if their data is encrypted and effectively invisible to the provider.

> **The provider knows what features you are using and when you are using them. This provides you with an advantage of being able to run analytics on your usage. However, at the same time, it opens you up to privacy risks because the cloud provider has the same visibility into your usage of the SaaS.**

Many of the typical advantages of buy-before-build sourcing apply to replacing applications with SaaS. Avoiding investment in mobilizing a

development team is a clear advantage when requirements for a business function change quickly.

Data within SaaS applications has a tendency to become immobile because it is difficult to interpret, access and use. The information model for the application is designed to be convenient to the vendor, and few SaaS applications include documentation on their logical data models. As a result, an additional set of data semantics must coalesce with existing meanings — of which there may be several — in an organization.

Furthermore, organizations may struggle to gain access to the data, even if it they comprehend it. Will data and processes within SaaS be usable within an organization's processes, workflows and analytical environments without data transformations that hinder performance? Latency may force the use of data extracts, file transports and data imports as the only means for reasonable data access. For more information on integrating applications in the cloud, see "How to Integrate Cloud-Hosted, SaaS and On-Premises Applications" (https://www.gartner.com/document/code/353947?ref=grbody&refval=3893681) and "Getting Started With Salesforce Sales Cloud Integration." (https://www.gartner.com/document/code/323137?ref=grbody&refval=3893681)

Unless a SaaS adoption plan includes a firm schedule for retiring replaced applications and the governance oversight to see it through, new software (even delivered as a service) adds more complexity and redundancy to the application portfolio landscape. For more information on validating the technical architecture of SaaS solutions, see "Evaluating Public Cloud SaaS Providers: Developing RFP Criteria." (https://www.gartner.com/document/code/273276?ref=grbody&refval=3893681)

Although SaaS can be extended through hpaPaaS, stretching SaaS to work beyond the basic applications that fit within the data and ecosystem of the provider's offering can be difficult. SaaS varies in the range of configuration and personalization possible for each tenant. A SaaS provider's ecosystem size and influence determine whether APIs and partner solutions can be used for heavier integration work. Unless the business processes replaced by the SaaS can be isolated, integration with existing systems and processes may be more difficult than with other migration options, because APIs may not be available, well-built, well-documented or usable. SaaS adoption often requires a change to internal business processes. SaaS is not a good option if:

- The process is unique or provides a competitive advantage.
- Business users are not willing to adopt a different process.
- The cost to adopt is simply too high, and a rebuild option is more cost-effective.

When to Replace

Financial control scenarios that favor SaaS include urgent cash flow and operational expense reduction requirements. Upfront costs are likely to be lower with SaaS than with alternatives, and providers often offer try-before-you-buy access, thereby providing application value more rapidly. SaaS also offers an opportunity to rapidly reassign or trim IT head count.

SaaS is also appropriate when demand for a commodity service (e.g., web conferencing and web analytics) can't be met in-house, or when the enterprise doesn't have the IT skills or resources to run complex software in-house.

Choosing SaaS is also a sensible strategy when business requirements change quickly, because SaaS avoids the involvement of a development team. However, the SaaS provider needs to be able to keep up with the changes you are releasing to your end users. For example, when a company is figuring out new conversational interfaces and AI/ML capabilities, prototyping different products can be valuable.

SaaS is best-avoided when the organization is unsure about the level of integration and customization the core feature set will require. Enterprises overlook crucial configuration and ecosystem requirements, limiting the partner solutions available 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 enterprise is not willing or able to modify its business processes.

Future Developments

Risks of platform lock-in exist at every tier of the cloud. Organizations should monitor standardization efforts and vendor support to enable portability of the following aspects:

- Containers
- Platforms
- VMs

- Code
- Data

Container Portability

The current trend in portability is driven by a finer-grained form of virtualization based on OS containers. This movement, popularized by Docker, has achieved enormous traction over the last four years (see "How to Prepare Your Enterprise for a Docker Containers Initiative" (https://www.gartner.com/document/code/344386?ref=grbody&refval=3893681) and "Orchestrating Container-Based Cloud and Microservice Applications" (https://www.gartner.com/document/code/344432?ref=grbody&refval=3893681)). It is bolstered by the emergence of a standards body, the Open Container Initiative (https://www.opencontainers.org/) (OCI), which defines a standard for container life cycle, runtime and file system structure.

The most widely adopted container runtime is the Docker engine, which is the most widely used in production. Additional container runtimes supported by the Container Runtime Interface (CRI) include containerd, OCI, frakti, CRI-O, and rkt. Containers are already supported in myriad PaaS frameworks (e.g., Cloud Foundry Container Runtime and OpenShift Kubernetes Distribution [OKD]) and container orchestration tools (such as Kubernetes and Azure Service Fabric). They have provided some promise in moving workloads from private to public clouds and across multiple cloud providers.

Platform Portability

There are application platforms that aim to provide end users with a cloudlike experience, regardless of where their infrastructure resides. These are called "cloud-native application platforms" (CNAPs). CNAPs are geared toward teams that build applications and associated dependencies on a single platform that can run on most any cloud infrastructure (e.g., Amazon EC2, Azure, Google Cloud Platform or VMware).

These platforms are primarily based on Kubernetes, but also come in the form of Cloud Foundry and Azure Service Fabric. CNAPs provide developers with common components that plug in with popular CI/CD toolchains. These components include:

- The packaging, build and deployment elements
- The runtime environment
- The APIs

CNAPs aim to enable developers to focus on writing and deploying code though the use of DevOps practices. CNAPs deliver this experience across a multitude of providers. (See "Comparing Leading Cloud-Native Application Platforms: Pivotal Cloud Foundry and Red Hat OpenShift" (https://www.gartner.com/document/code/370391?ref=grbody&refval=3893681) and "Assessing Microsoft Azure Service Fabric for DevOps and Microservice Architecture." (https://www.gartner.com/document/code/343217?ref=grbody&refval=3893681))

VM Portability

As discussed in the Rehost section of this Decision Point, organizations such as DMTF (http://dmtf.org/standards/vman) (formerly known as Distributed Management Task Force) are busily developing standards for the VM image formats and cloud management APIs submitted frequently by vendor coalitions. There are a variety of overlapping proposals to consider, including other open-cloud initiatives such as vCloud (https://www.vmware.com/products/vcloud-suite.html) and OpenStack (http://openstack.org/) , and standards efforts are fragmented. Without Amazon's participation, these efforts are valiant, but represent only a minority of the industry, and may not have significant impact on enterprise decision making.

Code Portability

Deploying applications across multiple cloud environments will become a driver and a differentiator. Standards efforts presently focus on IaaS and data portability issues. In the absence of open standards, progress on code portability tends to be limited to coalitions of vendors and service providers, such as the relationships forming around Cloud Foundry and OpenShift. Portability is such a hot adoption issue for organizations that proprietary PaaS offerings with high lock-in risk will not survive. This will force some innovative pure plays out of the market.

Data Portability

The Open Data Protocol (OData (http://www.odata.org/)) pioneered by Microsoft offers a web protocol for querying and updating data using HTTP, JavaScript Object Notation (JSON) and Atom Publishing Protocol (AtomPub).

Data access based on standards, interoperable or not, does not solve data portability issues, because access does not concern itself with data semantics. For example, if an organization wants to move its application back on-premises or to another provider, the following questions are pertinent:

- Can the organization get all of its data out of the provider's data store in a reasonable time and at a reasonable cost?
- Will it still understand that data?
- How many of the data's original relationships are preserved?
- Can the organization's current provider supply useful metadata (e.g., logical or physical data schema)?
- Now that the organization has migrated away from the provider, can it prove that its data has been deleted?

Example Implementations of This Framework

There is an example implementation of this framework that Northrop Grumman (https://urldefense.proofpoint.com/v2/url?u=http-3A__www.ngc.com_&d=DwMFAg&c=qRq7a-87GiVVW7v8KD1gdQ&r=pAN7mdy_ZcA4SRR91dGOJTu4EOYl_G5KHSbQEYnMUFA&m=x5xqc9O_XXDKOnOehgbLBaMbSqCpZjlNcFCY7SJuNTE&s=hYwL4mH3EZMFDtvpIE-u7m6DHSxQ3AjnOcAk_f_Rl_c&e=) produced called MApps2Cloud. The MApps2Cloud solution is an application and prescriptive guidance for securely and affordably migrating mission-critical applications to the cloud. This web-based application software helps organizations assess their portfolio of applications for readiness, profiling, security and cost, and develop a cloud migration plan. MApps2Cloud then offers a phased approach, outlining every step necessary to pilot, migrate, operate and manage applications in the cloud.

Another example is AWS's implementation (https://aws.amazon.com/cloud-migration/#application-migration-strategies) , which evolves the cloud migration process to "6 R's" and is a subcomponent to the Cloud Adoption Framework (CAF).

To quote the AWS literature on this framework:

> *"Organizations usually begin to think about how they will migrate an application during Phase 2 of the migration process. This is when you determine what is in your environment and the migration strategy for each application. The six approaches detailed below are common migration strategies employed and build upon 'The 5 R's' that Gartner outlined in 2011."*

Evidence

[1] **2017 Gartner Public Cloud Adoption Strategies Survey.** This research was conducted via an online survey from 9 August through 18 August 2017 among members of the Gartner Research Circle — a Gartner-managed panel composed of IT and business leaders. In total, 104 members completed the survey. Gartner Research Circle IT and IT-business members were invited to participate. The survey was developed collaboratively by a team of Gartner analysts, and was reviewed, tested and administered by Gartner's Research Data and Analytics team. The results of this study are representative of the respondent base and not necessarily the market as a whole.

Document Revision History

Decision Point for Choosing a Cloud Application Migration Strategy - 29 March 2016 (https://www.gartner.com/document/code/299768?ref=ddrec)

Recommended by the Author

Architecting Low-Code Cloud Applications With High-Productivity aPaaS (https://www.gartner.com/document/3883868?ref=ddrec&refval=3893681)

How to Assess Your Application to Adopt Cloud-Native Architecture (https://www.gartner.com/document/3813563?ref=ddrec&refval=3893681)

A Guidance Framework for Architecting Highly Available Cloud-Native Applications (https://www.gartner.com/document/3396654?ref=ddrec&refval=3893681)

Comparing Tools to Track Spend and Control Costs in the Public Cloud (https://www.gartner.com/document/3769138?ref=ddrec&refval=3893681)

How to Build Cloud-Native Applications Using Serverless PaaS (https://www.gartner.com/document/3892464?ref=ddrec&refval=3893681)

Adding Serverless Computing and fPaaS to Your Cloud-Native Architecture Toolbox (https://www.gartner.com/document/3587220?ref=ddrec&refval=3893681)

Evaluating Public Cloud SaaS Providers: Developing RFP Criteria (https://www.gartner.com/document/3062717?ref=ddrec&refval=3893681)

Comparing Leading Cloud-Native Application Platforms: Pivotal Cloud Foundry and Red Hat OpenShift (https://www.gartner.com/document/3898470?ref=ddrec&refval=3893681)

Recommended For You

Foundations of a Production-Grade Public Cloud IaaS and PaaS Architecture (https://www.gartner.com/document/3878720?ref=ddrec&refval=3893681)

2019 Planning Guide for Cloud Computing (https://www.gartner.com/document/3891095?ref=ddrec&refval=3893681)

Key Services Differences Between AWS, Azure and GCP: Compute and Storage (https://www.gartner.com/document/3891235?ref=ddrec&refval=3893681)

Cloud Computing Primer for 2019 (https://www.gartner.com/document/3899266?ref=ddrec&refval=3893681)