# Cloud-Native Applications and Cloud Migration

## The Good, the Bad, and the Points Between

Cloud-native features in your information technology (it) systems come with many advantages, but they also come with a cost. The costs vary greatly, based upon your applications and data. Sometimes being cloud native doesn't make economic sense, and sometimes it does.

Keep your eye on that ball as you migrate to the cloud.

A global 2000 company or a government agency typically has more than 5,000 applications on legacy platforms. These applications are placed into categories: ones that should move to the cloud and ones that should not. If they move to the cloud, then do they need to be altered to leverage cloud-native features (refactoring), or they could be moved with few or no changes (lift and shift)?

As you can see in Figure 1, an application can take many paths on its migration to the cloud. They're called the 7 R's of migration. This includes lift and shift (rehosting), or moving the applications with few or no changes. Partial refactoring means changing a small percentage of the code to leverage some cloud-native features. Complete refactoring means rewriting most of the applications so that they become cloud-native applications.

You need to pick a path from one of the 7 R's for each application, based upon its immediate, as well as long-term, return on investment. In some cases, you need to break applications apart into reusable components, and in other cases, applications can be replaced with a software as a service (SaaS) application analog. Some applications need to be removed, and many applications should stay put.

The toughest part of application migration to the cloud is figuring out the correct path. The retire path is usually obvious from the start. If applications are movable to the cloud, how should they be configured, changed, or not? Enterprises still struggle with these issues, but best practices are starting to emerge.

### Why Go Native

The pros of going to cloud-native features include the following:

- **Performance.** You'll typically be able to access the native features of public cloud services to

EDITOR
DAVID S. LINTHICUM
*david@davidlinthicum.com*

 2325-6095/17/$33.00 © 2017 IEEE

provide better performance than is possible with nonnative features. For example, you can deal with an input/output (I/O) system that works with autoscaling and load-balancing features.

- **Efficiency.** Cloud-native applications' use of cloud-native features and application programming interfaces (APIs) should provide more efficient use of underlying resources. That translates to better performance and/or lower operating costs.
- **Cost.** Applications that are more efficient typically cost less to run. Cloud providers send you a monthly bill based upon the amount of resources consumed, so if you can do more with less, you save on dollars spent.
- **Scalability.** Because you write the applications to the native cloud interfaces, you have direct access to the autoscaling and load-balancing features of the cloud platform.

The price you pay for these advantages is portability. Applications that are localized for specific cloud platforms are not easily ported to other cloud platforms. Doing so takes a great deal of rewriting or refactoring of the code. For all practical purposes, you are locked into that cloud platform.

If applications run on the target cloud platform for years, you're bound to get your investment back in code changes and testing. You have to look at the advantages of going cloud native case by case and application by application.

Unfortunately, there are no premade checklists you can use to make these assessments. You simply have to get as smart as you can about the trade-offs and make the best decisions you can (https://www.cloudtp.com/doppler/pros-cons-going-cloud-native/).

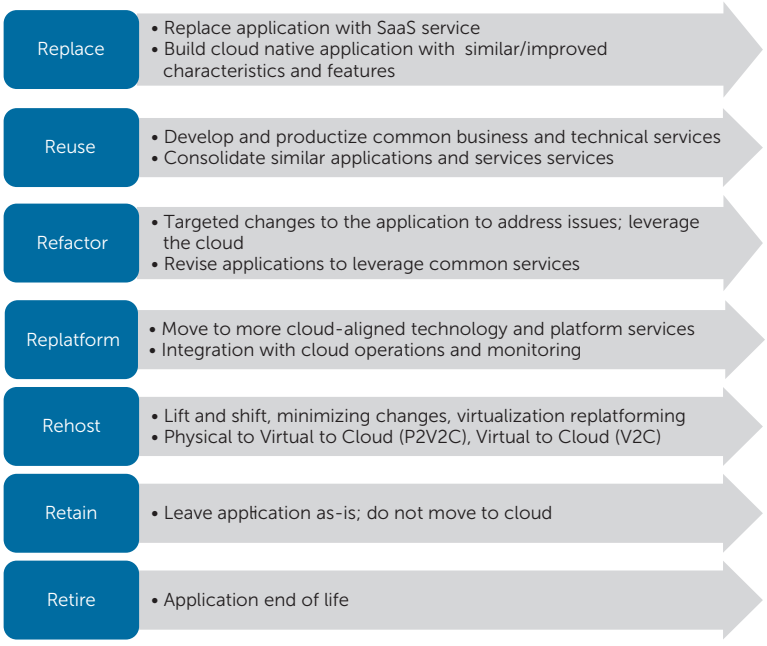


FIGURE 1. The 7 R's of cloud migration. Source: Cloud Technology Partners

## A General Approach to Becoming Cloud Native

To take proper advantage of a cloud platform, including infrastructure as a service and platform as a service (PaaS), you have to design the applications so that they're decoupled from any specific physical resource. For example, if you access I/O directly from a platform as Linux, you need to access the cloud's abstraction layer, or its native APIs.

Clouds can provide an abstraction or virtualization layer between the application and the underlying physical (or virtual) resources, whether they're designed for cloud or not. But that's not good enough. If you're truly going cloud native, you need to directly control the native resources.

When this architecture is considered in the design, development, and deployment of an application, the utilization of the underlying cloud resources can be as much as 70 percent more efficient. This cloud computing efficiency equals money. You're paying for the resources you use, so applications that work more efficiently with those resources run faster and generate smaller cloud service bills at the end of the month. Several cloud-cost governance tools can monitor these costs for you. I highly recommend using one or more of these tools if you plan to spend the time and money to move and refactor applications to be cloud native.

## The Downsides of Cloud Native

There are trade-offs to cloud-native application development and design. Be aware of the trade-offs before you bind your application to a specific cloud. You should do this with the heart of a chief financial officer, not a technologist. At the end of the day, this is about costs, not what is technically superior.

The biggest trade-off, as we covered already, is that you're giving up portability for the advantages of being cloud native. Applications that are localized or made native for specific cloud platforms are not portable to other cloud

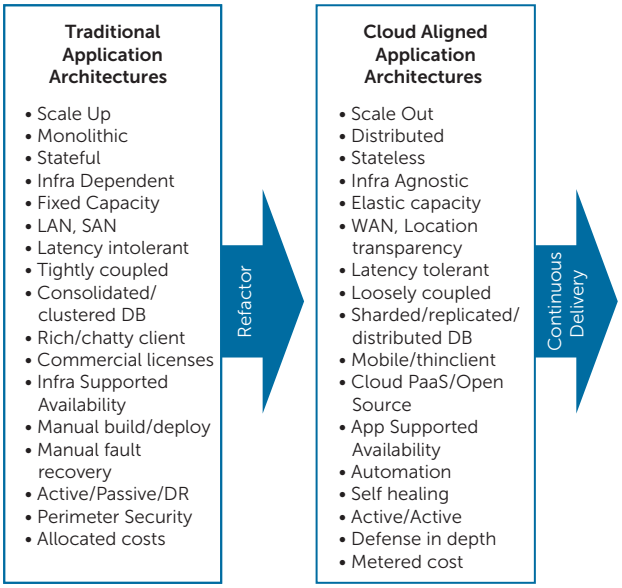


FIGURE 2. Being cloud native means more than changing code. You need to change your application architecture as well. Source: Cloud Technology Partners.

platforms without a great deal of rewriting or refactoring of the code.

That being said, few cloud-to-cloud migrations have happened yet. This is true even when rehosting or lift-and-shift migration paths are chosen. If the current trends continue, the argument against lock-in does not seem to be much of an issue for most enterprises. But you should at least keep this issue in the back of your mind.

Cloud lock-in does expose the enterprise to some risk if the cloud platform provider becomes difficult to deal with, raises prices, or most likely, some of the services you've purchased and use get turned off. In those cases, you're going to have to bite the bullet and re-refactor those applications to be cloud native on another cloud.

Some developers are getting good at placing the cloud-native features of an application into a specific domain of the application design. This allows them to more easily change the application for other cloud platforms, if needed. These good application design practices are not common; most developers exploit the cloud-native interfaces systemic to the application. Those are harder to change.

My best recommendation is that you at least consider cloud-native application development approaches and best practices—if not for the efficiency, then perhaps for cost or performance. Although there is a clear trade-off in portability, the benefits seem to outweigh that cost quickly if you're going to run the application for more than a couple of years.

## Emerging Architectures

Cloud native is not about just changing the code to follow the features of a specific cloud; it's also about changing your approach to architecture design. As you can see in Figure 2, what's emerging is a world of cloud-aligned application architectures.

These cloud-aligned architectures can autoscale and are distributed, stateless, and loosely coupled, to name a few features. If you want to make applications truly cloud native, the architecture must be rethought before you begin refactoring the code.

Does this new approach to application architecture suck resources and money, as well as adding a great deal of risk? Yes. However, the risk-reward scale typically leans to the reward side if the life of an application is 10–15 years (which it is for most enterprises). The effort of both rearchitecture and refactoring for an application with long-term use will pay for itself many times over.

However, enterprises in the US are not wired for long-term investiture. Most enterprises opt for lift and shift versus refactoring for cloud native. Enterprises are rewarded based upon earnings, and the lower the cost of new software development and IT, the more earnings they can claim—short term.

## Do the Right Thing

The evidence is compelling for the cloud-native application path when migrating applications to the cloud. The benefits outweigh the costs for most applications that are picked to be moved to the cloud, but given that refactoring costs 30 times simple rehosting, enterprises are reluctant to jump in with both feet.

So, this will be another learning process. Much like we saw when we made applications platform native, such as Win32- and Unix/Linux-native APIs, we had to fail first. In other words, working around the native platform features led to applications that did not live up to the expectations of the business, and IT had to go back to square one to redesign and refactor the applications to meet the needs of the business.

I suspect we'll follow the same patterns here. In a few years, cloud native will become the best practice. However, that won't happen until we fall on our faces a few more times. Some things never change.