DRAFT

Achieving True Hybrid Cloud By Building Applications For Cloud Portability

Without Compromising on the Least Common Denominator

Nati Shalom | @natishalom
July 2016

ABSTRACT

Hybrid cloud, what once was perceived as virtually mission impossible, is becoming pretty much mainstream. According to this survey, and this survey, some users are currently running on as many as **6 clouds** simultaneously on average per organization, with an even split between private and public clouds, both for real deployments as well as for experimentation, and 74% of enterprises are currently leveraging two or more cloud infrastructure vendors; making the need for robust cloud portability mission critical, not just nice to have.

In this white paper, I will tackle the concepts of hybrid cloud, cloud portability, and the compromise many organizations have to make when relying on a single point of abstraction at the infrastructure layer. I will provide use cases for cloud portability and then offer a real approach, using an open source tool, for building cloud applications with portability in mind so that you retain full control of your stack.

TABLE OF CONTENTS

INTRODUCTION

From my experience, when people refer to hybrid cloud, they often times aren't necessarily referring to the same thing. With the diversity of use cases for hybrid cloud, where each one of them drives a different strategy or approach, it is difficult to lump hybrid cloud into one simplistic category.

On top of this, most of the discussion, and even the tooling built for this purpose, is focused on bridging specific aspects of the infrastructure, e.g. compute and networking as an example, and quite often caters to a wider market, forcing a "least common denominator" approach which inherently limits the use of the underlying cloud (I will get into this more below).

In addition, many of the existing tools are missing the key part in such deployment models; the actual application itself. Running an application in a hybrid cloud environment requires the handling of the entire application stack in which the infrastructure is really only one component. This includes the configuration management, containers, monitoring, logging, and policies as well as maintenance of the application itself through its entire lifecycle.

I find myself writing a blog post on just this subject every couple of months, since the landscape changes so rapidly (see this disruption cycle post if you want more on that). That said, recent developments have gotten me thinking again, and I wanted to revisit the different hybrid cloud use cases and suggest a different approach to hybrid cloud that doesn't force a least common denominator and handles the entire application lifecycle and stack.

So just to set the stage, and make sure we're all on the same page, let's start with the obvious - the definitions of hybrid cloud and cloud portability and identifying the diversity of use cases.

HYBRID CLOUD DEFINED

Here is what the experts say is the definition of hybrid cloud:

> "Hybrid cloud is the consolidated coordination/management of multiple cloud services (onsite private cloud, dedicated hosted/offsite private cloud, and/or public cloud)."
>
> *– IDC Cloud Deployment Taxonomy*

Hybrid cloud is, in simple terms, the use of multiple clouds simultaneously, where cloud portability is the ***enabler*** of this deployment model.

CLOUD PORTABILITY DEFINED

Cloud portability is the ability to run the same application on multiple cloud infrastructures, private or public. This is basically what makes hybrid cloud possible.

THE DISTINCTION BETWEEN HYBRID CLOUD AND CLOUD PORTABILITY

The distinction between these two ideas is important, as in the case of hybrid cloud we're talking about multiple clouds attempting to act as one unified infrastructure, and with cloud portability, we're basically talking about the option to run on multiple clouds, but not necessarily at the same time.

CLOUD PORTABILITY USE CASES

We often tend to associate cloud portability with cloud bursting, and many times it's even used erroneously interchangeably with hybrid cloud, but these only represent a couple of use cases, and, truthfully, not even the most common ones. In fact, the need for cloud portability spans across a vastly wider number of use cases that are much more common, but at the same time less known, ironically. Let me explain with the following:

- **Future proofing** - With the uncertainty around VMware or OpenStack and the emergence of a new class of cloud native infrastructure it has become abundantly clear that we're going to continue to experience more disruption on the infrastructure level. In order for the strategy to "future proof" your application from those changes and keep your options open to benefit from new developments as they happen, it is important to decouple the application from the underlying infrastructure by designing the application for cloud portability.

- **Application deployment portability** - Many software vendors that develop software applications need to allow application portability to give their customers a simple way to provision and deploy their software products on their cloud of choice. In this context, cloud portability can be analogous to

operating system portability between Windows, Linux and Mac or even mobile app portability across iOS and Android. Cloud represents a market and by designing for portability you maximize the reach of your products to those markets.

- **Same application across multiple clouds -** The previous use case describes a situation in which we allow portability at deployment time, i.e. users are able to choose the target environment for deploying their application, but once they have completed the moving process from that environment, it would be considered a completely separate deployment.

There are a number of cases in which the same application would need to span its resources and services across multiple clouds at the same time. Here are a few:

- **Cloud Bursting** - Probably the most common use case for spanning application resources across clouds is known as cloud bursting. This use case is aimed to handle the need for on-demand access capacity and optimize the cost of those resources by allowing to run on a fixed pool of resources during the steady state periods and span to on-demand cloud resources during peak loads.

- **Migration** - Another lesser known use case for cloud portability is cloud migration. A common example would be an organization that is migrating from their VMware environment into OpenStack or from private cloud into public cloud. In this case, portability allows you to smooth out the process and reduce risk by providing a common management layer across the two environments, thus allowing the organization to selectively transition the application between the two environments while at the same time manage them as one.

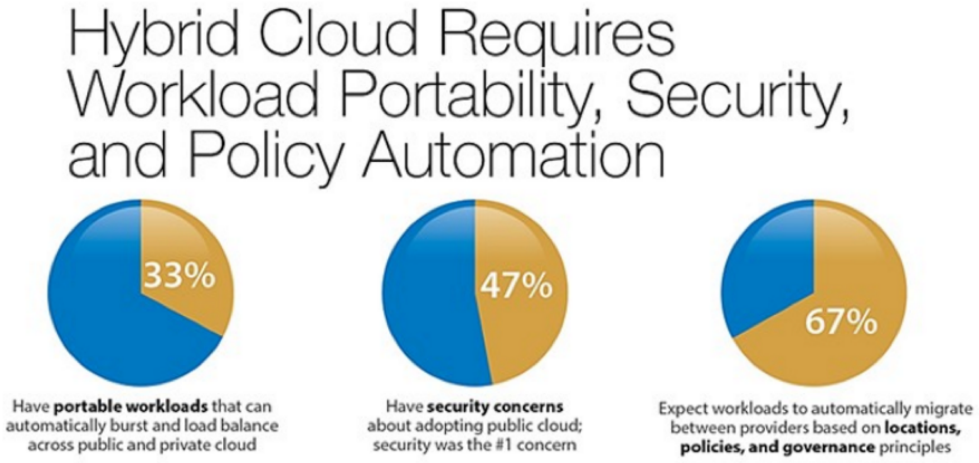
- **Portability between the same cloud versions** - Another lesser known, but probably the most common, cloud portability use case is the move between

versions of the same infrastructure. One of the common strategies to allow upgrade of the infrastructure is to create new instances (cloud sites) of a newer version, and then gradually transition apps onto this new version. Cloud portability make that process simpler as it decouples the application from the infrastructure, and in this way from the changes between versions.

THE LEAST COMMON DENOMINATOR APPROACH

The number of use cases that would benefit from cloud portability could be fairly vast. As noted above, though, the reality is that many of the existing solutions for cloud portability are fairly limited and are not well suited to fit into all of those use cases.

One of the main reasons for this is that most solutions take a least common denominator approach in which they rely on a common layer of API abstraction (mostly around the compute API and to a lesser degree storage and networking) across all clouds and by doing so force limited use of the underlying cloud infrastructure.



Cloud is much more than Compute, Storage, Network

The common API abstraction already limits itself to Compute, Storage and Network and even at that layer the abstraction tends to be fairly simplistic and quite often doesn't expose many of the more advanced features of the underlying infrastructure, and there are many exciting features constantly being rolled out in the cloud sphere.

In addition to features, cloud infrastructure today provides a rich set of services such as database services, analytics services, LBaaS, you name it... the list goes on, that just cannot be easily abstracted.

The result is that relying on this layer of abstraction comes with a high toll of compromising on the least common denominator, one size fits all model, and thus losing many of the benefits that modern clouds provide today. And we are rarely one size fits all.

In my next post, I'll dive into how to achieve true cloud portability without forgoing all of the benefits hybrid cloud deployments actually make possible.

AN ALTERNATIVE APPROACH TO CLOUD PORTABILITY

The One of the use cases I previously mentioned for allowing application deployment portability to an environment, that doesn't conform to the same set of features and APIs, is iOS and Android. With operating systems, we see that software providers are able to successfully solve the portability aspect without forcing a common abstraction.

What can we learn about cloud portability from the iOS/Android use case?

Treat portability differently between the application consumer and the application owner - One of the main observation from the iOS/Android case is that, while the consumer is often completely abstracted from the differences between the two platforms, the application developer is not abstracted and often needs to treat each platform differently and sometimes even duplicate certain aspects of the application's components and logic to suit the underlying environment. The application owner, therefore, has the incentive to support and even invest in portability as this increases the application's overall market reach.

Minimizing the differences, not eliminating them - While the application owner has more incentive to support each platform natively, it is important to use cloud portability as a framework that will allow for minimizing but not eliminating the differences to allow simpler development and maintenance.

The main lesson from this use case is that, to achieve a similar degree of cloud portability, we need to make a distinction between the application consumer and the application owner. For cloud portability, in order to ensure a native experience for the application consumer, we need to assume that the application owner will be required to duplicate their integration effort per target cloud.

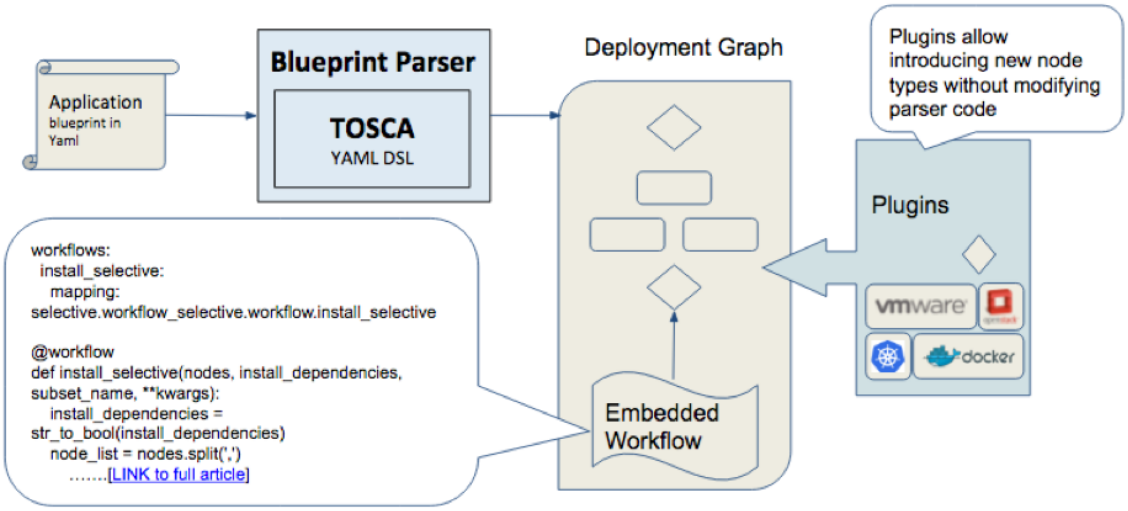
This is the same approach we should take with cloud application portability!

CLOUD PORTABILITY WITH ARIA

In this section, I will refer to this specific project as a means by which to illustrate the principles that I mentioned above in more concrete terms.

Project ARIA is a new Apache-licensed project that provide simple, zero footprint multi-cloud orchestration based on TOSCA. It was built originally as the core orchestration for Cloudify and is now an independent project.

The diagram below provides an inside look at the ARIA architecture.



There are three pillars, upon which ARIA is built, that are needed to manage the entire stack and lifecycle of an application:

1) An infrastructure-neutral, easily extensible templating language
2) Cloud plugins
3) Workflows

TOSCA Templating Language vs. API Abstraction

ARIA utilizes the TOSCA templating language in its application blueprints which provides a means for deploying and orchestrating a single application on multiple infrastructures through individual plugins, thereby circumventing the need for a single abstraction layer.

Templating languages, such as TOSCA, provide far greater flexibility for abstraction than API abstraction as it allows easy extensibility and customization without the need to develop or change the underlying implementation code. This is done by mapping the underlying cloud API into types and allowing the user to define the way it accesses and uses those types through scripts.

With Cloudify, we chose to use TOSCA as the templating language because of its inherent infrastructure-neutral design as well as being designed as a DSL which has lots of the characteristics of a language that utilizes the support of inheritance, interfaces and a strong typing system.

Cloud Plugins

Built-in plugins for a wide range of cloud services provide out of the box integration points with the most common of these services, but unlike the least common denominator approach (i.e. a single API abstraction layer), they can be easily extended to support any cloud service.

Workflows

Workflows enable interaction with the deployment graph and provide another way to abstract common cloud operational tasks such as upgrades, snapshots, scaling, etc.

PUTTING IT ALL TOGETHER

By combining the three aforementioned elements, the user is given a set of building blocks for managing the entire application stack and its lifecycle. It also

provides a richer degree of flexibility that allows users to define their own degree of abstraction per use case or application.

In this manner, cloud portability is achievable without the need to change your underlying code, and, in doing so, you enable true hybrid cloud

For more information, please contact marketing@getcloudify.org.