Multi-Cloud: You keep using that word...

It isn't surprising in 2019 how many times I bump into an environment or organization where the word multi-cloud comes up. Technology presents us with lots of architectural choices that often get very buzzword-centric. Multi-cloud leads the pack on popularity and buzz wordiness. Multi-cloud also continues to be a strangely ill-defined even at this point in the evolution of our cloud adoption in the industry.

Multi-Cloud 2012

In 2012 it was all about whether IT teams would be looking at the newly available public cloud alternatives to solve the woes of bursting for resource needs. That was the most often pitched use-case that we used to Devine what a "hybrid cloud" was. Vendors and architects showed exciting diagrams of on-premises applications that would suddenly burst out to the cloud to scale when demand increased. Sounded good at the time.

The issues we would face became very obvious, very quickly.

- Where is your data for that bursting application?
- How do you define and deploy networks that securely span the hybrid platform?
- What applications do you have today that can leverage this architectural pattern?

Multi-cloud was right on the heels of the hybrid cloud story because the "big 3" players (AWS, Azure, Google Cloud) were all pushing to become the target for your workloads. This meant that you could now run your bursting application across more than one cloud.

This new multi-cloud idea was fast becoming the classic "solution looking for a problem". It was technically interesting (read: challenging) and we begin hearing stories of Netflix and others who were taking advantage of the burstable options in and across clouds.

The questions we needed to ask began to solidify:

- Is cloud lock-in a concern? In other words, does being beholden to a single cloud provider put you at risk both technically and in your negotiation position?
- Are your engineering teams capable and eager to own the tooling to build for more than one cloud platform?
- Does the risk versus reward and ROI work out for a multi-cloud application scenario?

If you run single-purposed applications that are scale-out capable, the questions become easier to answer. Multi-cloud looked like a neat idea for most organizations, but a pipe dream at the same time. Most IT orgs weren't even widely deploying applications that could cross their own private data centers in an active-active scenario, let alone across multiple clouds.

Do a google image search for "multi-cloud 2012" and you'll see a treasure trove of presentations on the concepts and not a lot of live use-cases in production.

Multi-Cloud 2019

We've come a long way. More technical solutions arrived in the ensuing years to open the door to real potential of a legitimate multi-cloud deployment. Containerization and container schedulers

(read: Kubernetes) now make the underlayers virtually invisible. So, did that application that runs on one cloud and burst to the second cloud turn out to be the right use-case? Nope! Time has proven that use-case to be as long-lasting as the Palm Pilot. Good idea that didn't execute. But that's fine.

Orchestration and infrastructure-as-code with platforms like Puppet, Chef, RackN, Terraform, and others, truly open up the possibilities of architecting and deploying for any underlying platform and your developers can build their applications on higher abstractions that can finally span these clouds.

What Multi-Cloud Is and Is Not

It still does not really make sense to span your applications or burst your applications across cloud platforms in most cases. Despite the hype and technical capabilities these use-cases haven't proven out:

- Single application that spans across more than one cloud provider
- Scale-out apps with data

Why didn't these play out like the marketing messages of yesteryear would have indicated they could? Pretty simply:

- Data gravity for the apps – front-end servers kept so far from the back end create latency…so what if you distribute the data?
- Data consistency in a globally-distributed, scale out database requires low-latency between nodes or apps designed for eventual consistency

Data seems to be one of the biggest markers for why we localize applications. We can shard out the data all we want, but a true enterprise application with high throughput that needs to be resilient and distributed also comes with the baggage of needing high throughput on low-latency with consistency across the entire application set.

Basically, the most needy application that caused us to make it resilient by spanning the cloud also causes us to fall to the lowest common denominator of data gravity. Web applications are one thing. They can work in a distributed fashion without as much pain and re-architecture. The reality is that distributed apps that are backed by large data sets will be more likely candidates to build resiliency within a cloud rather than across multiple clouds.

Before you begin throwing names like LinkedIn, and Netflix, as examples, remember that they are apps which are purpose built for multi-cloud and they even had to build up their own tooling to do so because the native cloud platforms didn't provide a way to do it. CI/CD across clouds is a whole other beast which opened up a new and exciting conversation of "can we versus should we?" as we looked at borrowing the toolsets that Netflix, Walmart, and others open sourced.

What does make sense as multi-cloud use cases is using the best-of-breed solution for each application and leveraging software-defined networking to access and share resources across clouds. The magic of VPNs and persistent networks across cloud providers now easily enables valuable scenarios:

- Active Directory on Azure as the authoritative IAM for AWS and on-premises Windows workloads – federate your IAM across all clouds and on-premises using a common platform and then use it to manage Windows servers across all of your cloud providers

- AWS-specific apps and Azure-specific apps – Why force developers to abandon services that could be purpose-built to solve a problem. Need Rekognition for one application set and also want to use the Azure ML services for another? Don't port to a single cloud to achieve alleged reduction in complexity when it means sacrificing better services to meet the needs of your apps and business
- Acquisitions of teams, apps, and companies that already have a large footprint on one or more clouds – A huge use-case to maintain a multi=cloud architecture is that you acquired an already established platform through an acquisition. Why would you spend needless effort porting and migrating to another cloud. The ROI most likely won't be in your favor

The multi-cloud reality is treating each cloud like it's own data center and choosing to interconnect those clouds for resiliency and to leverage the best features and capabilities of each. There will be some added complexity as we adopt tooling to deploy and manage these environments, but the benefit far outweighs the cost of effort in using multiple tools. Forget the single pane of glass and forget the multi-cloud spanning application. Let's embrace multi-cloud for what it really is which is a beautiful opportunity to stop and ask what we really want to accomplish with the underlying tech.