Separating Multi-Cloud Strategy from Hype:

An Objective Analysis of Arguments in Favor of Multi-Cloud

CloudAcademy

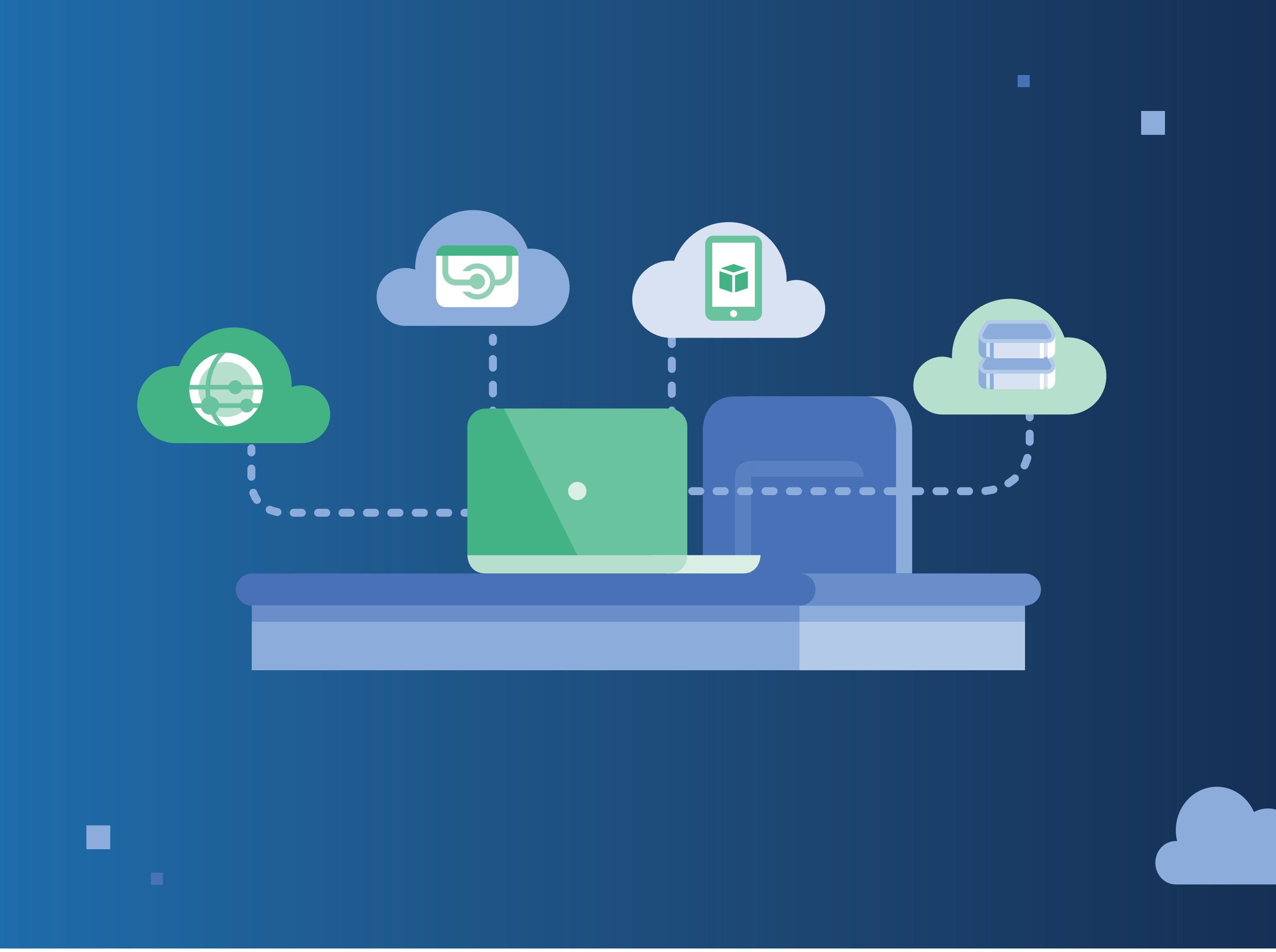
contents

written by Ben Lambert

Ben Lambert is the Director of Engineering and was previously the lead author for DevOps and Microsoft Azure training content at Cloud Academy. His courses and learning paths covered Cloud Ecosystem technologies such as DC/OS, configuration management tools, and containers. As a software engineer, Ben's experience includes building highly available web and mobile apps. He holds the following certifications: Microsoft Specialist: Architecting Microsoft Azure Solutions, Google Qualified Systems Operations Professional, and Google Cloud Platform Qualified Solution Developer.

When he's not building the first platform to run and measure enterprise transformation initiatives at Cloud Academy, he's hiking, camping, or creating video games.

A THING WITHOUT CLEAR DEFINITION WILL BE PERCEIVED THROUGH THE BIASED LENS OF ITS OBSERVER.



introduction

Depending on whom you ask, multi-cloud is either an essential enterprise strategy or a nonsense buzzword. Part of the reason for such opposing views is that we lack a complete definition of multi-cloud.

There is little controversy in stating that multi-cloud is "the simultaneous use of multiple public cloud vendors," but to what end, exactly? Many articles on multi-cloud superficially claim that multi-cloud is a strategy for avoiding vendor lock-in, for implementing high availability, for allowing teams to deploy to the best platform for their app, and the list goes on. Without any substance to these arguments, it can be difficult to determine if multi-cloud can live past its 15 minutes of fame as a buzzword.

One of our goals at Cloud Academy is to provide objective analysis of tools, techniques, platforms, and vendors. Anything we review should hold up under scrutiny regardless of brand, vendor, or marketing hype.

Of the many benefits often associated with multi-cloud, there are a few that are more commonly mentioned than others. Those will be the focus of this paper. By taking an objective look at its most frequently cited benefits, this paper sets out to establish an actionable definition of multi-cloud.

Each section that follows will inspect the arguments (or claims) that employing a multi-cloud strategy helps organizations:

- Avoid vendor lock-in
- Implement high availability
- Select best-fit technologies

A MULTI-CLOUD STRATEGY CAN HELP YOU AVOID VENDOR LOCK-IN,
BUT IT ISN'T A REQUIREMENT.

Vendor Lock-in

A common benefit ascribed to multi-cloud is the ability to avoid vendor lock-in. For some companies, avoiding vendor lock-in is a core business requirement, or a way to achieve greater portability for their applications.

With such portability, teams can move applications to another framework or platform, with less effort. Other organizations consider the loss of potential portability as an acceptable tradeoff for vendor-specific features that save time on initial development. Pursuing a strategy that avoids vendor lock-in at all costs does mean giving up on some functionality that is unique to the vendor.

In most cases, using multiple cloud providers is not required to avoid vendor lock-in. Applications do not need to run everywhere but can be designed to run on a single cloud provider and still avoid a high degree of lock-in.

Avoiding lock-in isn't a binary choice—it's about degrees of tolerance and design decisions. Teams running apps on a single cloud platform that don't rely on functionality unique to that vendor are the ones who, through disciplined design decisions, have mitigated potentially significant refactoring required to change providers.

If avoiding lock-in is a concern, then teams must work to abstract away the vendor-specific functionality. One example at the code level: access functionality such as blob storage through an interface that could be implemented using any storage back-end (local storage, S3, Azure Storage, Google Cloud Storage, among other options). In addition to the flexibility this provides during testing, this tactic makes it easier for developers to port to a new platform if needed.

Containers and container orchestration tools are additional abstraction layers that can aid in workload portability. Any technology decision represents some degree of lock-in, so organizations must weigh the pros and cons of depending too heavily on any single platform or tools.

**The bottom line:
A multi-cloud strategy
can help you avoid
vendor lock-in,
but it isn't a requirement.**

High Availability

A cloud outage is typically followed by a series of articles promoting multi-cloud deployments as a solution to avoiding downtime. This is one claim about multi-cloud that is technically feasible, but unnecessary and impractical.

In February 2017, Amazon S3 experienced a major service disruption in its US East region due to an engineer's typo. This impacted many companies that relied on S3, and specifically those that relied on S3 exclusively in the US East region. In this instance, replicating files on another cloud provider could have mitigated the effects of the disruption. Using a single provider with cross-region replication is another solution.

Workloads impacted by the S3 disruption fell into two categories:
(1) Workloads deemed "not mission critical"
(2) Workloads run by companies that lack sufficient architecture and chaos testing

The most severe impact was felt by companies who lacked robust architecture sufficient for testing. For these (and most) teams, the solution is not to add in the complexity of bidirectional cross-cloud replication, but to ensure that teams understand the technology and implement best practices.

For the sake of thoroughness, let's explore the technical feasibility of achieving high availability by using multiple cloud providers.

To achieve high availability for the same functionality implemented across different providers, teams need to follow the same rules for avoiding vendor lock-in. That means you will be limited to the features common to your selected platforms. At the individual service level, the differences between various cloud providers' implementations can create a lot of extra work in the form of abstraction layers.

At the application level, due to IaaS implementation differences across providers, containers could serve as a viable abstraction. This approach would require running the same container orchestrator on multiple platforms and limiting the use of underlying functionality (or accessing underlying functionality through a common interface). While using containers to run the same application across providers may be technically possible, the implementation is far from practical and will likely result in a high probability of outages caused by human error. The potential increase in errors may be caused by differences in how data is replicated and differences in the IaaS offerings themselves.

Finally, expect managing security to be a challenge for any single deployment across multiple public clouds. Setting up virtual networks, firewall rules, monitoring, logging, and identity and access management can be difficult and time consuming. Ensuring compliance across multiple providers adds a whole new level of complexity, especially at the rate that cloud providers release updates. Additional tooling, processes, and training will be required to ensure cross-platform consistency.

New tooling or processes should be added to solve problems, not side effects of other problems. Adding the tooling required to implement a multi-cloud deployment is solving a side effect of using multiple platforms to accomplish what could be done with a single platform.

**The bottom line:
Multi-cloud could theoretically solve certain high availability issues, but it's more likely to add undue complexity.**

THE UMBRELLA ARGUMENT THAT **MULTI-CLOUD ENABLES USE OF “BEST-FIT TECHNOLOGIES”** HAS QUANTIFIABLE VALUE.

Best-Fit Technology: Best Platform & Best APIs

Multi-cloud alone won't prevent vendor lock-in and is unlikely to aid in developing highly available systems. However, it is finding a place at a growing number of companies.

Two types of multi-cloud implementations are currently being used successfully in the enterprise:

Application- or team-driven: Deploying different applications to different platforms based on the needs of the application or team.

Task-driven: Using a front-end to coordinate calls to the best possible services for given tasks, typically in serverless applications.

BEST PLATFORM:

There are several reasons for deciding to deploy an application to one provider over another. These include cost, data sovereignty, team experience, ease of development, and ease of deployment.

Organizations are increasingly giving disparate teams flexibility to build and operate their applications as they see fit. Having this sort of flexibility can provide a lot of value because teams aren't artificially limited by an individual platform.

This added flexibility is not without its challenges. Each new platform adds to the amount of domain knowledge required for the company. Each adds to the overall attack surface that needs to be secured and to the toolset needed to build and deploy. Overall, using multiple platforms can add a lot of overhead that needs to be managed.

While allowing teams to choose the best platform for their application is becoming a more common practice, it should be paired with a careful evaluation process that considers the entire lifecycle of the application and team.

BEST APIS:

The default starting place for a new application used to be a framework such as Django, Rails, or Spark. Increasingly, companies are shifting application logic to the client-side and assembling the back-end with the best services available for each task.

Since leveraging multi-cloud in this scenario enables the selection of best-fit technology, it merits a review of the technologies enabling this shift.

A handful of technologies have facilitated this change including new JavaScript frameworks, JSON Web Tokens for stateless authentication, and container technologies.

First, the capability of JavaScript and mobile frameworks has grown. There are now dozens of JavaScript frameworks that enable developers to use well-established patterns to build feature-rich user interfaces. Many of these frameworks abstract away complexity, making development faster and easier. The same is true of mobile frameworks that now provide UI editors, debugging, and testing tools. Some mobile frameworks even allow for cross-platform compilation, making it easier to target different mobile platforms.

Second, JWT (JSON Web Tokens) facilitates stateless authentication. Session tokens have been used for years, however, stateful applications tend not to scale as easily as stateless. JWT makes it easy to implement single sign-on, which means all your services can use the same mechanism. This means that you can start incorporating the best service for a given task, regardless of where that service is located.

Third, containers such as lxc, Docker, and rkt are contributing to the shift toward client-side focused apps because they enable the use of microservices and serve as the basis for serverless technologies. Both containers and serverless offerings make it easy to get services into production across different cloud platforms.

Best-Fit Technology: Best Platform & Best APIs

Finally, there is an increasing availability of specialized, client-side consumable services for tasks such as authentication, machine learning, data storage, and payment processing. These services are managed by a third-party and are pre-built, enabling an à la carte development process in which teams can more easily evaluate, select, and implement the optimal services for each use case.

Together, these shifts have enabled apps that are UI-centric and that coordinate a series of APIs. UI-centric coordination generally avoids the latency that is inherent when services communicate directly with one another. Of course, there are instances when cross-service communication or the creation of wrapper services is required, but with the UI in charge of much of the logic, each service can remain an independent entity, largely unaware of the other services involved.

.

The bottom line: The umbrella argument that multi-cloud enables use of "best-fit technologies" has quantifiable value.

KEY TAKEAWAYS

- **Using multiple cloud platforms to avoid vendor lock-in is just as likely to add development and operational complexity.** To avoid a high degree of lock-in, avoid functionality that can't be easily replicated elsewhere. The costs to build, deploy, and operate the same components, services, or applications across multiple providers is high.

- **There are two sound arguments for multi-cloud corporate deployments:** best-fit platforms and best-fit services (the latter, for certain use cases).

- **Companies with multiple applications,** regardless of the number of teams managing them, benefit from being able to use the best platform for the application.

- **For companies that can shift more of their logic to the user interface,** best-fit services are becoming the norm.

- **The best-fit approach must be combined with best-fit decision making.** Teams working in a multi-cloud environment need ways to continually gain technical knowledge, hands-on experience, and the business context for the projects that are in flight.

conclusion

Multi-cloud is not the panacea as advertised, however there are two sound arguments for multi-cloud corporate deployments: best-fit platforms and best-fit services (the latter, for certain use cases).

First, multi-cloud enables teams to deploy different applications to different platforms based on the needs of the application or team.

Second, it can enable apps that are architected to coordinate from the front-end to consume best-fit services. The two patterns reviewed in this paper are currently the most common, though, it is likely that as higher-level abstractions are created, additional best-fit patterns will emerge.

In both cases, an essential element of your multi-cloud strategy will be the teams tasked with best-fit decision making. Evaluating platforms to determine fit based on workload or business requirements, and ensuring security and minimizing complexity across vendors and technologies requires deep cross-platform and best practice expertise.

Ongoing access to the most current technical knowledge, business context, and hands-on experience are what teams need to architect and select best-fit services.

As tools and technology evolve, so will this analysis. While a subset the purported benefits of multi-cloud can be classified as "hype" at this point in time, we can reasonably define multi-cloud as the use of multiple public clouds to enable enterprise teams to select best-fit platforms, services, and functionality.

CloudAcademy

Cloud Academy is the leading enterprise training platform that accelerates teams and digital transformation.

Companies trust Cloud Academy to deliver multimodal training on the leading clouds (AWS, Azure, Google Cloud Platform), on the essential methodologies needed to operate on and between clouds (DevOps, Security), and on the capabilities that are unlocked by the cloud (machine learning, IoT).

From the fundamentals to advanced scenario training, Cloud Academy empowers organizations with the knowledge, critical thinking, and hands-on experience needed to adopt, operate, and optimize the multi-cloud.

cloudacademy.com

UPDATED: 1/23/2018