appvia

Cloud Platforms- Build vs Buy

Deciding how your teams will deliver applications -
Build a bespoke environment or buy an off-the-shelf platform.

Table of Contents

Executive Summary

Migrating to the cloud means that you can keep pace with the competition, reduce operational risk and improve scale with the help of innovative cloud solutions. Cloud can be integral to your business, with the ability to package, ship and deploy applications using commodity services that can be delivered in minutes and “zero” upfront hardware investment. But it’s also a complex landscape that requires specialist knowledge to stitch together a developer workflow that enables software application iteration into production securely, simply and efficiently.

The question is: should you own the responsibility to build a platform for Developers and DevOps to underpin the business services or do you look to the industry to find a solution that meets your requirements?

If you’re starting your cloud journey for the first time it may seem like the cloud providers already have everything you need. However, even though there are many independent services you can consume, there’s no defined way of working for your engineering teams, no out-the-box automation stack you can adopt from the cloud vendor and no operating model that you can apply simply.

When you don’t have immediate scale and are in exploration mode, building organically around the developer requirements is the most natural process. You’ll soon find, however, that as requirements keep piling in from different business units, you’ll end up with an operational overhead in the team and a slew of technical debt in its wake. Are the technologies your teams and services are consuming production ready? Will the technologies meet the SLA’s requirements? And is the team structure and knowledge distribution in place to be able support them 24/7?

When deciding on what’s best for your organisation, consider whether you have the level of technical expertise and resource that is required to continuously and effectively manage the challenges of maintaining application infrastructure and tooling. And whether your business has the time and money to invest appropriately in building and integrating technologies to support and speed up the application delivery to give your business market edge.

It’s ultimately not just the initial build and maintenance of the infrastructure that you need to worry about, but selecting and maintaining the services that support and surround it.

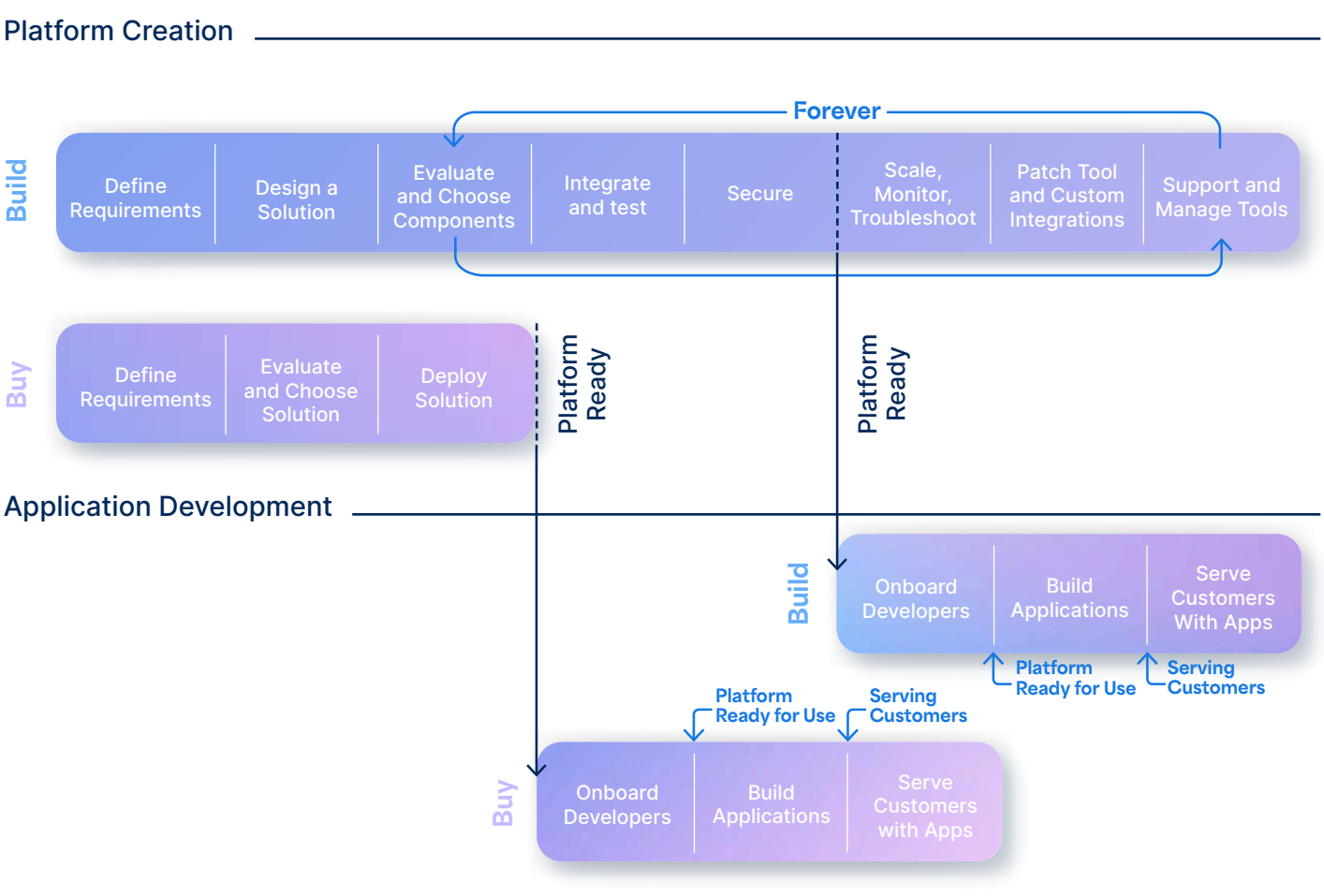
We ensure security is baked into every layer, every cluster, every account. And we make it possible for cloud to become the cost-effective catalyst for your business and technology innovation.

When you start to productionise services, or you need to scale teams, you'll need to design how to streamline delivery and what the relationship between cloud and your teams is, for example:

Technical Requirements for a Production App

1. Code Management & Access
2. Cloud Accounts for Teams (Not prod / Prod)
3. Infrastructure for Teams to Run Workloads
4. Environment & Access Management
5. Continuous Integration Setup
6. App Testing, Packaging & Deployment Setup
7. Certificates and Application Endpoints
8. Monitoring & Logging Solutions
9. Security Tools for Code and Infrastructure
10. Cost Management Tools for Teams

Workflow and Impact of Both Building and Buying



Approaches to Supporting the Development Lifecycle

There are several different ways you could approach supporting the software development lifecycle from an operational, security, delivery and tooling perspective.

Route 1 -

Build a central platform in-house:

A central platform team that builds its own platform to support the needs of developers. There's time spent working with teams, grabbing requirements, engineering and helping delivery.

Building a central platform in-house gives you better scalability with your resources, as the central platform tooling should be automating and providing a service back to teams. Doing that well, however, will require the right talent pool and a lot of engineering, which would result in a high cost of ownership going forward and delayed time to market.

Route 2 -

Project Based Approach, No Platform:

A technology stack is defined for the organisation, being used on a per project basis with no centricity on tooling. Time is only spent on project work, helping with delivery.

With a project based approach without a platform, you'll have a more tailored solution with more flexibility for projects, but that divergence will have a high resource overhead; you'll need to scale resources per project, have no reusability across project teams and have custom engineering hidden across project silos, which results in a high security risk profile.

Route 3 -

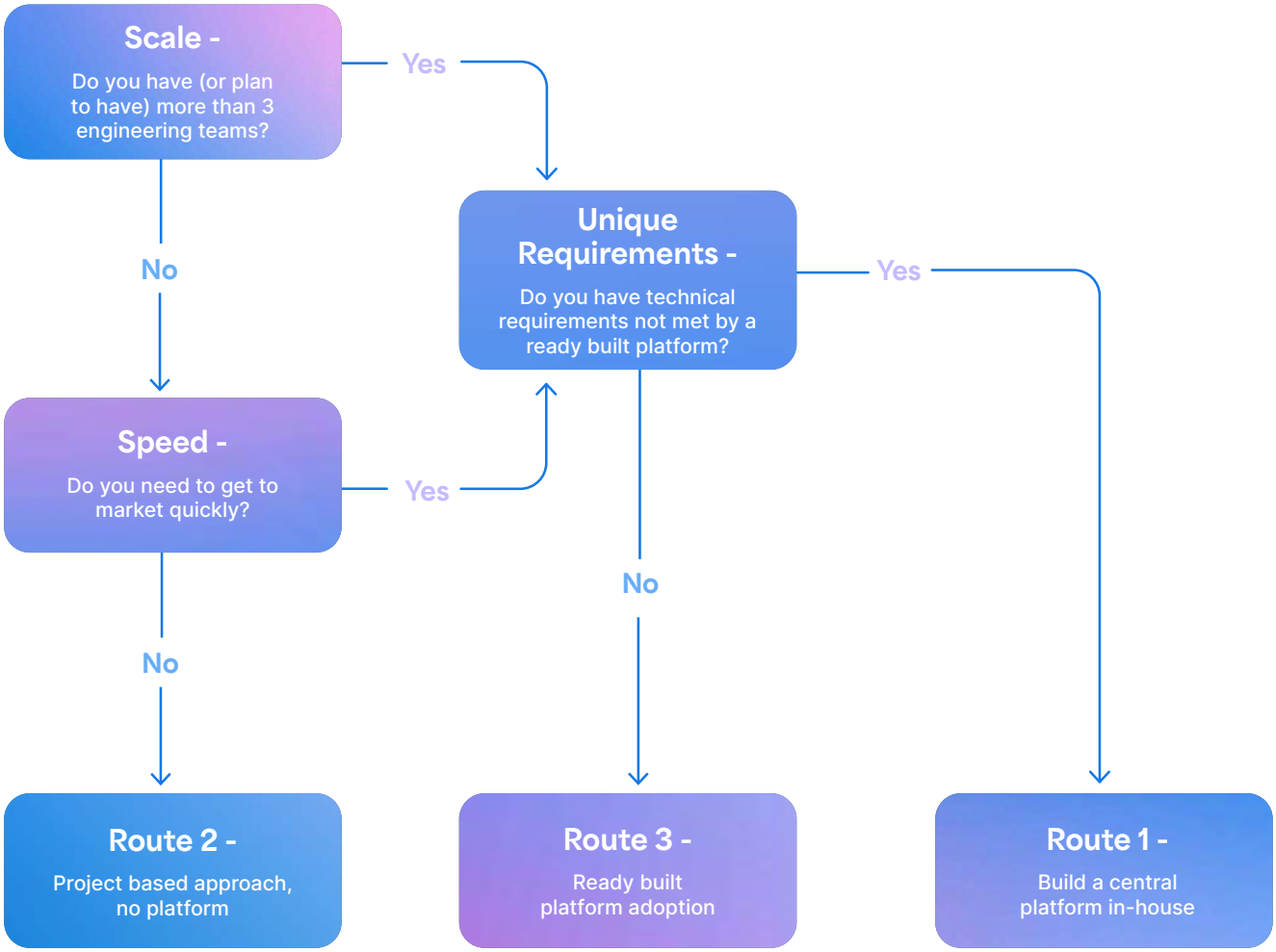
Ready Built Platform Adoption:

Adopting a platform that caters for 80% of the delivery requirements. There's time spent on working with project teams.

Adopting a ready-built platform caters for the majority of the delivery requirements enabling you to deliver with little cost, reduced expensive scarce skills and without delays. Choosing the right product that integrates well into other technologies and doesn't abstract too much away from the industry and community tools is paramount. If it's too custom and opinionated, it will hinder you from finding the relevant skills and require a lot of in-house training and domain knowledge.

There are advantages and disadvantages to each route, but it will cause frustration and hinder delivery speed if you don't adopt an approach that aligns with your business.

Which route might make the most sense for your business?



If you have scale in terms of engineering teams and number of applications and want to get to market quickly then you should adopt route 3, as you're buying a complete product and products can scale much more than people can.

If you don't have scale and have few applications, then going with route 2 can work if there is a limited number of projects. If you have very unique technical requirements that existing platform based products in the market can't cater for, then route 1 will be the best option as it can align to your business goals more accurately.

Below we'll look at the benefits, risks and cost of both **'building'** and **'buying'**, and outline the impact that your decision will have on ongoing delivery performance.

Understand Time Implications

Building in-house requires constant attention. There are the initial work-hours spent on building working infrastructure, which is just the tip of the iceberg. That effort increases drastically the more applications and functionalities are added, which is then doubled for each extra cloud that you're using.

Divided Business Focus

By building in-house, you're placing business focus in two different places, one around the business services you want to create for customers and the other on building a platform to meet the needs of the business services. Both focus points will be iterative, take time and have a cost implication, so it's important to decide if the ongoing cost of building and maintaining a platform is worth the overall investment and if the business is willing to accept the upfront time investment.

The end result of building a platform could be positive if the business type is unique but, unlike the business services, a platform isn't something the business can sell and offset costs over time. Given that, it's a commitment that needs to be fully understood.

Communication Struggles Increase Lead Time

Building out infrastructure to support application delivery takes time and effort, and crucially depends on solid communication practices and tooling. Developers will need to have a way of feeding in their needs and have those needs met in a timely manner to avoid any delays. If DevOps or platform engineers need to investigate solutions and then subsequently construct those solutions from the ground up, delays to developer needs could be significant. It might then require compromises to be made, increasing security risks and reducing the overall quality of production.

As services become more operational, communication channels become increasingly more important. Changes to the underlying supporting platform technologies could have high operational impact, if they aren't designed to be highly scalable and resilient. This can impact developer teams and services if the disruption isn't managed and communicated properly.

Steep Learning Curves for Developers

For technologies like Kubernetes, even when it's a managed cloud service, there's a need to enhance the ecosystem with add-ons and additional security configuration to influence a consistent, resilient and secure way of working across teams. Although that brings huge advantages, there are inherent complexities and the associated steep learning curve to achieve a strong outcome.

It isn't just technologies like Docker that developers need to learn, but all the features of Kubernetes — like deployments and secrets — as well as additional add-ons like certificate management, network policies and load balancing.

These knowledge requirements add to the overall complexity when shifting-left infrastructure responsibilities to the developer unless there is something that abstracts the complexity away lowering the barrier for entry and helping teams deliver quickly.

The average cost of an infrastructure failure is

77,000 per hour

(Source, IDC Report 'The Cost of Downtime')

Build for Production

As applications start getting closer to production it becomes crucial to have thought through monitoring processes and security best practices.

Monitoring and security aren't just application focused, but include all the surrounding services that the application depends on (directly or indirectly) from Docker registries through to Kubernetes cluster nodes supporting the workloads. Knowing the health of these services has a huge, positive impact on customer experience, overall service quality and the bottom line

Working through best practices takes a huge amount of time and resource investment to do properly, but if it's left to the last minute before going live, the overall risk to the business increases.

Monitoring Takes Time to do Well

Monitoring isn't as simple as buying a monitoring solution hoping that it will cover all the bases. It requires domain knowledge expertise to break down the intricacies of what could cause an outage. If the underlying infrastructure is complex in nature, like Kubernetes and cloud are, then making sure that all components are monitored appropriately with the right rules will take a lot of time and energy.

Balancing the amount of alerts is paramount. Too many false positives will cause engineers to accept that as 'normal behaviour', ignoring legitimate issues in the future. On the opposite end of the spectrum, being overly strict will make you unaware of the issues completely and you won't see any of the benefits from monitoring.

Service Level Agreements (SLAs) will also need to align to the business criticality of application services. If an application has a dependency on another service in order to function properly, then that service will need to be monitored and have the right uptime to meet the application SLA. Cloud providers give a lot of assurances in this space, as they're designed for scale, but, like anything, these services may not be configured in a way to meet the requirements of the business. Monitoring these services is as much about managing human error as it is about mitigating the impact.

"

96% of IT decision makers have experienced a **costly outage in the last three years**

"

(Source, 2019 LogicMonitor study)

97% of enterprises experienced at least
one IT brownout in the last three years

(Source, 2023 LogicMonitor study)

Making security decisions before production

The security measures put in place, no matter if you build or buy, determine whether the technology you adopt and use is left vulnerable to serious breaches. If you decide to build your own, then the responsibility of implementing best practices is exclusively yours. Like monitoring, security isn't just about mitigating the impact, but is also about managing human error.

Managing security risk should start at the development phase and continue through the entire application lifecycle. Deferring it may mean that applications have grown in complexity and features which in turn will require more time and effort to assess the risk, and ultimately will slow down delivery.

Although the production environment is the 'crown jewel' for businesses, breaches in the development environment can be damaging to organisation's reputation if undetected early and subsequently promoted to production.

It's essential that making sure what you build gives developers awareness of the risks and how to mitigate them, and will ensure security is part of the delivery lifecycle. Continuous deployment tooling in conjunction with this means that you're less exposed to vulnerabilities — if a risk is found, it can be mitigated quickly without any serious impact.

Consistency Across your Technology

Consistency is a key part in understanding your security landscape and building internal confidence on your overall posture. If there is variability and slight nuanced changes in technology and configuration, then the risk probability increases significantly.

Automating security rules across a diverged landscape requires domain expertise in all the technology in your business, and knowledge on how they're used and the risks they present. So, you're looking at an additional overhead impact on the business.

Having a platform that bakes in security best practices allows teams to focus more on the business application security rather than on solving non-application-related security aspects of technologies surrounding the applications. If you decide to build-your-own then it is imperative that you hire security minded engineers who have the right level of experience in new emerging technologies.

Onboarding and Offboarding Users

Onboarding and offboarding processes vary from company to company, but if they're not executed properly can leave users scattered across multiple products, platforms and cloud accounts. This often results in security breaches and/or poor developer experience.

Shifting the process left, to the team itself, helps reduce costs and risk, and provides better manageability as the team should be small enough to have good insight into active and inactive users. This means you're less likely to have users costing potential license fees or access systems which, in the wrong hands, could cause huge detriment to your business.

Making sure you build or adopt, a platform that allows for teams to shift the responsibility left, to those that are close enough to the team members, will help keep a healthy position on your users overall.

Nearly two thirds of IT pros (65%) believe that they could bring greater innovation to their organization with the **right expertise.**

(Reference, The Cost of Cloud Expertise Report)

Mishandling Role Based Access Control (RBAC) is Risky

The management of user access and roles is crucial to security, but is often left as an oversight when building because it's yet another time-consuming and manual process. Understanding which users need which permissions within a team requires an understanding of the topology inside of products, especially with Kubernetes, and a lot of configuration definition and relationship mapping. This complex setup is a manual process prone to human error and could result in a huge security breach. Complexity and risk aggregate over time as new permissions are granted and people move between teams or leave projects.

With a growing number of people in your business you need a clear overview and easy control over access permissions, and a platform that supports global policies to ensure permissions meet security requirements.

Evaluate the Skill Gap

You won't be able to manage the complexities of your in-house infrastructure without a suite of specialised, skilled engineers with experience across all implemented tooling.

High Cost of Engineers with Cloud Expertise

If you're building in-house the skills of your team have to match the ongoing growth requirements. Keeping up with the growth requirements will add human cost to the total cost of the platform, tooling and application cloud infrastructure, especially as cloud platform technologies are evolving rapidly.

Engineers with strong, proven cloud expertise are in high demand, and they're expensive to employ and retain.

According to a **study by RackSpace** on the cost of cloud expertise, 2 in 5 IT decision makers believe that there is a lag in their organisation's ability to deploy cloud platforms because of a lack of skills.

Steep competition for scarce talent

There's steep competition for the talent that exists in this space, with the majority of top engineers getting swooped up by Platform as a Service (PaaS) companies and cloud organisations where they expect to drive high impact, or banks and other financial institutions where they can afford larger salary brackets.

Nearly half (46%) of IT decision makers say that they find it hard to attract the right talent to help manage their organisation's clouds. And a third of those stated that their biggest recruitment challenge was competition for talent within the industry.

For organisations that are sticking to an in-house platform solution, hiring top talent is a fundamental aspect of keeping up with the ever-changing technological landscape that surrounds businesses and applications.

60% of global executives say they struggle to keep **workforce skills relevant and up-to-date** when faced with rapid technological change.

(Reference, The Human Capital Report)

> If we were starting today, **we wouldn't be running our own cluster.**
>
> (Suhail Parel, Platform Engineer At Monzo Bank)

Calculate The Cost of Ownership

The total cost of ownership, or TCO, compiles of all the direct and indirect costs of any IT component, ultimately used to determine your return on investment (ROI). To determine your TCO of building, you need to look at the initial cost of building (the engineering resource, databases, servers, storage, networking) as well as the continual operating and maintenance costs, including security monitoring, support and additional services as the products or platform increases with the requirements of the business. It is the overall cost of the lifetime of that investment.

In addition to the TCO, it's important to consider the opportunity cost that is lost when venturing down a build or buy route. If building has taken up more time than expected, then it's possible that you missed a market opportunity or weren't able to stay competitive during that process and also plays a factor in the overall cost analysis.

Making the Decision

As technology evolves and your business grows, you need software to effortlessly grow and scale with it. **Platforms are designed to allow users to easily iterate and develop applications by removing the burden from teams.** A cloud platform provides speed, security and scalability that would not be possible to achieve with building in-house at a reasonable cost.

When deciding what to do, it should boil down to how unique your business requirements are. If you have unique requirements that the majority of the industry doesn't appear to have and there are no solutions that will match your needs, then building the supporting infrastructure and ways of working will be necessary.

When you build infrastructure yourself, you might believe you're bringing the best standard, but you're really embracing a never-ending regime of ongoing management and maintenance. To take full advantage of cloud, organisations need to be able to automate the work that otherwise drains time, budget and resources.

Monzo is a great example of this: They have the knowledge, experience and skills, and they actually did build their own Kubernetes infrastructure and "rolled their own" in AWS. But in mid-2020 they came out to say that, if given the opportunity to do it over again, they would not have built it themselves.

In order to achieve strong results building yourself, you'd need to put in a lot of effort for potentially a small return. While it's relatively simple to set-up at the start, in continuing to manage infrastructure, organisations will quickly lose out on the benefits and continue to sacrifice resources to properly maintaining and iterating infrastructure.