REPLICATED

Modern On-Prem Software Delivery & Management: Build vs Buy

Michael Ka & AJ Jambu

Table of Contents

Introduction

Historically, delivering on-prem software has meant building bespoke, manual processes for every customer environment; costing time, resources, and money. Thankfully, that's all changing.

Now there are more choices, but that means more consideration must be given before planning the way forward. When determining how (and if) to deliver their software on-prem for customers, engineering and product teams must ask:

Is it better to BUILD a delivery and management solution from scratch, or BUY a prepackaged solution?

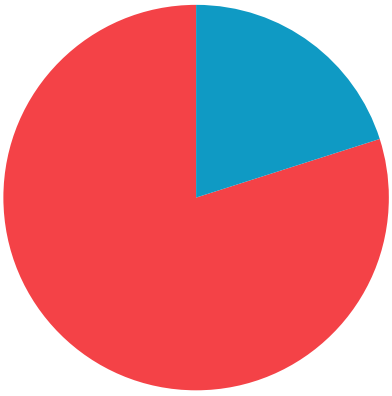
Which provides the most future-forward flexibility, with the fewest roadblocks? Factors like cost, time to market, internal resource capacity, control and compatibility, maintenance and support, and core business competitive advantage should all weigh in when it comes to making this decision.

There is no blanket answer to the question of whether to build or buy. But the details that we explore in this eBook can guide you to the right decision—specific to the needs of your company.

Why On-Prem Matters

SaaS continues to be a bright shiny object, but when it comes to the enterprise, the data tells a different story. In a recent report, Synergy Research Group noted that, **"While in many ways the SaaS market is now mature, it still accounts for barely more than 20% of total enterprise software spending and therefore remains small compared to on-premise software..."** In 2020, annual enterprise software spend for on-prem software totaled nearly $400 billion, while SaaS spend hovered around $100 billion.

Annual Enterprise Software Spend (2020)



Concerns about data loss prevention and regulatory and compliance requirements mean that many companies have never been able to fully embrace cloud solutions. Companies with federal and financial customers need to minimize the potential for data leakage in key systems. Other companies have external regulatory mandates that would make cloud solutions difficult—for example, companies in Germany, Russia and China typically dismiss SaaS because of their cross-border data flow restriction laws. In order to have more control and continuity, many enterprises have been moving internal workloads to IaaS to create private spaces where their apps can be secured away from the public internet.

But until recently, designing, deploying and maintaining bespoke on premise software was a laborious and time-consuming solution for the end-user. Now, thanks to the integration of cloud native technologies (things like containers and Kubernetes), on-prem delivery and management is becoming a much happier experience for both software vendors and their customers. Add to that the enterprise's growing concerns about security, compliance, and control in the midst of an increasingly hostile cloud landscape, and the picture becomes even clearer: on-prem is making a big comeback (even though it never really left).

DIY or DIBuy?

When it comes to delivering enterprise software on-premise, the big question is: should you build a solution or buy it? Here are some important considerations to weigh the pros and cons of each option.

Build vs Buy Considerations

Cost

Time to Market

Internal Resources

Customer Experience

Control & Compatibility

Maintenance & Support

Let's take a look at each one.

Cost

Buying off-the-shelf software comes with a fixed price, but building and maintaining your own solution can be much more expensive.

The up-front costs are obvious: design, engineering, support, and operational costs.

> However there are many hidden costs to rolling your own solution that are not easily quantified: missed opportunity costs, on-going support costs (issue triage, bug fixes), and productivity costs (pulling teams away from projects) can all create a hidden cost in terms of time and productivity. Before you start throwing money out of the window, consider the long-term costs and labor commitments you could be signing up for if you opt to build.

Time to Market

As every engineering and product team knows, there is an inverse relationship between team size and time to market. It takes two people longer to design, build, and paint a house than it takes twenty people to do it. If you've got plenty of leeway in terms of your deployment schedule, an unlimited number of team members, or a tiny house to build, then developing your delivery and management solution from scratch could be the way to go. But if you need to take advantage of streamlined processes in order to build more in less time, then buying an off-the-shelf, on-prem software delivery and management solution would be the fastest way to get your product into your users' hands (ahem, servers). And that gives you a better shot at picking up more slices of the $400 billion pie.

Internal Resources

Aside from the time and money part of the staffing equation, there are the logistics of building a new internal team (or tasking an existing one) to build a new software delivery and management solution. Will you hire full-time staff, or contract it out? It all comes down to your deployment needs and your access to a qualified talent pool.

If your in-house team already has the specialized knowledge and time available to use it, then building might be the way to go. But even if your team has time, do they know how to create a licensing system? And is that what you want them focusing on instead of other tasks?

In many cases, buying an off-the-shelf solution and letting your team focus on your core business is a much better use of time and resources.

After looking at the data from 3000 companies—including 96 that reached $1 billion in annual sales—McKinsey found that when it comes to the software business, high rates of growth are a major predictor of long-term success. That means if you want your business to succeed, you need to be prepared to grow quickly… not spend a lot of time reinventing the wheel along the way.

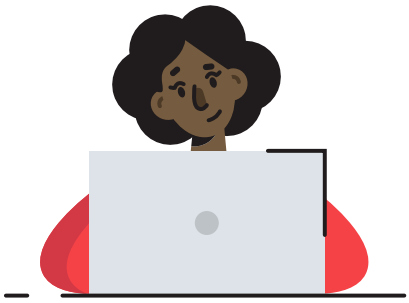
Business Continuity

One of the most important considerations in the build vs buy debate is the longevity of the chosen solution. Beyond thinking about the ongoing engineering hours that will be needed for development, security patches, Kubernetes releases etc, there is the question of succession planning.

The question isn't, "How much time will this take someone to maintain after they build it?"

It's really about, "Who will maintain this after them, and who after them?"

followed by, "How will each new person learn the intricacies of the system?"



If the answers to these questions aren't clear, then you may be in a situation where buying is the best option.

Think about it this way: most homegrown solutions are created by one or two engineers who become the experts in the systems they're creating. This makes a lot of sense for core business logic and apps that are going to constantly evolve with a steady stream of new members onboarded. However, for ancillary systems that support the core business, these applications end up becoming a silo of knowledge.

That's because a delivery solution vendor will be focusing on this problem exclusively, as delivery is their core product. In other words, part of the off-the-shelf product you're getting is the succession plan that comes with it. Since no specific person in your organization is building the system, no single person will be responsible for it. Instead, there will be several key points of contact engaging with the platform in several ways, starting with SREs/DevOps and expanding to Support Engineers, QA, and even Sales Engineers. Each of these functions gains some amount of user-level knowledge on how the system works, with a clear understanding that the vendor will own the ongoing improvements and extensions of the underlying system.

Customer Experience

Enterprises are businesses, but businesses are people. Your software solution needs to solve technical challenges while also being people-friendly. At a minimum, you'll need to ensure security and visibility in your distribution mechanism, and provide a polished end-user experience. That means more than just delivering a HELM chart, YAML files, and instructions.

Think of your software delivery as a virtual "unboxing." This is the first impression your customers will have with your solution, and it needs to be elegantly simple. Providing preflight checks and support tools as part of the installation will not only ensure a successful launch, they will also help to build customer loyalty.

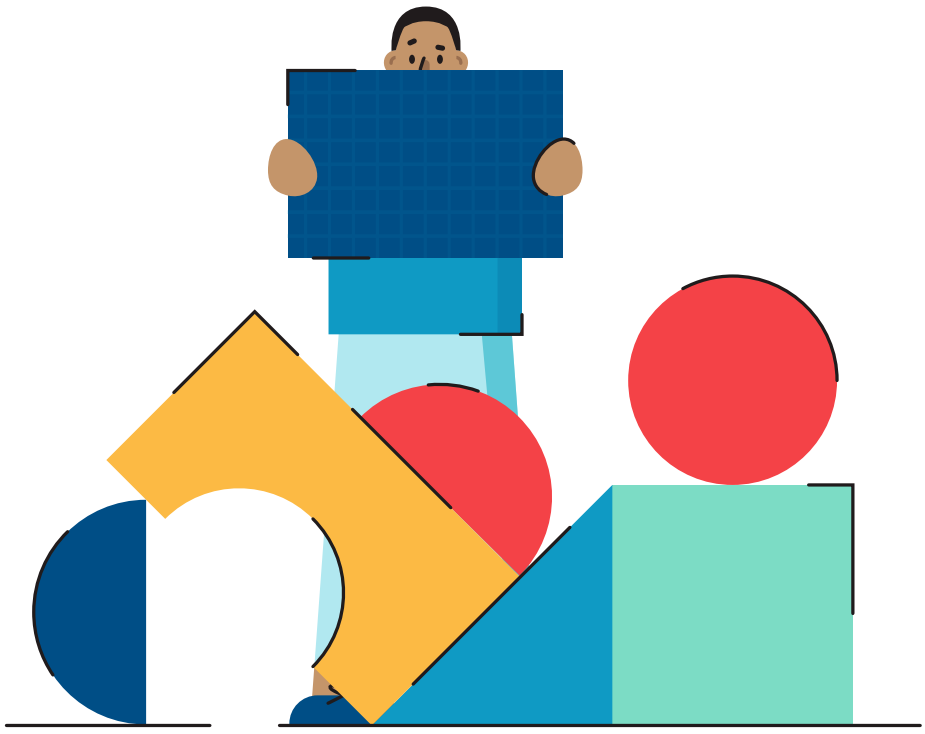
The positive customer experience needs to stretch throughout the entire lifecycle.

The product UX should be consistent, and ideally supported with updates, patches, licensing and entitlements. Managing all of this takes time and resources for each customer-specific deployment environment.

Unless you're prepared to make a significant upfront and ongoing investment, buying a third-party software solution that allows for easy installation, ongoing maintenance, and a great user experience is the safest bet.

Control & Compatibility

The idea of control is alluring. Building provides more control over the tooling and all of its moving parts. When you build, you're not ordering from a vendor's menu and you're not limited by their offering—you can control what and how your solution works with all of the parts you want and none of the ones you don't. But all of that control comes with additional responsibility, and most of it isn't very glamorous*.



When it comes to compatibility, you want to give your customers a polished, mature-looking, end-user experience with enterprise-class installers, support for GUI-driven deployments and installation that works across many different types of environments. Whether you're distributing to bare metal servers, VMs, PaaS, IaaS, SaaS, VPC, containers, managed Kubernetes, a VPC or even an air-gapped environment, you shouldn't need to require multiple bespoke paths for deployment.

Having an automated solution that provides a single pattern for shipping to all of the different environments is a more efficient way to go to market, and this is where an off-the-shelf solution really shines.

Maintenance & Support

The process and tools you build to deliver and manage your application is a product in itself. And every product requires ongoing maintenance and support. Larger enterprise software vendors all have product managers and engineers dedicated solely to distribution of their on-premise software for this reason.

Let's face it, if you build it, you will need to support it.

So, while building provides more control, it also requires ongoing support

(*this is the unglamorous additional responsibility mentioned earlier)

On the other hand, commercial vendor solutions can be more low-maintenance, making it easier for you to deploy patches and updates to customers. If you decide to buy, look for features like embedded tools for environment conformance, configuration validation and troubleshooting, and the ability to make last mile configuration changes that will persist through application updates.

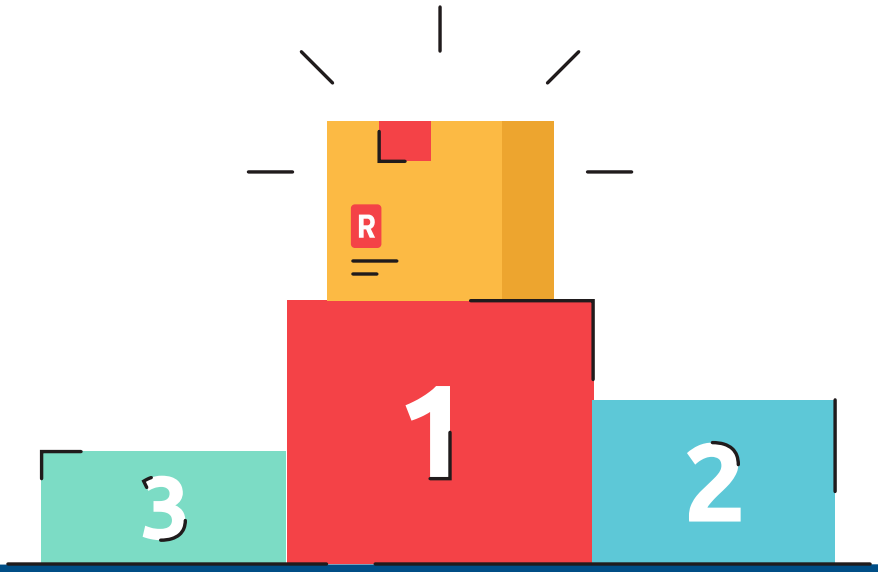
Competitive Advantage

Let's talk about resource allocation and smart spending. If the delivery experience is a core part of your product offering, then obviously a custom build makes sense. But if your core business is AI or machine learning, for example, then investing the time, money, and engineering resources into building a software delivery solution for multiple target environments—a feature that is not a part of your core business—could be a big risk.

Here's why: if you want to achieve a growth rate that will make your business profitable and sustainable, then you're going to need to delegate beyond your in-house team. Don't fall into the "humpty dumpty" trap: according to McKinsey,

"once growth is broken, it is impossible to put back together again."

Their data revealed that when software companies took a pause from growth—even when it was only temporary—they created less than a quarter of the value of companies that maintained their growth without pausing.



With this in mind, buying a commercial solution to deliver your application to customers could be a better choice. Letting your development team focus on creating core capabilities at higher velocity is a better return on engineering dollars and a smarter way to solidify long-term success.

That said, if you have the engineering overhead to spare, have at it. Build the exact solution you want and need; just make sure that there aren't any economic consequences for doing so.

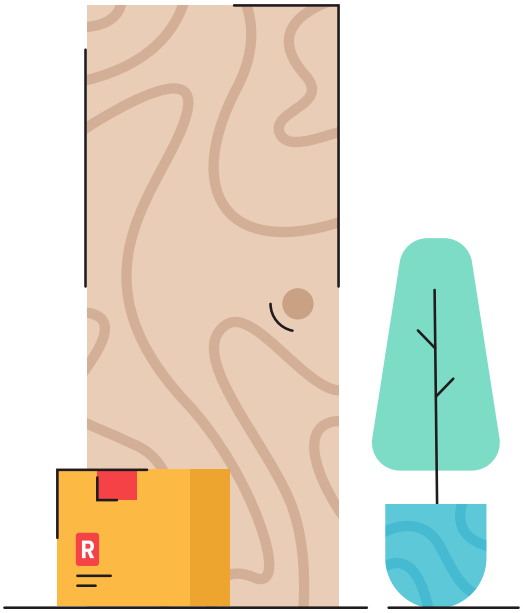
Build vs Buy Bake-off: Which should you choose?

Every case is different, and all of these factors should be considered carefully before making a decision. There are situations where building a solution makes sense, either because you have the time, your requirements aren't very complex, or if you simply don't have enough customers to justify investing in an off-the-shelf solution.

Summary

	Build	Buy
Cost	Headcount, opportunity, & time	Cost of product, often lower over time
Time to Market	However long it takes to build the solution	Now-ish
Internal Resources	Headcount and skills	Significantly lower overhead
Control & Compatibility	Total control	Out of the box limitations
Maintenance & Support	You build it, you maintain it	Done by vendor
Competitive Advantage	It's not your core business	Focus on core business
Business Continuity	Succession planning for silos of knowledge	Part of the off-the-shelf product

But if you're under time-to-market pressure, have limited resources, and need to focus your team on your product, then there is a lot to gain from working with a vendor. If this sounds like your situation, we built Replicated for you. Learn more about our modern on-prem solution that gives you the tools needed to scale, operationalize, deliver, and manage Kubernetes apps on any end customer environment.



Still not sure which option is best for you?

Let's talk.

About the Authors

Michael Ka

Strategic Account Executive at Replicated

Michael is an enterprise software veteran with 25+ years of industry experience that spans Virtualization, Cloud, SaaS, Containers, and Kubernetes.

AJ Jambu

Forward Deployed Engineer at Replicated

AJ helps companies of all sizes build, design, and maintain IT infrastructure—and he's been doing it for over 14 years for companies like Shutterstock, Venmo, and Tumblr.

www.replicated.com