What Is Package Management *Really* Costing You?

A Look At The Total Cost Of Ownership In Package Management

cloudsmith

February 2020

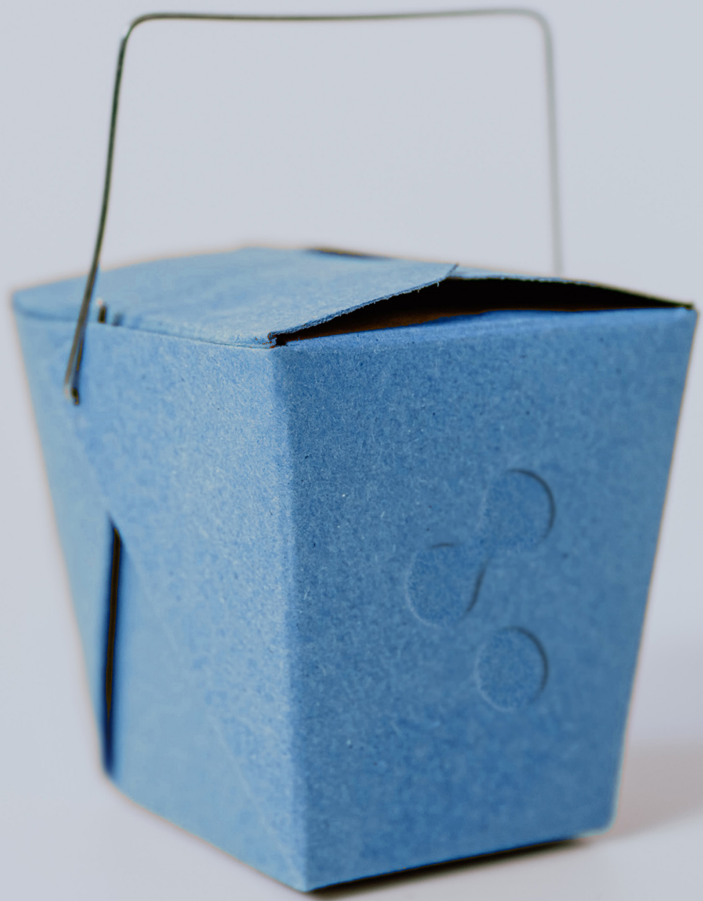
How much is package management really going to cost you?

Anyone who uses packages in their software development process (in other words, everyone) has to manage those packages somehow.

Broadly speaking there are four options when it comes to making that happen:

1. Rely on public package management solutions and repositories
2. Build your own private repository and package management solution (either on top of existing open-source solutions or from scratch)
3. Invest in a commercial on-premises package repository
4. Buy a cloud-native universal package cache: Cloudsmith

I want to talk about each in turn, and estimate how much they cost – or could cost – your business.



1 Rely on public package management solutions and repositories

There's no doubt this is the path of least resistance. After all, it costs absolutely nothing, or something close to nothing, to get started down this route. The costs come later.

When they hit, however, the potential costs of this approach are somewhere close to infinity. Here's the truth: although almost every application we build includes large amounts of third-party code, we need to remember that using this code opens up the business to all sorts of risks and potential costs.

Relying solely on public repositories isn't something that most serious development teams do, for a variety of reasons, including:

- **Existence.** If our build goes looking for a package and discovers it is no longer there, we have a problem. And if that package was central to the product we are building, **we have a big problem**. On a public repository, ultimately we have no control of availability. We just have to hope.

- **Stability.** What happens when a new version of a public package replaces the old in a public repository? The answer, all too frequently, is that it now conflicts with something in our own code and breaks our build. Worse, we won't necessarily know what went wrong and where else we may be exposed. Which brings us to...

- **Vulnerability.** No solution can 100% guarantee that the worst won't happen. But pulling down and integrating packages from a public repository without any intermediate step makes it a lot more likely. **Heartbleed**, a vulnerability that spread largely via software packages, was estimated to cost the industry upwards of $500m. And that certainly wasn't the only one. We don't control public repositories, and thus when we use them we run the risk of something unpleasant coming along for the ride. And when something does go wrong, we are usually the last to know, and even when we do it takes more time and more money to establish where we are vulnerable and resolve the issue.

I could go on. Ultimately this approach does nothing, or close to nothing, in terms of control of your software development process. There is no means to control who has access to what, or indeed establish who used what and where if things go wrong. I haven't even mentioned the fact that the business is exposed to vulnerabilities in terms of licensing as developers download and use packages from a variety of sources with a variety of different license models.

In short – this approach may seem cheap, but it can ultimately cost you everything. It is no surprise at all that no serious development teams are happy to stick with this approach.

> ***"On a public repository, ultimately we have no control of availability. We just have to hope."***

2 Build your own universal package manager and local repository

Having learned the lessons above (sometimes the hard way), most organizations now understand the need to have private repositories and some form of 'universal' package manager that gives control and oversight in terms of provenance and usage.

Of course, no two homegrown solutions are the same, but in most cases, they'll be trying to do similar jobs, including:

- Providing a private repository, in which the availability, provenance, and security of packages are controlled by the organization itself rather than a third party
- Include access controls and some form of auditing – enabling you to control who is using what, and know who used what in the past
- Provide a 'universal' package management layer, meaning a consistent way to integrate packages into projects and builds no matter what language they are being written in

All this is good stuff. It goes some way to avoiding the potentially disastrous consequences of the first approach. But what does it actually cost?

Firstly, it must be built. Yes, open-source solutions may get you some of the way, but they are unlikely to be enough in themselves. And developers, who we will need to get it over the line, cost money. At the last estimate, **a developer in the Bay Area will set you back $150,000 a year** (and even that feels optimistic). How many will you need, and for how long? We can't say for sure, but knowing what we do about the development process **you can probably take any estimate you care to make and double it**.

And the costs of this approach don't stop there. It must be remembered that we also need to include the following:

- **Product management.** What languages are going to be supported? Whose opinion takes precedence when it comes to adding features? Who is keeping track of changing requirements? These are complex and political decisions – they consume time and will never stop doing so.
- **Operations.** Who is keeping this solution online? Who is managing upgrades? Who do your engineers call when something is broken? It is expensive to manage a service that requires something very close to 100% uptime.
- **Version 2.0.** Don't imagine you will build this once and draw a line under it. You know that doesn't happen. You will need to continue to resource developers (developers you hired initially to build your product) for as long as your solution continues to exist.

> ***"We have worked with organizations employing up to a dozen individuals to handle package management across the business."***

As we said above, each home-built solution is unique, so we can't pretend to know exactly what the costs of this approach will be. What we can say is that we have worked with organizations employ-ing up to a dozen individuals to handle package management across the business.

That is over a million dollars a year. Worth it when compared to option 1. But we can do better again.

3 Invest in a commercial on-premises package repository

"Scalability and availability are both complex and expensive attributes to deliver in any circumstance."

Understanding the challenges and costs associated with the 'build your own' approach, many organizations make the logical decision to bring in a commercial off-the-shelf solution to handle the package management function.

As far as it goes, this is a good idea. It does remove the costs and decision-making required when building a solution from scratch. Unfortunately the dominant model to date in this space – usually for performance reasons – has been the on-premises solution. And whilst that removes some costs, it does nothing about others.

Specifically, an on-premises solution means the organization is still liable for management of the system, which includes maintaining hardware, scaling to meet new (and sometimes unexpected) requirements, ensuring availability and making sure integrations are working – and stay working.

These are not trivial undertakings. In fact, scalability and availability are both complex and expensive attributes to deliver in any circumstance.

Meanwhile, in organizations characterized by multiple locations, the job also involves building and maintaining complex solutions to the challenge of sharing packages across those locations, whilst maintaining the same standards of security, availability and performance.

Just as with the build solution above, this quickly becomes an expensive undertaking. It is common for at least one and often many more developers to become 'responsible' for the upkeep of these systems – and that is in addition to license costs of course.

All in, it becomes questionable whether this approach is any more cost effective than the alternatives we have already discussed.

4 Buy a cloud-native universal package cache: Cloudsmith

Fortunately, there is another way: the cloud-native universal cache, an approach that delivers all the benefits of active package management without the costs and limitations of traditional on-premises solutions.

Cloudsmith implements package management where it belongs: as a cloud-based cache accessed through a dynamic CDN, so that packages are fast to access and integrate from any location across the globe. And thanks to that implementation approach, all responsibility for the operational management of the service is removed from the customer.

Just as described above, by using a product like Cloudsmith your team has confidence that packages are there when they need them, are what they say they are, and won't change unless you want them to. In addition, we give you total control over how packages are used, who or what can use them, and provide details of how they have been used in the past. You can read more **here**.

By engaging with a business entirely dedicated to resolving the challenges and exploring the possibilities of package management, you can have confidence that the product you are using will continue to evolve to support your needs – sometimes in ways you could not even have anticipated.

Meanwhile, your business is supported by experts in the area who can provide expert insight based on having seen what works elsewhere (and what doesn't). We'll even set up your own Slack channel so you can chat with us whenever you need to.

In terms of cost, our pricing is open and transparent, and available **here**. For businesses and teams large and small, it will almost certainly end up being your best and most cost-effective option.

If you'd like to get on board, drop me a line today and we can talk.

Sign up: cloudsmith.com

Twitter: @cloudsmith

Facebook: facebook.com/cloudsmith.io

LinkedIn: linkedin.com/company/cloudsmith

Email: support@cloudsmith.io