

Packaging, CloudInit, and Revisiting “The Good, The Bad, and The Ugly”

An interview with Baptiste Daroussin

Tom Jones

Tom Jones: You’ve been using FreeBSD since 1998 and have been a committer since 2010. How has the project changed from when you started using it on the desktop?

Baptiste Daroussin: I don’t think it has changed much, but there are a lot of people connected to the project who twisted my arm into becoming a committer. I started on a laptop in 1998. The first thing I tried was to build OpenOffice from ports. I don’t remember whether it was already there or was soon to be. Someone told me, “Hey, if you’re crazy enough to try to get OpenOffice working on FreeBSD, you should join the project,” which I did 10 years later.

In 2010, I don’t remember who, but someone invited me to present at BSDCan. I was already working on pkgng at the time. I had to present that work at BSDCan, and that’s how I got directly involved in the project.

TJ: So, when pkg first came into the project, it was pkgng?

BAPT: Yeah, I called it pkgng first. I complained that nothing was usable directly as a binary package, and a friend of mine (Emile Heitor imil@netbsd.org) was working on pkgin for NetBSD. I started by porting his tool to FreeBSD, which worked. I quickly jumped into the limitations of the package tools under pkgin.

So, I started my own project with Julien Laffaye, who was already a committer at the time. It was just supposed to be between us to see how complicated it would be to make something from scratch. In parallel, we were pushing new patches to improve the package install that we have in the base system. I quickly figured out it wasn't too hard to achieve. So, I ended up with something that worked. I sent them an email saying, "Hey, it's kind of working," and that's why they invited me to Canada to talk about what I'd done. And I joined the project.

TJ: Were you expecting at that point that you'd still be working on pkg 15 years later?

BAPT: No, clearly not. The companion of pkg, Poudriere, has been there since the start because I needed something to test pkg and build packages quickly, so I could test pkg itself. And now, Bryan Drewery is taking care of it. I'm delighted that I don't have both, but I am still working on pkg, and it is still kind of fun. There are many things to do. I haven't done half of what I had in mind in the beginning yet.

TJ: So, there's a lot of work still to do?

BAPT: Yeah, there is. If you look at the history, I gave a talk about making PkgBase a reality, something like 10 years ago, at BSDCan 2015. We are only doing it right now.

I have many security plans for the package that are slowly landing right now, such as preventing any random root script from running by default, so that everything can be reviewed. Right now, all the scripts are put in by someone in a random port somewhere with, obviously, a lot of care (not). That's something I really hate. So, we now have triggers that can be reviewed in a single place. We can have sandboxed scripts. Everything that is running in Lua is sandboxed under Capsicum, which also limits what you can do. My long-term goal is to one day prevent any script in a package from running as root. I'm not sure I will manage to do that, but I'm still aiming at it.

TJ: What do you think the barrier to getting more people involved in package development is?

BAPT: I don't know. Probably me somehow. We have eight people who can merge things there, but no one is willing to do it. It's a highly complex piece of software, even though it looks simple at first. You only extract tarballs on the filesystem. What could go wrong? You don't have to make it complex. You don't have to do the whole thing; it's just a little task.

But then it becomes very complicated: if you want to do upgrades and add the flexibility people want, you end up with a complex solver, which, to be honest, I don't understand myself. We use SAT solving for that. It was done by Vsevolod Stakhov (who is also a FreeBSD committer) during the Summer of Code. He probably still knows better than I do, but he's not active anymore in this area.

Another issue is that it's the kind of software that needs a clear, long-term view, and I'm terrible at writing roadmaps. If people want to join, they fear changing things that would not go in the direction that I may have in mind. And that can block them from contributing. But we still have more

contributors than other subsystems in the project, so in the end I am probably not that bad.

TJ: You mentioned that PkgBase was a goal 10 years ago. Do you think the package tooling has been a significant barrier, or do you think there have been other factors to getting towards actual packages?

BAPT: I think it was other factors. We can see now that the to-do list for pkg was big, yes, but there was no motivation to do it. It's a chicken-egg problem. No one was willing to work on the PkgBase system itself. There is a will in the project to work on it for a while, but no one wants to put in the energy to finish it.

The real energy happened a year and a half ago. I was the one who put the energy into creating the initial packages and the initial framework, and nothing happened. Then there was a bit of energy among the OCI people, who were very interested in making tiny OCI images based on FreeBSD. So, they put some energy back. And recently, there has been a lot of push by Ed Maste, Kyle Evans, and Mark Johnston, and more recently, a lot of work from Lexi.

I keep forgetting about it, but manu@ did a tremendous amount of work after I set up the initial framework in the source tree for building packages because he had the motivation to do so. At the beginning, he was actually paid for that, so that helped at the beginning. He had the motivation to build some appliances that needed PkgBase. At his current company, there is a desire to migrate from another packaging base to this one.

The complexity of the packaging base is that we need to get the whole project on the same roadmap. There were more attempts in the past. Before

pkg, I think a company was making a packaging base. There was a presentation done somewhere, and they were using the old package install tools. Then, when I started speaking about this in 2015, there was a Russian company that took what this company was doing and switched it to pkgng at the time. Then iX Systems proposed a mechanism where it wasn't really packages, but the current sets were turned into packages—so it was FreeBSD base, FreeBSD kernel source, and so on. It was a bit more fine-grained, but not that much.

TJ: Do you think there is a project lesson for FreeBSD, and how we approach big system-wide changes like this?

BAPT: I thought about it, but the problem is we are mostly volunteers. If a company does not drive this work because they have a huge need for it and drive it from A to Z and provide enough time to the people working on it to do it, it cannot work. That's the kind of project we are. That's how it was for iX, and for the Russian company as well. They came up with something workable for them, but they never had enough time to complete it.

I understand that; I mean, it's not to blame them, they must switch to other things during their work time. So, taking the time to integrate that properly with all the connections and processes is too tricky. Usually, only one hero takes it to the end. I'm happy that PkgBase is not ending up like that.

TJ: But such is software.

BAPT: Yeah, yeah.

TJ: There's a lot of stuff that is good enough for a company for their shipping product, but it's not good enough for FreeBSD.

BAPT: Exactly.

TJ: Sometimes it feels like our standards are too high to get code in.

BAPT: I'm not sure the standard is too high. People keep claiming that, but I look at some open source projects, and they are often way harder to contribute to than ours. If you work with the Linux kernel, for example, or another operating system, it's often harder to get code in. There are way more companies working on those operating systems. But if you look at Linux, many companies aren't upstreaming anything because the barrier is too high, yet they still have lot of contributions in the end.

The barrier is okay; the thing is, how do you convince people to work on areas that go beyond what they had in mind when they started? Like, yes, this is good for servers, but think about desktop users, what would happen... "I don't care about desktop users. Should I pay my employees to do that?" And that's the problem we face. All the companies I've worked for have this problem across every operating system and every open source project they work with.

It's always the same problem: my experience with past companies that contributed code is that it's easier to get stuff into FreeBSD than in other areas if you know someone in the project who can help you. The problem is that there aren't many people, so it's not easy to find someone on the project to drive you.

TJ: Is there a way that project members could make themselves more available to corporate entities?

BAPT: For companies, this is a hot topic because, basically, you have Linux everywhere, and people expect to get all the technology from it, so I don't know how you can change that. If we're talking about contributions, it isn't

the members who should be more available. Of course, that works, but the company's culture should change.

As I said, in every company I've worked with, it's either an individual hero who upstreams everything to whatever open source project they work on, or they are just sitting on a massive pile of patches, which, yes, one day, maybe they will contribute. That's what I'm doing right now at OVHcloud: making sure that even if it looks harder at first, try to work first with the community, instead of piling patches of our own. Once you get used to it, and enough involved in the community, it gets easier and simpler. Another win is upgrading does not become, anymore, a nightmare where you have to rebase and retest all your accumulated patches.

In a previous company I was working for, because we built a relationship with some opensource project (not FreeBSD in this case) we were able to be in the path of many patches as reviewers, or just informed soon enough. It allowed us to weigh in on it. And while being still a very small team, it had, because of that, peace mind for each major upgrade which became mostly seamless.

It was easy just because we're in the community; on the other hand, we end up having to do a lot of things. We're now having to rebase all our patches before the new release comes. And of course, it comes when you are working on a very important project in your roadmap, so you don't have time to look for it. So, you postpone, and six months later, you have a big step to take. So, the project could be seen as more available, but it's a matter of the mindset of the people on the other side.

TJ: So, you mentioned OVH, and that's a good segue into another tool available in FreeBSD 15: your CloudInit tool, NuageInit.

BAPT: Yeah, it's in 14 already; it gets good starting at 14.3. When I joined OVHcloud, there was no FreeBSD image available, which was painful for me because there had been one before, and I was a client. To my knowledge there was always a FreeBSD person in the company to deal with it in the past. But we don't really have the time to maintain those images besides having clients who request them.

I figured out that the images FreeBSD provides work almost out of the box on bare metal, thanks to the “bring your own image” feature—basically, you just burn a raw image to the disks on bare metal, then boot up. I figured that one thing they did for the “bring your own image” on bare metal was to use CloudInit.

How they do CloudInit is not through fancy network things like you have in a virtual machine or cloud stuff. What they do is use a mechanism where a disk is presented to the machine, and when it boots, the disk contains the data. So, it's basically the CloudInit configuration on the disks. When I saw that, I thought if you want to install FreeBSD out of the box without using something like mfsBSD or other techniques, which involve using qemu and overwriting the disk, I must have a working CloudInit image.

I must have a working CloudInit image. And I looked at the official CloudInit, which I didn't like at all because it's enormous. FreeBSD support is fragile, and I don't blame them for that. It's a lot of work to put FreeBSD-specific stuff into this. It's too big because it requires Python and related tools, and we don't have Python in base. I wanted something that works in the base system. It shouldn't be hard to make something good enough to boot the machine. I wrote NuageInit this way, I did the minimum required for our bare metal usage.

Then I figured we have clients requesting a native image on our public cloud. I looked at what it was. It's an OpenStack. So, it also respects CloudInit, except that it fetches configuration from the network by default rather than from the disks. It can work on the disks, but then the user must find where the option is, and it's not on by default. I added to NuageInit the ability to fetch configuration from network. We were able to provide the image. I think we have had a native image available in the public cloud since June.

Then the VPS team came to me, to be able to propose FreeBSD to our VPS customers, they did the work themselves, did extensive testing and provided many bug fixes (which I committed).

Apparently, people kind of like NuageInit because now we have many people adding features. The version in FreeBSD 15.0 is, I won't say complete—because there is a lot more stuff that could be added—but complete enough to be usable almost everywhere. It works for everything that is based on OpenStack.

TJ: You spoke in 2018 at EuroBSDCon on “[The FreeBSD project, the good, the bad, and the ugly](#)”. How do you think we're doing against the bad and the ugly?

BAPT: I think we're getting better. Network and Wi-Fi were tough to get. Even if I disagreed with the approach taken, I have to admit it works fine now. And I'm using it right now. And you worked on the alternative approach, and it worked as well. It brings us back to a place we have been, both in the bad and the good. We have two solutions for the same problem, exactly like we have three firewalls. It's probably part of the reason why we like FreeBSD. You know, it makes it human; we have different tastes. There are places where we cannot agree on the solution. And, okay, we have two solutions, and people want to maintain it. It's okay. It makes it a family.

I think Bluetooth is still a pain. Workable, but still a pain. If I look at what I wrote as ugly at the time: CSH by default, I dealt with that. I have CSH, but I dealt with that. We have sh by default. Firewalls and mail, I think we are done. Vi has been upgraded since. So, we're okay-ish.

PXE boot is still a problem, though we have improved a lot. I remember the context I was working in at Gandi at the time. We needed to deploy some dev machines. For the filers, we booted all the Linux boxes over PXE, and FreeBSD was painful for that. We couldn't use iPXE; we had to go through MFS images and similar stuff, which was complicated. While on Linux, it was very easy. And well, 10 years later, Warner finally fixed it a couple of months ago. iPXE Memdisk support in the kernel so that we can boot a raw FreeBSD image directly from iPXE. I use that in OVH as well, since you can provide your own iPXE script to boot your bare-metal machine. So that's great. It takes us 10 years to fix them, but it's fine.

On the bad side, I mentioned a lack of an out-of-the-box cloud image. It's fixed. We have it now. We still have issues with local FreeBSD update mirrors that will probably be fixable by the switch to PkgBase, as discussed in the latest infrastructure thread. I don't know yet what release engineering will be doing, but everything will be CDN-compliant. So, if it's CDN-compliant, we can provide a mechanism to mirror it locally. And in companies with many FreeBSDs, you're happy to get your own FreeBSD mirror with each update.

We don't have a bhyve frontend. There is some GSoC code this year to add a bhyve backend to QEMU. So, a QEMU-aligned frontend would be great for bhyve. I also mentioned another missing feature for bhyve: live migration. We have pass-through now, but we still don't have live migration. And it's the main problem for bhyve in companies. If you are a cloud company providing VMs, it's interesting because you abstract away hardware problems. You need

to migrate the machine without the client noticing. So, not having it is fine for laptops, but not usable in companies

And I mentioned the lack of dynamic CPUs, which we still don't have. It is interesting because I don't think the use case we had for it at the time is still relevant today. But at the time at Gandi, we were providing some cloud services, and we could dynamically add CPU cores to VMs without rebooting them. It works on Linux. It wasn't working on FreeBSD. You had to reboot. But I'm not sure that's how people consume VMs anymore.

NFS was mentioned as ugly. We had a very hard time with it at the company I was working for at the time, but it's also ugly on other operating systems (for the kind of workload we had). Pmcstat UI is still ugly, but it works. Sanitizer is improving, but it's still not good. Forth in the loader, we have fixed that. The old GDB has been fixed. So, most of the ugly and the bad have been fixed.

TJ: For Linux, there's a clear workflow: generate your patch, do your work, run this script over it, and then follow its advice. We have a ton of ways to submit a patch and a ton of places to check. There's almost an implicit situation: you need to find a sponsor, find someone to help you, and even before you contact a sponsor, there's no way to know if you've reached a minimum level of quality. The source manager may be able to handle this in the future.

BAPT: Probably they can do that. It needs to work well with the Linux way of working and go back to the family-ish way we have, rather than the very structured way. That's the reason I joined FreeBSD in the first place, and why I worked on Linux and contributed to the Linux community. It's messy, but we can talk to everyone quite easily, and it feels like family. I joined the project because, in the end, I found that it was easier to get things done.

I could make mistakes. It's okay. I know some very talented people who can review the work, and I can reach out to them directly. So, when in your first patch you get some famous names coming in and jumping in, that is exciting. I've been working on contributing to a company where we contributed a considerable part of a subsystem on Linux. It was probably more professional, but more painful and less fun.

TJ: I found that from the Linux side, there are unspoken interests, and you'll never get an answer if someone in a key position doesn't want something. Maybe every open source project has this to some extent.

BAPT: Exactly

TJ: In your EuroBSDCon talk, you asked where the cool swag was—"Are we cool yet?"

BAPT: I don't think we are cool yet. We have very few newcomers to companies who turn up with: "Oh, you have Linux, but I've been working with FreeBSD."

It's not the company we need to convince to switch to FreeBSD, but the students, the youth, the sysadmins, and the developers in general. To show them that it's cool to work on FreeBSD, and they will bring that with them when they join companies: "It's very easy with FreeBSD," and companies will follow if that's the case. We totally fail at doing that. While we were very cool when I was a student, it was way cooler to have FreeBSD on this laptop than Linux. With Linux, you were kind of, "hey, you're interesting," but you were the hardcore guy when you had any BSD on your laptop.

TJ: What do you think a first step would be?

BAPT: Also, at the time, FreeBSD was huge in universities. In general, there was always a sysadmin on campus, so they were the ones polluting the kids' minds with the idea that FreeBSD is cool. We lost that position, so there may be a chicken-egg problem here. I really don't know.

There is something we do great in FreeBSD that other BSDs might criticize us for, but we are good at catching up with Linux. When Linux does something good, we can, at some point, provide either compatibility or catch up with something similar enough so that we are not left behind in the game. But there is something we failed at: explaining how we can do things the FreeBSD way. How can you do something the FreeBSD way rather than trying to do it the same way that Linux does it?

It's interesting because, in all the companies I've worked with, I have set up FreeBSD each time. And, I always found a system that was way simpler to implement and maintain than the equivalent on Linux, most of the time. That's something we should really push, communicate about, and show through tutorials and stuff like this, because we are very good at making appliances based on Linux.

Take a look at the framework to build embedded images on Linux like Buildroot. There are two or three different ones. You compare them to, for instance, doing the same with a Poudriere image. It's 10 times harder to do your image in Buildroot than in a Poudriere image. I did it to provide images at the time.

TJ: Do you think we're missing training materials, or is it more of a full-spectrum thing?

BAPT: I think it's full spectrum. We don't have a habit of communicating properly. When I saw most of the tutorials on the internet, I saw people trying

to say, “Hey, I’ve seen people doing that on Linux, look, I’m doing that on FreeBSD as well,” which is great, but why will I use FreeBSD if I have Linux that can do the same? It’s important to know that we are on par with many systems and can do the same thing, so don’t get me wrong. But we should also say, “This is how I build an image based on FreeBSD in 10 minutes.”

We don’t take the time to communicate.

When I wrote a MAC kernel module, I was contacted by the Journal. They said, “Hey, write an article; it would be great to explain what you are doing and how complicated it was.” I said, “Yes, I want to do it,” but I didn’t have the time—I didn’t take the time. That’s also why I jumped on this interview right now. If I didn’t do it now, in a year and a half, it would never happen. That’s the problem.

TJ: In the project, we are behind in many ways from the industry’s state of the art. We have some CI, but we don’t have GitHub Actions-compatible pipelines. We have bits of it, and I would like to know whether we are also missing other stuff, like growth hacking and people who do developer relations and sell FreeBSD as a platform for sysadmins.

BAPT: I agree with that. I personally maintain the CI for SourceHut, which I haven’t updated in one year, and I pay for my SourceHut subscription to run a CI on FreeBSD. So again, if we want this to work and to be easy, we need to be better at spreading the knowledge on how to do that easily. It took me something like a few hours to build. I have something easy based on PkgBase to get the CI out of the box working there, rather than the complex script I see everywhere else. We have knowledge that is not shared.

TJ: I want to thank you for your time; I don’t want to take up too much of it. Do you have anything you want to add that we’ve not discussed?

BAPT: I'm happy that you brought up "The good, the bad and the ugly" thing that I did at the time because I can see that we have fixed 99% of the bad and the ugly, so the project is in better shape than what I think sometimes, so it's good.

TJ: You may need to reprise the talk in three years and do a rundown of how we're doing. Through the podcast and conferences, I do a lot of evangelism. Still, I don't want to argue with anyone.

BAPT: That's why I don't want to convert people to using FreeBSD. I want them to come to FreeBSD for FreeBSD, so they understand where they go. I don't want them to copy-paste their Linux usage on FreeBSD because I think it isn't good. I want them to understand the difference, what FreeBSD can provide them, and, in many cases, why it's more useful than what they have. In some cases, if it's not, that's fine. But we do need to let people know that FreeBSD's value lies in its flexibility.

BAPTISTE DAROUSSIN is a Unix system engineer working at OVHCloud and has been a FreeBSD developer since 2010.

TOM JONES is a FreeBSD committer interested in keeping the network stack fast.

Copyright © 2026 Held by Owner/Author. Publication Rights Licensed to The FreeBSD Journal. This work is licensed under a [Creative Commons Attribution International 4.0 License](#).

3980 Broadway Street
STE #103-107
Boulder CO 80304
1-720-207-5142
info@freebsd.foundation.org

