PUBLIC

Podcast: The Open Source Way

Episode 49: QEMU

Fig. 1 – Cover art of The Open Source Way podcast
Figure description – A square cover art with a man standing in the center of two large hanger doors, and he is pushing them open. The title “The Open Source Way” appears in a blue anvil-shaped logo above the image.

Transcript

Karsten Hohage: Welcome to *The Open Source Way*. This is our podcast series, SAP's podcast series about the difference that open source can be. And in each episode, we'll talk with experts about open source and why they do it the open source way.

I'm your host, Karsten Hohage, and in this episode, I will talk to Alex Bennée and Helge Dellar about QEMU. I hope I'm pronouncing that right, but they all tell me.

Hi, Alex, and hi, Helge, nice to have you here.

Alex Bennée: Hi, Karsten, nice to be here.

Helge Dellar: Hi, Karsten, here's Helge, thanks a lot for inviting me.

Karsten Hohage: You are more than welcome. Let's look at who these guys are. Some of you out there may know them. Alex is a senior tech lead for virtualization and emulation at Linaro. Over his career, he has worked on devices from one-armed bandits to microwave communication systems. For the last 12 years at Linaro, he's worked on virtualization stacks from kernel virtual machine inside the Linux kernel to QEMU itself.

Helge is head of SAP's Linux lab, which is responsible for the development, certification and support for SAP products on Linux. And in his spare time, he is a maintainer of the risk or the PA risk Linux port and various areas of the Linux kernel. And he contributes to QEMU. That's probably why he's also here and other open source projects.

Now, one thing that we should say here, both of these guys are here because they're contributors or have some role in QEMU. We're not here expressing any Linaro or SAP corporate views on things. These are just individuals who are working on QEMU. That was a totally professional legal disclaimer. After that, I guess we can start.

First question, Helge and Alex, have you actually ever met in real life or has this all been a virtual friendship for you guys?

Helge Dellar: That's a good question. So actually, I think Alex and myself, we never directly met, at least we never shake hands or so. But I'm pretty sure we met somehow in between because when I started on QEMU in 2017, there was later on from the Linux Foundation, this kind of open source summit in Lyon in France. And there I visited it. And at the end of these ones, it's always the KVM forum. So, this is a meetup from the QEMU guys. And I'm pretty sure that... Alex has been there, and I talked to a few of the QEMU guys at that time. So maybe we walked by each other, but we never shaked hands.

In a personal thing, at least from a visible, I've met Alex because he also hosts a B-weekly QEMU developer conference call. And it's possible to see him and other QEMU developers. So, I think we never met. He's living in Southwest England and I'm at the SAP

headquarters close to Heidelberg. So, it's quite some distance. But yeah, I think it's like other open source projects. People know each other, but maybe didn't actually always met each other.

Karsten Hohage: Okay, Alex, did you notice Helge the other way around? Because it could have been at this KVM thing.

Alex Bennée: Well, I'm obviously aware of Helge's work because I see it all the time on the mailing list, him working with one of my colleagues to keep the old architectures running. The KVM forum is our annual conference and I've been to 11, 12 of them now. So, you meet a lot of people and you don't necessarily always remember everyone face to face.

Karsten Hohage: Okay. And there was one thing I was confused about just now. You live in England or in Wales?

Alex Bennée: In Wales.

Karsten Hohage: In Wales. Okay. Okay. And another thing, Benet, that's pretty French. Does that go back to the Normans or 1066 your name came over with William the Conqueror or how's that?

Alex Bennée: Uh, it's, it's a little murky. Our best guess is it came over with the Huguenots when there was a Huguenot migration from the continent.

Karsten Hohage: Yep. We got the Hugenotten in Germany as well when the French Catholics drove the Protestants out, right? Yeah. Okay. But this is not a history podcast really, although this could also be greatly interesting, definitely with someone from Wales.

Alex, let's turn to the subject; in short, QEMU. And am I saying that right?

Alex Bennée: There are lots of pronunciations, but I think the original formation is a quick emulator. So QEMU. So has been going a long time. It's about 22 years old as a project. Well, it was originally written to allow running of x86 Windows binaries on non-Windows machines. Although it never really got used for that in the end, it very quickly developed into a multi-architecture emulator which allows you to run binaries or even whole systems that have been developed for one architecture on another different architecture.

So, the common one I like to explain, most people who have PCs are familiar with x86, but there's a whole zoo of architectures out there. The most common one people will come across is probably the ARM architecture, which is what most mobile phones run, and even devices like the modern Apple Silicon Macs, basically ARM machines underneath.

Karsten Hohage: Okay, so one return question, maybe when you say architecture now, you're talking about processor architecture, right? Because I come from an area at SAP when we actually talk about the boxes and arrows of the larger landscape when we say architecture, right?

Alex Bennée: Yes, I mean CPU architectures or more technically an instruction set architecture. So, it's the set of instructions a particular type of CPU runs.

Karsten Hohage: All right. And in QEMU, what's your role again?

Alex Bennée: Well, so I work for a company called Linaro and we're paid to support the ARM ecosystem. And there's a couple of things that we're interested in. One is making development easy for developers. So, if developers want to use the ARM ecosystem or write code for the ARM ecosystem, rather than having to find the hardware and find hardware with the new enough CPUs with the new features, they can run a software model under QEMU and test all their changes under emulation. This is also very useful if you're doing things like debugging. It's often easier to debug in software. And if you want to do things like continuous integration and automation, QEMU performs a key part of that.

And the other part we're interested in is having reference implementations. So, we have a couple, there's a thing called the SBSA ref, which looks like a real ARM server. And it's a way that distros can test that their software works when you put a CD-ROM in a normal server. And it's also quite a common target for a lot of projects that target multiple architectures. So things like U-Boot or Zephyr will often have a QEMU target that they can run on. It's a good reference implementation as well.

Karsten Hohage: Okay, then I also understand from that, or at least that's what I'm getting so far, that it's largely a developer tool. My personal experience with emulators is limited to, I wanted to play age-old Monkey Island on a Windows 95 or later PC that didn't know about real DOS environments anymore, stuff like that. But I, I guess it's not for that, and it's also not really for replacing VMware or anything, because they do virtualization, not emulation, right?

Alex Bennée: Well, so this is the interesting thing. Although QEMU was originally developed as an emulator, part of developing an emulator means having to deal with the things like emulating pieces of hardware and devices. And that when KVM came along, which is basically an open source alternative to a product like VMware, it was very easy to repurpose QEMU to be able to run as a virtual machine monitor. So basically, to control hardware virtualization. And then all of the device emulation stuff you get for free.

And running the code either under virtualization or under emulation looks exactly the same from the command line, just under virtualization it will run faster. But the key differences with virtualization is you can only virtualize a processor on the same architecture for the guest and the host.

Karsten Hohage: Okay, and that already defines the difference between virtualization and emulation, right?

Alex Bennée: Yes. So, in emulation, you have to do steps on the host architecture to simulate what the guest architecture would be doing. Whereas in virtualization, you just let the CPU run and then you just deal with things like emulating a piece of hardware or dealing with storing the block device or interfacing with the network, that sort of thing.

Karsten Hohage: Okay, so could we in simple terms say emulation takes it a step deeper?

Alex Bennée: Yes, yes.

Karsten Hohage: Okay, and did I get that right before that QEMU is mostly used in development and quality assurance or is that also in production and operations or however you want to call that environments in use?

Alex Bennée: Both. So, although my interest is predominantly in supporting it for the developer ecosystem, it is actually used in production as well. So, QEMU forms the basis of the virtualization stack for things like Red Hat Enterprise. And there are a number of cloud providers that use QEMU as their base VMM for running things in the cloud. So yes, it can do both. It's a very flexible tool.

Karsten Hohage: Okay. Now, Alex, you've got a lot of talk time up to here. Let's maybe turn to Helge for a second there.

And Helge, you will get the task to explain if QEMU works like other emulators or if there is a significant difference.

Helge Dellar: Oh, I think there's a big significant difference. Like you and Alex just explained, QEMU is a great tool to emulate other machines on some other machine, right? in my case, it was really about pure RISC machines, old Unix machines. And they should be emulated. And for that, it's really my main goal at that time when I started developing for QEMU was the possibility to use QEMU to translate CPU instructions from one set, from the RISC machine, over to the Intel box, for example, on the laptop. So, my goal was to run another machine, the RISC box, on my Intel laptop.

And that's the big difference with QEMO. It really translates all the CPU instructions from the original one, in this case PRISC, over to the Intel CPU instruction. This is a big difference because then it means after a test translated it in runtime, so it's basically a JIT compiler, then it executes at full speed of the native machine. So, if you have started a program which was originally developed for another machine, in QEMU you emulate the hardware and it runs with native speed of the local machine. So you get really good performance, and I think that's a big difference because If you take a look at other emulators, they are emulating each instruction by itself. And that's possible maybe if you are emulating an old slow machine like an C-64 or an Amiga machine or so. There you can emulate each instruction by itself every time.

But this only works because the current machine is much faster than the old machine. If you try to emulate current CPU architectures, like for example ARM, ARM has become pretty fast CPU. If you do this, and by instruction, it will be much slower if you do it on local machines. So, there's a big benefit from QEMU from my point of view, what I see. You get native speed after starting these programs. And that's really, I think, the big difference, right?

Karsten Hohage: So, if I take that right, when you said all statements are translated into the target processor's language or in its command set, that means QEMU would take a bit longer to compile and thus with every change that you make. But then at runtime, basically, it would be more performant?

Helge Dellar: Yes, exactly. Just to just take a loop. If you are doing a loop, the first time it will be a little bit slower because it gets translated, but then it runs with full native speed.

Karsten Hohage: Okay. All right. I think I get it. And to take it to things that I might understand a bit easier, let's talk about the community aspect of it.

So, this is run as an open source project, right? Maybe Helge then again, does SAP have a particular part in it or do you just happen to work with it?

Helge Dellar: Yeah. Okay, it's my personal interest. It was really in my hobby time, but we actually also used it inside SAP or I'm pretty sure we still use it inside SAP without people noticing. One example where I know where we at least have used it and that was in my team, the Linux lab. That was when the first ARM processors came, which were targeting servers.

At that time, it was not that easy to buy physical machines to development. So, in our case, it was about compiling the SAP kernel for ARM server processors. And for that, we used QEMU, or at least we set it up, we used QEMU to emulate the ARM architecture. So, the initial ports were really actually done with QEMU emulation. Of course, it was not the speed, but it gives you the possibility to emulate it and to do the work, right?

Karsten Hohage: Okay, and you found that even for things like, I know that HANA is pretty performance touchy with certain commands or something. The emulation holds? It reliably...

Helge Dellar: Okay, for HANA, HANA team is another team. So if you were about HANA, what they did, I was talking about the SAP Netweaver kernel, the old SAP kernel.

Karsten Hohage: Okay.

Helge Dellar: But even for HANA, I'm pretty sure it would survive. Of course, you don't want to run it that way. You want to run HANA on a physical box.

Yeah, well, we were talking about trying out things, right? Yes. Not about running things. Yes.

Now, I was just wondering if the emulation is reliable, let's say, like for the, I don't know, performance behavior of left inner to right outer join or whatever kind of differences in large data operations HANA, for instance, would have.

It's pretty performance-wise. Just yesterday, I really used QEMU to emulate or to run a Debian x86 machine on my M1 Macintosh laptop. And it worked really native-performance-like on an x86 machine. So, it was really working like really good, let's say it that way.

Karsten Hohage: Okay, okay. I can only believe you at this point and probably there are tons of people out there who confirm that.

Maybe back to Alex. From a community perspective, QEMU is open source. Is it also open governance or does anyone have the thumb on it and always has to have the last say?

Alex Bennée: Right. Yes. So, to start with, yes, it's a fully open source project. It's licensed under the GPL B2 or later license. And it originally came from the community or more correctly, it came from Fabrice Bellard, who did the initial implementation. People may know him from other projects like FFmpeg is another one he's worked on. But then, as open source projects do, it accumulated a regular set of people that used to contribute to it. And eventually he handed over the maintenance to someone else.

But nowadays, we are incorporated under the SF Conservancy. So, the SF Conservancy is an umbrella organization, an open source umbrella organization, and they have a whole number of projects that they're responsible for, and they deal with all the paperwork issues, basically. Someone who can sign contracts, someone who can issue invoices or pay invoices. So, there is a small leadership committee, which is self-selected from the group of regular hackers. We try and balance it out so no one company has undue influence on the committee.

But the leadership committee is really only there for signing off, paying stipends, or agreeing where we're going to host the conference. All of the technical decisions are made by the maintainers of the project itself. And in that, it's like any other open source project. People volunteer to be maintainers. They get accepted by the other developers. They get added to the maintainers file, and then they're responsible for that area of the code base. I mean, the project itself, we've done surveys on the developers, and we reckon about 90% of the developers that work on QEMU are paid to work on it. Their employers either have an interest in engineers working full-time or part-time on QEMU for their own particular reasons. Linaro, where I work, for example, is funded by its members, but you'll also see contributions from companies like Red Hat, IBM, Intel, and many, many others that contribute to the upstream code.

Karsten Hohage: SAP?

Alex Bennée: I haven't seen SAP with an SAP-specific email address. Oh, my God. But obviously Helga contributes using his own spare time. Yes, so it's an open governance project. There is no benevolent dictator or anything at the top. We have three releases a year and generally one of the senior maintainers will take over the release for a release cycle. And their job is just to take the pull request from the maintainers and apply it to the main tree.

Karsten Hohage: Okay, okay. And by the way, Helge, as I just insisted on SAP, I guess, as far as I know, you guys at the Linux lab, it's sometimes a bit fuzzy anyway, where you do something personally and where you do something for SAP, right?

Helge Dellar: It can mix up, of course. We learn from the open source community and use it also, of course, for the production. So, it mixes together.

Karsten Hohage: Right. I guess that's what happens all over the place when we talk about open source community projects, right? Anyway, staying with community, Alex, how is it organized? Any particular process or the regular thing?

Alex Bennée: So, this is a topic that is one of regular discussion in the QEMU community, but the main way that we organize is via the mailing list. So, we have a main QEMU development mailing list, and then there's a bunch of sub-lists for different subsystems, but everyone generally always CCs the main mailing list. and that's where patches are posted, reviews are done out, and then when the final pull requests go in, they also go to the mailing list. The project is hosted in Git. We've been using Git for some time now, and the process is very similar to the way that Linux is done. As I mentioned earlier, we have maintainers for subsystems. They're the ones that gather the patches from individual contributors, and then they send pull requests up. And then we do three releases a year, so we've always got a regular cadence of releases and a nice predictable release schedule.

Karsten Hohage: Alex, I think concerning community still, I remember you told me something about a lot of downstream forks that don't come back upstream when we had some chats before this recording here. Is that a problematic issue?

It sounded to me like that. Or why did we mention that? I don't quite remember.

Alex Bennée: So, QEMU is quite interesting in the fact that it has a lot of downstream forks that are sort of forks that haven't fed back upstream. There's a couple of famous examples, for example, the Android emulator, the original Android emulator that is based off QEMU, but obviously Google have made a whole bunch of changes specifically for their use case, which weren't really appropriate to come back.

But there are other projects that have spun off, mostly in areas like malware analysis. There's the Unicorn engine is based off QEMU's TCG engine, but they stripped off all of the device emulation and all the other bits of QEMU and just concentrated on emulating the CPU instructions so they could do malware analysis. And there are a number of other projects that do instrumentation. So, I don't think these downstream forks are necessarily a bad thing. They show how powerful and flexible QEMU is. But if you need to make lots of invasive changes into the core code base for your particular niche use case, sometimes it's not possible to get that upstream.

Karsten Hohage: Okay.

Alex Bennée: That said, over time, there has been interest in getting something into the upstream QEMU codebase that can support some of these more interesting use cases. But the key is developing something that can be done in a lightweight way that doesn't affect the other common use cases that people have with the upstream codebase. Hopefully in time, we'll see more and more of the things you can do with downstream forks start arriving in the upstream code.

Karsten Hohage: All right. And then let's stay with things that are happening there for a bit. Is there any events in the near future or something where we can experience QEMU or meet some of the community?

Alex Bennée: We have a number of online events. We do outreach with a couple of projects, namely the Google Summer of Code and the outreach projects. They're paid internships so people can apply to do a project in QEMU. We'll allocate a mentor to introduce people to the code base and help be their first point of contact as they learn how to contribute to the project. It's an excellent way for students who are still at school or at university to get paid over the summer to make contributions to an open source project and build out their CV, so to speak.

Then we have a regular conference. KVM Forum happens every year. We used to be co-hosted with the Linux Foundation, but we're running it independently the last two years now. The next one is going to be in Milan. And that's the opportunity for a lot of the QEMU developers to get together alongside their hypervisor compatriots who work on the kernel and other related areas of the code base like LibVert. And we get together and discuss and do presentations on various work that we're doing.

Karsten Hohage: All right. I think we accidentally already landed in places to go. That was my fault. I was also still looking for some maybe interesting new developments in QEMU.

Helge, do you have anything?

Helge Dellar: I think you basically

already touched every point. If you're really interested in QEMU and where to find more information, it's really, I think, the best entry point is the website, of course. It has links to the GitHub sites and mailing lists and so on.

There was also mentioned the Google Summer of Code, which is also interesting. And in this area, I often get email requests, personal email requests from other people outside. Where should I start with QEMU development or if I'm interested in it? My answer often is, take a look at the bugzillas, the bug reports, which are open for QEMU and look through those if there is something interesting for you, what you would like to work on and then try to fix this issue.

And if you do that, try to send patches and bring it upstream, talk with the developers and just tell what you would like to change. And that's the way to start with QEMU development and to see all the benefits of QEMU.

So, if you're interested, look at the website, look at the bug reports, look at the git commit logs, and of course, try to visit the KVM meetings, for example.

Karsten Hohage: Okay, and discuss with Alex how to emulate a one-armed bandit from the 1970s on your current processor architecture or something. Did they have processors, armed bandits?

Alex Bennée: The ones I worked on were 6809s and 68000s, so yes.

Karsten Hohage: Okay. So yes, okay. I only remember from these days, I only remember one chip name that was the Z80. I think that was the 80s video games, right?

Alex Bennée: And the first home computer that I owned, which was the ZX81.

Karsten Hohage: Oh, that was a Z80 processor? I thought that was only in... How do you call them? Gaming Hall machines?

Alex Bennée: In the UK, we had the ZX81 and then the Sinclair Spectrum, which came from Sinclair Research.

Karsten Hohage: Ah, yeah, okay. I think I remember that one. I only had the C-64 because I was not on my way of becoming an actual developer.

Anyway, before we get lost into nostalgia things here, for QEMU, what would be your three key takeaways that you would like people to remember from this podcast?

Alex Bennée: Well, for me, I think the most interesting part about QEMU is its support for a wide range of CPU architectures. So, if you're interested in the details of how low-level CPU architectures work, then QEMU is a great opportunity to learn and experiment and even contribute to code changes to support that.

I would also say QEMU is used in a lot of places where you don't know it's being used below you. So, I've mentioned the downstream forks before. You've got things like the QEMU project, which emulates the original Xbox, and that's based on a QEMU fork.

One thing I should also mention, you were talking about interesting developments I should have mentioned earlier. We've recently merged support for the Rust language into QEMU . And the reason for this is that QEMU deals with a lot of untrusted guest data, because when you're emulating a guest, you want to provide a hardware model for it, but you don't trust the information the guest is putting in and collecting. that can lead to security issues. And Rust is a way of writing those device models in a more safe way where it's less likely that you could trigger buffer overflows and things like that.

Karsten Hohage: We've actually had an episode about Rust and I guess it's probably safer because during that podcast we coined the name the “Sergeant Major Compiler” because it's so heavily intolerant of anything suspicious or incorrect or fuzzy or whatever.

Right. Helge, maybe you have another key takeaway to add as these were all Alex's now?

Helge Dellar: Yeah, so I would really say if you are thinking about trying to emulate something, my suggestion would be take a look at QEMU. I'm pretty sure QEMU can do that in some or another way, because it has some possibilities to not just emulate in different ways. It has three different modes. If you really take a look at emulation, I would suggest take a look at QEMU. I'm pretty sure it's somehow supported already. And if not, try to contribute. That would be my last takeaway.

Karsten Hohage: All right. Okay. That was a great last takeaway. So, thank you, Alex. And thank you, Helge, for being our guest today.

Alex Bennée: Thanks. Thanks for having me. It's been great.

Helge Dellar: Thanks. Thanks a lot.

Karsten Hohage: You are more than welcome. And thank you. you all for listening to the open source way if you enjoyed this episode please share it and don't miss the next one we're not on a regular schedule any longer but you'll probably fully subscribed and will be notified and otherwise you'll find us on sap.com/podcasts and also in all your popular podcast players Apple, Spotify, and all the open source clients that you have out there etc., etc. Thanks again and bye bye.

Alex Bennée: Bye.

Helge Dellar: Thanks. Bye.

End of transcript. www.sap.com/podcasts | www.sap.com