2026-02-11 Core team technical discussion

- Date: 2026-02-11
- Participants: Raito, horrors, rootile, piegames, jade, kate, Qyriad
- Topic: Core team discussion on technical vision and alignment alignment

Conclusion / Action items

- @kate: suggestion, every topic we discussed above should be made into a Zulip thread ; please if you want to see a point you raised, please open a Zulip thread for it!

Meeting notes / Discussion

Relevant documents:

- Technical vision document WIP from @piegames (behind Lix SSO)
- Dreams

Agenda

- Context
- What do we want out of Lix on the short term?

Context

Kate explains there has been changes in the team activity, a need to update expectations and communicate on this.

We all came in this project with a laundry list of what is wrong with CppNix. A lot of us had a dream about what a Nix implementation should be.

In terms of what happened in practice, in the context of the a fall of a large nation to fascism, the fall of the NixOS project, we all are kinda in a situation where that laundry list of problems we had and opportunities we saw has become a responsibility for a lot of things.

We have to overcome the list of what is wrong and align on an actual technical vision.

Something we have all in common is we want to have this software being able to use it on day-to-day and have it work. @piegames worked on an evolution of Nix into Nix2. horrors worked a lot on the correctness, the nastyness of the codebase and improve the state of the codebase. There

might be also other people who want to do commercial-shaped things around. More implicit stakes are developed by people around the call such as @Raito's involvement with the public sector.

Lix on the short term?

- @Raito: Being able to use it day-to-day and not have it break and RPC: for remote building/CI/CD/AFNix usage
- @Qyriad: Being able to use it day-to-day and not have it break and RPC: for us, it's about enabling incremental evaluation usecases.
- @Jade: I pretty much used Lix exclusively at work these days, at work, we want operational visibility. Being able to write things in Rust instead of C++ would make it more easier for us to jump in and help. Having tracing of any sort, even coarse-grained tracing of "this pull request regresses eval tiem by 5 % — maybe you should not do that" kind of things. Being able to understand the impact of Lix performance is really important. The other thing related to that is I'd like to improve certain areas of Lix performance that are often a problem, especially store performance. Copying files into the store wastes over 30s/developer/day, probably more than that.
- @piegames: Language improvements iteratively and the tools. We are stuck with Nix and Nixpkgs, so might as well make it good. Good tools for distributed builds, good CLI, and evaluation APIs — remote evaluators. Bytecode interpreter, some JIT maybe, memoization, all that stuff. There's a lot of things to do. In the long term, component rewrite of everything we have in Rust, maybe something taken from Snix. With a language with a proper type system and more than an iteration of the Nix language. At this point, we are technologically stuck. What are options to do a next-generation Nix? Looking at it, there's no chance of getting such a thing out of the ground because it would die an immediate death of second system syndrome. You'd redo everything and not get anything done. I'd to build the platform where other people can modularize and do their own things: example with Flakes, I want to make it possible for people to build their own dependency management things. Same for the language, I'd like to standardize a semantic where people can build their own language and lower it to the bytecode and we can have piecewise evolution with an ecosystem that interacts with each other.
- @horrors: RPC. The store protocol we are using today is absolutely bad. It also binds us to CppNix decisions that were frozen in 2.18. Once we have RPC, many possibilities are opened. Evaluation RPC as Qyriad wanted is also much more feasible.
- @piegames: I'm not a strong user, I have deployed it on my personal system and so on. I'm not a power user of Lix. I'm not interested into getting caught into supporting a too-large userbase.
- @jade: One of the problems I had with Lix: we had this problem where querying substituters got regressed by a large percentage and it was impossible to run the 2.92 series; it was also really hard to measure the regression. I think there's a large opportunity if we are able to run more devbuilds of Lix at work than it will be possible assuming if we had some tracing of some kind. Then, we can turn that into signals of regression.
- @raito: Would like to bring the topic of opt-in voluntary telemetry for Lix in 6 months or more.

- @jade: 100% agreed and I'd like to suggest we look at OpenTelemetry traces. Especially because we don't need to specifically send them into a Lix blessed central place.
- @kate: 2 things are really important for the future of Lix: making the project sustainable — including the fact that the store is a problem for everyone — (background on the known inefficiences for the store: cache.nixos.org issues, etc.) *and* we are a very small team, we have very little ability to provide user support. One thing we need to work on sustainability is: how do we get more people to be able to take care of some of these roles that are difficult for such a small team? How do we get people who wants to profit off Lix give back? The second item I want to take the Lix monolith into multiple less-coupled pieces as much as possible. Take NixOS as an end user or Floral as an end user, I don't think there should be anything special there's a bunch of modules provided for NixOS. I don't think we should not have `nixos-rebuild` or `darwin-rebuild` where I would like to have some composability scheme where I want to do `nix $os rebuild` — I would like a strong enough plugin architecture where I want to build an hypervisor + VMs infrastructure concentrically without too much coupling. The whole ecosystem can work such in a way that `nixos-rebuild` is not THAT special cased in the long run. I'd like to enable lang versioning to happen easily by having this Nix-minimal fragment that we can convert things to. I'd like also to break out even things like the Git fetcher to interact via RPC. I don't want to care about how you fetch things.
- @piegames: We should focus more on what are the points of contention. Various different ways of getting there. Especially but not limited to community stuff. Because those are the points where we are not sustainable.
- @kate: I'm glad to hear we all have the broad same technical vision here. How we are getting there? Approaches like @raito or @horrors (more @horrors than @raito let's be honest) have been pushing us towards robust RPC. We make sure we have agreement on what are the priorities, we also need to make sure we have our expectations set right on the different paces and speeds so that our individual efforts meshes well. In the past, we had misunderstandings not based on differences in technical vision, but not what everyone else in the community was doing towards that technical vision.
- @kate: I suggest we take the 20 minutes left to identify the points we need to align on inside the pad. Also the points that would allow us to make a plan forward.
- @piegames: In terms of points I see: team structures reflected by a state of a codebase and we inherited a codebase from CppNix which reflects their own team structures and way of doing things. We need to break up how we work to better distribute the load but it also can be done if the software architecture follow through. RPC is the top priority then Rust is the top second priority because I believe those are what would enable us to onboard new devs without requiring super heavy mentoring like we did before *and* split up the codebase in more modular pieces. The other thing I want to talk about is the community side: it's a bit in the open air on how our community is looking. There has been discussions like Matrix is frankly a liability as a protocol. This is where also a lot of the socialization happens, even if it's not a lot, it's not nothing. We need to have a discussion about we want to proceed with Matrix and moderation. If we close down the Matrix channels, can we have the socializing bits on Zulip, etc. ? These are the pipelines which will get us new contributors in the project. We should be investing some resources into that. The community management in my eyes has been mostly fairly hands-off and

did not require a lot of efforts, apart from some very recent big conflict. I have spend more time dealing with Matrix issues and Draupnir being down and getting channels into Matrix *THAN* doing moderation stuff for the community. That's not sustainable. Something needs to change here.
- @kate: Let's keep this meeting about technical vision and let's have community vision calls as well.
- @horrors: The Blender community has exclusively moved to Zulip-like platforms, they do not have Matrix, they ditched IRC, they do not have Discord. It seems to be working really well for them. Zulip alone might not be a big of a problem. (NOTE(horrors): blender actually moved to matrix *after* we last looked, so. ugh. may have been rocketchat or whatever got eaten by matrix)
- @kate: Typical open source strategy would be to seek more volunteers but we are also having sustainability issues with volunteers. We wonder if we should have more boring people in our project who are able to work with the people who are willing to make computers work in a way that aligns with our technical agenda. We may be able to bring more hands that are going to work on this project because of a job / a money transaction rather than the architect who have profound opinions on how things work.
- @jade: I have been primarily interested into making things just work. I have had very limited emotional energy and this was one of the thing which made Lix hard to work on.
- @kate: Maybe we need a call to define what is the community and what it should focus on, what are the stakeholders, the community team.
- @raito: One problem as part of afnix as well:
  - Wants to make it possible for people to receive money to work on Lix
  - Make it possible for people to be paid to do things we want to have
  - Also including e.g. infrastructure or pay for infrastructure
  - e.g. working on Lixcon, would love to use this to community more of our agenda, especially to potential
  - Work on fundraising
- @qyriad: I'd like to have observability on what is getting built on my system, I want to remove a lot of barriers that makes Lix difficult work for new users. A lot of this is technically entangled which makes it difficult work on with these other big moving pieces like RPC.
- @rootile: our& shortterm vision: 1. kill functional1 with fire and spite; 2. get rust going bc we& won't touch cpp.
- @kate: Backward compatibility about CppNix and RPC, we need to decide whether we want to have backward compat daemon<->client. [@raito: lost notes because my brain crashed.]
- @jade: Most of the time, backward compat is needed by accidental situations, e.g. devshell.
- @piegames: graceful degradation / errors, my personal approach for langver is to let the old stuff rot and keep it for interop and at some point remove the old stuff

Things we wanted to discuss but did not / Agenda for next time

- raito: getting closer to an actual roadmapping document we can communicate
- availability (work-time & response-time) and expectations of core and non-core members
- tracing (zulip thread)
- piegames: Urgently needed devx improvements like a merge queue, less flaky CI
- piegames: agree on and clarify code owner semantics, committers etc.
- piegames: Documentation, issue triaging, wiki etc.

Revision #1
Created 2026-02-11 20:13:23 UTC by piegames
Updated 2026-02-11 20:18:11 UTC by piegames