Discussion notes

If you have a discussion or a meeting and are writing things down and want the meeting notes to persist, here is the correct place. Try to always move all relevant information off any scratch pads into appropriate more permanent places, like the wiki.

The template is optional and meant to help not forget any of the important aspects

Discussion notes template

- Date:
- Participants:
- Topic:

Conclusion / Action items

Meeting notes / Discussion

2024-04-29 – Lix Release Bootstrapping

Agenda

Prohibited items:

- Nix foundation politics, for everyone's safety

Core Agenda:

- Figure out an initial governance model we can use to bootstrap governance later
- Figure out how to do a soft release
- Figure out anything that needs to be done before this release

Other suggested agenda items:

- What to use for our contribution policy in the interim.
  - Raito going to come up with contributor doc, asking others for help.
- What to use for ~~code-of-conduct~~ community standard in the interim.
  - Jade going to either come up with community standards or make it happen otherwise.
- Contributor License Agreement T_T_T_T
  - None (thank goodness)

People

hexchen, Irenes, Irides, jade, ktemkin, puck, Osiria, Raito

Decisions made

- Bootstrap governance is made via:
  - Consensus first
  - 3/4 of bodies is quorum for non-governance decisions, otherwise it's the full set of bodies
  - 2/3 of the quorum number of bodies for a vote to succeed
  - Anyone can call for a vote at any time, including after a decision was made, except if a decision has been binding
  - Decisions can be binding or not, if they are not binding, they can be relitigated
  - Binding decisions are governance decisions
  - Focused on social governance but technical matters are a special case of social governance in the current model
  - Consult the governance before speaking on project's behalf for controversial topics
  - Delegation and on-the-spot decisions should be used carefully and consensus should be obtained back as soon as possible after such a thing is done

put here for edit history reasons

https://gist.github.com/RaitoBezarius/e8509c1bdae2980031e9630de4b6d69e

- We should accept PRs, from day one of soft launch, regardless of tooling status
  - "We do accept them, we will make it work."
  - If it gets bad we'll yoink the Go tooling

Action Items

- Write trivial community standards.
  - Write CoC that says "everyone is expected to follow community standards"
  - See governance pad thing with the general ideas. Want to write down principles of what we expect in the community.
  - jade owns this and causes it to happen
- Do something about continuous open beta
  - jade owns this and will make sure it doesn't get forgotten

Notes

Meeting starts at roughly 20:10 UTC

Governance Model

- irenes: how to avoid power structures being created by most jurisdictions forcing power structures for creating an entity. We would need an entity for money reasons due to infrastructure and needing to give people money
  - need to have an agreement on what we stand for overall
  - this is more of a consideration for the future
- kate suggestion: interim hypothesis: simple majority (of the core team list)
  - jade: constituency: people like Gabriella who are there for historical reasons but are not significant contributors
  - hexchen: 2/3 majority?
    - kate: ranked choice, or consensensus, approval voting also possible
  - puck: 50/50 split can happen, which can be a problem for simple majority
  - hexchen: for day to day decisions, 50% is good enough, higher level of agreement required to mess with governance
  - irides: consensus with majority vote to break ties?
  - kate: what is a quorum?
  - kate: if we can reach consensus on decisions, we can avoid doing things that people don't want/avoid using the voting system
  - kate: day to day vs governance decisions: what is a day to day decision? or we can treat every decision as governance decision
    - jade: what is a decision?
      - kate: some decisions are binding, some are made ad hoc. cannot rely on vibes to know when there might be disagreement

- hexchen: supports consensus for bootstrap but we do *need* a tiebreaker (2/3 reasonable)
- jade: should probably revert first in cases where things should need more discussion?
  - kate: stuff like potential security issues mean that revert-first doesn't work necessarily always
  - qyriad: revert-first favours the status quo
  - jade: this is kind of what we are doing
- hexchen: there is a difference between social and technical decisions, should focus to do social decisions in our governance model first. technical governance inflates the agenda for the meeting.
- raito: re revert first, consider a compromise: suspend people from involving themselves in a thing, for a period of time if things get heated?
- hexchen: re lazy consensus: make snap decisions, reëstablish consensus afterwards. we would prefer to not have to make such decisions, but if they are made, we should try to get consensus after they are made
- kate: use common sense, especially re speaking for the project. don't go around loudly speaking on the project's behalf without some consensus
- kate: what is the quorum? is the threshold *of the quorum*?
  - hexchen: possibility (remote for us) of excluding people maliciously and still forming quorum
  - kate: anyone outside the quorum can call revote?
    - jade: time bounds of revote?
  - puck: nixos foundation bylaws include some mechanisms to avoid quora being formed maliciously
    - some ways to deal with long term absentees
- kate: we want to make governance decisions in terms of actually making a real governance with the entire group of people. the entire group should be involved in the final governance making. everyone on the core team list should sign the final thing or so.
  two stage vote thing: vote initially and then everyone can change to be unanimous at the end as an explicit approval
- qyriad: concrete numbers: 2/3 is 7/10 bodies. 3/4 is 8/10.
  - kate: 3/4 for quorum, 2/3 for vote? everyone there -> binding decision, not relitigible
  - kate: something that is not relitigible is considered a governance decision, and thus requires a full quorum

summary by qyriad: generally try for consensus. when things have to be put to a vote, 3/4 is quorum, 2/3 of participants for vote. for governance quorum is everyone that can be reached. votes are done, for example, for decisions that either are expected to need agreement, or after the decision if agreement was necessary; also things that are project direction.

- irides: things that are votable but are not necessarily controversial.
- hexchen: how too moderation? generally want to reëstablish consensus after such decisions are made.

    - first put it on wiki then talk about it. kate: this is just a social expectation, does not necessarily need to be codified
- kate: need to write coc; general framework is using judgement based on safety of the community.
- raito: how does ownership of topics work? avoid cases of people doing diverging snap decisions on things outside their region
    - hexchen: delegating topics is something we need to do but not worry about *ownership* for bootstrap governance. delegate discussion space for certain things: e.g. ensure that discussions about things are in set places so that people see it.
    - irenes: consider how delegation is a hierarchical thing and avoid doing it immediately

Moving on to the soft release

- hexchen: I would simply drop a link on social media and let the everything spread!
    - Kate: just telling people in social circles and social media (save for Irenes and Kate) is probably good for a soft release
- Kate: Oh right we need to do binary cache for substituting Lix
- hexchen: are we still going to be reaching out to the Nix/Tvix/Guix developers a few days beforehand?
    - Kate: Nix kind of already got their notice
    - Raito: Most of the Tvix members already know about Lix
    - Kate: the soft release also kind of *is* notice for the hard release
        - But we can also send the Guix devs an email or something
        - Irides: why are we doing this in the first place?
        - jade: for security reasons! Nix's security response stuff has historically been disasterious
        - Raito: we share componenets in our codebases with Guix, so it makes sense to communicate in general
- hexchen: should we be PRing Lix to Nixpkgs for soft release?
    - Kate: we probably shouldn't touch Nixpkgs until the Foundation stuff stops exploding
- hexchen: I would *really* like to avoid making Lixpkgs a thing
    - Osiria: gods we would really like this too
- we want to reach out to the guix project
    - raito: should we reach out to the guix lead
    - general agreement to just reach out to them politely to say that we exist
- jake hamilton: decided not to disclose to him
- puck: maybe we should just tell Guix at the same time as the soft release?
    - jade: doing it ahead of time is a show of good faith
- raito: re jake hamilton: their fork is a "we are trying to do this", don't think they have the relevant skills/insider stuff to really get far soon
    - jade: doesn't really see any advantage to disclosing early
    - kate: we're going to be out so soon anyway
    - kate: also doesn't really see a *downside* to disclosing to them ¯_(□)_/¯
- hexchen: meow
    - maybe now is a good time to do a short couple-minute break

- reconvene at 21:25 UTC
- hexchen: how do we want to deal with expanding the core team?
  - should we continue the freeze?
  - Osiria: yes, we absolutely should continue the freeze
  - jade: the bootstrap model has loopholes that are exploitable. anyone to be onboarded would have to be deeply trusted
  - Kate: after we open up the "core team" distinction will likely break down as we do things more out in the open
  - jade: a lot of stuff lands in the core chat either because it's sensitive, or because Matrix is kind of bad, which actually really sucks
    - Having a less bad chat platform enables things more

~~Code of Conduct~~ Community Standard

- Kate: stuff on the wiki will probably be what we adopt long-term
  - In the meantime we should adopt something in the interm
  - Irides: maybe we should adopt something that just expresses intent instead of a traditional code of conduct
  - Kate: that's kind of a code of conduct with more steps — if someone has the time/spoons to write something up feel free
  - Raito: the team composition also kind of signals some degree of intent implicitly
  - Kate: code of conduct or not we're kicking out nazis and that expresses intent too
  - jade: coc explicitly a bad term
  - Irenes: if someone actually tries to call us out for no CoC yet we just point out that we actually have a functioning governance model and we're taking the time needed to figure out CoC stuff
  - hexchen: should we just do contributor covenant?
  - Irides & jade: that document would actually be a detriment
  - Kate: re: CoC vs CS: we just put a code of conduct that says "follow the community standards"
- Kate and jade both offered to do writing stuff
- How about jade takes ownership of the thing, and go ahead and probably write it, but if she doesn't/can't/etc then she can poke Kate who will write it
- hexchen: should we subscribe to NixOS's Matrix banlist?
  - Kate: pulling it in, sure, subscribing, probably not
  - jade: we can follow first and review after, or review first and follow after
  - Kate: we can also just follow their moderation decision repo
  - hexchen: draupnir can send a message when the banlist updates. or we can just have someone in the banlist room and action it

Contributing Policy

- Kate: this should be liftable from somewhere?
  - jade + Irides: what is a contributing policy?
  - Kate: it's the set of expectations we have for contributors
- We already have some stuff on the wiki
  - jade: Should do some crosslinking between CONTRIBUTING.md and the wiki

    - jade: Maybe find a user to experiment on
- Irides: a lot of contributor documentation is not very good
    - Raito: nixpkgs mood
    - We should have good onboarding process documentation
    - puck: we may be conflating "how to use our tooling" and "what kind of contributions are we accepting"
        - External contributors should probably be aware that we're not generally making large changes right now
        - Irides: the wiki already has a document on that
        - jade: ideally before you write a single line of code you should know this information
        - Osiria & jade: a clone/devShell/pre-commit hook that informs the user of our freeze-state
        - Raito & Irides: this is largely a social problem
        - jade: can we put the contributing channel in the devshell hook or something?
            - osiria & irenes: explicitly say "we want to talk to you", link to the channel
            - puck: put CL push webhook into a room maybe? not necessarily now. generates sense of activity
            - kate: specific dedicated channel for "prospective contributors/feedback"
            - puck: concerned about discussion fragmentation with many channels
            - jade: maybe like... a threadbot? that makes temporary channels because Matrix threads suck?
            - puck: more projects we need =P
            - Kate: I would simply make Zulip have a good UI
    - jade: how do we MAKE people read the "freeze policy" or so
    - jade: should there be synchronization between contributing document and the wiki (probably with repo as source of truth)?
    - Osiria: noting: what should the wiki/forgejo/gerrit permissions be for new logged-in-with-github users?
- jade: if the contribution document just links to the wiki, you force people to go the wiki
    - if it has a bunch of links to the wiki, no one will read the links
    - maybe should just be a summary
    - kate: "ask first, we will help you, here is where to ask"
- jade: noting: do something about the github pr bot thing, or set up a manual process in the interim
- jade: what should we do about github PRs?
    - Thinks we should accept PRs, from day one of soft launch, regardless of tooling status
        - "We do accept them, we will make it work."
        - If it gets bad we'll yoink the Go tooling
- Kate: if we're going to do governance stuff on git, *there* we should probably sign our commits
    - jade: aggressively sighs and concurs
- Osiria: permissions for soft launch?
    - Kate: for soft launch, signin w/ Github is disabled for the public

    - jade: we really really want to have the majority of accounts especially from people we don't trust be from github due to being harder to make socks on it
  - Kate: or we could do local accounts
  - jade + Osiria: that breaks bans and is SSO-messy
  - Kate: deferred, put it on the soft release board
- Kate: CLAs suck and we're stuck with LGPL forever?
  - Irides: throw MIT license and Apache license in there, and future contributions are licensed under *all three* licenses, allowing us to eventually migrate
  - Kate: that doesn't really work since there's all the existing licensed code
  - Irides: MIT and Apache are LGPL-compatible
    - Really would not like to be confinded to LGPL forever
    - We already have in-tree mixed licensing
  - Kate: we can't do that for anything that touches LGPL code
  - Osiria: defer to asynchronous
- Decision: no CLA
- Raito: why a permissive license anyway?
  - Irides: deferred to asynchronous
- Irides: we cannot *possibly* do worse than Nix at this point.
  - jade: that fact that we're doing *any* of this is already a major improvement

---

- Raito: how do we transition from beta to soft release? what do we do with the beta rooms?
  - we just leave them
  - jade: these are people we generally trust, and who are politically all on the same page. we must continue to have this space exist, and also not grow it
  - Raito: we need to address the fact that people want to add people
  - jade: not opposed to growing them, just opposed to growing them to people who don't meet the same demographics as the ones who are in there
  - Kate: try not to stray too far into politicking for now
- jade: all the people in the beta are running main, and that's good actually, and if we could get more people to run main that would be awesome
  - great for feedback
  - maybe even have a channel for "I'm running main", and have a permanent open beta
  - Kate: sounds good! make an issue :)
- irides: how should we do linking to scratch?
  - kate: can put sso on it possibly
  - jade: the link can be shared to more people

Bootstrap meeting complete! Well done everyone ♡

Roughly 22:25 UTC, meeting end

2025-06-27 Wiki work

- Date: 2025-06-27
- Present: jade, piegames
- Topic: Information organization in the wiki and the homepage

Summary: We have a mess of information in the pad system which is not well organized by quality, and it's hard to deal with it.

Action items

- Create a new book for founding-days Lix pad in the Wiki, migrate stuff there
- Create a place for meeting notes in the wiki, with a nice template and everything
- Clarify whether people are intended to edit each section of the wiki (or like somehow blanket make it clear that it is okay absent further information)
- Maybe document a procedure for taking a matrix log (with LLM prompt for initial phase?) and turning it into more useful information
- piegames can take care of migrating the governance-related pads, in the sense of finalizing them and getting them approved and making them official (this will take a lot of time though)
- Rework https://wiki.lix.systems/link/9#bkmrk-freezes to reflect current processes
- Create a "Developers Announcements" channel for low-volume communication between all devs
- Migrate https://lix.systems/resources/ to the wiki

Meeting notes

- Organization of the design documents is confusing
- Need to explain contributions better
  - The "how to not write C++" is ill-located
- Need governance meta related channel
- We should probably move the resources page
- We probably should have better organization of the "lix manifesto" related stuff
- Related projects is ill located
- Unclear where we want to put each thing, e.g. do community standards go on the website
- minor wiggles moment of the quicklinks page being added to the website but not being linked anywhere
- Stuff like cultural values and so need to escape the pad zone
- IFD discussion pads
- Discoverability problems
  - Wiki > Pads > Matrix
- Pads problems
  - We sometimes archive matrix discussions into pads

    - Throw them into a large language model lol
    - Eg. improving IFD pad
- Who does this work of archiving and librarian stuff
    - Could jade do this while tired at work?
- Information organization
    - Wiki: stuff that should not connote endorsement and does not allow people to mess with wiki
    - piegames: this is kind of a problem for the governance pages
    - The website has no way to have table of contents and nesting support
    - Failure mode of not caring too much about too wide edit permissions is that people don't think that they are allowed to edit it
- RFC process where it gets committed to a repo means that you know what actually is committed to
    - In general we cannot afford more process
- This will burn out piegames if they have to do all the coordination labour in the project
    - How do we make this sustainable?
- We can avoid having to do more of this by having a very minimal quality bar for the wiki and accept that it is going to become a mess
- Archived stuff
- Space for meeting notes
    - Write meeting notes with a template that requires that it says who was doing the meeting
- Issues with pads is that we don't separate trash and useful information
    - Pad index is a mixture of stuff that is partially dev log and partially dev result
    - We should organize the dev log by date/topic, rather than by topic
- piegames prefers a process and end result type note structure rather than these, which are a headache

2025-06-26 Lix NixOS Module

- Date: 2025-06-26
- Participants: Niko, k900, piegames, Raito
- Topic: The current state of the Lix NixOS module (https://git.lix.systems/lix-project/nixos-module/)

Conclusion / Action items

☑ piegames will reach out in *Lix on main* asking for volunteers

- Primary goal is to get rid of the module by integrating its overlay-style functionality into `pkgs.lixPackageSets`
- Stretch goal is to have a `lix` module in the NixOS name space for additional configuration

Discussion

Selective transcript from a Matrix discussion

- niko: Is lix-module something that's planned to be gotten rid of? I'm asking because since Lix now requires cgroups for auto-allocate-uids, whereas cppnix doesn't (unless I'm wrong) then it'd be nice for lix-module to extend the nix module and add checks for stuff like whether cgroups is enabled if auto-allocate-uids is, and similarly to check whether nix-daemon service unit has cgroups configured, and other potential differences that come up that nixpkgs doesn't necessairly take into account
  - raito: answer is yes
- k900: I think we can upstream all of that to nixpkgs tbh
- piegames: Even the lix overlay?
- piegames: This functionality should still be available for non-NixOS though. The module does little more than setting the nix package option and applying an overlay that does all the nix lix substitutions in dependents
  - k900: But we can have things in lixPackageSets that are lix-overlay-shaped
  - raito: lixPackageSets does it too – lixPackageSets is the Nix-dependent universe reinstantiated with Lix
- raito: If someone wants to take the lead on Nixpkgs packaging, I can provide guidance, review time and help, but I cannot spearhead this until we landed the start of RPC in Lix I'd say
  - piegames: Do we want to publish a larger call-for-participation on this? Maybe on *Lix on main* for example. Raito would you be open to mentor such a thing?
    - raito: Agreed
- k900: What are we missing on the nixpkgs side?
  - raito: Double checking there's nothing missing between the module, implementing a deprecation notice on nixos-module side, I'd say. Instructions & docs

    - k900: Oh you mean not nixpkgs packaging of lix, but absorbing nixos-module to nixpkgs
    - raito: I meant the nixos-module replacement inside nixpkgs, yep
    - raito: Possibly, designing for `lix.enable = true;`, etc. We are increasingly diverging from CppNix, the `nix.*` namespace will feel constrained at some point
- raito: `import <nixpkgs> { config.nixImplementation = "lix"; }` something like this, which dumbs it down to an overlay or anything idc

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.

2026-03-11 DevX meeting

- Date: 2026-03-11
- Participants: piegames, Qyriad, horrors, rutile, raito (15 minutes late)
- Topic: Lix DevX

Agenda

- On Doxygen
- Merge queue
- Other DevX issues folks are facing?

Meeting notes / Discussion

On Doxygen

Objective: document the RPC protocol APIs in Lix, esp. the async bits.

Takeaway from @piegames: Keep Doxygen for now, but don't rely on it for new documentation. Improving it somewhat would be possible but require someone to step up and own the effort.

Actionable @Qyriad: Publish the API documentation on docs.lix.systems, reusing the nightly docs pipeline. [Send this information to AFNix]

Merge queue

State of things: autosubmit bot, only works for already rebased CLs. Wanted: merge queue, that automatically rebases applicable commits and merges them.

Raito: can implement the most trivial change but only in 3rd week of April.

Usual annoyances that folks silently accepts that we should not

- Non-flakey CI @piegames
  - @pennae: functional tests should be unflaked by migrating them
  - @pennae: NixOS tests cannot be unflaked
    - Unless replaced entirely
    - All NixOS tests failures are related to the virtual Ethernet switch use and the setup sometimes fails and doesn't get detected or fails and reports success.
    - It seems that network operational readiness reporting is failing and we do not know why. networkd reports that an interface is operational but it does not say that a service is ready.
    - `wait_for_open_port` is the thing that blocks and never completes.
    - NixOS tests failures are highly correlated with CI loads.
    - The best solution is not to use multiple nodes.

        - Proposal 1: Turn every multi-nodes NixOS tests into a single VM NixOS test containing multiple containers (implementing the multiple nodes).
            - Implementation details: have systemd spawn the containers and write a pytest-based framework to run f2-style tests against the containers spawned inside the single VM.
- Justfile could get better @piegames
    - Justfile to build Hydra tasks
        - Wants to provide the Hydra test name
        - @raito: nix-build -A hydraJobs.tests
        - @piegames: I did not know this! This is why I want Justfile!
    - Justfile task to build the reference manual & other documentation
        - without doing nix-build -A lix.dev
    - @raito: I want the `-custom` split to disappear
        - @Qyriad: if it takes arguments, it cannot take multiple targets at once
        - `just setup && just build` vs `just setup build`
            - Qyriad: currently: `just setup && just build-custom manual` works
        - **@raito will own this and run a poll on this topic on the Zulip thread and implement the popular vote**
    - List all available build targets: `meson introspect build --targets | jq '[.[] | .name] | sort | unique | .[]' --raw-output`
- `just test` requires `just install`, even functional2 requires that!
    - Proposal: make `just` do a `just install` for `just test-functional2` *and* `just test`
        - Qyriad: we cannot make `meson test` automatically install, but we can make `just test` automatically install

F2 needs

- @rutile: command wrapping
- @pennae: i need command wrapping as well, esp. for the PTY stuff
- @pennae: ability to asynchronously communicate with a running command for interactions

Actionable @rutile (?): adding async(io) wrappers for command subprocesses (stdin/stdout/stderr)

2026-03-13 State of the Matrix

- Date: 2026-03-13
- Participants: piegames, Raito, hexchen, Qyriad,
- Topic: Community team discussion on Lix's Matrix server, space, and channels

Conclusion / Action items

- ☑ Make a backup the current database
  - Owner: @afnix @raito
- ☐ Check the PL levels are what they should to allow removing Draupnir
  - Owner: @piegames
- ☐ Get rid of Draupnir
  - Owner: @piegames
- ☐ Nuke our server and deploy a Synapse
  - ☐ Writing a plan and a rollback plan incrementally
    - ☐ Owner: @hexchen @raito
  - ☐ Execute the plan in a distant future afternoon (post-LixCon)
    - ☐ Owner: @hexchen @raito
- ☐ Announce the consolidation plan on blog + Matrix announcements
  - Owner: @piegames
- ☐ Tombstone and redo the #space:lix.systems room to heal the split brain
  - ☐ Consolidate the rooms via tombstones
  - ☐ Upgrade all our rooms to v12 via tombstones
  - Owner: @hexchen @piegames
- ☐ Update our documentation
  - Owner: @piegames

Meeting notes / Discussion

- State of the current Matrix issues
  - Draupnir is broken, homeserver is broken, also possible network split
- piegames: So, how much Matrix do we want
- Raito: architecture proposal
  - The core issue is the way we deploy Matrix
  - If we just do a boring synapse thing we should have far fewer issues
  - piegames:
    - issues with draupnir
    - room state issues

        - e.g. draupnir issues aren't necessarily homeserver problems
            - likewise with room state desyncs
        - Swapping homeserver is necessary but not sufficient
    - Matrix, unfortunately, is currently one of the main ways people get involved in the Lix project
- piegames:
    - Reduce number of rooms
    - Remove Draupnir and do manual moderation
        - hexchen: we lose importing bans from NixOS moderation
            - …But that ship kind of sailed anyway
        - Raito: are they even doing bans rn anyway
        - hexchen: don't think so
            - We don't have a lot of common ground with them at this point
    - Once we replace Draupnir, our server has zero state
        - The things that matter can be recovered via federation
    - Should we consolidate rooms?
        - Right now we have (besides moderation rooms):
            - Lix Dev
            - Lix Off Topic
            - Lix
            - Lix on main
            - #off-topic
            - Lix project infrastructure
            - Community
            - Announcements
            - functional2
            - Nix lang 2
        - Raito: kill Lix Dev, Lix on main, Community, Nix lang 2, Lix project infrastructure
        - piegames: Maybe keep Lix on main
            - It has better SNR than the Lix
            - Qyriad: I agree
            - raito: fine with it
    - hexchen: we need to make sure draupnir allow to elevate the other users to the right privilege level we want
    - hexchen: re: room v12
        - All members of the community team should be set as Founders

2026-03-18 - Governance weekly

- Date: 2026-03-18
- Participants: raito, horrors, piegames, rootile, kate, Qyriad.
- Topic: Lix governance weekly

Agenda

- Job channel
- Inactive core team members
- Retrospective on the 2.95 release situation
- Various status updates from @raito for AFNix
- LixCon 2026 status

Conclusion / Action items

Job channel

Lix will postpone exploring a job channel idea until past LixCon and will mesh this problem with a sponsorship policy.

Inactive core team members

https://git.lix.systems/lix-project/lix-website/pulls/69 is approved and merged along

https://git.lix.systems/lix-project/lix-website/pulls/70.

- Lily Foster
- 9999years (Rebecca Turner)

are disboarded from the core team role in the Lix project. Thanks for all their help and support towards the Lix project.

@piegames will adjust the governance document to better match our current procedures.

The Lix core team will examine in a future meeting whether to maintain a "historical core team member" section.

Retrospective release situation

Split into two topics:

- the current 2.95 crisis, @raito @kate @horrors are addressing it.
- how to avoid repeating this mistake? post-poned to future meetings as we would like to have Jade part of this.

Status updates from @raito

- AFNix implemented the fundraising proposals in https://zulip.lix.systems/#narrow/channel/16-Governance/topic/Fundraising.20in.20the.20name.20of.20Lix.20for.20AFNix/with/8503 on three fronts: the Open Collective is set up by @delroth, Matrix announcement and social media post has been done by @raito. Next will be Open Collective link in manual and project README.
- Currently, the AFNix project representative of Lix is @raito, @raito would welcome someone else to decrease the CoI issues of being AFNix president as well.

LixCon 2026

- @raito expects more core team talks for LixCon and will pester core team members in private.
- @raito needs graphical design for LixCon, @hexchen seems sick and someone needs to take over. @kate offered to work on this.
- DGNum proposed a meeting at M-1 for LixCon but Lix did not respond. @kate propose we set up a weekly (in 4 weeks) between DGNum and Lix for LixCon. @raito proposed a scheduling link to both sides.

Future agenda

- Examining our posture on perfectionism vs incremental changes.
- Historical core team members section.

Meeting notes / Discussion

Agenda:

- Job channel
  - pennae: We want to do it mostly for employees than employers.
  - pennae: We should maybe make it the other way around and let people post they're looking for jobs.
  - kate: it would prove difficult for companies to actually operate with this due to the whole bias.
  - kate: having employers post ads as long as they behave in our code of conduct.
  - let's shelve this topic as there might not be enough demands to justify the costs of doing it right, at least, after lixcon.
  - @piegames own the fact of starting a draft on sponsorship policy.
- Inactive core team members
  - Merge https://git.lix.systems/lix-project/lix-website/pulls/70 and https://git.lix.systems/lix-project/lix-website/pulls/69
  - Qyriad: what about the proposal for alumni/emeritus section?

    - Kate: thinks this would be good to be able to list historical core team members to permit core team members to use this for hiring decisions.
    - pennae: wouldn't archive.org satisfy this objective too?
    - Kate: a hiring manager would not bother with that.
    - No objection to merge these two PRs.
    - @piegames will adjust the governance document to better match our current procedures
- Retrospective release situation
    - QA is required for 2.95.1 & 2.94.1
    - releng for 2.95.1 is owned by @raito if no one else will do it
    - Feature / merge freeze policy for releases owned by @raito but after LixCon
    - Testing commonly used 3rd party programs (`direnv` / `devenv`) with new Lix releases
    - Static builds should go into Hydra for the next releases, not sure we should put them back into CI
        - expensive
        - we do not need them very often
    - AFNix: convert releng to Hydra and automatic things
- [Status update, Raito] LixCon talks from the core team
- [Status update, Raito] Fundraising for Lix project [AFNix]
- [Status update, Raito] Sponsorship of Mercury in LixCon
- [Status update, Raito] AFNix project representative [AFNix]
    - https://docs.afnix.fr/policies/afnix-membership.html
- [Status update, Raito] LixCon 2026
    - We need graphic design for LixCon
    - DGNum team has proposed a meeting
    - Scheduling a weekly between DGNum and Lix for LixCon

2026-03-27 - LixCon 2026 weekly

- Date: 2026-03-27
- Participants: raito (Lix/DGNum/AFNix), sinavir (DGNum), soyouzpanda (DGNum)
- Topic: LixCon 2026 weekly

Call for Proposals (CfP)

- Proposals received: 8-9
- Core team talks and transactional conference talks (intro/outro, etc.) not included in the Pretalx
- Some invited talks are pending, awaiting final approval
- No issues anticipated regarding the current lineup.

Proposal review plan

- 1 proposal already accepted for logistical reasons (travel support)
- Review schedule:
  - Tonight: @raito reviews proposals
  - Tomorrow (or so): @kate reviews proposals
  - This weekend or Monday: converge on final proposals
- Goal: complete review quickly to submit acceptance or rejections to our current proposals

Remarks

- Overall proposals goes over the Friday talk budget but aligns with overall planning
- No conference rooms required for Saturday and Sunday

Special attention AFNix general assembly room

- Requires Internet setup for videoconference
- Capacity: ~10–15 people

Postcards

- **Deadline for order** 2 weeks before event; faster option: 1 week (+€2)
- Kate is aligned with Monday deadline

Content

- Decide back-of-card content:
  - Important information and emergency contacts
  - Room list / orientation tips
  - WiFi info (if a shorter deadline)

Assembling method Punch postcards without plastic coating

Inspiration: Good-to-Know PDF from UndoneCS 2026

Website

- Let DGNum host this website: https://lixcon.dgnum.eu on their S3 (Garage) instead of AFNix except if AFNix redeploys it quickly
- **Announcement post** email + Mastodon (high priority) → owned by @raito
- **Content to finish**
  - Schedule → Pretalx widget
  - Venue info → owned by @soyouzpanda
    - Rooms
    - Internet
  - Travel information
  - Physical Code of Conduct (LixCon/AFNix) → owned by @raito
  - Cross check with https://www.undonecs.org/2026/ information to see if we are lacking something
  - Ideally: RSS feed
- Optional: DNS CNAME `https://con.lix.systems/2026/` → main site (low priority)

Equipment rental

METRO refers to https://www.metro.fr/ here.

Fridges and Drinks

- Drinks brought by @hexchen and our METRO provider
- Suggested setup: 1–2 fridges in hacking rooms
- Suggested: additional payment terminal purchase (SumUp Plus) as AFNix one might be used for other purpose
- Deposit may apply to drinks, @sinavir will check with @hexchen directly

Snacks

- METRO card available
- A car can be lended by @sinavir
- Which snack to buy / who will buy them → Can Anaëlle handle it?

Seating and tables

- Cozy sitting (beanbags, etc.) for friendliness in hacking rooms → owned by @sinavir
  1. Source price
  2. Determine quantity
  3. Propose quote
  4. Execute if approved

C3VOC equipment

- Coordination for setup → @raito connects @sinavir with @hexchen
- Free parking at ENS: confirm dimensions and availability for the van

Room planning

- Core team members to book tickets explicitly on Pretix for full visibility on DGNum → owned by @raito
- Expected variance: ±10–20 people
- Goals:
  - 3 public rooms
  - 1 non-public room
  - Release remaining rooms
- More than 4 continuous rooms over the weekend is non-trivial
- Final room list to be confirmed by @soyouzpanda and @sinavir
- @raito wants to be kept in the loop and this topic should be discussed further next weekly

Network setup

- Backbone v2 functional (Ielo uplink), deployed by @sinavir
- Connect LixCon rooms to Internet: requires coordination with the school IT to switch VLAN on the Ethernet ports in the rooms, owned by @sinavir
- Room setup: 1 switch and 1 AP per room min
  - 4 APs for the amphitheater
- Multiple SSIDs (firewalled and non-firewalled, OWE)
  - Non-firewalled is not a priority
- Forensics logging: DHCP leases and MAC/IPv6 via NDP as per law requires
  - IPv4 is already done in the current network
  - IPv6 will be new

Priorities

1. Secure VLANs in rooms
2. Secure hardware (APs, switches, cables)
   - Friday: 4 + 3 APs, 4 switches
   - Saturday: 6 APs, 3 switches
   - Sunday: 6 APs, 3 switches
3. Configure router with forensic logging (DHCP + IPv6)
4. Configure switches
5. Configure APs
6. Optional: non-firewalled SSID

lead by @sinavir, will delegate whatever he can to other folks

Next agenda

- Room planning
- Network setup
- Commit to conference start/end information to communicate to our attendees
- Approve the final operational details to send to our attendees

2026-04-03 - LixCon 2026 weekly

Participants: raito (Lix/AFNix/DGNum), Kate (Lix), soyouzpanda (DGNum), sinavir (DGNum) Topic: LixCon 2026 weekly 2/4

Agenda:

- Calls for Proposals
- Postcards
- Website
- Equipment rentals
- Distinguishing staff from attendees
- Network setup
- Room planning
- Accessibility
- Conference start/end times
- Final operational email to attendees
- Physical code of conduct

Calls for Proposals (CfP)

- @raito: CfP wrapping up, schedule almost ready, will integrate into website.

Postcards

- Kate: designs ready, awaiting approval.
- raito: aligned with Monday deadline.
- Stickers for attendee names:
  - soyouzpanda: can buy sticky papers and print names.
  - @sinavir: will handle small materials purchase.

Website

- Kate: possible redesign to fit more into the Lix theme, low priority.
- @raito: schedule to include Pretalx widget; DGNum reviewing/polish ongoing.
  - Physical Code of Conduct is blocked on Lix team.
  - RSS feed pending on soyouzpanda.
  - Target website announcement around Monday.

Equipment rentals

Drinks

- Number of Fritz Cola: @sinavir to confirm with @hexchen.
- Deposit questions:
  - Kate: standard German bottle deposit applies; returning bottles reduces next purchase cost.
  - @sinavir: hexchen will collect bottles; can adjust order accounting.
  - @raito: money handling to be figured out later.

Fridges

- Proposal by Mathieu; @sinavir to obtain a quote.

Payment terminals

- AFNix: 3 terminals (2 @thubrecht, 1 @delroth).
- DGNum: 2 terminals already ordered and obtained.

Largely enough.

Snacks

- @sinavir: plan to buy usual student bar items; leftovers can be redistributed.
- @raito: requests list of student bar items.
- @soyouzpanda: list is here.
- @raito: will create final list for snacks/coffee.

Seating and Tables

- Discussion on this topic adjourned due to @sinavir leaving early.

C3VOC Equipment

- @sinavir coordinating with @hexchen.
- Remote speaker: pre-recorded session + live interactive Q&A.
- @sinavir: confirm ENS parking max height for C3VOC van.

Distinguishing staff from attendees

- soyouzpanda: staff badge holders need distinction; color-coded lanyards suggested.
- Kate: suggest to procure lanyards with text to identify staff.
- soyouzpanda: student association has 12 organizer armbands.
- @raito: both lanyards and armbands to be used.
- soyouzpanda: owns procurement of lanyards.

Network setup

- @raito: school uplink not used due to IT human resource limits.
  - Consequence: no IPv6.

    - Nonetheless: wired Internet, APs, private VLAN available.
- @soyouzpanda: alternative solutions may exist (@sinavir).
- Alternative: routers connecting via VPN; MTU adjustments likely.
- Action items:
  1. Communicate limitations → @raito
  2. Invite attendees to rely on LTE/5G → @raito
  3. Prepare local Nix(OS) cache → @raito
  4. Prepare B plan → @sinavir

Room planning

- @sinavir: all fine according to previous communications.
- Requirements (from prior meeting):
    - 3 public rooms including quiet room
    - 1 non-public room

Accessibility

- Kate: per European accessibility act:
    - Audit all graphics (screen reader alternatives)
    - Provide paper documentation at 200% scale
    - Comply with subtitle/sign language requests
    - Publish a11y contact email and designate on-site contact
- Action items:
    - @soyouzpanda: audit website with RGAA checker, publish a11y contact
    - @raito: check with @hexchen regarding captioning/sign language; announce the on-site contact at the intro talk

Conference start/end times

- Friday 8:00 staff / 9:00 public check-in → 10:00 first talk → 18:00 public end → 20:00 staff
- Saturday: 9:00 public check-in → 18:00 public end → 19:00 late (no entrance anymore) → 20:00 staff
- Sunday: same as Saturday

Plan for teardown: grab volunteers, no formal plan needed.

Final operational email to attendees

Dependencies: website ready, conference schedule and start/end finalized

Checklist:

- Conference start/end times
- Travel info (public transport)

- Schedule of talks
- Lunch/dinner info
- Encourage interest-based group formation

Physical code of conduct

- @raito to coordinate with Kate; implement quickly.

Next agenda

- Seating and tables (beanbags)
- Snacks, coffee, and hacking snack plan
- Equipment rentals update (quotes, purchases)
- Accessibility updates
- Final drinks order (Fritz Cola count, etc.)

2026-04-01 Governance weekly

- Date: 2026-04-01
- Participants: raito, horrors, Qyriad, kate

Scheduling release retrospective

- In-real-life at LixCon if time permits
- Meeting otherwise
- Owner: Qyriad

Roadmap focus meeting

- Produce a draft based on technical vision document / previous conversations
- Owner: Raito

Status update on LixCon

- Having a list of people who are expert/relevant on certain topics
- Will status update on the next weekly

Schedule a sync Lix/AFNix regarding governance/funding/etc. at LixCon

- Should be discussed with all of the core team, in text
- Owner: Raito

2026-04-18 LixCon meetup

- Date: 2026-04-18
- Participants: raito, rootile, piegames, Jade, lunaphied, qyriad, horrors

Agenda

- Code owners
- +2/-2 review semantics (somewhat adjacent to code owners)
- AI policy

Code owners

- piegames: I want to see some consensus and have written it down; what code ownership means to us? which permissions go with that? how do we distribute +1/+2 rights?
- jade: at the technical level, code ownership just sets recommended reviewers for a tree and also sets who is required to approve a change for a tree
  - +2 does not override codeowners
- piegames: so we can give +2 while keeping code owners coverage or we have +2 rights and code owners who are +2 rights for something?
- jade: those two looks the same to me because if you give a way +2 candy and you also give code owners of particularly directory like candy, this doesn't give particular broad merge capabilities; if you give +2 and code owners to F2 maintainers, they can merge tests
- [description of the system]
- piegames: it looks like the current system is quite complicated, is it worth keeping this level of flexibility?
- raito: i still think that we do not have the current right configuration for code owners, esp. wrt to security
- jade: if people self merge, that's bad and should be audited
- horrors: but the releng process has shown prior self merges
- jade: but this is automatically generated
- horrors: yes but that doesn't exempt it from code review
- [could not catch the rest of the conversation]
- piegames: now i understand how it works but i would like to think about how should it work; i would like a more traditional style
- jade: what we have at work is we have gh owners and it setups so it send pings if certain files changes, these people are not required to approve to merge things; you get someone from some other team and goes stamp a change to your code and that other engineer merges it within 15 minutes, this is obviously a culture failing but i also think this does happen in nixpkgs, code owners are not blocking
- piegames: ok, i think then code owners are doing too many things at the same time; assuming code owners are blocking code owners, does that allow them to merge only code they own? should it?

- rootile: i.e. Should code-owners have scoped +2 permission on their ownership
  - jade note: (this is what happens when someone has "+2" on gerrit in general. we could eliminate +2 rights as a requirement altogether and just use codeowners. then have global approvers list?)
- Qyriad: if the answer to that question is no, either we can give +2 for that purpose or i would suggest having a model where say: if two different code owners of the same tree approve the change,
  - Raito: s/2/N where N is number of codeowners-trees changed
- piegames: What even are "committers" in this model?
- rootile: Keeping ownership dynamically up-to-date will require some maintenance and might lag behind
- Raito: what do you mean by "governance overhead"?
  - My vision is like: you're working on the subsystem a lot, and then you're absent for a while, right?
  - In a reasonable amount of time it'd be good to communicate that you're not going to be active and so you should update your OWNERS file
  - rootile: Who's responsible for poking people to check their OWNERS file?
  - Raito: could make an automated proposal after N inactivity
    - Yes it is bookkeeping but if the bookkeeping provides value then it's worth it
    - jade note: some trees don't move that fast and so idk if it should be per-tree but maybe across the entire repo
  - jade: see also: the psychological fear of, say, adding new owners to a tree
- horrors: any meaningful change should go through a short discussion
- Qyriad: any meaningful modification of the code owners should probably NOT go solely through Gerrit
- Raito: I think the initial vision behind, e.g., handing out these things easily, was to make onboarding people much more accessible (unlike, say, CppNix)
  - However we've been handing out access but that hasn't corresponded to people taking responsibility on these things
    - Qyriad: It doesn't help that people weren't clear on what codeowners meant
  - So now we have accumulated ownership but people may or may not really care
- Raito:
  - Re: docs on these things
- jade: agreement with those present that we can delegate ownership of things like tests to the maintainers of those areas. ie. functional2 owners can add more functional2 owners at a governance level.
- piegames wishlist:
  - List of all code owners -> where? global approvers?
    - OWNERS files in the Lix tree
      - -> jade note: backports, old branches? probably need an all-branches thing for global approvers at least
      - some form of merged summary?
  - List of all people with +2 rights
    - refs/meta/config in the Lix tree (requires some privileges to read)
  - Documentation of +2 and codeowners
  - Documentation of submit requirements

    - Automatically keep that documentation up to date
    - A definition of "committers" for the purpose of our governance document

Action items

```
- [piegames] Write documentation
- [raito] go through the permissions and review them
- [raito] make `refs/meta/config` public
```

+2/-2 review semantics

- raito: +1 is someone else should look at it, +2 is this can go, -1 is meh, needs more discussion, -2 is strong disagreement, might require escalation to core team vote
- Core question: is -2 disagreement with the change in its current form, or with the general idea of the change
- rootile: my interpretation is that -2 means "the concept is fundementally flawed and this should not be worked on again"
- jade: -2 is a blocker, the reviewer needs to re-review and aggree; -2 does not mean bad concept
- piegames: my interpretation is: -1 is a fixable issue, -2 is an unfixable/needs major rework issue
- piegames: From a contributor perspective, -2 might be a critical but easily fixable issue, or it might be a fundamental rejection of the change, and only comments will tell them apart. This is a bad experience
- jade: -2 is just the "request changes" thing on github
- qyriad: i want a distinction between desirable and undesirable
- rootile: Suggestion: Introduce -3 to distinguish those
    - Qyriad: when would we need -3?
        - rootile: say something was ported from CppNix that is just not something we want at all
            - Raito: we need to document that
- piegames: seems like blocking vs unlocking is kind of orthogonal
    - Raito: maybe we should make Unresolved comments blocking?
- Agreement on current state and change being necessary; though not on a solution
- needs re-discussion

Action items

- No general agreement found, will have to rediscuss

☐ Rename abandon to close (maybe fuck with the translation file)

☐ Change the label on -2 and -1 to e.g. "... see comment"

☐ Allow committers to abandon/close CLs by other people (document that doing this is generally rude and should be avoided)

Deferred

- Final semantics of -1/-2

AI Policy

Strongly shortened version:

- Anthropic has been working on things that would probably be good for Lix, but they have been unable to contribute to Lix
  - e.g., a system to compile code in a fully trusted and sandboxed and enclaved way; supply chain security stuff
- Lix is an AFNix project. as-of rn, lix is inheriting the AI policy of AFNix
- if lix wants to deviate, lix needs to discuss it with AFNix
- I'm strongly against allowing any kind of generated code within changes because once we allow some amount of AI-generated code, people will be tempted to find out the limits.
  - People will submit slop if we accept anything at all.
  - The act of dealing with bad changes is a lot of emotional/review capacity work.
- How do people become an expert? Easy contributions *should not be done with AI*, should be done by new contributors.
- If we do the easy work with AI as SMEs, we starve the newcomers of opportunities to do things.
- Having to switch to another infra than AFNix would be a lot of work! Their board is not interested in changing the AI policy.
- All vulns should be hand tested before submission.
- If your thing is good enough, it's not possible to know it's AI generated
- We might want to recommend against trying to use AI to understand the codebase because it might lie.
- **Everyone present agrees with the AFNix policy.**

2026-04-22 Governance weekly

- Date: 2026-04-22
- Participants: ???

Meeting notes / Discussion

Agenda

- Codeownership of f2/testlib (removal of more broad owners)
- Perfectionism vs. incremental changes
- Flakes eval/language restrictions
- Discuss moving the testlib bits outside of the monorepo to address Python dependencies issues

Code ownership of F2/testlib

- rootile: Entire test folder is owned by everyone. It makes sense, everyfew can make tests, delete them, etc. This does includes the testlib directory. IMO, while it belongs to tests, it is its own codebase, it should not be changed or owned by everyone.
- raito: proposal, put OWNERS in testlib with disabled inheritance and put the ownership to F2's codeowners.
- rootile: I'd also like to remove helle from code ownership of F2 because they have not been responsive in the past, if they become active again, I'd add them back. The only codeowner of F2 is me then.
- raito: currently, there's "root code owners" / "default code owners", so you are not the only CO on F2 in practice
- kate: I'd suggest code owners for F2 pieces a couple of core team members e.g. Osiria, just there, so that there's review distribution besides rootile.
- jade: a general comment on things of the shape of testlib, this is a thing that is well trodden, the google stuff is in a testing subdirectory of a source directory, we do not have a good place for python sources, it might be reasonable put testing sources in a src directory as well. Folks mentioned they would like to use F2 for other code outside of Lix, perhaps, we could consider F2 as its own software outside of Lix.
- kate: let's consider using other repositories even if Raito is a monorepo maximalist for things that are not just for Lix.
- qyriad: there might be python packaging concerns
- jade: i am very against moving anything outside of the lix repo for now because you have to deal with lockfiles, nix-eval-jobs was moving inside because it was breaking it all the time; we could use copybara which is how i release snowydeer, to create an export of that part of the lix tree if that's something useful
- moved into its own topic: moving Python outside the monorepo

Action items

- `set noparent; set rootile` and add folks who wants to help
- disboard helle from the F2 code ownership

Perfectionism vs incremental changes

- raito: [insert later the contextualization] (editor's not: lolsob)
- jade: this was written in the context of trying to get the work of unpacking changes in and it did not feel like we were start incrementally merging under a flag
- kate: i feel like there's a bunch of topics that was merged into this loaded topic; what is our obligation and standards of what we were doing, what are expectations for experimental, code that is going to grow vs. code we know that is less likely to have bugs — i don't find the culture of code perfectionism does much to address the number of bugs there, i do think it's a complete separate topic from one of the thing we have issues with: communication, plans on how to get things merged, what is acceptable, general communication about it — there's a definitive issue i have seen a lot, it's that people have a lot of trauma, their own expectations of themselves, things they expect to burn them in a codebase ; someone make an issue, "i don't want to deal with this trauma issue again" and it creates a situation where this is not even about a technical argument anymore because it is going to cause something negative to me happening — in a summary: we have a problem of discussing how to communicate to each other, what to do about the fact that a lot of us are afraid to drive changes in the codebase
- piegames: i'd like to split up the perfectionism from the culture of wanting to do things right, i do think it's a very good thing, that i think lix prides itself with, this can be a thinner line at times; one of the values i wrote back for the first fork and i think lix adopted was that we are willing to go to the extra mile and to do things right and prefer solutions that are long term and sustainable than quick and easy hacks to get things working if that's possible
- jade: i totally agree with kate that the underlying problem is the fear more so than it is anything else, i'd really like to decrease the fear, we can improve our testing story, we can start having fuzzing, we can start having other ways to hunt down bugs before merging, there's verification methodology to decrease the sense of fear, it's a pretty large problem; other methodologies could include having ways to persist warnings, we could surface later
- raito: [oh no no one took note :D]
- kate: the thing i said i realize could be taken the way raito took it :D — there's a difference between doing perfectionism and having a culture of doing things well; perfectionism encapsules to me the standard of "i have a way i think this should be done and i won't tolerate deviating from the way i think this should be done", i think we have accomplished a lot compared to CppNix in how we are taking time to plan and execute them properly, i think this is a culture of doing things right and i think this is different from a culture of perfectionism; can we introduce bugs as part of experimental features as long as they are caught in user testing? these kind of things are implicit in the idea of perfectionism
- horrors: phrasing this entire thing in testing or the fear, it does touch on important points, but not all of them, regarding the macOS stuff that created this topic, there were bugs surfaced and due to communication disasters, it ended up being a minor incident; if we

add features that people really obviously want and there's dangerous, they should come with grave warnings they are dangerous, we only have experimental features with various degrees of communicating dangers from none to "hey this is stable!"; experimental features alone is just not a good framing for the kind of rare-to-common bugs that such features may bring and we do need get better to communicate to our users too

- piegames: about the testing, in spirit of doing things that are long term sustainable, i'm really fan of formal verification, strong type system, things that are resilient/correct/secure by design, fault tolerant by design; as much as I appreciate the efforts like doing the fuzzing stuff, doing these things have an high opportunity cost on NOT writing new code in Rust, NOT writing new systems that can improve upon the old ones that doesn't have failure modes in the first place. About the experimental features, everything is tangled with everything and all failures are very tricky and catastrophic. The focus to me is rather on the modularization and kinda refactoring in Rust and being fearless over there, if something goes wrong, it won't be that bad.
- kate: first, I'd like to address what piegames brought with formal verification, for 10 years, I was the head of being computer division, making a very very high assurance product with a full budget, trying to make it happen and it's — in scale — formal verification is not used because it's in general intractable for something that is a side effect manager. It cannot be the sole method of architecturing code, throwing so much money did not get fractions of the Linux kernel running correctly. I don't think fuzzing is taking time away from formal methods, because formal methods are inherently very hard to apply.
- piegames: Having type systems or encoding grammars are also acts of formal verification in my eyes.
- kate: language constructs as a FV is also insufficient — making a sound language that is capable of having things like that typing depends on data, you start more and more into mathematics and formalism. I don't think getting to Rust is going to change dramatically this. Things that can find bugs in other methods can inform on the overall software engineering. This is kind-of like defense in depth for testing.
- kate: we should talk about ways on reducing conflicts we had, for example, with the archival CL, something that could have solved the whole thing is that if there was a communication: I'm going to do this and I will architect and can we figure out the whole way on how to solve the problem? That didn't seem to happen in a public space.
- jade: I did!
- kate: We did find a better solution by doing investigations on this topic (wrt filesystem slowness on macOS) and we were able to come up with a better plan for virtual filesystem management that both solves the NAR archival by engaging a group of people.
- jade: I completely agree with many of this, it would be incorrect to say this was not discussed before it got written ; the implementation we came up was suggested by horrors. Anyway, we should lean on how other projects improve their quality: we can do somethingASAN, we can fuzz more bugs to appear or to be found. We can decrease the scariness of things. This is what I want to work on in Lix in the next few months.
- kate: As one of the two Darwin maintainers, the filesystem issue was not ever brought up to me or Osiria in a way that this is the plan we are going to do and discussing it. We should agree on a policy to develop expectations on developing of any piece of development.
- raito: re: Kate's thing about formal verification

    - Worked on translating Rust to Lean
    - There's a project called "Signal Shot" (?)
    - Formal verification is still worth investigating
    - Though overall agrees with Kate
    - Thinks the true problem is in the communication
- Raito: Re: Darwin maintainers were not made aware of the unpacking thing design
    - Doesn't think that Kate/Qyriad were not yet considered the true Darwin maintainers
    - Thinks what horrors tried to do was give incremental feedback as they found issues
    - Thinks it's not about fear of the codebase and more about discovering things as we go
    - When you open the Lix devshell it asks you to reach out
        - We don't do that that much these days

Action items

- Raito: thinks that further chat about this can be asynchronous
    - Can create Zulip topics about e.g. fuzzing, reducing conflicts, developing expectations (reinforcing the policy we have on the devshell contribmsg)

Flakes eval/language restrictions

- piegames: Flakes uses a lang subset, which is full of defects and is an enormous layer of violations; this has been shakey grounds because of the layer violations, my current plans with bytecode conflicts with that piece of code. I'd like to make some breaking changes: removing all of the language restrictions in flake.nix. I don't know enough about Flakes to fully grasp the consequences but I do know that it will break CppNix unless they also do something, this will require coordination.
- Qyriad: this is an incompatibility we would be introducing with CppNix users (rather than a Breaking Change™ for Lix users).
- jade: exactly the same thing. Actually, it's less bad than we would see from touching libfetchers, we have some existing incompatibility in libfetchers, this will probably break some Lix users. We have been wanting to do this for a quite while. also, there's holes in the current model which makes it possible to evaluate complicated thunks in CppNix by abusing dynattrs!

(continued in next meeting)

2026-04-29 Governance weekly

- Date: 2026-04-29

Conclusion / Action items

Flakes eval/lang restrictions

- piegames: let's rehash things to see if we agree or not, there has been conversations in the zulip
- piegames: basically, the main unresolved question is: do we feel strongly about those checks in the first place? how do we feel about changing them? how do we feel about removing them? from what i understood: we see those checks as mostly a guideline we can change over time but we want to keep them. one proposal that stuck out how to modify them would be to lift all the parser restrictions and go to only having a "no-function-calls" eval restriction.
  - Raito: full agreement on "from what I understood"
    - Hates those checks
    - However would be a huge pain in the ass to deal with users that are confused by the discrepancy of CppNix vs Lix
      - Because users often send us traces from not realizing it' CppNix instead of Lix
      - Unsolved problem of making users more sure of this
      - Predicts we get a lot of reports of "ahh everything is broken" because they try to eval CppNix-incompatable flakes not realizing they're using CppNix
  - Lunaphied: why do we want to remove the checks anyway?
  - raito: because in the bytecode world, there's no AST anymore.
  - qyriad: it would make error messages way better.
  - Qyriad: Good point on getting reports not realizing they're using CppNix. I would like though to consider how many people are going to make use of non-trivial flake.nix entrypoints and especially users that are not rea
  - Raito: two kinds of users of Nix:
    - Users of things like flake-parts
    - Knowledgeable about Nix; understand the two "languages"
    - They might exploit intentional features of Lix
    - People already do this; e.g. Lix can have a devshell with systemd; CppNix cannot
    - Other kind of users will fall into these errors
  - horrors: the delta between "no-function-calls" and "no weird AST" is thin and CppNix will probably follow this
  - piegames: if we are having an issue having breaking changes, i don't understand the difference between relaxing changes and removing them, it will cause the same kind

of incompatibility either way, it's a question of magnitude in a way, we will always have leaky abstractions; imagine someone doing a toString interpolation, with Lix now, we have `${32}` with `coerce-integers`, those details will always leak. I think the entire concept of restricting evaluation of expressively strong language is fundamentally the wrong approach, unless Flakes manage to go to JSON/TOML or something, those will always happen.
- raito: in full agreement with horrors
  - Also agree with piegames
  - The magnitude will be very low
  - Most users won't run into this
- raito: thinks we can merge Qyriad's original CL (Q: yay!)
- raito: envisions two outcomes:
  - People that want to replace npins/whatever
  - Granted without function calls this is pretty restricted
  - Taking a step back: we want some interface that works quickly, and flakes try to make that happen
    - This is a failure
    - The general idea, of enumerating metadata without long eval times (+ IFD, etc) is an interesting and valuable idea
- Qyriad: regarding the magnitude of concerns, it is low enough we can handle it, if it's not, then, i think we can put efforts to ensure that users can discover if they're not using lix or not; yes this idea of Flakes staying fast to enumerate is a complete failure, when it works, it barely works, the restriction are so much — most reasonable usecases don't even bother and go for legacy packages. The only thing that you can enumerate in Flakes without guaranteeing relatively short eval are the inputs and inputs alone. Everything else can take as long as it wants. I think if we want specifically make something that is quick to eval, we want to decide what are actual things we want to make it quick eval and put it in a TOML format.
- Raito: behind the current idea; just wanted to add some nuance
- piegames: lol i dont care

Summary / Action items

- Remove the AST-shape check
- forbid function calls (maxCallDepth=1)
- typecheck the evaluated result instead of checking the AST
- don't do anything about dyn-attrs (leave them as-is)
- notify the CppNix team

Discuss moving the testlib bits outside of the monorepo to address Python dependencies issues

- rootile: folks want to use F2 outside of the Lix codebase, what is the best way to facilitate that? snix used F2 as well; in my opinion, we might want to move testlib out of the F2 folder and make it a Python package which then you can use inside of test/f2 and even

publish it to PyPI so that people outside of Lix can use it too
- raito: someone could try to do an experiment of using f2 testlib outside the monorepo and show what it looks like and how it could be improved
  - For users of f2 you'd probably want both the Nix interpreter and the Python dependencies
  - Probably *wouldn't* publish it to Pypi
    - Since it depends on the Nix interpreter. Kinda weird
    - Then we have to maintain a publishing pipeline and stuff
  - If it's just a Nixlang entry point they can just put it in their build inputs and go
  - Wonders if the rest could be moved to async discussion
  - rootile: thinks that's fine

Trailing slashes in fetchurl (re: snix)

Context: https://git.lix.systems/lix-project/lix/issues/1185

- Difference between CppNix and Lix
- raito: What should the behavior be?
- horrors: we should change it; the current behavior makes no sense
- raito: :+1:

Quick check-in

- Raito: limited availablity
  - Flakes extraction is affected
    - Either need more help or move to next release cycle
  - Can still tackle merge queue
  - Also planning a LixCon retrospective