Attention, Nix and Tacos Is All You Need

TacoSprint 2026 — La Saladita, Guerrero, Mexico

A week of Nix hacking, collaboration, and knowledge sharing.

Domen Kožar*
domen@cachix.org

Alan Urmancheev
alan.urman@gmail.com

Farid Zakaria*
farid.m.zakaria@gmail.com

Victor Borja
vborja@apache.org

Marijan Petričević
marijan.petricevic94@gmail.com

Geoffrey Huntley
ghuntley@ghuntley.com

Alex Decious
alex@decio.us

Jared Siegel
hi@jared.ltd

Jonathan Ringer
jonringer117@gmail.com

Figure 1. Original TacoSprint Nixers, 2026.

Abstract

The dominant approach to reproducible software distribution relies on a global store, root privileges, and tightly coupled package closures. We present TacoSprint, a week-long collaborative effort demonstrating that significant advances in the Nix ecosystem can be achieved through a simpler recipe: co-location on the Pacific coast, a steady supply of tacos, and a shared package manager. Over the course of one week, a small group of contributors advanced work spanning dynamic linking, package relocatability, peer-to-peer remote builds, faster module systems, cross-distribution packaging, and LLM-assisted developer tooling, showing that tacos and Nix, applied together, are sufficient. Our results establish new state-of-the-art on the benchmark of getting things done while traveling to remote locations. Ultimately, we deposit the conjecture that one may mix business and pleasure with astonishing results. We further observe that at least one beer was spilled in the process.

ACM Reference Format:
Domen Kožar, Alan Urmancheev, Farid Zakaria, Victor Borja, Marijan Petričević, Geoffrey Huntley, Alex Decious, Jared Siegel, and Jonathan Ringer. 2026. Attention, Nix and Tacos Is All You Need: TacoSprint 2026 — La Saladita, Guerrero, Mexico *A week of Nix hacking, collaboration, and knowledge sharing.*. In *Proceedings of the 2026 TacoSprint Workspace and Surf Retreat*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/tacos.sprint2026

*Domen Kožar and Farid Zakaria served as event organizers.

TacoSprint '26, La Saladita, Guerrero, Mexico

ACM ISBN 978-1-NIX-TACO-2026
https://doi.org/10.1145/tacos.sprint2026

1 Introduction

Recurrent collaboration models have long dominated the open-source landscape, processing contributions sequentially through asynchronous review queues. While effective,

these models suffer from high latency and an inability to share context in parallel. We dispense with asynchrony entirely and propose a mechanism based solely on physical co-location and shared meals known as a *Sprint*.

Taco*Sprint* 2026 [1] gathered 9 Nix developers of all skill levels in two beachfront villas at La Saladita, Guerrero, on the Pacific coast of Mexico, from 20 to 26 June 2026. The sprint brings together a diverse set of contributors, each attending to a different sub-problem of the broader reproducibility landscape, yet all attending to one another. In the following sections we describe the principal lines of work pursued during the sprint. This report summarizes progress through Thursday, 25 June 2026, the penultimate day.

2 Background

Prior work in reproducible builds requires either elevated privileges, a fixed store prefix, or both. Several efforts at the sprint independently attacked these assumptions. We group the contributions into five thematic areas: (i) dynamic linking and startup, (ii) relocatability, (iii) distributed and remote builds, (iv) module systems, and (v) packaging and tooling.

3 Model Architecture

3.1 Dynamic Linking and Startup

Domen Kožar advanced the dynamic linker startup path, landing a `patchelf` change [12] and subsequently fixing two bugs blocking the `patchelf` bump in `nixpkgs` [13]. By the end of the sprint the work was ready, accompanied by a blog post on making `devenv`, and the whole of `nixpkgs`, start fast [11]. In a parallel contribution, Domen added Chromecast and AirPlay support to `danklinux` [10], which, after an initial setback involving a spilled beer, works.

Farid Zakaria collaborated with Domen on the `patchelf` approach due to his experience from past research [29]. Farid investigated the orthogonal technique of hijacking ELF entry points which had similar usefuleness for relocatability, documented in his *wrap-buddy* compiler design [27]. This technique leverages post-link analysis to safely wedge alternative execution vectors into arbitrary binaries. To solidify these user-space bounds, he posted a comprehensive kernel patch series to the Linux memory management mailing list entitled *fs: support* `$ORIGIN` *in ELF interpreter paths* [28], complementing his broader research on mapping out deployment and dependency complexities in highly parallel execution environments [29].

3.2 Relocatability

Alan Urmancheev developed *relocatable-nix* [22], an overlay adding a `stdenv` hook that wraps executables and launches them via a custom loader, explicitly resolving relative references for both dynamic linkers and shebanged scripts. The mechanism requires neither root, a custom Nix, nor a custom `nixpkgs`, and operates either globally or opt-in per package. It works on Linux, with macOS support pending testing.

A companion project, *relocatable-shebangs* [23], takes a kernel-side approach: a Linux kernel patch expands a leading `${ORIGIN}/` in a shebang to the directory containing the script (symlinks resolved), so packages can reference their interpreter relative to their own store location; it ships a modified `patchShebangs` and a QEMU demo relocating a package to `/store` rather than `/nix/store`. This dovetails with Farid's `$ORIGIN` ELF interpreter patch [28]. Alan additionally submitted a kernel documentation patch correcting an inconsistent comment regarding recursive shebangs [21], which was accepted into the kernel by the end of the sprint.

3.3 Distributed and Remote Builds

Alex Decious built *drv-thru* [9], a system for remote Nix builds over the Iroh p2p library. The design supports just-in-time store path signing with a binary cache served over Iroh, a daemon with trusted users or one-time redeemable tickets, a local mirror, and a NixOS module providing both client and server configuration, including a helper for non-trusted users. Over the sprint Alex refactored the codebase and file structure, added support for persistent builds in the NixOS modules with client-side status reporting, and recorded a demo for the project's README.

3.4 Module Systems

Victor Borja developed the Zen module system [5]. A fair benchmark against `nixpkgs.lib.evalModules` showed Zen to be **up to 20 times faster**, a result cross-validated by gathering performance numbers from four other attendees' machines. Victor opened a PR to *nix-effects*' substrate [4] after identifying a stack-overflow limitation, produced runnable demos (`just demo`) showcasing actor-based modules, streams, and an MLTT substrate, and posted an overview on Reddit [6].

He also reviewed and shipped a v0.18.0 release of the Den framework [3]. Beyond the sprint, Victor engaged Mika Bohinen and Jason Bowman on applying formal proof checking to Nix configurations. Mika is the the author of *nix-effects*' MLTT type system [2] and Jason is the author of the `gen` libraries [7] and a core contributor to Den on the upcoming `hola` modules [8], which aim for full `nixpkgs-modules` parity using algebraic graph libraries.

3.5 Packaging and Tooling

Marijan Petričević reduced the OCaml runtime closure size. After initial attempts to split the compiler from stdlib proved both difficult and, ultimately, unnecessary, the actual solution turned out to be much simpler: splitting the OCaml metadata (`.cmt`/`.cmti` files), the files referencing the compiler package, into a separate `dev` output [14]. The effect on closure size is dramatic: `nixtamal` shrinks from 874.9 MiB to 483.2 MiB, a 1.8x savings, while library packages such as `eio` (400.6 →

41.0 MiB) and `lwt` (389.9 → 42.3 MiB) drop by roughly an order of magnitude.

Jonathan Ringer made substantial progress on Ekapkgs: fixing pypy, libsecret, and flex v2.5 [17], porting killall and hostname for Linux [16], and landing ∼ 30 package fixes for corepkgs [15], with python-pkgs and node-pkgs in progress. On eka-ci, Jonathan normalized actor service usage [19] and added support for `nix derivation show` v4 (nix 2.28+) [18].

Jared Siegel opened a series of PRs against *mcp-nixos* [20], pursued simple version bumps in `nixpkgs` toward a first NixOS PR, practiced with Ralph loops, and authored Claude skills for SLOs, alerts, and log parsing. Later in the week he built *gregg*, a Claude skill derived from Brendan Gregg's writings bundling scripts and a Go binary for agent-driven performance analysis, and ran Ralph loops against an MCP server exposing an internal metrics system, building in rate-limiting politeness. He also discussed *drv-thru* and McNixolds with Alex.

Farid Zakaria additionally launched *guixpkgs* [25], a monumental infrastructure project designed to auto-generate and expose every package within the GNU Guix package map natively as an independent, fully isolated Nix flake [26]. The deployment serves as a massive interoperability bridge across standard declarative distributions; however, evaluation metrics highlighted substantial wait boundaries, with local compilation clusters spending extensive compute resources waiting synchronously for the fundamental `mesaboot` substrate to complete its bootstrap phase. He also submitted an `AGENTS.md` PR to `nixpkgs` [24], began investigating Nix adoption at Meta, and set up eternal-terminal with Alex.

4 Training Details

All work was conducted at La Saladita, Guerrero, from 20–26 June 2026, where roughly 9 Nix developers shared two beachfront villas [1]. The event was organized by Domen Kožar and Farid Zakaria and sponsored by Meta, Cachix, and Numtide. Infrastructure consisted of shared workspaces, eternal-terminal sessions, and tacos. Several contributors balanced sprint work with day jobs. One participant ported their NixOS configuration from river+waybar to niri+noctalia mid-sprint.

Logistical parameters experienced acute real-world jitter. Most notably, a cascading scheduling dependency error occurred when attendee Alex Decious completely missed his connecting flight infrastructure delaying his physical arrival at the workspace grid by exactly three days. Fortunately, he remained resilient to partitioning and arrived in good-spirits and enjoyed the remainder of the sprint.

Between sessions, contributors surfed La Saladita's famous left point break. Regularization was provided by daily standups. LLM-based agents featured prominently as a productivity multiplier throughout the sprint whether from Ralph loops driving MCP server development to Claude skills for performance analysis and operations, suggesting that, much like tacos, they are becoming a staple of the modern Nix developer's workflow. We were fortunate to have **Geoff Huntley**, the originator of Ralph Loops, as our AI spiritual guide throughout this adventure.

4.1 Gastronomy and Culinary Infrastructure

A foundational component of the TacoSprint architecture was the culinary cooking, which guaranteed high-throughput developer energy distribution. This operations tier was entirely managed and sustained by **Gladys**, our local culinary engineer.

Gladys designed and deployed a robust menu three times a day, consisting primarily of fresh handmade corn tortillas, beans, cheese and a variety of proteins. Her food was outstanding and we were truly fortunate to have her fueling us as it mitigated standard hackathon fatigue.

5 Results

The overall summary of the areas of focus and current deliverables is outlined in Table 1.

Table 1. TacoSprint 2026 Key Projects and Status

Theme / Project	Contributors	Status / Result
Dynamic Linking & Startup		
Dynamic linker startup	Domen	Blog published; ready
Chromecast / AirPlay	Domen	WIP; working on `danklinux`
Relocatability		
Relocatable Nix & Shebangs	Alan, Farid	patches submitted to Linux
Distributed Systems		
`drv-thru` remote builds	Alex	Repository shared; demo recorded
Module Systems		
Zen module system	Victor	PR open; up to 20× faster
Packaging & Tooling		
OCaml closure size	Marijan	PR merged; ∼ 45% reduction
Corepkgs / `eka-ci`	Jonathan	Multiple PRs merged/open
`mcp-nixos` & agent tools	Jared	PRs open; skills demoed
`guixpkgs` infrastructure	Farid	Repository shared; blog post written

Figure 2. A compilation of collaborative hacking, surfboard setups, and compiler hacking at La Saladita.

6 Conclusion

We have shown that a week of co-located hacking, fueled by tacos and unified by Nix, is sufficient to make meaningful progress across dynamic linking, relocatability, remote builds, module systems, cross-distribution packaging, and LLM-assisted tooling. The approach generalizes to other coastlines and other foods, though we leave such ablations to future work. We conjecture that attention, Nix and tacos is, indeed, all you need.

Acknowledgements

We thank all attendees of TacoSprint 2026: Domen Kožar, Marijan Petričević, Alex Decious, Jared Siegel, Victor Borja, Geoffrey Huntley, Alan Urmancheev, Farid Zakaria, and Jonathan Ringer, as well as Mika Bohinen, Jason Bowman, and Jörg Thalheim (Mic92) for fruitful discussions, and Meta, Cachix, and Numtide for sponsorship. We are also no strangers to one another, and we acknowledge the tacos.

Figure 3. Gladys managing the BBQ to cook for us a ton of steaks.

References

[1] 2026. Nix TacoSprint 2026. https://tacosprint.org/.

[2] Mika Bohinen. 2026. nix-effects. https://github.com/kleisli-io/nix-effects.

[3] Victor Borja. 2026. Den framework, v0.18.0. https://github.com/denful/den.

[4] Victor Borja. 2026. PR to nix-effects' substrate, #27. https://github.com/kleisli-io/nix-effects/pull/27.

[5] Victor Borja. 2026. Zen: an experimental actor-based module system. https://github.com/denful/zen.

[6] Victor Borja. 2026. Zen overview on r/NixOS. https://www.reddit.com/r/NixOS/comments/1uevpug/zen_an_experimental_actorbased_module_system_for/.

[7] Jason Bowman. 2026. gen libraries. https://github.com/sini/gen.

[8] Jason Bowman. 2026. hola-architecture. https://github.com/sini/hola-architecture.

[9] Alex Decious. 2026. drv-thru: remote Nix builds over Iroh. https://github.com/adeci/drv-thru.

[10] Domen Kožar. 2026. Chromecast/AirPlay support, DankMaterialShell #2697. https://github.com/AvengeMedia/DankMaterialShell/pull/2697.

[11] Domen Kožar. 2026. Making devenv start fast — and the whole of nixpkgs with it (draft). https://devenv.pages.dev/blog/2026/06/26/making-devenv-start-fast-and-the-whole-nixpkgs-with-it/.

[12] Domen Kožar. 2026. patchelf #647. https://github.com/NixOS/patchelf/pull/647.

[13] Domen Kožar. 2026. patchelf #652. https://github.com/NixOS/patchelf/pull/652.

[14] Marijan Petričević. 2026. nixpkgs, branch improve-ocaml. https://github.com/marijanp/nixpkgs/tree/improve-ocaml.

[15] Jonathan Ringer. 2026. corepkgs #107 (~30 package fixes). https://github.com/ekala-project/corepkgs/pull/107.

[16] Jonathan Ringer. 2026. corepkgs #108 (killall, hostname). https://github.com/ekala-project/corepkgs/pull/108.

[17] Jonathan Ringer. 2026. corepkgs #109 (pypy, libsecret, flex). https://github.com/ekala-project/corepkgs/pull/109.

[18] Jonathan Ringer. 2026. eka-ci #204 (nix derivation show v4). https://github.com/ekala-project/eka-ci/pull/204.

[19] Jonathan Ringer. 2026. eka-ci #205 (actor service usage). https://github.com/ekala-project/eka-ci/pull/205.

[20] Jared Siegel et al. 2026. mcp-nixos #173–177. https://github.com/utensils/mcp-nixos/pull/177.

[21] Alan Urmancheev. 2026. Kernel patch: recursive-shebang comment fix. https://lore.kernel.org/all/20260623052322.74711-1-alan.urman@gmail.com/T/#u.

[22] Alan Urmancheev. 2026. relocatable-nix. https://github.com/alurm/relocatable-nix.

[23] Alan Urmancheev. 2026. relocatable-shebangs. https://github.com/alurm/relocatable-shebangs.

[24] Farid Zakaria. 2026. AGENTS.md PR to nixpkgs, #534657. https://github.com/NixOS/nixpkgs/pull/534657.

[25] Farid Zakaria. 2026. guixpkgs. https://github.com/fzakaria/guixpkgs.

[26] Farid Zakaria. 2026. guixpkgs: every Guix package as a Nix flake. https://fzakaria.com/2026/06/25/guixpkgs-every-guix-package-as-a-nix-flake.

[27] Farid Zakaria. 2026. Hijacking ELF entry points for NixOS compatibility, or wtf is wrap-buddy. https://fzakaria.com/2026/06/22/hijacking-elf-entry-points-for-nixos-compatibility-or-wtf-is-wrap-buddy.

[28] Farid Zakaria. 2026. [PATCH 0/2] fs: support $ORIGIN in ELF interpreter paths. https://lore.kernel.org/linux-mm/20260622043934.179879-1-farid.m.zakaria@gmail.com/T/#t.

[29] Farid Zakaria, Thomas R. W. Scogland, Todd Gamblin, and Carlos Maltzahn. 2022. Mapping Out the HPC Dependency Chaos. arXiv:2211.05118 [cs.SE] https://arxiv.org/abs/2211.05118

Organized by Nixers, Powered by Tacos.