Why Choose Nix for develoment?

Why opt for Nix when developping a software project instead of language-specific alternatives (such as Stack or GHCup for Haskell)?

- **Instantaneous Onboarding**: Typical project READMEs detail environment setup instructions that often fail to work uniformly across different developers' machines, taking hours or even days to configure. Nix offers an instant and reproducible setup, allowing any newcomer to get their development environment ready swiftly with one command.
- **Boosted Productivity**: Developers can dedicate more time to writing software, as Nix ensures a fully functional development environment through `nix develop`.
- **Multi-Platform Support**: The same configuration reliably works across macOS, Linux, and WSL.

macOS support

While macOS doesn't enjoy first-class support in nixpkgs yet, improvements are underway.

Links to this page

Replacing docker-compose with Nix for development

> Ever since I first started using Nix for development, I have enjoyed the simplicity of setup: `nix develop`, make the code change and see it work. That's all well and good, but when your project keeps growing, you need to depend on external services like databases, message brokers, etc. And then, a quick search will tell you that docker is the way to go. You include it, add one more step in the setup guide, increasing the barrier to entry for new contributors. Not to mention, eating up all the system resources* on my not so powerful, company-provided MacBook.

Nixifying a Haskell project using nixpkgs

> To appreciate why Nix is a great choice for Haskell development, see Why Choose Nix for develoment?

> This tutorial pratically demonstrated why Nix is a great choice for Haskell development:

Nix for Development

> Why Choose Nix for develoment?