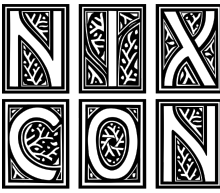


# This year in Nix documentation

Valentin Gagarin — NixCon 2023



Vision



- Contributor experience
- Onboarding and learning
- Reference documentation

# Past contributors



**Domen Kožar**  
domenkozar



**olaf**  
olafklingt



**Solène Rapenne**  
rapenne-s



**renoire**



**Luc Perkins**  
lucperkins



**milibopp**



**erooke**



**j-k**  
06kellyjac



**Jeremiah S**  
JeremiahSecrist



**JacklineYim**



**Rémi NICOLE**  
minijackson



**Philipp**  
pstn

# The Nix Documentation Team



Valentin Gagarin  
fricklerhandwerk



Silvan Mosberger  
infinisil



Yuki Langley  
yukiisbored



Alejandro Sánchez Medina  
alejandrosame



Zach Mitchell, PhD  
zmitchell



Robert Hensing  
roberth



Henrik  
henrik-ch



Jill Thornhill



Alexander Groleau  
proofconstruction



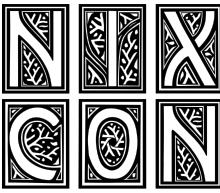
asymmetric  
asymmetric



Shahar "Dawn" Or  
mightyiam



Brian McGee  
brianmcgee



# Contributor guides

# expand contribution guidelines #653



Merged

zmitchell merged 4 commits into [NixOS:master](#) from [fricklerhandwerk:contributing](#) on Jul 27

## Concepts

Flakes

## Contributing

How to contribute

How to get help

## Contributing Documentation



Documentation framework

Style guide

How to write a tutorial

## Acknowledgements

Sponsors

## Contributing Documentation

This is an overview of documentation resources for Nix, Nixpkgs, and NixOS, with suggestions how you can help to improve them. Documentation contributions should follow the [style guide](#).

Feel free to get in touch with the [Nix documentation team](#) if you want to help out.

### Attention

If you cannot contribute time, consider [donating to the NixOS Foundation's documentation project on Open Collective](#) to fund ongoing maintenance and development of reference documentation and learning materials.

## Reference manuals

The manuals for [Nix \(source\)](#), [Nixpkgs \(source\)](#), and [NixOS \(source\)](#) are purely reference documentation, specifying interfaces and behavior.

# Add CONTRIBUTING.md #7968

 Merged

thufschmitt merged 1 commit into `NixOS:master` from `bobvanderlinden:pr-contributing`  on Mar 20



## Contributing to Nix

---

Welcome and thank you for your interest in contributing to Nix! We appreciate your support.

Reading and following these guidelines will help us make the contribution process easy and effective for everyone involved.

### Report a bug

---

1. Check on the [GitHub issue tracker](#) if your bug was already reported.
2. If you were not able to find the bug or feature [open a new issue](#)
3. The issue templates will guide you in specifying your issue. The more complete the information you provide, the more likely it can be found by others and the more useful it is in the future. Make sure reported bugs can be reproduced easily.
4. Once submitted, do not expect issues to be picked up or solved right away. The only way to ensure this, is to [work on the issue yourself](#).

### Report a security vulnerability

---

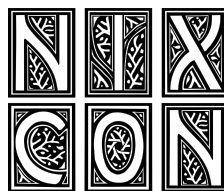
Check out the [security policy](#).

### Making changes to Nix

---



# Improve hacking.md and add clangd+bear to devshell #7433



Merged

thufschmitt merged 2 commits into [NixOS:master](#) from [yorickvp:improv-onboarding](#) on Feb 20

## Building Nix

To build all dependencies and start a shell in which all environment variables are set up so that those dependencies can be found:

```
$ nix-shell
```

To get a shell with one of the other [supported compilation environments](#):

```
$ nix-shell --attr devShells.x86_64-linux.native-clang11StdenvPackages
```

### Note

You can use `native-ccacheStdenvPackages` to drastically improve rebuild time. By default, `ccache` keeps artifacts in `~/.cache/ccache/`.

To build Nix itself in this shell:

```
[nix-shell]$ ./bootstrap.sh
```

# Clean up contributing documentation #245243



infinisil merged 42 commits into `NixOS:master` from `tweag:contributing-combining` 3 weeks ago



## Contributing to Nixpkgs packages

This document is for people wanting to contribute specifically to the package collection in Nixpkgs. See the [CONTRIBUTING.md](#) document for more general information.

### Overview

- `top-level` : Entrypoints, package set aggregations
  - `impure.nix` , `default.nix` , `config.nix` : Definitions for the evaluation entry point of `import <nixpkgs>`
  - `stage.nix` , `all-packages.nix` , `by-name-overlay.nix` , `splice.nix` : Definitions for the top-level attribute set made available through `import <nixpkgs> {...}`
  - `*-packages.nix` , `linux-kernels.nix` , `unixtools.nix` : Aggregations of nested package sets defined in `development`
  - `aliases.nix` , `python-aliases.nix` : Aliases for package definitions that have been renamed or removed
  - `release*.nix` , `make-tarball.nix` , `packages-config.nix` , `metrics.nix` , `nixpkgs-basic-release-checks.nix` : Entry-points and utilities used by Hydra for continuous integration
- `development`
  - `*-modules` , `*-packages` , `*-pkgs` : Package definitions for nested package sets
  - All other directories loosely categorise top-level package definitions, see [category hierarchy](#)
- `build-support` : Builders
  - `fetch*` : Fetchers

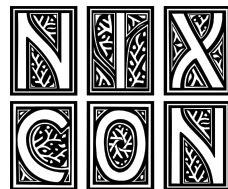


nix.dev is official

# Improve landing page #584

Merged

fricklerhandwerk merged 3 commits into `NixOS:master` from `yukiisbored:landing-page` on Jun 5



Official documentation for getting things done with Nix.



Search...



## Tutorials

Install Nix

First Steps

Ad hoc shell environments

Reproducible interpreted scripts

Nix language basics

Towards reproducibility: pinning  
Nixpkgs

Declarative and reproducible  
developer environments

## Welcome to nix.dev

nix.dev is the home of official documentation for the Nix ecosystem, it contains:

### Tutorials

Series of lessons to get started

### Recipes

Guides to getting things done

### Reference

Collections of detailed technical descriptions

### Concepts

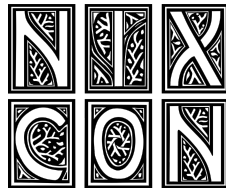
Explanations of history and ideas in the Nix  
ecosystem

If you're new to the Nix ecosystem, begin your journey with [First Steps](#)!

## What can you do with Nix?

The following illustrate of what can be achieved with the Nix ecosystem:

# Improve readability via styling improvements #405



Merged

lucperkins merged 14 commits into `NixOS:master` from `yukiisbored:styling` on Jan 5

Recursive attribute set `rec { ... }`

You will sometimes see attribute sets declared with `rec` prepended. This allows access to attributes from within the set.

Example:

```
rec {  
  one = 1;  
  two = one + 1;  
  three = two + 1;  
}
```

Expression

```
{ one = 1; three = 3; two = 2; }
```

Value

## Note

Elements in an attribute set can be declared in any order, and are ordered on evaluation.

# add tutorial for nix-shell in shebang #325



Merged

fricklerhandwerk merged 16 commits into `NixOS:master` from `rapenne-s:nix_shell_shebang` on Feb 14



We will use the shebang line `#!/usr/bin/env nix-shell`.

`env` is a program available on most modern Unix-like operating systems at the file system path `/usr/bin/env`. It takes a command name as argument and will run the first executable by that name it finds in the directories listed in the environment variable `$PATH`.

We use `nix-shell` as a shebang interpreter. It takes the following parameters relevant for our use case:

- `-i` tells which program to use for interpreting the rest of the file
- `--pure` excludes most environment variables when the script is run
- `-p` lists packages that should be present in the interpreter's environment
- `-I` explicitly sets the search path for packages

More details on the options can be found in the `nix-shell` reference documentation.

Create a file named `nixpkgs-releases.sh` with the following content:

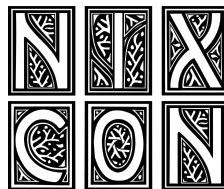
```
#!/usr/bin/env nix-shell
#! nix-shell -i bash --pure
#! nix-shell -p bash cacert curl jq python3Packages.xmljson
#! nix-shell -I nixpkgs=https://github.com/NixOS/nixpkgs/archive/2a601aafdc5605a5133a2ca506a34e
```

# tutorial: nixos configurations on vm #334



Merged

fricklerhandwerk merged 28 commits into [NixOS:master](#) from [olafklingt:nixos-configuration-on-vm](#) on Mar 16



## NixOS virtual machines

One of the most important features of NixOS is the ability to configure the entire system declaratively, including packages to be installed, services to be run, as well as other settings and options.

NixOS configurations can be used to test and use NixOS using a virtual machine, independent of an installation on a “bare metal” computer.

### Important

A NixOS configuration is a Nix language function following the [NixOS module](#) convention.

## What will you learn?

This tutorial serves as an introduction creating NixOS virtual machines. Virtual machines are a practical tool for debugging NixOS configurations.

# add a tutorial on packaging existing software #650



proofconstruction merged 74 commits into [NixOS:master](#) from [proofconstruction:packaging-existing-software](#)  last month



## A Simple Project

To start, consider this skeleton derivation:

```
1 { stdenv }:  
2  
3 stdenv.mkDerivation { };
```

This is a function which takes an attribute set containing `stdenv`, and produces a derivation (which currently does nothing).

As you progress through this tutorial, you will update this several times, adding more details while following the general pattern.

## Hello, World!

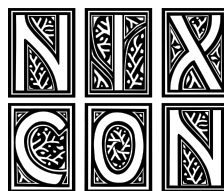
GNU Hello is an implementation of the “hello world” program, with source code accessible [from the GNU Project’s FTP server](#).





# Nix Reference Manual

# separate man pages for `nix-store` and `nix-env` subcommands #7518



Merged

fricklerhandwerk merged 3 commits into [NixOS:master](#) from [fricklerhandwerk:doc-commands](#) on Mar 30

## 7.3.2. nix-shell

### 7.3.3. nix-store

7.3.3.1. nix-store --add-fixed

7.3.3.2. nix-store --add

7.3.3.3. nix-store --delete

7.3.3.4. nix-store --dump-db

7.3.3.5. nix-store --dump

7.3.3.6. nix-store --export

7.3.3.7. nix-store --gc

7.3.3.8. nix-store --generate-binary-cache-key

7.3.3.9. nix-store --import

7.3.3.10. nix-store --load-db

7.3.3.11. nix-store --optimise

7.3.3.12. nix-store --print-env

7.3.3.13. nix-store --query

7.3.3.14. nix-store --read-log

☰ ✎ 🔍 Nix Reference Manual 🖨️ ↺ 📝

## Synopsis

`nix-store` *operation* [*options...*] [*arguments...*] [ `--option` *name value* ]  
[ `--add-root` *path* ]

## Description

The command `nix-store` performs primitive operations on the Nix store. You generally do not need to run this command manually.

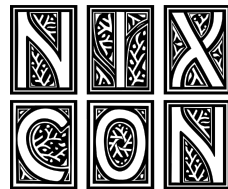
`nix-store` takes exactly one *operation* flag which indicates the subcommand to be performed. The following operations are available:

- `--realise`
- `--serve`
- `--gc`
- `--delete`
- `--query`

# Document user files of nix #8141



Ericson2314 merged 11 commits into `NixOS:master` from `tweag:user-files-doc` on May 15



7.5.85. nix upgrade-nix

7.5.86. nix why-depends

## 7.6. Files

7.6.1. nix.conf

7.6.2. Profiles

7.6.2.1. manifest.nix

7.6.2.2. manifest.json

7.6.3. Channels

7.6.4. Default Nix expression

## 8. Architecture and Design

8.1. File System Object

## 9. Protocols

9.1. Serving Tarball Flakes

## 10. Glossary

## 11. Contributing

11.1. Hacking

11.2. Testing



## Nix Reference Manual



## Profiles

A directory that contains links to profiles managed by `nix-env` and `nix profile`:

- `$XDG_STATE_HOME/nix/profiles` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root` if the user is `root`

A profile is a directory of symlinks to files in the Nix store.

## Filesystem layout

Profiles are versioned as follows. When using a profile named *path*, *path* is a symlink to *path - N-link*, where *N* is the version of the profile. In turn, *path - N-link* is a symlink to a path in the Nix store. For example:

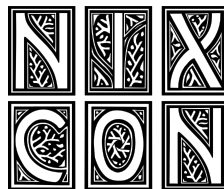
```
$ ls -l ~alice/.local/state/nix/profiles/profile*
lrwxrwxrwx 1 alice users 14 Nov 25 14:35 /home/alice/.local/state/nix/profiles/pro
lrwxrwxrwx 1 alice users 51 Oct 28 16:18 /home/alice/.local/state/nix/profiles/pro
```

# add information on the system type string #8315



Merged

roberth merged 11 commits into `NixOS:master` from `fricklerhandwerk:doc-system` on Jul 19



## Nix Reference Manual

- `system`

The system type of the current Nix installation. Nix will only build a given `derivation` locally when its `system` attribute equals any of the values specified here or in `extra-platforms`.

The default value is set when Nix itself is compiled for the system it will run on. The following system types are widely used, as [Nix is actively supported on these platforms](#):

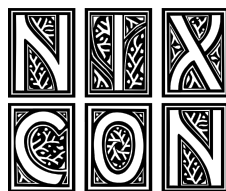
- `x86_64-linux`
- `x86_64-darwin`
- `i686-linux`
- `aarch64-linux`
- `aarch64-darwin`
- `armv6l-linux`
- `armv7l-linux`

In general, you do not have to modify this setting. While you can force Nix to run a Darwin-specific `builder` executable on a Linux machine, the result would obviously be wrong.

This value is available in the Nix language as `builtins.currentSystem`.

**Default:** `x86_64-linux`

# Documentation: list experimental features in manual #7798



Merged

edolstra merged 14 commits into [NixOS:master](#) from [peeley:list-experimental-features](#) on Apr 11

Nix Reference Manual

## Currently available experimental features

### auto-allocate-uids

Allows Nix to automatically pick UIDs for builds, rather than creating `nixbld*` user accounts. See the `auto-allocate-uids` setting for details.

### ca-derivations

Allow derivations to be content-addressed in order to prevent rebuilds when changes to the derivation do not result in changes to the derivation's output. See `__contentAddressed` for details.

### cgroups

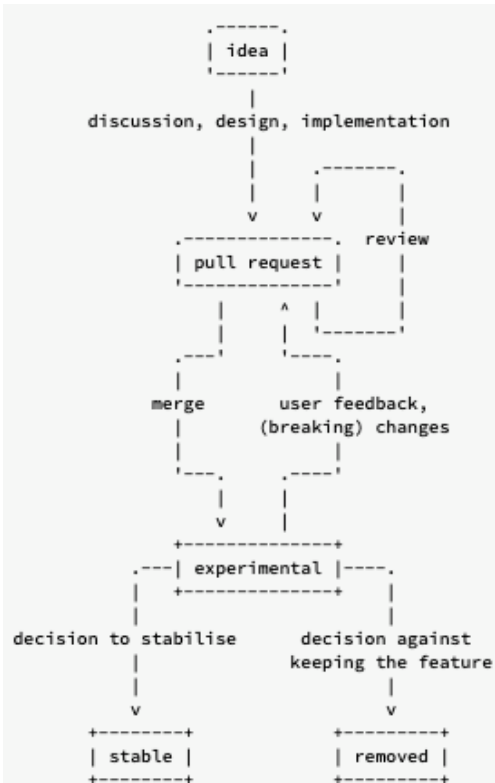
Allows Nix to execute builds inside cgroups. See the `use-cgroups` setting for details.

### daemon-trust-override

Allow forcing trusting or not trusting clients with `nix-daemon`. This is useful for testing, but possibly also useful for various experiments with `nix-daemon --stdio` networking.

### dynamic-derivations

Allow the use of a few things related to dynamic derivations:

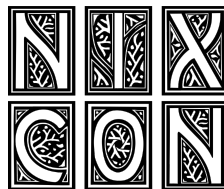


# Generate API docs with Doxygen #7896



Merged

Ericson2314 merged 1 commit into [NixOS:master](#) from [obsidiansystems:doxygen](#) on Mar 10



## Public Member Functions

virtual void	<b>show</b>	(const <b>SymbolTable</b> &symbols, std::ostream &str) const
virtual void	<b>bindVars</b>	( <b>EvalState</b> &es, const std::shared_ptr< const <b>StaticEnv</b> > &env)
virtual void	<b>eval</b>	( <b>EvalState</b> &state, <b>Env</b> &env, <b>Value</b> &v)
virtual <b>Value</b> *	<b>maybeThunk</b>	( <b>EvalState</b> &state, <b>Env</b> &env)
virtual void	<b>setName</b>	( <b>Symbol</b> name)
virtual <b>PosIdx</b>	<b>getPos</b>	() const

The documentation for this struct was generated from the following files:



- [src/libexpr/nixexpr.hh](#)
- [src/libexpr/eval.cc](#)



# Nixpkgs Manual

# doc: dedocbookify nixpkgs manual #239636



 Merged pennae merged 15 commits into [NixOS:master](#) from [pennae:nixpkgs-manual-nrd](#)  on Jul 3



▼ 11  doc/.gitignore 

☐ Viewed



[Load diff](#)

This file was deleted.



▼ 114  doc/Makefile 

☐ Viewed



[Load diff](#)

This file was deleted.

▼ 23  doc/build-aux/pandoc-filters/docbook-reader/citerefentry-to-rst-role.lua 

☐ Viewed

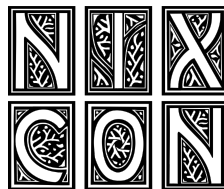


[Load diff](#)

This file was deleted.



# Tracking - Documentation - Python section in nixpkgs manual. #240118

[Open](#)[21 tasks](#)

alejandrosame opened this issue on Jun 27 · 0 comments

## Python section issues

Reported problems with the `nixpkgs manual` Python section.

- ☐ [Doc: example dont work in 15.17.1.4. Including a derivation using callPackage](#) #72167
- ☐ [Documentation bug or incomplete check for python shellHook development mode](#) #66460

## Missing Python reference documentation

Python related Nix APIs, use cases, etc., possibly lacking reference documentation. These are to be checked, but can be useful for other types of documentation.

- ☐ [python-language-server packaging issues](#) #229337
- ☐ [Add documentation for writePython3 and friends](#) #89759
- ☐ [Improve documentation around building python packages with poetry](#) #78442
- ☐ [PYTHONPATH is not available in environment when using python.buildEnv or python.withPackages](#) #61144
- ☐ [Unable to override "disabled" attribute with overridePythonAttrs](#) #48663
- ☐ [Guidance for packaging packages providing Python bindings](#) #40835
- ☐ [Python: wrapPythonPrograms should not add \(propagated\) build inputs to wrappers](#) #24128 (comment)
- ☐ [Python: wrapped python packages don't propagate their PYTHONPATH](#) #23676
- ☐ [Python packages installing absolute paths to \\$out/lib/python-xxx/site-packages](#) #23438 (comment)
- ☐ [python: can not use imperatively installed packages as libraries](#) #10597
- ☐ [Python ecosystem improvements \(placeholder issue\)](#) #1819

## Issues created as part of Documentation Project 2023 efforts

- ☐ [nixpkgs manual: Python chapter structural problems](#) #246234
- ☐ [Documentation: Update bespoke markdown syntax reference](#) #246357
- ☐ [nixpkgs manual: python section: interface reference coverage](#) #250376
- ☐ [nixpkgs manual: python section: comments on interpreters section](#) #251959
- ☐ [nixpkgs manual: python section: comments on attributes section](#) #251962
- ☐ [nixpkgs manual: python section: comments on building packages section](#) #251963

- ☒ executable
  - ☒ interpreter
  - ☒ libPrefix
  - ☐ packageOverrides (in FAQ)
  - ☒ \${python}Packages
  - ☒ pkgs (in FAQ, user guide)
    - ☒ buildPythonApplication
    - ☒ buildPythonPackage (in FAQ, user guide)
      - ☐ buildInputs (in FAQ, user guide)
        - ☐ postShellHook (in FAQ)
        - ☐ postVenvCreation (in FAQ)
      - ☐ CFLAGS (in user guide)
    - ☒ buildPhase
    - ☒ catchConflicts
    - ☐ checkPhase (in FAQ, in user guide)
    - ☒ disabled
    - ☐ disabledTests (in user guide)
    - ☐ disabledTestPaths (in user guide)
    - ☒ doCheck (in user guide)
    - ☒ dontWrapPythonPrograms
    - ☒ format
    - ☒ installCheck
    - ☒ installPhase
    - ☐ LDFLAGS (in user guide)
    - ☒ makeWrappersArgs
    - ☒ namePrefix
  - ☒ postFixup
    - ☐ preInstall (in FAQ)
    - ☒ preShellHook
    - ☒ propagatedBuildInputs (in user guide)
    - ☐ pytestFlagsArray (in user guide)
    - ☐ pythonImportsCheck (in user guide)
    - ☒ pythonPath
      - ☐ pythonRelaxDeps (in user guide)
      - ☐ pythonRemoveDeps (in user guide)
    - ☒ removeBinByteCode
    - ☒ setupPyBuildFlags
    - ☒ setupPyGlobalFlags
    - ☒ shellPhase
      - ☐ sphinxBuilders (in user guide)
      - ☐ sphinxRoot (in user guide)
      - ☐ src (custom logic for development mode, in user guide)
      - ☐ unittestFlagsArray (in user guide)
  - ☒ override
    - ☐ enableOptimizations (in FAQ)
    - ☐ reproducibleBuild (in FAQ)
  - ☐ python (in FAQ)
  - ☒ toPythonApplication
  - ☒ toPythonModule
  - ☐ \${pypkg}
    - ☐ overridePythonAttrs (in FAQ)
- setupHooks

# Open Collective fundraiser

[Solutions](#) ▾[Product](#) ▾[Company](#) ▾[Help & Support](#)[Search](#)[Sign In](#)

## Documentation Project

PROJECT



Part of: [The NixOS Foundation](#) Fiscal Host: Stichting NixOS Foundation

Design a learning journey ranging from the first encounter with Nix to mastering the skills needed to leverage common use cases.

[BUDGET](#)[CONTRIBUTE](#)[ABOUT](#)[CONTRIBUTE MONEY](#)[ACTIONS](#) ▾

# Open Collective fundraiser



## Budget ⓘ

Transparent and open finances.

All

Expenses

Transactions

€ TODAY'S  
BALANCE

**€12,300.53**  
EUR

~ TOTAL  
RAISED

**€18,363.04**  
EUR

💸 TOTAL  
DISBURSED

**€6,062.51**  
EUR

📅 ESTIMATED  
ANNUAL  
BUDGET

**€18,962.50**  
EUR

## Documentation Project is all of us

### Our contributors 57

Thank you for supporting Documentation Project.



flox  
€5,000 EUR

AM

Amjad Masad  
€5,000 EUR

GF

Gerd Flaig  
€500 EUR

CC

Contria Connect  
€250 EUR

# Our Stewards



**rWEAG**











**antithesis**

**flox**

# Grant applications



  nix-community / projects

 |     

[Code](#) [Issues 1](#) [Pull requests 1](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [...](#)

 **projects** Public

 Sponsor

 Edit Pins ▾

 Watch 2 ▾

 Fork 1 ▾

 Star 4 ▾

☰ README.md 

## Project proposals for the Nix ecosystem

The Nix ecosystem is developed by many volunteers and a few paid developers. Among many other things, they maintain one of the largest open source software distributions in the world. Keeping it working and up to date

### Releases

No releases published  
[Create a new release](#)

### Sponsor this project

 [opencollective.com/nix-comm...](https://opencollective.com/nix-community)

# Stay up to date



## This Month in Nix Docs - #4 - July/August 2023

■ Announcements



zmitchell

5d

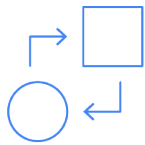
### Community

The Documentation Team can be found in a number of places. Check the [Documentation Team page](#) 9 and drop in if you'd like to contribute!

The Learning Journey Working Group won't have meetings in the month of September (or at least I/ @zmitchell won't be leading them). Next week is NixCon, and several members of the Documentation Team will be there either attending and/or speaking (including me/ @zmitchell , @fricklerhandwerk , and @infinisil )!



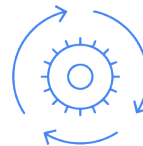
Assessment



Initial set-up &  
migration



Training



Tuning &  
maintenance



Upstreaming



Come meet our team! [Tweag is a Gold sponsor.](#)

Learn more about Tweag and our extended capabilities in the Modus Create platform.



Check out **The Nix Hour** on YouTube.

